

```

      TITLE      'ACQCONST-B00,08/21/73,DWG702985'
      SYSTEM     SIG7F
      CSECT      1
      PCC        0
ACQCONS@ RES     0      CONTROL CARDS NOT PRINTED.
                        ORIGIN OF CONSTANT ACQ. MODULE.
*
*   REF'S   AND   DEF'S
*
      DEF       ACQCONS@      = START OF ACQCONST MODULE.
      DEF       ACQCONST      ACQUIRE CONSTANT -- ENTRY PT.
SPACE 3
*   REF       ACQCC           REFS TO PROCEDURE:
      REF       ACQCODE       REFS TO PROCEDURE:
                                ACQ. CURRENT CHAR AND ITS CODE.
                                ACQ. CURRENT CHAR'S CODE.
SPACE 2
*   REF       CONSTDT         REFS TO CONTEXT:
      REF       CONSTDTX      DBLWD TEMP.
      REF       CONSTAD       EXTRA DBLWD TEMP.
      REF       CONSTBUF      DBLWD TEMP -- 1ST WD IS ALWAYS ZERO.
      REF       CONSTTYP      BUFFER TO HOLD CONSTS (FOR VECTORS).
                                TYPE-OF-CONSTANT INDICATOR:
*                                0 OR 1 = LOGICAL.
*                                > 1 = INTEGER.
*                                - 1 = REAL.
      REF       NSPILLED      TEMP FOR NO.OF DIGITS SPILLED.
*   REF       ZEROZERO       REFS TO CONSTANTS
      REF       F0F9          0,0
      REF       X4E1          '0','9'
      REF       BITPOS        X'4E100000', 0
                                32-WD BIT-TBL (BITPOS-K CONTAINS A
                                WD HAVING A 1 ONLY IN BIT POS. K).
*
*   EQU'S RELATED TO CONTEXT
*
CONSTTMP EQU      NSPILLED      TEMP WD
SPACE
*
*   STANDARD EQU'S
*
REGISTERS
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
*   OTHER EQU'S
*
NEGSIGN EQU      X'72'        INTERNAL (EBCDIC) NEGATIVE SIGN.
FNEG    EQU      2          FLAGS NEGATIVE REAL NOS.
ENEG    EQU      1          FLAGS REAL NOS. WITH NEG. EXPONENTS.
*
*   DOUBLEWORD CONSTANTS
*
      BOUND      8
PTORE    DATA    '.,','E'
PTORNEG  DATA    '.,',NEGSIGN
M100P100 DATA    -100,100
1TENTH   DATA    X'0CCCCCCC',X'CCCCCCE' ANY NO. LESS THAN THIS CAN BE
*                                MULTIPLIED BY 10 WITHOUT OVERFLOW.
SPACE
X4E      DATA    X'4E000000',0
*
*   WORD CONSTANTS
*
X488     DATA    X'48800000'
EXPON48  DATA    X'48000000'
PAGE
*

```

```

* POWERS OF TEN
*
* THE POWERS OF TEN USED FOR SCALING ARE DEFINED IN TABLES BELOW.
* EACH POWER HAS THREE NUMBERS ASSOCIATED: (P) POWER (SEE TENP),
* (F) FRACTION (SEE TENF), AND (E) BINARY EXPONENT (SEE TENE).
* THEY SATISFY:
*          1 <= P <= 13
*          10**P = F*(2**E)
*          .5 <= F < 1.0
*
* CURRENTLY USED P-VALUES ARE: 1, 4, AND 13.
*
TENP    EQU      %-1
        DATA    1,4,13
NTENS   EQU      %-TENP-1
*
TENE    EQU      %-1
        DATA    4,14,44
*
TENF    EQU      %-1
        DATA    X'50000000' = 10      * 2**(-4)
        DATA    X'4E200000' = 10**4   * 2**(-14)
        DATA    X'48C27395' = 10**13  * 2**(-44)
*
PAGE
*****
*
* ACQCONST -- SUBROUTINE TO ACQUIRE CONSTANTS (VECTOR OR SCALAR) THAT
* ARE IN INTERNAL FORM, PRODUCING THE REAL OR INTEGER EQUIVS.
* IN THE CONSTANT BUFFER -- CONSTBUF. THE TYPE OF THE CONSTANT
* IS PRODUCED IN -- CONSTTYP:
*
*          0 OR 1  INDICATES LOGICAL DOMAIN,
*          > 1  INDICATES INTEGER DOMAIN,
*          AND -1  INDICATES REAL DOMAIN.
*
* ACQCONST IS ENTERED WHEN (IN REASONABLE CONTEXT) A DIGIT,
* DECIMAL POINT, OR NEGATIVE SIGN IS PICKED UP. IT IS POSSIBLE
* THAT, DUE TO SYNTAX ERROR, ACQCONST WILL NOT ENCOUNTER A
* NUMBER; HOWEVER, ACQCONST DOES NOT GENERATE ANY ERROR DIAGS.
* IN FACT, THE ONLY ERROR IT RECOGNIZES IS AN OVERFLOW -- WHICH
* RESULTS IN A UNIQUE EXIT. UNDERFLOW PRODUCES A ZERO (IN THE
* APPROPRIATE DOMAIN).
*
* ACQCONST ATTEMPTS TO OPTIMIZE THE HANDLING OF INTEGER DOMAIN
* CONSTANTS BY UTILIZING SOMEWHAT REDUNDANT PROCEDURE. IT ALSO
* ATTEMPTS TO STAY IN INTEGER DOMAIN AS LONG AS POSSIBLE, FOR
* EXAMPLE ' 1.23E5 ' WILL BE TREATED AS AN INTEGER UNLESS THE
* CONSTANT ALREADY HAS CONTAINED PRIOR REAL NUMBERS.
*
* REGS:  R4 -- LINK; EXIT (EFFECTIVELY) IS VIA
*          0,R4 IF OVERFLOW OR
*          1,R4 IF NO OVERFLOW (ACTUALLY, THE
*          NORMAL EXIT IS MADE BY LINKING
*          THRU 'ACQCC' OR 'ACQCODE' IN
*          ORDER TO INSURE THAT THE CHAR
*          AND CODE (NON-BLANK) THAT
*          TERMINATES THE CONSTANT IS
*          READY FOR FURTHER CODESTRING
*          WORK.
*
*          R1 -- (ENTRY) PTS AT DIGIT, DEC.PT, OR NEG.SIGN.
*                (EXIT) PTS AT TERMINATION CHAR (POSSIBLY SAME
*                AS AT ENTRY IF SYNTAX ERROR).
*
*          R2 -- (ENTRY) CONTAINS DIGIT, DEC.PT, OR NEG.SIGN.
*                (EXIT) CONTAINS TERMINATION CHAR (NON-BLANK).
*
*          R3 -- (EXIT) CONTAINS CODE FOR TERMINATION CHAR.
*
*          R6 -- (EXIT) CONTAINS THE NO.OF NUMBERS IN THE
*                CONSTANT (ZERO IF SYNTAX ERROR).
*
*          R5 THRU R14 ARE VOLATILE.
*
ACQCONST AI,R4    1      ASSUME NO-OVERFLOW EXIT.
          LI,R6    0      R6 = NO.OF NOS. IN CONSTANT.
          STW,R6   CONSTTYP TYPE 0 -- START AS LOGICAL CONSTANT.
          BAL,R5   NSET    DO SET-UPS FOR NUMBER.
ISET      LI,R9    0      R9 = INTEGER ACCUMULATOR.
          AI,R2    -'0'    DOES MAGNITUDE START WITH DIGIT...
          BGEZ     IDIG1   YES, 1ST DIGIT.
IPT        LI,R5   RPT     NO, DEC.PT.
IE         LI,R8   0
          B        TRYREAL

```

IOK	AI, R2	- '0'		STRIP OFF HI-ORDER DIGIT OF EBCDIC.
IDIG1	STW, R2	CONSTAD+1		(INDICATES PRESENCE OF A DIGIT).
	AW, R9	CONSTAD+1		ACCUMULATE IT.
	BNOV	INXT		
	LI, R5	PNXT		OH-OH, INTEGER REACHES R9'S SIGN-BIT
	B	TOOBIG		
IDIG	MI, R8	10		PREPARE FOR NEXT ACCUMULATE.
	BNOV	IOK		
	LI, R5	PM		OH-OH, INTEGER TOO BIG.
TOOBIG	LI, R11	0		PREPARE TO ENTER THE 'PACK' ROUTINE,
	STW, R11	NSPILLED		
	LI, R14	RPTQ		LIKE STARTING A REAL NO.
TRYREAL	SLD, R8	1		REAL NO. IS SHIFTED 1.
	LI, R3	0		R3 = DECIMAL EXPONENT.
	B	0, R5		
INXT	AI, R1	1		GET NEXT CHAR.
	LB, R2	0, R1		
	CLM, R2	F0F9		CK FOR DIGIT.
	BCR, 9	IDIG		YES.
	CLM, R2	PTORE		NO, CK FOR DEC.PT OR AN E.
	BCS, 9	IZ		NO -- END INTEGER.
	BCR, 2	IPT		DEC.PT.
	BCS, 4	IZ		NO -- END INTEGER.
	LI, R5	RE		E
	B	IE		
RICK	LW, R7	CONSTTYP		ARE WE IN INTEGER DOMAIN, SO FAR...
	BLZ	RZ		NO -- REAL.
	CW, R8	X488		YES, IS REAL VALUE TO BIG FOR INTG
	BGE	REALSW		YEP, SWITCH TO REAL DOMAIN.
	STD, R8	CONSTDTX		NOPE, SAVE OUR REAL VALUE.
	FAL, R8	X4E		CK FOR LOSS OF FRACTION.
	CD, R8	CONSTDTX		
	BNE	REALSW8		YES, SWITCH TO REAL DOMAIN.
	FAL, R8	X4E1		NO, GET INTEGER VALUE IN R9.
IZ	STS, R9	CONSTTYP		(THIS GIVES INTG VS LOGL INDICATION)
	AI, R10	-FNEG		CK FOR NEGATIVE NO.
	BLZ	ISTORE		NO, POSITIVE.
	LCW, R9	R9		YES, NEGATE.
	STS, R1	CONSTTYP		INSURE TYPE=INTEGER, NOT LOGICAL.
ISTORE	STW, R9	CONSTBUF, R6		STORE THE INTEGER.
	LI, R5	ISET		EXIT 'FINQ' TO ISET IF NO. SEEMS UP.
FINQ	XW, R1	CONSTAD+1		SAVE R1, TEST FOR NO DIGITS ACCUM.
	CI, R1	18		(R1<19 MEANS DIGIT OCCURRED).
	BG	ACQCC	-EXIT-	NO, R1 PTS AT FALSE-NO.-START; REACQ
	LW, R1	CONSTAD+1		OK, RESTORE R1.
	AI, R6	1		WE HAVE 1 MORE NO. IN THE CONSTANT.
	CI, R2	' '		DID WE END ON A BLANK...
	BE	BLNSKIP		YES, SKIP IT.
	CLM, R2	PTORNEG		NO, CK FOR DEC.PT OR NEG.SIGN...
	BCS, 9	NSET		NOPE, OK.
	BCR, 2	DBLNUM		DEC.PT (UNEXPECTED).
	BCS, 4	NSET		NOPE, OK.
*				NEG.SIGN (UNEXPECTED).
DBLNUM	LI, R6	0		SYNTAX ERROR INDICATION.
	B	ACQCODE	-EXIT-	ACQ CHAR'S CODE.
BLNKQ	CI, R2	' '		SKIP BLANKS.
	BNE	NSET		
BLNSKIP	AI, R1	1		
	LB, R2	0, R1		
	B	BLNKQ		
NSET	STW, R1	CONSTAD+1	(>19)	SAVE CURR.CHAR LOC IN CASE FALSE-ST.
	LI, R10	0		CLEAR NEG-FLAGS FOR NO. AND EXPON.
	CLM, R2	F0F9		CK FOR DIGIT.
	BCR, 9	0, R5		YES, WORK THAT NO.
	CLM, R2	PTORNEG		NO, CK FOR DEC.PT OR NEG.SIGN.
	BCS, 9	ACQCODE	-EXIT-	NEITHER; ACQ CHAR'S CODE.
	BCR, 2	0, R5		DEC.PT (COULD BE FALSE-START).
	BCS, 4	ACQCODE	-EXIT-	NEITHER.
	LI, R10	FNEG		NEG-SIGN -- SET NEG-FLAG FOR NO. --
	AI, R1	1		(COULD BE FALSE-START).
	LB, R2	0, R1		GET NEXT CHAR.
	CLM, R2	F0F9		CK FOR DIGIT AFTER NEG-SIGN...
	BCR, 9	0, R5		YES, WORK THAT NO.
	CI, R2	' '		NO, CK FOR DEC.PT...
	BE	0, R5		YEP (COULD BE FALSE-START).
	BDR, R1	ACQCC	-EXIT-	REACQ NEG-SIGN FOR FALSE-START.
	B	0, R5		
REALSW8	LD, R8	CONSTDTX		RESTORE REAL VALUE.

REALSW	LW, R7	R6	GET NO.OF EARLIER INTEGERS...
	BEZ	REALD	NONE, NO NEED TO CONVERT.
REALC	LW, R12	CONSTBUF-1, R7	SOME; IN REVERSE ORDER, CONVERT
	LI, R13	0	OLD INTEGERS TO REALS.
	SAD, R12	-8	
	EOR, R12	EXPON48	
	FAL, R12	ZEROZERO	
	STD, R12	CONSTBUF-2, R7	
	BDR, R7	REALC	
REALD	LI, R7	-1	SET REAL DOMAIN INDICATION.
	STW, R7	CONSTTYP	
RZ	AI, R10	-FNEG	CK FOR NEGATIVE NO.
	BLZ	RSTORE	NO, POSITIVE.
	LCD, R8	R8	YES, NEGATE.
RSTORE	STD, R8	CONSTBUF, R6	STORE THE REAL NO.
	BAL, R5	FINQ	MAKE FINALIZATION QUERIES.
	LD, R8	ZEROZERO	CONTINUE -- CLEAR REAL ACCUMULATOR
	LI, R14	RPTQ	EXIT FROM 'PACK' TO 'RPTQ'.
PACK	LI, R11	0	R11 = NO.OF DIGITS PACKED.
	STW, R11	NSPILLED	(CLEAR NO.OF DIGITS SPILLED, TOO).
PDQ	CLM, R2	F0F9	DIGIT QUERY...
	BCR, 9	PDIG	YES, DIGIT.
	B	*R14	NO, EXIT FROM PACK.
PDIG	CD, R8	1TENTH	SEE IF ACCUM. WOULD OVERFLOW...
	BL	POK	NO.
	MTW, 1	NSPILLED	YES, COUNT THAT DIGIT POSITION,
	B	PNXT	BUT DON'T ACCUMULATE ANY MORE.
POK	AI, R11	1	COUNT THE DIGIT TO BE PACKED.
	STD, R8	CONSTDT	MULTIPLY CURRENT ACCUMULATION
	SLD, R8	2	BY 10 (IT WAS GUARANTEED THAT
	AD, R8	CONSTDT	THIS WILL REACH NO FURTHER THAN
	SLD, R8	1	BIT POS. 1 OF R8).
PM	AI, R2	- '0'	(POST-MULT) STRIP EBCDIC CODE OFF.
	SLS, R2	1	NOTE--INSURE ACCUM. BIT POS. 63 NOT USED;
*			THUS PACK PRODUCES A 62-BIT NO.,
*			AND BITS 0 AND 63 ARE ZERO. (THE
*			REASON IS TO ALLOW ACCUM TO BE
*			SPLIT INTO 2 31-BIT WDS LATER;
*			OTHERWISE, SIGNS COULD CAUSE
*			CONFUSION).
	STW, R2	CONSTAD+1	(CONSTAD+0 IS ALWAYS ZERO).
	AD, R8	CONSTAD	ACCUMULATE.
PNXT	AI, R1	1	
	LB, R2	0, R1	GET NEXT CHAR.
	B	PDQ	
RPTQ	LW, R3	NSPILLED	R3 = DECIMAL EXPONENT (DE).
	CI, R2	'.'	DECIMAL PT. QUERY...
	BNE	REQ	NO, TRY FOR E.
RPT	AI, R1	1	GET CHAR AFTER DEC PT.
	LB, R2	0, R1	
	BAL, R14	PACK	ACCUMULATE FRACTION DIGITS, IF ANY.
	SW, R3	R11	DECR DE BY THAT MANY DIGITS.
REQ	CI, R2	'E'	E QUERY...
	BNE	REAL	NO, PROCESS REAL NO.
RE	XW, R1	CONSTAD+1	SAVE R1, TEST FOR NO DIGITS YET
	CI, R1	18	(R1 < 19 MEANS DIGIT OCCURRED)...
	BG	ACQCC	-EXIT- NO, R1 PTS AT FALSE-START, REACQ.
	XW, R1	CONSTAD+1	OK, RE-EXCHANGE.
	AI, R1	1	GET NEXT CHAR.
	LB, R2	0, R1	
	CLM, R2	F0F9	DIGIT...
	BCR, 9	RED	YES, E FOLLOWED BY DIGIT.
	CI, R2	NEGSIGN	NO, NEGATIVE SIGN...
	BE	RENQ	YES.
	LI, R2	'E'	NO, RESTORE THE E, PT TO IT, AND
	BDR, R1	REAL	PROCESS REAL NO.
RENQ	AI, R10	ENEG	SET EXPONENT-NEGATIVE FLAG.
	AI, R1	1	GET NEXT CHAR.
	LB, R2	0, R1	
	CLM, R2	F0F9	DIGIT...
	BCR, 9	RED	YES.
	LI, R2	NEGSIGN	NO, RESTORE THE NEG.SIGN, BACK UP
	BDR, R1	REAL	
RED	STD, R8	CONSTDTX	SAVE ACCUM. 'FRACTION'
	LD, R8	ZEROZERO	CLEAR TO ACCUM. EXPONENT.
	BAL, R14	PACK	ACCUM. IT.
	OR, R8	NSPILLED	CK FOR RIDICULOUSLY BIG EXPONENT...
	BEZ	REOK	NO, OK SO FAR

	CI, R10	ENEG	YES, CK 0'FLO VS U'FLO.
	BANZ	UFLO	
	B	-1, R4	-EXIT- OVERFLOW
*			
REOK	SLS, R9	-1	ACCUM. EXP. WAS SHIFTED 1.
	CI, R10	ENEG	CK EXPONENT SIGN...
	BAZ	RPE	POSITIVE.
	SW, R3	R9	NEGATIVE, DECR. DE.
	B	ROUT	
RPE	AW, R3	R9	INCR. DE.
ROUT	BNOV	RG0	GO UNLESS RESULT RIDICULOUSLY BIG...
	BLZ	-1, R4	-EXIT- OVERFLOW (SIGN REVERSAL).
UFLO	LD, R8	ZEROZERO	UNDERFLOW -- ASSUME ZERO REAL NO.
	B	RICK	CK FOR REALLY INTEGER
RG0	LD, R8	CONSTDTX	RESTORE ACCUM. FRACTION.
REAL	CD, R8	ZEROZERO	TEST FOR ZERO...
	BEZ	RICK	YES, CK FOR REALLY INTEGER.
	CLM, R3	M100P100	NO, IS DE REASONABLE...
	BCR, 9	BINORM	OK -- DO BINARY NORMALIZATION.
HIORLO	BG	-1, R4	-EXIT- OVERFLOW
	LD, R8	ZEROZERO	UNDERFLOW -- ASSUME ZERO REAL NO.
	B	RICK	CK FOR REALLY INTEGER.
BINORM	AI, R8	0	IF FRACTION'S HI-HALF IS 0,
	BEZ	RNORM	USE RIGHT-HAND NORMALIZATION.
	LI, R5	63	R5= BIN.EXP (BE) -- PRESET & BIASED.
LLOOP	SLD, R8	1	LEFT SHIFT,
	AI, R5	-1	DECR. BE,
	AI, R8	0	TEST FRACTION'S SIGN...
	BGZ	LLOOP	+ -- SHIFT AGAIN.
	B	CLRBIT0	- -- CLEAR SIGN POS. AND GO.
RNORM	LI, R5	-1	R5= BIN.EXP (BE) -- PRESET & BIASED.
RLOOP	SCD, R8	-1	ROTATE RIGHT,
	AI, R5	1	INCR. BE,
	AI, R9	0	TEST LO-HALF EMPTIED OF 1'S...
	BNEZ	RLOOP	NO, RE-CYCLE.
CLRBIT0	SLD, R8	-1	MOVE OVER TO CLEAR SIGN BIT.
	LI, R7	NTENS	R7 = 'TEN-POWER' NO.
	AI, R3	0	TEST DECIMAL EXPONENT (DE)...
	BLEZ	DENORZ	NEGATIVE OR ZERO.
	LI, R14	0	POSITIVE, CLEAR TOP WORD OF THE
	STW, R14	CONSTDT	DBLWD TEMP.
DEPOS	SW, R3	TENP, R7	TRY SCALING DOWN DE...
	BLZ	DEPR	TOO FAR, RECOVER IT.
	STW, R8	CONSTTMP	OK, SAVE HI HALF OF FRACTION.
	SLS, R9	-1	GET MAGNITUDE OF LO HALF.
	MW, R8	TENF, R7	MULT BY A PURE POWER OF TEN.
	SLD, R8	1	PREPARE TO ADD IN THE LO HALF'S
	STW, R8	CONSTDT+1	PRODUCT (CONSTDT+0 IS ZERO).
	LW, R9	CONSTTMP	RECOVER HI HALF FOR ITS
	MW, R8	TENF, R7	MULTIPLICATION.
	AD, R8	CONSTDT	ADD IN THE LO HALF'S PRODUCT.
	SLD, R8	1	
	CW, R8	BITPOS-1	POST-NORMALIZE.
	BANZ	DEPBE	
	SLD, R8	1	
	AI, R5	-1	
DEPBE	AW, R5	TENE, R7	ADJ BINARY EXPONENT.
	B	DEPOS	
DEPR	AW, R3	TENP, R7	RECOVER DE.
	BEZ	DEZERO	OK, IT IS ZEROED.
	BDR, R7	DEPOS	TRY ANOTHER PURE POWER OF TEN.
*	NEVER	FALLS THRU.	
DENORZ	BEZ	DEZERO	OK, DE IS ZEROED.
DENEG	AW, R3	TENP, R7	TRY SCALING UP DE...
	BGZ	DENR	TOO FAR, RECOVER IT.
	CW, R8	TENF, R7	POSITION DIVIDEND TO GET NORM. QUOT.
	BL	DENQ	
	SLD, R8	-1	
	AI, R5	1	
DENQ	SLD, R8	-1	
	DW, R8	TENF, R7	HI-ORDER QUOTIENT.
	STW, R9	CONSTTMP	SAVE IT.
	LW, R9	TENF, R7	NEW DIVIDEND = HI-ORDER REMAINDER
	SLD, R8	-1	+ HALF OF DIVISOR (FOR ROUNDING).
	DW, R8	TENF, R7	LO-ORDER QUOTIENT.
	SLS, R9	1	PACK IT TOGETHER
	LW, R8	CONSTTMP	WITH THE HI-ORDER QUOTIENT.
	SW, R5	TENE, R7	ADJ BINARY EXPONENT.

DENR	B	DENEG	RECOVER DE.
	SW, R3	TENP, R7	OK, IT IS ZEROED.
	BEZ	DEZERO	TRY ANOTHER PURE POWER OF TEN.
*	BDR, R7	DENEG	
	NEVER	FALLS THRU.	
DEZERO	AI, R5	1+4*64	BIAS BINARY EXPON (APPX HEX EXPON).
	LI, R7	3	CK ITS 2 LOW BITS.
	AND, R7	R5	CALC SHIFT NECESSARY TO NORMALIZE
	CI, R7	1	HEXADECIMALLY.
	BLE	NSHIFT	
	AI, R7	-4	
NSHIFT	SLD, R8	-8, R7	POSITION FRACTION.
	SW, R5	R7	CORRECT BINARY EXPONENT.
	SAS, R5	-2	CHANGE TO (EXCESS-64) HEX EXPONENT.
	STB, R5	R8	SET CHARACTERISTIC OF REAL NO.
	CI, R5	X'FFF80'	WAS IT IN RANGE (0 - 127)...
	BAZ	RICK	YES -- CK FOR REALLY INTEGER.
	AI, R5	0	NO, TEST IT FOR AN
	B	HIORLO	0'FLO VS U'FLO DETERMINATION.

PAGE

SPACE 2

Z SET %-ACQCONS@ SIZE OF ACQCONST IN HEX.

SPACE

Z SET Z+Z/10*6+Z/100*96+Z/1000*1536 SIZE IN DECIMAL.

SPACE 2

END

```

        TITLE      'CONSTS-B00,10/16/73,DWG702985'
        SYSTEM     SIG7F
        CSECT      1
        PCC        0
CONSTS@ RES      0          CONTROL CARDS NOT PRINTED.
                                ORIGIN OF CONSTANTS MODULE.
*
*   DEF 'S
*
        DEF      CONSTS@          = START OF CONSTS MODULE.
        DEF      APLSINON        APL'S SIGN-ON MESSAGE.
SPACE
        DEF      BITMASK          32-WD TBL OF 1-BIT-SETTINGS.  THIS
*                                BLOCK IS IN ASCENDING ORDER--E.G.
*                                BITMASK+1 = 1 AND
*                                BITMASK+32 = SIGN-BIT ONLY.
        DEF      BITPOS          32-WD TBL OF 1-BIT-SETTINGS.  TO GET
*                                A WD HAVING ONLY BIT POSITION K
*                                (SET TO 1), REFER TO 'BITPOS-K'.
*
        DEF      XFFFF            = X'FFFF' (RIGHT HALFWORD).
        DEF      X1FFFF            = X'1FFFF' (ADDRESS).
        DEF      XFF0000          = X'FF0000' (BYTE 1 ONLY).
        DEF      X4E1            = X'4E100000', 0 (FOR FL.PT CVS).
        DEF      ZEROZERO        = 0,0
        DEF      F0F9            = '0','9' (EBCDIC DIGIT RANGE).
        DEF      BLANKS          WD CONTAINING ALL BLANKS.
SPACE
SPACE
        DEF      NONAME          NON-NAME START & NON-DIGIT CODES.
SPACE
        DEF      FUNTYPES        RANGE OF FUNCTION TYPES-- 8 & 13.
        DEF      HEXTBL          EBCDIC HEX DIGITS.
SPACE
*   ERROR I.D. DEFS:
        DEF      IDIOERR
        DEF      IDWSFULL
        DEF      IDXEQERR
        DEF      IDBADCH
        DEF      IDTRUNC
        DEF      IDOPENQT
        DEF      IDSYMFUL
        DEF      IDLSCAN
        DEF      IDDEFN
        DEF      IDSIDAM
        DEF      IDBADCMD
        DEF      IDNOTGRP
        DEF      ID2BIGLD
        DEF      IDFILREF
        DEF      IDFILBSY
        DEF      IDWSNOTF
        DEF      IDNOTSAV
        DEF      IDNERASE
        DEF      IDNOTCPY
        DEF      IDNOTFND
        DEF      IDTOOBIG
        DEF      ID2NAMEY
        DEF      IDFILSPC
        DEF      IDTERMAL
        DEF      IDFILNAM
        DEF      IDNOTAPL
        DEF      IDFTFULL
        DEF      IDFILACC
        DEF      IDFILTIE
        DEF      IDNOPACK
        DEF      IDFILIDX
        DEF      IDFILDAM
        DEF      IDFIOERR
PAGE
*
*   EQU REGION -- ERROR I.D. TABLE.
*
*   THIS IS A SEQUENTIAL TABLE OF INTERNAL ERROR IDENTIFIERS.
*   (THE ERROR I.D. IS NOT THE SAME AS THE 'ERROR NUMBER' WHICH IS
*   USED FOR ERROR-CONTROL.)
*   NEW ERROR I.D.'S CAN BE ADDED AT THE END OF THIS TABLE; SEE
*   ALSO THE ERROR MODULE, WHERE ERROR I.D.'S ARE USED TO ASSIGN
*   ERROR NUMBERS AND MESSAGE STRINGS.
*

```

```

*   ERROR I.D.'S   -1 THRU 12 ARE ASSIGNED TO EXECUTION ERRORS:
*
*   -1             BREAK
*   0             LINESCAN ERR (SYNTAX ERR MSG USED)
*   1             SYNTAX ERR
*   2             UNDEFINED
*   3             NO RESULT
IDIOERR EQU      4             I/O ERR
*   5             DOMAIN ERR
*   6             RANK ERR
*   7             LENGTH ERR
IDWSFULL EQU     8             WS FULL
*   9             SINGULAR MATRIX
*  10             FORMAT SYNTAX ERR
*  11             INDEX ERR
IDXEQERR EQU    12             CMD OR FUNDEF ERR BY EXECUTE-OPERATR
IDBADCH EQU     13             BAD CHAR
IDTRUNC EQU     14             TRUNCATED INPUT
IDOPENQT EQU    15             OPEN QUOTE
IDSYMFUL EQU    16             SYM TBL FULL
IDLSCAN EQU     17             LINESCAN ERR
IDDEFN EQU      18             DEFN ERR
IDSIDAM EQU     19             SI DAMAGE
IDBADCMD EQU    20             BAD COMMAND
IDNOTGRP EQU    21             NOT GROUPED
ID2BIGLD EQU    22             TOO BIG TO LOAD
IDFILREF EQU    23             BAD FILE REF
IDFILBSY EQU    24             FILE IN USE
IDWSNOTF EQU    25             WS NOT FOUND
IDNOTSAV EQU    26             NOT SAVED, THIS WS IS ...
IDNERASE EQU    27             ... NOT ERASED
IDNOTCPY EQU    28             ... NOT COPIED
IDNOTFND EQU    29             ... NOT FOUND
IDT00BIG EQU    30             TOO BIG
ID2NAMEY EQU    31             TOO MANY SYMBOLS
IDFILSPC EQU    32             FILE SPACE TOO LOW
IDTERMAL EQU    33             WRONG TERMINAL
IDFILNAM EQU    34             FILE NAME ERR
IDNOTAPL EQU    35             NOT APL FILE
IDFTFULL EQU    36             FILE TBL FULL
IDFILACC EQU    37             FILE ACCESS ERR
IDFILTIE EQU    38             FILE TIE ERR
IDNOPACK EQU    39             PRIVATE PACK UNAVAIL, CALL OPR.
IDFILIDX EQU    40             FILE INDEX ERR
IDFILDAM EQU    41             FILE DAMAGE
IDFIOERR EQU    42             FILE I/O ERR ....
PAGE
*
*   DOUBLEWORD CONSTANTS
*
      BOUND      8
SPACE 2
ZEROZERO DATA  0,0
SPACE
F0F9 DATA      '0','9'
SPACE
X4E1 DATA      X'4E100000',0
SPACE
NONAME DATA     10,138      USED TO FIND NON-DIGIT, NON-NAME
*                           CHARS IN KEY CODESTRING DESIGNS.
SPACE
FUNTPES DATA    8,13      FUNCTION TYPES (NILADIC W/ NO-RESULT
*                           THRU DYADIC WITH A RESULT).
SPACE 3
*
*   SINGLE WORD CONSTANTS
*
BITMASK DATA    X'80000000' @
DATA            X'00000001' @
DATA            X'00000002' @
DATA            X'00000004' @
DATA            X'00000008' @
DATA            X'00000010' @
DATA            X'00000020' @
DATA            X'00000040' @
DATA            X'00000080' @
DATA            X'00000100' @
DATA            X'00000200' @

```


	DATA	X'00000400'	@
	DATA	X'00000800'	@
	DATA	X'00001000'	@
	DATA	X'00002000'	@
	DATA	X'00004000'	@
	DATA	X'00008000'	@
	DATA	X'00010000'	@
	DATA	X'00020000'	@
	DATA	X'00040000'	@
	DATA	X'00080000'	@
	DATA	X'00100000'	@
	DATA	X'00200000'	@
	DATA	X'00400000'	@
	DATA	X'00800000'	@
	DATA	X'01000000'	@
	DATA	X'02000000'	@
	DATA	X'04000000'	@
	DATA	X'08000000'	@
	DATA	X'10000000'	@
	DATA	X'20000000'	@
	DATA	X'40000000'	@
BITPOS	DATA	X'80000000'	@
*			
XFFFF	DATA	X'FFFF'	
X1FFFF	DATA	X'1FFFF'	
XFF0000	DATA	X'FF0000'	
BLANKS	TEXT	' '	
SPACE 2			
APLSINON	TEXTC	'APL 10/16/73'	
PAGE			
HEXTBL	TEXT	'0123456789ABCDEF'	

SPACE 2			
Z	SET	%-CONSTS@	SIZE OF CONSTS MODULE IN HEX.
SPACE			
Z	SET	Z+Z/10*6+Z/100*96	SIZE IN DECIMAL.
SPACE 2			
END			


```

TITLE      'CONTEXT-B00,09/14/73,DWG702985'
SYSTEM     SIG7F
CSECT      0
PCC         0
CONTEXT@   RES      0      CONTROL CARDS NOT PRINTED.
*                                     ORIGIN OF PROCESSOR CONTEXT MODULE.
*
*   DEF 'S
*
DEF CONTEXT@      =START OF CONTEXT MODULE.
DEF STRAYS        AREA HOLDING STRAY DATA BLK PTRS.
DEF LBLOCK        LOOP CTRL BLK FOR TRICKY OPERATORS.
DEF FDEFPTR       (DB PTR) FUNCTION DESCRIPTOR.
DEF XTEMP         (DB PTR) INDEX OR COORDINATE.
DEF NILCK         (DB PTR)
DEF LFARG         (DB PTR) LEFT ARGUMENT.
DEF RTARG         (DB PTR) RIGHT ARGUMENT.
DEF RESULT        (DB PTR) RESULT.
DEF NSTRAYS       NO.OF STRAY PTRS.
DEF DBROOT        PTR TO ROOT DB (LIST OR FUN.DESCR.).
DEF DBSERIES      PTR INTO A LIST OR FUN. DESCRIPTOR.
DEF GRAFPNTR      (DB PTR) GRAPHICS SPECIAL BLOCK.
DEF CURRCS        PTS AT CURRENT CODESTR. DATA BLK +2.
DEF STRAYBLK      = SIZE OF TOTAL BLOCK OF STRAY PTRS.
DEF OFFSET        OFT USED TO SAVE CURR.CODESTR.OFFSET
DEF WHATERR       HOLDS ERROR INFO WHEN ENTERING EXERR
DEF XSEGBRK       = LOC OF 'OPBREAK' ONLY DURING
*                                     EXECUTION SEGMENT OPERATION.
DEF CXSCRTCH      SCRATCH AREA FOR CODESTRING EXEC.
DEF CONSTTYP      INDICATES TYPE OF ACQUIRED CONSTANT.
DEF NSPILLED      NO.OF DIGITS SPILLED IN ACQ'ING NO.
DEF CONSTAD       DBLWD -- 1ST WD MUST ALWAYS BE ZERO
*                                     2ND WD EITHER CONTAINS PTR TO THE
*                                     BEGINNING OF A NO. OR (AFTER ACQ
*                                     DIGITS WITHIN THAT NO.) A 'DIGIT
*                                     VALUE' (RANGE IS 0 THRU 18).
DEF MAXCOL        MAX. COLUMNS ACCEPTED ON INPUT.
DEF HICOL         INDICATES HI COLUMN IN IMAGE.
DEF ERRCOL        BYTE ADDR.OF ERR COLUMN IN IMAGE.
DEF LINKCS        LINK TO CODESTRINGER.
DEF CONSTKEY      BYTE ADDR.OF 1ST CHAR OF CONSTANT.
DEF OFFSETK       PTR TO CODESTR.DESIG. BEFORE CONST.
DEF IMAGE         INTERNAL CHAR IMAGE OF LINE.
DEF SYMT          SYMBOL TABLE PTR.
DEF NAMEPTR       PTR INTO SYMBOL TABLE (AN OFFSET).
DEF SYMTSIZE      # ENTRIES (DBLWDS) IN SYMBOL TABLE.
DEF NSYMTWDS      # WORDS IN SYMBOL TABLE.
DEF NAMLIMIT      = MAX # CHARS USED IN A NAME.
DEF NAMEWDSZ      = MAX # WORDS NEEDED TO HOLD A NAME.
DEF NAMEWDS       HOLDS # WORDS USED BY A NAME.
DEF LOCNEED       NEW COMMON LOCATION NEEDED.
DEF NEWBOUND      NEW DYNAMIC BOUNDARY NEEDED.
DEF FREETOTL      TOTAL AMT OF FREE DYNAMIC.
DEF FREETBL       FREE TABLE (2-WD ITEMS--LOC & SIZE).
DEF MAXFRENS      MAX # FREE TABLE ENTRIES (DBLWDS).
DEF FBOUNDS       TBL OF FREE BOUNDARIES IN GARB.COLL.
DEF FAQMS         TBL OF FREE ACCUMULATIONS TO FBOUNDS
DEF NR2MOVE       NO.OF REGIONS TO MOVE IN GARB.COLL.
DEF BLKWANTD      SIZE OF NEW DATA BLOCK WANTED.
DEF DREFSAVE      REG SAVE AREA WHILE DE-REFERENCING.
DEF GARBSAVE      REG SAVE AREA FOR GARB.COLL. ET AL.
DEF GCTEMP        TEMP FOR GARBAGE COLLECTOR.
DEF LINKGC        LINK HOLDER FOR GARBAGE COLLECTOR.
DEF LINKWS        LINK HOLDER FOR WORKSPACE MGMTS.
DEF LINKOLAY      LINK HOLDER FOR OVERLAY SWITCHING.
DEF OLDMODE       HOLDS MODE INFO DURING FUN DEFN.      U03-0004
DEF QUIETFLG      = 0 IF SAVED MSG TO BE DISPLAYED.
DEF CATCHTBL      CATCH-COMMAND TABLE.
DEF OBSFLAG       OBSERVE-COMMAND FLAG.
DEF OBSERVE       OBSERVATION SETTING (0=NO & -1=YES)
DEF OBSAVE        REG.SAVE AREA DURING AN OBSERVATION.
DEF BACKUP        COPY SYM TBL RE-SCAN FLAG.
DEF BLANKCNT      # HOLES LEFT IN ACTIVE SYM TBL.
DEF COPYBASE      LOC OF COPY DATA REGION IN COPY WS.
DEF COPYHOME      LOC OF COPY DATA REGION IN ACTIVE WS
DEF COPYSAVE      16 WD SAVE AREA ON DBLWD BOUND.
DEF COPYSIZE      # WDS COPY DYN. OR COPY DATA REGION.
DEF CORVSFIL      = 0 (CORE-COPY) = -1 (FILE-COPY).
DEF CSYMT         LOC OF COPIED SYM TBL.

```

```

DEF      DISPLAC      HOLDS DISPLACEMENT VALUE.
DEF      LINKCY       LINK HOLDER FOR COPY WORK.
DEF      LINKWY       LINK HOLDER WHILE WINDOWING.
DEF      NAMERKEY     KEY FOR NAME RECORD ACCESS.
DEF      GOSTATE      PTR TO GO-STATE IN STATE-INDICATOR.
DEF      BRNVAL       BRANCH VALUE FOR ERR-CTRL.
DEF      ERRORID      INTERNAL I.D. FOR ERROR.
DEF      ELINK        LINK HOLDER FOR ERR HANDLING.
DEF      EREGS        REG SAVE AREA DURING ERR-CTRL TEST.
DEF      ERRLOC       HOLDS ERROR LOC (CURRLINO,NAMEPTR).
DEF      ERRNUM       HOLDS ERROR NUMBER.
DEF      STRAPOPT     4013 STRAP OPTION,NO. OF CHARS.

SPACE 3
*
*   PARAMETERS
*
SPACE
MAXCOL   EQU      131      MAX. NO.OF COLUMNS OF INPUT ACCEPTED
*                               (INCLUDING THE CARRIAGE RETURN).
SPACE
MAXFRENS EQU      32      MAX. NO.OF FREE-TABLE ENTRIES.
SPACE
NAMLIMIT EQU      77      MAX. NO.OF CHARS USED IN A NAME.
SPACE
NAMEWDSZ EQU      (NAMLIMIT+3)**-2  MAX. NO.OF WDS TO HOLD A NAME.
SPACE 3
*
*   DOUBLEWORD BOUNDARY DEPENDENT CONTEXT
*
*       BOUND      8
*
*   (NON-OVERLAYABLE DOUBLEWORD CONTEXT)
*
CONSTAD  DATA    0      MUST BE ZERO.
RES      1      TEMP FOR ACQUIRING A CONSTANT.
SPACE
FREETBL  D01      MAXFRENS
DATA     0,0      FREE TABLE -- FOR EACH ENTRY:
*                               EVEN WD= LOC OF FREE BLK, ODD= SIZE.
*                               NOTE--UNUSED ENTRIES MUST CONTAIN ZEROES.
SPACE 2
IMAGE    RES      256**-2  IMAGE BUFFER (HI FOR OUTPUT WORK).
SPACE 2
*   BLOCK OF 'STRAY' DATA BLOCK PTRS. THIS BLOCK MUST CONTAIN ALL CELLS
*   THAT MAY REFERENCE A DATA BLK (EXCEPT FOR DATA BLK PTRS IN: THE
*   SYMBOL TABLE, THE EXECUTION STACK, OR OTHER DATA BLOCKS).
*
*   NOTE: IT IS ABSOLUTELY NECESSARY THAT 'STRAY' CELLS CONTAIN ZERO
*   WHEN NOT ACTUALLY REFERENCING A DATA BLOCK.
*
STRAYS   RES      0 (STARTS ORDINARY STRAY DATA BLK POINTERS).
LBLOCK  EQU      STRAYS   LOOP CTRL BLK FOR TRICKY OPERATORS
FDEFPTR DATA    0      FUNCTION DESCRIPTOR.
XTEMP   DATA    0      INDEX OR COORDINATE.
NILCK   DATA    0
LFARG   DATA    0      LEFT ARG.
RTARG   DATA    0      RIGHT ARG.
RESULT  DATA    0      RESULT.
NSTRAYS EQU      %-STRAYS =NO.OF ORDINARY STRAY DB PTRS.
SPACE 2
*
*   (INSERT UNIQUE STRAY DATA BLK PTRS. HERE)
*
DBROOT   DATA    0      ROOT PTR FOR LIST OR FUNC.DESRIPTOR
DBSERIES DATA    0      PTR INTO A LIST OR FUNC.DESRIPTOR
GRAFPNTR DATA    0      GRAPHICS SPECIAL BLOCK.
CURRCS   DATA    0      PTS AT CURRENT CODESTRING DATA BLK,
*                               AT ITS 'OFFSET' WD--I.E. HDR + 2.
SPACE 2
STRAYBLK EQU      %-STRAYS  TOTAL SIZE OF STRAY BLOCK.
*
*   END OF STRAY DATA BLOCK POINTER REGION. *****
*
SPACE 3
EREGS    RES      8      R1 - R6 SAVED DURING ERR-CTRL TEST.
ERRLOC   EQU      EREGS+6  ERR LOC (CURRLINO,NAMEPTR) OR 0, JUNK
ERRNUM   EQU      EREGS+7  LATEST RECORDED ERROR NUMBER OR 0.
SPACE
ERRORID  RES      1      ERROR I.D. (INTERNAL).

```

SPACE					
ELINK	RES	1		LINK HOLDER DURING ERR HANDLING.	
SPACE					
GOSTATE	RES	1		GO-STATE FOR STATE-INDIC CLEARANCE.	
SPACE					
BRNVAL	RES	1		ERR-CTRL BRANCH VALUE OR NEGATIVE.	
SPACE					
OFFSET	RES	1		OFTEN CONTAINS BYTE OFFSET TO LATEST	
*				BYTE SCANNED IN CODESTRING.	
*					
WHATERR	RES	1		HOLDS ERROR INFO WHEN DIAGNOSTIC IS	
*				NEEDED, SEE FOR INSTANCE 'EXERR'.	
*					
LINKOLAY	RES	1		LINK HOLDER FOR OVERLAY SWITCHING.	
*					
LINKCS	RES	1		LINKAGE TO CODESTRINGER.	
*					
ERRCOL	RES	1		BYTE ADDR.OF ERR COLUMN IN IMAGE.	
*					
HICOL	RES	1		INDICATES HI COLUMN IN IMAGE.	
*					
CONSTKEY	RES	1		BYTE ADDR.OF 1ST CHAR OF CONSTANT.	
*					
OFFSETK	RES	1		PTR TO CODESTR.DESIG. BEFORE CONST.	
*					
SYMT	RES	1		PTS AT 1ST WD OF SYMBOL TABLE.	
*					
NAMEPTR	RES	1		PTR INTO SYM TBL (AN OFFSET).	
*					
SYMFSIZE	RES	1		HOLDS # DBLWD ENTRIES IN SYMBOL TBL.	
*					
NSYMTWDS	RES	1		TWO TIMES SYMFSIZE.	
*					
FREETOTL	DATA	0		HOLDS TOTAL AMT OF FREE SPACE GIVEN	
*				BY THE FREE TABLE ENTRIES.	
LINKGC	RES	1		LINKAGE FOR GARBAGE COLLECTOR.	
*					
LINKWS	RES	1		LINKAGE FOR WORKSPACE MGMT ROUTINES.	
SPACE					
CXSCRTCH	RES	4			
SPACE					
CONSTTYP	RES	1			
SPACE					
NSPILLED	RES	1			
SPACE					
NAMEWDS	RES	1		TEMP FOR ACQNAME	
SPACE					
FBOUNDS	RES	MAXFRENS+1		BOUNDS FOR GARBAGE COLLECTION.	
FAQMS	RES	MAXFRENS+1		ACCUMS FOR GARBAGE COLLECTION.	
SPACE					
NR2MOVE	RES	1		# REGIONS TO MOVE FOR GARB. COLL.	
SPACE					
GCTEMP	RES	1		TEMP FOR GARBCOLL	
SPACE					
XSEGBRK	DATA	0		USED TO DETECT BREAK DURING	
*				EXECUTION SEGMENT OPERATION.	
SPACE					
OLDMODE	RES	1		HOLDS MODE INFO DURING FUN DEFN.	U03-0006
SPACE					U03-0007
QUIETFLG	RES	1		= 0 IF SAVED MSG TO BE DISPLAYED.	
SPACE					
CATCHTBL	DATA	-1		CATCH TABLE: VAR.NAME PTR # 1	
	RES	1		FUN.NAME PTR # 1	
	DATA	-1		VAR.NAME PTR # 2	
	RES	1		FUN.NAME PTR # 2	
SPACE					
OBSFLAG	DATA	0		OBSERVE-CMD FLAG (0=NO & NEG.=YES).	
OBSERVE	RES	1		OBSERVATION SETTING (0=NO & -1=YES).	
OBSAVE	RES	16		REG.SAVE AREA DURING AN OBSERVATION.	
SPACE					
	BOUND	8	SOME COPY CONTEXT		
COPYSAVE	RES	16	COPY	REG SAVE & SCRATCH.	
BACKUP	RES	1	COPY	COPY SYM TBL RE-SCAN FLAG.	
BLANKCNT	RES	1	COPY	# HOLES LEFT IN ACTIVE SYM TBL.	
COPYBASE	RES	1	COPY	LOC OF COPY DATA REGION IN COPY WS.	
COPYHOME	RES	1	COPY	LOC OF COPY DATA REGION IN ACTIVE WS	
COPYSIZE	RES	1	COPY	# WDS COPY DYN. OR COPY DATA REGION.	
CORVSFIL	RES	1	COPY	= 0 (CORE-COPY) = -1 (FILE COPY).	

CSYMT	RES	1	COPY	LOC OF COPIED SYM TBL.
DISPLAC	RES	1	COPY	HOLDS DISPLACEMENT VALUE.
LINKCY	RES	1	COPY	LINK HOLDER FOR COPY WORK.
LINKWY	RES	1	COPY	LINK HOLDER WHILE WINDOWING.
NAMERKEY	RES	1	COPY	KEY FOR NAME RECORD ACCESS.
SPACE				
GARBSAVE	RES	16		REG SAVE FOR GARBCOLL & OTHERS.
SPACE				
DREFSAVE	RES	6		REG SAVE FOR DREF.
SPACE				
LOCNEED	RES	1		LOC NEEDED FOR EXPANDING COMMON.
NEWBOUND	EQU	LOCNEED		LOC NEEDED FOR EXPANDING DYNAMIC.
SPACE				
BLKWANTD	RES	1		SIZE OF NEW DATA BLK FOR ALOCBLK.
STRAPOPT	DATA	7		DEFAULT 7-CHAR READ ON GRAF INPUT.
PAGE				

SPACE 2				
Z	SET	%-CONTEXT@		SIZE OF CONTEXT MODULE IN HEX.
SPACE				
Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536		SIZE IN DECIMAL.
SPACE 2				
END				

	TITLE	'CS-B00,08/22/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	
CS@	RES	0	CONTROL CARDS NOT PRINTED.
*			ORIGIN OF CODESTRINGER MODULE.
*	REF'S	AND	DEF'S
*			
	DEF	CS@	= START OF CS MODULE.
	DEF	CS1	ENTER TO CODESTRING ALL OF IMAGE.
	DEF	CSN	ENTER TO CODESTRING PART OF IMAGE.
	DEF	CSZ	ENTER TO FINISH CODESTRING.
	DEF	ERLSCAN	ENTER TO PRODUCE LINE-SCAN ERROR.
	DEF	RESXTEND	RESUME AFTER CALLING INPXTEND.
SPACE 3			
*			REFS TO PROCEDURE:
	REF	GIVEBACK	TO GIVE BACK UNUSED WDS OF DATA BLK.
	REF	CMDREC	COMMAND RECOGNIZER.
	REF	DELCK	ON DEL, GO TO FUNDEF MODULE FOR CKS.
	REF	LSCANERR	LINE-SCAN ERROR ROUTINE.
	REF	ERXEQ	LINE-SCAN ERR ATTEMPTING 'EXECUTE'.
	REF	CSERRH	HANDLE CODESTRING ERROR.
	REF	DREF	DE-REFERENCES BAD CODESTRING BLK.
	REF	ACQNXCC	ACQUIRE NEXT CHAR AND ITS CODE.
	REF	ACQNB	ACQUIRE NON-BLANK CHAR AND ITS CODE.
	REF	ACQNXNB	ACQ NEXT NON-BLANK & ITS CODE.
	REF	ACQCONST	TRIES TO ACQUIRE A CONSTANT.
	REF	ACQNAME	ACQUIRES A NAME.
	REF	ALOCNONX	ALLOCs DATA BLK, HEADER + N WDS.
	REF	GARBCOLL	GARBAGE COLLECTOR.
	REF	BCONTOFF	EXIT IN CASE OF HANG-UP DURING
	REF	INPXTEND	INPUT FOR LINE EXTENSION.
SPACE 2			
*			REFS TO CONTEXT:
	REF	LINKCS	LINKAGE TO CODESTRINGER.
	REF	MODE	INPUT MODE.
	REF	CURRCS	CODESTRING DATA BLOCK PTR (HDR+2).
	REF	STATEPTR	PTS AT TOP STATE-ENTRY IN STACK.
	REF	IMAGE	INTERNAL CHAR IMAGE OF INPUT LINE.
	REF	BREAKFLG	SET NEG. IF HANG-UP OCCURS.
	REF	ERRCOL	BYTE ADDR.OF ERR DETECTION COLUMN.
	REF	HICOL	BYTE ADDR.OF NEWLINE'S COL. IN IMAGE
	REF	OFFSETK	PTR TO CODE DESIG BEFORE CONSTANT.
	REF	OFFSET	PTR TO CODE DESIG (USUALLY LATEST).
	REF	CONSTKEY	BYTE ADDR.OF 1ST CHAR OF CONSTANT.
	REF	CONSTBUF	CONSTANT BUFFER.
	REF	CONSTTYP	TYPE OF CONSTANT.
	REF	QCNT	ODD IF 1ST IMAGE HAS UNBALANCED 'S.
	REF	FREETBL	PTS AT 1ST FREE LOC FOR NEW DATA BLK
	REF	BLKWANTD	ACTUAL SIZE ALLOCATED FOR LATEST DB.
	REF	SYMT	PTS AT 1ST WD OF SYMBOL TABLE.
	REF	SYMFSIZE	NO.OF SYMBOL TABLE ENTRIES.
SPACE			
*			REFS TO CONSTANTS:
	REF	X1FFFF	X'1FFFF'
	REF	F0F9	'0', '9'
	REF	NONAME	10,138 = CODE DESIGS NOT NAME-CHARS
SPACE			REFS TO ERROR I.D. EQU'S:
*			
SPACE			
	REF	IDSYMFUL	SYM TBL FULL.
	REF	IDWSFULL	WS FULL.
*			
*	STANDARD EQU'S		
*		REGISTERS	
R1	EQU	1	
R2	EQU	2	
R3	EQU	3	
R4	EQU	4	
R5	EQU	5	
R6	EQU	6	
R7	EQU	7	
R8	EQU	8	
R9	EQU	9	
R10	EQU	10	
R11	EQU	11	
R12	EQU	12	
R13	EQU	13	

```

R14      EQU      14
R15      EQU      15
*
* CODESTRING DESIGNATORS
*
BYNCODE  EQU      10      BYTE-FORM INTEGER SCALAR (10 - 255).
INTGCODE EQU      11      WORD-FORM INTEGER SCALAR.
REALCODE EQU      12      REAL SCALAR.
TEXTCODE EQU      13      TEXT SCALAR.
VLCODE   EQU      14      LOGICAL VECTOR.
VICODE   EQU      15      INTEGER VECTOR.
VRCODE   EQU      16      REAL VECTOR.
VTXCODE  EQU      17      TEXT VECTOR.
VTXLCODE EQU      18      LONG TEXT VECTOR (EXCEEDS 255 CHARS)
ERRCODE  EQU      19      ERROR.
LAMPCODE EQU      20      COMMENT
NAMECODE EQU      23      NAME (BUT NOT STOP OR TRACE NAME).
BOSCODE  EQU      37      BEGINNING-OF-STMT.
MONADIC  EQU      50      1ST MONADIC OPERATOR.
DOMCODE  EQU      80      DOMINO OPERATOR.
DYADIC   EQU      DOMCODE 1ST DYADIC OPERATOR.
BANGCODE EQU      101     EXCLAMATION POINT OPERATOR.
COLNCODE EQU      135     COLON.
DMYCODE  EQU      138     DUMMY.
*
* OTHER EQU'S
*
NEWLINE  EQU      X'15'   END MARK.
EFLAG    EQU      X'10000' 'EXECUTE' FLAG FOR QUAD-STATE.
TYPECS   EQU      7       CODESTRING TYPE OF DATA BLOCK.
*
* DOUBLEWORD CONSTANTS
*
BOUND     8
ZEROONE   DATA 0,1
ZERONINE  DATA 0,9
POSSMON   DATA DOMCODE,BANGCODE CODESTR.OPS CONVERTIBLE TO MONADICS.
PAGE
*****
*
* EASY AND SPECIAL DEFINITIONS FOR CODESTRINGING.
*
* THESE DEFINE A VALUE FOR EACH 'IN2CODE' CHARACTER IN THE
* RANGE BETWEEN -- COMMENT (LAMP) AND COLON.
* THE VALUES ARE ORDERED AND ARE ASSOCIATED WITH:
* LC4TBL -- LEFT CONTEXT ASSIGNMENTS
* ITEMCTX -- SCAN ITEM CONTEXT (FOR CONTACT ANALYSIS)
* CSV -- A VECTORED TABLE WITHIN THE CODESTRING
* ROUTINE.
*
EASY      EQU      0       EASILY HANDLED SCAN ITEMS
QUADS     EQU
SEMICOLN  EQU      QUADS+1
LBRACKET  EQU      SEMICOLN+1
RBRACKET  EQU      LBRACKET+1
LPAREN    EQU      RBRACKET+1
RPAREN    EQU      LPAREN+1
BRANCH    EQU      RPAREN+1
ASSIGN    EQU      BRANCH+1
SMCIRCLE  EQU      ASSIGN+1
SPECIAL   EQU      SMCIRCLE+1     SPECIALLY HANDLED SCAN ITEMS
BLANK     EQU      SPECIAL
DEL       EQU      BLANK+1
NEGATIVE  EQU      DEL+1
QUOTE     EQU      NEGATIVE+1
COMMENT   EQU      QUOTE+1
ENDMARK   EQU      COMMENT+1
DOT       EQU      ENDMARK+1
COP       EQU      DOT+1         (OP MAY TAKE COORDINATE SPECIFICTN.)
OP        EQU      COP+1
BAD       EQU      OP+1         BAD SCAN ITEMS
PAGE
*****
*
* SCANITEM -- A TABLE OF BYTES. THERE IS A BYTE CORRESPONDING TO
* EACH 'IN2CODE' CHARACTER VALUE BETWEEN COMMENT (LAMP) AND
* COLON. IT IS CONSIDERED IMPOSSIBLE TO OBTAIN SCAN ITEMS
* OUTSIDE THIS RANGE (DIGITS AND NAME-START CHARS ARE DEALT

```


* WITH SEPARATELY). NOT ALL VALUES WITHIN THIS RANGE ARE *
 * POSSIBLE EITHER--SUCH ENTRIES ARE MARKED WITH AN ASTERISK *
 * IN THE COMMENTS FIELD. *
 * THE VALUE CONTAINED IN EACH BYTE OF THE TABLE IS DEFINED *
 * BY THE EASY AND SPECIAL DEFINITIONS ABOVE. *
 * *
 * *
 * *
 * *
 * *
 * *

	VALUE	HEX OF INTERNAL FORM	INTERNAL CHARACTER	
	-----	---	-----	
SCANITEM	DATA,1	COMMENT	@ 20	COMMENT (LAMP)
	DATA,1	BAD	@ 21	*
	DATA,1	BAD	@ 22	*
	DATA,1	BAD	@ 23	*
	DATA,1	BAD	@ 24	UNUSED
	DATA,1	QUADS	@ 25	QUAD+0
	DATA,1	QUADS	@ 26	QUAD-1
	DATA,1	QUADS	@ 27	QUAD-2
	DATA,1	QUADS	@ 28	QUAD-3
	DATA,1	QUADS	@ 29	QUAD-4
	DATA,1	QUADS	@ 30	QUAD-5
	DATA,1	QUADS	@ 31	QUAD-6
	DATA,1	QUADS	@ 32	QUAD-7
	DATA,1	QUADS	@ 33	QUAD-8
	DATA,1	QUADS	@ 34	QUAD-9
	DATA,1	QUADS	@ 35	QUAD
	DATA,1	QUADS	@ 36	QUOTE-QUAD
	DATA,1	BAD	@ 37	*
	DATA,1	SEMICOLN	@ 38	SEMICOLON
	DATA,1	LBRACKET	@ 39	LEFT BRACKET
	DATA,1	RBRACKET	@ 40	RIGHT BRACKET
	DATA,1	LPAREN	@ 41	LEFT PAREN
	DATA,1	RPAREN	@ 42	RIGHT PAREN
	DATA,1	BRANCH	@ 43	BRANCH ARROW
	DATA,1	ASSIGN	@ 44	ASSIGN ARROW
	DATA,1	DOT	@ 45	DOT OR DECIMAL PT
	DATA,1	SMCIRCLE	@ 46	SMALL CIRCLE
	DATA,1	BAD	@ 47	UNUSED
	DATA,1	BAD	@ 48	UNUSED
	DATA,1	BAD	@ 49	UNUSED
	DATA,1	ENDMARK	@ 50	END-OF-INPUT (KEY CHAR)
	DATA,1	BLANK	@ 51	BLANK (KEY CHAR)
	DATA,1	QUOTE	@ 52	QUOTE (KEY CHAR)
	DATA,1	BAD	@ 53	*
	DATA,1	BAD	@ 54	*
	DATA,1	BAD	@ 55	*
	DATA,1	BAD	@ 56	*
	DATA,1	BAD	@ 57	*
	DATA,1	BAD	@ 58	*
	DATA,1	BAD	@ 59	*
	DATA,1	BAD	@ 60	*
	DATA,1	BAD	@ 61	*
	DATA,1	BAD	@ 62	*
	DATA,1	BAD	@ 63	*
	DATA,1	BAD	@ 64	*
	DATA,1	BAD	@ 65	*
	DATA,1	BAD	@ 66	*
	DATA,1	BAD	@ 67	*
	DATA,1	BAD	@ 68	*
	DATA,1	BAD	@ 69	*
	DATA,1	BAD	@ 70	*
	DATA,1	BAD	@ 71	BAD CHAR (KEY CHAR)
	DATA,1	BAD	@ 72	UNUSED
	DATA,1	BAD	@ 73	UNUSED
	DATA,1	OP	@ 74	I-BEAM
	DATA,1	OP	@ 75	GRADE-UP
	DATA,1	OP	@ 76	GRADE-DOWN
	DATA,1	OP	@ 77	NOT
	DATA,1	OP	@ 78	UNUSED
	DATA,1	OP	@ 79	UNUSED
	DATA,1	OP	@ 80	MATRIX-DIVIDE
	DATA,1	OP	@ 81	ROLL OR DEAL
	DATA,1	OP	@ 82	T-BAR
	DATA,1	OP	@ 83	COMPRESS OR REDUCE ON 1ST COORD
	DATA,1	COP	@ 84	COMPRESS OR REDUCE
	DATA,1	OP	@ 85	IOTA
	DATA,1	OP	@ 96	RHO

DATA,1	COP	@ 87	COMMA
DATA,1	OP	@ 88	REVERSE OR ROTATE ON 1ST COORD
DATA,1	COP	@ 89	REVERSE OR ROTATE
DATA,1	OP	@ 90	TRANSPOSE
DATA,1	OP	@ 91	PLUS
DATA,1	OP	@ 92	MINUS
DATA,1	OP	@ 93	TIMES
DATA,1	OP	@ 94	DIVIDE
DATA,1	OP	@ 95	EXPONENT
DATA,1	OP	@ 96	LOG
DATA,1	OP	@ 97	CIRCLE
DATA,1	OP	@ 98	MAX OR CEILING
DATA,1	OP	@ 99	MIN OR FLOOR
DATA,1	OP	@100	ABSOLUTE OR RESIDUE
DATA,1	OP	@101	EXCLAMATION PT
DATA,1	OP	@102	LESS THAN
DATA,1	OP	@103	LESS THAN OR EQUAL
DATA,1	OP	@104	GREATER THAN
DATA,1	OP	@105	GREATER THAN OR EQUAL
DATA,1	OP	@106	NOT EQUAL
DATA,1	OP	@107	EQUAL
DATA,1	OP	@108	AND
DATA,1	OP	@109	OR
DATA,1	OP	@110	NAND
DATA,1	OP	@111	NOR
DATA,1	OP	@112	UNUSED
DATA,1	OP	@113	UNUSED
DATA,1	OP	@114	DECODE
DATA,1	OP	@115	ENCODE
DATA,1	OP	@116	TAKE
DATA,1	OP	@117	DROP
DATA,1	OP	@118	EXPAND ON 1ST COORD
DATA,1	COP	@119	EXPAND
DATA,1	OP	@120	EPSILON
DATA,1	BAD	@121	UNUSED
DATA,1	BAD	@122	UNUSED
DATA,1	BAD	@123	DIERESIS
DATA,1	NEGATIVE	@124	NEGATIVE SIGN
DATA,1	BAD	@125	UNDERSCORE
DATA,1	BAD	@126	DOLLAR SIGN
DATA,1	BAD	@127	ALPHA
DATA,1	BAD	@128	OMEGA
DATA,1	DEL	@129	DEL
DATA,1	DEL	@130	LOCKED DEL
DATA,1	BAD	@131	LEFT CUP
DATA,1	BAD	@132	RIGHT CUP
DATA,1	BAD	@133	CAP
DATA,1	BAD	@134	CUP
DATA,1	BAD	@135	COLON

*

BOUND	4	INSURE WORD BOUNDARY AFTER SCANITEM
-------	---	-------------------------------------

 *

PAGE

*

LCBITS	-- LEFT CONTEXT BITS. EACH ENTRY SIGNIFIES A LEFT CONTEXT ASSOCIATED WITH AN ITEM DESCRIBED IN THE COMMENTS FIELD. BIT ENTRY (FOR M -- WHICH MUST BE MOST SIGNIFICANT).
NOTE	-- IT IS ESSENTIAL THAT NO LEFT CONTEXT BIT REACH BIT POSITION 16.

 *

LCBITS	EQU	1	LEFT CONTEXT BIT SETTINGS
D	EQU	LCBITS	DOT
B	EQU	D**1	BRANCH
A	EQU	B**1	ASSIGN
RO	EQU	A**1	RPAREN OR SMALL CIRCLE
LB	EQU	RO**1	LBRACKET
RB	EQU	LB**1	RBRACKET
ZC	EQU	RB**1	ENDMARK OR COMMENT
S	EQU	ZC**1	SEMICOLN
LQ	EQU	S**1	LPAREN OR QUADS
CN	EQU	LQ**1	CONSTANT OR NAME
O	EQU	CN**1	OPERATOR
M	EQU	O**1	MONADIC IMPLIED (M MUST HAVE THE HIGHEST BIT SETTING)

*

PAGE

 *

*

```

* 'LC4' DEFINITIONS
*   DEFINES A LEFT CONTEXT FOR ITEMS DESCRIBED IN THE COMMENTS
*   FIELD.
*
*   THE PRESENCE OF A PARTICULAR BIT INDICATES THAT LEFT CONTEXT
*   IS CORRECT FOR A SCAN ITEM (SEE ALSO 'ITEM' DEFINITIONS,
*   BELOW) HAVING MATCHING BIT SET.
*
*   THE M BIT POSITION IMPLIES THAT THE SCAN ITEM IS A MONADIC
*   OPERATOR IF IT HAPPENS TO BE AN OPERATOR AT ALL.
*
LC4STNAM EQU      A                      STOPNAME OR TRACENAME
LC4NAME  EQU      0+CN+LQ+S+ZC+RB+LB+R0+A  ORDINARY NAME
LC4RB    EQU      0+CN+LQ+S+ZC+RB+LB+R0+A  RIGHT BRACKET
LC4QUADS EQU      0+CN+S+ZC+RB+LB+R0+A     QUADS
LC4CONST EQU      0+CN+S+ZC+RB+LB+R0       CONSTANT
LC4RP    EQU      0+CN+S+ZC+RB+LB+R0       RIGHT PAREN
LC4SEMI  EQU      M+O+CN+LQ+S+ZC+RB+B     SEMICOLON
LC4BOS   EQU      M+O+CN+LQ+S+ZC+B        BEGINNING-OF-STMT
LC4BRNCH EQU      M+O+CN+LQ+S+ZC          BRANCH
LC4LB    EQU      M+O+CN+LQ+S+RB          LEFT BRACKET
LC4COP   EQU      M+O+CN+LQ+LB            COORDINATIBLE OPERATOR
LC4OP    EQU      M+O+CN+LQ+D            OPERATOR
LC4ASS   EQU      M+O+CN+LQ              ASSIGN
LC4LP    EQU      M+O+CN+LQ              LEFT PAREN
LC4DOT   EQU      0                      DOT (NOT DEC.PT.)
LC4SMCRC EQU      D                      SMALL CIRCLE
LC4ALL   EQU      X'7FFF'                BAD CHAR (ALL SCAN ITEMS ACCEPTED)
PAGE
*****
*
* LC4TBL  -- ORDERED TABLE OF HALFWORDS USED TO ASSIGN A LEFT CONTEXT
*          SETTING FOR THE ITEM DESCRIBED IN THE COMMENTS FIELD.
*          (SEE 'EASY' AND 'SPECIAL' DEFINITIONS FOR ASSOCIATION OF
*          THE CHOSEN ORDERING.)
*
LC4TBL  DATA,2  LC4QUADS      @ QUADS
          DATA,2  LC4SEMI     @ SEMICOLON
          DATA,2  LC4LB       @ LEFT BRACKET
          DATA,2  LC4RB       @ RIGHT BRACKET
          DATA,2  LC4LP       @ LEFT PAREN
          DATA,2  LC4RP       @ RIGHT PAREN
          DATA,2  LC4BRNCH    @ BRANCH
          DATA,2  LC4ASS      @ ASSIGN
          DATA,2  LC4SMCRC    @ SMALL CIRCLE
          DATA,2  0           @ (BLANK DOESN'T AFFECT LEFT CONTEXT)
          DATA,2  0           @ DEL (SHOULD OPEN OR CLOSE FUNCTION)
          DATA,2  0           @ NEGATIVE SIGN (CONTEXT SET SPECIAL)
          DATA,2  LC4CONST    @ QUOTE (I.E. TEXT CONSTANT)
          DATA,2  ITEMEND     @ COMMENT (ENDMARK GUARANTEED NEXT)
          DATA,2  LC4ALL      @ (ENDMARK DOESN'T HAVE LEFT CONTEXT)
          DATA,2  LC4DOT      @ DOT
          DATA,2  LC4COP      @ COP
          DATA,2  LC4OP       @ OP
*
          BOUND      4                      INSURE WORD BOUNDARY AFTER LC4TBL
*
PAGE
*****
* 'ITEM' DEFINITIONS
*   DEFINES A CURRENT CONTEXT FOR A SCAN ITEM DESCRIBED IN THE
*   COMMENTS FIELD.
*
*   THE SCAN ITEM IS IN PROPER CONTEXT (I.E. CONTACT ANALYSIS
*   FOR CODESTRINGING) IF A MATCHING BIT OCCURS IN THE LEFT
*   CONTEXT SETTING MOST RECENTLY ESTABLISHED.
*
*   THE M BIT POSITION IS USED TO DETERMINE IF LEFT CONTEXT
*   IMPLIES THAT AN OPERATOR (ITEMOPER) IS MONADIC.
*
ITEMOPER EQU      M+O                      ANY OPERATOR
ITEMCNST EQU      CN                      CONSTANT
ITEMNAME EQU      CN                      NAME
ITEMQUAD EQU      LQ                      QUADS
ITEMLP   EQU      LQ                      LEFT PAREN
ITEMSEMI EQU      S                      SEMICOLON
ITEMEND  EQU      ZC                      END-OF-INPUT

```

```

ITEMLAMP EQU      ZC      COMMENT
ITEMRB EQU        RB      RIGHT BRACKET
ITEMLB EQU        LB      LEFT BRACKET
ITEMSMCR EQU      RO      SMALL CIRCLE
ITEMRP EQU        RO      RIGHT PAREN
ITEMASS EQU       A       ASSIGN
ITEMBRCH EQU      B       BRANCH
ITEMDOT EQU       D       DOT
ITEMBAD EQU       0       BAD CHAR (NO LEFT CONTEXT MATTERS)
ITEMDEL EQU       ZC      DEL
ITEMBLNK EQU      X'7FFF' BLANK (ANY LEFT CONTEXT IS OK)
PAGE
*****
*
* ITEMCNTX -- ORDERED TABLE OF HALFWORDS USED TO ESTABLISH CURRENT
* CONTEXT FOR A SCAN ITEM DESCRIBED IN THE COMMENTS FIELD.
* (SEE 'EASY' AND 'SPECIAL' DEFINITIONS FOR ASSOCIATION OF
* THE CHOSEN ORDERING.)
*
ITEMCNTX DATA,2 ITEMQUAD QUADS @
DATA,2 ITEMSEMI SEMICOLON @@
DATA,2 ITEMLB LEFT BRACKET @
DATA,2 ITEMRB RIGHT BRACKET @
DATA,2 ITEMPLP LEFT PAREN @
DATA,2 ITEMRP RIGHT PAREN @
DATA,2 ITEMBRCH BRANCH @
DATA,2 ITEMASS ASSIGN @
DATA,2 ITEMSMCR SMALL CIRCLE @
DATA,2 ITEMBLNK BLANK @
DATA,2 ITEMDEL DEL OR LOCKED DEL @
DATA,2 ITEMCNST NEGATIVE SIGN @
DATA,2 ITEMCNST QUOTE @
DATA,2 ITEMLAMP COMMENT @
DATA,2 ITEMEND ENDMARK @
DATA,2 ITEMDOT|ITEMCNST DOT OR DECIMAL PT @
DATA,2 ITEMOPER COP @
DATA,2 ITEMOPER OP @
DATA,2 ITEMBAD BAD @
*
* BOUND 4 INSURE WORD BOUNDARY AFTER ITEMCNTX
*
CSTEXT STW,R7 OFFSET SAVE POS OF LATEST CODE DESIGNATOR.
CSTCHAR AI,R1 1 PT AT NEXT INTERNAL CHAR.
LB,R2 0,R1 GET IT.
CI,R2 ' ' CK FOR QUOTE MARK.
BNE CSTXMARK NO.
AI,R1 1 YES, LOOK AHEAD IN CASE DBL-QUOTE.
LB,R2 0,R1
CI,R2 ' '
BE CSTXMARK DBL-QUOTE, USE 2ND QUOTE.
LI,R3 TEXTCODE END TEXT, ASSUME TEXT SCALAR.
LW,R12 R7 = POS OF LAST ITEM STRUNG.
SW,R12 OFFSET = SIZE OF TEXT STRUNG.
CI,R12 1 JUST ONE MARK...
BE CSEXIT YES, IT IS A TEXT SCALAR.
LI,R3 VTXCODE NO, ASSUME (ORDINARY) TEXT VECTOR.
CI,R12 X'FF' IS SIZE BYTE-SIZE...
BLE CSLENGTH YES.
LI,R3 VTXLCODE NO, USE LONG TEXT VECTOR DESIGNATOR.
SCS,R12 -8 GET MOST SIGNIFICANT BYTE
AI,R7 1 AND PUT IT IN
STB,R12 *CURRCS,R7 CODESTRING.
SCS,R12 8 THEN GET LEAST SIGNIFICANT BYTE.
CSLENGTH AI,R7 1 PUT LENGTH BYTE IN CODESTRING.
STB,R12 *CURRCS,R7
CSEXIT BDR,R1 CSSETLC BACK UP ONE CHAR POS AND GO SET
* LEFT CONTEXT AND SO FORTH.
CSTXMARK AI,R7 1 STRING LATEST TEXT MARK.
STB,R2 *CURRCS,R7
CI,R2 NEWLINE WAS IT END-OF-CURRENT LINE...
BNE CSTCHAR NO, TRY NEXT CHAR.
LI,R11 0 YES, EXTENSION NEEDED.
STW,R11 CONSTKEY RESET PTR TO KEY CHAR OF CONSTANT.
STW,R7 OFFSETK SAVE CURRENT CODESTRING OFFSET.
B INPXTEND INPUT THE NEXT IMAGE FOR EXTENSION;
RESXTEND LW,R11 BREAKFLG AFTER RESUMING, TEST FOR HANG-UP...
BLZ HANGUP YES.
LW,R11 HICOL NO.

```

	AI, R11	1-BA(IMAGE)	= NO.OF CHARS TO CODESTRING.
	SLS, R11	1	DBL TO GET NO.OF WDS FOR THAT STUFF.
	LW, R2	CURRCS	PT AT CURRENT CODESTRING OFFSET WD.
	INT, R3	-2, R2	= CURRENT SIZE OF DATA BLK.
	LI, R12	X'7FFFC'	COMPUTE APPROXIMATE NO.OF WDS
	AND, R12	OFFSETK	ACTUALLY USED SO FAR.
	SLS, R12	-2	
	SW, R12	R3	= - APPX. AMT LEFT OVER IN THAT BLK.
	AW, R11	R12	= APPX. NEW AMT NEEDED.
	BLEZ	RESUME	RESUME IF WE ALREADY HAVE ENUF.
	AW, R3	R2	PT AT 2ND WD AFTER CURRENT CS BLOCK.
	SW, R3	FREETBL	= (2 - NO.OF WDS BETWEEN CURRENT CS
*			DATA BLK & 1ST FREE LOC) OR
*			= 2 - SIZE OF LONG-NAME DATA BLKS
*			THAT FOLLOW CURRENT CODESTR.
*	NOTE --	ONLY LONG-NAME DATA BLKS COULD FOLLOW SINCE THE ODD QCNT	
*		FORCED GARBAGE COLLECTION PRIOR TO CODESTRING DB ALLOCATION	
*		AND THEY ARE THE ONLY DATA BLKS ALLOCATED WHILE CODESTRINGING	
	BAL, R14	ALOCNONX	ALLOC NEW AMT, IF POSSIBLE...
	B	WSFULLN	OOPS -- WS FULL.
	LW, R11	BLKWANTD	OK, GET SIZE ACTUALLY ADDED.
	AWM, R11	-2, R2	ADD THAT TO CODESTR. DATA BLK HDR.
	AI, R3	-2	= - SIZE OF ANY LONG-NAME DATA BLKS.
	BEZ	RESUME	NONE BARRING CODESTRING EXTENSION.
	STW, R4	CONSTBUF+1	SOME, SAVE PTR TO WD AFTER LAST ONE.
	AW, R4	R3	PT AT 1ST WD OF 1ST LONG NAME DATA
*			BLK BARRING CODESTRING EXTENSION.
*	STW, R4	CONSTBUF	NOW CONSTBUF IS A DBLWD THAT BOUNDS
*			THE LONG NAME DATA BLOCK BARRIER.
*			ALSO, CONSTBUF ITSELF ACTS AS A
	AW, R4	BLKWANTD	SOURCE-BOUNDARY FOR MOVING THE BAR
	STW, R4	CONSTBUF+2	COMPUTE THE DESTINATION-BOUNDARY FOR
	LCW, R3	R3	MOVING THE BARRIER & SAVE IT.
*			= SIZE OF BARRIER. READY TO MOVE
*			THE LONG-NAME DATA BLOCKS TO THE
*			END OF THE NEW ALLOCATION, WORK
*			FROM HI END TO LO END OF THE
			BARRIER WHEN MOVING IT.
MOVEQ	AI, R3	-15	DO 15 WDS REMAIN TO MOVE...
	BGEZ	MOVE15	YES, MOVE THE HI 15 STILL REMAININ
	AI, R3	15	NO, GET SMALL AMT LEFT; SHIFT FOR
	SCS, R3	-4	ADDR = 0 & AMT IN BITS 0-3, AND
	LC	R3	PUT AMT IN CONDITION CODE.
	B	MOVE	
MOVE15	LCI	15	SET TO MOVE HIGHEST 15 REMAINING.
MOVE	LM, R4	*CONSTBUF, R3	GET WDS ACCORDING TO SOURCE-BOUND.
	STM, R4	*CONSTBUF+2, R3	STORE ACCORDING TO DESTINATION-BOUND
	AND, R3	X1FFFF	TEST FOR ANY AMT REMAINING TO MOVE.
	BGZ	MOVEQ	KEEP MOVING IF ANY REMAIN.
	LW, R1	SYMT	DONE. PT AT 1ST WD OF SYMBOL TBL.
	LI, R6	0	SET FOR SELECTIVE LOADS.
	LI, R7	X'1FFFF'	
	LW, R11	BLKWANTD	GET SIZE ADDED (FOR DISPLACEMENTS).
	LI, R14	20	= MAX NO. OF WDS PER NAME.
	LW, R2	SYMTSIZE	= NO.OF SYMBOL TABLE ENTRIES.
	AI, R1	1	PT AT 1ST NAME-INDICATOR WD ON SYMT.
SYMLOOK	CB, R14	*R1	IS THIS A LONG OR SHORT NAME...
	BL	NEXTSYM	SHORT, SKIP IT.
	LS, R6	0, R1	LONG, GET ITS LONG-NAME PTR.
	CLM, R6	CONSTBUF	IN RANGE OF LONG-NAME BLKS JUST HIT.
	BCS, 9	NEXTSYM	NO, GO ON TO NEXT SYM TBL ENTRY.
	AWM, R11	0, R1	YES, DISPLACE BY SIZE ADDED.
NEXTSYM	AI, R1	2	PT AT NEXT SYMBOL TABLE ENTRY
	BDR, R2	SYMLOOK	AND LOOP TILL DONE.
RESUME	LI, R1	BA(IMAGE) -1	PT JUST BEFORE 1ST NEW IMAGE CHAR.
	LW, R7	OFFSETK	RESTORE CODESTRING OFFSET.
	LI, R5	QUOTE	RESTORE SCAN ITEM INDICATOR.
	LI, R4	CSACQCK	RESET R4 FOR LATER CHAR ACQUISITIONS
	B	CSTCHAR	RESUME ADDING TO THE TEXT VECTOR.
SYMFULL	LI, R8	IDSYMFUL	= ERROR I.D. FOR 'SYM TBL FULL'.
CSERRROUT	LI, R7	CSERRH	HANDLE CODESTRING ERROR
	B	THROWOUT	AFTER THROWING CODESTRING AWAY.
WSFULLN	LI, R8	IDWSFULL	= ERROR I.D. FOR 'WS FULL'.
	B	CSERRROUT	
WSFULLC	LI, R8	IDWSFULL	= ERROR I.D. FOR 'WS FULL'.
	B	CSERRH	HANDLE CODESTRING ERR (NONE ALLOC'D)
HANGUP	BAL, R7	THROWOUT	THROW CODESTRING AWAY.

ERLSCAN	B	BCONTOFF	DO LIKE A)CONTINUE CMD.
	STW,R1	ERRCOL	= BYTE ADDR.OF ERROR PT IN IMAGE.
	LW,R14	MODE	CK FOR FUNCTION DEFINITION MODE.
	BEZ	ERLSCANF	YES, COMPLETE (MAYBE FIX) CODESTRING
	AI,R14	-2	CK FOR EVAL-INPUT MODE.
	BNEZ	ERLSCAND	NO.
	LI,R14	EFLAG	MAYBE, CK FOR 'EXECUTE'.
	CW,R14	*STATEPTR	
	BANZ	ERLSCANX	
*			YES, PRETEND THAT WE CAN EXECUTE,
ERLSCAND	LI,R7	LSCANERR	BUT WE WILL HIT SYNTAX ERR THEN.
THROWOUT	LI,R4	0	DISPLAY -- LINE-SCAN ERROR -- SOON.
	XW,R4	CURRCS	THROW OUT CURRENT CODESTRING:
	AI,R4	-2	ELIMINATE REFERENCE TO DATA BLK,
	B	DREF	PT AT ITS HEADER, AND
ERLSCANF	LI,R14	LSCANERR	GO TO ERR ROUTINE AFTER DE-REFING.
	STW,R14	LINKCS	EXIT VIA 'LINKCS' TO 'LSCANERR',
ERLSCANX	LW,R14	CONSTKEY	FORGETTING ORIGINAL LINKAGE.
	BEZ	ERLSCANT	HAS CONSTANT BEEN STARTED...
	LW,R1	CONSTKEY	NO, LAST CODE DESIGNATOR IS OK.
*			YES, PT AT KEY (START) OF LATEST
	LW,R7	OFFSETK	CONSTANT; WE'LL BACK UP THAT FAR
*			FORGETTING ANY CODESTRING FROM
ERLSCANT	AI,R1	-1	THAT POINT ON (GOOD OR BAD).
	LI,R3	ERRCODE	BACK UP 1 CHAR TO COUNTER 'CSCOMC'.
	LI,R5	ENDMARK	= CODESTRING DESIGNATOR FOR ERROR.
*			THIS PREVENTS LOOPING INDEFINITELY
*			FOR CONTACT ERROR AT THE ENDMARK
*			(THE ENDMARK IS RE-ACQ'D AND ITS
			CONTEXT CK'D BEFORE EXIT BY 'CSZ')
			RE-SET RETURN FROM 'ACQNXCC'.
CSCOMMNT	LI,R4	CSACQCK	SAVE PTR TO LAST ACCEPTED DESIGNATOR
CSCOMC	STW,R7	OFFSET	PEEK AT NEXT CHAR IN IMAGE.
	AI,R1	1	
	LB,R2	0,R1	
	CI,R2	NEWLINE	
	BE	CSCOMNL	CK FOR END MARK...
	AI,R7	1	YES
	STB,R2	*CURRCS,R7	NO, PLANT IT IN CODESTRING.
	B	CSCOMC	
CSCOMNL	CI,R3	ERRCODE	WAS THIS A LINE-SCAN ERROR...
	BNE	CSCOMEND	NO, END THE COMMENT.
	LW,R12	R1	YES, CALC. NO.OF CHARS TO THE RT
	SW,R12	ERRCOL	OF THE ERROR PT. (INCL. ENDMARK)
	AI,R7	1	PLANT THIS 'ERROR-POINTER' AT THE
	STB,R12	*CURRCS,R7	SPOT THE END MARK WOULD OCCUPY.
CSCOMEND	LW,R12	R7	CALC. LENGTH OF CODESTRING SINCE
	SW,R12	OFFSET	LAST ACCEPTED CODESTRING DESIG.
	B	CSELENGTH	
CSDOTPT	AI,R1	1	PEEK AHEAD FOR DIGIT...
	LB,R13	0,R1	
	AI,R1	-1	
	CLM,R13	F0F9	
	BCR,9	CSPT	YES, FRACTION CONSTANT.
	CI,R12	ITEMDOT	NO, CK CONTEXT FOR A DOT.
	BANZ	CSSETLC	OK, SET LEFT CONTEXT, STRING, GO ON.
	B	ERLSCAN	BAD -- LINE-SCAN ERROR.
CSPT	RES	0	@
CSNEG	RES	0	@ (ASSUME CONSTANT COMING UP).
CSNUM	CI,R12	ITEMCNST	@ CK CONTEXT FOR A CONSTANT ITEM...
	BANZ	CSCNST	OK, TRY FOR A CONSTANT.
	B	ERLSCAN	BAD -- LINE-SCAN ERROR.
CSCNST	STW,R7	OFFSETK	SAVE CODESTRING OFFSET.
	STW,R1	CONSTKEY	SAVE POS OF 1ST CHAR OF CONST.
	BAL,R4	ACQCONST	@ ACQUIRE CONSTANT (UNLESS FALSE-STRT)
	BDR,R1	ERLSCAN	@ OVERFLOW, PT AT THE LAST CHAR (PROB.
*			@ A DIGIT) OF NO. -- LINE-SCAN ERROR
	LW,R7	OFFSETK	@ NO 0'FLO -- RESTORE OFFSET.
	AI,R6	-1	TEST LENGTH OF CONSTANT...
	BEZ	CSS	1 -- SCALAR.
	BLZ	ERLSCAN	0 -- FALSE START (NOT NUMBER).
	AI,R6	1	>1-- VECTOR, R6 = LENGTH AGAIN.
	LI,R5	0	SET ELEMENT COUNTER.
	LW,R13	CONSTTYP	WHAT TYPE OF VECTOR...
	CLM,R13	ZEROONE	(0 OR 1 MEANS LOGICAL)
	BCS,9	CSVIR	INTEGER OR REAL.
	LI,R13	VLCODE	LOGICAL -- VECTOR-LOGL CODE DESIG.
CSLV	AI,R7	1	PT AT NEXT BYTE FOR CODESTRING.
	LI,R14	8	8 BITS PER LOGICAL BYTE VALUE.
CSLVN	SLS,R8	1	SHIFT TO MAKE ROOM FOR EACH BIT.

	OR, R8	CONSTBUF, R5	PLUG IN THAT BIT.
	AI, R5	1	PT AT NEXT ELEMENT, IF ANY.
	AI, R6	-1	DECR LENGTH.
	BEZ	CSLVD	DONE -- LENGTH IS IN R5 NOW.
	BDR, R14	CSLVN	LOOP TILL BYTE FULL.
	STB, R8	*CURRCS, R7	FULL -- STRING BYTE.
	B	CSLV	START A NEW ONE.
CSLVS	SLS, R8	1	SHIFT TO LEFT JUSTIFY LAST LOGL BYTE
CSLVD	BDR, R14	CSLVS	FINAL LOOP.
	STB, R8	*CURRCS, R7	STRING LAST LOGL BYTE VALUE.
	B	CSCLEN	PLUG IN NO.OF ELEMS. & CODE DESIG.
CSDMY	LI, R8	DMYCODE	DUMMY CODESTRING DESIGNATOR USED TO
	STB, R8	*CURRCS, R7	FILL OUT TO A WD BOUNDARY.
CSWSET	AI, R7	1	PT AT NEXT BYTE OF CODESTRING.
	CI, R7	3	TEST FOR WD BOUND...
	BANZ	CSDMY	NOT YET.
	SLS, R7	-2	NOW -- SHIFT TO WD RESOLUTION.
	B	*R14	EXIT FROM CSWSET.
CSVIR	BAL, R14	CSWSET	SET TO WD BOUNDARY CODESTRING.
	LW, R13	CONSTTYP	WHICH TYPE OF NUMERIC CONSTANT...
	BGZ	CSIV	INTEGER VECTOR.
	LI, R13	VRCODE	REAL VECTOR -- CODE DESIGNATOR.
	LI, R14	CSRVN	RETURN FROM 'CSRIWD' TO 'CSRVN'.
CSRVN	LD, R8	CONSTBUF, R5	GET REAL NUMBER.
	STW, R8	*CURRCS, R7	STRING MOST SIGNIFICANT WD.
	AI, R7	1	PT TO NEXT WD IN CODESTRING.
	B	CSRIWD	STRING IT AND TEST FOR LAST ELEMENT.
CSIV	LI, R13	VIWCODE	CODestring DESIGNATOR--VECTOR INTG.
	LI, R14	CSIVN	RETURN FROM 'CSRIWD' TO 'CSIVN'.
CSIVN	LW, R9	CONSTBUF, R5	GET INTEGER.
CSRIWD	STW, R9	*CURRCS, R7	STRING NUMBER IN R9.
	AI, R7	1	PT TO NEXT WD OF CODESTRING.
	AI, R5	1	PT AT NEXT ELEMENT, IF ANY.
	BDR, R6	*R14	LOOP TILL LAST (THEN R5 = LENGTH).
	SLS, R7	2	RESUME BYTE BOUNDARY FOR STRINGING.
	B	CSCLENB	STRING LENGTH & NO.OF ELEMENTS.
CSS	LW, R13	CONSTTYP	REAL NUMBER...
	BGEZ	CSI	NO, INTEGER (INCLUDES LOGICAL).
	LI, R13	REALCODE	SCALAR REAL CODESTRING DESIGNATOR.
	BAL, R14	CSWSET	SET TO WORD BOUNDARY IN CODESTRING.
	LD, R8	CONSTBUF	GET THE REAL NO.
	STW, R8	*CURRCS, R7	STRING MOST SIGNIFICANT WD.
	AI, R7	1	PT TO NEXT WD IN CODESTRING.
CSSWD	STW, R9	*CURRCS, R7	STRING NUMBER IN R9.
	AI, R7	1	PT TO NEXT WD IN CODESTRING.
	SLS, R7	2	RESUME BYTE BOUNDARY FOR STRINGING.
	B	CSCCD0	STRING CONSTANT'S CODE DESIGNATOR.
CSI	LW, R13	CONSTBUF	GET THE INTEGER.
	CLM, R13	ZERONINE	IF 0 THRU 9, INTEGER IS ITS OWN
	BCR, 9	CSCCD	CODestring DESIGNATOR.
	CI, R13	X'FFF00'	WELL, DOES INTG ONLY OCCUPY 1 BYTE
	BAZ	CSIBYTE	YEP.
	LI, R13	INTGCODE	NOPE, INTEGER CODE DESIGNATOR.
	LW, R9	CONSTBUF	GET IT AGAIN.
	LI, R14	CSSWD	GO TO 'CSSWD' AFTER
	B	CSWSET	SETTING CODESTRING TO WD BOUNDARY.
CSIBYTE	LI, R13	BYNCODE	BYTE-NUMBER CODESTRING DESIGNATOR.
	LW, R5	CONSTBUF	GET THE NO. THAT FITS IN 1 BYTE.
CSCLEN	AI, R7	1	PT TO NEXT BYTE OF CODESTRING.
CSCLENB	STB, R5	*CURRCS, R7	PLUG IN LENGTH BYTE OR SMALL INTG.
CSCCD	AI, R7	1	PT TO NEXT BYTE OF CODESTRING.
CSCCD0	STB, R13	*CURRCS, R7	PLUG IN CODESTRING DESIGNATOR.
	LI, R12	LC4CONST	SET LEFT CONTEXT FOR CONSTANT.
	B	CSARSET	REGS WERE PREPARED EARLIER, CK THE
*			NON-BLANK CHAR THAT TERMINATED
*			THE CONSTANT.
CSZ	STH, R7	*CURRCS	SET OFFSET--TO LAST CODESTRING BYTE.
	SLS, R7	-2	CALC ACTUAL SIZE OF DATA BLK NEEDED
	AI, R7	3	FOR THIS CODESTRING.
	LW, R4	CURRCS	PT AT THE DATA BLOCK HEADER.
	AI, R4	-2	
	INT, R11	0, R4	GET SIZE FIELD.
	SW, R11	R7	= AMT THAT COULD BE GIVEN BACK TO
	LW, R7	LINKCS	THE FREE TABLE. EXIT VIA LINK
	B	GIVEBACK	AFTER GIVING BACK EVEN AMOUNT.
CSNON	LB, R5	SCANITEM-(LAMPCODE** -2), R3	CONTACT CK THE SCAN ITEM
	CH, R12	ITEMCNTX, R5	R12 CONTAINS LEFT CONTEXT.
	BANZ	CSSPECCK	OK.

```

CSSPECK B ERLSCAN BAD, LINE-SCAN ERROR.
CI, R5 SPECIAL IS THIS A SPECIAL CHAR...
BGE CSV-SPECIAL, R5 YES, BRANCH INTO CSV-RANGE.
CSSETLC LH, R12 LC4TBL, R5 SET NEW LEFT CONTEXT.
AI, R7 1 STRING NEW CODESTRING DESIGNATOR.
STB, R3 *CURRCS, R7
B %+1 FALLS INTO CSV-RANGE FOR NEXT CHAR
* ACQ. NOTE THAT R4 STILL PTS TO
* 'CSACQCK'.
CSV B ACQNXCC CSV-RANGE--BRANCH TBL: @ BLANK
B DELCK * @ DEL
B CSNEG * @ NEGATIVE-SIGN
B CSTEXT * @ TEXT CONSTANT
B CSCOMMNT * (SEE ALSO THE @ COMMENT
B CSZ * 'SPECIAL' @ END-OF-CODESTRING
B CSDOTPT * DEFINITIONS-- @ DOT OR DEC.PT
B CSOP * I.E. EQU-CARDS) @ COORDINATIBLE OP
CSOP AI, R12 -M * @ ORDINARY OP
BLZ CSSETLC BRANCH IF NOT MONADIC CONTEXT.
CLM, R3 POSSMON MONADIC CONTEXT. CONVERT OP...
BCS, 9 CSSETLC NO, LEAVE IT ALONE.
AI, R3 MONADIC-DYADIC YES, MAKE IT A MONADIC OPERATOR.
B CSSETLC

PAGE
*****
*
* THE CODESTRINGER
*
* CS1--ENTRY PT TO START CODESTRINGING AT THE BEGINNING OF IMAGE BUF.
* CSN--ENTRY PT TO START CODESTRING AT A GIVEN POSITION IN IMAGE BUF.
* REGS FOR CSN ENTRY:
* R1 PTS AT THAT POSITION.
* R2 CONTAINS THAT CHAR.--REQUIRED TO BE NON-BLANK.
* R3 CONTAINS ITS CODE.
* REGS FOR CS1 AND CSN ENTRY:
* R12 LINK REGISTER. EXIT VIA SAVED LINK (IN LINKCS)
* REGS AT NORMAL EXIT (EXIT FROM CODESTRINGER NORMALLY OCCURS
* BY ENTERING 'GIVEBACK' TO RETURN ANY UNUSED WORDS
* FOR THE CODESTRING DATA BLOCK):
* R4 PTS AT DATA BLOCK HEADER.
* R11 CONTAINS AMOUNT OF UNUSED WORDS.
* R7 LINKAGE TO CS1 OR CSN.
* ALL OTHER REGS CAN BE PRESUMED VOLATILE.
* (ERROR EXIT IS TO 'LSCANERR' -- LINE-SCAN ERROR -- ALSO TAKES
* PLACE VIA 'GIVEBACK' FOR FUNCTION LINES. FOR THESE LINES, A
* PORTION OF THE LINE IS CODESTRUNG AS AN ERROR DESCRIPTION:
* TEXT, ERROR-PTR, LENGTH, ERROR-CODESTRING-DESIGNATOR. FOR NON-
* FUNCTION LINES, CODESTRING IS THROWN AWAY. SEE 'ERLSCAN'.)
*
CS1 LI, R1 BA(IMAGE) PT AT 1ST BYTE OF INTERNAL IMAGE.
BAL, R4 ACQNB ACQ NON-BLANK CHAR. AND CODE.
CSN CI, R2 ') IF 1ST 'CODESTRINGABLE' CHAR IS RT.
BE CMDREC PAREN, GO TO CMND. PROCESSOR.
STW, R12 LINKCS SAVE LINK TO CODESTRINGER.
LI, R12 1 DOES IMAGE CONTAIN AN ODD NO. OF
AND, R12 QCNT QUOTES...
BEZ ONELINER NO.
BAL, R8 GARBCOLL YES, TAMP SO CS DATA BLK CAN GRO
ONELINER LI, R12 0
STW, R12 CONSTKEY RESET PTR TO KEY CHAR OF CONSTANT.
LW, R11 HICOL = LOC OF ENDMARK IN INTERNAL IMAGE.
SW, R11 R1
AI, R11 1 = NO.OF CHARS TO CODESTRING.
SLS, R11 1 DBL THAT = NO.OF WDS TO ALLOCATE
* FOR CODESTRING; IT IS UNLIKELY
* THAT THIS MANY WDS WILL BE USED.
BAL, R14 ALOCNONX ALLOC THOSE WDS PLUS DATA BLK HDR.
B WSFULLC YUCK -- WS FULL BEFORE ANY CS.
LI, R11 TYPECS SET DATA BLK TYPE FIELD--CODESTRING.
STB, R11 *R4
AI, R4 2 PT AT 1ST WD TO RECEIVE CODESTRING.
STW, R4 CURRCS = PTR TO CODESTRING BLOCK, CURRCS
* WILL BE OK EVEN IF DATA BLK MOVES.
LI, R7 2 CODESTR STARTS AT BYTE 2.
CSBOS LI, R13 BOSCODE PUT BEGINNING-OF-STMT CODESTRING
STB, R13 *CURRCS, R7 DESIGNATOR INTO CODESTR. BLOCK.
LI, R12 LC4BOS = LEFT CONTEXT FOR BEGINNING-OF-STMT
CSARSET LI, R4 CSACQCK SET RETURN FROM 'ACQNXCC'.

```


CSACQCK	CLM, R3	NONAME	CK LATEST ACQUISITION...
	BCR, 9	CSNON	NOT A NAME OR NUMBER START
	BCS, 1	CSNUM	DIGIT, STARTS NUMBER
*			NAME-START CHAR
	STW, R7	OFFSET	SAVE POS OF LATEST CODESTR DESIGNATR
	CI, R12	ITEMNAME	CK FOR CONTACT ERROR FOR NAME.
	BANZ	CSANAME	OK.
	B	ERLSCAN	OOPS -- LINE-SCAN ERROR.
CSANAME	BAL, R12	ACQNAME	ACQUIRE THE NAME--PTR AND TYPE.
	B	SYMFULL	(SYM TBL FULL RETURN)
	B	WSFULLN	(WS FULL RETURN, DUE TO LONG NAME)
	LW, R7	OFFSET	RESTORE CODESTRING PTR.
	CI, R2	':'	DID NAME END ON A COLON...
	BNE	CSNAME	NO.
	CI, R7	2	YES, HAS CODESTRING REALLY BEGUN...
	BNE	ERLSCAN	YEP -- LINE-SCAN ERROR.
	LW, R4	MODE	NOPE -- FUNCTION DEFN MODE...
	BNEZ	ERLSCAN	NO -- LINE-SCAN ERROR.
	CI, R13	NAMECODE	YES, AN ORDINARY NAME...
	BNE	ERLSCAN	YUCK, STOP OR TRACE -- LINE-SCAN ERR
	LI, R13	COLNCODE	GET COLON CODESTRING DESIGNATOR.
	B	CSLBLN	START LABEL NAME OVER BOSCODE.
CSUNLBL	LI, R12	LC4NAME	SET LEFT CONTEXT FOR ORDINARY NAME.
	AI, R13	-NAMECODE	IS IT...
	BEZ	CSARSET	YES, RESUME WITH CHAR AFTER NAME.
	LI, R12	LC4STNAM	NO, STOP OR TRACE NAME.
	B	CSARSET	
CSNAME	AI, R7	1	UPDATE CODESTRING PTR.
CSLBLN	SCS, R6	-8	CODESTRING THE NAME PTR--
	STB, R6	*CURRCS, R7	MOST SIGNIFICANT BYTE
	AI, R7	1	FOLLOWED BY
	SCS, R6	8	
	STB, R6	*CURRCS, R7	LEAST.
	AI, R7	1	UPDATE CODESTRING PTR.
	STB, R13	*CURRCS, R7	ENTER NAME TYPE (OR COLON CODE).
	CI, R13	COLNCODE	WAS IT A LABEL...
	BNE	CSUNLBL	NO.
	LI, R13	NAMECODE	YES, REPLACE COLON CODE WITH
	STB, R13	*CURRCS, R7	ORDINARY NAME CODE.
	AI, R7	1	
	LI, R13	COLNCODE	THEN PLUG IN THE COLON CODE.
	STB, R13	*CURRCS, R7	
	AI, R7	1	PREPARE TO RESTART CODESTRINGING
	LI, R4	CSBOS	BY PUTTING BEGINNING-OF-STMT CODE
	B	ACQNXNB	AFTER THE COLON. EXIT TO CSBOS
*			AFTER GETTING NEXT NON-BLANK.

PAGE

SPACE 2

Z SET %-CS@ SIZE OF CS IN HEX.

SPACE

Z SET Z+Z/10*6+Z/100*96+Z/1000*1536 SIZE IN DECIMAL.

SPACE 2

END


```

TITLE      'INTRINS-B00,08/22/73,DWG702985'
SYSTEM     SIG7F
CSECT      1
PCC         0
INTRINS@   RES      0

```

CONTROL CARDS NOT PRINTED.
ORIGIN OF INTRINSIC FUNCTIONS MODULE

```

*
*  REF'S  AND  DEF'S
*

```

```

DEF      INTRINS@      = START OF INTRINS MODULE.
DEF      MIBEAM        MONADIC I-BEAM FUNCTIONS.
DEF      MTBAR         MONADIC T-BAR FUNCTION.
DEF      DTBAR         DYADIC T-BAR FUNCTION.
DEF      MINTRIN       MONADIC -- INTRINSIC FUNCTIONS.
DEF      DINTRIN       DYADIC -- INTRINSIC FUNCTIONS.
DEF      NINTRIN       NILADIC -- INTRINSIC FUNCTIONS.
DEF      QZOUT         STARTS QUAD-0 (GRAPHICS) OUTPUT.
DEF      QZOUTRET      RESUMES AFTER QUAD-0 OUTPUT.
DEF      QZIN         STARTS QUAD-0 (GRAPHICS) INPUT.
DEF      QZINRET       RESUMES AFTER QUAD-0 INPUT.
DEF      MXRETURN      RETURNS: MONADIC INTRINSIC OR OPERATOR,
DEF      DXRETURN      DYADIC INTRINSIC OR OPERATOR,
DEF      NIRETURN      NILADIC INTRINSIC,
DEF      SXRETURN      SUBSCRIPTING,
DEF      AXRETURN      ASSIGNED INDEXING,
DEF      XEQNIL        NO CODESTRING 'EXECUTE' OPERATION.
DEF      BCBRANCH      CLR BREAK & DO ERR-CTRL BRANCH.
DEF      ECBRANCH      DO ERR-CTRL BRANCH.
DEF      IV1           GET INTEGER NO. FROM RT ARG.
DEF      OPBREAK       BRK DURING OPERATOR EXECUTION.
DEF      ERLSERR       LINE-SCAN ERR (DURING EXECUTION).
DEF      ERSYN         SYNTAX ERROR.
DEF      ERUND         UNDEFINED.
DEF      ERNOR         NO RESULT.
DEF      ERIO          I/O ERR (DURING CODESTRING EXECUT'N)
DEF      ERDOMAIN      DOMAIN ERROR.
DEF      ERRANK        RANK ERROR.
DEF      ERLLENGTH     LENGTH ERROR.
DEF      ERWSFUL       WS FULL.
DEF      ERSING        SINGULAR MATRIX.
DEF      ERFORMAT      FORMAT ERROR.
DEF      ERINDEX       INDEX ERROR.
DEF      ERXEQ         ERROR DURING AN 'EXECUTE' OPERATION.
DEF      ERTERMAL      WRONG TERMINAL
DEF      ERFILEIO      FILE I/O ERR (IDENTIFIED BY R2).

```

```
SPACE 3
```

```

*
*  REFS TO PROCEDURE:
*
REF      ERRN          GEN. VECTOR -- ERROR NO. & LINE NO.
REF      ERRF          GEN. VECTOR -- ERROR FUN.NAME.
REF      ERRX          GEN. VECTOR -- HEX I/O ERR CODE.
REF      LOBRNCK       START ERR-CTRL BRANCH CHECKING.
REF      TIMODAY       TIME-OF-DAY.
REF      DATE          DATE.
REF      CPUTIME       CPU TIME.
REF      OVERTIME      OVERHEAD TIME.
REF      UNSYMS        UNUSED SYMBOLS.
REF      NUMUSERS      NUMBER OF USERS.
REF      SETWIDTH      SETS WIDTH.
REF      SETDIGIT      SETS DIGITS.
REF      SETORG        SETS ORIGIN.
REF      SETFUZZ       SETS FUZZ.
REF      TABSET        SETS TABS.
REF      DELAYER       DELAYS EXECUTION.
REF      DELTAGRF      DELTA-GRF =GRAPHIC SERVICES ROUTINE.
REF      DELTAFMT      DELTA FMT (SPECIAL OUTPUT FORMATTER)
REF      FILEOPS       INTRINSIC-OP USED BY FILE I/O SUB-
                      SYSTEMS.
*
REF      RETURNMX      RESUME AFTER: MONADIC
REF      RETURNNDX     DYADIC
REF      RETURNNI      NILADIC
REF      RETURNSX      SUBSCRIPTING
REF      RETURNAX      ASSIGNED INDEXING
REF      XEQEMPTY      NIL 'EXECUTE'
REF      QOUTRET       QUAD-0 (GRAPHICS) OUTPUT.
REF      QDBDONE       QUAD-0 (GRAPHICS) INPUT.
REF      QZOUTPUT      QUAD-0 (GRAPHICS) OUTPUT ROUTINE.
REF      QZINPUT       QUAD-0 (GRAPHICS) INPUT ROUTINE.
REF      ERRSET        ERROR HANDLER.
REF      DREF          DE-REFERENCER.

```

REF	MAYDREF	DE-REFS IF R4 IS NON-ZERO.	U05-0004
REF	SYSTEMERR	SYSTEM ERROR (UNIMPLEMENTED INTRINS)	
REF	ALOCBLK	ALLOCATES A DATA BLOCK.	
REF	ALOCHNW	ALLOCDS DATA BLK -- HDR + N WDS, EVEN	
REF	ALOCTRES	ALLOCDS A TEXT RESULT DATA BLK.	
REF	F2I	FLOATING-TO-INTEGGER CONVERTER.	

* REFS TO CONTEXT:

REF	LOGONTIM	LOG-ON TIME.
REF	TERMTYPE	TERMINAL TYPE.
REF	USERACCT	USER'S ACCOUNT.
REF	RANDOM	RANDOM VALUE.
REF	OPER	DBLWD TEMP FOR OPERATOR INFO.
REF	ON%OFF	ON-LINE VS BATCH FLAG (1 VS 0).
REF	BREAKFLG	BREAK FLAG (0 = CLEAR).
REF	XSEGBRK	EXECUTION SEGMENT BREAK TRIGGER.
REF	OFFSET	USED TO SET 'ERROR POSITION'.
REF	LBLOCK	PTS AT LOOP-CONTROL BLOCK.
REF	RTARG	PTS AT RT ARG'S DATA BLOCK.
REF	LFARG	PTS AT LF ARG'S DATA BLOCK.
REF	RESULT	PTS AT RESULT'S DATA BLOCK.
REF	RSTYPE	HOLDS TYPE -- USUALLY FOR RESULT.
REF	RSRANK	HOLDS RANK -- USUALLY FOR RESULT.
REF	RSSIZE	HOLDS SIZE -- USUALLY FOR RESULT.
REF	CONSTBUF	USED AS A TEMP STORAGE BLK.
REF	DYNBOUND	PTS AT HI BOUND OF DYNAMIC AREA.
REF	FREETOTL	# UNUSED WDS IN CURR. DYNAMIC AREA.
REF	TOPOSTAK	PTS AT TOP OF EXECUTION STACK.
REF	STATEPTR	PTS AT TOP STATE-ENTRY IN STACK.

SPACE 3

* STANDARD EQU'S

* REGISTERS

R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6
R7	EQU	7
R8	EQU	8
R9	EQU	9
R10	EQU	10
R11	EQU	11
R12	EQU	12
R13	EQU	13
R14	EQU	14
R15	EQU	15

* OTHER EQU'S

TYPETEXT	EQU	2	DATA BLK TYPE -- TEXT.
TYPELOGL	EQU	1	LOGICAL.
TYPEINTG	EQU	3	-- INTEGER.
TYPEXSEQ	EQU	5	-- INDEX SEQUENCE.
CATQ	EQU	7	EXEC.STACK CATEGORY FOR EVAL-INPUT.
DYINT	EQU	14	DYADIC INTRINSIC TYPE.
NYINT	EQU	16	NILADIC INTRINSIC TYPE.
FLINFLG	EQU	X'10000'	FUNCTION-LINE FLAG BIT (S-CATEGORY).

* DOUBLEWORD CONSTANTS

DELAYRNG	DATA	1,86400	DELAY RANGE -- 1 SECOND TO 1 DAY.
ONE	EQU	DELAYRNG	= 1
INTRANGE	DATA	DYINT,NYINT	INTRINSIC FUNCTION TYPE RANGE.

* OTHER CONSTANTS

24HRS	DATA	5184000	# 60THS OF A SECOND IN ONE DAY.
PAGE			
PAGE			
OPBREAK	BAL,R2	SETERR	-1 @ BREAK
ERLSERR	BAL,R2	SETERR	0 @ LINE-SCAN ERR ('SYNTAX ERR')
ERSYN	BAL,R2	SETERR	1 @ SYNTAX ERROR
ERUND	BAL,R2	SETERR	2 @ UNDEFINED
ERNOR	BAL,R2	SETERR	3 @ NO RESULT
ERIO	BAL,R2	SETERR	4 @ I/O ERROR
ERDOMAIN	BAL,R2	SETERR	5 @ DOMAIN ERROR
ERRANK	BAL,R2	SETERR	6 @ RANK ERROR

ERLENGTH	BAL,R2	SETERR	7	@ LENGTH ERROR
ERWSFUL	BAL,R2	SETERR	8	@ WS FULL
ERSING	BAL,R2	SETERR	9	@ SINGULAR MATRIX
ERFORMAT	BAL,R2	SETERR	10	@ FORMAT ERROR
ERINDEX	BAL,R2	SETERR	11	@ INDEX ERROR
ERXEQ	BAL,R2	SETERR	12	@ ERROR DURING AN EXECUTE-OPERATION.

ERFILEIO	EQU	SETERR2	R2 SET	FILE I/O ERR (R2 SAYS WHICH ONE).
ERTERMAL	EQU	SETERR2	R2 SET	WRONG TERMINAL
SPACE				
SETERR	AI,R2	-OPBREAK-2		SET ERROR I.D.
SETERR2	LI,R3	0		CLEAR EXECUTION-SEGMENT BREAK
	STW,R3	XSEGBRK		TRIGGER.
	INT,R3	OPER+1		SET BYTE OFFSET IN CURR. CODESTRING
	STW,R3	OFFSET		AT WHICH ERROR WAS DETERMINED.
	B	ERRSET		DO ERROR HANDLING.

PAGE

*
* I N T E R F A C E S T O R E S U M E E X E C U T I O N
*

* A F T E R S U C C E S S F U L O P E R A T I O N . . .
*

XEQNIL	B	XEQEMPTY		ASSUME EMPTY RESULT FOR AN 'EXECUTE'
NIRETURN	LI,R4	0		NILADIC INTRINSIC COMPLETED.
	STW,R4	XSEGBRK		(CLR EXEC.SEGMENT BREAK TRIGGER)
	LW,R3	OPER+1		(RESTORE CODESTRING OFFSET)
	B	RETURNNI		(RESUME -- R4 = 0)
SXRETURN	LI,R4	0		SUBSCRIPTING COMPLETED.
	STW,R4	XSEGBRK		(CLR EXEC.SEGMENT BREAK TRIGGER)
	XW,R4	LBLOCK		
	BAL,R7	MAYDREF		(DE-REF. ANY LOOP CONTROL BLOCK) U05-0006
	LW,R3	OPER+1		(RESTORE CODESTRING OFFSET)
	B	RETURNSX		(RESUME)
AXRETURN	LI,R4	0		ASSIGNED-INDEXING COMPLETED.
	STW,R4	XSEGBRK		(SEE COMMENTS FOR SXRETURN)
	XW,R4	LBLOCK		
	BAL,R7	MAYDREF		U05-0008
	LW,R3	OPER+1		
	B	RETURNAX		
	LI,R4	0	@ MON.OP (USING LBLOCK) COMPLETED.	
	STW,R4	XSEGBRK	@ (SEE COMMENTS FOR SXRETURN)	
	XW,R4	LBLOCK	@	
	BAL,R7	DREF	@	
MXRETURN	LI,R2	0	@ MONADIC OPERATION COMPLETED.	
	STW,R2	XSEGBRK	(CLR EXEC.SEGMENT BREAK TRIGGER)	
	B	RETURNMX	(RESUME WITH R2 = 0)	
	LI,R4	0	@ DY.OP (USING LBLOCK) COMPLETED.	
	STW,R4	XSEGBRK	@ (SEE COMMENTS FOR SXRETURN)	
	XW,R4	LBLOCK	@	
	BAL,R7	DREF	@	
DXRETURN	LI,R2	0	@ DYADIC OPERATION COMPLETED.	
	STW,R2	XSEGBRK	(CLR EXEC.SEGMENT BREAK TRIGGER)	
	B	RETURNDX	(RESUME WITH R2 = 0)	

PAGE

* BCBRANCH -- CLEARS BREAK FLAG & DOES ERROR-CONTROL BRANCH, ON ENTRY	*
* R12 = MINUS THE CONTENTS OF BREAKFLG.	*
* ECBRANCH -- DOES ERROR-CONTROL BRANCH.	*
* REGS: R1 (ENTRY) PTS AT PENDENT FUNCTION STATE THAT IS	*
* TAKING ERROR CONTROL (TOP OF STACK).	*
* R12 (EXIT) = FLINFLG -- AS IF BRANCH IS FROM INSIDE	*
* THAT FUNCTION: STOP WILL BE	*
* HONORED IF ENCOUNTERED.	*
* OTHER REGS MAY BE REGARDED VOLATILE.	*

BCBRANCH	AWM,R12	BREAKFLG	CLEAR THE BREAK.
ECBRANCH	LI,R12	FLINFLG	SET R12 AS IF FUNCTION DID BRANCH.
	B	LOBRNCK	START BRANCH CK FOR ERR-CTRL FUNC.

PAGE

*
* G R A P H I C S I N T E R F A C I N G . . .
*

QZOUT	B	QZOUTPUT	START GRAFIX OUTPUT.
QZOUTRET	LI,R4	0	RESUME AFTER GRAFIX OUTPUT.
	XW,R4	RTARG	(DEREF. RT. ARG. PTR)
	BAL,R7	DREF	
	LW,R3	OPER+1	RESTORE CS OFFSET TO THE QUAD-0.

```

    B          QOUTRET          RESUME CODESTRING EXECUTION.
SPACE 2
QZIN          B          QZINPUT          START GRAFIX INPUT.
QZINRET      RES          0          RESUME AFTER GRAFIX INPUT.
            LW,R1        TOPOSTAK        PT AT TOP OF EXEC.STACK.
            LW,R3        OPER+1         RESTORE CS OFFSET TO THE QUAD-0.
            B          QDBDONE         RESUME CODESTR.XEQ WITH RESULT SET.
PAGE
*****
*
* IV1 -- GETS A SINGLE INTEGER VALUE FROM RT ARG'S DATA BLOCK.
*      DOMAIN ERROR IF RT ARG: IS TEXT OR LIST TYPE
*                                IS REAL AND VALUE IS NOT WITHIN FUZZ
*                                OF AN INTEGER.
*      LENGTH ERROR IF RT ARG HAS MORE THAN 1 ELEMENT OR IS EMPTY.
*
*      REGS:  R14 -- LINK, EXIT VIA *R14.
*             R2  -- (EXIT) CONTAINS RT ARG'S DATA BLK TYPE.
*             R7  -- (EXIT) CONTAINS THE INTEGER VALUE (ALSO STORED
*                   IN 'CONSTBUF')
*             R1,R4,R5,R6,R8 ARE VOLATILE (SEE ALSO 'F2I').
*
IV1          LW,R4        RTARG          PT AT WD AFTER RT ARG'S REF-COUNT.
            AI,R4        2
            LI,R1        1
            LB,R8        *RTARG,R1      GET RT ARG'S RANK...
            BEZ          IV10K          SCALAR.
IV1Q         AI,R4        1          ARRAY, VERIFY LENGTHS ARE 1...
            CW,R1        -1,R4
            BNE          ERLLENGTH     NOT -- LENGTH ERROR.
            BDR,R8        IV1Q
IV10K        LW,R7        0,R4        GET (SUPPOSED) VALUE WD.
            LB,R2        *RTARG        GET RT ARG'S TYPE.
IV1TYPE      B          IV1TYPE,R2    @ VECTOR ON TYPE: 0 -- IMPOSSIBLE.
            B          IV1LOGL        @ 1 -- LOGICAL.
            B          ERDOMAIN       @ 2 -- TEXT.
            B          IV1INTG        @ 3 -- INTEGER.
            B          IV1REAL        @ 4 -- REAL.
            B          IV1XSEQ        @ 5 -- INDEX SEQUENCE.
            B          ERDOMAIN       @ 6 -- LIST.
IV1LOGL      SLS,R7        -31        GET THE LOGICAL VALUE.
            B          IV1INTG
IV1XSEQ      AW,R7        1,R4        ADD COEFFICIENT TO BIAS.
            B          IV1INTG
IV1REAL      AI,R4        1          PT AT NEXT WD (ODD OR EVEN OK).
            SLS,R4        -1          PT AT REAL DBLWD VALUE.
            LD,R6        0,R4        GET REAL VALUE.
            BAL,R5        F2I        CONVERT TO INTEGER IF POSSIBLE...
            B          ERDOMAIN       NO LUCK -- DOMAIN ERROR.
IV1INTG      STW,R7        CONSTBUF    OK, SAVE INTEGER VALUE.
            B          *R14          EXIT.
PAGE
SPACE
MIBEAM      BAL,R14      IV1          GET SINGLE INTEGER ELEMENT.          U05-0010
            AI,R7        -19         OFFSET BY MINIMUM ACCEPTED NO.    U05-0011
            BLZ          NEGIBEAM     OOPS, TRY NEGATIVE I-BEAM.
            CI,R7        29-19       VERIFY NOT TOO BIG...          U05-0013
            BLE          IGO,R7       OK, TAKE CORRESPONDING BRANCH.    U05-0014
            B          ERDOMAIN       OOPS -- DOMAIN ERROR.
IGO          B          I19          @ SESSION TIME.
            B          I20          @ TIME OF DAY.
            B          I21          @ CPU TIME.
            B          I22          @ BYTES OF WORKSPACE REMAINING.
            B          I23          @ NUMBER OF USERS.
            B          I24          @ LOG-ON TIME.
            B          I25          @ DATE
            B          I26          @ TOP LINE NO.
            B          I27          @ VECTOR OF LINE NOS.
            B          I28          @ TERMINAL TYPE.
            B          I29          @ USER'S ACCOUNT.
*           B          I29          @
I29          LI,R11      6          @ ALLOCATE A 6-WD DATA BLK.
            BAL,R7        ALOCBLK
            STW,R4        RESULT      SAVE PTR TO RESULT DATA BLK.
            LI,R11      (TYPETEXT**8)+1 TYPE = TEXT RANK = 1 (VECTOR).
            STH,R11      *RESULT
            LI,R11      8          LENGTH = 8.
            LW,R12      USERACCT     USER'S ACCOUNT (8 CHARS, POSSIBLY
            LW,R13      USERACCT+1   INCLUDING TRAILING BLANKS).

```

	LCI	3		
	STM, R11	2, R4		
I19	B	MXRETURN		FILL IN LENGTH AND ACCOUNT.
	BAL, R14	TIMODAY		RETURN FROM MONADIC-OP EXECUTION.
	SW, R11	LOGONTIM		GET TIME-OF-DAY (R11) 60THS OF SEC.
	BGEZ	SCLRRES		SUBTRACT LOG-ON-TIME.
	AW, R11	24HRS		= SESSION TIME...
	B	SCLRRES		OOPS, CORRECT FOR MIDNITE CROSSED.
I20	LI, R14	SCLRRES		ITS A SCALAR RESULT.
	B	TIMODAY		SCALAR RESULT IN R11 IS
I21	LI, R14	SCLRRES		TIME-OF-DAY IN 60THS OF SEC.
	B	CPUTIME		SCALAR RESULT IN R11 IS
I22	LW, R11	TOPOSTAK		CPU TIME IN 60THS OF SEC.
	SW, R11	DYNBOUND		
	AW, R11	FREETOTL		= # WDS BETWEEN STACK & END OF DYN.
	SLS, R11	2		+ UNUSED WDS OF DYNAMIC.
	AI, R11	-64		IN BYTES.
	BGEZ	SCLRRES		WITH 64-BYTE CUSHION.
	LI, R11	0		SCALAR RESULT.
	B	SCLRRES		USE 0 IF PUSH INTO CUSH.
I23	LI, R14	SCLRRES		
	B	NUMUSERS		SCALAR RESULT IN R11 IS
I24	LW, R11	LOGONTIM		NO.OF USERS IN 100THS OF A PERSON.
	B	SCLRRES		= TIME OF DAY AT LOG-ON IN 60THS/SEC
I25	LI, R14	SCLRRES		FOR THE SCALAR RESULT.
	B	DATE		SCALAR RESULT IN R11 IS 'MMDDYY' AS
I26	LI, R7	SCLRRES-1		BASE 10 VERSION OF THE DATE.
SVSTART	LW, R5	STATEPTR		(SET EXIT FROM SVSTART).
	LI, R8	0		PT AT TOP STATE-ENTRY.
	LI, R9	X'7FFF'		SET FOR SELECTIVE LOADS.
STATE	LS, R8	0, R5		
	BEZ	0, R7		GET 'NEXT' FIELD OF STATE-ENTRY.
	LB, R6	*R5		NONE -- TAKE FINAL-ENTRY EXIT.
	CI, R6	CATQ		GET CATEGORY.
	BNE	STATEF		
	LI, R11	0		FUNCTION-CATEGORY.
	B	1, R7		QUAD-CATEGORY USE 0.
STATEF	LW, R11	1, R5		TAKE NORMAL EXIT.
	BNEZ	STATEFL		GET LINE NO. & FUNC.PTR, IF ANY...
	LI, R11	-1		OK.
STATEFL	SAS, R11	-17		DAMAGED, USE -1.
	B	1, R7		= LINE NO. ONLY.
	LI, R11	0		TAKE NORMAL EXIT.
SCLRRES	LW, R4	RTARG	@	USE 0 FOR FINAL-ENTRY ON I26.
	LW, R5	1, R4	@	PT AT RIGHT ARGUMENT DATA BLK.
	AI, R5	-1		GET ITS REF-COUNT.
	BNEZ	SCLRINTG		IF NOT EXACTLY ONE,
	LH, R2	*RTARG		ALLOCATE A FRESH BLOCK FOR RESULT.
	AI, R2	-(TYPEINTG**8)		IF EXACTLY ONE, IT'S A TEMP.
	BEZ	RESRTARG		SO, IF SCALAR INTEGER ALREADY,
SCLRINTG	STW, R11	CONSTBUF		USE RT ARG'S BLOCK FOR RESULT.
SCLRINTA	LI, R11	4		SAVE INTEGER VALUE FOR RESULT.
	BAL, R7	ALOCBLK		ALLOCATE A 4-WD DATA BLK.
	LI, R11	TYPEINTG		
	STB, R11	*R4		INTEGER TYPE.
	LW, R11	CONSTBUF		RECOVER THE VALUE.
	B	SETVALUE		
RESRTARG	MTW, 1	1, R4		BUMP REF-COUNT.
SETVALUE	STW, R4	RESULT		SAVE PTR TO RESULT DATA BLK.
	STW, R11	2, R4		SET ITS VALUE.
	B	MXRETURN		
I27	LI, R11	0		COUNTS NO.OF STATE-ENTRIES TILL FIN.
	LW, R5	STATEPTR		PT AT TOP STATE-ENTRY.
	LI, R8	0		SET FOR SELECTIVE LOADS.
	LI, R9	X'7FFF'		
	B	I27Q		START QUERY.
I27N	AW, R5	R8		PT AT NEXT STATE-ENTRY.
	AI, R11	1		COUNT THE PREVIOUS ONE.
I27Q	LS, R8	0, R5		IS THIS THE FINAL ENTRY...
	BNEZ	I27N		NO.
	STW, R11	CONSTBUF		YES, SAVE STATE-ENTRY COUNT.
	AI, R11	1		ADD 1 FOR THE LENGTH OF THE VECTOR.
	BAL, R7	ALOCHNW		ALLOC THAT MANY WDS + HDR & EVEN NO.
	STW, R4	RESULT		SAVE PTR TO RESULT DATA BLK.
	LI, R7	(TYPEINTG**8)+1		TYPE = INTEGER RANK = 1 (VECTOR).
	STH, R7	*RESULT		
	LW, R11	CONSTBUF		GET NO.OF STATE ENTRIES.
	STW, R11	2, R4		THAT'S THE LENGTH OF THE VECTOR.

	AI, R4	2	
	BAL, R7	SVSTART	@ START LOOKING AT STATE-VECTOR AGAIN.
	B	MXRETURN	@ FINAL ENTRY.
	AI, R4	1	@ OTHER.
	STW, R11	0, R4	FILL IN ITS LINE NO. INDICATION.
	AW, R5	R8	PT AT NEXT STATE-ENTRY.
	B	STATE	FIND ITS CATEGORY ETC. LOOPING.
I28	LW, R11	TERMTYPE	GET TERMINAL TYPE NUMBER FOR THE
	B	SCLRRES	SCALAR RESULT.
PAGE			
NEGIBEAM	AI, R7	19	RESTORE OFFSET.
	LCW, R7	R7	GET (HOPEFULLY) PLUS VALUE...
	BLEZ	ERDOMAIN	OOPS -- DOMAIN ERROR.
	CI, R7	NEGIBBND	VERIFY IN RANGE...
	BLE	NIGO, R7	OK, TAKE CORRESPONDING BRANCH.
NIGO	B	ERDOMAIN	@ OOPS -- DOMAIN ERROR.
	B	NI1	@ OVERHEAD TIME.
	B	NI2	@ NO.OF UNUSED SYMBOLS.
	B	NI3	@ ON-LINE VS BATCH.
SPACE 2			
NEGIBBND	EQU	%-NIGO-1	= NO.OF NEGATIVE I-BEAM BRANCHES.
SPACE 2			
NI1	LI, R14	SCLRRES	SCALAR RESULT IN R11 IS
	B	OVERTIME	OVERHEAD TIME IN 60THS OF SEC.
NI2	LI, R14	SCLRRES	SCALAR RESULT IN R11 IS
	B	UNSYMS	NO.OF UNUSED SYMBOLS.
NI3	LW, R11	ON%OFF	= 1 IF ON-LINE OR 0 IF BATCH.
	B	SCLRRES	SCALAR, INTEGER RESULT.
MTBAR	LB, R11	*RTARG	GET RT ARG'S TYPE &
	B	SCLRINTG	USE THAT VALUE FOR SCALAR, INTEGER
*			RESULT.
DTBAR	LI, R5	TYPEINTG	VERIFY LEFT ARG IS INTEGER TYPE...
	CB, R5	*LFARG	
	BNE	ERDOMAIN	NO -- DOMAIN ERROR.
	LI, R8	X'FF'	VERIFY LEFT ARG IS SCALAR...
	CH, R8	*LFARG	
	BANZ	ERRANK	NO -- RANK ERROR.
	LW, R4	LFARG	PT AT LEFT ARG.
	LW, R6	2, R4	GET ITS VALUE.
	CI, R6	2	IS IT 2...
	BNE	INTRIGUE	NO, GUESS ITS FOR INTRINSIC.
	STW, R6	RSTYPE	YES, SET TYPE OF RESULT.
	CB, R5	*RTARG	VERIFY RT ARG IS INTEGER TYPE...
	BE	DTBARI	OK.
	LI, R5	TYPEXSEQ	NO, TRY FOR INDEX SEQUENCE...
	CB, R5	*RTARG	
	BNE	DTBART	NOPE, LOGICAL PERHAPS.
	LW, R4	RTARG	OK, PT AT ITS DATA BLOCK.
	LW, R11	2, R4	GET ITS LENGTH VALUE.
	STW, R11	RSSIZE	THAT IS THE NO.OF ELEMENTS.
	LI, R5	1	
	STW, R5	RSRANK	RANK OF RESULT IS 1 (VECTOR).
	BAL, R14	ALOCTRES	ALLOC A TEXT TYPE RESULT DATA BLK.
	LW, R5	RTARG	PT AT RT ARG AGAIN.
	LW, R14	RSSIZE	GET NO.OF ELEMENTS AGAIN.
	STW, R14	2, R4	THAT IS LENGTH OF NEW TEXT VECTOR...
	BEZ	DXRETURN	ZERO.
	LW, R9	3, R5	GET INDEX SEQ'S BIAS VALUE.
	AI, R4	3	PT AT 1ST VALUE WD OF RESULT.
	SLS, R4	2	USE BYTE ADDRESSING.
DTBARX	AW, R9	4, R5	ADD INDEX SEQ'S COEFFICIENT VALUE.
	STB, R9	0, R4	PUT BYTE IN RESULT.
	AI, R4	1	PT AT NEXT RESULT BYTE POSITION.
	BDR, R14	DTBARX	LOOP TILL LAST VALUE.
	LW, R11	3, R5	RE-CALC. FIRST VALUE.
	AW, R11	4, R5	
	OR, R11	R9	MERGE THE FIRST & LAST VALUES.
	B	DTBARQ	CK THAT RANGE.
DTBART	LI, R5	TYPELOGL	IS RT ARG LOGICAL TYPE...
	CB, R5	*RTARG	
	BNE	ERDOMAIN	NO -- DOMAIN ERROR.
DTBARI	LI, R5	1	
	LB, R4	*RTARG, R5	GET RANK OF RT ARG AND SAVE IT.
	STW, R4	RSRANK	
	BEZ	DTBARS	SCALAR.
	LW, R14	RSRANK	ARRAY, CALC. NO.OF ELEMENTS.
	AW, R4	RTARG	
DTBARN	MW, R5	1, R4	

	AI, R4	-1	
	BDR, R14	DTBARN	
DTBARS	STW, R5	RSSIZE	SAVE NO.OF ELEMENTS.
	LW, R11	RSSIZE	ALLOCATE A TEXT TYPE DATA BLK RESULT
	BAL, R14	ALOCTRES	FOR THAT MANY ELEMENTS, SAME RANK.
	LW, R5	RTARG	PT AT WD AFTER RT ARG'S REF-COUNT.
	AI, R5	2	
	AI, R4	2	LIKEWISE FOR RESULT.
	LW, R14	RSRANK	GET RANK...
	BEZ	DTBARE	SCALAR, WE PT AT THE VALUE WD.
DTBARF	LW, R9	0, R5	ARRAY, COPY LENGTHS OF RT ARG
	STW, R9	0, R4	INTO LENGTHS OF RESULT.
	AI, R4	1	
	AI, R5	1	
	BDR, R14	DTBARF	FINISH POINTING AT 1ST VALUE WD.
DTBARE	LW, R14	RSSIZE	GET NO.OF ELEMENTS...
	BEZ	DXRETURN	NONE.
	SLS, R4	2	SOME, USE BYTE ADDRESSING ON RES.
	LI, R11	0	
	LI, R9	TYPEOGL	IS RT ARG LOGICAL... (TYPE = 1)
	CB, R9	*RTARG	
	BNE	DTBARK	NO, INTEGER.
DTBARLW	LW, R10	0, R5	YES, LOAD LOGL VALUE WD.
	AI, R5	1	PT AT NEXT VALUE WD, IF ANY.
	LI, R8	32	32 LOGL VALUES PER WD.
DTBARLB	AI, R10	0	TEST A LOGL VALUE...
	BGEZ	DTBARLZ	ZERO.
	STB, R9	0, R4	ONE, STORE BYTE = 1.
DTBARLU	AI, R4	1	PT AT NEXT BYTE.
	BDR, R14	DTBARLS	LOOP TILL
	B	DXRETURN	DONE WITH LOGL ELEMENTS.
DTBARLZ	STB, R11	0, R4	STORE BYTE = 0.
	B	DTBARLU	
DTBARLS	SLS, R10	1	GET NEXT LOGL VALUE BIT,
	BDR, R8	DTBARLB	IF ANY REMAIN IN THIS VALUE WD.
	B	DTBARLW	GET NEXT WD.
DTBARK	LW, R9	0, R5	MOVE EACH ELEMENT FROM RT ARG
	OR, R11	0, R5	(ACCUMULATING BITS FOR VALIDATION)
	STB, R9	0, R4	INTO RESULT.
	AI, R4	1	
	AI, R5	1	
	BDR, R14	DTBARK	
DTBARQ	CI, R11	X'FFF00'	VALIDATE NO ELEMENT EXCEEDED 8 BITS.
	BAZ	DXRETURN	OK.
	B	ERDOMAIN	OOPS -- DOMAIN ERROR.
ITEST	CI, R2	#DINTRS	@ NO.OF DYADIC INTRINSICS.
	CI, R2	#MINTRS	@ NO.OF MONADIC INTRINSICS.
NINTRIN	CI, R2	#NINTRS	@ NO.OF NILADIC INTRINSICS.
	BL	NYV, R2	OK, VECTOR ON THE INTRINSIC NO.
	BAL, R15	SYSTERR	TOO BIG
MINTRIN	CI, R6	#MINTRS	NO.OF MONADIC INTRINSICS.
	BL	MYV, R6	OK, VECTOR ON THE INTRINSIC NO.
	BAL, R15	SYSTERR	TOO BIG
DINTRIN	CI, R6	#DINTRS	NO.OF DYADIC INTRINSICS.
	BL	DYV, R6	OK, VECTOR ON THE INTRINSIC NO.
	BAL, R15	SYSTERR	TOO BIG
INTRIGUE	CLM, R6	INTRANGE	VERIFY TYPE VALUE IS INTRINSIC...
	BCS, 9	ERDOMAIN	NO -- DOMAIN ERROR.
	CH, R8	*RTARG	OK, VERIFY SCALAR RT ARG...
	BANZ	ERRANK	NOPE -- RANK ERROR.
	LW, R4	RTARG	YEP, PT AT RT ARG DATA BLK.
	LW, R2	2, R4	GET RT ARG VALUE WD.
	CB, R5	*RTARG	IS RT ARG AN INTEGER TYPE...
	BE	INTRIGUI	YES.
	SLS, R2	-31	NO, PROBABLY LOGICAL TYPE, GET BIT
	LI, R5	TYPEOGL	VERIFY LOGICAL TYPE...
	CB, R5	*RTARG	
	BNE	ERDOMAIN	OOPS -- DOMAIN ERROR.
INTRIGUI	AI, R2	0	VERIFY POSITIVE VALUE.
	BLZ	ERDOMAIN	NEGATIVE -- DOMAIN ERROR.
	EXU	ITEST-DYINT, R6	VERIFY NOT TOO BIG...
	BGE	ERDOMAIN	IT IS -- DOMAIN ERROR.
	LI, R11	2	OK, ALLOCATE A 2-WD DATA BLK.
	BAL, R7	ALOCBLK	
	STW, R4	RESULT	SAVE PTR TO RESULT DATA BLK.
	STH, R2	*RESULT	PLUG IN THE INTRINSIC NO.
	STB, R6	*RESULT	PLUG IN THE INTRINSIC TYPE.
	B	DXRETURN	RETURN FROM DYADIC-OP EXECUTION,

DYV	B	DELTAfmt	@	DYADIC INTRINSIC NO. 0	
	BDR, R6	%+1	@	(FILEOPS--SET R6=0)	1
	B	FILEOPS	@	(APL ERR-HANDLING VERSION)	2
	B	DELTAgrf	@		3
	BAL, R15	SYSTERR	@		4
	BAL, R15	SYSTERR	@		5
	BAL, R15	SYSTERR	@		6
	BAL, R15	SYSTERR	@		7
#DINTRS	EQU	%-DYV	@	# DYAD.INTRINSICS.	
MYV	B	IORIGIN	@	MONADIC INTRINSIC NO. 0	
	B	IWIDTH	@		1
	B	IDIGITS	@		2
	B	ITABS	@		3
	B	ISSETLINK	@		4
	B	ISSETFUZZ	@		5
	B	IDELAY	@		6
	BAL, R15	SYSTERR	@		7
	BAL, R15	SYSTERR	@		8
	BAL, R15	SYSTERR	@		9
	BAL, R15	SYSTERR	@		10
	BAL, R15	SYSTERR	@		11
#MINTRS	EQU	%-MYV	@	# MON.INTRINSICS.	
NYV	B	ERRN	@	NILADIC INTRINSIC NO. 0	
	B	ERRF	@		1
	B	ERRX	@		2
	BAL, R15	SYSTERR	@		3
#NINTRS	EQU	%-NYV	@		
IORIGIN	BAL, R14	IV1	@	GET THE ELEMENT.	
	LI, R5	EZQ-1	@	GO TO 'EZQ' AFTER	
	B	SETORG	@	SETTING THE ORIGIN.	
	B	ERDOMAIN	@	(ERR RETURN FROM THE SET-ROUTINES)	
EZQ	LW, R11	R7	@	PUT OLD VALUE (RESULT) IN R11.	U05-0024
	B	SCLRRES	@	PRODUCE SCALAR, INTEGER RESULT.	U05-0025
IWIDTH	BAL, R14	IV1		GET THE ELEMENT.	
	LI, R5	EZQ-1		GO TO 'EZQ' AFTER	
	B	SETWIDTH		SETTING THE WIDTH.	
IDIGITS	BAL, R14	IV1		GET THE ELEMENT.	
	LI, R5	EZQ-1		GO TO 'EZQ' AFTER	
	B	SETDIGIT		SETTING DIGITS.	
ISSETFUZZ	BAL, R14	IV1		GET THE ELEMENT.	
	LI, R5	EZQ-1		GO TO 'EZQ' AFTER	
	B	SETFUZZ		SETTING FUZZ.	
ISSETLINK	BAL, R14	IV1		GET THE ELEMENT.	
	OR, R7	ONE		INSURE THAT WE USE ODD LINK.	
	BLZ	ERDOMAIN		NEGATIVE -- DOMAIN ERROR.	
	XW, R7	RANDOM		SET NEW, GET OLD LINK.	
	B	EZQ			
IDELAY	BAL, R14	IV1		GET THE ELEMENT.	
	CLM, R7	DELAYRNG		VERIFY REASONABLE DELAY VALUE.	
	BCS, 9	ERDOMAIN		NOT -- DOMAIN ERROR.	
	BAL, R5	DElayer		OK -- DELAY EXECUTION.....	
	LI, R11	0		(RESUME EXECUTION)	
	STW, R11	CONSTBUF		SET UP TO FAKE OUT THE TAB RESULT	
	B	IMEMPTY		GENERATOR SO THAT AN EMPTY VECTOR	
*				RESULTS.	
ITABS	LB, R2	*RTARG		GET RT ARG'S TYPE...	
	AI, R2	-TYPEINTG			
	BEZ	ITABSI		INTEGER.	
	AI, R2	TYPEINTG-TYPEXSEQ			
	BNEZ	ITABSQ		LOGICAL PERHAPS.	
	LW, R4	RTARG		INDEX SEQUENCE.	
	LW, R6	2, R4		GET ITS LENGTH.	
	CI, R6	16			
	BG	ERLENGTH		TOO LONG -- LENGTH ERROR.	
	LW, R7	3, R4		GET ITS BIAS VALUE.	
	LI, R5	CONSTBUF+17		PT AT BUFFER TO SAVE NEW TAB VECTOR.	
ITABSX	AW, R7	4, R4		ADD ITS COEFFICIENT VALUE.	
	STW, R7	0, R5		PUT ANSWER IN BUFFER.	
	AI, R5	1		PT AT NEXT WD IN BUFFER.	
	BDR, R6	ITABSX			
	LW, R6	2, R4		GET LENGTH AGAIN.	
	LI, R4	CONSTBUF+17		PT AT 1ST VALUE IN BUFFER.	
	B	ITABSET			
ITABSQ	LI, R2	TYPELOGL**8		VERIFY LOGICAL SCALAR.	
	CH, R2	*RTARG			
	BNE	ERDOMAIN		OOPS -- DOMAIN ERROR.	
ITABSI	LW, R4	RTARG		PT AT WD AFTER RT ARG'S REF-COUNT WD	
	AI, R4	2			

	LI, R6	1	INIT LENGTH = 1 (ALSO BYTE OFFSET).
	LB, R8	*RTARG, R6	GET RANK OF RT ARG.
	BEZ	ITABSET	SCALAR.
	AI, R8	-1	
	BNEZ	ERRANK	NON-VECTOR ARRAY -- RANK ERROR.
	LW, R6	0, R4	VECTOR, GET ITS LENGTH.
	AI, R4	1	PT AT 1ST VALUE WD.
ITABSET	BAL, R2	TABSET	SET TABS IF OK.
	B	ERDOMAIN	OOPS -- DOMAIN ERROR.
	LW, R11	CONSTBUF	OK, GET NO.OF OLD TAB VALUES.
IMEMPTY	AI, R11	1	ALLOCATE VECTOR OF THAT LENGTH
	BAL, R7	ALOCHNW	PLUS 2-WD HDR & EVEN SIZE.
	STW, R4	RESULT	SAVE PTR TO RESULT DATA BLK.
	LI, R11	(TYPEINTG**8)+1	TYPE = INTEGER RANK = 1.
	STH, R11	*RESULT	
	LW, R6	CONSTBUF	GET NO.OF OLD TAB VALUES.
	STW, R6	2, R4	SET LENGTH OF VECTOR RESULT.
	AI, R4	2	PT AT LENGTH WD.
ITABSZ	LW, R7	CONSTBUF, R6	FILL IN VALUES
	STW, R7	*R4, R6	FROM LAST
	BDR, R6	ITABSZ	TO FIRST.
	B	MXRETURN	

PAGE

SPACE 2			
Z	SET	%-INTRINS@	SIZE OF INTRINS IN HEX.
SPACE			
Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536	SIZE IN DECIMAL.
SPACE 3			
	END		


```

TITLE      'WMAQ-B00,08/22/73,DWG702985'
SYSTEM     SIG7F
CSECT      1
PCC        0
WMAQ@      RES      0  CONTROL CARDS NOT PRINTED.
              ORIGIN OF WORKSPACE MGMT & ACQUISITION MODULE.
*
*  REF'S  AND  DEF'S
*
DEF      WMAQ@      = START OF WMAQ MODULE.
DEF      ALOCRES     ALLOCATES A 'RESULT' DATA BLK.
DEF      ALOCNONX    NON-EXEC. ALLOCS., CALLS 'ALOCHNW'.
DEF      ALOCHNW     ALLOCS. N WDS + 2-WD HDR + 1 IF ODD.
DEF      ALOCTRES    ALLOCATES A TEXT 'RESULT' DATA BLK.
DEF      ALOCBLK     ALLOCS. DATA BLK, SETS SIZE & REF=1.
DEF      GIVEBACK    GIVE TAIL OF D.B. BACK TO FREE TBL.
DEF      CKVDB       CK STRUCTURE OF VAR. DATA BLK.
DEF      WSCHEK      CK WS (DIAG. TOOL)
DEF      WSCKDSPL    CK WS STRUCT., DISPLACING DB PTRS
*
*                      BY THE DYNAMIC OFFSET FOR THE WS
*                      WHEN SAVED VS. WHEN LOADED.
DEF      CTEST       TRIES TO GET MORE COMMON.
DEF      SICLR       CLEAR STATE-INDICATOR TO GO-STATE.
DEF      SICLR%      (ALTERNATE ENTRY TO SICLR).
DEF      MAYDREF     DE-REFS DATA BLK IF 1 IS POINTED TO.
DEF      DREF        DE-REFERENCES DATA BLK PT'ED TO.
DEF      GARB COLL   GARBAGE COLLECTOR.
DEF      ACQNAME     ACQUIRES A NAME, PUTS NEW IN SYM TBL
DEF      HASHINC     HASH INCREMENT (SEE COPY CMD 'HASH')
DEF      FINDNAME    FINDS NAME IF IN SYMBOL TABLE
DEF      IN2CODE     INTERNAL CHAR'S CODE BYTE TABLE.
DEF      ACQNXCC     ACQ NEXT INTERNAL CHAR & ITS CODE.
DEF      ACQCC       ACQ CURRENT CHAR & ITS CODE.
DEF      ACQCODE     ACQ CODE OF CURRENT CHAR.
DEF      ACQNXNB     ACQ NEXT NON-BLANK & ITS CODE.
DEF      ACQNB       ACQ NON-BLANK & ITS CODE.
DEF      ACQIT       ACQ NAME OR NUMERIC ITEM.
*
REF      GETDYN      GETS MORE DYNAMIC PAGES.
REF      GETCOM      GETS 1 MORE PAGE OF COMMON.
REF      ERWSFUL     ERROR -- WORKSPACE FULL.
REF      SYSTERR     ERROR -- SYSTEM ERROR.
*
REFS TO CONTEXT:
REF      DYNBOUND    UPPER BOUND FOR DYNAMIC REGION.
REF      STKLIMIT    CURRENT LIMIT FOR EXECUTION STACK.
REF      LOCNEED     NEW LOC. NEEDED WHEN OVER STACK LMT
REF      CORLEFT     ZERO IF NO MORE PAGES AVAILABLE.
REF      NEWBOUND    NEW UPPER BOUND NEEDED FOR DYNAMIC.
REF      SYMT        PTS AT 1ST WD OF SYMBOL TABLE.
REF      SYMTSIZE    NO.OF ENTRIES FOR SYMBOL TABLE.
REF      NSYMTWDS    NO.OF WORDS IN SYMBOL TABLE.
REF      NAMEBUF     BUFFER TO HOLD ACQUIRED NAME.
REF      NAMEWDS     TEMP TO HOLD # WDS CONTAINING A NAME
REF      NAMLIMIT    = MAX # CHARS USED FOR A NAME.
REF      NAMEWDSZ    = MAX # WDS USED TO CONTAIN A NAME.
REF      FREETOTL    TOTAL SPACE CONTAINED IN FREE TABLE.
REF      FREETBL     FREE TABLE (2-WD ENTRIES--LOC & AMT)
REF      MAXFRENS    MAX NO.OF FREE TABLE ENTRIES.
REF      FBOUNDS     HOLDS FREE-BLK BOUNDS FOR GARB.COLL.
REF      FAQMS       HOLDS FREE-ACCUMULATIONS TO FBOUNDS.
REF      NR2MOVE     NO.OF REGIONS TO MOVE IN GARB.COLL.
REF      BLKWANTD    HOLDS SIZE OF NEW DATA BLK WANTED.
REF      RESULT      PTR TO 'RESULT' DATA BLK.
REF      RSRANK      RANK FOR 'RESULT' DATA BLK.
REF      RSTYPE      TYPE FOR 'RESULT' DATA BLK.
REF      STRAYS      AREA FOR STRAY DATA BLK PTRS.
REF      STRAYBLK    TOTAL SIZE OF STRAY DATA BLK PTR SET
REF      DBROOT      IF NZ, PTS AT ROOT DB FOR LIST-FDEF.
REF      DBSERIES    PTR INTO A LIST OR FUN. DESCRIPTOR.
REF      CURRCS      PTR TO CODESTRING DATA BLK (+2)
REF      DREFSAVE    SAVE REGS DURING DE-REFERENCING.
REF      GARBSAVE    SAVE REGS FOR GARBAGE COLLECT, ET AL
REF      GCTEMP      TEMP FOR GARBAGE COLLECT.
REF      LINKGC      LINK TO GARBAGE COLLECTOR.
REF      LINKWS      LINK FOR WORKSPACE MGMT ROUTINES.
REF      TOPOSTAK    PTS AT TOP OF EXECUTION STACK.
REF      STATEPTR    PTS AT TOP STATE-ENTRY IN STACK.
REF      GOSTATE     PTS AT GO-STATE ENTRY IN STACK.
*
REFS TO CONSTANTS:

```

```

REF      X1FFFF      X'1FFFF'
REF      ZEROZERO    0,0
REF      BLANKS      WD CONTAINING ALL BLANKS.
REF      NONAME      RANGE THAT EXCLUDES NAME-CHAR CODES.
REF      FUNTYPES    RANGE OF FUN.DESRIPTOR D.B. TYPES.
REF      BITPOS      32-WD TBL OF BITS (BITPOS-K CONTAINS
                     A WD HAVING A 1 ONLY IN BIT POS K)
*
*
* EQU'S RELATED TO CONTEXT
*
HASHAQM EQU      NAMEWDS      PARTIAL HASH VALUE ACCUMULATOR.
*
* STANDARD EQU'S
*
REGISTERS
R0      EQU      0
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
* OTHER EQU'S
*
STOPNMCD EQU      21      STOPNAME CODESTRING DESIGNATOR.
NAMECODE EQU      23      ORDINARY NAME CODESTRING DESIGNATOR.
LASTCSV EQU      138     LAST CODESTRING DESIGNATOR, CODES
*                          ABOVE THIS VALUE ARE FOR THE
*                          NAME-START CHARS.
DELTA EQU      X'48'     DELTA CHAR (EBCDIC MAPPING).
HASHINC EQU      32      HASHING INCREMENT ON HASH-CRASH.
TOPRANK EQU      63      MAX. ALLOWED RANK.
TYPEXSEQ EQU      5      TYPE FOR INDEX-SEQUENCE DATA BLKS.
TYPELIST EQU      6      SIGNIFIES LIST TYPE DATA BLOCKS.
TYPLONGN EQU      X'12'  SIGNIFIES LONG-NAME TYPE DATA BLKS.
CATLC EQU      11      LINE-CHAIN CATEGORY OF STACK ENTRY.
LISTLOFF EQU      2      OFFSET FROM LIST DB HDR TO LENGTH WD
XSIZOFF EQU      2      OFFSET TO FUN.DESCRS. XSIZE WD.
NFLOFF EQU      7      OFFSET FROM FUN.DESCR. HDR TO THE
*                          WD CONTAINING NO.OF FUNCTION LINES
TRNKXSEQ EQU      X'0501' TYPE & RANK OF INDEX-SEQ. DATA BLK.
*
* DOUBLEWORD CONSTANTS
*
BOUND 8
SANDT DATA 'S','T'      USED IN TESTING FOR STOP OR TRACE.
*
* WORD CONSTANTS
*
MINUS2 DATA X'FFFFFFFE'  ALL BUT LAST BIT POSITION.
PAGE
*****
*
* ALLOCATION REQUEST ROUTINES:
*
* ALOCRS -- ALLOCATES A DATA BLOCK FOR THE RESULT OF AN OPERATION.
* IT SETS 'RESULT' TO PT AT THAT BLOCK AND PLANTS TYPE (GIVEN
* BY 'RSTYPE') AND RANK (VIA 'RSRANK') IN THE HEADER. SEE ALSO
* 'ALOCBLK' FOR SIZE AND REF-COUNT SETTINGS.
* ALOCTRES -- ENTRY PT UNDER ALOCRS FOR TEXT TYPE DATA BLK.
* REGS: R11 (ENTRY) NO.OF ELEMENTS (BITS, BYTES, WDS, OR
* DBLWDS) NEEDED FOR RESULT.
* R14 LINK, EXIT VIA *R14.
* R4 (EXIT) ALSO PTS AT NEW DATA BLOCK.
* R7 AND R11 ARE VOLATILE; SEE ALSO 'ALOCBLK'.
*
* ALOCNONX -- ALLOCATES SPACE FOR NON-EXECUTION PROCESSES, USING
* ALOCHNW. LINK IS R14: RETURN+0 IF WS FULL & RETURN+1 IF OK.
*

```

```

* ALOCHNW -- ALLOCATES A DATA BLOCK OF N WDS + 2 WDS FOR HDR + AN      *
* EXTRA WD IF NECESSARY TO REACH AN EVEN NO.  SEE ALSO 'ALOCBLK' *
* REGS:  R11 (ENTRY) N WDS REQUESTED, NOT COUNTING HDR, NOT      *
*         NECESSARILY AN EVEN NO.                                *
*         R11 IS CLOBBBERED; SEE ALSO 'ALOCBLK'.                *
*         R7 LINK; EXIT IS IN 'ALOCBLK'.                        *
*                                                                *
* ALOCBLK -- ALLOCATES A DATA BLOCK, SETS ITS SIZE FIELD, AND SETS  *
* ITS REF-COUNT TO ONE.                                           *
* REGS:  R11 (ENTRY) SIZE OF BLOCK WANTED, MUST BE EVEN NO.      *
*         R7 LINK, EXIT VIA 0,R7                                  *
*         R4 (EXIT) PTS AT FIRST WD OF NEW DATA BLOCK.          *
*         R5, R10, AND R11 ARE VOLATILE.                         *
*                                                                *
ALOCNONX BAL,R7    ALOCHNW    ALLOC. HDR + N WDS & EVEN SIZE...
          AI,R14    1          OK, SET RETURN+1.
          B         *R14      (IF WS FULL, ALOCBLK EXITS HERE)

*
ALOCRS   LW,R4     RSTYPE     VECTOR ON THE TYPE OF RESULT...
TYPER    B         TYPERR,R4  0 -- IMPOSSIBLE @
          B         LOGL      1 -- LOGICAL @
          B         TEXT      2 -- CHARACTER @
          B         UPRANK     3 -- INTEGER, ERGO WDS. @
          B         REAL      4 -- REAL, ERGO DBLWDS. @
          B         XSEQ      5 -- XSEQ (INDEX SEQUENCE) @
          B         UPRANK     6 -- LIST, ERGO WDS. @

REAL     SLS,R11    1
UPRANK   AW,R11     RSRANK     DATA BLK NEEDS 1 WD PER DIMENSION.
          LI,R7     SETRS      RETURN TO 'SETRS' AFTER GETTING BLK.
ALOCHNW  AI,R11     2          2 WDS NEEDED FOR THE HEADER
          CI,R11    1          CK FOR EVEN NO.OF WDS...
          BAZ       ALOCBLK    YEP, OK.
          AI,R11    1          NOPE, GET AN EXTRA WD.
ALOCBLK  STW,R11    BLKWANTD    SAVE TOTAL BLK SIZE WANTED.
RETRY    SW,R11     FREETOTL    IS THAT MUCH FREE CURRENTLY...
          BLEZ      FRETSET     YES -- SEARCH THE FREE TABLE.
          STW,R8    LINKWS      NO, SAVE R8.
          BAL,R8    GARBCOLL    DO GARBAGE COLLECTION.
          LW,R8     LINKWS      RESTORE R8.
          LW,R10    CORLEFT     ARE MORE PAGES AVAILABLE...
          BEZ       OLAP        NO -- TRY TO OVERLAP INTO COMMON.
          AW,R11    DYNBOUND     YES, CALC. HI NEEDED DYNAMIC BOUND
          STW,R11   NEWBOUND     (WE'LL PROBABLY GET MUCH MORE).
          LCW,R11   DYNBOUND     = - THE 'OLD' BOUND.
          BAL,R14   GETDYN       GET NEW DYNAMIC PAGE OR PAGES.
          LW,R14    GARBSAVE+14 NOTE--GARBCOLL SAVED R14 HERE, SO THIS OK.
          AW,R11    DYNBOUND     CALC SIZE = 'NEW' - 'OLD' BOUNDS.
          AWM,R11   FREETOTL     ADD THAT SIZE INTO THE FREE TABLE.
          AWM,R11   FREETBL+1
          LW,R11    BLKWANTD     GET SIZE WANTED (WE MAY NOT SUCCEED)
          B         RETRY
OLAP     LW,R11     TOPOSTAK    CALC. SPACE BETWEEN STACK & LAST
          SW,R11    FREETBL     DYNAMIC WD IN USE.
          SW,R11    BLKWANTD    = SPACE LEFT OVER AFTER THE NEW BLK.
          SAS,R11   -2          DIVIDE BY 4 (4 IS MINIMUM ACCEPTBLE)
          SAS,R11   1          TIMES 2 GIVES APPX. MIDPOINT.
          AI,R11    0          TEST...
          BGZ       GOODY       OK.
          CI,R7     ALOCNONX+1   NONE OR LESS -- WS FULL, WHO CALLS
          BNE       ERWSFUL     EXECUTION PROCESS CALLED.
          B         1,R7        NON-EXECUTION PROCESS CALLED.
GOODY    AW,R11     BLKWANTD    = AMT ADDED TO DYNAMIC.
          STW,R11   FREETOTL    PUT THAT AMT IN THE FREE TABLE.
          STW,R11   FREETBL+1
          AW,R11    FREETBL     CALC NEW DYNAMIC BOUNDARY, AND MAKE
          STW,R11   DYNBOUND     IT THE CURRENT EXEC. STACK LIMIT
          STW,R11   STKLIMIT    ALSO; THEN 'SEARCH' FOR NEW BLK.
FRETSET  LI,R5      FREETBL-2   PT AT ENTRY BEFORE FREE TABLE.
FREESRCH AI,R5      2          PT AT NEXT FREE TABLE ENTRY.
          LW,R11    1,R5        GET ITS SIZE...
          BNEZ      SIZETEST     OK, TRY IT.
          STW,R8    LINKWS      ZERO MEANS NO ENTRY BIG ENUF.
          BAL,R8    GARBCOLL    DO GARBAGE COLLECTION.
          LW,R8     LINKWS      RESTORE R8.
          LW,R11    FREETBL+1   GET SIZE COLLECTED, WE KNOW IT'S OK.
          LI,R5     FREETBL     PT AT THAT ENTRY.
SIZETEST CW,R11     BLKWANTD    IS FREE BLK BIG ENUF...
          BL        FREESRCH     NO, KEEP SEARCHING THE FREE TBL.

```

	BG	SHRINK	YES, BIGGER THAN NEEDED.
	LW, R4	0, R5	EXACT SIZE NEEDED, PURGE THIS FREE
	SLS, R5	-1	TABLE ENTRY BY MOVING LATER
MOVEUP	AI, R5	1	ENTRIES UP THE TABLE.
	LD, R10	0, R5	
	STD, R10	-2, R5	
	BNEZ	MOVEUP	BOTTOM ENTRY IS DOUBLE ZERO.
	LW, R11	BLKWANTD	GET SIZE OF DATA BLOCK.
	AI, R5	-DA(FREETBL) -1	HAVE WE CLEARED OUT ALL OF FREE TBL
	BNEZ	FINALE	NO, JUST ONE ENTRY AMONG OTHERS.
	LW, R5	DYNBOUND	YES, SET 1ST FREE LOCATION TO BE
	STW, R5	FREETBL	THE BOUNDARY FOR DYNAMIC.
FINALE	STW, R11	0, R4	SET THE DATA BLK HDR'S SIZE FIELD.
	LCW, R11	BLKWANTD	
	AWM, R11	FREETOTL	DECREASE THE TOTAL NO.OF FREE WDS.
	LI, R11	1	
	STW, R11	1, R4	SET DATA BLK'S REF-COUNT = 1.
	B	0, R7	EXIT.
SHRINK	LW, R4	0, R5	= LOC OF NEW DATA BLOCK
	SW, R11	BLKWANTD	SHRINK THIS FREE TBL ENTRY'S SIZE.
	STW, R11	1, R5	
	LW, R11	BLKWANTD	GET SIZE OF DATA BLOCK, AND ADJUST
	AWM, R11	0, R5	LOC OF FREE BLK THAT REMAINS.
	B	FINALE	
SETRS	STW, R4	RESULT	RESULT REFS THE NEW DATA BLOCK.
	LW, R11	RSTYPE	GET TYPE
	SLS, R11	8	MAKE ROOM FOR
	OR, R11	RSRANK	THE RANK AND
	STH, R11	*RESULT	FINISH DATA BLK HEADER.
	B	*R14	EXIT.
TEXT	AI, R11	3	ROUND NO.OF CHARS UP TO WD MULTIPLE.
	SLS, R11	-2	NO.OF WDS NEEDED FOR TEXT DATA.
	B	UPRANK	
LOGL	AI, R11	31	ROUND NO.OF BITS UP TO WD MULTIPLE.
	SLS, R11	-5	NO.OF WDS NEEDED FOR LOGICAL DATA.
	B	UPRANK	
ALOCTRES	EQU	TEXT	ALLOCATE TEXT RESULT DATA BLK.
XSEQ	LI, R11	6	AN INDEX SEQ BLK USES 6 WDS, ALMOST.
	BAL, R7	ALOCBLK	
	STW, R4	RESULT	RESULT REFS THE NEW DATA BLOCK.
	LI, R11	TRNKXSEQ	SET TYPE AND RANK (= 1) FOR XSEQ.
	STH, R11	*RESULT	
	B	*R14	EXIT.

	* GIVEBACK -- GIVES BACK A PORTION OF A DATA BLOCK. *		
	* REGS: R11 (ENTRY) NO.OF WDS NOT NEEDED. *		
	* R7 LINK, EXIT VIA 0, R7 *		
	* R4 (ENTRY) PTS AT DATA BLOCK HEADER. *		
	* NOTE -- SEE ALSO DREF. R4 & R11 ARE VOLATILE. *		
	* *		
GIVEBACK	AND, R11	MINUS2	WON'T GIVE BACK AN ODD NO.OF WDS.
	BEZ	0, R7	FORGET IT.
	STW, R11	BLKWANTD	SAVE EVEN NO.OF WDS TO GIVE UP.
	LCW, R11	BLKWANTD	= - NO.OF WDS TO GIVE UP.
	AWM, R11	0, R4	TAKE THAT MANY FROM THE DATA BLK.
	INT, R11	0, R4	GET SIZE (SHRUNKEN) OF THAT DATA BLK
	AW, R4	R11	PT AT 1ST WD AFTER THAT DATA BLK.
	LW, R11	BLKWANTD	FAKE DATA BLK (TYPE ZERO) CONTAINING
	STW, R11	0, R4	'GIVE UP' AMT FOR ITS SIZE FIELD.
	LI, R11	1	ALSO FAKE A REF-COUNT OF ONE FOR IT.
	STW, R11	1, R4	
	B	DREF	NOW DEREFERENCE THE FAKE DATA BLK.
PAGE			

	* CKVDB -- CHECKS STRUCTURE OF A VARIABLE OR TEMP DATA BLK. *		
	* CKVDB13 -- ALT.ENTRY; R13= EVEN, NONZERO SIZE *		
	* VERIFIES: *		
	* SIZE IS EVEN AND NONZERO *		
	* RANK IS PERMISSIBLE *		
	* LENGTH WORDS LIE INSIDE THE DATA BLK *		
	* TYPE IS LOGL, TEXT, INTG, REAL, XSEQ *		
	* DATA LIES INSIDE THE DATA BLK *		
	* NO LENGTH WORD IS NEGATIVE *		
	* REGS: R4 (ENTRY) PTS AT DATA BLK HEADER. *		
	* R6 (ENTRY) CONTAINS TYPE. *		
	* R7 (LINK) RETURN-0 IF BAD DATA BLK. *		


```

*                                     RETURN-1 IF OK.
*                                     R4,R5,R12,R13 ARE VOLATILE.
*
CKVDB  INT,R13  0,R4          GET DATA BLK SIZE.
      CI,R13   1             VERIFY EVEN, NONZERO SIZE.
      BCS,5    0,R7          ERROR EXIT.
CKVDB13 AI,R13   -2          DISCOUNT DATA BLK HEADER.
      LI,R5    1             SET FOR 1 ELEMENT OF DATA.
      LB,R12   *R4,R5        GET RANK.
      CI,R12   TOPRANK       VERIFY REASONABLE RANK...
      BG      0,R7          ERROR EXIT.
      SW,R13   R12          DISCOUNT ANY LENGTH WDS...
      BLZ     0,R7          TOO SHORT -- ERROR EXIT.
      CI,R6    TYPEXSEQ      VERIFY TYPE...
      BLE     CKVDBV,R6      @ OK, VECTOR ON TYPE.
CKVDBV  B      0,R7          @ 0 OR HIGH -- ERROR EXIT.
      B      CKVDBL          @ LOGL
      B      CKVDBT          @ TEXT
      B      CKVDBI          @ INTG
      B      CKVDBR          @ REAL
      AI,R12   -1           @ XSEQ, VERIFY RANK =1...
      BNEZ    0,R7          ERROR EXIT.
      AI,R13   -3           VERIFY SIZE...
      BGEZ    1,R7          OK EXIT.
      B      0,R7          ERROR EXIT.
CKVDBL  SLS,R13  5          32 ELEMS/WORD.
      B      CKVDBI
CKVDBT  SLS,R13  2          4 ELEMS/WORD.
      B      CKVDBI
CKVDBR  SLS,R13  -1         HALF ELEM/WORD.
CKVDBI  AI,R4    1          PT AT REF-COUNT WD.
CKVDBQ  AI,R12   -1         DECR RANK REMAINING...
      BGEZ    CKVDBM        MULT NEXT LENGTH.
      CW,R5    R13          DONE, DO # ELEMS FIT...
      BLE     1,R7          OK EXIT.
      B      0,R7          ERROR EXIT.
CKVDBM  AI,R4    1          PT AT NEXT LENGTH WD.
      MW,R5    0,R4          MULT THAT LENGTH...
      BOV     0,R7          OFLO -- ERROR EXIT.
      BGZ     CKVDBQ        LOOP IF ABOVE ZERO.
CKVDBN  BLZ     0,R7          NEG LENGTH WD -- ERROR EXIT.
      AI,R12   -1           DECR RANK REMAINING...
      BLZ     1,R7          DONE (EMPTY ARRAY) OK EXIT.
      AI,R4    1           PT AT NEXT LENGTH WD.
      LW,R5    0,R4          TEST FOR NEG. LENGTH.
      B      CKVDBN

```

PAGE

```

*
* WSCHEK -- CHECKS WORKSPACE STRUCTURE. THIS ENTRY PROVIDES
*          A DIAGNOSTIC TOOL; CALLS TO WSCHEK CAN BE PATCHED
*          IN WHEN IT IS SUSPECTED THAT SOME OPERATION IS DAMAGING
*          THE WORKSPACE.
* WCKDSPL -- CHECKS A LOADED WORKSPACE STRUCTURE AND
*            DISPLACES DATA BLOCK POINTERS IF THE WORKSPACE
*            WAS SAVED WITH A DIFFERENT DYNAMIC ORIGIN
*            (R2, UPON ENTRY, MUST CONTAIN THE DISPLACEMENT
*            VALUE OR ZERO -- NOTE R2 IS VOLATILE FOR THIS
*            ENTRY POINT).
* REGS:    R14 -- LINK, RETURN-0 IF WS CHECKS OUT
*           RETURN-1 IF BAD WS.
*           REGS ARE VOLATILE IF BAD WS
*           R2 IS VOLATILE FOR THE WCKDSPL ENTRY;
*           OTHERWISE, REGS ARE PRESERVED.
* CHECKS PERFORMED:
*   - EACH EXECUTION-STACK ENTRY HAS A VALID CATEGORY
*   - NO DATA BLOCK OVERLAPS THE DYNAMIC BOUNDARY.
*   - EACH DATA BLK HAS A POSITIVE REF COUNT.
*   - EACH DATA BLK SIZE IS EVEN & NONZERO.
*   - EACH DATA BLK TYPE FIELD IS VALID:
*     - (LOGL,TEXT,INTG,REAL,XSEQ) FURTHER TESTS ARE
*       MADE BY THE 'CKVDB' ROUTINE.
*     - (LIST) TRACKING WORD MUST BE ZERO.
*     - (CODESTRING) OFFSET ABOVE ZERO AND RTMOST
*       BYTE WITHIN 7 BYTES OF THE NEXT BLOCK.
*     - (INTRINSIC FUNCTION) SIZE = 2.
*     - (GROUP) LAST HALFWO WITHIN 3 HALFWDS OF THE
*       NEXT BLOCK.

```

U06-0004

```

*          - TOTAL REF-COUNT ACCUMULATION BALANCES THE TOTAL
*          NO.OF DATA BLOCK POINTERS.
* THE DATA BLK POINTERS OF INTEREST ARE THOSE OCCURRING IN:
* THE STRAY BLOCK 'STRAYS',
* THE EXEC. STACK (V,X,Q,D,F CATEGORY ENTRIES),
* THE SYMBOL TABLE (REF-INDICATORS & LONG-NAME PTRS),
* THE DATA BLOCKS THEMSELVES (LISTS & FUNCTION DESCR'S).
*
WSCHEK  STW,R2    GARBSAVE+2    SAVE R2 FOR THIS ENTRY PT.
        LI,R2     0             SET DISPLACEMENT VALUE = 0.
WSCKDSPL LCI      11           SAVE R3 THRU R13.
        STM,R3    GARBSAVE+3
        LI,R3     0             CLEAR REF-ACCUMULATION.
        LI,R7     -1
WFREESET AI,R7     1
        LD,R4     FREETBL,R7    FREE FRAG. (LOC & SIZE)...
        AI,R5     0
        BEZ       WSTRAYS       NO MORE IN USE.
        STW,R3    1,R4          FAKE A DATA BLK OF THAT SIZE
        STW,R5    0,R4          MAKE IT A FREE BLK,
        B         WFREESET      REF-CNT = 0 & TYPE = 0.
WSTRAYS LI,R13    STRAYBLK     = # STRAY DATA BLK PTRS.
        LI,R4     STRAYS       PT AT 1ST STRAY D. B. PTR.
        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     1            PT AT NEXT.
        BDR,R13   AQMDP        LOOP THRU STRAYS.
        LW,R4     TOPOSTAK     PT AT TOP OF EXEC. STACK.
        B         WSTAK
WDROPOP AI,R4     2            SKIP 2-WD 0-CATEGORY.
        B         WSTAK
W1       BAL,R7    AQMDP        ACCUM REF & DISPLACE.
WPOP     AI,R4     1            PT AT NEXT EXEC.STACK WD.
WSTAK    LB,R6     *R4          = STACK ENTRY CATEGORY...
        AI,R6     -CATLC
        BLEZ      WSCAT+CATLC,R6 VECTOR IF LEGAL CATEGORY.
        BAL,R15   WSERR        OOPS -- BAD CATEGORY.
WSCAT    B         W1          @ V
        B         WPOP        @ A'
        B         WDROPOP     @ O
        B         W1          @ X
        B         WPOP        @ B
        B         WPOP        @ P
        B         WPOP        @ S
        AI,R4     1           @ Q -- MOVE TO CURRPTR WD.
        B         W1          @ D
        B         WPOP        @ A
        B         WFCAT       @ F
        B         WLC         @ LINE-CHAIN
*        B         WLC         @
WLC      BAL,R7    AQMDP        @ ACCUM REF & DISPLACE.
        AI,R4     3           @ PT AT NEXT ENTRY IN EXEC.STACK.
        B         WSTAK
WFCAT    LI,R5     X'7FFF'      EXTRACT ITS 'NEXT' FIELD...
        AND,R5    0,R4
        BEZ       WSYM        (FINAL) HIT SYMBOL TABLE.
        AI,R4     1           (FUNC.) PT AT FDEFPTR WD.
        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     1           PT AT CALLPTR WD.
        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     3           PT 2 WDS PAST # SHAD. PAIRS.
        LW,R13    -2,R4        HOW MANY...
        BEZ       WFBU        NONE.
        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     2           PT 2 WDS FURTHER.
        BDR,R13   AQMDP        LOOP TILL BEYOND.
WFBU     BDR,R4    WSTAK        BACK UP TO ENTRY PAST F-ENTRY &
*        B         WSTAK        GO CHECK ITS CATEGORY.
WSYM     LW,R13    SYMTSIZE     = # ENTRIES FOR SYM TBL.
        LI,R12    NAMEWDSZ     = MAX # WDS TO HOLD NAME.
        LW,R4     SYMT         PT AT 1ST REF-INDIC WD.
WSYME    LI,R7     WSYMN        EXIT AQMDP AT WSYMN.
AQMDP    LI,R5     X'1FFFFF'
        AND,R5    0,R4
        BEZ       0,R7         TEST FOR DATA BLK PTR...
        AWM,R2    0,R4         NIL -- EXIT.
        AI,R3     1           YES, DISPLACE IT PERHAPS.
        B         0,R7        ACCUM 1 MORE REFERENCE.
        AI,R4     1           EXIT.
WSYMN    AI,R4     1           PT AT NEXT NAME-INDIC WD
        CB,R12    *R4          & CK FOR SHORT NAME...

```

	BL	WSYMS	YES.	
	BAL, R7	AQMDP	NO, LONG OR NIL.	
WSYMS	AI, R4	1	PT AT NEXT REF-INDIC WD &	
	BDR, R13	WSYME	LOOP TILL DONE.	
	STW, R4	GCTEMP	DONE, R4 PTS AT 1ST DATA	
	B	WDBQ	BLK, PROBABLY.	
WDBV	BAL, R7	CKVDB13	@ CK VAR OR TEMP DATA BLK...	
	BAL, R15	WSERR	@ OOPS -- BAD.	
WNBLK	LW, R4	GCTEMP	@ OK, PT AT NEXT BLK UNLESS	
WDBQ	CW, R4	DYBOUND	REACHED DYNAMIC BOUNDARY...	
	BL	WDB	NOT YET.	
	BE	WEND	YES, GOOD SO FAR.	
	BAL, R15	WSERR	TOO FAR -- BAD.	
WDB	INT, R13	0, R4	GET SIZE OF THIS DATA BLK.	
	AWM, R13	GCTEMP	(UPDATE PTR TO NEXT BLK).	
	CI, R13	1	SIZE EVEN & NONZERO...	
	BCR, 5	WDBREF	OK.	
	BAL, R15	WSERR	OOPS -- BAD.	
WDBREF	SW, R3	1, R4	ACCT FOR BLK'S REF-COUNT.	
	LB, R6	*R4	GET DATA BLK'S TYPE...	
	BEZ	WNBLK	FREE BLOCK.	
	LW, R8	1, R4	(CK REF-COUNT)	U06-0006
	BGZ	WDBREFOK	(OK)	U06-0007
	BAL, R15	WSERR	(BAD)	U06-0008
WDBREFOK	CI, R6	TYPELIST		U06-0009
	BL	WDBV	VARIABLE OR TEMP.	
	AI, R6	-TYPLONGN		
	BLEZ	WDBT-TYPELIST+TYPLONGN, R6		
	BAL, R15	WSERR	INVALID TYPE -- BAD.	
WDBT	B	WLIST	@ LIST.	
	B	WCS	@ CODESTRING.	
	B	WFUND	@ FUNCTION DESCRIPTOR.	
	B	WFUND	@ '' ''	
	B	WFUND	@ '' ''	
	B	WFUND	@ '' ''	
	B	WFUND	@ '' ''	
	B	WFUND	@ '' ''	
	B	WIF	@ INTRINSIC FUNCTION.	
	B	WIF	@ '' ''	
	B	WIF	@ '' ''	
	B	WGRP	@ GROUP.	
	B	WLN	@ LONG NAME.	
WLN	EQU	WNBLK		
WIF	AI, R13	-2	IS SIZE = 2...	
	BEZ	WNBLK	YES.	
	BAL, R15	WSERR	NO -- BAD.	
WGRP	AI, R4	2	PT AT WD CONTAINING	
	LH, R8	*R4	# NAMES IN THIS GROUP.	
	SLS, R8	-1	GO WITH # OF FURTHER WDS	
	B	WOFFCK	IN GROUP DATA BLK.	
WCS	AI, R4	2	PT AT WD CONTAINING OFFSET.	
	LH, R8	*R4	VERIFY ABOVE ZERO...	
	BGZ	WCSO	OK # BYTES OF CODESTRING.	
	BAL, R15	WSERR	BAD.	
WCSO	SLS, R8	-2	# MORE WDS IN CS DATA BLK.	
WOFFCK	AW, R4	R8	PT AT LAST WD IN USE.	
	OR, R4	BITPOS-31	(OR ODD GARBAGE WD).	
	AI, R4	1	PT AT NEXT WD; IT SHOULD BE	
	CW, R4	GCTEMP	NEXT DATA BLK LOC.	
	BE	WDBQ	OK, IT IS.	
	BAL, R15	WSERR	BAD.	
WLIST	AI, R4	3	PT PAST LENGTH WD.	
	LW, R13	-1, R4	GET LENGTH OF LIST...	
	BEZ	WLISTZ	0, CK TRACKING WD ONLY	
	BAL, R7	AQMDP	ACCUM REF & DISPLACE.	
	AI, R4	1	PT AT NEXT WD.	
	BDR, R13	AQMDP	LOOP TILL TRACKING WD HIT.	
WLISTZ	LW, R13	0, R4	TRACKING WD = 0...	
	BEZ	WNBLK	YES -- OK.	
	BAL, R15	WSERR	NO -- BAD.	
WFUND	AI, R4	XSIZEOFF	PT AT XSIZE WD.	
	LI, R13	X'E0000'		
	CW, R13	0, R4	DOES IT PT AT ERR-CTRL TBL...	
	BAZ	WFUNDNFL	NO.	
	BAL, R7	AQMDP	YES, ACCUM REF & DISPLACE.	
WFUNDNFL	LW, R13	NFLOFF-XSIZEOFF, R4	GET # FUNCTION LINE PTRS...	
	BEZ	WNBLK	NONE.	
	AI, R4	-XSIZEOFF+NFLOFF+1	PT AT 1ST LINE PTR.	

```

        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     1            PT AT NEXT WD.
        BDR,R13   AQMDP        LOOP TILL LOCAL-LABEL WD.
        LI,R13    X'FFFF'
        AND,R13   0,R4        GET # LABEL ENTRIES...
        BEZ       WNBLK        NONE.
        AI,R4     2            PT AT ENTRY'S DB PTR WD.
        BAL,R7    AQMDP        ACCUM REF & DISPLACE.
        AI,R4     2            PT AT NEXT, IF ANY.
        BDR,R13   AQMDP        LOOP TILL DONE
        B         WNBLK
WEND      AI,R3     0            CK REF BALANCE...
        BEZ       WSOK        OK.
        LI,R15    WSERR        BAD (SET R15).
WSERR     AI,R14   1            SET FOR RETURN-1 (ERROR).
        B         *R14
WSOK      LCI      12
        LM,R2     GARBSAVE+2    RESTORE REGS SAVED ON ENTRY.
        B         *R14        RETURN.

PAGE
*****
*
* CTEST -- WHEN MORE COMMON IS NEEDED, CTEST IS ENTERED. IT DECIDES
* WHETHER TO GET A NEW PAGE OF COMMON OR TO OVERLAP INTO THE
* CURRENT DYNAMIC AREA.
* REGS:  R8 -- LINK (VOLATILE) EXIT VIA *LINKWS.
* RETURN+0 IF WS FULL.
* RETURN+1 IF ENUF COMMON OBTAINED.
* R1 -- (EXIT) PTS AT LOC REQUESTED BY LOCNEED IF OK.
*
CTEST     AI,R8     1            SET FOR NORMAL RETURN.
        STW,R8     LINKWS        SAVE THAT LINKAGE.
        STW,R1     TOPOSTAK      SAVE PTR TO CURRENT TOP OF STACK.
CTEST1    LW,R8     CORLEFT      ARE PAGES AVAILABLE...
        BNEZ      GET1C        YES -- GET 1 MORE.
        BAL,R8     GARBCOLL      NO, DO GARBAGE COLLECTION.
        LW,R8     LOCNEED      CALC. SPACE BETWEEN LOC NEEDED &
        SW,R8     FREETBL      LAST DYNAMIC WD IN USE.
        SAS,R8     -2          DIVIDE BY 4 (4 IS MINIMUM ACCEPTBLE)
        SAS,R8     1          TIMES 2 TO GET APPX. MIDPOINT
        AI,R8     0          TEST...
        BGZ       CTESTOK      ENUF.
        MTW,-1     LINKWS      NOT ENUF, BACK UP FOR THE
        B         *LINKWS      WS FULL EXIT.
CTESTOK   STW,R8     FREETOTL     PUT ABOUT HALF OF LEFT-OVER SPACE
        STW,R8     FREETBL+1     IN FREE TABLE, SETTING THE
        AW,R8     FREETBL      DYNAMIC BOUNDARY & STACK LIMIT
        STW,R8     DYNBOUND     EQUAL (AT MID POINT).
GOTC      STW,R8     STKLIMIT     SET NEW STACK LIMIT.
        LW,R1     LOCNEED      SET STACK PTR TO LOC NEEDED.
        B         *LINKWS      EXIT
GET1C     STW,R14    GARBSAVE+14  SAVE R14.
        BAL,R14    GETCOM       GET ANOTHER COMMON PAGE.
        LW,R14    GARBSAVE+14  RESTORE R14.
        LW,R1     LOCNEED      SET STACK PTR TO LOC NEEDED.
        CW,R1     STKLIMIT     MAKE SURE WE HAVE ENUF STACK...
        BG        *LINKWS      YES, EXIT
        LW,R1     TOPOSTAK      NO, RESET R1 IN CASE WS FULL.
        B         CTEST1      TRY FOR ANOTHER PAGE.

PAGE
*****
*
* SICLR -- CLEARS ENTRIES OFF THE STATE INDICATOR UNTIL REACHING THE
* ENTRY THAT 'GOSTATE' POINTS TO.
* SICLR% -- ALTERNATE ENTRY PT, R1 PTS AT TOP OF STACK.
* REGS:  R14 LINK, EXIT VIA *R14.
* R1 (EXIT) PTS AT NEW TOP OF STACK AFTER CLEARANCE.
* R2 (EXIT) CATEGORY AT TOP OF STACK AFTER CLEARANCE.
* VOLATILE: R4,R5,R7,R13 SEE ALSO 'DREF'.
*
SICLR     LW,R1     TOPOSTAK      PT AT TOP OF STACK.
        B         SICLR%
SICLR0P   AI,R1     2            DROP BOTH WDS OF 0-CATEGORY.
        B         SICLR%
SICLRQ    AI,R1     1            DROP 1ST WD OF Q-CATEGORY.
        MTW,-2     0,R1        MAKE CODESTRING PTR PT AT DB HDR.
SICLR1    LI,R4     0
        XW,R4     0,R1        CLEAR & DEREFERENCE IF DB PTR.

```

	BAL,R7	MAYDREF	
SICLRG	AI,R1	1	DROP A WD.
SICLR%	LB,R2	*R1	GET CATEGORY NOW AT TOP.
	CW,R1	GOSTATE	HAVE WE REACHED THE GO-STATE YET...
	BL	SICLRT,R2	NO, VECTOR ON THE CATEGORY.
	STW,R1	TOPOSTAK	YES, UPDATE TOP OF STACK PTR.
	STW,R1	STATEPTR	UPDATE STATE PTR EQUIVALENTALLY.
	B	*R14	EXIT.
SICLRT	B	SICLR1	@ V
	B	SICLRG	@ A-PRIME
	B	SICLROP	@ O
	B	SICLR1	@ X
	B	SICLRG	@ B
	B	SICLRG	@ P
	B	SICLRG	@ S
	B	SICLRQ	@ Q
	B	SICLR1	@ D
	B	SICLRG	@ A
*	B	SICLRF	@ F
*			@ (LINE-CHAIN NOT ALLOWED)
SICLRF	LI,R4	0	@
	XW,R4	1,R1	CLEAR & DERE FDEFPTR WD UNLESS SI-
	BAL,R7	MAYDREF	DAMAGED ENTRY.
	LI,R4	0	
	XW,R4	2,R1	CLEAR & DERE CALLPTR WD AFTER
	AI,R4	-2	CODESTRING PTR ADJUSTED TO DB HDR.
	BAL,R7	DREF	
	AI,R1	4	PT AT WD AFTER NO.OF SHADOW PAIRS.
	LW,R13	-1,R1	HOW MANY...
	BEZ	SICLR%	NONE, RESUME CLEARANCE.
	LI,R7	SICLRFX	SET RETURN FROM MAYDREF CALL, BELOW.
SICLRFS	LW,R5	0,R1	GET NAME PTR OF SHADOW PAIR ENTRY.
	LI,R4	0	CLEAR & GET SHADOWED REFERENT.
	XW,R4	1,R1	
	XW,R4	*SYMT,R5	UNSHADOW.
	AI,R1	2	PT PAST THIS SHADOW PAIR.
	B	MAYDREF	DEREF LOCAL DATA BLK, IF ANY.
SICLRFX	BDR,R13	SICLRFS	LOOP TILL DONE WITH SHADOW PAIRS.
	B	SICLR%	

PAGE

* IN2CODE--BYTE TABLE. FOR EACH POSSIBLE INTERNAL CHARACTER, THERE IS A
 * CORRESPONDING BYTE. IN MOST CASES THIS IS THE CODESTRING VALUE
 * FOR THAT CHARACTER OR (FOR NAME CHARS) THE HASH CODE. BLANK,
 * END-OF-INPUT, AND QUOTE USE SPECIAL VALUES; BAD (UNEXPECTED)
 * CHARS. USE THE SPECIAL CODE X'47'.

	CODE BYTES	* HEX * *RANGE*	TRANSLATION CORRESPONDENCES:			
			(BYTE 0)	(BYTE 1)	(BYTE 2)	(BYTE 3)
IN2CODE	DATA X'47474747'	@00-03	BAD	BAD	BAD	BAD
	DATA X'47474747'	@04-07	BAD	BAD	BAD	BAD
	DATA X'47474747'	@08-0B	BAD	BAD	BAD	BAD
	DATA X'47474747'	@0C-0F	BAD	BAD	BAD	BAD
	DATA X'47474747'	@10-13	BAD	BAD	BAD	BAD
	DATA X'47324747'	@14-17	BAD	END-INPT	BAD	BAD
	DATA X'47474747'	@18-1B	BAD	BAD	BAD	BAD
	DATA X'47474747'	@1C-1F	BAD	BAD	BAD	BAD
	DATA X'47474747'	@20-23	BAD	BAD	BAD	BAD
	DATA X'47474747'	@24-27	BAD	BAD	BAD	BAD
	DATA X'47474747'	@28-2B	BAD	BAD	BAD	BAD
	DATA X'47474747'	@2C-2F	BAD	BAD	BAD	BAD
	DATA X'47474747'	@30-33	BAD	BAD	BAD	BAD
	DATA X'47474747'	@34-37	BAD	BAD	BAD	BAD
	DATA X'47474747'	@38-3B	BAD	BAD	BAD	BAD
	DATA X'47474747'	@3C-3F	BAD	BAD	BAD	BAD
	DATA X'3365727E'	@40-43	BLANK	%FCT	%ECD	%
	DATA X'63784B9D'	@44-47	%MIN	%E	%GU	%UDL
	DATA X'D055832D'	@48-4B	%DLT	%I	%CPL	.(DOT)
	DATA X'66295B64'	@4C-4F	%LT	(	+	%ABS
	DATA X'854A2423'	@50-53	%CAP	%IB	%QQ	%Q
	DATA X'52736160'	@54-57	%TBR	%DCD	%O	%LOG
	DATA X'59142E86'	@58-5B	%REV	%COM	%SC	%CUP
	DATA X'5F2A264D'	@5C-5F	*	)	;	%NOT
	DATA X'5C546253'	@60-63	%-(SUBR)	/	%MAX	%RD1
	DATA X'75478084'	@64-67	%DRP	BAD	%W	%CPR
	DATA X'58477457'	@68-6B	%RV1	BAD	%TAK	,
	DATA X'567D6851'	@6C-6F	%R	%U	%GT	%RND

DATA X'6C7B7C4C'	@70-73	&	%DRS	-(NEG)	%GD
DATA X'6782696E'	@74-77	%LE	%LOK	%GE	%NND
DATA X'6F6D876A'	@78-7B	%NOR	%OR	:	%NE
DATA X'7F346B81'	@7C-7F	@	'(QUOTE)	=	%DEL
DATA X'4799BBCE'	@80-83	BAD	%UA	%UB	%UC
DATA X'F0BFA1FC'	@84-87	%UD	%UE	%UF	%UG
DATA X'A9F44747'	@88-8B	%UH	%UI	BAD	BAD
DATA X'47474747'	@8C-8F	BAD	BAD	BAD	BAD
DATA X'47E6D2E2'	@90-93	BAD	%UJ	%UK	%UL
DATA X'C5D6B58F'	@94-97	%UM	%UN	%UO	%UP
DATA X'B1954747'	@98-9B	%UQ	%UR	BAD	BAD
DATA X'47474747'	@9C-9F	BAD	BAD	BAD	BAD
DATA X'4747C393'	@A0-A3	BAD	BAD	%US	%UT
DATA X'A5F8EADA'	@A4-A7	%UU	%UV	%UW	%UX
DATA X'DECA4747'	@A8-AB	%UY	%UZ	BAD	BAD
DATA X'47474747'	@AC-AF	BAD	BAD	BAD	BAD
DATA X'76775A47'	@B0-B3	%XP1	%XPD	%TPS	BAD
DATA X'2728191A'	@B4-B7	%(	%)	%Q0	%Q1
DATA X'1B1C1D1E'	@B8-BB	%Q2	%Q3	%Q4	%Q5
DATA X'1F202122'	@BC-BF	%Q6	%Q7	%Q8	%Q9
DATA X'47AB9BBD'	@C0-C3	BAD	A	B	C
DATA X'91C7A3A7'	@C4-C7	D	E	F	G
DATA X'B3EE4747'	@C8-CB	H	I	BAD	BAD
DATA X'47474747'	@CC-CF	BAD	BAD	BAD	BAD
DATA X'47D8DC9F'	@D0-D3	BAD	J	K	L
DATA X'E4CC97B9'	@D4-D7	M	N	O	P
DATA X'F6AF4747'	@D8-DB	Q	R	BAD	BAD
DATA X'47474747'	@DC-DF	BAD	BAD	BAD	BAD
DATA X'47478DFA'	@E0-E3	BAD	BAD	S	T
DATA X'F2C1FEE8'	@E4-E7	U	V	W	X
DATA X'D4E04747'	@E8-EB	Y	Z	BAD	BAD
DATA X'47474747'	@EC-EF	BAD	BAD	BAD	BAD
DATA X'00010203'	@F0-F3	0	1	2	3
DATA X'04050607'	@F4-F7	4	5	6	7
DATA X'08095D5E'	@F8-FB	8	9	#	%
DATA X'2B2C5047'	@FC-FF	%G0	%IS	%MDV	BAD

U06-0011

PAGE

```

*****
*
* ACQNXCC -- ACQUIRES THE NEXT INTERNAL-FORM CHAR AND ITS CODE (E.G.
* CODESTRING VALUE, KEY INDICATOR, OR HASH VALUE).
* ACQCC -- ENTRY POINT FOR CURRENT CHAR, RATHER THAN NEXT ONE.
* ACQCODE -- ENTRY POINT TO JUST SET THE CODE
*
* REGS:  R1 (ENTRY) PTS AT LATEST, OR CURRENT, CHAR IN THE
*         INPUT STRING. (EXIT) PTS AT CHAR ACQUIRED.
*         R2 (EXIT) THE CHAR, ITSELF.
*         R3 (EXIT) THE CODE FOR THAT CHAR.
*         R4 LINK, EXIT VIA 0,R4
*
ACQNXCC  AI,R1      1          PT AT NEXT CHAR.
ACQCC    LB,R2      0,R1      GET CHAR.
ACQCODE  LB,R3      IN2CODE,R2 GET ITS CODE.
          B         0,R4      EXIT.

```

PAGE

```

*****
*
* ACQNXNB -- ACQUIRES THE NEXT NON-BLANK INTERNAL-FORM CHAR AND ITS
* CODE (E.G. CODESTRING VALUE, KEY INDICATOR, OR HASH VALUE).
* ACQNB -- ENTRY POINT THAT STARTS WITH CURRENT CHAR, RATHER THAN NEXT.
* REGS:  R1 (ENTRY) PTS AT LATEST, OR CURRENT, CHAR IN THE
*         INPUT STRING. (EXIT) PTS AT NON-BLANK CHAR ACQ'D.
*         R2 (EXIT) THE NON-BLANK CHAR, ITSELF.
*         R3 (EXIT) THE CODE FOR THAT CHAR.
*         R4 LINK, EXIT VIA 0,R4
*
ACQNXNB  AI,R1      1          PT AT NEXT CHAR.
ACQNB    LB,R2      0,R1      GET CHAR.
          AI,R2      -' '      CK FOR BLANK.
          BEZ        ACQNXNB   YES, TRY NEXT ONE.
          AI,R2      ' '      NO, RESTORE THAT CHAR.
          LB,R3      IN2CODE,R2 GET ITS CODE.
          B         0,R4      EXIT.

```

PAGE

```

*****
*
* ACQNAME -- ACQUIRES A NAME (GIVEN THE NAME-START UPON ENTRY), HUNTS
* FOR THAT NAME IN THE SYMBOL TABLE, IF NEW NAME ENTERS THAT

```

```

*      NAME, AND EXITS WITH THE PTR TO REFERENT-INDICATOR AND TYPE      *
*      OF NAME ACQUIRED (STOPNAME, TRACENAME, OR ORDINARY NAME).      *
*      REGS:  R1  (ENTRY) PTS AT NAME-START CHAR IN INPUT STRING.      *
*              (EXIT) PTS AT (NON-BLANK) CHAR AFTER NAME.              *
*      R2  (ENTRY) CONTAINS NAME-START CHAR.                            *
*              (EXIT) CONTAINS CHAR AFTER NAME.                        *
*      R3  (ENTRY) CODE FOR NAME-START CHAR, ITS HASH VALUE.          *
*              (EXIT) CODE FOR CHAR AFTER NAME.                        *
*      R4, R5, R7, R10, R11, R14 ARE VOLATILE.                          *
*      R6  (EXIT) PTR TO REFERENT INDICATOR WD FOR NAME                *
*      R12 LINK -- 3 RETURNS:                                           *
*              RETURN-0 -- SYMBOL TABLE FULL.                          *
*                      (ALSO USED FOR 'NAME-                            *
*                      NOT-FOUND' RETURN WHEN ACQNAME IS CALLED BY      *
*                      FINDNAME, SEE BELOW).                             *
*              RETURN-1 -- WS FULL ON LONG NAME.                        *
*              RETURN-2 -- NORMAL.                                       *
*      R13 (EXIT) TYPE OF NAME:                                         *
*              21 = STOPNAME (S-DELTA-NAME)                             *
*              22 = TRACENAME (T-DELTA-NAME)                             *
*              23 = ORDINARY NAME                                         *
*
ACQNAME  AI,R1      1      PT AT CHAR AFTER THE NAME-START
        LI,R13     DELTA
        CB,R13     0,R1      IS IT A DELTA...
        BNE        NO, ORDINARY NAME
        CLM,R2     SANDT     YES, DID NAME START ON S OR T...
        BCS,9      NORMAL    NO, ORDINARY NAME
        LW,R13     R2        YES, COMPUTE STOPNAME OR THE
        AI,R13     STOPNMCD-'S' TRACENAME CODESTRING DESIGNATOR.
        BAL,R4     ACQNXCC   ACQ NEXT CHAR AND CODE
        CI,R3      LASTCSV   IS IT A NAME-START CHAR...
        BG         STPORTRC  YES, BEGINS NAME TO STOP OR TRACE.
        AI,R1      -3        NO, ASSUME ORDINARY NAME
        BAL,R4     ACQNXCC   RE-ACQ THE S OR T.
        AI,R1      1        PRETEND TO PT AT THE DELTA.
NORMAL   AI,R1      -1        FORGET ABOUT THE CHAR AFTER 1ST ONE.
        LI,R13     NAMECODE  ORDINARY NAME CODESTR. DESIGNATOR.
STPORTRC LI,R5      -1        PRE-SET NAME BUFFER BYTE OFFSETTER.
        LI,R14     NAMLIMIT  = NO.OF CHARS ACCEPTED PER NAME
        LI,R6      0        CLEAR FOR DIVISION OF HASH TOTAL.
        LI,R7      0        CLEAR HASH ACCUMULATOR.
        STW,R7     HASHAQM
        LI,R4      ENDTESTR  SO 'BDR,R14 ACQNXCC' WILL CAUSE A
*
*      RETURN TO 'ENDTESTR' UNLESS NAME
*      GETS TOO LONG.
NAMECHAR AWM,R3     HASHAQM   ADD LATEST HASH VALUE.
        AI,R5      1        NEXT CHAR OFFSET INTO NAME BUFFER.
        STB,R2     NAMEBUF,R5 PUT CHAR IN NAME BUFFER.
        BDR,R14    ACQNXCC   ACQ NEXT CHAR IF NAME ISN'T TOO LONG
*
*      (RETURN IS TO 'ENDTESTR').
        BAL,R4     ACQNXCC   TOO LONG, SKIP TILL NON-NAME-CHAR.
        CLM,R3     NONAME    SHOWS UP.
        BCS,9      ACQNXCC   NOT YET, TRY AGAIN
        B          NAMEDONE  BINGO, FINALLY.
BLANKER  STB,R14    NAMEBUF,R5 FILL IN A BLANK.
        B          FILLER
ENDTESTR CLM,R3     NONAME    ANOTHER NAME-CHAR...
        BCS,9      NAMECHAR  YES, TRY AGAIN
NAMEDONE LW,R7      HASHAQM   NO, NAME IS DONE.
        SLS,R7     1        GET PARTIAL HASH VALUE & CALC TOTAL
        LB,R4      NAMEBUF,R5 HASH VALUE = 2 * PARTIAL HASH VALUE
        AW,R7      R4        + LAST CHAR (EBCDIC).
        BAL,R4     ACQNB     ACQ THE NON-BLANK AFTER THE NAME.
FINNAME  LI,R14     ' '      PREPARE FOR BLANK FILLING.
FILLER   AI,R5      1        CK FOR WD BOUNDARY
        CI,R5      3
        BANZ       BLANKER   NO, APPEND A BLANK IN NAME BUFFER.
        SLS,R5     -2        YES, GET NO.OF WDS FOR NAME.
        DW,R6      SYMTSIZE  REMAINDER (R6) WILL BE TENTATIVE
        SLS,R6     1        ENTRY INTO SYM TBL. CONVERT TO
        AI,R6      1        POINTER TO NAME INDICATOR WD.
        LW,R14     SYMTSIZE  FOR COUNTDOWN IF HASHING CRASHING.
        CI,R5      1        IS THIS A LONG OR SHORT NAME...
        BE         SHORT     SHORT NAME
*
*      LONG NAME
SCS,R5   -8        PUT NO.OF NAME WDS IN BYTE 0, AND

```

LONG	STW, R5	NAMEWDS	SAVE IT.
	LW, R5	*SYMT, R6	GET A NAME INDICATOR WD.
	BEZ	NEWLNAME	NEW LONG NAME.
	LB, R4	R5	LOOK FOR MATCHING WORD COUNT.
	CB, R4	NAMEWDS	
	BNE	RELONG	NO, TRY ANOTHER HASH LOC.
	AI, R5	-1	YES, PT AT REF WD IN OLD DATA BLK.
	AI, R6	-1	
	LW, R7	*SYMT, R6	(TRY ANOTHER HASH LOC IF THIS
	AI, R6	1	IS AN ALIEN LONG NAME PTR,
	CI, R7	X'40000'	I.E. DURING FILE-TYPE COPY CMD)
	BANZ	RELONG	
LGNAMECK	LW, R7	NAMEBUF-1, R4	TEST LONG NAME FOR MATCHING WORDS,
	CW, R7	*R5, R4	GOING FROM LAST WD TO FIRST WD...
	BNE	RELONG	NO, TRY ANOTHER HASH LOC.
	BDR, R4	LGNAMECK	YES, LOOP TILL FIRST WD.
	B	FOUND	OK, FOUND MATCHING NAME.
RELONG	LI, R7	LONG	SET TO RETURN TO 'LONG'
NEWSTLOC	AI, R6	2*HASHINC	INCR TO ANOTHER NAME INDICATOR WD.
	CW, R6	NSYMTWDS	ARE WE STILL IN THE SYM TBL...
	BL	INSIDE	YES.
	SW, R6	NSYMTWDS	NO, CYCLE BACK IN.
INSIDE	BDR, R14	0, R7	RETURN UNLESS WE'VE HIT ALL ENTRIES.
	B	*R12	OH-OH, TAKE SYM TBL FULL RETURN.
WSFULLLN	AI, R12	1	TAKE WS FULL RETURN, LONG NAME DATA
	B	*R12	BLK WOULDN'T FIT.
NEWLNAME	CI, R12	FINDNAME+1	WAS ACQNAME CALLED BY FINDNAME...
	BE	*R12	YES -- TAKE 'NAME-NOT-FOUND' EXIT.
	LB, R11	NAMEWDS	NO, GET NO.OF WDS FOR THE NAME.
	BAL, R14	ALOCNONX	ALLOC DATA BLK, HEADER + THOSE WDS.
	B	WSFULLLN	OH-OH -- WS FULL.
	LI, R11	TYPLONGN	LONG NAME TYPE OF DATA BLOCK IS
	STB, R11	*R4	SET IN TYPE-FIELD OF NEW D.B.
	AI, R4	2	PT AT 1ST NAME WD POS IN NEW D.B.
	OR, R4	NAMEWDS	FILL IN WORD COUNT (= 2 TO 20) AND
	STW, R4	*SYMT, R6	SET NEW NAME INDICATOR WORD.
	LB, R7	NAMEWDS	PREPARE TO PUT NEW NAME IN THE D.B.
	AI, R4	-1	PT AT REF WD IN NEW DATA BLOCK.
NEWNAMWD	LW, R10	NAMEBUF-1, R7	PUT NEW NAME IN LONG NAME DATA BLOCK
	STW, R10	*R4, R7	FROM LAST NAME WD TO FIRST.
	BDR, R7	NEWNAMWD	
	B	FOUND	WE'VE FOUND A NEW NAME.
RESHORT	BAL, R7	NEWSTLOC	TRY ANOTHER HASH LOC.
SHORT	LW, R7	*SYMT, R6	GET A NAME INDICATOR WD.
	BEZ	NEWSNAME	NEW SHORT NAME.
	CW, R7	NAMEBUF	OLD ENTRY, DOES IT MATCH THIS NAME..
	BNE	RESHORT	NO, TRY AGAIN.
FOUND	AI, R6	-1	PT AT REFERENT-INDICATOR WD (I.E.
*			LOC RELATIVE TO 'SYMT')
	AI, R12	2	SET FOR NORMAL RETURN.
	B	*R12	EXIT.
NEWSNAME	CI, R12	FINDNAME+1	WAS ACQNAME CALLED BY FINDNAME...
	BE	*R12	YES -- TAKE 'NAME-NOT-FOUND' EXIT.
	LW, R7	NAMEBUF	NO, GET NEW SHORT NAME & USE IT AS
	STW, R7	*SYMT, R6	THE NEW NAME INDICATOR WORD.
	B	FOUND	WE'VE FOUND A NEW NAME.

PAGE

```

*****
*
* FINDNAME -- FINDS A NAME (SEE 'ACQNAME' FOR ENTRY & EXIT SET-UPS).
* R8 IS THE LINK -- IF NAME-NOT-FOUND, RETURN-0
* -- IF NAME FOUND, RETURN-1
*
* WHEN 'FINDNAME' IS USED DURING COPYING, R6 PTS AT THE NAME-
* INDICATOR WD OF AN EMPTY SYMBOL TABLE ENTRY IF THE NAME-NOT-
* FOUND RETURN OCCURS.
* (SEE ALSO 'ACQNAME' FOR OTHER REGISTER USAGES).
*

```

FINDNAME	BAL, R12	ACQNAME	LOOK FOR THE NAME...
	B	*R8	NOT FOUND (NEW NAME)
	B	*R8	NOT FOUND (IMPOSSIBLE -- WS FULL)
	AI, R8	1	FOUND.
	CI, R13	NAMECODE	BUT IS IT A NORMAL NAME...
	BE	*R8	YES.
	BDR, R8	*R8	NO, STOP OR TRACE NAME (ASSUME
			NAME NOT FOUND).

PAGE

```

*
* ACQIT -- ACQUIRES A NAME OR NUMERIC ITEM, IF ANY, ENDING ON THE NEXT
* NON-BLANK AFTER THE ITEM. ITEM PUT IN NAMEBUF, INIT. BLANKED.
* REGS:  R14 -- LINK, EXIT IS VIA 'ACQNB'
*        R1  -- (ENTRY) PTS TO 1ST CHAR OF ITEM, IF ANY (AT
*              THE VERY LEAST, PTS TO A NON-BLANK).
*        (EXIT) PTS TO NON-BLANK DELIMITER FOR ITEM.
*        R2  -- (EXIT) CONTAINS THAT DELIMITER.
*        R3  -- (EXIT) CONTAINS ITS CODE.
*        R5  -- (EXIT) CONTAINS NO.OF CHARS MAKING UP THE ITEM.
*        R4 AND R8 ARE VOLATILE.
*
ACQIT  AI,R1    -1          BACK UP MOMENTARILY.
      LI,R5    -NAMEWDSZ   = NO.OF WDS TO HOLD MAX POSS. NAME.
      LW,R8    BLANKS      BLANK THE ENTIRE NAME BUFFER.
ACQITB STW,R8   NAMEBUF+NAMEWDSZ, R5
      BIR,R5   ACQITB      (R5 ENDS UP AT ZERO).
      LI,R8   NAMLIMIT    = MAX ACCEPTED CHARS PER NAME.
      BAL,R4   ACQNXCC     ACQ NEXT CHAR & ITS CODE.
      CLM,R3   NONAME      TEST FOR NAME-CHAR (INCLUDING DIGIT)
      BCR,9    ACQITZ      NO.
      STB,R2   NAMEBUF,R5  YES, PUT CHAR IN NAME BUFFER.
      AI,R5    1           COUNT THAT CHAR (= OFFSET TO NEXT
*                               BYTE IN NAMEBUF AS WELL).
      BDR,R8   ACQNXCC     LOOP TILL MAX CHAR RUNOUT.
      BAL,R4   ACQNXCC     GET NEXT CHAR & CODE.
      CLM,R3   NONAME      CHECK IT...
      BCS,9    ACQNXCC     NAME-CHAR OR DIGIT, SKIP & RETRY.
ACQITZ LW,R4    R14        SWITCH LINKAGE TO EXIT AFTER
      B        ACQNB      SKIPPING BLANKS.
PAGE
*****
*
* MAYDREF -- EXTRACTS ADDR. FIELD OF R4 AND EITHER EXITS IF ZERO OR
* ELSE ENTERS DREF.
* DREF -- DEREFERENCES THE DATA BLOCK POINTED TO BY R4. IF THE REF-
* COUNT OF THAT BLOCK DECREASES TO ZERO, THE BLOCK IS RETURNED
* TO THE FREE TABLE. HOWEVER, FUNCTION DESCRIPTARS AND LISTS
* CANNOT BE FREED UNTIL THE DATA BLOCKS THEY REFERENCE HAVE
* BEEN DEREFERENCED.
* REGS:  R7 -- LINK (EXIT VIA 0,R7)
*        R4 -- (ENTRY) PTS AT DATA BLOCK TO BE DEREFERENCED.
*        R4 IS VOLATILE, ALL OTHER REGS ARE PRESERVED.
*
MAYDREF AND,R4    X1FFFF    EXTRACT ADDRESS FIELD.
      BEZ      0,R7        EXIT IF NIL.
DREF    MTW,-1    1,R4      DECR DATA BLOCK'S REF-COUNT.
      BGZ      0,R7        EXIT IF STILL BEING REFERENCED.
      LCI      6           NO LONGER NEEDED -- SAVE REGS.
      STM,R5    DREFSAVE
      LB,R5     *R4        IS THE DATA BLK A LIST OR FUN.DESCR.
      AI,R5     -TYPELIST
      BLZ      DFREE      NO, ORDINARY DATA.
      BEZ      DLIST      YES, A LIST.
      AI,R5     TYPELIST
      CLM,R5    FUNTYPES
      BCS,9     DFREE      NO, GROUP, CODESTRING, OR INTRINS.
      STW,R4    DBROOT    YES, FUN.DESCR, SAVE PTR TO IT.
      LI,R9     2         REF IT TWICE -- DBROOT & DBSERIES.
      STW,R9    1,R4
      AI,R4     XSIZOFF    PT AT XSIZE WD.
      LI,R9     X'E0000'
      CW,R9     0,R4      DOES IT PT AT ERR-CTRL TBL...
      BAZ      DFNFL      NO.
      STW,R4    DBSERIES  YES, SAVE PTR TO XSIZE WD.
      BAL,R10   DSDREF    DE-REF THE ERR-CTRL DATA BLK.
      LW,R4     DBROOT    PT AT FUN.DESCR. AGAIN.
      LI,R6     NFLOFF+1  = OFFSET TO 1ST LINE PTR WD.
      AI,R4     NFLOFF    PT AT NO.OF FUNCTION LINES WD.
      LW,R9     0,R4
      BEZ      DSOUT      NONE, FREE THE FUNC.DEScriptor.
      AI,R4     1         PT AT FUNC.LINE PTR WD, AND MAKE IT
      MTW,-2    0,R4      AIM AT ITS DATA BLOCK HDR.
      BDR,R9    DFLPTR
      LW,R4     DBROOT    RESTORE PTR TO FUNCTION DESCRIPTOR.
      B        DSER      HANDLE DATA BLOCK PTR SERIES.
      LI,R6     LISTLOFF+1 OFFSET TO 1ST DB PTR IN THE LIST.
      STW,R4    DBROOT    SAVE PTR TO ROOT OF THE LIST.

```

	LI, R9	2	WE WILL REF. ROOT TWICE FOR NOW (IN	
	STW, R9	1, R4	'DBROOT' AND 'DBSERIES').	U06-0014
DSER	AW, R4	R6	PT AT 1ST DB PTR IN THE SERIES.	
	STW, R4	DBSERIES	SAVE IT AS THE SERIES PTR.	
	LW, R9	-1, R4	GET LENGTH OF THE SERIES...	
	BEZ	DSOUT	0 -- FREE THE ROOT, IT DOESN'T PT.	
	BAL, R10	DSDREF	DE-REF THE 1ST DB THE ROOT PTS AT.	
	MTW, 1	DBSERIES	INCR SERIES PTR.	
	AI, R9	-1	DECR LENGTH OF SERIES REMAINING...	
	BGZ	DSDREF	ANOTHER PTR EXISTS, LOOP TO DSDREF	
	LB, R9	*DBROOT	END-SERIES -- TEST ROOT'S TYPE...	
	AI, R9	-TYPELIST		
	BEZ	DENDLIST	A LIST HAS ENDED.	
	LI, R9	X'FFFF'	END OF FUN.DESCR LINE PTR SERIES.	
	AND, R9	*DBSERIES	DO LABELS FOLLOW...	
	BEZ	DSOUT	NO -- FREE THE FUN.DESCRITOR.	
DFLBL	MTW, 2	DBSERIES	YES, PT AT DB PTR FOR A LABEL.	
	BAL, R10	DSDREF	DE-REF THE LABEL'S SCALAR DATA BLK.	
	AI, R9	-1	WAS THAT THE LAST LABEL...	
	BGZ	DFLBL	NO, LOOP BACK.	
DSOUT	LI, R4	0		
	STW, R4	DBSERIES	CLEAR 'DBSERIES'.	
	XW, R4	DBROOT	CLEAR 'DBROOT' AND PT TO ROOT DB.	
	B	DFREE	FREE THE ORIG. LIST OR FUN.DESCR.	
DSDREF	LI, R4	0	CLEAR AND EXTRACT THE DB PTR	
	XW, R4	*DBSERIES	CONTAINED IN THIS SERIES.	
	AND, R4	X1FFFF		
	BEZ	*R10	NIL -- RETURN.	
	MTW, -1	1, R4	DECR ITS REF-COUNT...	
	BGZ	*R10	STILL BEING REF'D -- RETURN.	
	LB, R5	*R4	NO LONGER NEEDED, IS THIS ALSO LIST	
	AI, R5	-TYPELIST		
	BNEZ	DFREE	NO, FREE THAT DATA BLOCK.	
	STW, R4	*DBSERIES	YES, RESTORE ITS DB PTR. (WE HAVE	
	MTW, 2	1, R4	A TREE). WE WILL REFERENCE THE	
*			NEW LIST TWICE (IN OLD LIST AND	
*			IN 'DBSERIES').	
	AI, R4	LISTLOFF+1	OFFSET TO 1ST DB PTR IN NEW LIST.	
	LW, R5	-1, R4	GET LENGTH OF NEW LIST.	
	BEZ	EMPTY	0 -- FREE NEW LIST, RESUME OLD.	
	XW, R9	-1, R4	SAVE OLD LENGTH REMAINING, SET NEW.	
	XW, R4	DBSERIES	SET NEW SERIES PTR, GET OLD ONE.	
	AW, R5	DBSERIES	PT AT NEW LIST'S TRACKING WORD, AND	
	STW, R4	0, R5	SAVE OLD SERIES PTR THERE (OLD PTR	
*			ACTUALLY AIMS AT THE DB PTR TO	
*			THE NEW LIST -- RESTORED ABOVE).	
	B	DSDREF	START WORKING THE NEW SERIES.	
DENDLIST	LW, R6	*DBSERIES	PICK UP LIST'S TRACKING WORD (THE	
	BEZ	DSOUT	DEFAULT VALUE IS ZERO, SO ZERO	
*			INDICATES THE ROOT LIST; HOWEVER,	
*			A SUB-LIST'S TRACKING WD PTS BACK	
*			TO THE PRIOR NODE WHICH CONTAINS	
*			A DB PTR TO THAT SUB-LIST -- SEE	
*			CODE IN DSDREF).	
	STW, R6	DBSERIES	RESTORE PRIOR SERIES PTR (TO THE	
*			LIST JUST ENDED).	
	LW, R6	0, R6	PT AT LIST JUST ENDED. WE SAVED	
	LW, R9	LISTLOFF, R6	PRIOR LENGTH-REMAINING THERE;	
*			RESTORE IT.	
EMPTY	LI, R4	0	CLEAR AND GET THE DB PTR THAT AIMED	
	XW, R4	*DBSERIES	AT THE SUB-LIST TO BE FREED.	
*	(FALL INTO	DFREE)	NOTE--ITS REF-COUNT IS LEFT = 2.	
DFREE	AND, R4	X1FFFF	USE ADDRESS FIELD ONLY.	
	LI, R5	FREETBL-2	SET PTR TO FREE TABLE AND	
	INT, R7	*R4	SIZE OF BLK TO BE FREED.	
	AWM, R7	FREETOTL	INCR TOTAL FREE SPACE NOW.	
DSRCH	AI, R5	2	PT AT LOC WD OF NEXT ENTRY.	
	LW, R6	0, R5	GET THAT LOC...	
	BEZ	DINSERT	0 MEANS WE'LL INSERT A NEW ENTRY.	
	CW, R6	R4	TEST LOC VERSUS NEW LOC TO FREE...	
	BG	DFHI	HI MEANS INSERT OR BACK-COALESCE.	
	AW, R6	1, R5	LO -- ADD SIZE OF FREE-TBL ENTRY.	
	CW, R6	R4	DOES IT FORWARD-COALESCE WITH NEW...	
	BNE	DSRCH	NO, TRY NEXT FREE-TABLE ENTRY.	
	AWM, R7	1, R5	YES, LENGTHEN THAT ENTRY.	
	AW, R6	R7	DOES THAT ENTRY NOW MERGE WITH NEXT	
	CW, R6	2, R5	FREE-TABLE ENTRY (2-WAY COALESCE)	
	BNE	DEND	NO.	

	LW, R6	3, R5	YES, GET NEXT ENTRY'S SIZE.
	AWM, R6	1, R5	ADD IT TO CURRENT ENTRY'S SIZE.
DSHRINK	SLS, R5	-1	PT AT CURRENT DBLWD ENTRY.
	AI, R5	1	PT AT NEXT DBLWD ENTRY.
	LD, R6	2, R5	MOVE ITS SUCCESSOR UP ONE ENTRY.
	STD, R6	0, R5	
	BNEZ	DSHRINK	KEEP MOVING UNTIL REACHING 0,0
*			(WHICH ENDS THE FREE-TBL).
	B	DEND	
DCOALBAK	AWM, R7	1, R5	LENGTHEN FREE-TABLE ENTRY, AND SET
	STW, R4	0, R5	ITS LOC TO BE NEW FREE BLK LOC.
DEND	LW, R4	DBROOT	IS THIS A LIST OR FUN.DESRIPTOR...
	BNEZ	*R10	YES, RESUME SERIES WORK.
	LCI	6	NO, RESTORE REGS.
	LM, R5	DREFSAVE	
	B	0, R7	EXIT.
DFHI	SW, R6	R7	BACK UP CURRENT LOC BY NEW'S SIZE.
	CW, R6	R4	DO THE FREE BLOCKS JOIN...
	BE	DCOALBAK	YES, BACK-COALESCE.
DINSERT	AI, R5	2	PT AT CURRENT ENTRY'S SUCCESSOR.
	XW, R4	-2, R5	INSERT LOC & GET CURRENT LOC.
	XW, R7	-1, R5	INSERT SIZE & GET CURRENT SIZE.
	BNEZ	DINSERT	KEEP GOINT TILL ZERO IS GOTTEN.
	AI, R5	-FREETBL-MAXFRENS-MAXFRENS	DID FREE TABLE FILL UP...
	BLZ	DEND	NO.
	LI, R8	DEND	YES, RETURN TO 'DEND' AFTER
	B	GARBCOLL	GARBAGE COLLECTION.
PAGE			

*			
*	GARBCOLL -- PERFORMS GARBAGE COLLECTION, TERMINATING WITH A SINGLE		
*	FREE-TABLE ENTRY INDICATING THE TOTAL FREE SPACE BETWEEN THE		
*	LAST DATA BLOCK AND THE HIGH BOUND FOR DYNAMIC. THIS MAY		
*	REQUIRE EXTENSIVE DATA BLOCK POINTER ADJUSTMENTS (PERFORMED		
*	PRIOR TO MOVING THE DATA BLOCKS).		
*	REGS:	R8 -- LINK (EXIT VIA *LINKGC).	
*		R8 IS VOLATILE, ALL OTHER REGS ARE PRESERVED.	
*			
GARBCOLL	STW, R8	LINKGC	SAVE LINK.
	LCI	0	SAVE ALL REGS.
	STM, R0	GARBSAVE	
	LW, R8	FREETOTL	GET TOTAL AMT OF FREE SPACE.
	BEZ	*LINKGC	NONE -- EXIT.
	MTW, 0	FREETBL+2	IS THERE MORE THAN 1 FREE FRAGMENT.
	BNEZ	GFREE	YES, START WORKING ON FREE TABLE.
	AW, R8	FREETBL	NO, IS FREE BLK ADJACENT TO THE
	CW, R8	DYNBOUND	HIGH DYNAMIC BOUNDARY...
	BGE	*LINKGC	JA -- EXIT VERY GRATIFIED.
GFREE	LI, R8	0	CLEAR FREE-ACCUMULATION.
	LD, R2	ZEROZERO	R2 = 0 AND R3 = 0 (R3 WILL BE USED
*			LATER AS A REF-COUNT ACCUMULATOR).
	LI, R1	-1	PRESET R1, IT IS DUALY USED --
*			OFFSET FOR FREE-TBL MGMT. AND
*			NO.OF DATA REGIONS THAT NEED TO
*			BE MOVED.
GFSETS	AI, R1	1	OFFSET TO A FREE ENTRY.
	LD, R4	FREETBL, R1	GET LOC & SIZE OF FREE BLOCK.
	STD, R2	FREETBL, R1	CLEAR THAT FREE TABLE ENTRY.
	STW, R2	1, R4	SET FREE-BLK'S REF-COUNT TO ZERO;
	STW, R5	0, R4	MAKE IT A TYPE-ZERO DATA BLK
*			WHOSE SIZE IS FOR WHOLE FREE
*			ENTRY.
	AW, R4	R5	GET BOUNDARY FOR THIS FREE REGION.
	STW, R4	FBOUNDS, R1	
	AW, R8	R5	ACCUMULATE FREE FRAGS BELOW THIS
	STW, R8	FAQMS, R1	BOUNDARY.
	CW, R8	FREETOTL	HAVE WE REACHED TOTAL AMT FREE YET.
	BL	GFSETS	NO, WORK ON NEXT FREE-TABLE ENTRY.
	CW, R4	DYNBOUND	YES --IF LAST FREE REGION WAS AT
	BGE	GNRSET	END OF DYNAMIC, R1= NO.OF DATA
*			REGIONS TO BE MOVED.
	AI, R1	1	OTHERWISE, WE HAVE TO MOVE 1
*			MORE (THE REGION AT END).
	STW, R8	FAQMS, R1	ITS FREE-ACCUMULATION IS SAME & WE
*			WILL SET THAT BOUND WHEN WE ARE
*			DAMNED GOOD AND READY (WHICHEVER
*			HAPPENS FIRST).
GNRSET	STW, R1	NR2MOVE	= NO.OF DATA REGIONS TO BE MOVED.

REF	LW, R4	FBOUNDS	PT AT 1ST DATA BLOCK THAT WILL MOVE.
*	AW, R3	1, R4	ACCUMULATE NO.OF REFERENCES TO THE
*			DATA BLKS THAT WILL MOVE (WE WILL
*			HAVE TO FIND EACH SUCH REF. AND
			DISPLACE IT APPROPRIATELY).
	INT, R13	0, R4	GET THE SIZE OF THAT BLOCK.
	AW, R4	R13	PT AT ITS SUCCESSOR.
	CW, R4	DYNBOUND	HAVE WE HIT END OF DYNAMIC...
	BL	REF	NO, KEEP GOING.
	STW, R4	FBOUNDS, R1	READY -- THAT BOUNDS LAST REGION.
	LI, R9	X'1FFFF'	NOTE--USED BY 'DISPLACE' FOR SELECTIVE
*			LOADS & STORES, LEAVE R9 ALONE.
*			-----
	LI, R11	STRAYBLK	= TOTAL # OF STRAY DATA BLK PTRS.
	LI, R1	STRAYS	PT AT 1ST STRAY DATA BLK PTR.
	BAL, R7	DISPLACE	DISPLACE IT IF APPR.
	AI, R1	1	PT AT NEXT ONE.
	BDR, R11	DISPLACE	LOOP TILL DONE.
	LW, R1	TOPOSTAK	PT AT TOP ENTRY IN EXECUTION STACK.
	B	GSTAK	START LOOKING AT THE EXEC. STACK.
GDRPOP	AI, R1	2	SKIP BOTH WDS OF 0-CATEGORY.
	B	GSTAK	
GDROP	AI, R1	1	SKIP & PT AT WD HAVING A DB PTR.
G1	BAL, R7	DISPLACE	DISPLACE IT IF APPR.
GPOP	AI, R1	1	PT AT NEXT ENTRY IN EXECUTION STACK.
GSTAK	LB, R2	*R1	GET CATEGORY OF EXEC. STACK ENTRY.
	B	GSCAT, R2	VECTOR ACCORDING TO CATEGORY.
GSCAT	B	G1	@ V
	B	GPOP	@ A-PRIME
	B	GDRPOP	@ 0
	B	G1	@ X
	B	GPOP	@ B
	B	GPOP	@ P
	B	GPOP	@ S
	B	GDROP	@ Q
	B	G1	@ D
	B	GPOP	@ A
	B	GFCAT	@ F
*	B	GLC	@ LINE-CHAIN
GLC	BAL, R7	DISPLACE	@ DISPLACE ITS LINE-PTR IF APPR.
	AI, R1	3	PT AT NEXT ENTRY IN EXEC. STACK.
	B	GSTAK	
GFCAT	LI, R4	X'7FFF'	EXTR. ITS 'NEXT' FIELD. ZERO TELLS
	AND, R4	0, R1	US WE'VE HIT THE 'FINAL' ENTRY.
	BEZ	GSYM	OK, NOW CHECK THE SYMBOL TABLE.
	AI, R1	1	RATS, PT AT 'FDEFPTR' ENTRY.
	BAL, R7	DISPLACE	DISPLACE IT IF APPR.
	AI, R1	1	PT AT 'CALLPTR' ENTRY.
	BAL, R7	DISPLACE	
	AI, R1	3	PT AT 2ND WD PAST '# OF SHADOW PAIRS
	LW, R4	-2, R1	FOR THIS FUNCTION STATE', & GET #.
	BEZ	GFBU	NONE -- BACK UP 1 WD & CK IT.
	BAL, R7	DISPLACE	DISPLACE SHADOWED REFERENT-INDICATOR
*			IF APPROPRIATE.
	AI, R1	2	PT 2 ENTRIES FURTHER INTO STACK.
GFBU	BDR, R4	DISPLACE	LOOP IF ITS A SHADOWED REF-INDIC.
*	BDR, R1	GSTAK	PT AT WD AFTER FUNCTION-STATE BLOCK
			AND CHECK ITS CATEGORY.
GSYM	LW, R11	SYMTSIZE	= NO.OF DBLWDS IN SYMBOL TABLE.
*	LI, R4	NAMEWDSZ	= MAX # WDS TO HOLD A NAME (A 77-
			CHAR NAME OCCUPIES 20 WORDS).
	LW, R1	SYMT	PT AT 1ST WD OF SYMBOL TABLE (EVEN).
	LW, R12	BITPOS-12	(R-BIT FOR COPY REFERENT PTRS)
	LI, R13	X'40000'	(W-BIT FOR COPY NAME-INDICATORS)
GSYMW	LW, R6	1, R1	LOOK AT THIS ENTRY'S NAME-INDIC WD.
	BEZ	GSYMR	UNUSED NAME ENTRY.
	CB, R4	R6	USED. LONG OR SHORT NAME...
	BL	GSYMR	SHORT.
	CW, R13	0, R1	LONG, IS W-BIT SET...
	BANZ	GSYMR	YES, DON'T DISPLACE COPY PTR.
	AI, R1	1	NO, PT AT NAME-INDIC WD.
	BAL, R7	DISPLACE	DISPLACE LONG-NAME PTR, IF APPR.
	AI, R1	-1	PT AT REF-INDIC WD AGAIN.
GSYMR	CW, R12	0, R1	IS R-BIT SET...
	BANZ	GSYMU	YES, DON'T DISPLACE COPY PTR.
	BAL, R7	DISPLACE	NO, DISPLACE REF-PTR, IF APPR.
GSYMU	AI, R1	2	PT AT NEXT ENTRY.

```

BDR,R11 GSYMW LOOP TILL PAST SYMBOL TABLE.
B GDBT NOTE--END OF SYM TBL, R1 PTS AT 1ST DATA
* (OR FREE) BLOCK AUTOMATICALLY.
GDBLIST LW,R11 LISTLOFF,R1 = NO.OF DB PTRS IN LIST,
AI,R11 1 + 1 MORE FOR ITS TRACKING WORD.
STW,R1 GCTEMP SAVE PTR TO THIS LIST DATA BLK.
AI,R1 LISTLOFF+1 PT AT 1ST PTR WD IN THE LIST.
BAL,R7 DISPLACE DISPLACE IT IF APPR.
AI,R1 1 PT AT NEXT WD.
BDR,R11 DISPLACE LOOP TILL PAST THE TRACKING WORD.
GDBRR1 LW,R1 GCTEMP RESTORE DATA BLK PTR.
GNBLK INT,R13 0,R1 GET SIZE OF THIS BLOCK.
AW,R1 R13 PT AT NEXT BLOCK.
CW,R1 DYNBOUND HAVE WE EXCEEDED THE DATA BLK AREA.
BL GDBT NO.
BAL,R15 SYSTERR YES -- SYSTEM ERROR.
GDBT LB,R2 *R1 GET TYPE OF DATA (OR FREE) BLOCK.
GDBQ B GDBQ,R2 VECTOR ACCORDING TO TYPE.
B GNBLK 0 @ FREE
B GNBLK 1 @ LOGICAL DATA
B GNBLK 2 @ TEXT DATA
B GNBLK 3 @ INTEGER DATA
B GNBLK 4 @ REAL DATA
B GNBLK 5 @ INDEX-SEQUENCE DATA
B GDBLIST 6 @ * LIST
B GNBLK 7 @ CODESTRING
B GDBFUND 8 @ * FUNCTION DESCRIPTOR
B GDBFUND 9 @ * FUNCTION DESCRIPTOR
B GDBFUND A @ * FUNCTION DESCRIPTOR
B GDBFUND B @ * FUNCTION DESCRIPTOR
B GDBFUND C @ * FUNCTION DESCRIPTOR
B GDBFUND D @ * FUNCTION DESCRIPTOR
B GNBLK E @ INTRINSIC FUNCTION
B GNBLK F @ INTRINSIC FUNCTION
B GNBLK 10 @ INTRINSIC FUNCTION
B GNBLK 11 @ GROUP
B GNBLK 12 @ LONG-NAME
*
* NOTE--* INDICATES TYPES CONTAINING DB PTRS
*
GDBFUND STW,R1 GCTEMP SAVE PTR TO FUN.DESCR. DATA BLK.
AI,R1 XSIZOFF PT AT XSIZE WD.
LI,R11 X'E0000'
CW,R11 0,R1 DOES IT PT AT ERR-CTRL TBL...
BAZ GDBFUNDN NO.
BAL,R7 DISPLACE YES, DISPLACE IT IF APPR.
GDBFUNDN LW,R1 NFLOFF-XSIZOFF,R1 GET NO.OF LINES IN FUNCTION...
BEZ GDBRR1 NONE -- FORGET IT.
AI,R1 -XSIZOFF+NFLOFF+1 PT AT 1ST FUN.LINE DB PTR WD.
BAL,R7 DISPLACE DISPLACE IT IF APPR.
AI,R1 1 PT AT NEXT WD.
BDR,R11 DISPLACE LOOP TILL PAST LAST FUN.LINE PTR WD.
LI,R11 X'FFFF' EXTR. NO.OF LABEL ENTRIES.
AND,R11 0,R1
BEZ GDBRR1 NONE.
AI,R1 2 PT AT 1ST LABEL'S DB PTR WD.
BAL,R7 DISPLACE DISPLACE IT IF APPR.
AI,R1 2 PT AT NEXT ONE, IF ANY.
BDR,R11 DISPLACE LOOP TILL PAST LAST LABEL ENTRY.
B GDBRR1
*
* DISPLACE -- THIS IS A WEIRD ROUTINE -- HALF-OPEN, HALF-CLOSED.
* IT TESTS A DATA BLOCK POINTER TO DETERMINE IF IT REFERENCES A
* DATA BLOCK THAT WILL CHANGE ITS POSITION IN CORE DUE TO AN
* IMPENDING MOVE. IF NOT, THE ROUTINE SIMPLY EXITS. OTHERWISE
* THE POINTER IS ADJUSTED, DISPLACING ITS VALUE TO THE POSITION
* THAT DATA BLOCK WILL OCCUPY AFTER THE MOVE. THEN, THE
* ROUTINE DECREMENTS THE 'REFERENCE-ACCUMULATOR' WHICH INDICS
* THE NO.OF DATA BLK PTRS STILL REQUIRING ADJUSTMENT. IF MORE
* REMAIN, THE ROUTINE EXITS, BUT AFTER ADJUSTING THE LAST SUCH
* PTR THE ROUTINE GOES ON TO PERFORM THE MOVEMENT OF DATA BLK
* REGIONS; IT BECOMES AN OPEN ROUTINE AT THIS POINT, BY THE WAY.
* THE DISPLACEMENT STRATEGY IS TO ADJUST PTRS IN THE
* FOLLOWING ORDER:
*
* 1. STRAY PTRS (IN PROCESSOR CONTEXT),
* 2. PTRS IN THE EXECUTION STACK,
* 3. PTRS IN THE SYMBOL TABLE, AND
* 4. PTRS CONTAINED INSIDE DATA BLOCKS.

```

```

*      IF LUCKY, HOWEVER, THE REFERENCE-ACCUMULATOR WILL CLEAR      *
*      QUICKLY -- SHORT-CUTTING THIS PROCEDURE, POSSIBLY EARLY IN    *
*      STEP 1.                                                         *
*
*      REGS:  R7 -- LINK (EXIT VIA 0,R7 UNTIL R3 CLEARS, THEN THE     *
*              ROUTINE SWITCHES INTO OPEN PROCEDURE).                 *
*      R9 -- (ENTRY) MUST BE X'1FFFFF'                                *
*      R1 -- (ENTRY) CONTAINS LOC.OF WD CONTAINING THE DATA          *
*              BLOCK POINTER (OR NIL) TO BE TESTED.                  *
*      R3 -- REFERENCE-ACCUMULATOR.  IT BETTER BE CORRECT.          *
*      R6 AND R8  ARE VOLATILE.  -- -----                          *
*
DISPLACE  LS,R8      0,R1      EXTR. THE ADDR. INDICATED VIA R1.
          BEZ        0,R7      NIL -- EXIT.
          CW,R8      FBOUNDS    IS IT BELOW THE 1ST DATA BLK TO MOVE
          BL         0,R7      YES -- EXIT, THE DB PTR IS OK.
          LW,R6      NR2MOVE    NO, DB PTR REQUIRE ADJUSTMENT;
*                                     GET NO.OF DATA REGIONS TO BE MOVED.
DLOCQ     AI,R6      -1        FIND THE HIGHEST REGION THAT IS
          CW,R8      FBOUNDS,R6 ABOVE THE DB PTR ADDRESS; THAT'S
          BL         DLOCQ     THE REGION CONTAINING THE ADDR.
          SW,R8      FAQMS,R6  ADJ. ADDR BY THE FREE ACCUMULATION
          STS,R8     0,R1      BELOW THAT DATA REGION.
          AI,R3      -1        DECR # OF REFS REMAINING TO MOVED
          BNEZ       0,R7      BLKS, AND EXIT IF ANY REMAIN.
          STW,R3     GCTEMP    CLEAR THE MOVE-REGION COUNT.
GMVSET    MTW,1      GCTEMP    BUMP THE MOVE-REGION COUNT.
          LW,R2      GCTEMP
          LW,R1      FBOUNDS,R2 = BOUND OF NEXT FREE BLOCK.
          SW,R1      FAQMS,R2  -(FREE ACCUMULATION BELOW THAT BND).
          STW,R1     FREETBL   SET DESTINATION-BOUND FOR MOVE.
          AW,R1      FAQMS-1,R2 +(PRIOR FREE ACCUMULATION).
          STW,R1     FREETBL+1 SET SOURCE-BOUND FOR MOVE; IT IS
*                                     HIGHER THAN THE DESTINATION BOUND.
          LW,R1      FBOUNDS-1,R2 GET PRIOR FREE BLOCK BOUND.
          SW,R1      FREETBL+1  = -(NO.OF WDS TO MOVE).
GMVQ      CI,R1      -15       BIG OR SMALL BLK REMAINING...
          BLE        GMV15     BIG -- MOVE 15 WDS OF THE BLK.
          LCW,R2     R1        SMALL, GET SIZE OF BLK LEFT TO GO.
          SCS,R2     -4        MOVE SIZE TO BITS 0 - 3.
          LC         R2        SET COND. CODE = THAT SIZE.
          B          GMV
GMV15     LCI        15        SET COND. CODE FOR 15-WORD MOVE.
GMV        LM,R2     *FREETBL+1,R1 LOAD APPROACHING THE SOURCE-BOUND.
          STM,R2     *FREETBL,R1 STORE APPROACHING DESTINATION-BOUND.
          AI,R1      15        STEP UP BY 15 WORDS, AND LOOP UNTIL
          BLZ        GMVQ      LAST 1 TO 15 HAVE MOVED.
          MTW,-1     NR2MOVE    DECR # OF REGIONS REMAINING TO MOVE.
          BGZ        GMVSET    MORE, SET UP FOR MOVING NEXT REGION.
*                                     DONE -- FREETBL NOW CONTAINS LOC OF
*                                     1ST FREE WD IN COLLECTED FREE AREA
          LW,R1      FREETOTL   SET SIZE OF THAT FREE TABLE ENTRY TO
          STW,R1     FREETBL+1 THE TOTAL THAT WERE FREE ORIG'NLY.
*                                     (ALL OTHER FREETBL WDS WERE ZEROED).
          LCI        0          RESTORE REGS, EXCEPT FOR R8.
          LM,R0      GARBSAVE
          B          *LINKGC    EXIT -- GARBAGE HAS BEEN COLLECTED.
*
PAGE
*****
SPACE 2
Z      SET          %-WMAQ@      SIZE OF WMAQ IN HEX.
SPACE
Z      SET          Z+Z/10*6+Z/100*96+Z/1000*1536  SIZE IN DECIMAL.
SPACE 2
END

```

TITLE 'LIB-B00,08/22/73,DWG702985'
PAGE

*
*
* E X T E R N A L C O M M U N I C A T I O N
*
*

* DEFINITIONS
*

DEF	CIRCULAR	'CIRCULAR' FUNCTION EVAL
DEF	FEXP	FLOT EXPONENTIAL EVAL
DEF	FFLOG	FLOT DYADIC LOGARITHM
DEF	FFPOWER	FLOT BASE TO FLOT POWER EVAL
DEF	FIPOWER	FLOT BASE TO INTG POWER EVAL
DEF	FLOG	FLOT MONADIC LOGARITHM
DEF	FSQRT	FLOT SQUARE ROOT
DEF	IIPower	INTG BASE TO INTG POWER EVAL
DEF	LIB@	START OF PROCEDURE

* REFERENCES
*

REF	ERDOMAIN	DOMAIN ERROR	
REF	F2I	CONVERT FLOATING TO INTEGER	
REF	INTGOVFL	INTEGER OVERFLOW (DOMAIN CHANGE)	
REF	LIBTEMPS	TEMPS ARE IN WINDOW IN APLUTSI	U07-0004

PAGE

*
*
* A S S E M B L Y P A R A M E T E R S
*
*

SYSTEM SIG5F
PROGSECT CSECT 1

* REGISTERS
*

N	EQU	3	INDEX REG
LX	EQU	5	INDEX LINK REG
AI	EQU	7	LEFT ARG INTG
BI	EQU	9	RIGHT ARG INTG
CI	EQU	13	EXTRA INTG
AF	EQU	6	LEFT ARG FLOT
AF1	EQU	7	*
BF	EQU	8	RIGHT ARG FLOT
BF1	EQU	9	*
CF	EQU	12	EXTRA FLOT
CF1	EQU	13	*
FL	EQU	14	FLAG REG

PAGE

*
*
* P R O C S
*
*

TLOC SET 0 U07-0006

TEMP CNAME 1
DTEMP CNAME 2

DO1 NAME=2
TLOC SET TLOC+(TLOC&1) U07-0009
DISP TLOC U07-0010
LF EQU LIBTEMPS+TLOC U07-0011
TLOC SET TLOC+NAME U07-0012
PEND

*
*
EVEN CNAME 0
ODD CNAME 1
PROC
LF EQU %
ERROR,1,(CF(2)+NAME)&1 'REGISTER HAS WRONG PARITY'
PEND

*
*
EQUAL CNAME
PROC
LF EQU %

```

ERROR,1,1-(CF(2)=CF(3)) 'REGISTERS MUST BE EQUAL'
PEND
PAGE
*
*
*   F U N C T I O N   E V A L U A T O R S
*
*   THE FUNCTION EVALUATION SUBROUTINES MUST NOT Clobber
*   REGISTERS 1, 2, 4, 10, 11 (KNOWN BY XSEG-GENERATING
*   MODULES AS N, K, N1, N2, AND N3).
*
*
LIB@      USECT      PROGSECT
RES      0
START OF PROCEDURE

*   CIRCULAR FUNCTION JUMP TABLE
*
*   CALLED BY 'BAL,LX CIRCULAR,X' WITH LEFT ARG (INTEGER
*   BETWEEN -7 AND +7) IN X AND RIGHT ARG IN AF/AF1.
*   RESULT RETURNED IN AF/AF1.
*
CIRCULAR EQU      %+7
B      FARCTANH      -7
B      FARCCOSH      -6
B      FARCSINH      -5
B      FCIRCM4      -4
B      FARCTAN      -3
B      FARCCOS      -2
B      FARCSIN      -1
B      FCIRC0      0
B      FSIN      +1
B      FCOS      +2
B      FTAN      +3
B      FCIRC4      +4
B      FSINH      +5
B      FCOSH      +6
B      FTANH      +7
PAGE
*
*
*   DOUBLE PRECISION SINE      FSIN
*   DOUBLE PRECISION COSINE      FCOS
*
*   CONSTANTS
*
C0MC0      BOUND      8
S5      DATA      X'3A27311C',-X'3A27311C' +-RT(6/16**14)=0.9125056E-8
S4      DATA      X'C3C39FF1',X'C5110D64' -0.359864337061349258D-5
S3      DATA      X'3DA83C17',X'ED1FC206' 0.160441150747149909D-3
S2      DATA      X'401466BC',X'677587FA' 0.796926262460430410D-1
S1      DATA      X'BF5AA218',X'CED20DF6' -0.645964097506244317
S0      DATA      X'411921FB',X'54442D18' 1.57079632679489662
C5      DATA      X'C2E59363',X'E0BB8C31' -0.252001354549174792D-4
C4      DATA      X'3E3C3E9F',X'6C141B84' 0.919259950095279119D-3
C3      DATA      X'C0AA8B0E',X'09009BCE' -0.208634807349535199D-1
C2      DATA      X'4040F07C',X'206C1E1C' 0.253669507899865139
C1      DATA      X'BEEC42C3',X'3641BA75' -1.23370055013615135
C0      DATA      X'40FFFFFF',X'FFFFFFFD' 0.999999999999999953
MASK      DATA      X'3FFFFFFF'
MC6      DATA      X'C4830908' -C6
C6      DATA      X'3B7CF6F8' 0.465529873E-6
S6      DATA      X'3AF1F9C8' 0.563393613E-7
FSIN      SPACE      3
RES      0
CLM,AF      C0MC0
BCS,2      2A3
BCS,4      2A1
B      0,LX
RETURN X FOR SIN(SMALL X)
SPACE      3
FCOS      RES      0
LI,FL      X'100'
CLM,AF      C0MC0
BCS,2      2A4
BCS,4      2A2
LD,AF      ONE
RETURN 1.0D0

```


	B	0, LX	FOR COS(SMALL X)
2A1	LW, FL	ONE	INDICATE SIN(NEGATIVE)
2A2	FML, AF	M20PI	CONVERT TO QUADRANTS, CHANGE SIGN
	B	3A1	PROCEED
2A3	LI, FL	0	INDICATE SIN(POSITIVE)
2A4	FML, AF	20PI	CONVERT TO QUADRANTS
3A1	LB, N	AF	BIASED EXPONENT
	AI, N	-X'4E'	CHECK FOR SIGNIFICANCE
	BLZ	3A3	BRANCH IF OK
	B	ERDOMAIN	ERROR
3A3	AI, N	X'E'	UNBIASED EXPONENT
	BLZ	3A7	BRANCH IF < 1/16 QUADRANT
	SLS, N	2	MULTIPLY BY 4
	SLD, AF	0, N	SCALE B7
	AND, AF	MASK	MOD 4 NUMBER OF QUADRANTS
	AH, FL	AF	NUMBER OF QUADRANTS WITH COS MOD
	CI, FL	X'200'	CHECK UPPER/LOWER SEMICIRCLE
	BAZ	%+2	SKIP IF UPPER
	EOR, FL	ONE	INVERT SIGN CHANGE FLAG
	CI, FL	X'80'	FRACTION : 0.5
	BAZ	3A6	BRANCH IF < 0.5
	LI, N	X'BF'	INSERT
	STB, N	AF	EXPONENT
	CI, FL	X'4100'	CHECK SIGN CHANGE AND EVEN/ODD QUAD.
	BL	3A4	BRANCH IF NO SIGN CHANGE
	BANZ	4A6	BRANCH IF ODD QUADRANT
4A1	FML, AF	AF	H=F*F
4A3	STD, AF	TEMP0	COPY H
4A4	FMS, AF	MC6	NEGATIVE
	FSL, AF	C5	COSINE
	FML, AF	TEMP0	*H, -C(I)
	FSL, AF	C4	
	FML, AF	TEMP0	
	FSL, AF	C3	
	FML, AF	TEMP0	
	FSL, AF	C2	
	FML, AF	TEMP0	
	FSL, AF	C1	
	FML, AF	TEMP0	
	FSL, AF	C0	
	B	0, LX	RETURN
3A4	BAZ	4A11	BRANCH IF EVEN QUADRANT
3A5	LCD, AF	AF	NEGATIVE FOR SINE EXPANSION
4A6	RES	0	PLACE
4A8	STD, AF	TEMP0	COPY F
	FML, AF	TEMP0	H=F*F
	STD, AF	TEMP2	COPY H
4A9	FMS, AF	S6	SINE
	FAL, AF	S5	POLYNOMIAL
	FML, AF	TEMP2	*H, +S(I)
	FAL, AF	S4	
	FML, AF	TEMP2	
	FAL, AF	S3	
	FML, AF	TEMP2	
	FAL, AF	S2	
	FML, AF	TEMP2	
	FAL, AF	S1	
	FML, AF	TEMP2	
	FAL, AF	S0	
	FML, AF	TEMP0	*F
	B	0, LX	RETURN
3A6	LI, N	X'40'	INSERT
	STB, N	AF	EXPONENT
3A7	CI, FL	X'4100'	CHECK SIGN CHANGE AND ODD/EVEN QUAD.
	BL	3A8	BRANCH IF NO SIGN CHANGE
	BANZ	4A1	BRANCH IF ODD QUADRANT
	OR, AF1	=1	FORCE NON-ZERO MANTISSA
	B	3A5	NEGATIVE RESULT, USE SIN EXPANSION
3A8	BAZ	4A6	BRANCH IF EVEN QUADRANT
4A11	FML, AF	AF	H=F*F
4A13	STD, AF	TEMP0	COPY H
4A14	FMS, AF	C6	POSITIVE
	FAL, AF	C5	COSINE
	FML, AF	TEMP0	*H, +C(I)
	FAL, AF	C4	
	FML, AF	TEMP0	
	FAL, AF	C3	
	FML, AF	TEMP0	

```

FAL,AF C2
FML,AF TEMP0
FAL,AF C1
FML,AF TEMP0
FAL,AF C0
B 0,LX RETURN
PAGE

```

```

*
* DOUBLE PRECISION TANGENT FTAN
*
* CONSTANTS
*

```

```

OPEN COMC0
BOUND 8
X4140 DATA X'41000000',FS'0.25' 1,.25
A9 DATA,8 FL'1.75402692800053906'
A8 DATA,8 FL'1.20545135789649175'
A7 DATA,8 FL'1.27837471033592976'
A6 DATA,8 FL'1.27300750870003719'
A5 DATA,8 FL'1.27324841140728890'
A4 DATA,8 FL'1.27326113329539502'
A3 DATA,8 FL'1.27343712586329919'
A2 DATA,8 FL'1.27508201993348727'
A1 DATA,8 FL'1.29192819501250237'
A0 DATA,8 FL'1.57079632679489662'
COMC0 DATA X'3A1BB67B',-X'3A1BB67B' +-RT(3)/16**7=0.64523921E-8
20PI DATA X'40A2F983',X'6E4E4415' 2/PI=.636619772367581343
M20PI DATA X'BF5D067C',X'91B1BBEB' -2/PI
SPACE 3
FTAN RES 0 ENTRY
CLM,AF COMC0 CHECK FOR VERY SMALL X
BCS,2 2B1 BRANCH IF LARGE POSITIVE
BCS,4 1B1 BRANCH IF LARGE NEGATIVE
B 0,LX TAN(SMALL X)=X
1B1 LI,FL 1 INDICATE NEGATIVE X
FML,AF M20PI CONVERT TO QUADRANTS, CHANGE SIGN
B 3B1 PROCEED
2B1 LI,FL 0 INDICATE POSITIVE X
FML,AF 20PI CONVERT TO QUADRANTS
3B1 LB,N AF EXPONENT TO INDEX
CLM,AF X4140 X : 1.0,0.25
BCS,4 6B1 BRANCH IF X<0.25
BCR,2 5B1 BRANCH IF X<1.0
CI,N X'4E' CHECK MAGNITUDE
BL 4B1 BRANCH IF OK
B ERDOMAIN ERROR
4B1 SLS,N 2 MULTIPLY BY 4
SLD,AF 0,N SCALE B7
CW,AF =X'01000000' CHECK ODD/EVEN QUADRANT
BAZ %+2 SKIP IF EVEN
AI,FL 9 ODD QUAD - INVERT SIGN CHANGE
5B1 CW,AF =X'00800000' : 0.5
BAZ %+3 SKIP IF < 0.5
LCD,AF AF 1-FRACTION
EOR,FL =8 INVERT QUADRANT INDICATOR
LI,N X'40' INSERT
STB,N AF EXPONENT
6B1 RES 0 PLACE
6B3 CW,AF X4140+1 :0.25
BLE %+3 BRANCH IF OK
LI,N -1 INDICATE > 0.25
FSL,AF HALF F-0.5
STD,AF TEMP0 COPY X
FML,AF TEMP0 Z=X*X
STD,AF TEMP2 COPY Z
FML,AF A9 POLY-
FAL,AF A8 NOMIAL
FML,AF TEMP2 OF ODD POWERS
FAL,AF A7
FML,AF TEMP2
FAL,AF A6
FML,AF TEMP2
FAL,AF A5
FML,AF TEMP2
FAL,AF A4
FML,AF TEMP2
FAL,AF A3
FML,AF TEMP2

```

	FAL,AF	A2	
	FML,AF	TEMP2	
	FAL,AF	A1	
	FML,AF	TEMP2	
	FAL,AF	A0	
	FML,AF	TEMP0	
	AI,N	0	X
	BGEZ	6B5	CHECK FOR 0.5
	LCD,BF	AF	BRANCH IF NOT
	FAL,BF	ONE	-V (POSITIVE)
	FAL,AF	HALF	1-V
	FAL,AF	HALF	ADD
	FDL,AF	BF	1.0D0
6B5	RES	0	(1-V)/(1+V)
6B7	CI,FL	5	PLACE
	BL	8B1	IFLAG : 5
	BAZ	7B1	BRANCH IF ABS(RESULT)<1
	LCD,BF	AF	BRANCH IF NON-NEGATIVE RESULT
	B	7B2	-P
7B1	LD,BF	AF	PROCEED
7B2	LD,AF	ONE	P
	FDL,AF	BF	
	B	0,LX	RESULT
8B1	BAZ	0,LX	RETURN
	LCD,AF	AF	RETURN IF POSITIVE
	B	0,LX	CHANGE SIGN
	PAGE		RETURN
*			
*	DOUBLE PRECISION ARCSINE		FARCSIN
*	DOUBLE PRECISION ARCCOSINE		FARCCOS
*			
*	CONSTANTS		
*			
	BOUND	8	
X40	DATA	X'40000000'	BIAS BIT
X208	DATA	FX'32.5B7'	TO CORRECT BIAS AND ADD 1/2
ONEM	DATA	X'40FFFFFF',-1	1-2*(-56)
	SPACE	3	
FARCCOS	RES	0	ARCCOSINE ENTRY
	OR,LX	X40	INDICATE ARCCOSINE
	SPACE	3	
FARCSIN	RES	0	ARCSINE ENTRY
1C2	STD,AF	TEMP6	COPY X
	LCD,BF	TEMP6	-X
1C3	BLEZ	1C5	BRANCH IF X>=0.0D0
	CD,BF	ONE	ABS(X) : 1.0D0
	BL	5C1	BRANCH IF LESS
	BE	3C1	BRANCH IF EQUAL
	B	ERDOMAIN	ERROR
1C5	CD,AF	ONE	X : 1.0D0
	BL	6C1	BRANCH IF LESS
	BG	ERDOMAIN	ERROR IF GREATER
	CW,LX	X40	CHECK FN. IND.
	BAZ	2C1	BRANCH IF ASIN
	LI,AF	0	ACOS(1)=0, LSH ALREADY 0
	B	0,LX	RETURN
2C1	LD,AF	PI2	ASIN(1)=PI/2
	B	0,LX	RETURN
3C1	CW,LX	X40	CHECK FN. IND.
	BAZ	4C1	BRANCH IF ASIN
	LD,AF	PI	ACOS(-1)=PI
	B	0,LX	RETURN
4C1	LCD,AF	PI2	ASIN(-1)=-PI/2
	B	0,LX	RETURN
5C1	CW,BF	X40	ABS(X) : 1/16
	BAZ	7C1	BRANCH IF LESS
	SLD,BF	-1	OBTAIN
	AW,BF	X208	(1+ABS(X))/2
	LI,N	X'40'	AND
	STB,N	AF	1-ABS(X)
	B	8C2	PROCEED
6C1	CW,AF	X40	X : 1/16
	BANZ	8C1	BRANCH IF NOT LESS
7C1	EQU	%	
	FML,AF	BF	-X*X
	FAL,AF	ONEM	1-X*X
	B	9C1	PROCEED
8C1	SLD,AF	-1	OBTAIN

	AW, AF	X208	(1+X)/2
	LI, N	X'40'	AND
	STB, N	BF	1-X
8C2	FML, AF	BF	(1-X*X)/2
	FAL, AF	AF	1-X*X
9C1	RES	0	PLACE
9C3	STW, LX	TEMP5	SAVE LINK
9C4	BAL, LX	FSQRT	Y=RT(1-X*X)
10A2	LW, LX	TEMP5	RESTORE LINK
	CW, LX	X40	CHECK FUNCTION INDICATOR
	BANZ	10C3	BRANCH IF ACOS
	LD, BF	AF	2ND ARG= Y
	LD, AF	TEMP6	1ST ARG = X
10C5	B	FARCTAN2	OBTAIN RESULT FOR ASIN
10C3	LD, BF	TEMP6	2ND ARG =X
10C4	B	FARCTAN2	OBTAIN RESULT FOR ACOS
	PAGE		

*
 * DOUBLE PRECISION ARCTANGENT-1 ARGUMENT, 2 QUADRANT RESOLUTN. FARCTAN
 * DOUBLE PRECISION ARCTANGENT-2 ARGUMENTS, 4 QUADRANT RESOLUTN. FARCTAN2
 *
 * CONSTANTS - DO NOT REORDER CONSTANTS
 *

	OPEN	MASK, A4, A3, A2, A1, A0	
	BOUND	8	
TABLE	DATA, 8	0	0.0D0 ATAN(0)
ONE	DATA, 8	FL'1.0'	1.0D0
	DATA	X'401FD5BA', X'9AAC2F6E'	.124354994546761435=ATAN(1/8)
PI2	DATA	X'411921FB', X'54442D18'	1.57079632679489656=PI/2
	DATA	X'403EB6EB', X'F25901BB'	.244978663126864154=ATAN(1/4)
PI	DATA	X'413243F6', X'A8885A30'	3.14159265358979312
	DATA	X'405BD865', X'07937BC2'	.358770670270572220=ATAN(3/8)
A4	DATA	X'401C37E9', X'6B3C1CDF'	.110228146248221286
	DATA	X'4076B19C', X'1586ED3E'	.463647609000806116=ATAN(1/2)
A3	DATA	X'BFDB6DE9', X'65E1CA41'	-.142854130388704964
	DATA	X'408F005D', X'5EF7F5A0'	.558599315343562436=ATAN(5/8)
A2	DATA	X'40333333', X'212A84EB'	.199999995801153276
	DATA	X'40A4BC7D', X'1934F709'	.643501108793284387=ATAN(3/4)
A1	DATA	X'BFAAAAAA', X'AAACEB70'	-.333333333331284232
	DATA	X'40B8053E', X'2BC2319E'	.718829999621624505=ATAN(7/8)
A0	DATA	X'40FFFFFF', X'FFFFFFF5'	.999999999999999840
	DATA	X'40C90FDA', X'A22168C2'	.785398163397448310=ATAN(1)
MASK	DATA	X'FFFE0000'	MODULO 1/8
	SPACE	3	
FARCTAN	RES	0	1 ARGUMENT ENTRY
	LD, BF	ONE	X=1.0D0
	SPACE	3	
FARCTAN2	RES	0	2 ARGUMENT ENTRY - USED BY
*			FARCSIN, FARCCOS
1D1	AI, AF	0	Y : 0.0
	BG	10D1	BRANCH IF Y>0
	BL	6D1	BRANCH IF Y<0
2D1	AI, BF	0	X : 0.0
4D1	BG	0, LX	Y=0, X>0, ATAN(Y/X)=0.0D0
	BL	12D1	BRANCH IF X<0
	B	ERDOMAIN	ERROR
6D1	AI, BF	0	X:0
	BGEZ	9D1	BRANCH IF X>=0
	LI, FL	3	INDICATE 3RD QUADRANT
	CW, AF	BF	MSH(Y) : MSH(X)
	B	13D2	PROCEED
9D1	LCD, AF	AF	ABS(Y)
	LI, FL	1	INDICATE 4TH QUADRANT
	B	13D1	PROCEED
10D1	LI, FL	0	ASSUME 1ST QUADRANT
	AI, BF	0	X : 0.0
	BGEZ	13D1	BRANCH IF X>=0
12D1	LI, FL	2	INDICATE 2ND QUADRANT
	LCD, BF	BF	ABS(X)
13D1	CW, BF	AF	MSH(X) : MSH(Y)
13D2	BGE	14D2	BRANCH IF FORMER >= LATTER
13D22	AI, FL	8	INDICATE NORTH/SOUTH OCTANT
	FDL, BF	AF	U=X/Y
	STD, BF	TEMP0	COPY U
	LW, AF	TEMP0	COPY MSH(U)
	B	15D1	PROCEED
14D2	FDL, AF	BF	U=Y/X
	STD, AF	TEMP0	COPY U

```

15D1      FAS,AF    ONE          SCALE U
          AI,AF    X'10000'      ROUND TO
          AND,AF   MASK         NEAREST EIGHTH
          LH,N     AF           INDEX TO TABLE
          FSS,AF   ONE          V
          LI,AF1   0            CLEAR LSH
          LCD,BF   AF           -V
          FML,BF   TEMP0        -V*U
          FSL,AF   TEMP0        V-U
          FSL,BF   ONE          DENOMINATOR
          FDL,AF   BF           (U-V)/(U*V+1.0D0)=Z
          STD,AF   TEMP0        COPY Z
          FML,AF   TEMP0        U=Z*Z
          STD,AF   TEMP2        COPY U
13D23     FML,AF   A4           POLYNOMIAL
          FAL,AF   A3           EXPANSION
          FML,AF   TEMP2        *U+A(I)
          FAL,AF   A2
          FML,AF   TEMP2
          FAL,AF   A1
          FML,AF   TEMP2
          FAL,AF   A0
          FML,AF   TEMP0        *Z
16D1      FAL,AF   TABLE-X'4110'-X'4110',N DATAN=TABLE(J)+DATAN(Z)
* ABOVE   HEX CORRECTIONS COMPENSATE FOR 1.0 ADDED TO INDEX ABOVE
16D12     CI,FL    6            CHECK OCTANT AND SIGN OF X
          BL      16D3          BRANCH IF EAST/WEST OCTANT
          BAZ     16D2          BRANCH IF X POSITIVE
          FAL,AF   PI2          +PI/2
16D14     AI,FL    -10
          BGZ     16D5          BRANCH IF Y NEGATIVE
16D15     B       0,LX         RETURN
16D2      FSL,AF   PI2          -PI/2
16D22     AI,FL    -8
          BEZ     16D5          BRANCH IF Y POSITIVE
16D23     B       0,LX         RETURN
16D3      BAZ     16D4          BRANCH IF X POSITIVE
          FSL,AF   PI          -PI
16D32     AI,FL    -2
          BEZ     16D5          BRANCH IF Y POSITIVE
16D33     B       0,LX         RETURN
16D4      RES     0            PLACE
16D42     AI,FL    0
          BEZ     0,LX         RETURN IF Y POSITIVE
16D5      LCD,AF   AF          CHANGE SIGN
          B       0,LX         RETURN
          PAGE

*
* DOUBLE PRECISION HYPERBOLIC SINE          FSINH
* DOUBLE PRECISION HYPERBOLIC COSINE      FCOSH
*
* CONSTANTS
*
          OPEN      A4,A3,A2,A1,A0
          BOUND     8
C1MC1     DATA    X'42AF5DC3',-X'42AF5DC3' +-175.36628-
MCC       DATA    -X'404285FC',X'404285FC' -+ 0.26
A4        DATA    X'3C2E4DF9',X'BDA9EEDB' .275996875723123788D-5
A3        DATA    X'3DD00CEF',X'963E69E9' .198412447744192579D-3
A2        DATA    X'3F22222',X'228C9C82' .833333333938590834D-2
A1        DATA    X'402AAAAA',X'AAAA9C45' .1666666666666615519
A0        DATA,8   FL'1.0'      1.0D0
TWENTY8   DATA    FS'28.0'      28.0
          SPACE     3
FSINH     RES      0            HYPERBOLIC SINE ENTRY
          CLM,AF    C1MC1        ABS(X) : 175.36628-
          BCR,6     2E1          BRANCH IF OK
          B         ERDOMAIN     ERROR
2E1       CLM,AF    MCC          ABS(X) : 0.26
2E3       BCS,8     7E1          BRANCH IF X > 0.26
          BCS,1     4E1          BRANCH IF X < -0.26
          STD,AF    TEMP4        COPY X
          FML,AF    TEMP4        Z=X*X
          STD,AF    TEMP6        COPY Z
2E4       FML,AF    A4           POLYNOMIAL
          FAL,AF    A3           OF
          FML,AF    TEMP6        ODD POWERS OF X
          FAL,AF    A2

```

	FML,AF	TEMP6	
	FAL,AF	A1	
	FML,AF	TEMP6	
	FAL,AF	A0	
	FML,AF	TEMP4	FINAL FACTOR
	B	0,LX	RETURN
4E1	FML,AF	MLN2INV	X=X/LN(2)
	STW,LX	TEMP4	SAVE LINK
4E2	CW,AF	TWENTY8	X : 28.0
	BL	6E1	BRANCH IF RECIPROCAL IS SIGNIFICANT
	FSL,AF	ONE	TO DIVIDE RESULT BY 2.0D0
5E1	BAL,LX	FEXP1	OBTAIN ABS(RESULT)
	LCD,AF	AF	SET NEGATIVE SIGN
5E3	B	*TEMP4	RETURN
6E1	BAL,LX	FEXP1	OBTAIN 2**X
	LCD,BF	ONE	-1.0
	FDL,BF	AF	-1.0/2**X
	FAL,AF	BF	2**X-1.0/2**X
	FML,AF	MHALF	RESULT
6E3	B	*TEMP4	RETURN
7E1	FML,AF	LN2INV	X=X/LN(2)
7E2	CW,AF	TWENTY8	X : 28.0
	BL	8E1	BRANCH IF RECIPROCAL IS SIGNIFICANT
7E3	FSL,AF	ONE	TO DIVIDE RESULT BY 2.0D0
7E4	B	FEXP1	OBTAIN RESULT
8E1	RES	0	PLACE
8E3	STW,LX	TEMP4	SAVE LINK
8E4	BAL,LX	FEXP1	OBTAIN 2**X
	LCD,BF	ONE	-1.0
9E7	FDL,BF	AF	-1.0/2**X
	FAL,AF	BF	2**X-1.0/2**X
	FML,AF	HALF	RESULT
8E6	B	*TEMP4	RETURN
	SPACE	3	
FCOSH	RES	0	HYPERBOLIC COSINE ENTRY
	LAD,AF	AF	ABS(X)
	CW,AF	C1MC1	X : 175.36628-
	BG	ERDOMAIN	BRANCH IF RESULT OVERFLOWS
	FML,AF	LN2INV	X=X/LN(2)
9E2	CW,AF	TWENTY8	X : 28.0
	BGE	7E3	BRANCH IF X >= 28.0
9E5	STW,LX	TEMP4	SAVE LINK
9E6	BAL,LX	FEXP1	OBTAIN 2**X
	LD,BF	ONE	1.0
	B	9E7	PROCEED
	PAGE		

*

* DOUBLE PRECISION HYPERBOLIC TANGENT

FTANH

*

* CONSTANTS

*

	OPEN	AA,BB,CC,DD,EE	
	BOUND	8	
KMK	DATA	X'4213687B',-X'4213687B'	+ -28LN(2)=19.408121
CMC	DATA	X'4058B90C',-X'4058B90C'	+ -LN(2)/2=0.34657359
20LN2	DATA	X'412E2A8E',X'CA5705FC'	2/LN(2)=2.88539008177792682
M20LN2	DATA	X'BED1D571',X'35A8FA04'	-2/LN(2)
AA	DATA,8	FL'3465.0'	3465.0D0
BB	DATA,8	FL'189.0'	189.0D0
CC	DATA,8	FL'10395.0'	10395.0D0
DD	DATA,8	FL'1260.0'	1260.0D0
EE	DATA,8	FL'21.0'	21.0D0
	SPACE	3	
FTANH	RES	0	ENTRY
	CLM,AF	KMK	ABS(X) : 28LN(2)
	BCR,6	2F1	BRANCH IF LESS
	BCS,2	1F1	BRANCH IF X IS POSITIVE
	LCD,AF	ONE	X<-28LN(2), RESULT=-1.0D0
	B	0,LX	RETURN
1F1	LD,AF	ONE	X>28LN(2), RESULT=1.0D0
	B	0,LX	RETURN
2F1	CLM,AF	CMC	ABS(X) : LN(2)/2
	BCS,2	4F1	BRANCH IF X>LN(2)/2
	BCS,4	5F1	BRANCH IF X<-LN(2)/2
3F2	STD,AF	TEMP4	COPY X
	FML,AF	TEMP4	Y=X*X
	STD,AF	TEMP6	COPY Y
	FML,AF	EE	

```

FAL,AF DD
FML,AF TEMP6
FAL,AF CC Q = C+Y*(D+Y*E)
LD,BF TEMP6
FAL,BF BB
FML,BF TEMP6
FAL,BF AA
FML,BF TEMP6 P=Y*(A+Y*(B+Y))
FAL,BF AF P+Q
FML,AF TEMP4 X*Q
FDL,AF BF RESULT
3F4 B 0,LX RETURN
4F1 FML,AF 20LN2 Y=2*X/LN(2)
4F3 STW,LX TEMP4 SAVE LINK
4F4 BAL,LX FEXP1 OBTAIN 2**Y = EXP(2*X)
FAL,AF ONE Y=EXP(2*X)+1.0D0
LCD,BF ONE
FDL,BF AF -1.0D0/Y
LD,AF HALF
FAL,AF BF 0.5D0-1.0D0/Y
FAL,AF AF RESULT
4F6 B *TEMP4 RETURN
5F1 FML,AF M20LN2 Y=2*X/LN(2)
5F3 STW,LX TEMP4 SAVE LINK
5F4 BAL,LX FEXP1 OBTAIN 2**Y = EXP(-2*X)
FAL,AF ONE Y=EXP(-2*X)+1.0D0
LD,BF ONE
FDL,BF AF 1.0D0/Y
LCD,AF HALF
FAL,AF BF 1.0D0/Y-0.5D0
FAL,AF AF RESULT
5F6 B *TEMP4 RETURN
PAGE
*
* DOUBLE PRECISION INVERSE HYPERBOLIC SIN FARCSINH
* DOUBLE PRECISION INVERSE HYPERBOLIC COSINE FARCCOSH
* DOUBLE PRECISION INVERSE HYPERBOLIC TANGENT FARCTANH
*
FARCSINH EQU % INVERSE SINH ENTRY
STW,LX LINKTEMP SAVE LINK
STD,AF TEMP6 SAVE X
BAL,LX FCIRC4 SQRT(X**2+1)
FAL,AF TEMP6 X+SQRT(X**2+1) >=0
BGZ 1Z1 IF X+SQRT(X**2+1) IS NEAR 0,
LCD,AF TEMP6 (AS IT WOULD BE FOR X<<0),
FSL,AF TEMP6 THEN ARCSINH(X)=-LOG(-2*X).
BAL,LX FLOG
LCD,AF AF
B *LINKTEMP
1Z1 BAL,LX FLOG ARCSINH(X)
B *LINKTEMP RETURN
*
FARCCOSH EQU % INVERSE COSH ENTRY
STW,LX LINKTEMP SAVE LINK
STD,AF TEMP6 SAVE X
BAL,LX FCIRCM4 SQRT(X**2-1), ABS(X)>=1
FAL,AF TEMP6 X+SQRT(X**2-1) >=1 OR <=-1
BAL,LX FLOG ARCCOSH(X) >=0 (OR DOMAIN ERR)
B *LINKTEMP
*
FARCTANH EQU % INVERSE TANH ENTRY
STW,LX LINKTEMP SAVE LINK
FSL,AF ONE X-1
STD,AF TEMP6
FAL,AF TWO X+1
FDL,AF TEMP6 (X+1)/(X-1) (NEGATIVE)
LCD,AF AF (1+X)/(1-X)
BAL,LX FLOG 2*ARCTANH(X)
FML,AF HALF ARCTANH(X)
B *LINKTEMP RETURN
PAGE
*
* SPECIAL CIRCULAR FUNCTIONS:
*
* SQRT(X**2 -1) FCIRCM4
* SQRT(X**2 +1) FCIRC4
* SQRT(1- X**2) FCIRC0

```

```

*
FCIRCM4 EQU % SQRT(X**2-1) ENTRY
        CLM,AF 2Z1
        BCR,9  IF ABS(X) LARGE ENOUGH,
        LAD,AF AF  USE F(X)=ABS(X).
        B 0,LX
2Z1 FML,AF AF X**2
    FSL,AF ONE X**2-1
    B FSQRT COMPUTE SQRT AND RETURN
*
FCIRC4 EQU % SQRT(X**2+1) ENTRY
        CLM,AF 2Z2
        BCR,9  IF ABS(X) LARGE ENOUGH,
        LAD,AF AF  USE F(X)=ABS(X).
        B 0,LX
2Z2 FML,AF AF X**2
    FAL,AF ONE X**2+1
    B FSQRT COMPUTE SQRT AND RETURN
*
FCIRC0 EQU % SQRT(1-X**2) ENTRY
        FML,AF AF X**2
        LCD,AF AF -X**2
        FAL,AF ONE 1-X**2
        B FSQRT COMPUTE SQRT AND RETURN
*
HIGHSQ BOUND 8
        DATA -X'47100000',+X'47100000'
        PAGE
*
* DOUBLE PRECISION EXPONENTIAL FEXP
* DOUBLE PRECISION NEGATIVE POWER OF TWO FEXP2
* DOUBLE PRECISION POSITIVE POWER OF TWO FEXP1
*
* CONSTANTS
*
        OPEN TABLE
        BOUND 8
TABLE DATA,8 FL'2',FL'4',FL'8',FL'16'
        DATA,8 FL'-2',FL'-1',FL'-.5',FL'-.25'
BIG DATA X'46100000',X'80000000' TO SCALE INTEGER & ROUND
C1MC2 DATA X'42AEAC4F',X'BD4C20D9' 174.67308,-179.87169
LN2INV DATA X'41171547',X'652B82FE' 1.44269504088896341=1/LN(2)
MLN2INV DATA X'BEE8EAB8',X'9AD47D02' -1/LN(2)
P00 DATA,8 FL'1513.9067990543389159'
P01 DATA,8 FL'20.202065651286927228'
P02 DATA,8 FL'-0.023093347753750233624'
Q00 DATA,8 FL'4368.2116627275584985'
Q01 DATA,8 FL'233.18421142748162379'
MHALF DATA,8 FL'-.5'
HALF DATA,8 FL'0.5'
        SPACE 3
FEXP RES 0 MAIN ENTRY
        CLM,AF C1MC2 CHECK RANGE
        BCR,6 2G1 BRANCH IF OK
        BCS,4 1G1 UNDERFLOW
        B ERDOMAIN ERROR
1G1 SD,AF AF SET UNDERFLOW RESULT TO 0.0D0
    B 0,LX RETURN
2G1 RES 0 PLACE
2G3 AI,AF 0 X : 0.0
    BGEZ 4G1 BRANCH IF X NON-NEGATIVE
    FML,AF MLN2INV X=-X/LN(2)
    SPACE 3
FEXP2 RES 0 NEGATIVE POWER OF 2.0 ENTRY
* USED BY FFPOWER
3G1 LI,N 1 INDICATE NEGATIVE X
    B 6G1 PROCEED
4G1 FML,AF LN2INV X=X/LN(2.)
    SPACE 3
FEXP1 RES 0 POSITIVE POWER OF 2.0 ENTRY
* USED BY FSINH, FFPOWER
5G1 LI,N 0 INDICATE POSITIVE X
6G1 STD,AF TEMP0 COPY X
    FAL,AF BIG SCALE INTEGER
    LI,AF1 0 CLEAR FRACTION
    LD,CF AF COPY J
    FSS,AF BIG NORMALIZE J
    FSL,AF TEMP0 -U, POST SHIFTS

```


7G1	STD, AF	TEMP0	COPY -U
	FML, AF	TEMP0	V=U*U
	STD, AF	TEMP2	COPY V
	LD, BF	TEMP2	COPY V
	FAL, BF	Q01	RATIONAL
	FML, BF	TEMP2	
	FAL, BF	Q00	
	FML, AF	P02	
	FSL, AF	P01	APPROXIMATION
	FML, AF	TEMP2	
	FSL, AF	P00	
	FML, AF	TEMP0	
8G1	EXU	OPN, N	-+ U
	FDL, AF	BF	+ - (2.0 ** (+-U))/2.0 -+ 0.5
	FSL, AF	HALF-2, N	+ - (2.0 ** (+-U))/2.0
	SLD, CF	-2	J/4
	SLS, CF	24	POSITION J/4 TO EXPONENT
	AW, AF	CF	16** (+- J/4)
	OR, N	CF1	
	SCS, N	2	MOD(J/4)
9G1	FML, AF	TABLE, N	FINAL FACTOR
9G3	B	0, LX	RETURN
OPN	FSL, BF	AF	
	FAL, BF	AF	
	PAGE		
*			
*	DOUBLE PRECISION NATURAL LOGARITHM		FLOG
*	DOUBLE PRECISION BASE 2 LOGARITHM		FLOG2
*			
*	CONSTANTS		
*			
	OPEN	C6, C5, C4, C3, C2, C1, C0, D4, D3, D2, D1, MASK	
	BOUND	8	
FOURRT2	DATA	X'415A8279', X'99FCEF32' 5.65685424949238019=4RT(2)	
C6	DATA	X'437C60DA', X'FF254B2A' 1990.05346598213023	
C5	DATA	X'432176F9', X'A61B2FED' 535.435949426841753	
C4	DATA	X'42A427D5', X'CF33479E' 164.155606222171279	
C3	DATA	X'4234C2EA', X'CF2396D6' 52.7613954030883501	
C2	DATA	X'4212776C', X'516A782D' 18.4664965519843255	
C1	DATA	X'417B1C27', X'70E727E0' 7.69437355139461230	
C0	DATA	X'415C551D', X'94AE0BFD' 5.77078016355585460	
D4	DATA	X'403920FC', X'72219AEB' .223159578193740481	
D3	DATA	X'40492475', X'F6A71ECB' .285712597578022078	
D2	DATA	X'40666666', X'6BBA27A8' .400000001240327858	
D1	DATA	X'40AAAAAA', X'AAAA50E8' .666666666666347771	
TWO	DATA, 8	FL'2.0'	
LN2	DATA	X'40B17217', X'F7D1CF7A' .693147180559945309=LN(2)	
C0C1	DATA	X'411185F1', X'40E9CA3B' 1.095, 1/1.095=0.913242	
MASK	DATA	X'80FFFFFF' MANTISSA	
X44	DATA	X'44000000' EXPONENT	
QTR	DATA	X'400000' 0.25B7	
	SPACE	3	
FLOG2	RES	0	
*		LOG BASE 2.0 ENTRY	
	AI, LX	X'80000' USED BY FFPOWER	
FLOG	RES	0 INDICATE LOG BASE 2	
1H3	LFI	0	
	CLM, AF	C0C1 SET FF=0, POST SHIFTS POSSIBLE	
	BCS, 6	3H1 X:1.095, 1/1.095	
	FSL, AF	HALF BRANCH IF OUTSIDE INTERVAL	
	FSL, AF	HALF X-1.0D0	
	LD, BF	AF RETAINING SIGNIFICANCE	
	FAL, BF	TWO COPY X-1	
	FDL, AF	BF X+1	
	STD, AF	TEMP0 Z=(X-1)/(X+1)	
2H1	FML, AF	TEMP0 COPY Z	
	STD, AF	TEMP2 Y=Z*Z	
	FML, AF	D4 COPY Y	
	FAL, AF	D3 P	
	FML, AF	D2 0	
	FAL, AF	D2 LY-	
	FML, AF	TEMP2 NOMIAL	
	FAL, AF	D1 OF	
	FML, AF	TEMP2 ODD	
	FAL, AF	TWO POWERS	
	FML, AF	TEMP0 OF	
2H3	BDR, LX	0, LX Z	
	FML, AF	LN2INV RETURN IF NATURAL LOGARITHM	
		CONVERT TO BASE 2.0	

3H1	B	1, LX	RETURN, BDR HAS REDUCED LX BY 1
	LB, CF	AF	EN=EXPONENT
	SLS, CF	10	SCALE B21
	AND, AF	MASK	EM=MANTISSA
3H2	BGZ	3H3	
3H3	B	ERDOMAIN	ERROR
	RES	0	PLACE
	CW, AF	HALF	EM:0.5
	BANZ	6H1	BRANCH IF 0.5<=EM<1
	AI, CF	-X'10180'	REMOVE BIAS, SUBTRACT 1.5
	CW, AF	QTR	EM:0.25
	BANZ	7H1	BRANCH IF 0.25<=EM<0.5
	CW, AF	TWO	EM:0.125
	BANZ	4H1	BRANCH IF 0.125<=EM<0.25
	SLD, AF	2	EM=EM*4.0D0
	AI, CF	X'10180'-X'10380'	SUBTRACT 3.5
	B	7H1	PROCEED
4H1	SLD, AF	1	EM=EM*2.0D0
	AI, CF	X'10180'-X'10280'	SUBTRACT 2.5
	B	7H1	PROCEED
6H1	SLD, AF	-1	EM=EM*0.5D0
	AI, CF	-X'10080'	REMOVE BIAS, SUBTRACT 0.5
7H1	EOR, CF	X44	INSERT EXPONENT
	SFS, CF	5	NORMALIZE
	LI, CF1	0	CLEAR LSH OF EN
	AD, AF	FOURRT2	EM+4*SQRT(2.)
	LCD, BF	FOURRT2	-4*SQRT(2.)
	FDL, BF	AF	Y-0.5D0
	FAL, BF	HALF	Y
	STD, BF	TEMP0	COPY Y
	FML, BF	TEMP0	Z=Y*Y
	STD, BF	TEMP2	COPY Z
	LD, AF	C6	POLYNOMIAL
	FML, AF	TEMP2	
	FAL, AF	C5	
	FML, AF	TEMP2	
	FAL, AF	C4	
	FML, AF	TEMP2	
	FAL, AF	C3	
	FML, AF	TEMP2	
	FAL, AF	C2	
	FML, AF	TEMP2	
	FAL, AF	C1	
	FML, AF	TEMP2	
	FAL, AF	C0	
	FML, AF	TEMP0	
	FAL, AF	CF	DLOG2(X)
7H3	BIR, LX	0, LX	RETURN IF LOG BASE 2
	FML, AF	LN2	CONVERT TO NATURAL LOGARITHM
	B	-1, LX	RETURN, BIR HAS INCREASED R6 BY 1
	PAGE		

*

*

* DYADIC LOGARITHM FFLOG

*

* CALLED WITH LEFT ARG IN AF/AF1 AND RIGHT ARG IN BF/BF1.
* RESULT IS RETURNED IN AF/AF1. LINK IS LX.

*

FFLOG	EQU	%	DYADIC LOG ENTRY
	STD, BF	TEMP6	SAVE B
	STW, LX	LINKTEMP	
	BAL, LX	FLOG	LOG(A)
	XW, AF	TEMP6	SAVE IT, RESTORE B
	XW, AF1	TEMP7	
	BAL, LX	FLOG	LOG(B)
	FDL, AF	TEMP6	LOG(B)/LOG(A) = LOG BASE A OF B
	B	*LINKTEMP	RETURN
	PAGE		

*

* DOUBLE PRECISION SQUARE ROOT FSQRT

*

* CONSTANTS

*

	OPEN	AA, BB, CC, DD	
X80F	DATA	X'80FFFFFF'	
AA	DATA	X'1BE10000'	.108902
BB	DATA	X'907C0'	.03527451
CC	DATA	X'12E20000'	.07376

DD	DATA	X'D5A80'	.05216
	SPACE	3	
FSQRT	RES	0	ENTRY, USED BY FARCSIN, FARCCOS
	LI,N	2	RESET FLAG
	LB,BF	AF	COPY EXPONENT
	AND,AF	X80F	EXTRACT MANTISSA
	BGZ	2K7	BRANCH IF OK
	BEZ	0,LX	RETURN 0 FOR DSQRT(0)
	B	ERDOMAIN	ERROR
2K7	CW,AF	=X'00C00000'	BRANCH IF
	BANZ	2K5	HI SCALE
	STD,AF	TEMP2	LO SCALE
	SLD,AF	2	*4
	STD,AF	TEMP0	HI SCALE
	LW,AF	TEMP2	RESTORE LO
	BDR,N	2K6	SET FLAG AND JOIN
2K5	STD,AF	TEMP0	HI SCALE
	SLD,AF	-2	/4
	STD,AF	TEMP2	LO SCALE
2K6	LH,AF1	TEMP0	HI SCALE MSQTR
	CI,AF1	X'80'	BRANCH IF
	BANZ	2K3	HIST SCALE
	MH,AF1	AA	LINEAR
	AW,AF1	BB	APPROXIMATION
	B	2K4	JOIN
2K3	MH,AF1	CC	LINEAR
	AW,AF1	DD	APPROXIMATION
2K4	STW,AF1	TEMP4	SQRT/8
	FDS,AF	TEMP4	FIRST
	AW,AF	TEMP4	NEWTON
	XW,AF	TEMP0	SQRT
	LW,AF1	TEMP1	/4
	FDL,AF	TEMP0	SECOND
	AD,AF	TEMP0	NEWTON
	XW,AF	TEMP2	SQRT
	XW,AF1	TEMP3	/2
	OR,AF	=X'10000000'	FOR EXPONENT GARBAGE
	FDL,AF	TEMP2	THIRD
	AD,AF	TEMP2	NEWTON
	SCS,BF	-1	HALVE EXPONENT,REMAINDER TO BIT 0
	AI,BF	X'20'	FIX BIAS, CHECK REMAINDER
	BGEZ	2K1	BRANCH IF NO REMAINDER
	SLD,AF	-4,N	MULTIPLY BY
	BIR,BF	2K2	2 OR 4 AND JOIN
2K1	SLD,AF	-2,N	NOP OR DIVIDE BY 2
2K2	STB,BF	AF	INSERT EXPONENT
	B	0,LX	RETURN
	PAGE		
*			
*	DOUBLE PRECISION RAISED TO A DOUBLE PRECISION POWER		FFPOWER
*			
*	CONSTANTS		
*			
MEXPOV	DATA	X'43103800'	259.5
EXPOV	DATA	X'42FC0000'	252.0
	SPACE	3	
FFPOWER	RES	0	ENTRY
	STW,LX	LINKTEMP	SAVE LINK
	STD,BF	TEMP6	COPY Y
	AI,AF	0	X : 0.0
	BGZ	4L1	BRANCH IF X > 0.0
	BLZ	2L1	BRANCH IF X < 0.0
1L1	AI,BF	0	Y : 0.0
	BGZ	0,LX	RETURN 0.0D0 FOR 0**POSITIVE
	BEZ	2M1	X=Y=0, X**Y=1
	B	ERDOMAIN	ERROR
2L1	STD,AF	TEMP4	SAVE X
	LD,AF	BF	Y HAD BETTER BE AN INTEGER
	BAL,LX	F2I	
	B	ERDOMAIN	(ERROR IF NOT)
	CI,AI	1	CHECK Y FOR ODD/EVEN = FNEG
	STCF	LINKTEMP	SAVE FNEG
	LCD,AF	TEMP4	ABS(X)
4L1	RES	0	PLACE
4L5	BAL,LX	FLOG2	OBTAIN LOG BASE 2 OF X
	LI,LX	WA(7L2)	FORCE RETURNS BELOW
4L6	FML,AF	TEMP6	Y*LOG2(X)
	BGEZ	6L1	BRANCH IF RESULT >= 1.0D0

	LCD,AF	AF	ABS(Y*LOG2(X))
	CW,AF	MEXPOV	CHECK FOR UNDER FLOW
5L1	BL	FEXP2	BRANCH IF OK
	SD,AF	AF	UNDERFLOW RESULT = 0.0D0
5L2	B	*LINKTEMP	RETURN
6L1	CW,AF	EXPOV	CHECK FOR OVERFLOW
7L1	BL	FEXP1	BRANCH IF OK
	B	ERDOMAIN	ERROR
7L2	RES	0	PLACE
7L4	LC	LINKTEMP	FNEG
	BAZ	*LINKTEMP	RETURN IF RESULT IS POSITIVE
	LCD,AF	AF	SET RESULT NEGATIVE
	B	*LINKTEMP	RETURN
	PAGE		

*

* DOUBLE PRECISION RAISED TO AN INTEGER POWER

FIPOWER

*

FIPOWER	RES	0	ENTRY
	AI,AF	0	X : 0
	BNEZ	1M1	BRANCH IF X NE 0
	AI,BI	0	J : 0
	BGZ	0,LX	RETURN 0 IF J<0
	BEZ	2M1	X=J=0, X**J=1
	B	ERDOMAIN	ERROR
1M1	AI,BI	0	J : 0
	BGZ	5M1	BRANCH IF J>0
	BLZ	3M1	BRANCH IF J<0
2M1	LD,AF	ONE	RESULT = 1.0D0
	B	0,LX	RETURN
3M1	STD,AF	TEMPO	
	LD,AF	ONE	X=
	FDL,AF	TEMPO	1.0
	LCW,BI	BI	J=-J
	B	5M1	ENTER LOOP
4M1	SLS,BI	-1	J=J/2
	FML,AF	AF	X=X*X
5M1	CI,BI	1	CHECK J FOR ODD/EVEN
	BAZ	4M1	LOOP IF EVEN
	BE	0,LX	RETURN IF J=1
	LD,CF	AF	COPY X
	B	7M1	ENTER LOOP
6M1	FML,AF	CF	CURRENT POWER IS A FACTOR
7M1	SLS,BI	-1	J=J/2
	FML,CF	CF	X=X*X
	CI,BI	1	CHECK J FOR ODD/EVEN
	BAZ	7M1	LOOP IF EVEN - NO FACTOR THIS TIME
	BNE	6M1	LOOP IF J>1
	FML,AF	CF	FINAL FACTOR
	B	0,LX	RETURN
	PAGE		

*

* INTEGER RAISED TO AN INTEGER POWER

IIPower

*

* CONSTANTS

*

ONEM1	BOUND	8	
	DATA	1, -1	+- 1
	SPACE	3	
IIPower	RES	0	ENTRY
	CLM,AI	ONEM1	ABS(I) : 1
	BCS,6	3N1	BRANCH IF GREATER
	BCR,3	0,LX	RETURN 1 IF I=1
	BCS,8	2N1	BRANCH IF I=0
	CI,BI	1	CHECK POWER FOR ODD/EVEN
	BANZ	0,LX	RETURN -1 IF ODD
1N1	LI,AI	1	RESULT = 1
	B	0,LX	RETURN
2N1	AI,BI	0	J : 0
	BGZ	0,LX	RETURN 0 FOR 0**POSITIVE
	BEZ	1N1	I=J=0, I**J=1
	B	ERDOMAIN	ERROR
3N1	AI,BI	0	J : 0
	BGZ	6N1	BRANCH IF J>0
	BEZ	1N1	BRANCH IF J=0
	B	INTGOVFL	WONT FIT AS INTEGER
5N1	SLS,BI	-1	J=J/2
	MW,AI	AI	I=I*I
	BOV	INTGOVFL	CHECK FOR OVERFLOW

6N1	CI, BI	1	CHECK J FOR ODD/EVEN
	BAZ	5N1	LOOP IF EVEN
	BE	0, LX	RETURN IF J=1
	LW, N	BI	MOVE J
	LW, BI	AI	COPY I
	B	8N1	ENTER LOOP
7N1	MW, AI	BI	CURRENT POWER IS A FACTOR
	BOV	INTGOVFL	CHECK FOR OVERFLOW
8N1	SLS, N	-1	J=J/2
	MW, BI	BI	I=I*I
	BOV	INTGOVFL	CHECK FOR OVERFLOW
	CI, N	1	CHECK J FOR ODD/EVEN
	BAZ	8N1	LOOP IF EVEN - NO FACTOR THIS TIME
	BNE	7N1	LOOP IF J>1
	MW, AI	BI	FINAL FACTOR
	BOV	INTGOVFL	CHECK FOR OVERFLOW
	B	0, LX	RETURN
	PAGE		

*
*
* TEMPS
*

TEMP0	TEMP
TEMP1	TEMP
TEMP2	TEMP
TEMP3	TEMP
TEMP4	TEMP
TEMP5	TEMP
TEMP6	TEMP
TEMP7	TEMP
LINKTEMP	TEMP

LINK TEMP

		ERROR, X'F', TLOC>9	'TOO MANY TEMPS'	U07-0014
NTEMPS	SET	TLOC		U07-0015
2Z	END			

TITLE 'OPR-B00,08/22/73,DWG702985'
PAGE

E X T E R N A L C O M M U N I C A T I O N

DEFINITIONS

DEF	BALCOMP	'BAL,LX FFCOMPAR' INSTRUCTION
DEF	BASEADR	BASE ADR FOR SHORT ADR OFFSETS
DEF	CHAREQ	CHARACTER EQUAL/NOT EQUAL
DEF	COMPINST	COMPARE INST TABLE
DEF	CONVTABL	TYPE CONVERSION CODE TABLE
DEF	DIVZERO	DIVIDE BY ZERO TRAP ROUTINE
DEF	DTYPEF	DYADIC FLOT TYPE SETUP
DEF	DTYPEIF	INTG/FLOT TYPE SETUP
DEF	DTYPEIF1	* (ALT. ENTRY)
DEF	DTYPEIF2	* (ALT. ENTRY)
DEF	DXDRIVER	DYADIC EXECUTION DRIVER
DEF	DXTABLE	DYADIC OP ROUTINE ENTRY TABLE
DEF	EXECUTE	EXECUTE XSEG
DEF	FLOTINF	FLOATING POINT INFINITY
DEF	FLOT0	FLOATING POINT 0.0
DEF	FLOT01	FLOATING POINT 0.0, 1.0
DEF	FLOT1	FLOATING POINT 1.0
DEF	FLOT2	FLOATING POINT 2.0
DEF	GENLOAD	GEN LOAD BY RSTYPE
DEF	GENLOADT	GEN LOAD TO TEMP
DEF	GXSEGDL1	GEN DYADIC LOAD (ALT. ENTRY)
DEF	GXSEGINI	GEN XSEG INITIALIZATION
DEF	GXSEGML	GEN MONADIC LOAD
DEF	GXSTEXC1	GEN STORE/EXECUTE (ALT. ENTRY)
DEF	GXSTEXEC	GEN XSEG STORE; EXECUTE XSEG
DEF	INTGOVFL	INTEGER OVERFLOW (DOMAIN CHANGE)
DEF	LFADR	LEFT ARG ADDRESS
DEF	LFLGLADR	LEFT LOGICAL ADDRESS
DEF	LFLGLCNT	LEFT LOGICAL BIT COUNT
DEF	LFRANK	LEFT ARG RANK
DEF	LFSIZE	LEFT ARG SIZE
DEF	LFTEMP	LEFT ARG VALUE TEMP
DEF	LFTYPE	LEFT ARG TYPE
DEF	LOADINST	LOAD INSTRUCTION TABLE, BY TYPE
DEF	LODBINST	LOAD 2ND ACCUM INST TABLE
DEF	LOOPLOC	LOOP LOCATION
DEF	MNOP	MONADIC NO OP ROUTINE
DEF	MTYPEF	MONADIC FLOT TYPE SETUP
DEF	MXDRIVER	MONADIC EXECUTION DRIVER
DEF	OPR@	START OF PROCEDURE
DEF	RSADR	RESULT ADDRESS
DEF	RSRANK	RESULT RANK
DEF	RSSIZE	RESULT SIZE
DEF	RSTYPE	RESULT TYPE
DEF	RTADR	RIGHT ARG ADDRESS
DEF	RTRANK	RIGHT ARG RANK
DEF	RTSIZE	RIGHT ARG SIZE
DEF	RTTEMP	RIGHT ARG VALUE TEMP
DEF	RTTYPE	RIGHT ARG TYPE
DEF	SETADR	SET UP ARG ADR CELL
DEF	SETADRS1	SET UP ADDRESS(ES), SEQUENTIAL
DEF	STCCSEQ	STORE CC CODE SEQ
DEF	STMPINST	STORE IN TEMP INST TABLE
DEF	STORINST	STORE INSTRUCTION TABLE
DEF	TYPETEMP	TYPE TEMP

U08-0004

REFERENCES

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS
REF	ALOCRS	ALLOCATE RESULT DATA BLOCK
REF	BITMASK	LOGICAL BIT SELECTION TABLE
REF	CIRCULAR	CIRCULAR FUNCTION EVAL
REF	CLOADLNK	DERAIL LINK FOR GXSEGDL
REF	COPTRIG	COMPOSITE OP TRIGGER
REF	CSETLNK	DERAIL LINK FOR DSETUP
REF	CSTORLNK	DERAIL LINK FOR GXSTEXEC
REF	CTYPELNK	DERAIL LINK FOR DTYPEIF
REF	DCATEN	DYADIC CATENATE OP ROUTINE
REF	DCOMPRES	DYADIC COMPRESS OP ROUTINE

REF	DDEAL	DYADIC DEAL OP ROUTINE	
REF	DDECODE	DYADIC DECODE OP ROUTINE	
REF	DDROP	DYADIC DROP OP ROUTINE	
REF	DENCODE	DYADIC ENCODE OP ROUTINE	
REF	DEXPAND	DYADIC EXPAND OP ROUTINE	
REF	DINDEXOF	DYADIC INDEX-OF OP ROUTINE	
REF	DMATDIV	DYADIC MATRIX DIVIDE OP ROUTINE	
REF	DMEMBER	DYADIC MEMBERSHIP OP ROUTINE	
REF	DREF	DE-REF	
REF	DRESHAPE	DYADIC RESHAPE OP ROUTINE	
REF	DROTATE	DYADIC ROTATE OP ROUTINE	
REF	DTAKE	DYADIC TAKE OP ROUTINE	
REF	DTBAR	DYADIC T-BAR OP ROUTINE	
REF	DTRANS	DYADIC TRANSPOSE ROUTINE	
REF	DXRETURN	DYADIC EXECUTION DRIVER RETURN	
REF	ERDOMAIN	DOMAIN ERROR	
REF	ERLENGTH	LENGTH ERROR	
REF	ERRANK	RANK ERROR	
REF	ERSYN	SYNTAX ERROR HANDLER	
REF	FCEILING	F CEILING EVAL	
REF	FEXP	F EXPONENTIAL EVAL	
REF	FFACT	F FACTORIAL EVAL	
REF	FFCOMB	F COMBINATORIAL EVAL	
REF	FFCOMPAR	F COMPARISON EVAL	
REF	FFLOG	F DYADIC LOG EVAL	
REF	FFLOOR	F FLOOR EVAL	
REF	FFPOWER	F POWER EVAL	
REF	FFRESIDU	F RESIDUE EVAL	
REF	FIPOWER	F TO I POWER EVAL	
REF	FLOG	F MONADIC LOG EVAL	
REF	FSQRT	F SQUARE ROOT EVAL	
REF	F2I	CONVERT F TO I	
REF	ICEILING	I CEILING EVAL	
REF	IFACT	I FACTORIAL EVAL	
REF	IFLOOR	I FLOOR EVAL	
REF	IICOMB	I COMBINATORIAL EVAL	
REF	IPOWER	I POWER EVAL	
REF	IIRESIDU	I RESIDUE EVAL	
REF	IROLL	I ROLL EVAL	
REF	LFARG	LEFT ARG PNTR	
REF	MDIMEN	MONADIC DIMENSION OP ROUTINE	
REF	MGRADEDN	MONADIC GRADE DOWN OP ROUTINE	
REF	MGRADEUP	MONADIC GRADE UP OP ROUTINE	
REF	MIBEAM	MONADIC I-BEAM OP ROUTINE	
REF	MINDEX	MONADIC INDEX GENERATOR OP ROUTINE	
REF	MMATINV	MONADIC MATRIX INVERT OP ROUTINE	
REF	MRVEL	MONADIC RAVEL OP ROUTINE	
REF	MREVERSE	MONADIC REVERSE OP ROUTINE	
REF	MTBAR	MONADIC T-BAR OP ROUTINE	
REF	MTRANS	MONADIC TRANSPOSE OP ROUTINE	
REF	MXRETURN	MONADIC EXECUTION DRIVER RETURN	
REF	OPBREAK	OP BREAK HANDLER	
REF	OPER	OPERATOR WORDS	
REF	OPRTEMPS	TEMPS ARE IN WINDOW IN APLUTSI	U08-0006
REF	RESULT	RESULT PNTR	
REF	RETURN	RETURN ADR CELL	
REF	RTARG	RIGHT ARG PNTR	
REF	SYSTERR	SYSTEM ERROR	
REF	XSEGBASE	XSEG BASE	
REF	XSEGBRK	XSEG BREAK FLAG	
	PAGE		

*
*
*
*
*

A S S E M B L Y P A R A M E T E R S

	SYSTEM	SIG5F	
PROGSECT	CSECT	1	
OPR@	RES	0	START OF PROCEDURE
BASEADR	EQU	%	BASE ADR FOR SHORT ADR OFFSETS
	REGISTERS		
IX	EQU	0	INTERPRET REG PAIR
IX1	EQU	1	*
N	EQU	1	XSEG EXECUTION-TIME INDEX
X	EQU	1	GENERAL INDEX REG
T	EQU	2	TYPE

XL	EQU	3	XSEG LOC	
A	EQU	4	ARG ADR/INDEX	
LX	EQU	5	INDEX LINK REG	
LX7	EQU	7	INDEX LINK REG	
OP	EQU	6	OP CODE INDEX	
AI	EQU	7	ACCUM FOR LOGL/CHAR/INTG VALUES	
AF	EQU	6	ACCUM FOR FLOT VALUES	
AF1	EQU	7	*	
BI	EQU	9	2ND ACCUM FOR LOGL/CHAR/INTG VALUES	
BF	EQU	8	2ND ACCUM FOR FLOT VALUES	
BF1	EQU	9	*	
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS	
R	EQU	8	GENERAL WORK REG	
S	EQU	11	SIZE	
L3	EQU	12	LINK REG	
L2	EQU	13	LINK REG	
L1	EQU	14	LINK REG	
* ARG TYPE CODES				
*				
WORDLOGL	EQU	0	WORD LOGICAL (WORD)	
LOGL	EQU	1	LOGICAL (BIT)	
CHAR	EQU	2	CHARACTER (BYTE)	
INTG	EQU	3	INTEGER (WORD)	
FLOT	EQU	4	FLOATING (DOUBLEWORD)	
ISEQ	EQU	5	INDEX SEQUENCE VECTOR	
LIST	EQU	6	LIST	
* CODESTRING DESIGNATIONS				
*				
MOPLOW	EQU	50	MONADIC OPS: LOW END	
MOPROLL	EQU	51	MONADIC ROLL	
MOPEXP	EQU	65	MONADIC EXPONENTIAL	
MOPCEIL	EQU	68	MONADIC CEILING	
MOPABS	EQU	70	MONADIC ABSOLUTE VALUE	
MOPFACT	EQU	71	MONADIC FACTORIAL	
DOPLOW	EQU	80	DYADIC OPS: LOW END	
DOPADD	EQU	91	DYADIC ADD	
DOPMUL	EQU	93	DYADIC MULTIPLY	
DOPDIV	EQU	94	DYADIC DIVIDE	
DOPPOWER	EQU	95	DYADIC POWER	
DOPMAX	EQU	98	DYADIC MAXIMUM	
DOPRESID	EQU	100	DYADIC RESIDUE	
DOPLESS	EQU	102	DYADIC LESS	
DOPNEQ	EQU	106	DYADIC NOT EQUAL	
DOPAND	EQU	108	DYADIC AND	
DOPNAND	EQU	110	DYADIC NAND	
PAGE				
* P R O C S				
* TLOC SET 0 U08-0008				
* TEMP CNAME 1				
DTEMP CNAME 2				
PROC				
DO1 NAME=2				
TLOC	SET	TLOC+(TLOC&1)	U08-0011	
	DISP	TLOC	U08-0012	
LF	EQU	OPRTEMPS+TLOC	U08-0013	
TLOC	SET	TLOC+NAME	U08-0014	
PEND				
* EVEN CNAME 0				
ODD CNAME 1				
PROC				
LF	EQU	%		
ERROR,1,(CF(2)+NAME)&1		'REGISTER HAS WRONG PARITY'		
PEND				
* EQUAL CNAME				
PROC				
LF	EQU	%		
ERROR,1,1-(CF(2)=CF(3))		'REGISTERS MUST BE EQUAL'		

```

*      PEND
*
EXCHANGE CNAME
GROUP   OPEN      I, K, GROUP
        EQU       LFARG, LFTYPE, LFSIZE, LFRANK, LFADR
        PROC
LF      EQU       %
I       DO        NUM(AF)
K       SET       SCOR(AF(I), ARGS, TYPES, SIZES, RANKS, ADRS)
        ERROR, 1, K=0      'UNKNOWN GROUP INDICATOR'
        LW, R      GROUP(K)
        XW, R      GROUP(K)+1
        STW, R     GROUP(K)
        FIN
        PEND
        CLOSE      I, K, GROUP
*
*
NB      CNAME     X'680'
NBGE    CNAME     X'681'
NBLE    CNAME     X'682'
NBE     CNAME     X'683'
NBL     CNAME     X'691'
NBG     CNAME     X'692'
NBNE    CNAME     X'693'
        PROC
        ERROR, 1, (AF>=0)+(NUM(AF)>1)  'AF MUST BE NEG CONST ADR'
LF      GEN, 12, 20  NAME-1, AF
        PEND
        PAGE
*
*
*      XSEG GEN PROCS
*
CODE    CNAME
        PROC
        DO        CF(2)>0
LF      GEN, 4, 12, 16  CF(2), CF(2), AF(1)-BASEADR
        ELSE
LF      GEN, 32      0
        FIN
        PEND
*
*
GENX    CNAME
        PROC
LF      INT, IX      AF
        BCR, 15     %+4
        LM, BUF     BASEADR, IX1
        STM, BUF    0, XL
        AW, XL      IX
        PEND
*
*
GEN     OPEN      GEN
        CNAME
        OPEN      M, N, MN, I
        PROC
LF      EQU       %
        ERROR, 1, 1-(NUM(CF)=3)  'WRONG NUMBER OF CF ARGS'
M       SET       CF(2)
N       SET       CF(3)
MN      SET       M+N
        ERROR, 1, 1-(NUM(AF)=(M>0)+(N>0))  'WRONG NUMBER OF AF ARGS'
        DO        M>0
I       DO        N
        LW, AF(1)+M+I-1  AF(2)+I-1
        FIN
I       DO        MN*(MN<3)
        STW, AF(1)+I-1  I-1, XL
        ELSE
        LCI        MN
        STM, AF(1)  0, XL
        FIN
        ELSE
I       DO        MN*(MN<3)

```

```

        LW, BUF      AF(1)+I-1
        STW, BUF     I-1, XL
    ELSE
        LCI          N
        LM, BUF      AF(1)
        STM, BUF     0, XL
    FIN
FIN
AI, XL      MN
PEND
CLOSE      M, N, MN, I
PAGE

*
*
*   TABLE BUILDING PROCS
*
TABLE      CNAME
OPEN      T, N
PROC
T          SET      %-AF(1)
LF         EQU      T
          DISP     T
          PEND
*
*
ITEM      CNAME
PROC
N          SET      T+AF(1)-%
          ERROR, 1, N<0      'ITEM OUT OF SEQUENCE'
          RES      N*(N>0)
          DISP     AF(1)
          PEND
          CLOSE    T, N
          PAGE

*
*
*   O P E R A T O R   E X E C U T I O N   D R I V E R S
*
*
*       USECT      PROGSECT
*
*   MONADIC EXECUTION DRIVER
*
*       ENTERED WITH OPTYPE IN 'OP' REGISTER, AND THE ARG
*       POINTER IN RTARG. IF NO ERRORS OCCUR, IT RETURNS
*       TO 'MXRETURN' WITH THE RESULT POINTER IN 'RESULT'.
*       ERRORS LEAD TO ERDOMAIN, ERRANK, OR ERLLENGTH.
*
*
MXDRIVER  EQU      %
          LI, A     MXRETURN      MONADIC EXECUTION DRIVER
          STW, A    RETURN        SET UP RETURN ADR
          CI, OP    DOPLOW        CHECK OP RANGE
          BL        MXTABLE, OP   LOW: GO DIRECTY INTO JUMP TABLE
          AI, OP    MOPLOW-DOPLOW HIGH: SHIFT TO LOWER RANGE FIRST
          STW, OP   OPER          SAVE MODIFIED OP (FOR 'INTGOVFL')
          B         MXTABLE, OP
*
MXTABLE   TABLE  MOPLOW      MONADIC OP JUMP TABLE
          B        MMATINV    MONADIC MATRIX INVERT
          B        MROLL      MONADIC ROLL
          B        MTBAR      MONADIC TYPE CONVERSION
          B        ERSYN      UNUSED (REDUCTION, 1ST COORD)
          B        ERSYN      UNUSED (REDUCTION)
          B        MINDEX     MONADIC INDEX GENERATOR
          B        MDIMEN     MONADIC DIMENSION
          B        MRABEL     MONADIC RABEL
          AI, OP    1          MONADIC REVERSE (1ST COORD)
          B        MREVERSE   MONADIC REVERSE
          B        MTRANS     MONADIC TRANSPOSE
          B        MNOP       MONADIC PLUS
          B        MMINUS     MONADIC MINUS
          B        MSIGNUM    MONADIC SIGNUM
          B        MRECIP     MONADIC RECIPROCAL
          B        MEXP       MONADIC EXPONENTIAL
          B        MLOG       MONADIC LOGARITHM

```

B	MPITIMES	MONADIC	PI TIMES
B	MCEILING	MONADIC	CEILING
B	MFLOOR	MONADIC	FLOOR
B	MABS	MONADIC	ABSOLUTE VALUE
B	MFACT	MONADIC	FACTORIAL
BAL,15	SYSTERR	UNUSED	
BAL,15	SYSTERR	UNUSED	
B	MIBEAM	MONADIC	I-BEAM
B	MGRADEUP	MONADIC	GRADE UP
B	MGRADEDN	MONADIC	GRADE DOWN
B	MCOMPL	MONADIC	COMPLEMENT
BAL,15	SYSTERR	UNUSED	
BAL,15	SYSTERR	UNUSED	
PAGE			

*
*
*
*
*
*
*
*
*
*
*

DYADIC OPERATOR EXECUTION DRIVER

ENTERED WITH OPTYPE IN 'OP' REGISTER, AND THE ARG
POINTERS IN LFARG AND RTARG. IF NO ERRORS OCCUR,
IT RETURNS TO 'DXRETURN' WITH THE RESULT POINTER
IN 'RESULT'. ERRORS LEAD TO ERDOMAIN, ERRANK, OR
ERLENGTH.

DXDRIVER	LI,A	DXRETURN	SET UP RETURN ADDRESS
	STW,A	RETURN	
	B	DXTABLE,OP	BRANCH TO APPROPRIATE OP ROUTINE
*			
DXTABLE	TABLE	DOPLOW	DYADIC OP JUMP TABLE
	B	DMATDIV	DYADIC MATRIX DIVIDE
	B	DDEAL	DYADIC DEAL
	B	DTBAR	DYADIC TYPE CONVERSION
	AI,OP	1	DYADIC COMPRESSION (1ST COORD)
	B	DCOMPRES	DYADIC COMPRESSION
	B	DINDEXOF	DYADIC INDEX OF
	B	DRESHAPE	DYADIC RESHAPE
	B	DCATEN	DYADIC CATENATE/LAMINATE
	AI,OP	1	DYADIC ROTATE (1ST COORD)
	B	DROTATE	DYADIC ROTATE
	B	DTRANS	DYADIC TRANSPOSE
	B	DADD	DYADIC ADD
	B	DSUB	DYADIC SUBTRACT
	B	DMUL	DYADIC MULTIPLY
	B	DDIV	DYADIC DIVIDE
	B	DPOWER	DYADIC POWER
	B	DLOG	DYADIC LOGARITHM
	B	DCIRC	DYADIC CIRCULAR
	B	DMAXIMUM	DYADIC MAXIMUM
	B	DMINIMUM	DYADIC MINIMUM
	B	DRESIDUE	DYADIC RESIDUE
	B	DCOMB	DYADIC COMBINATORIAL
	B	DLESS	DYADIC LESS
	B	DLESSEQ	DYADIC LESS OR EQUAL
	B	DGREAT	DYADIC GREATER
	B	DGREATEQ	DYADIC GREATER OR EQUAL
	B	DNEQUAL	DYADIC NOT EQUAL
	B	DEQUAL	DYADIC EQUAL
	B	DAND	DYADIC AND
	B	DOR	DYADIC OR
	B	DNAND	DYADIC NAND
	B	DNOR	DYADIC NOR
	BAL,15	SYSTERR	UNUSED
	BAL,15	SYSTERR	UNUSED
	B	DDECODE	DYADIC DECODE
	B	DENCODE	DYADIC ENCODE
	B	DTAKE	DYADIC TAKE
	B	DDROP	DYADIC DROP
	AI,OP	1	DYADIC EXPAND (1ST COORD)
	B	DEXPAND	DYADIC EXPAND
	B	DMEMBER	DYADIC MEMBERSHIP

*
*
*
*

MONADIC NO-OP ROUTINE (RESULT = ARG)

MNOP	EQU	%	
	LW,A	RTARG	GET RIGHT ARG PNTR
	MTW,1	1,A	BUMP ITS REF COUNT

	STW, A B PAGE	RESULT *RETURN	STORE AS RESULT PNTR RETURN
* * * M O N A D I C S C A L A R O P R O U T I N E S *			
MMINUS	EQU LI, T CB, T BNE BAL, XL LI, OP	% ISEQ *RTARG 16Z1 BILDISEQ DOPMUL+1	MONADIC MINUS IS ARG AN ISEQ ? NO, HANDLE NORMALLY YES, BUILD ISEQ RESULT SET OP CODE TO FALL INTO RANGE OF ISEQ TABLES (RETURNS IF ISEQOVFL). FIX OP TO FALL INTO ABS/NEG RANGE
16Z1	LI, OP	MOPABS+1	
MABS	EQU BAL, LX AI, OP BAL, L2 BAL, L1 LW, IX BAL, LX CI, OP BL B	% MTYPEIF -2 MSETUP GXSEGML MOPTBL2, OP CHANGOP MOPABS GXOVSTEX GXSTEXEC	MONADIC ABSOLUTE VALUE ALLOW NUMERIC ARG; RESULT = I/F INTG: MODIFY OP FLOT; SET UP ADDRESSES GEN INIT XSEG CODE GEN LOAD MINUS/ABS OF ARG IF IT'S INTG, GEN OVFL TEST CODE GEN STORE, ETC.
MOPTBL2	TABLE LAW, AI LCW, AI LAD, AF LCD, AF PAGE	MOPABS-2 AI AI AF AF	I/F MINUS/ABS TABLE I ABS I MINUS F ABS F MINUS
MEXP	EQU	%	MONADIC EXPONENTIAL
MLOG	EQU	%	MONADIC LOGARITHM
MPITIMES	EQU	%	MONADIC PI TIMES
MONOP1	BAL, L1 BAL, L2 BAL, L1 LW, R GEN, 1, 0 B	MTYPEF MSETUP GXSEGML MOPTBL3, OP R GXSTEXEC	SET UP TYPES; RESULT = FLOT SET UP ADDRESSES GEN LOAD/CONVERT GEN BAL TO EVAL ROUTINE GEN STORE, ETC.
MOPTBL3	TABLE BAL, LX BAL, LX FML, AF BAL, L1 BAL, LX BAL, LX BAL, LX BAL, LX BAL, LX	MOPEXP FEXP FLOG FLOTPI FCEILING FFLOOR IFACT FFACT IROLL	F EXP F LOG PI TIMES F CEILING F FLOOR I FACTORIAL F FACTORIAL I ROLL
MROLL	EQU BAL, L1 AI, OP B	% MTYPEI MOPFACT+1-MOPROLL MONOP1	MONADIC ROLL SET UP TYPES, RESULT = INTG MODIFY OP (TO SHARE OP TABLE) HENCEFORTH, TREAT LIKE 'EXP', ETC.
MFACT	EQU BAL, LX BDR, OP B PAGE	% MTYPEIF MONOP1 MONOP1	MONADIC FACTORIAL SET UP TYPES, RESULT = INTG/FLOT INTG: MODIFY OP, FINISH LIKE EXP FLOT: FINISH LIKE EXP
MSIGNUM	EQU LI, T CB, T BE BAL, L1	% LOGL *RTARG MNOP MTYPEI	MONADIC SIGNUM IF ARG IS LOGL, SIGNUM(ARG) = ARG. SET TYPES; RESULT = INTG

	BAL, L2	MSETUP	SET ADRS
	LW, T	RTTYPE	SET RSTYPE SO THAT ARG'S
	STW, T	RSTYPE	WONT BE CONVERTED.
	BAL, L1	GXSEGM	GEN LOAD
	GEN, 0, 5	SIGCODE	GEN SIGNUM CODE SEQUENCE
	AWM, XL	-2, XL	FILL IN BRANCH ADDRESSES
	AWM, XL	-4, XL	
	AWM, XL	-5, XL	
	LI, T	INTG	SET TO STORE BY SIGNUM'S
	STW, T	RSTYPE	RESULT TYPE (INTG).
	B	GXSTEXEC	GEN STORE, ETC.
*			
*			
SIGCODE	EQU	%	SIGNUM CODE SEQUENCE
	NBG	-1 (%+4)	-5
	BEZ	0 (%+4)	-4
	LI, AI	-1	-3
	B	0 (%+2)	-2
	LI, AI	+1	-1
	PAGE		
*			
*			
MRECIP	EQU	%	MONADIC RECIPROCAL
	BAL, L1	MTYPEF	SET TYPES, RESULT = FLOT
	BAL, L2	MSETUP	SET ADDRESSES
	BAL, LX	GXSEGINI	GEN XSEG INIT CODE
	LI, T	FLOT	IF ARG IS NOT FLOT,
	CW, T	RTTYPE	
	BE	15Z1	
	LI, A	1	LOAD/CONVERT IT,
	BAL, L2	GENLOADT	STORE IT IN TEMP.
15Z1	LD, R	RECIPINS	GEN: LD, AF =1.0
	AW, R+1	RTADR	FDL, AF ARG (OR TEMP)
	GEN, 2, 0	R	
	B	GXSTEXEC	GEN STORE, ETC.
*			
*			
RECIPINS	BOUND	8	
	LD, AF	FLOT1	RECIPROCAL CODE SEQ
	FDL, AF	0	
	PAGE		
*			
*			
MFLOOR	EQU	%	MONADIC FLOOR
MCEILING	EQU	%	MONADIC CEILING
	AW, OP	=X'80000000'	SET 1ST-TIME FLAG; 'INTGOVFL' WILL
*			RESTORE OP WITH THIS FLAG RESET.
	BAL, LX	MTYPEIF	SET TYPES, RESULT = INTG/FLOT
	B	MNOP	INTG (I/L): NO-OP (RESULT=ARG)
	AI, OP	0	FLOT: IS THIS 1ST TIME ?
	BGEZ	MSETRT	NO, DO IT IN FL. DOMAIN
	MTW, INTG-	FLOT RSTYPE	YES, SET UP ADDRESSES FOR INTEGER
	BAL, L2	MSETUP	RESULT, AND ATTEMPT OPERATION
	BAL, LX	GXSEGINI	
	LW, R	RTADR	IN INTG DOMAIN.
	AW, R	LOADINST+FLOT	LOAD IT (FLOT)
	LW, R+1	MOPTBL4, OP	GEN BAL ICELING/IFLOOR
	GEN, 2, 0	R	
	B	GXSTEXEC	GEN STORE, ETC.
MSETRT	STW, T	RTTYPE	SET RTTYPE TO FL.
	B	MONOP1	
*			
*			
MOPTBL4	TABLE	MOPCEIL	INTG FLOOR/CEILING SEQ TABLE
	BAL, L1	ICEILING	I CEILING
	BAL, LX	IFLOOR	I FLOOR
	PAGE		
*			
*			
MCOMPL	EQU	%	MONADIC COMPLEMENT
	LI, T	LOGL	IF ARG IS LOGL, DO IT WORD-WISE
	CB, T	*RTARG	
	BE	17Z1	
	BAL, L1	MTYPEL	NOPE - SET UP TYPE, RESUL = LOGL
	B	17Z2	SET ADR AND DO IT BIT-WISE.
17Z1	STW, T	RSTYPE	WORDWISE: SET ARG TYPE TO
	LI, T	WORDLOGL	'WORD LOGICAL' FOR ADDRESSING
	STW, T	RTTYPE	'EM 32 BITS AT A TIME.

17Z2	BAL,L2 BAL,L1 GEN,0,1 B PAGE	MSETUP GXSEGML EORINST GXSTEXEC	GEN BIT/WORD LOAD GEN: EOR,AI =-1 GEN BIT/WORD STORE, ETC.
* * * * *	D Y A D I C	S C A L A R	O P R O U T I N E S
DADD	EQU	%	DYADIC ADD
DSUB	EQU	%	DYADIC SUBTRACT
DMUL	EQU	%	DYADIC MULTIPLY
	LW,X	COPTRIG	IF WE'RE DOING A COMPOSITE OP,
	BLZ	NORMALOP	DON'T DO SPECIAL CASING.
	LI,T	ISEQ	IS LEFT ARG AN 'INDEX SEQUENCE' ?
	CB,T	*LFARG	
	BNE	8Z2	NO, CHECK RIGHT ARG
	CB,T	*RTARG	YES, IS RIGHT ARG ALSO ISEQ ?
	BNE	8Z1	NO
	CI,OP	DOPMUL	YES, BOTH ARGS ARE ISEQ'S. WE MAY
	BE	NORMALOP	ADD OR SUBTRACT THEM, BUT NOT MUL.
	BAL,XL	BILDISEQ	FORM RESULTANT ISEQ
	B	ISEQ1	
	B	NORMALOP	OVFL: DO IT NORMALLY
8Z1	LH,R	*RTARG	LEFT ARG IS ISEQ. IS RIGHT ARG
	CLM,R	LISCALAR	AN INTEGER OR LOGICAL SCALAR ?
	BCR,12	8Z3	
	BNE	NORMALOP	NO, DO NORMALLY
8Z3	BAL,XL	BILDISEQ	YES, FORM RESULT ISEQ
	XW,X	T	SWAP PTRS
	B	NORMALOP	OVFL: DO IT NORMALLY
8Z2	CB,T	*RTARG	LEFT ARG NOT ISEQ; IS RIGHT ARG ?
	BNE	NORMALOP	NO
	LH,R	*LFARG	YES, IS LEFT ARG AN INTEGER
	CLM,R	LISCALAR	OR LOGICAL SCALAR ?
	BCR,12	8Z4	
	BNE	NORMALOP	NO, DO NORMALLY
8Z4	BAL,XL	BILDISEQ	YES, FORM RESULT ISEQ
	NOP		DON'T SWAP ARG PTRS
* * *			OVFL: DO IT NORMALLY
NORMALOP	EQU	%	NORMAL ADD/SUB/MUL
	BAL,LX	DTYPEIF	ALLOW NUMERIC ARGS; RSTYPE= I/F
	AI,OP	-3	INTG: MODIFY OP
DYOP1	BAL,L2	DSETUP	EXAMINE ARG SHAPES; GET RS DATA BLOK
	BAL,L3	GXSEGD	GEN 1ST PART OF XSEG: LOADS/CONVERTS
	AW,R	DOPTBL2,OP	IN-LINE: BUILD OP
	GEN,1,0	R	GEN 'OP RTADR'
	CI,OP	DOPADD	IF IT'S INTEGER,
	BL	GXOVSTEX	GEN OVERFLOW TEST CODE.
	B	GXSTEXEC	
* * *			
DDIV	EQU	%	DYADIC DIVIDE
	BAL,L1	DTYPEF	ALLOW NUMERIC ARGS; RSTYPE= F
	B	DYOP1	CONTINUE AS FOR '+' ETC.
* * *			
DOPTBL2	TABLE	DOPADD-3	ADD/SUB/MUL/DIV TABLE
	ODD,AI		
	AW,AI	0	I ADD
	SW,AI	0	I SUBTRACT
	MW,AI	0	I MULTIPLY
	FAL,AF	0	F ADD
	FSL,AF	0	F SUBTRACT
	FML,AF	0	F MULTIPLY
	FDL,AF	0	F DIVIDE
* *			
	BOUND	8	
LISCALAR	DATA	LOGL**8+0, INTG**8+0	LOGICAL/INTEGER SCALAR HEADINGSU08-0031
	PAGE		
* * *			
DLOG	EQU	%	DYADIC LOGARITHM
	BAL,L1	DTYPEF	ALLOW NUMERIC ARGS; RSTYPE= F

	B	DYOP2	
*			
*			
DRESIDUE	EQU	%	DYADIC RESIDUE
DCOMB	EQU	%	DYADIC COMBINATORIAL
DYOP5	BAL, LX	DTYPEIF	ALLOW NUMERIC ARGS; RSTYPE= I/F
DYOP6	AI, OP	-2	INTG: MODIFY OP INDEX
DYOP2	BAL, L2	DSETUP	SET UP SIZES, ADRS
DYOP7	BAL, L3	GXSEGD	GEN LOAD LEFT, CONVERTS
	LW, T	RSTYPE	
	AW, R	LODBINST, T	GEN LOAD RIGHT
	LW, R+1	DOPTBL3, OP	
	GEN, 2, 0	R	BAL, .. EVALUATOR
	B	GXSTEXEC	GEN STORE, ETC.
*			
*			
DOPTBL3	TABLE	DOPPOWER-2	I/F - POWER/LOG/RES/COMB TABLE
	BAL, LX	IPOWER	I POWER
	ITEM	DOPPOWER	
	BAL, LX	FFPOWER	F POWER
	BAL, LX	FFLOG	F LOG
	ITEM	DOPRESID-2	
	BAL, LX	IRESIDU	I RESIDUE
	BAL, LX	IICOMB	I COMBINATORIAL
	BAL, LX	FFRESIDU	F RESIDUE
	BAL, L1	FFCOMB	F COMBINATORIAL
	PAGE		
*			
*			
DPOWER	EQU	%	DYADIC POWER
	LW, X	COPTRIG	IF WE'RE DOING A COMPOSITE OP,
	BLZ	DYOP5	DON'T DO SPECIAL CASING.
	LI, R	INTG**8+0	IS RIGHT ARG AN
	CH, R	*RTARG	INTEGER SCALAR ?
	BE	12Z3	YES, SEE IF IT'S 2
	LI, R	FLOT**8+0	NO, IS IT A
	CH, R	*RTARG	FLOATING POINT SCALAR ?
	BE	12Z1	YES, SEE IF IT'S 2.0 OR 0.5
NORMPWR	BAL, LX	DTYPEIF	NO, CHECK TYPES; RSTYPE = I/F
	B	DYOP6	I*I: HANDLED LIKE I-COMB/RESIDUE
	BAL, L2	DSETUP	I*F, F*I, OR F*F: SET UP STUFF
	LI, T	FLOT	
	CW, T	RTTYPE	CHECK RIGHT ARG TYPE
	BE	DYOP7	F*F OR I*F: TREAT LIKE F-COMB/RESID
	BAL, LX	GXSEGINI	F*I: GEN XSEG INIT CODE
	MTW, INTG-	FLOT RSTYPE	
	LI, A	1	GEN LOAD OF RIGHT ARG
	BAL, L1	GENLOAD	IN INTG DOMAIN
	LW, IX	LREGBI	TO REGISTER 'BI'.
	BAL, LX	CHANGREG	
	MTW, FLOT-	INTG RSTYPE	GEN LOAD OF LEFT ARG
	LI, A	0	IN FLOT DOMAIN
	BAL, L1	GENLOAD	TO REGISTERS 'AF/AF1'.
	GEN, 0, 1	FIPWRINS	GEN: BAL, LX FIPPOWER
	B	GXSTEXEC	GEN STORE, ETC.
*			
12Z1	LI, X	2	RIGHT ARG = FLOT SCALAR
	LW, R	*RTARG, X	GET IT'S 1ST WORD
	CLM, R	PWRLIMS	IS IT 0.5 OR 2.0 ?
	BCR, 12	12Z4	= 2.0, SQUARE LFARG (FLOT)
	BCS, 3	NORMPWR	NEITHER, DO NORMAL POWER
*			= 0.5, SQUARE ROOT LFARG (FLOT)
	EXCHANGE	ARGS	MAKE RTARG ACTIVE (LIKE MONADIC OPS)
	BAL, L1	MTYPEF	PRETEND IT'S A MONADIC 'SQRT' OP
12Z2	BAL, L2	MSETUP	CHECK TYPE, SET UP STUFF
	BAL, L1	GXSEGML	GEN LOAD/CONVERT
	LW, R	MOPTBL1, OP	GET SQUARE/SQRT INST
	GEN, 1, 0	R	
	CI, OP	DOPPOWER-2	IF IT'S IN INTG DOMAIN,
	BE	GXOVSTEX	GEN OVFL TEST CODE.
	B	GXSTEXEC	GEN STORE, ETC.
*			
12Z3	LI, X	2	RIGHT ARG IS AN INTG SCALAR
	CW, X	*RTARG, X	IS IT 2 ?
	BNE	NORMPWR	NO, DO NORMAL POWER
12Z4	EXCHANGE	ARGS	= 2 (OR 2.0), TREAT LIKE A
	BAL, LX	MTYPEIF	MONADIC 'SQUARE' OP.
	AI, OP	-1	I*2: DECR OP BY 2

	BDR, OP	12Z2	F*2: DECR OP BY 1
* *			
MOPTBL1	TABLE MW, AI FML, AF BAL, LX	DOPPOWER-2 AI AF FSQRT	SQUARE/SQRT INST TABLE I SQUARE F SQUARE F SQRT
* *			
PWRLIMS	BOUND DATA	8 FS'0.5',FS'2.0'	SPECIAL EXPONENT VALUES
* FIPWRINS	BAL, LX PAGE	FIPOWER	F TO I POWER INST
* *			
DCIRC	EQU BAL, L1 BAL, L2 LW, X BGEZ BAL, L3 GEN, 0, 3 LW, R AW, R GEN, 1, 0 B	% DTYPEF DSETUP COPTRIG 20Z1 GXSEGD CIRCSEQ1 RTADR LOADINST+FLOT R 20Z3	DYADIC CIRCULAR CHECK TYPES; RESULT = FLOT TYPE SET UP SIZES/ADRS IF WE'RE DOING A COMPOSITE OP, WE MUST GO THROUGH GXSEGD AND GEN CODE TO CONVERT LFARG FROM FLOT IN AF TO INTG IN T. GEN LOAD (FLOT) OF RT ARG
20Z1	BAL, LX MTW, INTG- LI, A BAL, L1 LW, IX BAL, LX LI, S CW, S BNE GEN, 0, 1 STW, XL GEN, 0, 1	GXSEGINI FLOT RSTY 0 GENLOAD LREGTX CHANGREG 1 LFSIZE 20Z2 STORETX LOOPLOC LOADTX	GEN XSEG INIT CODE GEN FETCH OF LEFT ARG IN INTG MODE. MOVE IT TO 'TX' (IT'S SAFE THERE) IF LEFT ARG IS ONE ELEMENT, SAVE LFARG OUTSIDE LOOP, AND RESTORE IT INSIDE LOOP.
20Z2	EQU MTW, FLOT- LI, A BAL, L1	% INTG RSTY 1 GENLOAD	GEN LOAD OF RIGHT ARG IN FLOT MODE (REG 'AF/AF1')
20Z3	EQU GEN, 0, 3 B	% CIRCSEQ GXSTEXEC	GEN RANGE CHECK, INDEXED BAL GEN STORE, ETC.
* *			
TX	EQU	3	INDEX REG FOR 'CIRCULAR' BRANCH
* CIRCSEQ	CLM, TX BCS, 9 BAL, LX	CIRCLIMS ERDOMAIN CIRCULAR, TX	LEFT ARG RANGE CHECK INDEXED BAL INTO JUMP TABLE
* CIRCSEQ1	BAL, LX B LW, TX	F2I ERDOMAIN AI	CONVERT LFARG VALUE TO INTG .. IN TX.
* LOADTX STORETX	LW, TX STW, TX	LFTEMP LFTEMP	
* CIRCLIMS	BOUND DATA PAGE	8 -7, +7	LEFT ARG RANGE
* *			
DEQUAL DNEQUAL	EQU EQU	% %	DYADIC EQUAL DYADIC NOT EQUAL
* *			
	SPECIAL CASE FOR CHARACTER ARGS		
	LW, X BLZ LI, T CB, T BNE CB, T BNE	COPTRIG DYOP3 CHAR *LFARG 19Z1 *RTARG 19Z2	IF WE'RE DOING A COMPOSITE OP, DON'T DO SPECIAL CASING. IF BOTH ARGS NUMERIC, TREAT LIKE OTHER RELATIONALS. IF BOTH ARGS CHARACTER,

U08-0033

U08-0034

U08-0035

CHAREQ	STW, T	LFTYPE	SET ALL TYPES = CHAR	
	STW, T	RTTYPE		
	STW, T	RSTYPE		
	B	DYOP4		
19Z1	CB, T	*RTARG		U08-0037
	BNE	DYOP8		
19Z2	LI, T	LOGL	TYPES MIXED,	
	STW, T	RSTYPE	RESULT = ALL 0'S OR ALL 1'S.	
	LI, T	WORDLOGL	(STORED 32 BITS AT A TIME,	
	STW, T	LFTYPE	OF COURSE).	
	STW, T	RTTYPE		
	BAL, L2	DSETUP		
	BAL, LX	GXSEGINI		
	LW, R	DOPTBL7, OP		
	GEN, 1, 0	R	GEN LI, AI 0/1	
	B	GXSTEXEC		
*				
*				
DLESS	EQU	%	DYADIC LESS	
DGREAT	EQU	%	DYADIC GREATER	
DLESSEQ	EQU	%	DYADIC LESS OR EQUAL	
DGREATEQ	EQU	%	DYADIC GREATER OR EQUAL	
*				U08-0039
*				U08-0040
*				U08-0041
	LW, X	COPTRIG	IF WE'RE DOING A COMPOSITE OP,	U08-0042
	BLZ	DYOP3	DON'T DO SPECIAL CASING.	U08-0043
DYOP8	LI, T	LOGL		U08-0044
	CB, T	*LFARG	TEST FOR BOTH-ARGS-LOGICAL	U08-0045
	BNE	DYOP3		U08-0046
	CB, T	*RTARG		U08-0047
	BNE	DYOP3		U08-0048
	STW, T	RSTYPE	BOTH ARGS LOGL: SET UP TO DO	U08-0049
	LI, T	WORDLOGL	OP 32 BITS AT A TIME WITH	U08-0050
	STW, T	LFTYPE	LOGICAL OPERATORS.	U08-0051
	STW, T	RTTYPE		U08-0052
	ERROR, 0, 1-((DOPLESS&7)=6)		'NEXT INST WONT WORK'	U08-0053
	CI, OP	2	SINGLES OUT > AND >= OPS	U08-0054
	BANZ	19Z4	FOR > AND >= SWAP ARGS AND	U08-0055
	EXCHANGE	ARGS	TREAT AS < AND <=.	U08-0056
19Z4	BAL, L2	DSETUP	SET UP ARG/RESULT PARAMS	U08-0057
	BAL, L3	GXSEGDL	GEN LOAD OF LF ARG	U08-0058
	CI, OP	DOPNEQ	FOR ALL EXCEPT 'NOT ='	U08-0059
	BE	19Z5	GEN 'NOT' OPERATION.	U08-0060
	GEN, 0, 1	EORINST		U08-0061
19Z5	LW, R	DOPTBL8, OP	GEN AND/OR/EOR OF RT ARG	U08-0062
	AW, R	RTADR		U08-0063
	GEN, 1, 0	R		U08-0064
	B	GXSTEXEC	GEN STORE/LOOP CODE; EXECUTE XSEG	U08-0065
*				U08-0066
*				U08-0067
*				U08-0068
DYOP3	BAL, LX	DTYPEIF	ALLOW NUMERIC ARGS; RSTYPE= I/F	
	B	DYOP4	INTG	
	BAL, L3	DSETUPL	FLOT: SET UP LOGL DB	
	BAL, L3	GXSEGDL	GEN LOAD LEFT	
	AW, R	LODBINST+FLOT	GEN LOAD OF RIGHT ARG	
	GEN, 1, 1	R, BALCOMP	GEN FUZZ APPLIER	
	LI, R	BF	SET NEW RT ADR = REG 'BF'	
	B	GENCOMP		
DYOP4	BAL, L3	DSETUPL	INTG: SET UP LOGL DB	
	BAL, L3	GXSEGDL	GEN LOAD LEFT	
GENCOMP	LW, T	RSTYPE		
	AW, R	COMPINST, T	GEN COMPARE INST	
	LW, R+1	DOPTBL4, OP	GET BRANCH INST	
	GEN, 2, 4	R, STCCSEQ	GEN: C., AF RTARG	
	AWM, XL	-3, XL	B., %+3 (TRUE)	
*			LI, AI 0 (FALSE)	
	AWM, XL	-5, XL	B, %+2	
*			LI, AI -1	
*			BAL, LX STLOGLRS	
	LI, T	LOGL	RESTORE CORRECT RESULT TYPE	
	XW, T	RSTYPE		
	CI, T	CHAR		
	BNE	19Z3		
	LW, R	COPTRIG	IF IT'S A CHARACTER COMPARISON,	
	BLZ	19Z3	BEING DONE AS A SCALAR OP,	
	LI, R	3		

	AW, R	RSSIZE	SET RSSIZE = NEXT HIGHER	
	AND, R	=-4	MULTIPLE OF FOUR CHARS.	
19Z3	STW, R	RSSIZE		
*	BDR, XL	GXSTEXEC	DISCARD BAL...STLOGLRS, DO NORMAL	
*			STORE GEN (BECAUSE OF COMPOSITE OPS).	
*				
DSETUPL	LI, T	LOGL	SET RSTYPE=LOGL	
	XW, T	RSTYPE	SAVE OLD RSTYPE (INTG/FLOT)	
	STW, T	TYPETEMP		
	BAL, L2	DSETUP	NOW DO NORMAL DYADIC SETUP	
	LW, T	TYPETEMP	RESTORE OLD RSTYPE	
	STW, T	RSTYPE		
	B	*L3	RETURN	
*				
TYPETEMP	TEMP		TEMP FOR SAVING RSTYPE	
*				
*				
DOPTBL4	TABLE	DOPLESS	RELATIONAL BRANCH TABLE	
	NBL	-2	<	
	NBLE	-2	<=	
	NBG	-2	>	
	NBGE	-2	>=	
	NBNE	-2	NOT =	
	NBE	-2	=	
*				
*				
DOPTBL7	TABLE	DOPNEQ	INST TABLE FOR 1='A' AND THE LIKE	
	LI, AI	-1	NOT EQUAL	
	LI, AI	0	EQUAL	
*				U08-0070
*				U08-0071
DOPTBL8	TABLE	DOPLESS	INST TBL FOR LOGL RELATIONALS	U08-0072
	AND, AI	0	A < B LW A; EOR =-1; AND B	U08-0073
	OR, AI	0	A <= B LW A; EOR =-1; OR B	U08-0074
	AND, AI	0	A > B LW B; EOR =-1; AND A	U08-0075
	OR, AI	0	A >= B LW B; EOR =-1; OR A	U08-0076
	EOR, AI	0	A NOT= B LW A; EOR B	U08-0077
	EOR, AI	0	A = B LW A; EOR =-1; EOR B	U08-0078
*				
*				
STCCSEQ	EQU	%	CODE FOR END OF RELAT OP	
	LI, AI	0	-4	
	NB	-1 (%+2)	-3	
	LI, AI	-1	-2	
	BAL, LX	STLOGLRS	-1	
*				
*				
BALCOMP	BAL, LX	FFCOMPAR	FUZZIFIER	
*				
*				
COMPINST	TABLE	CHAR	COMPARE INST - BY MODE:	
	CB, AI	0	CHAR	
	CW, AI	0	INTG	
	CD, AF	0	FLOT	
	PAGE			
*				
*				
DAND	EQU	%	DYADIC AND	
DOR	EQU	%	DYADIC OR	
DNAND	EQU	%	DYADIC NAND	
DNOR	EQU	%	DYADIC NOR	
	LW, X	COPTRIG	IF WE'RE DOING A COMPOSITE OP,	
	BLZ	11Z1	DON'T DO SPECIAL CASING.	
	LI, T	LOGL	ARE BOTH TYPES LOGL ?	
	CB, T	*LFARG		
	BNE	11Z1	NO	
	CB, T	*RTARG		
	BE	11Z2	YES	
11Z1	BAL, L1	DTYPEL	NO: SET UP TYPES	
	B	11Z3	JOIN OTHER CASE	
11Z2	STW, T	RSTYPE	BOTH ARGS LOGL: SET THEIR TYPES TO	
	LI, T	WORDLOGL		
	STW, T	RTTYPE	'WORD LOGICAL' FOR ADDRESSING	
	STW, T	LFTYPE	THEM 32 BITS AT A TIME.	
11Z3	BAL, L2	DSETUP		
	BAL, L3	GXSEGD	GEN LOADS/CONVERTS	
	AW, R	DOPTBL6, OP	GEN AND/OR INST (DOES 1 OR 32 OPS)	

```

GEN,1,0 R
CI,OP DOPNAND IS 'NEGATE' NEEDED ?
BL GXSTEXEC NO, GEN STORE, ETC.
GEN,0,1 EORINST YES, GEN NEGATE-ALL-BITS INST
B GXSTEXEC GEN STORE, ETC.
*
*
DOPTBL6 TABLE DOPAND AND/OR/NAND/NOR INST TABLE
AND,AI 0 AND
OR,AI 0 OR
AND,AI 0 NAND (FOLLOWED BY NEGATE)
OR,AI 0 NOR (FOLLOWED BY NEGATE)
*
EORINST EOR,AI =-1 LOGICAL NEGATE INST (ALL 32 BITS)
PAGE
*
*
DMINIMUM EQU % DYADIC MINIMUM
DMAXIMUM EQU % DYADIC MAXIMUM
BAL,LX DTYPEIF CHECK TYPES
NOP (DONT CARE IF IT'S I OR F)
BAL,L2 DSETUP SET UP SIZES/ADDRESSES
BAL,L3 GXSEGD L GEN LOADS/CONVERTS
LW,T RSTYPE
AW,R COMPINST,T GEN: COMPARE RTADR
LW,R+1 DOPTBL5,OP BLE/BGE %+2
LW,R+2 LOADINST,T LOAD RTADR
AW,R+2 RTADR
GEN,3,0 R
AWM,XL -2,XL
B GXSTEXEC GEN STORE, ETC.
*
*
DOPTBL5 TABLE DOPMAX MAX/MIN TABLE
BGE 0 (%+2) MAX
BLE 0 (%+2) MIN
PAGE
*
*
* TYPE CHECKING ROUTINES
*
* MONADIC I/F TYPE
*
* ALLOWS A NUMERIC (RIGHT) ARG. SETS RTTYPE TO ARG
* TYPE AND RSTYPE TO RESULT TYPE (INTG FOR LOGL/INTG
* ARG, FLOT FOR FLOT ARG). LINK IS LX; RETURNS TO
* BAL+1 FOR I/L ARG, BAL+2 FOR F ARG.
*
MTYPEIF EQU % NUMERIC ARG, I/F RESULT
STW,LX TYPELINK SAVE LINK IN CASE OF INTGOVFL
LB,X *RTARG GET ARG TYPE
STW,X RTTYPE SAVE IT
B %+1-LOGL,X TEST IT:
B 13Z1 L RSTYPE = I
B ERDOMAIN C NOT ALLOWED
B 13Z2 I RSTYPE = I
B 13Z3 F RSTYPE = F
B 13Z1 ISEQ TREAT AS INTG
B ERDOMAIN LIST NOT ALLOWED
*
13Z1 LI,X INTG LOGL: SET TYPE = INTG
13Z2 STW,X RSTYPE INTG/LOGL: RSTYPE = INTG
B 0,LX INTG RETURN
*
13Z3 STW,X RSTYPE FLOT: RSTYPE = FLOT
B 1,LX FLOT RETURN
*
*
* MONADIC F TYPE
*
* ALLOWS NUMERIC ARG; SETS RTTYPE = ARG TYPE AND
* RSTYPE = FLOT. LINK IS L1.
*
MTYPEF EQU % NUMERIC ARG, F RESULT
BAL,LX MTYPEIF ALLOW NUMERIC ARG; RSTYPE =
MTW,FLOT-INTG RSTYPE INTG: CHANGE TO FLOT
B *L1 FLOT: RETURN

```

```

*
*
* MONADIC I TYPE
*
*           ALLOWS NUMERIC ARG; SETS RTTYPE = ARG TYPE AND
*           RSTYPE = INTG.  LINK IS L1.
*
MTYPEI    EQU      %           NUMERIC ARG, I RESULT
          BAL,LX   MTYPEIF    ALLOW NUMERIC ARG; RSTYPE =
          NOP
          LI,T     INTG        INTG
          STW,T    RSTYPE
          B        *L1         RETURN
*
*
* MONADIC L TYPE
*
*           ALLOWS NUMERIC ARG; SETS RTTYPE = ARG TYPE AND
*           RSTYPE = LOGL.  LINK IS L1.
*
MTYPEL    EQU      %           NUMERIC ARG, L RESULT
          BAL,LX   MTYPEIF    ALLOW NUMERIC ARG; RSTYPE =
          NOP
          LI,T     LOGL        LOGL
          STW,T    RSTYPE
          B        *L1         RETURN
          PAGE
*
*
* DYADIC I/F TYPE
*
*           ALLOWS NUMERIC ARGS.  SETS LFTYPE/RTTYPE TO ARG TYPES
*           AND RSTYPE TO RESULT TYPE (INTG FOR LOGL/INTG ARGS,
*           FLOT IF EITHER ARG IS FLOT).  LINK IS LX; RETURNS TO
*           BAL+1 FOR INTG RESULT, BAL+2 FOR FLOT RESULT.
*
DTYPEIF   EQU      %           NUMERIC ARGS, I/F RESULT
          LW,X     COPTRIG    IF WE'RE DOING A COMPOSITE OP,
          BLZ      *CTYPELNK  DERAIL TO SPECIAL SUBROUTINE.
DTYPEIF1  LB,T     *LFARG     GET LEFT TYPE
DTYPEIF2  STW,T    LFTYPE     SAVE IT
          LB,X     *RTARG     GET RIGHT TYPE
          STW,X    RTTYPE     SAVE IT
          B        %+1-LOGL,T TEST LEFT TYPE:
          B        5Z1-LOGL,X L     TEST RIGHT TYPE
          B        ERDOMAIN    C     NOT ALLOWED
          B        5Z2-LOGL,X I     TEST RIGHT TYPE
          B        5Z4-LOGL,X F     TEST RIGHT TYPE
          B        5Z1-LOGL,X ISEQ  TREAT AS INTG
          B        ERDOMAIN    LIST  NOT ALLOWED
*
5Z1        B        5Z6        L,L   RSTYPE=I
          B        ERDOMAIN    L,C   NOT ALLOWED
          B        5Z7        L,I   RSTYPE=RTTYPE (I)
          B        5Z3        L,F   RSTYPE=RTTYPE (F)
          B        5Z6        L,ISQ TREAT AS INTG
          B        ERDOMAIN    L,LST NOT ALLOWED
*
5Z2        B        5Z8        I,L   RSTYPE=LFTYPE (I)
          B        ERDOMAIN    I,C   NOT ALLOWED
          B        5Z7        I,I   RSTYPE=RTTYPE (I)
          B        5Z3        I,F   RSTYPE=RTTYPE (F)
          B        5Z8        I,ISQ TREAT AS INTG
          B        ERDOMAIN    I,LST NOT ALLOWED
*
5Z3        STW,X    RSTYPE     =F
          B        1,LX       RETURN
*
5Z4        B        5Z5        F,L   RSTYPE=LFTYPE (F)
          B        ERDOMAIN    F,C   NOT ALLOWED
          B        5Z5        F,I   RSTYPE=LFTYPE (F)
          B        5Z5        F,F   RSTYPE=LFTYPE (F)
          B        5Z5        F,ISQ TREAT AS INTG
          B        ERDOMAIN    F,LST NOT ALLOWED
*
5Z5        STW,T    RSTYPE     =F
          B        1,LX       RETURN
5Z6        LI,X     INTG        L,L   RSTYPE=I

```

```

5Z7      STW,X      RSTYPE      =I
          STW,LX     TYPELINK    SAVE LINK IN CASE OF INTGOVFL
          B          0,LX        RETURN
5Z8      STW,T      RSTYPE      =I
          STW,LX     TYPELINK    SAVE LINK IN CASE OF INTGOVFL
          B          0,LX        RETURN
*
TYPELINK TEMP                                REMEMBER LINK FOR INTGOVFL
*
*
*   DYADIC F TYPE
*
*           ALLOWS NUMERIC ARGS; SETS LFTYPE/RTTYPE TO ARG TYPES
*           AND RSTYPE TO FLOT.  LINK IS L1.
*
DTYPEF   EQU      %           NUMERIC ARGS, F RESULT
          BAL,LX   DTYPEIF     ALLOW NUMERIC ARGS; RSTYPE=
          MTW,FLOT-INTG RSTYPE  INTG: CHANGE TO FLOT
          B        *L1         FLOT: RETURN
*
*
*   DYADIC L TYPE
*
*           ALLOWS NUMERIC ARGS; SETS LFTYPE/RTTYPE TO ARG TYPES
*           AND RSTYPE TO LOGL.  LINK IS L1.
*
DTYPEL   EQU      %           NUMERIC ARGS, L RESULT
          BAL,LX   DTYPEIF     ALLOW NUMERIC ARGS; RSTYPE=
          NOP
          LI,T     LOGL        LOGL
          STW,T    RSTYPE
          B        *L1         RETURN
          PAGE
*
*
*   I S E Q   M A N I P U L A T I O N   R O U T I N E S
*
*   BUILD ISEQ RESULT
*
*           BUILDS AN ISEQ DATA BLOCK FOR THE RESULT, UNLESS
*           ANY OPERATION OVERFLOWS.  THE OP TO BE DONE IS
*           SPECIFIED IN 'OP' REG, AND THE INSTRUCTION AT BAL+1
*           WILL BE EXECUTED.  IF THERE'S NO OVERFLOW, THIS
*           ROUTINE BUILDS THE RESULT AND EXITS FROM THE OP
*           DRIVER (THROUGH 'RETURN'); OTHERWISE, IT RETURNS TO
*           ITS CALLER AT BAL+2, HAVING DEREFFED THE ISEQ DB
*           IT ESTABLISHED FOR THE RESULT.  LINK IS XL.
*
BILDISEQ LI,S      3           ALLOCATE DB FOR RESULT ISEQ; IT'S
          BAL,LX7   ALOCHNW     SIZE IS HEADER + 3 WORDS.
          STW,A     RESULT      COPY RESULT DB PNTR
          LI,R      ISEQ**8+1   SET TYPE = ISEQ, RANK = 1.
          STH,R     *RESULT
          LW,T      LFARG       SET T= LEFT PTR
          LW,X      RTARG       SET X= RIGHT PTR
          EXU       0,XL        SET T=SCALAR ARG, X=ISEQ ARG
*   ONE ARG ISEQ, OTHER IS SCALAR (IF OP DYADIC)                                U08-0080
          LW,S      2,X         GET ISEQ LENGTH
          STW,S     2,A         SET RESULT ISEQ LENGTH
          LW,S      2,T         GET VALUE OF INTG/LOGL SCALAR
          LI,R      LOGL
          CB,R      *T
          BNE       4Z1         IF IT'S LOGICAL,
          SCS,S     1           CONVERT TO INTEGER.
          AND,S     =1
          4Z1      STW,S     DUMYISEQ+3   MAKE SCALAR VALUE LOOK LIKE ISEQ
          EXU       ISEQTBL1,OP   WITH BASE = VALUE,
          STW,S     DUMYISEQ+4   AND STEP = 0 (+-) OR VALUE (*).
          LI,T      DUMYISEQ      T= NEW ISEQ PTR
          EXU       0,XL        RESET T=LEFT, X=RIGHT
          4Z2      LW,S      3,T      GET LEFT BASE/VAL
          ISEQ2     EXU       ISEQTBL2,OP  ADD/SUB/MUL RIGHT BASE/VAL
          BNOV      4Z3
          B         ISEQOVFL
          *   BOTH ARGS ISEQ (DYADIC OPS ONLY)                                U08-0097
          ISEQ1     LW,S      2,X      BOTH ARGS ISEQ,
          STW,S     2,A      COPY SIZE TO RESULT ISEQ

```

	CW, S	2, T	AND MAKE SURE SIZES AGREE.	
	BE	4Z2		U08-0099
	B	ERLENGTH		
4Z3	STW, S	3, A	SET RESULT ISEQ BASE	U08-0101
	AI, X	1	POINT TO STEP	
	LW, S	4, T	GET LEFT STEP/VAL	
	EXU	ISEQTBL2, OP	ADD/SUB/MUL RIGHT STEP/VAL	
	BOV	ISEQOVFL		
	STW, S	4, A	SET RESULT ISEQ STEP	
	MW, S	2, A	MAKE SURE RESULT ISEQ	
	BOV	ISEQOVFL	REPRESENTS COMPUTABLE INTEGERS.	
	AW, S	3, A		
	BNOV	*RETURN	RETURN	
*				
ISEQOVFL	LI, A	0	OVFL: DISCARD ISEQ PNTR,	
	XW, A	RESULT		
	BAL, LX7	DREF	DE-REF ISEQ DATA BLOCK,	
	B	1, XL	AND RETURN.	
*				
*				
ISEQTBL1	TABLE	DOPADD	EXU TABLE TO SET 'STEP' OF SCALAR	
	LI, S	0	DYADIC ADD	
	LI, S	0	DYADIC SUBTRACT	
	NOP		DYADIC MULTIPLY	U08-0103
	B	ISEQ2	MONADIC NEGATE	
*				
*				
ISEQTBL2	TABLE	DOPADD	EXU TABLE TO COMBINE BASES/STEPS	
	AW, S	3, X	DYADIC ADD	
	SW, S	3, X	DYADIC SUBTRACT	
	MW, S	3, X	DYADIC MULTIPLY	
	LCW, S	3, X	MONADIC NEGATE	
*				U08-0105
*				U08-0106
DUMYISEQ	EQU	LFTEMP-3	DUMMY ISEQ DATA BLOCK: ONLY WORDS	U08-0107
*			3 (BASE) AND 4 (STEP) NEEDED.	U08-0108
	PAGE			
*				
*				
*	INTEGER OVERFLOW			
*				
*		WE ARRIVE HERE WHENEVER AN INTEGER OPERATION OVERFLOWS.		
*		THE OPERATION IS RESTARTED IN THE FLOATING DOMAIN BY		
*		CHANGING 'RSTYPE' FROM INTG TO FLOT AND GOING BACK TO		
*		THE OP ROUTINE AT THE POINT WHERE IT CALLED THE TYPE		
*		SETUP ROUTINE.		
*				
INTGOVFL	EQU	%		
	LI, A	0	DISCARD RESULT PNTR	
	STW, A	XSEGBRK	DON'T ALLOW BREAKS NOW	
	XW, A	RESULT	DE-REF THE INTG RESULT DB	
	BAL, LX7	DREF		
	LW, OP	OPER	RESTORE THE OP CODE	
	LI, T	INTG	PUT ARG TYPES BACK THE WAY THEY	
	LB, T	*LFARG		
	STW, T	LFTYPE	WERE AT 'DTYPEIF' TIME.	
	LB, T	*RTARG		
	STW, T	RTTYPE		
	LI, T	FLOT	SET RESULT TYPE TO FLOT	
	STW, T	RSTYPE		
	LW, LX	TYPELINK	RESTORE OP ROUTINE'S LINK INTO	
	B	1, LX	TYPE SUBR; TAKE 'FLOT' RETURN.	
*				
*				
*	FLOATING DIVIDE BY ZERO TRAP			
*				
*		RESULTS IN DOMAIN ERROR IN ALL CASES EXCEPT		
*		WHEN DOING A DYADIC DIVIDE OF 0 BY 0 (=1).		
*				
DIVZERO	EQU	%		
	LI, A	X'FF'	MAKE SURE OP IS 'DYADIC DIVIDE'	
	AND, A	OPER		
	CI, A	DOPDIV		
	BNE	0, LX	IF NOT, DOMAIN ERROR EXIT	
	CD, AF	FLOT0	YES: MAKE SURE NUMERATOR	
	BNEZ	0, LX	WAS ZERO.	
	LD, AF	FLOT1	OK: SUBSTITUTE ANSWER OF 1.0	
	B	1, LX	RETURN TO OK EXIT	

PAGE

*
*
* O P E R A T O R S E T U P R O U T I N E S
*
*

* MONADIC OPERATOR SETUP ROUTINE
*

* SETS UP FOR EXECUTION OF A MONADIC SCALAR OPERATOR.
* IT IS CALLED WITH THE ARG AND RESULT TYPES SET UP
* IN 'RTTYPE/RSTYPE'. LINK IS L2.
* (1) ALLOCATES RESULT DATA BLOCK
* (2) ESTABLISHES RESULT DIMENSIONS
* (3) SETS UP THE SIZES IN 'RT/RSSIZE' AND
* THE ADDRESSES IN 'RT/RSADR'.
*

MSETUP EQU %
 LI,X 1
 LB,R *RTARG,X GET ARG RANK
 STW,R RTRANK SAVE IT FOR 'GETSIZE'
 LI,A 1
 BAL,LX GETSIZE GET ARG SIZE
 NOP (NO MATTER IF IT'S 1)
 STW,S RSSIZE = RESULT SIZE
 BAL,L1 ALOCRS ALLOCATE RESULT DATA BLOCK
 LW,X RSRANK
 BEZ 14Z2 IF RANK>0,
 MTW,1 RTARG
 MTW,1 RESULT
14Z1 LW,R *RTARG,X COPY ARG DIMENSIONS
 STW,R *RESULT,X TO RESULT DATA BLOCK.
 BDR,X 14Z1
 MTW,-1 RTARG
 MTW,-1 RESULT
14Z2 LI,X -2 SET UP RTADR AND RSADR
 B SETADRS1

*
*
* DYADIC OPERATOR SETUP ROUTINE
*
*

* SETS UP FOR EXECUTION OF A DYADIC SCALAR OPERATOR.
* IT IS CALLED WITH THE ARG AND RESULT TYPES SET UP
* IN 'LF/RT/RSTYPE'. LINK IS L2.
* (1) PERFORMS CONFORMABILITY CHECKS
* (2) ALLOCATES RESULT DATA BLOCK
* (3) ESTABLISHES RESULT DIMENSIONS
* (4) SETS UP THE SIZES IN 'LF/RT/RSSIZE' AND
* THE ADDRESSES IN 'LF/RT/RSADR'.
*

DSETUP EQU %
 LW,X COPTRIG IF WE'RE DOING A COMPOSITE OP,
 BLZ *CSETLNK DERAIL TO SPECIAL SUBROUTINE.
 LI,X 1
 LI,S 1 1 FOR VECTOR/SCALAR TESTS
 LB,R *LFARG,X GET LEFT RANK
 STW,R LFRANK REMEMBER IT
 BEZ 1Z7 IS LEFT ARG A SCALAR ?
 LB,R *RTARG,X NO, GET RIGHT RANK
 STW,R RTRANK REMEMBER IT
 BEZ 1Z6 IS RIGHT ARG A SCALAR ?
 LI,A 1 NO, GET ITS SIZE
 BAL,LX GETSIZE
 B 1Z1 IS IT A ONE-ELMT ARRAY ?
 BAL,LX GETLSIZE YES, GET LEFT ARG SIZE
 B 1Z9 IS IT 1? IF NOT, USE LEFT RANK
 LW,R RTRANK NO, USE HIGHEST RANK
 CW,R LFRANK WE'VE ALREADY GOT THE LEFT RANK
 BLE 1Z9 SET UP; IF IT'S THE HIGHEST,
 B 1Z2 GREAT; ELSE, SWITCH TO RIGHT.
1Z1 BAL,LX GETLSIZE RT ARG ISN'T 1-ELMT; GET LF ARG SIZE
 B 1Z3 IS LF ARG 1-ELMT ?
 LW,R RTRANK YES, USE RIGHT RANK
1Z2 STW,R RSRANK (SET UP RT ARG RANK/SIZE)
 LW,S RTSIZE
 LI,A 1
 B 1Z9

1Z3	LW, R	LFRANK	BOTH ARGS MULTI-ELMT, MAKE SURE
	CW, R	RTRANK	THEY ARE SHAPED ALIKE.
	BNE	ERRANK	
	STW, S	RSSIZE	SET RESULT SIZE
	BAL, L1	ALOCRS	GET RESULT DATA BLOCK; SET TYPE/RANK
	LW, X	RSRANK	IF RANK>0,
	BEZ	SETADRS	
	MTW, 1	LFARG	SET ARG POINTERS TO WORD BEFORE
	MTW, 1	RTARG	FIRST DIMEN.
	MTW, 1	RESULT	SET PNTR TO WORD BEFORE DIMENS
1Z4	LW, R	*LFARG, X	COPY ARG DIMENS TO RESULT DIMENS
	STW, R	*RESULT, X	
	CW, R	*RTARG, X	MAKE SURE ARG DIMENS AGREE
	BE	1Z5	
	MTW, -1	LFARG	RESTORE POINTERS TO THEIR
	MTW, -1	RTARG	ORIGINAL POSITION
	MTW, -1	RESULT	
	B	ERLENGTH	
1Z5	BDR, X	1Z4	
	MTW, -1	LFARG	RESTORE POINTERS TO THEIR
	MTW, -1	RTARG	ORIGINAL POSITION
	MTW, -1	RESULT	
	B	SETADRS	GO SET ADDRESSES
1Z6	STW, S	RTSIZE	RIGHT ARG SCALAR, SET SIZE=1
	BAL, LX	GETLSIZE	GET LEFT ARG SIZE
	B	1Z9	(MULTI-ELEM) IN EITHER CASE, USE 08-00003
	B	1Z9	(ONE-ELEM) LEFT RANK/SIZE/DIMENS.08-00004
1Z7	STW, S	LFSIZE	LEFT ARG SCALAR, SET SIZE=1
	LB, R	*RTARG, X	GET RIGHT RANK
	STW, R	RTRANK	REMEMBER IT
	BEZ	1Z8	IS RT ARG SCALAR, TOO?
	LI, A	1	NO, GET RT ARG SIZE
	BAL, LX	GETSIZE	
	B	1Z9	(MULTI-ELEM) IN EITHER CASE, USE 08-00006
	B	1Z9	(ONE-ELEM) RT ARG RANK/SIZE/DIMS.08-00007
1Z8	LI, R	0	SCALAR RESULT:
	STW, R	RSRANK	RANK=0
	STW, S	RTSIZE	
	STW, S	RSSIZE	SIZE=1
	BAL, L1	ALOCRS	GET RESULT DATA BLOCK; SET TYPE/RANK
	B	SETADRS	GO SET ADDRESSES (NO DIMENS TO MOVE)
1Z9	STW, S	RSSIZE	USE LF/RT RANK
	LW, T	A	REMEMBER WHICH ARG TO USE
	BAL, L1	ALOCRS	GET RESULT DATA BLOCK; SET TYPE/RANK
	LW, T	LFARG, T	
	LW, X	RSRANK	IF RANK>0,
	BEZ	SETADRS	
	MTW, 1	RESULT	SET PNTR TO WORD BEFORE DIMENS
	AI, T	1	
1Z10	LW, R	*T, X	COPY LF/RT DIMENS
	STW, R	*RESULT, X	TO RESULT DIMENS.
	BDR, X	1Z10	
	MTW, -1	RESULT	RESTORE RESULT PNTR TO NORMALCY

*

*

* SET ARG ADDRESS CELL(S)

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

SETADRS	EQU	%	SET UP ADDRESSES
	LI, X	-3	
SETADRS1	BAL, LX	SETADR	SET UP ONE ADR CELL
	BIR, X	SETADR1	SET UP ALL THREE OF 'EM
	B	*L2	RETURN
SETADR	LI, S	1	SIZE OF 1 FOR VECTOR/SCALAR TESTS
SETADR1	EQU	%	
3Z1	LW, A	RESULT+1, X	GET ARG WORD
	LW, T	RSTYPE+1, X	INITIALIZE ARG/RESULT ADDRESSES
	B	3Z12, T	..DEPENDS ON DATA TYPE
3Z12	TABLE	WORDLOGL	
	B	3Z11	WORD LOGICAL
	B	3Z10	LOGL

	B	3Z9	CHAR	
	B	3Z6	INTG	
	B	3Z4	FLOT	
	LW,R	ISEQADR,X	ISEQ, SET ARG ADR = ADR OF	
	STW,R	RSADR+1,X	ISEQ ELEMENT CALC ROUTINE.	
	AI,A	4	COMPUTE ISEQ STEP-VALUE LOC	
	STW,A	LFSTEPAD+3,X	SAVE IT FOR ISEQ ELEMENT ROUTINE	
	LW,R	-1,A	FETCH ISEQ BASE VALUE	
	STW,R	LFVALU+3,X	STORE IT FOR ISEQ ELEMENT ROUTINE	
	B	0,LX	RETURN	
3Z4	EQU	%		
	CW,S	RSSIZE+1,X	FLOT, IS IT SCALAR?	
	BE	3Z2	YES, USE ARG+RANK+2	
	INT,AI	0,A	GET DB SIZE	
	AW,A	AI		
	AI,A	N**17	USE ARG+(DB SIZE),N	
	B	3Z3		
3Z11	LI,T	WORDLOGL	WORD LOGICAL: CHANGE RSTYPE	
*	STW,T	RSTYPE	TO MATCH ARGS.	
			IF IT'S A SCALAR OR 1-ELMT	
	CW,S	RSSIZE+1,X	THING, CHANGE ITS TYPE	
	BE	3Z13	TO (BIT) LOGL & DO NORMALLY.	
	LW,R	RSSIZE+1,X	OTHERWISE, KEEP WORDLOGL TYPE,	
	AI,R	31	CHANGE SIZES FROM BIT COUNT	
	SLS,R	-5	TO WORD COUNT; THEN CONTINUE AS	
	STW,R	RSSIZE+1,X	FOR INTEGER TYPES.	
3Z6	CW,S	RSSIZE+1,X	INTG: IS IT SCALAR?	
	BE	3Z8	YES, USE ARG+RANK+2	
	AW,A	RSSIZE+1,X	NO, USE ARG+SIZE+RANK+2,N	
3Z7	AW,A	RSRANK+1,X	ADD ARG RANK TO ITS ADR	
	AI,A	N**17+2	ADD 2 AND INDEX FIELD	
	STW,A	RSADR+1,X	STORE ADDRESS	
	B	0,LX	DONE	
3Z9	CW,S	RSSIZE+1,X	CHAR: IS IT SCALAR?	
	BE	3Z8	YES, USE ARG+RANK+2	
	LW,A	RSSIZE+1,X	NO, ADVANCE SIZE TO NEXT	
	AI,A	3	MULTIPLE OF FOUR CHARS.	
	AND,A	=-4		
	STW,A	RSSIZE+1,X		
	SLS,A	-2	USE ARG+CEILING(SIZE/4)+RANK+2,N	
	AW,A	RESULT+1,X		
	B	3Z7		
3Z13	MTW, LOGL	-WORDLOGL RSTYPE+1,X		
3Z10	LW,R	LOGLADR,X	LOGL: USE ADDRESS OF APPROPRIATE	
	CW,S	RSSIZE+1,X	(SPECIAL CASE LOGL SCALAR)	08-00009
	BNE	3Z14		08-00010
	LW,R	SLOGLADR,X		08-00011
	AI,A	1		08-00012
3Z14	EQU	%		08-00013
	STW,R	RSADR+1,X	LOGICAL LOAD/STORE ROUTINE	
	LI,R	1	INITIALIZE LOGL COUNT TO 1	
	STW,R	RSLGLCNT+1,X		
	AW,A	RSRANK+1,X	INITIALIZE LOGL ADDRESS TO	
	AI,A	1	ARG+RANK+1	
	STW,A	RSLGLADR+1,X		
	B	0,LX	DONE	
3Z2	AW,A	RSRANK+1,X	FLOT SCALAR: USE	
	AI,A	3	ARG+RANK+(2 OR 3)	
	AND,A	=-2	(WHICHEVER IS EVEN)	
	STW,A	RSADR+1,X		
	B	0,LX		
3Z8	AW,A	RSRANK+1,X	SCALAR	
	AI,A	2	USE ARG+RANK+2	
3Z3	STW,A	RSADR+1,X	STORE ADR	
	B	0,LX	RETURN	
*				
*				
*	GET ARG SIZE			
*				
*		GETS SIZE OF LEFT (A=0) OR RIGHT (A=1) ARG TO 'S'		
*		AND 'LFSIZE'/'RTSIZE'; COPIES ARG RANK TO 'RSRANK'.		
*		LINK IS 'LX'. IF SIZE=1, IT RETURNS TO BAL+2;		
*		OTHERWISE, RETURNS TO BAL+1.		
*		'GETLSIZE' IS ALTERNATE ENTRY WHICH SETS A=0 (LEFT).		
*				
GETLSIZE	LI,A	0	SET UP LF ARG INDEX	
GETSIZE	LI,S	1	1 TO COMPARE AGAINST DIMENS	
	LW,R	LFRANK,A	GET ARG RANK	

	STW,R	RSRANK	SET TENTATIVE RESULT RANK
	BEZ	ZZ6	HANDLE ZERO-RANK CASE
	LW,X	LFARG,A	GET DIMEN PTR
	AI,X	1	POINT TO WORD BEFORE 1ST DIMEN
2Z1	CW,S	*R,X	1:DIM
	BL	ZZ3	1<DIM, START MULTIPLYING (SIZE>1)
	BE	ZZ2	1=DIM, CONTINUE IN THIS LOOP
	LI,S	0	1>DIM (DIM=0), STOP
	STW,S	LFSIZE,A	SET SIZE=0
	B	0,LX	RETURN TO 'NOT EQUAL 1' LOC
2Z2	BDR,R	ZZ1	
2Z6	STW,S	LFSIZE,A	SET ARG SIZE=1
	B	1,LX	SIZE=1 ; RETURN
2Z3	LW,S	*R,X	DIM>1, INITIALIZE SIZE=DIM
2Z4	BDR,R	ZZ5	ARE THERE ANY MORE ?
	STW,S	LFSIZE,A	SET SIZE >1
	B	0,LX	NO, RETURN TO 'NOT EQUAL 1' LOC
2Z5	ODD,S		
	MW,S	*R,X	SIZE = SIZE*DIM
	BNEZ	ZZ4	CONTINUE IF NONZERO (SIZE STILL >1)
	STW,S	LFSIZE,A	SET SIZE=0
	B	0,LX	RETURN, SIZE NOT 1 (=0)
	PAGE		

*
*
*
*
*

TEMPS FOR ARG SETUP

LFADR	TEMP	LEFT ARG ADDRESS
RTADR	TEMP	RIGHT ARG ADDRESS
RSADR	TEMP	RESULT ADDRESS
LFLGLADR	TEMP	LOGICAL WORD ADR FOR LEFT ARG
RTLGLADR	TEMP	LOGICAL WORD ADR FOR RIGHT ARG
RSLGLADR	TEMP	LOGICAL WORD ADR FOR RESULT
LFLGLCNT	TEMP	LOGICAL BIT COUNT FOR LEFT ARG
RTLGLCNT	TEMP	LOGICAL BIT COUNT FOR RIGHT ARG
RSLGLCNT	TEMP	LOGICAL BIT COUNT FOR RESULT
LFSTEPAD	TEMP	LEFT ISEQ STEP ADDRESS
RTSTEPAD	TEMP	RIGHT ISEQ STEP ADDRESS
LFVALU	TEMP	LEFT ISEQ ELEMENT VALUE
RTVALU	TEMP	RIGHT ISEQ ELEMENT VALUE
LFRANK	TEMP	RANK OF LEFT ARG
RTRANK	TEMP	RANK OF RIGHT ARG
RSRANK	TEMP	RANK OF RESULT
LFSIZE	TEMP	SIZE OF LEFT ARG
RTSIZE	TEMP	SIZE OF RIGHT ARG
RSSIZE	TEMP	SIZE OF RESULT
LFTYPE	TEMP	TYPE OF LEFT ARG
RTTYPE	TEMP	TYPE OF RIGHT ARG
RSTYPE	TEMP	TYPE OF RESULT
	PAGE	

*
*
*
*
*

X S E G G E N R O U T I N E S

*
*
*
*
*
*
*

GEN XSEG INITIALIZATION CODE

GENS 'LCW,N RSSIZE' AND DEFINES LOOP-TOP LOC.
LINK IS LX. IF THE RESULT IS NULL, THIS ROUTINE EXITS
THROUGH 'RETURN'.

GXSEGINI	LI,XL	XSEGBASE	INIT XSEG LOC COUNTER
	GEN,0,1	INITINST	GEN 'LCW,N RSSIZE'
	STW,XL	LOOPLOC	DEFINE TOP-OF-LOOP LOC HERE
	LW,S	RSSIZE	IS RESULT NULL ?
	BNEZ	0,LX	NO, RETURN
	B	*RETURN	YES, EXIT FROM ENTIRE DRIVER

*
*
*
*
*

GEN MONADIC LOAD/CONVERT

*
*
*
*
*
*
*

SETS UP THE XSEG LOC COUNTER AND GENS:
(1) LOOP INITIALIZATION
(2) LOAD OF ARG
(3) CONVERSION TO RESULT TYPE, IF NECESSARY
ALSO, IT ESTABLISHES THE LOOP-TOP LOC JUST AFTER (1).
LINK IS L1.

```

*
*
GXSEGL EQU %
BAL,LX GXSEGINI GEN INIT, SET LOOPLOC
LI,A 1 GEN LOAD RIGHT ARG, CONVERT
B GENLOAD TO RSTYPE; RETURN.
PAGE

*
*
* GEN DYADIC LOADS/CONVERTS
*
* SETS UP THE XSEG LOC COUNTER AND GENS THE FOLLOWING:
* (1) LOOP INITIALIZATION
* (2) CONVERSIONS (IF NECESSARY) OF EACH ARG TO
* THE RESULT TYPE
* (3) LOAD OF THE LEFT ARG INTO THE 'A' REG
* IN ADDITION, THE ROUTINE DOES THE FOLLOWING:
* (4) ESTABLISHES THE LOOP-TOP LOC (LEAVING STUFF
* OUTSIDE THE LOOP WHEREVER POSSIBLE); ITEMS
* (2) AND (4) MAY IMPLY GENNING LOADS/STORES
* TO TEMPS
* (5) LEAVES THE RIGHT ARG ADDRESS (WHICH MAY HAVE
* BEEN CHANGED TO A TEMP ADR) IN 'R'.
* LINK IS L3.
*
*
GXSEGL EQU % GEN XSEG LOADS/CONVERTS
BAL,LX GXSEGINI GEN XSEG INIT CODE
LW,X COPTRIG IF WE'RE DOING A COMPOSITE OP,
BLZ *CLOADLNK DERAIL TO SPECIAL SUBROUTINE.
LI,S 1 SIZE OF 1 FOR VECTOR/SCALAR TESTS
CW,S LFSIZE
BNE 6Z1 IF LEFT ARG IS A SCALAR,
LI,A 0 LOAD IT TO ITS TEMP.
BAL,L2 GENLOADT
LI,S 1 (GENLOADT MAY CLOBBER S)
6Z1 CW,S RTSIZE
BNE 6Z2 IF RIGHT ARG IS A SCALAR,
LI,A 1 LOAD IT TO ITS TEMP,
BAL,L2 GENLOADT THEN ESTABLISH 'TOP OF
STW,XL LOOPLOC LOOP' HERE.
B 6Z3
6Z2 STW,XL LOOPLOC IF RIGHT ARG IS VECTOR, 'TOP OF
LW,T RTTYPE LOOP' IS HERE;
CW,T RSTYPE IF IT'S OF THE WRONG TYPE,
BNE 6Z4 OR IF IT'S LOGL,
CI,T LOGL
BNE 6Z3 LOAD (AND CONVERT) IT INTO
GXSEGL1 EQU % ITS TEMP.
6Z4 LI,A 1
BAL,L2 GENLOADT
6Z3 LI,A 0 LOAD UP LEFT ARG
BAL,L1 GENLOAD
LW,R RTADR RETURN WITH R= RT ADR
B *L3 RETURN

*
*
* GEN STORE AND LOOP CONTROL
*
* GENS STORE INTO RESULT, FOLLOWED BY BIR TO LOOP-TOP
* AND XSEG EXIT INST. LINK IS LX.
*
GXSEGST EQU % GEN XSEG STORE, LOOP CONT
LW,R RSADR GEN STORE INTO RESULT
LW,T RSTYPE
AW,R STORINST,T
GEN,1,0 R
LW,R LOOPINST
AW,R LOOPLOC GEN LOOP CONTROL INST
GEN,1,1 R,EXITINST GEN LOOP CONTROL AND EXIT CODE
B 0,LX RETURN
PAGE

*
*
* GEN LOAD
*
* GENS LOAD OF LEFT (A=0) OR RIGHT (A=1) ARG FOLLOWED,
* IF NECESSARY, BY CONVERSION TO 'RSTYPE'. LINK IS L1.

```

```

*
GENLOAD  LW,R      LFADR,A      GET ADDRESS/INDEX FIELDS
          LW,T      LFTYPE,A
          CLM,R      TEMPADR
          BCS,9      18Z1      IF ADR IS LFTEMP/RTTEMP,
          AW,R      LTMPINST,T    GEN 'LOAD TEMP' INST (DIFFERS
          B          18Z2      FROM 'LOADINST' ONLY FOR LOGL.)
18Z1      AW,R      LOADINST,T    INSERT OP/REG FIELDS FOR LOAD
18Z2      GEN,1,0    R          GEN LOAD INST
          CW,T      RSTYPE      IS TYPE CORRECT ?
          BE        *L1        YES, RETURN
          SLS,T      2          NO, GEN TYPE CONVERSION:
          AW,T      RSTYPE      T = 4*(ARG TYPE)+(RESULT TYPE),
          GENX      CONVTabl,T  GEN CONVERSION CODE
          B          *L1        RETURN

*
*
*   GEN LOAD TO TEMP
*
*       GENS LOAD AND CONVERT, AS ABOVE, FOLLOWED BY STORE
*       INTO TEMP; SUBSTITUTES TEMP ADR FOR ARG ADR. LINK IS L2.
*
GENLOADT  BAL,L1    GENLOAD      GEN LOAD AND CONVERSION TO 'RSTYPE'
          LW,R      TEMPADR,A    SUBSTITUTE TEMP ADR
          STW,R      LFADR,A      FOR ARG ADR.
          LW,T      RSTYPE
          STW,T      LFTYPE,A    SUBSTITUTE RESULT TYPE
          AW,R      STMPINST,T   GEN STORE INTO TEMP
          GEN,1,0    R          PUT IN XSEG
          B          *L2        RETURN

*
*
*   GEN CHANGE OF REGS OR OP CODE
*
*       IF THE LAST GENNED INST WAS A LOAD INST, ITS OP CODE
*       AND REG FIELDS WILL BE CHANGED TO THOSE GIVEN IN 'IX';
*       OTHERWISE, THE INST IN 'IX' WILL BE GENNED. LINK IS LX.
*
CHANGOP   EQU      %
CHANGREG  LW,IX1    =X'56000000'  IS LAST GEN'ED INST A LOAD ?
          CS,IX      -1,XL
          BE        7Z1
          GEN,1,0    IX          NO, GEN THE WHOLE INST
          B          0,LX
7Z1      LW,IX1    =X'7FF00000'  YES, CHANGE OP/REG FIELDS
          STS,IX      -1,XL
          B          0,LX

*
LREGBI    LW,BI     AI          CHANGE AI TO BI
LREGTX    LW,TX     AI          CHANGE AI TO TX
          PAGE

*
*
*   DATA/TEMPS FOR XSEG GEN ROUTINES
*
*
LOOPLOC    TEMP      ADDRESS IN XSEG OF LOOP TOP
LFTEMP     DTEMP     LEFT ARG TEMP
RTTEMP     DTEMP     RIGHT ARG TEMP
*
*
LOADINST   TABLE    WORDLOGL    LOAD INSTRUCTION - BY ARG TYPE:
          LW,AI      0          WORD LOGICAL
          BAL,LX     0          LOGL
          LB,AI      0          CHAR
          LW,AI      0          INTG
          LD,AF      0          FLOT
          BAL,LX     0          ISEQ
*
LODBINST   TABLE    CHAR        LOAD 2ND ACCUM - BY ARG TYPE:
          LB,BI      0          CHAR
          LW,BI      0          INTG
          LD,BF      0          FLOT
*
LTMPINST   TABLE    WORDLOGL    LOAD FROM TEMP INST - BY ARG TYPE:
          LW,AI      0          WORD LOGICAL
          LW,AI      0          LOGL
          LB,AI      0          CHAR

```

	LW, AI	0	INTG	
	LD, AF	0	FLOT	
	LW, AI	0	ISEQ	
*				
STORINST	TABLE	WORDLOGL	STORE INSTRUCTION - BY RESULT TYPE:	
	STW, AI	0	WORD LOGICAL	
	BAL, LX	0	LOGL	
	STB, AI	0	CHAR	
	STW, AI	0	INTG	
	STD, AF	0	FLOT	
*				
STMPINST	TABLE	WORDLOGL	STORE TO TEMP INST - BY RS TYPE:	
	STW, AI	0	WORD LOGICAL	
	STW, AI	0	LOGL	
	STB, AI	0	CHAR	
	STW, AI	0	INTG	
	STD, AF	0	FLOT	
*				
LOGLADR	TABLE	-3	LOGL LOAD/STORE ROUTINES	
	PZE	LDLOGLLF	LOAD LOGICAL LEFT ARG	
	PZE	LDLOGLRT	LOAD LOGICAL RIGHT ARG	
	PZE	STLOGLRS	STORE LOGICAL RESULT	
*				
SLOGLADR	TABLE	-3	SCALAR LOGL LOAD/STORE ROUTINES	08-00015
	PZE	LDLGLLF	LOAD SCALAR LOGICAL LEFT	08-00016
	PZE	LDLGLRT	LOAD SCALAR LOGICAL RIGHT	08-00017
	PZE	STLGLRS	STORE SCALAR LOGICAL RESULT	08-00018
*				
ISEQADR	TABLE	-3	ISEQ LOAD ROUTINES	
	PZE	LDISEQLF	LOAD ISEQ ELEMENT LEFT	
	PZE	LDISEQRT	LOAD ISEQ ELEMENT RIGHT	
*				
TEMPADR	BOUND	8		
	TABLE	0	ARG TEMP ADDRESS	
	PZE	LFTEMP	LF	
	PZE	RTTEMP	RT	
*				
CONVTABL	TABLE	4*LOGL+WORDLOGL	CONVERSION CODE GROUPS - BY 2 TYPES:	
	CODE, 0		L TO WORD LOGICAL	
	CODE, 0		L TO L	
	CODE, 1	TYPEXTOC	L TO C (FOR EQ/NEQ OPS)	
	CODE, 1	TYPELTOI	L TO I	
	CODE, 1	TYPELTOF	L TO F	
	CODE, 0		C TO L ILLEGAL	
	CODE, 0		C TO C	
	CODE, 0		C TO I ILLEGAL	
	CODE, 0		C TO F ILLEGAL	
	CODE, 3	TYPEITOL	I TO L	
	CODE, 1	TYPEXTOC	I TO C (FOR EQ/NEQ OPS)	
	CODE, 0		I TO I	
	CODE, 5	TYPEITOF	I TO F	
	CODE, 5	TYPEFTOL	F TO L	
	CODE, 1	TYPEXTOC	F TO C (FOR EQ/NEQ OPS)	
	CODE, 2	TYPEFTOI	F TO I	
	CODE, 0		F TO F	
	CODE, 3	TYPEITOL	ISEQ TO L	
	CODE, 1	TYPEXTOC	ISEQ TO C	
	CODE, 0		ISEQ TO I	
	CODE, 5	TYPEITOF	ISEQ TO F	
*				
TYPELTOI	AND, AI	=1	INTG VAL = 0 OR 1	
*				
TYPELTOF	LD, AF	FLOT01, AI	CONV LOGL 0/1 TO FLOT 0.0/1.0	
*				
TYPEITOF	LW, AF	AI	GET INTG VALUE	
	LI, AF1	0	CLEAR 2ND WORD	
	SAD, AF	-8	MAKE ROOM FOR EXPONENT	
	EOR, AF	=X'48000000'	INCLUDE EXPONENT	
	FAL, AF	FLOT0	NORMALIZE	
*				
TYPEFTOL	EQU	%		
*				
TYPEFTOI	BAL, LX	F2I	F TO I CONVERSION SUBROUTINE	
	B	ERDOMAIN	ERROR IF NOT INTG VALUE	
*				
TYPEITOL	CI, AI	-2	ERROR IF INTG NOT 0/1	
	BANZ	ERDOMAIN		
	LCW, AI	AI	SET TO 0 OR -1	

```

*
TYPEXTOC LI, AI      -1          WON'T MATCH ANY CHAR
*
INITINST LCW, N      RSSIZE      XSEG LOOP INIT INST
LOOPINST BIR, N      0          XSEG LOOP CONTROL INST
EXITINST B          *RETURN      XSEG EXIT INST
*
      BOUND          8
FLOTCONS DATA, 8    FL'2', FL'1', FL'0'
FLOT0 EQU          FLOTCONS+4    FLOATING 0.0
FLOT1 EQU          FLOTCONS+2    FLOATING 1.0
FLOT2 EQU          FLOTCONS      FLOATING 2.0
FLOT01 EQU         FLOT0         FLOATING 0.0, 1.0
FLOTPI DATA, 8     FL'3.141592653589793'
FLOTINF DATA       X'7FFFFFFF', X'FFFFFFF'
      PAGE
*
*
*   X S E G   E X E C U T I O N   R O U T I N E S
*
*   GEN INTEGER OVERFLOW TEST; GEN STORE; EXECUTE
*
GXOVSTEX GEN, 0, 2   OVTSTINS      GEN:      BNOV      %+2
      AWM, XL      -2, XL          B          INTGOVFL
*
*
*   GEN STORE; EXECUTE
*
GXSTEXEC LW, X      COPTRIG      IF WE'RE DOING A COMPOSITE OP,
      BLZ          *CSTORLNK      DERAIL TO SPECIAL SUBROUTINE.
GXSTEXC1 BAL, LX    GXSEGST      GEN STORE, BY RESULT TYPE
*
*   EXECUTE XSEG
*
EXECUTE LI, IX      OPBREAK      SET XSEG BREAK FLAG TO SAY,
      STW, IX      XSEGBRK      'XSEG BREAK IS OK NOW.'
      B          XSEGBASE      GO TO THE XSEG
*
*
*   LOAD/STORE LOGICAL DATA
*
*           LDLOGLLF/RT LOADS INTO ALL BIT POSITIONS OF AI
*           THE NEXT BIT OF LOGICAL DATA FROM LEFT/RIGHT
*           ARG. STLOGLRS STORES AI (WHICH MUST BE 0 OR -1)
*           INTO THE NEXT LOGICAL RESULT BIT. LINK IS LX.
*
LDLOGLLF LI, AF      0          AF= LEFT ARG INDEX
      B          10Z1
*
LDLOGLRT LI, AF      1          AF= RIGHT ARG INDEX
10Z1 LW, AI      LFLGLCNT, AF    GET COUNT (NR BITS REMAINING +1)
      BDR, AI      10Z2        DECR IT
      LI, AI      32          NO MORE BITS, RESET COUNT
      MTW, 1      LFLGLADR, AF  AND GO TO NEXT WORD OF BITS.
10Z2 STW, AI      LFLGLCNT, AF    STORE NEW COUNT
      LW, AF      LFLGLADR, AF  FETCH CURRENT DATA ADR
      LW, AI      BITMASK, AI   SELECT APPROPRIATE BIT
*           BITMASK+J = 2**(J-1) FOR 1<=J<=32
      AND, AI      0, AF        GET THE BIT
      BEZ          0, LX        IF IT'S ZERO, RETURN WITH AI=0
      LI, AI      -1          NONZERO,
      B          0, LX        RETURN WITH AI=-1.
*
STLOGLRS LW, AF      RSLGLCNT    GET COUNT (NR POSITIONS LEFT +1)
      BDR, AF      10Z3        DECR IT
      MTW, 1      RSLGLADR      NO MORE HOLES, RESET COUNT
      STW, AF      *RSLGLADR    AND MOVE TO NEXT FULL WORD.
      LI, AF      32          ZERO OUT NEW WORD
10Z3 STW, AF      RSLGLCNT    STORE NEW COUNT
      AND, AI      BITMASK, AF  SELECT APPROPRIATE BIT
      ODD, AI      SINCE AI IS ODD, 'STS, AI' DOES
      STS, AI      'OR TO MEMORY'
      B          *RSLGLADR
      B          0, LX        RETURN
*
*
LDSLGLLF LW, AI      *LFLGLADR    GET LEFT ARG

```

```

08-00021
08-00022
08-00023

```

10Z5	BGEZ LI, AI B	10Z4 -1 0, LX	TEST 1ST LOGL BIT 1: RETURN ALL 1'S	08-00024 08-00025 08-00026
*				08-00027
LDSLGLRT	LW, AI BLZ	*RTLGLADR 10Z5	GET RIGHT ARG TEST 1ST LOGL BIT	08-00028 08-00029
10Z4	LI, AI B	0 0, LX	0: RETURN ALL 0'S	08-00030 08-00031
*				08-00032
STSLGLRS	STW, AI B	*RSLGLADR 0, LX	STORE LOGL SCALAR	08-00033 08-00034
*				
*				
*	LOAD ISEQ ELEMENT			
*				
*		LDISEQLF/RT LOADS INTO AI THE NEXT ELEMENT FROM		
*		THE ISEQ ARG. LINK IS LX.		
*				
LDISEQLF	LW, AI AW, AI STW, AI B	LFVALU *LFSTEPAD LFVALU 0, LX	GET LAST VALUE ADD ISEQ STEP STORE AS NEXT VALUE AND RETURN THAT VALUE	
*				
LDISEQRT	LW, AI AW, AI STW, AI B	RTVALU *RTSTEPAD RTVALU 0, LX	GET LAST VALUE ADD ISEQ STEP STORE AS NEXT VALUE AND RETURN THAT VALUE	
*				
*				
OVTSTINS	BNOV B	0 (%+2) INTGOVFL	-2 -1	
*				
NTEMPS	SET	ERROR, X'F', TLOC>30	'TOO MANY TEMPS'	U08-0110
20Z	END	TLOC		U08-0111

TITLE 'MIX-B00,10/12/73,DWG702985'
PAGE

*
*
* E X T E R N A L C O M M U N I C A T I O N
*
*
*

* DEFINITIONS
*

DEF	CLOADLNK	DERAIL LINK FOR GXSEGD1	
DEF	COPTRIG	COMPOSITE OP TRIGGER	
DEF	CSETLNK	DERAIL LINK FOR DSETUP	
DEF	CSTORLNK	DERAIL LINK FOR GXSTEXEC	
DEF	CTYPELNK	DERAIL LINK FOR DTYPEIF	
DEF	DDEAL	DYADIC DEAL OP ROUTINE	
DEF	DDECODE	DYADIC DECODE OP ROUTINE	
DEF	DENCODE	DYADIC ENCODE OP ROUTINE	
DEF	DINDEXOF	DYADIC INDEX-OF OP ROUTINE	
DEF	DMEMBER	DYADIC MEMBERSHIP OP ROUTINE	
DEF	DRESHAPE	DYADIC RESHAPE OP ROUTINE	
DEF	INNER	INNER PRODUCT EXECUTION DRIVER	
DEF	MDIMEN	MONADIC DIMENSION OP ROUTINE	
DEF	MGRADEDN	MONADIC GRADE DOWN OP ROUTINE	
DEF	MGRADEUP	MONADIC GRADE UP OP ROUTINE	
DEF	MINDEX	MONADIC INDEX GENERATOR OP ROUTINE	
DEF	MIX@	START OF PROCEDURE	
DEF	MRABEL	MONADIC RABEL OP ROUTINE	
DEF	OUTER	OUTER PRODUCT EXECUTION DRIVER	
DEF	REDUCE	REDUCTION EXECUTION DRIVER	
DEF	RETURN	RETURN ADR CELL	
DEF	SCAN	SCAN EXECUTION DRIVER	U09-0004
DEF	SETUPARG	SET UP ARG PARAMS	
DEF	TYCOMPAT	TYPE COMPATIBILITY CHECK	
DEF	VECTORRS	ALOC VECTOR RESULT	

* REFERENCES
*
*

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS
REF	ALOCRS	ALLOCATE RESULT DATA BLOCK
REF	BALCOMP	'BAL,LX FFCOMPAR' INSTRUCTION
REF	BASEADR	BASE ADR FOR SHORT ADR OFFSETS
REF	CHAREQ	CHARACTER EQ/NEQ
REF	COMPINST	COMPARE INST TABLE
REF	CONVTABL	TYPE CONVERSION CODE TABLE
REF	COORDK	K'TH COORDINATE SPEC
REF	DBUFEND	DIMENSION BUFFER END
REF	DREF	DE-REF
REF	DTYPEIF	INTG/FLOT TYPE SETUP
REF	DTYPEIF1	* (ALT. ENTRY)
REF	DTYPEIF2	* (ALT. ENTRY)
REF	DXRETURN	DYADIC EXEC DRIVER RETURN
REF	DXTABLE	DYADIC OP ROUTINE ENTRY TABLE
REF	DYNBUND	UPPER BOUND OF DYNAMIC MEMORY
REF	ERDOMAIN	DOMAIN ERROR
REF	ERINDEX	INDEX RANGE ERROR
REF	ERLENGTH	LENGTH ERROR
REF	ERRANK	RANK ERROR
REF	EXCHLUPS	EXCHANGE MIDDLE/INNER LOOPS
REF	EXECUTE	EXECUTE XSEG
REF	FFRESIDU	FLOT RESIDUE EVALUATOR
REF	FLOTINF	FLOATING POINT INFINITY
REF	FLOT0	FLOATING POINT 0.0
REF	FLOT01	FLOATING POINT 0.0, 1.0
REF	FLOT1	FLOATING POINT 1.0
REF	FREETOTL	NR WORDS FREE IN DYNAMIC MEMORY
REF	F2I	CONVERT F TO I
REF	GENLOAD	GEN LOAD BY RSTYPER
REF	GENLOADT	GEN LOAD TO TEMP
REF	GIVEBACK	GIVE BACK PART OF DATA BLOCK
REF	GSCLRVAL	GET SCALAR INTEGER VALUE
REF	GXSEGD1	GEN DYADIC LOAD (ALT. ENTRY)
REF	GXSEGINI	GEN XSEG INITIALIZATION
REF	GXSEGML	GEN MONADIC LOAD
REF	GXSTEXC1	GEN STORE/EXECUTE (ALT. ENTRY)
REF	GXSTEXEC	GEN XSEG STORE; EXECUTE XSEG
REF	IIRRESIDU	INTG RESIDUE EVALUATOR
REF	INDXLDLR	INDEXED LOAD LOGICAL RIGHT
REF	INERCNT	INNER LOOP COUNT

	REF	INNERSTEP	INNER LOOP STEP SIZE	
	REF	INTGOVFL	INTEGER OVERFLOW (DOMAIN CHANGE)	
	REF	IROLL	INTEGER ROLL FUNCTION EVALUATOR	
	REF	LFADR	LEFT ARG ADDRESS	
	REF	LFARG	LEFT ARG PNTR	
	REF	LFLGLADR	LEFT LOGICAL ADDRESS	
	REF	LFRANK	LEFT ARG RANK	
	REF	LFSIZE	LEFT ARG SIZE	
	REF	LFTEMP	LEFT ARG VALUE TEMP	
	REF	LFTYPE	LEFT ARG TYPE	
	REF	LOADINST	LOAD INSTRUCTION TABLE, BY TYPE	
	REF	LODBINST	LOAD 2ND ACCUM INST TABLE	
	REF	LOOPLOC	LOOP LOCATION	
	REF	MAXDIMEN	MAXIMUM NUMBER OF DIMENSIONS	
	REF	MBUFDIMS	MOVE BUFFER DIMENSIONS TO RESULT	
	REF	MIDLCNT	MIDDLE LOOP COUNT	
	REF	MIDLSAVE	MIDDLE LOOP SAVE TEMP	
	REF	MIDLSTEP	MIDDLE LOOP STEP SIZE	
	REF	MIXTEMPS	TEMPS ARE IN WINDOW IN APLUTSI	U09-0006
	REF	MNOP	MONADIC NO OP ROUTINE	
	REF	MRTDIMS	COPY RTARG DIMENS TO RESULT	
	REF	MXRETURN	MONADIC EXEC DRIVER RETURN	
SCRIPT	REF	NILCK	'NIL CHECK' = SCRIPT LIST PNTR	
	EQU	NILCK	SUBSCRIPT LIST POINTER	
	REF	OPBREAK	OP BREAK HANDLER	
	REF	OPER	OPERATOR WORDS	
	REF	ORGADJ	ORIGIN ADJUSTED (1-ORIGIN)	
	REF	ORIGIN	INDEX ORIGIN VALUE (0 OR 1)	
	REF	OUTRCNT	OUTER LOOP COUNT	
	REF	OUTRSAVE	OUTER LOOP SAVE TEMP	
	REF	OUTRSTEP	OUTER LOOP STEP SIZE	
	REF	RANDOM	RANDOM SEED	
	REF	RESULT	RESULT DATA BLOCK POINTER	
	REF	RSADR	RESULT ADDRESS	
	REF	RSLIKRT1	ALOC RESULT LIKE RTARG	
	REF	RSLIKRT2	ALOC RESULT LIKE RTARG	U09-0008
	REF	RSRANK	RESULT RANK	
	REF	RSSIZE	RESULT SIZE	
	REF	RSTYPE	RESULT TYPE	
	REF	RTADR	RIGHT ARG ADDRESS	
	REF	RTARG	RIGHT ARG PNTR	
	REF	RTRANK	RIGHT ARG RANK	
	REF	RTSIZE	RIGHT ARG SIZE	
	REF	RTTEMP	RIGHT ARG VALUE TEMP	
	REF	RTTYPE	RIGHT ARG TYPE	
	REF	SETADR	SET ARG ADR CELL	
	REF	SETADRS1	SET UP ADDRESS(ES), SEQUENTIAL	
	REF	SETSPEC1	SPECIAL ADR SETUP ROUTINE	
	REF	SETSPEC2	SPECIAL ADR SETUP ROUTINE	
	REF	STCCSEQ	STORE CC CODE SEQ	
	REF	STMPINST	STORE IN TEMP INST TABLE	
	REF	STORINST	STORE INSTRUCTION TABLE	
	REF	ST3LUPSN	SET LOOP PARAMS; NO ALOC	
	REF	SXRETURN	SUBSCRIPTED EXP EXEC DRIVER RETURN	
	REF	SYSTERR	SYSTEM ERROR	
	REF	TOPOSTAK	LOWER BOUND OF STACK MEMORY	
	REF	TYPETEMP	TYPE TEMP	U09-0010
	REF	XSEGBASE	BASE OF XSEG AREA	
	REF	XSEGBRK	XSEG BREAK FLAG	
	REF	XSETUP	SET UP ADR FOR INDEXED ACCESS	
		PAGE		

*
*
*
*
*

A S S E M B L Y P A R A M E T E R S

	SYSTEM	SIG5F	
PROGSECT	CSECT	1	
MIX@	RES	0	START OF PROCEDURE

*
*
*

REGISTERS

IX	EQU	0	INTERPRET REG PAIR
IX1	EQU	1	*
X	EQU	1	SCRIPT POINTER
N	EQU	1	XSEG EXECUTION REG
T	EQU	2	TYPE REG
K	EQU	2	XSEG EXECUTION REG

XL	EQU	3	XSEG LOC REG
K1	EQU	4	XSEG EXECUTION REG
N1	EQU	4	XSEG EXECUTION REG
N2	EQU	11	XSEG EXECUTION REG
N3	EQU	10	XSEG EXECUTION REG
A	EQU	4	ARG ADR/INDEX
LZ	EQU	4	INDEX LINK REG
LX	EQU	5	INDEX LINK REG
OP	EQU	6	OP CODE REG
AF	EQU	6	ACCUM FOR FLOT VALUES
LX7	EQU	7	INDEX LINK REG
AI	EQU	7	ACCUM FOR LOGL/CHAR/INTG VALUES
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS
R	EQU	8	GENERAL WORK REG
BF	EQU	8	2ND ACCUM FOR FLOT VALUES
BI	EQU	9	2ND ACCUM FOR L/C/I VALUES
S	EQU	11	SIZE
L3	EQU	12	LINK REG
L2	EQU	13	LINK REG
L1	EQU	14	LINK REG

* ARG TYPE CODES

WORDLOGL	EQU	0	WORD LOGICAL (WORD)
LOGL	EQU	1	LOGICAL (BIT)
CHAR	EQU	2	CHARACTER (BYTE)
INTG	EQU	3	INTEGER (WORD)
FLOT	EQU	4	FLOATING (DOUBLEWORD)
ISEQ	EQU	5	INDEX SEQUENCE VECTOR
LIST	EQU	6	LIST

* CODESTRING DESIGNATIONS

MOPGRDUP	EQU	75	MONADIC GRADE UP
DOPADD	EQU	91	DYADIC ADD
DOPLESS	EQU	102	DYADIC LESS
DOPNEQ	EQU	106	DYADIC NOT EQUAL
DOPEQUAL	EQU	107	DYADIC EQUAL
DOPAND	EQU	108	DYADIC AND
DOPNOR	EQU	111	DYADIC NOR
DOPMEMBR	EQU	120	DYADIC MEMBERSHIP
	PAGE		

* P R O C S

TLOC	SET	0	U09-0012
TEMP	CNAME	1	
DTEMP	CNAME	2	
	PROC		
	DO1	NAME=2	
TLOC	SET	TLOC+(TLOC&1)	U09-0015
	DISP	TLOC	U09-0016
LF	EQU	MIXTEMPS+TLOC	U09-0017
TLOC	SET	TLOC+NAME	U09-0018
	PEND		

EVEN	CNAME	0	
ODD	CNAME	1	
	PROC		
LF	EQU	%	
	ERROR, 1, (CF(2)+NAME)&1		'REGISTER HAS WRONG PARITY'
	PEND		

EQUAL	CNAME		
	PROC		
LF	EQU	%	
	ERROR, 1, 1-(CF(2)=CF(3))		'REGISTERS MUST BE EQUAL'
	PEND		

EXCHANGE	CNAME		
	OPEN	I, K, GROUP	
GROUP	EQU	LFARG, LFTYPE, LFSIZE, LFRANK, LFADR	

```

PROC
LF EQU %
I DO NUM(AF)
K SET SCOR(AF(I), ARGS, TYPES, SIZES, RANKS, ADRS)
ERROR, 1, K=0 'UNKNOWN GROUP INDICATOR'
LW, R GROUP(K)
XW, R GROUP(K)+1
STW, R GROUP(K)
FIN
PEND
CLOSE I, K, GROUP

*
*
NB CNAME X'680'
NBGE CNAME X'681'
NBLE CNAME X'682'
NBE CNAME X'683'
NBL CNAME X'691'
NBG CNAME X'692'
NBNE CNAME X'693'
PROC
ERROR, 1, (AF>=0)+(NUM(AF)>1) 'AF MUST BE NEG CONST ADR'
LF GEN, 12, 20 NAME-1, AF
PEND

*
*
ISEQFIX CNAME
PROC
LF CI, CF(2) ISEQ
BNE %+2
LI, CF(2) INTG
PEND
PAGE

*
*
* XSEG GEN PROCS
*
*
CODE CNAME
PROC
DO CF(2)>0
LF GEN, 4, 12, 16 CF(2), CF(2), AF(1)-BASEADR
ELSE
LF GEN, 32 0
FIN
PEND

*
*
GENX CNAME
PROC
LF INT, IX AF
BCR, 15 %+4
LM, BUF BASEADR, IX1
STM, BUF 0, XL
AW, XL IX
PEND

*
*
GEN OPEN GEN
CNAME
OPEN M, N, MN, I
PROC
LF EQU %
ERROR, 1, 1-(NUM(CF)=3) 'WRONG NUMBER OF CF ARGS'
M SET CF(2)
N SET CF(3)
MN SET M+N
ERROR, 1, 1-(NUM(AF)=(M>0)+(N>0)) 'WRONG NUMBER OF AF ARGS'
DO M>0
I DO N
LW, AF(1)+M+I-1 AF(2)+I-1
FIN
I DO MN*(MN<3)
STW, AF(1)+I-1 I-1, XL
ELSE
LCI MN
STM, AF(1) 0, XL
FIN

```

```

ELSE
DO      MN*(MN<3)
LW,BUF  AF(1)+I-1
STW,BUF I-1,XL
ELSE
LCI      N
LM,BUF   AF(1)
STM,BUF  0,XL
FIN
FIN
AI,XL    MN
PEND
CLOSE    M,N,MN,I
PAGE

```

```

*
*
*  TABLE BUILDING PROCS
*
*

```

```

TABLE    CNAME
OPEN      T,N
PROC
T         SET      %-AF(1)
LF        EQU      T
          DISP     T
          PEND

```

```

*
*
ITEM      CNAME
N         PROC
          SET      T+AF(1)-%
          ERROR,1,N<0      'ITEM OUT OF SEQUENCE'
          RES      N*(N>0)
          DISP     AF(1)
          PEND
          CLOSE    T,N
          PAGE

```

```

*
*
*  M O N A D I C      M I X E D      O P      R O U T I N E S
*
*

```

```

*          USECT      PROGSECT
*
MDIMEN    EQU      %          MONADIC DIMENSION
          LI,X      1
          LB,S      *RTARG,X  GET ARG RANK (= RESULT SIZE)
          STW,S      RSSIZE
          AI,S      1          ADD ONE FOR RESULT RANK
          BAL,LX7    ALOCHNW   ALLOCATE RESULT DB
          STW,A      RESULT    COPY RESULT POINTER
          LI,R      INTG**8+1  SET TYPE/RANK FOR INTEGER VECTOR
          STH,R      *RESULT
          LW,X      RSSIZE     SET SIZE = NUMBER OF DIMENS
          STW,X      2,A
          BEZ        *RETURN   IF THERE ARE NO DIMENS, WE'RE DONE
          MTW,1      RTARG      POINT TO WORD BEFORE 1ST DIMEN
          MTW,2      RESULT    POINT TO WORD BEFORE 1ST DATA WORD
1Z1       LW,R      *RTARG,X   MOVE A DIMEN
          STW,R      *RESULT,X
          BDR,X      1Z1       MOVE 'EM ALL
          MTW,-1     RTARG      RESTORE POINTERS TO NORMALCY
          MTW,-2     RESULT
          B          *RETURN   EXIT

```

```

*
*
MINDEX    EQU      %          MONADIC INDEX GENERATOR
          LI,S      3
          BAL,LX7    ALOCHNW   ALLOCATE RESULT DB
          STW,A      RESULT    COPY RESULT PNTR
          LI,R      ISEQ**8+1  SET TYPE/RANK = 'INDEX SEQUENCE
          STH,R      *RESULT    VECTOR'.
          BAL,LZ      INTSCALR  GET (INTEGER SCALAR) ARG VALUE (N)
          AI,AI      0
          BLZ        ERDOMAIN  WE REQUIRE N>=0
          LCW,AI+1    ORGADJ    SET:
          LI,AI+2    1          SIZE = N

```

	LW,A	RESULT	
	LCI	3	BASE = ORIGIN-1
	STM,AI	2,A	STEP = 1
	B	*RETURN	EXIT
*			
*			
MRAVEL	EQU	%	MONADIC RAVEL
	LI,A	1	IF ARG RANK =1,
	CB,A	*RTARG,A	RESULT = ARG.
	BE	MNOP	
	STW,A	RSRANK	SET RESULT RANK =1
	BAL,LX	SETUPARG	SET (RT) ARG RANK/SIZE/TYPE
	LW,S	RTSIZE	
	STW,S	RSSIZE	RESULT SIZE = ARG SIZE
	LW,T	RTTYPE	
	ISEQFIX,T		USE TYPE INTG INSTEAD OF ISEQ
	STW,T	RSTYPE	RESULT TYPE = ARG TYPE
	BAL,L1	ALOCRS	ALLOCATE RESULT DB
	LW,S	RSSIZE	SET RESULT'S ONE DIMENSION
	STW,S	2,A	= RESULT SIZE.
	LI,X	-2	SET UP ARG/RESULT ADR CELLS
	BAL,L2	SETADRS1	
	BAL,LX	GXSEGINI	INIT XSEG; EXIT IF RESULT NULL
	LI,A	1	
	BAL,L1	GENLOAD	GEN LOAD OF ARG
	B	GXSTEXEC	GEN STORE, LOOP; EXEC XSEG
	PAGE		
*			
*			
MGRADEUP	EQU	%	MONADIC GRADE UP
MGRADEDN	EQU	%	MONADIC GRADE DOWN
	LB,T	*RTARG	GET ARG TYPE
	LI,X	1	
	CB,X	*RTARG,X	CHECK ARG RANK:
	BE	6Z1-LOGL,T	=1 (VECTOR), GO TO JUMP TABLE;
	BL	ERRANK	>1 (ARRAY), NOT ALLOWED;
	LI,S	1	=0 (SCALAR), SET SIZE=1 AND
	B	6Z3	TREAT AS 1-ELEMENT VECTOR.
6Z1	B	6Z2	LOGL VECTOR: OK
	B	ERDOMAIN	CHAR VECTOR: NOT ALLOWED
	B	6Z2	INTG VECTOR: OK
	B	6Z2	FLOT VECTOR: OK
	B	6Z12	ISEQ: ALREADY SORTED
	B	ERDOMAIN	LIST: NOT ALLOWED
6Z2	LW,A	RTARG	L/I/F VECTOR: GET SIZE
	LW,S	2,A	
6Z3	STW,S	RSSIZE	RESULT SIZE = VECTOR LENGTH
	BAL,L2	INTVECRS	ALOC INTG VECTOR RESULT DB
	CI,S	1	CHECK SIZE:
	BG	6Z4	>1, TYPICAL;
	LW,AI	ORIGIN	=1, SET RESULT'S SOLE DATUM
	STW,AI	3,A	TO ORIGIN VALUE.
	B	*RETURN	
6Z4	LB,T	*RTARG	GET ARG TYPE
	CI,T	LOGL	IF IT'S INTG OR FLOT,
	BNE	FASTGRAD	USE THE FAST ALGORITHM.
	LI,A	1	SET UP RTADR FOR INDICIAL ACCESS
	BAL,LX	XSETUP	
	LI,X	-1	SET UP RSADR FOR SEQUENTIAL ACCESS
	BAL,LX	SETADR	
	LI,XL	XSEGBASE	PREPARE TO GEN XSEG
	LW,A	GRADINS4	LOGL VECTOR: GEN XSEG TO SWEEP
	LW,A+1	OPTBL3,OP	
	LW,A+2	GRADINS5	THROUGH ARG, MARKING IN RESULT
	AW,A+2	RSADR	THE K-INDECES OF ALL 0 (1) VALUES.
	GEN,3,6	A,GRADINS6	
	LCW,N	RSSIZE	INIT RESULT INDEX
6Z10	LI,K	0	INIT ARG INDEX.
	BAL,L1	EXECUTE	EXECUTE XSEG: MARK ALL 0'S (1'S)
	LW,OP	OPER	RESTORE OP CODE
	LW,R	OPTBL4,OP	NOW MODIFY XSEG TO MARK
	STW,R	XSEGBASE+1	ALL 1'S (0'S), AND EXECUTE
	B	6Z10	IT ONCE AGAIN. WHEN DONE, IT
			RETURNS HERE:
*			
APPLYORG	LW,AI	ORIGIN	IF ORIGIN NONZERO,
	BEZ	*RETURN	
	LCW,N	RSSIZE	ADD IT TO ALL RESULT DATA.
6Z11	AWM,AI	*RSADR,N	

```

        BIR,N      6Z11
        B          *RETURN
*
6Z12    LI,S       3                      ISEQ ARG: IT'S ALREADY MONOTONIC,
        BAL,LX7    ALOCHNW                ALL WE HAVE TO DO IS BUILD AN
        STW,A      RESULT                ISEQ RESULT DEFINING VALUES
        LI,R       ISEQ**8+1              (1,2,...,SIZE) OR (SIZE,...,2,1).
        STH,R      *RESULT
        LW,X       RTARG
        LW,R       2,X
        EXU        OPTBL5,OP              GET SIZE
        BGEZ       6Z13                  GET STEP (-STEP, FOR GRADE DOWN)
        LW,R+1     2,X
        LI,R+2     -1                     STEP<0: RESULT BASE=SIZE (0-ORIGIN)
        B          6Z14                  AND RESULT STEP = -1; DEFINES
        LI,R+1     -1                     (SIZE-1,...,1,0).
        LI,R+2     +1                     STEP>=0: RESULT BASE=-1, STEP=1;
        AW,R+1     ORIGIN                 DEFINES (0,1,...,SIZE-1).
6Z14    LCI        3                     APPLY ORIGIN TO BASE VALUE
        STM,R      2,A
        B          *RETURN              STORE RESULT SIZE/BASE/STEP
                                         RETURN
*
*
*

```

LOGL GRADE UP/DOWN XSEG CODE:

```

GRADINS4 BAL,LX    INDXLDLR              0
*          BLZ/BEZ  %+4                  1
GRADINS5 STW,K     0                     (RESULT(N)) 2
GRADINS6 BIR,N     XSEGBASE+5 (%+2)      3
        B          APPLYORG             4
        AI,K       1                     5
        CW,K       RSSIZE                6
        BL         XSEGBASE+0 (%-7)      7
        B          *L1                  8
*
*
OPTBL3   TABLE    MOPGRDUP              TABLE FOL LOGL SWEEP 0(1) INST
OPTBL4   EQU        OPTBL3+1              TABLE FOR LOGL SWEEP 1(0) INST
        BLZ        XSEGBASE+5            TBL3: UP
        BEZ        XSEGBASE+5            TBL3: DOWN      TBL4: UP
        BLZ        XSEGBASE+5            TBL4: DOWN
*
OPTBL5   TABLE    MOPGRDUP              TABLE FOR ISEQ STEP FETCH
        LW,R+2     4,X                   GRADE UP
        LCW,R+2    4,X                   GRADE DOWN
        PAGE
*
*

```

FAST INTEGER/FLOATING GRADE UP/DOWN

THIS ALGORITHM BUILDS THE RESULT BY PERFORMING BINARY MERGES OF SUCCESSIVELY LONGER SORTED STRINGS OF ARG ELEMENTS. AT THE K'TH STAGE ($K=0, \dots, \text{CEILING}(\text{LOG}_2(N))$), WE HAVE $\text{FLOOR}(N/2^{**K})$ STRINGS OF LENGTH 2^{**K} , AND ONE OF LENGTH $\text{MOD}(N, 2^{**K})$. IN PASSING TO THE NEXT STAGE, WE MERGE ADJACENT PAIRS OF THESE STRINGS INTO NEW STRINGS OF DOUBLED LENGTH. THE STRINGS ARE REPRESENTED BY ARG INDEXES MAINTAINED IN ONE OF TWO HALFWORD TABLES IN THE RESULT DATA BLOCK. INITIALLY, THE LOWER TABLE IS SET TO (ORIGIN,...,ORIGIN+N-1), THE IDENTITY PERMUTATION OF ARG ELEMENTS (N STRINGS OF LENGTH 1). EACH SET OF MERGES COPIES THE INDEXES FROM ONE TABLE TO THE OTHER; AFTER THE LAST MERGE, THE FINAL HALFWORD TABLE IS EXPANDED OUT INTO THE ENTIRE RESULT AS WORD VALUES. ARG ELEMENTS ARE INDEXED AS WORDS OR DOUBLEWORDS, RELATIVE TO THE INDEX ORIGIN; RESULT ELEMENTS ARE INDEXED AS HALFWORDS, RELATIVE TO ORIGIN-0.

THIS ROUTINE IS ENTERED WITH OP CODE IN 'OP' AND ARG TYPE IN 'T'. RESULT DATA BLOCK HAS ALREADY BEEN ALLOCATED.

NEW REGISTERS

```

        OPEN      IX
IA      EQU       1                      LEFT SOURCE ADR
JA      EQU       3                      RIGHT SOURCE ADR

```

KA	EQU	5	DESTINATION ADR
IX	EQU	4	LEFT SOURCE ARG INDEX
JX	EQU	7	RIGHT SOURCE ARG INDEX
IN	EQU	12	LEFT COUNT
JN	EQU	13	RIGHT COUNT
*			
*			
FASTGRAD	EQU	%	
	LI,A	OPBREAK	ALLOW BREAKS DURING EXECUTION
	STW,A	XSEGBRK	
	LCW,A	ORIGIN	SET RTADR =
	EXU	STBL,T	-2*ORIGIN + ...
	AW,A	RTARG	ADR OF 1ST ARG
	AW,A	T	ELEMENT (NOTE: WE ARE ADDING
	STW,A	RTADR	3 FOR INTG, 4 FOR FLOT).
	LW,A	RESULT	SET RSADR = ADR OF 1ST RESULT
	AI,A	3	ELEMENT.
	STW,A	RSADR	
	SLS,A	1	HALFWORD ADR OF LOWER TABLE
	STW,A	SORCBASE	INITIALLY, SOURCE IS LOWER TABLE
	AW,A	RSSIZE	HALFWORD ADR OF UPPER TABLE
	STW,A	DESTBASE	INITIALLY, DEST IS UPPER TABLE
	STW,A	SORCEND	SOURCE END = END OF LOWER TABLE
	LW,R	RSSIZE	INITIALIZE 1ST SOURCE TABLE
	AW,R	ORIGIN	TO (ORG,...,ORG+N-1).
	CI,R	X'8000'	SINCE THESE ARE HALFWORDS AND
	BG	ERLENGTH	'LH' DOES SIGN EXTENSION, WE
*			MUST NOT ALLOW ANY VALUE TO
*			BE >=2**15.
16Z1	LW,N	RSSIZE	
	STH,R	*RSADR,N	
	AI,R	-1	
	BDR,N	16Z1	
	STH,R	*RSADR	
	LI,A	1	INIT STRING SIZE TO 1
	STW,A	STRSIZE	
*			
NXTPASS	LW,IA	SORCBASE	INIT LEFT SOURCE TO 1ST STRING
	LW,JA	SORCBASE	
	AW,JA	STRSIZE	INIT RIGHT SOURCE TO 2ND STRING
	LW,KA	DESTBASE	INIT DESTINATION ADR TO OTHER TABLE
NXTMERGE	LW,JN	SORCEND	COMPUTE LENGTH OF RIGHT STRING
	SW,JN	JA	
	BGZ	16Z7	JUMP IF NOT EMPTY
	LW,IN	SORCEND	RIGHT STRING EMPTY: COMPUTE
	SW,IN	IA	LENGTH OF LEFT STRING
	BGZ	FINLEFT	JUMP IF NOT EMPTY
	LW,A	SORCBASE	BOTH EMPTY: THIS PASS IS DONE
	XW,A	DESTBASE	SWAP ADDRESSES OF SOURCE AND
	STW,A	SORCBASE	DESTINATION TABLES.
	AW,A	RSSIZE	UPDATE 'END OF SOURCE' ADR
	STW,A	SORCEND	
	LW,R	STRSIZE	DOUBLE STRING SIZE
	SLS,R	1	
	STW,R	STRSIZE	
	CW,R	RSSIZE	IF NEW STRING SIZE >= N,
	BL	NXTPASS	WE'RE DONE.
	CW,A	DESTBASE	DONE: WHICH TABLE IS RESULT IN ?
	BG	16Z5	UPPER
	LW,N	RSSIZE	LOWER: EXPAND RESULT UPWARDS,
	AI,N	-1	STARTING AT THE LAST ELEMENT.
16Z4	LH,R	*RSADR,N	
	STW,R	*RSADR,N	
	BDR,N	16Z4	
	LH,R	*RSADR	
	STW,R	*RSADR	
	B	*RETURN	EXIT
16Z5	LW,A	RSSIZE	UPPER: EXPAND RESULT DOWNWARDS,
	AWM,A	RSADR	STARTING WITH 1ST ELEMENT.
	LCW,N	RSSIZE	
16Z6	LH,R	*RSADR,N	
	STW,R	*RSADR,N	
	BIR,N	16Z6	
	B	*RETURN	EXIT
*			
16Z7	LW,IN	STRSIZE	RIGHT STRING NONEMPTY, SET
*			LEFT STRING COUNT = 2**K.
	CW,JN	STRSIZE	SET RIGHT STRING COUNT


```

        BLE      16Z8      = MIN(WORDS LEFT, 2**K).
        LW, JN    STRSIZE
16Z8      LH, JX    0, JA    GET 1ST INDEX FROM RIGHT STRING
        AI, JA    1
NEWLEFT   LH, IX    0, IA    GET NEW INDEX FROM LEFT STRING
        AI, IA    1
        EXU      LTBL, T  (LX, BX *RTADR, IX)  GET CORRESPONDING ARG VALUE
COMPARE    EXU      CTBL, T  (CX, BX *RTADR, JX)  COMPARE W/RIGHT STRING VAL
        EXU      BTBL, OP (BG/BL RIGHT)  JUMP IF TO USE RIGHT STRING
LEFT       STH, IX    0, KA    PLACE LEFT STRING INDEX
        AI, KA    1
        BDR, IN  NEWLEFT   IF MORE LEFT VALUES, USE NEXT ONE
        B        FINRIGHT1 LEFT STRING EMPTIED, COPY RIGHT
FINRIGHT   LH, JX    0, JA    GET NEXT RIGHT STRING INDEX
        AI, JA    1
FINRIGHT1  STH, JX    0, KA    COPY TO DESTINATION STRING
        AI, KA    1
        BDR, JN  FINRIGHT   CONTINUE UNTIL THEY'RE ALL GONE
        B        16Z9
RIGHT      STH, JX    0, KA    PLACE RIGHT STRING INDEX
        AI, KA    1
        BDR, JN  NEWRIGHT   IF MORE RIGHT VALUES, USE NEXT ONE
        B        FINLEFT1  RIGHT STRING EMPTIED, COPY LEFT
NEWRIGHT   LH, JX    0, JA    GET NEW RIGHT STRING INDEX
        AI, JA    1
        B        COMPARE    COMPARE LEFT/RIGHT STRING ELMTS
FINLEFT    LH, IX    0, IA    GET NEXT LEFT STRING INDEX
        AI, IA    1
FINLEFT1   STH, IX    0, KA    COPY TO DESTINATION STRING
        AI, KA    1
        BDR, IN  FINLEFT    CONTINUE UNTIL THEY'RE ALL GONE
16Z9      AW, IA    STRSIZE  END OF MERGE; BUMP BOTH STRING PNTRS
        AW, JA    STRSIZE  OVER ONE STRING.
        B        NXTMERGE

```

```

*
*
SORCBASE  TEMP      HALFWORD ADR OF 1ST SOURCE STRING
DESTBASE  TEMP      HALFWORD ADR OF 1ST DEST STRING
SORCEND   TEMP      HALFWORD ADR OF SOURCE TBL END
STRSIZE   TEMP      STRING SIZE
        STW, A    RSADR

```

```

*
*
STBL      TABLE    INTG      SHIFT TABLE
        NOP
        SLS, A    1          INTG
                                FLOT
*
LTBL      TABLE    INTG      LOAD ARG ELMT TABLE
        LW, BI    *RTADR, IX  INTG
        LD, BF    *RTADR, IX  FLOT
*
CTBL      TABLE    INTG      COMPARE INST TABLE
        CW, BI    *RTADR, JX  INTG
        CD, BF    *RTADR, JX  FLOT
*
BTBL      TABLE    MOPGRDUP  BRANCH TABLE
        BG        RIGHT      GRADE UP
        BL        RIGHT      GRADE DOWN
        PAGE

```

```

*
*
*   D Y A D I C   M I X E D   O P   R O U T I N E S
*
*

```

```

DRESHAPE  EQU      %        DYADIC RESHAPE
        LI, A      0        SET UP LEFT RANK, SIZE,
        BAL, LX    SETUPARG  AND TYPE CELLS.
        LI, R      1
        STW, R     RSSIZE    INIT RESULT SIZE TO 1
        CW, R      LFRANK    MAKE SURE LEFT ARG IS SCALAR/VECTOR
        BL        ERRANK
        LW, R      LFSIZE    IF IT'S NULL, SKIP THE 1ST
        BEZ        3Z1       XSEG GEN/RUN.
        CI, R      MAXDIMEN  IF NOT NULL, MAKE SURE IT
        BG        ERLNGTH    ISN'T TOO BIG.

```

```

*
*   GEN 1ST XSEG WHICH, WHEN RUN, COPIES (AND CONVERTS
*   TO INTG, IF NECESSARY) LEFT ARG VALUES (RESULT DIMENS)

```

```

*          TO BUFFER (FRONT END OF XSEG AREA) AND ACCUMULATES
*          RESULT SIZE IN 'RSSIZE'.
*
LI,X      -3
BAL,LX    SETADR          SET LEFT ADR CELL
LI,XL     DBUFEND         START GENNING XSEG ABOVE DIMEN BUF
GEN,0,1   RSH Pins1      GEN:   LCW,N   LFSIZE
LI,T      INTG           SET RSType TO INTG SO THAT GENLOAD
STW,T     RSType         WILL GEN CONVERSION TO INTG.
LI,A      0
BAL,L1    GENLOAD        GEN LOAD/CONVERT OF LEFT ARG VAL
GEN,0,6   RSH Pins2      GEN CODE TO STORE DIMEN, MULTIPLY
*                                     IT INTO RSSIZE, LOOP, AND EXIT.
*
BAL,L1    DBUFEND        EXECUTE 1ST XSEG
*
*          ALLOCATE AND SET UP RESULT; SET UP RIGHT ARG.
*
3Z1      LI,A      1          SET UP RIGHT SIZE, RANK,
BAL,LX    SETUPARG        AND TYPE CELLS.
LW,T      RTTYPE
ISEQFIX,T          USE TYPE INTG INSTEAD OF ISEQ
STW,T     RSType         RESULT TYPE = RIGHT TYPE
LW,R      LFSIZE
STW,R     RSRANK         RESULT RANK = LEFT SIZE
LW,S      RSSIZE         RESULT SIZE WAS SET BY 1ST XSEG
BAL,L1    ALOC RS        ALLOCATE RESULT DB; SET 'RESULT'
BAL,LX    MBUF DIMS      MOVE DIMENS FROM BUFFER TO RS
*
*          NOW BUILD AND EXECUTE THE SECOND XSEG, WHICH MOVES
*          THE DATA FROM RIGHT ARG DB TO RESULT DB.
*
BAL,LX    GXSEGINI        INIT XSEG; EXIT DRIVER IF RSLT NULL.
LW,R      RTSIZE         RESULT NOT NULL; MAKE SURE THERE
BEZ       ERLNGTH        IS SOME DATA TO FILL IT.
AI,R      -1             GEN ALTERNATE XSEG IF RIGHT          U09-0022
BEZ       3Z2            ARG IS 1-ELEMENT.                  U09-0023
LI,X      -1             MULTI-ELMT: SET UP RSADR           U09-0024
BAL,LX    SETADR         FOR SEQUENTIAL STORE.             U09-0025
GEN,0,1   RSH Pins3      GEN:   LI,K   0
STW,XL    LOOPLOC        LOOP LOCATION IS HERE
LI,A      1             COMPUTE ADDRESS OF RIGHT ARG
BAL,LX    XSETUP         FOR INDEXED (0-ORIGIN) ACCESS.
BAL,L1    GENLOAD        GEN INDEXED LOAD OF RIGHT ARG
*                                     GEN MODULAR (WRAP-AROUND) INCREMENT
*                                     OF RIGHT ARG INDEX.
GEN,0,4   RSH Pins4
AWM,XL    -2,XL          FILL IN ADR
B         GXSTEXEC        GEN STORE/LOOP INST; EXEC XSEG.
3Z2      LI,T      LOGL    IF RT ARG IS LOGL SCALAR, CHANGE    U09-0027
CW,T      RTTYPE        TO 'WORD LOGL' SO THAT WE'LL          U09-0028
BNE       3Z3            COPY 32 ELEMENTS AT A TIME.          U09-0029
MTW,WORD LOGL-LOGL      RTTYPE          U09-0030
3Z3      LI,X      -2          SET RTADR AND RSADR FOR          U09-0031
BAL,L2    SETADRS1        SEQUENTIAL ADDRESSING.             U09-0032
LI,A      1             U09-0033
BAL,L1    GENLOAD        GEN LOAD OF RT ARG                  U09-0034
STW,XL    LOOPLOC        LOOP RETURNS TO STORE INST          U09-0035
B         GXSTEXEC        GEN STORE/LOOP CODE; EXECUTE XSEG   U09-0036
*
*          XSEG 1 CODE:
*
RSH Pins1 LCW,N   LFSIZE          INIT INDEX TO MOVE DIMENS
* LOAD/CONVERT LFADR(N)
RSH Pins2 STW,AI  DBUFEND,N        COPY DIMEN TO BUFFER
MW,AI     RSSIZE          MULTIPLY DIMEN INTO RSSIZE
BCS,5     ERDOMAIN        ERR IF SIZE OVFL, OR DIMEN<0
STW,AI     RSSIZE
BIR,N     DBUFEND+1        LOOP
B          *L1             EXIT
*
*          XSEG 2 CODE FOR MULTI-ELEMENT RIGHT ARG          U09-0038
*
* LCW,N     RSSIZE          INIT RESULT INDEX
RSH Pins3 LI,K   0          INIT RIGHT ARG INDEX
* LOOPLOC  LOAD   RTADR(K)
RSH Pins4 AI,K   1          BUMP RIGHT ARG INDEX
CW,K      RTSIZE          IT GOES THRU 0,1,...,RTSIZE-1,
BL        0              (%+2) THEN RESETS TO 0.
LI,K      0

```

```

*      STORE      RSADR(N)
*      BIR,N      *LOOPLOC
*      B          *RETURN
*
*                      XSEG 2 CODE FOR 1-ELEMENT RIGHT ARG
*
*      LCW,N      RSSIZE      INIT RESULT ADR
*      LOAD      RTADR      LOAD SCALAR RT ARG VALUE
*      *LOOPLOC STORE      RSADR(N)      STORE IN RESULT
*      BIR,N      LOOPLOC      LOOP
*      B          *RETURN      EXIT
*      PAGE
*
*
DDEAL      EQU      %
           BAL,LZ    INTSCALR      GET INTG SCALAR VALUE OF RT ARG, Y
           STW,AI    RTTEMP      SAVE IT
           EXCHANGE  ARGS
           BAL,LZ    INTSCALR      GET INTG SCALAR VALUE OF LF ARG, X
           STW,AI    LFTEMP      SAVE IT
           AI,AI     0            MAKE SURE 0<=X<=Y
           BLZ      ERDOMAIN
           CW,AI     RTTEMP
           BG        ERDOMAIN
*
*      PICK THE ALGORITHM TO USE
*
           LI,OP     2
           LW,R      TOPOSTAK      COMPUTE AVAILABLE MEMORY =
           SW,R      DYNBOUND      SPACE BETWEEN STACK & DYNAMIC
           AW,R      FREETOTL      MEM + FREE DYNAMIC WORDS
           AI,R      -8            - CREDIBILITY GAP.
           CW,R      RTTEMP      IF SUFFICIENT MEMORY FOR ENTIRE
           BGE       7Z0          UNIVERSE SET, USE ALGORITHM 'B'.
           LI,OP     3            OTHERWISE, USE ALGORITHM 'C'
           EXU       DEALSZTB,OP   SIZE OF RESULT = X (ALG 'C'),
           STW,S      RSSIZE      OR Y (ALG 'B').
           BAL,L2    INTVECRS     ALLOCATE INTG VECTOR FOR RESULT;
*                                     IF SIZE=0, EXIT FROM OP DRIVER.
           AI,A      2            POINT TO 1ST DATA WORD -1
           STW,A      RSADR
           LI,A      OPBREAK      SET XSEG BREAK FLAG TO SAY,
           STW,A      XSEGBRK     'BREAK IS NOW OK'.
           B          DEALALG,OP   DO THE SELECTED ALGORITHM
*
*      ALGORITHM 'C' - MEMORY USED = SUBSET SIZE, NO BOUND ON NUMBER
*                      OF RANDOM NUMBER SELECTIONS REQUIRED.
*
DEALC      EQU      %
           LI,N      1            INIT COUNT TO 1
           STW,N      COUNT
           LW,AI     RTTEMP      GET 1ST RANDOM VALUE FROM SET:
           BAL,LX    IROLL      ORIGIN,...,ORIGIN+Y-1.
           B          7Z4        GO INTO LOOP
           LW,N      COUNT      REJECTED: SET N TO SEARCH AGAIN
           LW,AI     RTTEMP      GET A NEW RANDOM VALUE IN RANGE
           BAL,LX    IROLL      ORIGIN <= (AI) <= ORIGIN+Y-1.
           CW,AI     *RSADR,N    SEARCH FOR IT AMONG ALL THE RESULT
           BE        7Z1        VALUES WE HAVE CURRENTLY; IF WE
           BDR,N     7Z3        ALREADY HAVE IT, REJECT IT.
           MTW,1     COUNT      ACCEPTED: BUMP THE RESULT COUNT
           LW,N      COUNT
           LW,AI     *RSADR,N    STORE NEW VALUE INTO RESULT
           CW,N      RSSIZE      SEE IF WE'VE GOT THEM ALL, YET
           BL        7Z2        NO, GET ANOTHER
           B          *RETURN    YES, RETURN
*
*      ALGORITHM 'B' - MEMORY USED = UNIVERSE SIZE, NUMBER OF RANDOM
*                      HITS = SUBSET SIZE.
*
DEALB      EQU      %
           LW,R      RTTEMP      INITIALIZE RESULT TO
           SW,R      ORGADJ      ORIGIN,ORIGIN+1,...,
           LW,N      RTTEMP      ...,ORIGIN+(UNIVERSE SIZE)-1.
           STW,R     *RSADR,N
           AI,R      -1
           BDR,N     7Z5
           LI,K      1

```

U09-0040
U09-0041
U09-0042
U09-0043
U09-0044
U09-0045
U09-0046
U09-0047

7Z6	LW, N	LFTEMP	SET TO PICK (SUBSET SIZE) NUMBERS
	LW, AI	RANDOM	UPDATE RANDOM SEED
	MI, AI	65539	
	AND, AI	=X'7FFFFFFF'	
	STW, AI	RANDOM	
	MW, AI-1	RTTEMP	PICK RANDOM NR FROM 0 TO Y-1
	SLD, AI-1	-31	
	AW, AI	K	PUT IT IN RANGE K TO K+Y-1
	LW, R	*RSADR, AI	EXCHANGE RANDOMLY SELECTED ELMT
	XW, R	*RSADR, K	WITH K'TH ELMT.
	STW, R	*RSADR, AI	
	AI, K	1	INCR K
	MTW, -1	RTTEMP	DECREASE Y
	BDR, N	7Z6	SELECT NEXT NUMBER
	STW, N	XSEGBRK	DON'T ALLOW BREAKS DURING
*			'GIVE BACK' OPERATION.
	LW, A	RESULT	GET RESULT DB POINTER
	LW, S	LFTEMP	CHANGE RESULT SIZE TO SUBSET
	XW, S	2, A	SIZE, AND COMPUTE NUMBER OF
	SW, S	2, A	WORDS TO GIVE BACK.
	BAL, LX7	GIVEBACK	GIVE THEM BACK
	B	*RETURN	EXIT
*			
COUNT	TEMP		RESULT COUNT TEMP
*			
*			
DEALSZTB	TABLE	2	DEAL: RESULT SIZE EXU TABLE
	LW, S	RTTEMP	ALG B: UNIVERSE SIZE
	LW, S	LFTEMP	ALG C: SUBSET SIZE
*			
DEALALG	TABLE	2	DEAL ALGORITHM BRANCH TABLE
	B	DEALB	ALG B
	B	DEALC	ALG C
	PAGE		
*			
*			
DINDEXOF	EQU	%	DYADIC INDEX OF
	LI, X	1	
	CB, X	*LFARG, X	MAKE SURE THAT LEFT ARG
	BNE	ERRANK	IS A VECTOR;
	LI, OP	DOPMEMBR+1	MAKE IT LOOK MORE LIKE ...
	B	MIXOP1	
DMEMBER	EQU	%	DYADIC MEMBERSHIP
	EXCHANGE	ARGS	SWAP ARGS
MIXOP1	EQU	%	
	LI, A	0	
	BAL, LX	SETUPARG	SET LEFT RANK/SIZE/TYPE
	LI, A	1	
	BAL, LX	SETUPARG	SET RIGHT RANK/SIZE/TYPE
	LW, T	OPTBL1, OP	RESULT TYPE = LOGL (MEMBERSHIP)
	STW, T	RSTYPE	OR INTG (INDEX OF)
	BAL, L1	RSLIKRT1	ALOC RESULT LIKE RIGHT ARG
	BAL, LX	MRTDIMS	COPY RTARG DIMENSIONS TO RESULT
	LI, X	-2	RIGHT ARG AND RESULT ARE TO BE
	BAL, L2	SETADRS1	ACCESSED NORMALLY (SEQ).
	LI, A	0	LEFT ARG IS TO BE ACCESSED
	BAL, LX	XSETUP	INDICIALLY.
	BAL, LX	TYCOMPAT	IF BOTH ARGS ARE CHAR, OR BOTH
	B	5Z4	NUMERIC, AND IF THE LEFT ARG
*			IS NON-NULL, THEN WE MAY GO
*			AHEAD ON IT.
*			IF THE ARGS ARE OF CONFLICTING
*			KINDS, OR THE LEFT ARG IS NULL,
*			THE RESULT WILL CONSIST ENTIRELY
	LW, S	LFSIZE	OF ZEROS (MEMBERSHIP) OR
	BNEZ	5Z7	LFSIZE+ORIGIN (INDEX OF).
5Z4	LW, X	RSSIZE	CREATE SPECIAL RESULT (IF IT'S TO
	BEZ	*RETURN	BE NON-NULL):
	LW, A	RESULT	
	AW, A	RSRANK	CREATE PNTR TO 1ST DATA WORD -1;
	AI, A	1	
	CI, OP	DOPMEMBR	
	BNE	5Z5	
	AI, X	31	FOR MEMBERSHIP, CHANGE SIZE
	SLS, X	-5	FROM BITS TO WORDS, AND
	LI, AI	0	SET RESULT DATA = 0'S;
	B	5Z6	
5Z5	LW, AI	LFSIZE	FOR INDEX-OF, SET RESULT DATA

Address	Instruction	Comment	Label
5Z6	AW, AI STW, AI BDR, X B	ORIGIN *A, X 5Z6 *RETURN	= SIZE+ORIGIN; STORE RESULT DATA; RETURN.
5Z7	BAL, LX GEN, 0, 1 LW, T CI, T BNE LI, T STW, T CI, OP BNE LI, A CW, T BE BAL, L2 STW, XL LI, A BAL, L1 LW, R B	GXSEGINI KLUPINIT RSTYPE LOGL 5Z9 INTG RSTYPE DOPMEMBR LGINDXOF 1 LFTYPE 5Z10 GENLOADT LOOP2LOC 0 GENLOAD RTADR 5Z11	GEN XSEG INIT; EXIT IF RESULT NULL GEN K-LOOP (LOOP2) INIT INST WE MUST USE 'INTG' IF BOTH ARGS ARE LOGL ('CAUSE LOGL ARGS AREN'T DIRECTLY ADDRESSABLE). IF IT'S 'INDEX OF' WITH BOTH ARGS LOGL, HANDLE SPECIALLY. SET UP TO GEN LOAD OF RIGHT ARG MUST THE LEFT ARG BE CONVERTED ? YES: GEN LOAD/CONVERT RIGHT ARG TO TEMP OUTSIDE LOOP2; GEN LOAD/CONVERT LEFT ARG TO REG INSIDE LOOP2; COMPARE ADR = RIGHT TEMP.
5Z10	BAL, L1 STW, XL LW, R	GENLOADX LOOP2LOC LFADR	NO: GEN LOAD/CONVERT OF RIGHT ARG TO REG OUTSIDE OF LOOP2; COMPARE ADR = LEFT ARG.
5Z11	LW, T CI, T BNE AW, R GEN, 1, 1 LI, R AW, R LW, X GEN, 1, 4 CI, OP BNE GEN, 0, 3 LW, R AW, R GEN, 1, 0 AWM, XL B	RSTYPE FLOT 5Z8 LODBINST+FLOT R, BALCOMP BF COMPIINST, T XL R, KLUPCONT DOPMEMBR 5Z12 STCCSEQ RSADR STORINST+LOGL R -3, XL 5Z13	IF FLOATING RESULT, LOAD BOTH ARGS TO REGS, AND APPLY FUZZ TO ARGS. ADR 2ND REG FOR COMPARISON GEN COMPARE INST, AND INDEX CONTROL FOR LOOP2. FOR MEMBERSHIP, GENERATE LOGICAL STORE OF 0 OR 1.
5Z12	GEN, 0, 2 LW, R AWM, R AWM, XL LW, R AWM, R GEN, 0, 2 B	INDOFSEQ RSADR -1, XL 1, X LOOP2LOC 4, X NLUPCONT EXECUTE	FOR INDEX-OF, GEN STORE OF ORIGIN+INDEX VALUE. FILL IN BRANCH ADDRESSES GEN N-LOOP (LOOP1) CONTROL/RETURN GO EXECUTE XSEG AND RETURN
5Z13	AWM, R GEN, 0, 2 B	DELTA DELTA RTADR -2, XL *L1	DELTA = 1, 2, OR 3 GEN: SW, N DELTA LB, AI RTARG(N) AW, N DELTA RETURN
GENLOADX	EQU CI, T BNE LW, R SW, R BEZ STW, R GEN, 0, 3 LW, R AWM, R B	% CHAR GENLOAD RTSIZE RSSIZE GENLOAD DELTA DELTA RTADR -2, XL *L1	SPECIAL GEN LOAD SPECIAL ONLY FOR CHARACTER ARGS WHEN SIZE IS NOT A MULTIPLE OF FOUR. DELTA = 1, 2, OR 3 GEN: SW, N DELTA LB, AI RTARG(N) AW, N DELTA RETURN
DELCODE	SW, N LB, AI AW, N	DELTA 0 RTARG(N) DELTA	FIX INDEX TO START WITH 1ST CHAR LOAD SELECTED CHAR RESTORE INDEX
DELTA	TEMP		DELTA (1,2,3)
OPTBL1	TABLE PZE PZE	DOPMEMBR LOGL INTG	RESULT TYPE TBL FOR OPS ... MEMBERSHIP: LOGL RESULT INDEX OF: INTG RESULT

```

*
*      LCW,N      RSSIZE      0  INIT RESULT COUNT
KLUPINIT  LI,K      0      1  K-LOOP INITIALIZATION INST
*      LOAD/CNV  RTARG(N)      GET RIGHT ARG ELMT
*      <STORE    RTTEMP>      IF LEFT ARG MUST BE CONVERTED,
*LOOP2    <LOAD/CNV LFARG(K)>      SAVE RIGHT ARG & LOAD LFARG.
*      COMPARE  RTTEMP/LFARG(K)  COMPARE LF/RT ARG ELEMENTS
KLUPCONT  NBE      -2      (FOUND)  K-LOOP CONTROL SEQUENCE
        AI,K      1
        CW,K      LFSIZE
        BL      0      (LOOP2)

*
*      FOR 'INDEX OF' ...

*FOUND    EQU      %
INDOFSEQ  AW,K      ORIGIN      INDEX STORE SEQUENCE
        STW,K      0  RESULT(N)

*
*      FOR 'MEMBERSHIP' ...

*      LI,AI      0      NOT FOUND, RS ELMT =0
*      B          %+2
*FOUND    LI,AI      -1      FOUND: RS ELMT =1
*      BAL,LX      STLOGLRS    STORE LOGICAL RS ELEMENT
*
*      ---
*
NLUPCONT  BIR,N      XSEGBASE+1    N-LOOP CONTROL SEQUENCE
        B          *RETURN

*
LOOP2LOC  TEMP      LOOP 2 (INNER LOOP) BRANCH LOC
        PAGE

*
*
*      SPECIAL CASE HANDLING FOR 'INDEX OF' WITH BOTH
*      ARGS LOGICAL.
*
LGINDEXOF EQU      %
        LW,A      LFARG      SET UP LOGL WORD PNTR
        AW,A      LFRANK
        LW,R      ORIGIN      = INDEX OF 1ST BIT
        LI,X      -1
        LW,AI     2,A      IF 1ST BIT IS 0 (1),
        BLZ      18Z1      SET X TO SKIP WORDS
        LI,X      0      CONTAINING ALL 0'S (1'S).
18Z1      STW,R      INDEXOF01,X  STORE INDEX OF 1ST 0 OR 1
        STW,X      XTEMP      INITIALIZE XTEMP
        LCW,R      LFSIZE      INIT BIT COUNT
18Z2      CW,X      2,A      IS ENTIRE WORD LIKE 1ST BIT?
        BNE      18Z3      NO - GO LOCATE DIFFERING BIT
        AI,A      1      YES - SKIP OVER THIS WORD
        AI,R      32      COUNT 32 BITS AT A TIME
        BLZ      18Z2      ..UNTIL DONE.
        B          18Z5      DONE: OTHER BIT ABSENT FROM DATA
18Z3      STW,X      XTEMP      FOUND WORD CONTAINING OTHER BIT:
        EOR,X      2,A      SAVE ITS VALUE, SEARCH FOR IT (EOR
18Z4      AI,R      1      MAKES IT A 1-BIT WE'RE SEEKING).
        SLS,X      1
        BEV      18Z4
        AI,R      -1
        BLZ      18Z6
18Z5      LI,R      0
18Z6      AW,R      LFSIZE      SET R = INDEX (0-ORIGIN)
        AW,R      ORIGIN      R = INDEX OF DIFFERING BIT
        LCW,X      XTEMP      REMEMBER WHICH BIT IT WAS
        STW,R      INDEXOF01-1,X  STORE INDEX

*
        AI,XL     -1      DISCARD LAST INST GEN'D BY 'INDEXOF'
        LD,R      CODE9
        AW,R      RTADR
        GEN,2,0    R      GEN LOGL LOAD, LOAD OF 0/1 BIT INDEX
        B          GXSTEXEC    GEN STORE/LOOP CONTROL; EXECUTE XSEG.

*
*
*      XSEG CODE:
*
*
*      LCW,N      RSSIZE      0  INIT COUNT = SIZE

```

```

CODE9      BOUND      8
BAL,LX     0 (LDLOGLRT) 1 LOAD BIT FROM RIGHT ARG
LW,AI      INDXOF01,AI  2 GET INDEX OF THAT BIT
*          STW,AI      RESULT(N) 3 STORE IN RESULT
*          BIR,N      (CODE9)    4 LOOP
*          B          *RETURN    5 EXIT
*
XTEMP      EQU        DELTA      TEMP IN WHICH TO SAVE X
INDXOF01   EQU        LFTEMP+1   INDEX OF 0/1 BIT IN ARG DATA
*
*
DENCODE    EQU        %          DYADIC ENCODE
BAL,LX     DTYPEIF    CHECK ARG TYPES; SET RESULT
NOP        TYPE = INTG/FLOT.
BAL,L2     SETOUTER   SET UP RESULT FOR ENCODE
LI,S       1          COMPUTE 2 LOOP COUNTS:
CW,S       LFSIZE     INNER COUNT = LFARG'S FIRST
BE         11Z1       DIMEN (=1, IF SCALAR);
LW,X       LFARG
LW,S       2,X
11Z1       STW,S      INERCNT
LW,S       LFSIZE
DW,S       INERCNT
STW,S      MIDLCNT
MW,S       RTSIZE
STW,S      RSSTEP
BAL,LX     GXSEGINI   RESULT STEP FOR INNER LOOP:
MTW,-1     XSEGBASE   ALL BUT 1ST RESULT DIMENS.
GEN,0,4    CODE10     INIT XSEG CODE; EXIT IF RS NULL
LI,A       1          CHANGE 'RSSIZE' TO 'RTSIZE'
BAL,L2     GENLOADT   GEN LOOP INIT CODE,
STW,XL     LOOPLOC    LOAD/CONVERT/STORE OF RTARG,
SW,R       =X'03000000' MIDDLE LOOP LOC,
GEN,1,5    R,CODE11   LOAD RTARG VAL,
LW,R       STORINST,T INNER LOOP INIT,
AI,R       BTEMP      INNER LOOP CODE,
GEN,1,0    R
LI,A       0
BAL,L2     GENLOADT
LW,A       XL
GENX       CODETBL5,T
LW,R       RSADR
AWM,R      -6,XL
AWM,A      -5,XL
GEN,0,1    CODE14
LW,R       LOOPLOC
AWM,R      -1,XL
B          EXECUTE    AND FINAL LOOP CONTROL CODE.
*
*
CODETBL5   TABLE    INTG
CODE,8     CODE13I    INTG
CODE,8     CODE13F    FLOT
*
*
*          XSEG CODE:
*
CODE10     LCW,N      RTSIZE    0 INIT RTARG INDEX/OUTER LOOP COUNT
LW,K       RSSIZE     1 INIT RESULT INDEX
LW,K1      LFSIZE     2 INIT LFARG INDEX (OUTR LOOP HERE)
LW,N2      MIDLCNT    3 INIT MIDDLE LOOP COUNT
STW,K      OUTRSAVE   4 REMEMBER RS INDEX VALUE
*          LOAD/CNV RTARG(N) 5 GET RTARG ELMT
*          STORE    RTTEMP   SAVE IT
*LOOPLOC   LOAD      RTTEMP   RE-INIT RTARG VAL (MIDL LOOP HERE)
CODE11     LW,N3      INERCNT  INIT INNER LOOP COUNT
STW,K      MIDLSAVE   REMEMBER RS INDX FOR MIDL LOOP
STW,K1     K1TEMP     REMEMBER LF INDX FOR MIDL LOOP
SW,K       RSSTEP     BUMP RS INDEX (INER LOOP HERE)
SW,K1      MIDLCNT    BUMP LF INDEX
*          STORE    BTEMP    SAVE PREVIOUS QUOTIENT
*          LOAD/CNV LFARG(K1) GET NEXT WEIGHT FROM LFARG
*          STORE    LFTEMP    SAVE IT
*
*          FOR FLOT RESULT -
*

```

```

CODE13F LD,BF      BTEMP
        BAL,LX     FFRESIDU      COMPUTE NEW REMAINDER
        STD,AF      0             STORE IT AS NEXT RESULT ELMT
        BDR,N3      5 (%+2)      COUNT INNER LOOP
        B           CODE15
        LCD,AF      AF
        FAL,AF      BTEMP        MAKE FOLLOWING DIVISION EXACT
        FDL,AF      LFTEMP       COMPUTE NEW QUOTIENT

*
*
*       FOR INTG RESULT -
*
CODE13I LW,BI      BTEMP          (SEE ABOVE)
        BAL,LX     IIRESIDU      *
        STW,AI      0             RESULT(K)      *
        BDR,N3      5 (%+2)      COUNT INNER LOOP
        B           CODE15
        LCW,AI      AI           *
        AW,AI       BTEMP        *
        DW,AI       LFTEMP       *

*
*       - - -
*
CODE14  B          4 +LOOPLOC     CONTINUE INNER LOOP
CODE15  LW,K1      K1TEMP         * RESTORE MIDL LOOP LF INDX
        AI,K1      1             * BUMP IT
        LW,K       MIDLSAVE      * RESTORE MIDL LOOP RS INDX
        AW,K       RTSIZE        * BUMP IT
        BDR,N2     *LOOPLOC      * COUNT MIDDLE LOOP
        LW,K       OUTRSAVE      * RESTORE AND
        AI,K       1             * BUMP RESULT INDEX.
        BIR,N      XSEGBASE+2    * COUNT OUTER LOOP
        B          *RETURN       * EXIT

*
*
BTEMP   DTEMP
K1TEMP  TEMP
RSSTEP  TEMP
PAGE

*
*
DDECODE EQU        %             DYADIC DECODE
        BAL,LX     DTYPEIF      CHECK ARG TYPES; SET RESULT
        NOP                                     TYPE = INTG/FLOT.
        BAL,L2     SETINNER      SET UP RESULT/PARAMS FOR DECODE
        BAL,LX     GXSEGINI      GEN XSEG INIT; EXIT IF RS NULL
        GEN,0,8    CODE1         GEN OUTER/MIDDLE LOOP INIT CODE,
        LW,T       RSTYPE
        GENX       CODETBL1,T    INNER LOOP INIT,
        LI,A       1
        BAL,L1     GENLOAD      RTARG LOAD/CONVERT,
        LI,S       1
        CW,S       RTCOMDIM
        BE         12Z1
        GEN,0,1    CODE3         BUMP OF RTARG INDEX,
12Z1    LW,T       RSTYPE
        GENX       CODETBL2,T    MULTIPLY/ADD,
        LI,A       0
        BAL,L1     GENLOAD      LFARG LOAD/CONVERT,
        LW,T       RSTYPE
        GENX       CODETBL3,T    MULTIPLY TO WEIGHT,
        LI,S       1
        CW,S       LFCDIM
        BE         12Z2
        GEN,0,1    CODE6         BUMP OF LFARG INDEX,
12Z2    GENX       CODETBL4,T    RESULT STORE, AND INER LOOP
        LW,R       RSADR        CONTROL CODE.
        AWM,R      -3,XL
        B          EXECUTE      EXECUTE XSEG

*
*
CODETBL1 TABLE    INTG
        CODE,4     CODE2I      INTG
        CODE,4     CODE2F      FLOT

*
CODETBL2 TABLE    INTG
        CODE,4     CODE4I      INTG
        CODE,3     CODE4F      FLOT

*

```


CODETBL3	TABLE	INTG	
	CODE, 3	CODE5I	INTG
	CODE, 2	CODE5F	FLOT
*			
CODETBL4	TABLE	INTG	
	CODE, 5	CODE7I	INTG
	CODE, 5	CODE7F	FLOT
*			
*			
* XSEG CODE:			
*			
CODE1	LCW, N	RSSIZE	0 INIT STORE INDX / OUTER LOOP COUNT
	LW, K1	LFCOMDIM	1 INIT LFARG INDX
	AI, K1	-1	2
	LW, K	RTSIZE	3 INIT RTARG INDX
	SW, K	MIDLCNT	4 OFFSET RTARG INDX(OUTER LOOP HERE)
	STW, K1	OUTRSAVE	5 SAVE LFARG INDX
	LW, N2	MIDLCNT	6 INIT MIDDLE LOOP COUNT
	STW, K	MIDLSAVE	7 SAVE RTARG INDX (MIDDLE LOOP HERE)
	LW, N3	INERCNT	8 INIT INNER LOOP COUNT
*			
* FOR FLOT RESULT -			
*			
CODE2F	LD, AF	FLOT1	9 INIT WEIGHT FACTOR = 1
	STD, AF	WEIGHT	10
	LD, AF	FLOT0	11 INIT SUM = 0
	STD, AF	SUM	12
*	LOAD/CNV	RTARG(K)	13 LOAD NEW RT VAL (INNER LOOP HERE)
CODE3	SW, K	MIDLCNT	(CONDITIONALLY GEN'D) BUMP RT INDX
CODE4F	FML, AF	WEIGHT	APPLY WEIGHT FACTOR
	FAL, AF	SUM	ACCUMULATE INTO SUM
	STD, AF	SUM	
*	LOAD/CNV	LFARG(K1)	LOAD NEW MULTIPLIER
CODE5F	FML, AF	WEIGHT	UPDATE WEIGHT
	STD, AF	WEIGHT	
CODE6	AI, K1	-1	(COND. GEN'D) BUMP LF INDX
CODE7F	BDR, N3	XSEGBASE+13	COUNT INNER LOOP
	LD, AF	SUM	STORE SUM AS NEXT RESULT ELMT
	STD, AF	0	RESULT(N)
	BIR, N	CODE8	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
*			
* INTG RESULT -			
*			
CODE2I	LI, AI	1	9 INIT WEIGHT FACTOR = 1
	STW, AI	WEIGHT	10
	LI, AI	0	11 INIT SUM = 0
	STW, AI	SUM	12
*	LOAD/CNV	RTARG(K)	13 LOAD NEW RT VAL (INNER LOOP HERE)
*	SW, K	MIDLCNT	(COND. GEN'D) BUMP RT INDEX
CODE4I	MW, AI	WEIGHT	APPLY WEIGHT FACTOR
	BOV	INTGOVFL	
	AWM, AI	SUM	ACCUMULATE INTO SUM
	BOV	INTGOVFL	
*	LOAD/CNV	LFARG(K1)	LOAD NEW MULTIPLIER
CODE5I	MW, AI	WEIGHT	UPDATE WEIGHT
	BOV	INTGOVFL	
	STW, AI	WEIGHT	
*	AI, K1	-1	(COND. GEN'D) BUMP LF INDEX
CODE7I	BDR, N3	XSEGBASE+13	COUNT INNER LOOP
	LW, AI	SUM	STORE SUM AS NEXT RESULT ELMT
	STW, AI	0	RESULT(N)
CODE16	BIR, N	CODE8	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
*			
* ---			
*			
CODE8	LW, K1	OUTRSAVE	* RESTORE LF INDEX
	LW, K	MIDLSAVE	* AND RT INDEX.
	AI, K	1	* BUMP RT INDEX
	BDR, N2	XSEGBASE+7	* COUNT MIDDLE LOOP
	AW, K1	LFCOMDIM	* BUMP LF INDEX
	B	XSEGBASE+4	* COUNT OUTER LOOP
*			
* WEIGHT EQU LFTEMP			
SUM EQU RTTEMP			
LFCOMDIM TEMP			

RTCOMDIM TEMP
PAGE

*
*
* C O M P O S I T E O P E X E C U T I O N D R I V E R S
*
*

* SPECIAL EFFECTS DEPARTMENT:
*

* EACH OF THE COMPOSITE OP ROUTINES (REDUCTION, SCAN, U09-0052
* INNER/OUTER PRODUCT) GENERATES ITS OWN BRAND OF U09-0053
* LOOP CONTROL, LOAD, AND STORE CODE; TO GENERATE
* OP EVALUATION CODE, THE APPROPRIATE DYADIC SCALAR
* OP ROUTINE(S) IS (ARE) INVOKED. COMMUNICATION BETWEEN
* THE SCALAR AND COMPOSITE OP ROUTINES IS ESTABLISHED
* VIA DETOURS ENCOUNTERED IN THE SUBROUTINES LISTED
* BELOW; AS EACH OF THESE SUBR'S IS CALLED, FINDING
* THE COMPOSITE OP TRIG 'COPTRIG' SET, IT DETOURS VIA
* THE CORRESPONDING LINK, INSTEAD OF PERFORMING ITS
* USUAL FUNCTION.
*

* SUBR: DETOUR LINK: FUNCTION PERFORMED:
*

* DTYPEIF CTYPELNK TYPE CHECKING/SETUP
* DSETUP CSETLNK CONFORMANCE CHECK, ALOC RS U09-0055
* GXSEGD L CLOADLNK GEN ARG LOADS/CONVERTS
* GXSTEXEC CSTORLNK GEN STORE; EXECUTE XSEG
*

* NOTE: 'GXSEGD L' INVOKES 'GXSEGINI' BEFORE DETOURING;
* THE OTHERS DETOUR IMMEDIATELY.
*

PAGE

* REDUCTION:
*

* OP/(K) ARG
*

REDUCE EQU % (MONADIC) REDUCTION
BAL, LX SETLINKS SET DETOUR LINKS; ENTER DYADIC
* PZE *MXRETURN SCALAR OP ROUTINE.
PZE REDTYPE RETURN ADR / COPTRIG
PZE REDSET CTYPELNK FOR REDUCTION
PZE REDLOAD CSETLNK FOR REDUCTION
PZE REDSTOR CLOADLNK FOR REDUCTION
CSTORLNK FOR REDUCTION
*

REDTYPE EQU % SUBSTITUTE FOR DTYPEIF
LB, T *RTARG DO JUST WHAT DTYPEIF WOULD DO,
CI, T CHAR
BNE DTYPEIF2 GIVEN IDENTICAL ARGS.
TESTEQ CLM, OP COMPAROP EXCEPT, ALLOW CHAR ARG FOR
BCR, 9 CHAREQ EQUAL AND NOT EQUAL OPS.
B ERDOMAIN
*

REDSET EQU % SUBSTITUTE FOR DSETUP U09-0057
STW, L2 CLINKTMP SAVE LINK
BAL, L2 ST3LUPSN SET LOOP PARAMS (USE COORD SPEC)
BAL, LX EXCHLUPS EXCHANGE MIDDLE/INNER LOOPS
MTW, -1 INERCNT SET INNER LOOP COUNT = D(K)-1
BGZ 13Z3 D(K)>1: GENERAL CASE: USE RSTYPE
BLZ 13Z1 D(K)=0: RESULT = IDENTITY ELMT(S)
LW, T RTTYPE D(K)=1: RESULT DATA = ARG DATA,
ISEQFIX, T USE TYPE INTG INSTEAD OF ISEQ
B 13Z2 RSTYPE = ARG TYPE.
13Z1 LW, OP OPER
LB, X IDENTBL, OP
CI, X 4 D(K)=0: GET IDENTITY VALUE CODE
BE ERLLENGTH ERROR IF NO IDENTITY VALUE
LB, T IDTYPTBL, X GET IDENTITY TYPE (USUALLY LOGL)
13Z2 STW, T RSTYPE CHANGE RSTYPE FOR SPECIAL CASE
13Z3 LW, R RTRANK
BDR, R 13Z4
LI, R 0 RSRANK =
13Z4 STW, R RSRANK MAX(RTRANK-1, 0)
LW, S OUTRCNT
MW, S MIDLCNT RSSIZE =

	STW, S	RSSIZE	$D(1) \dots D(K-1) \cdot D(K+1) \dots D(N)$	
	BAL, L1	ALOCRS	ALLOCATE RESULT DATA BLOCK	
	AW, A	RSRANK	SET RESULT DIMENS	
	LW, X	RTRANK	= $D(1), \dots, D(K-1), D(K+1), \dots, D(N)$	
	BEZ	13Z7		
	MTW, 1	RTARG		
13Z5	CW, X	COORDK	COPY ALL EXCEPT K'TH ARG	
	BE	13Z6	DIMEN TO RESULT DIMENS.	
	LW, R	*RTARG, X		
	STW, R	1, A		
	AI, A	-1		
13Z6	BDR, X	13Z5		
	MTW, -1	RTARG		
13Z7	LW, S	INERCNT	TEST FOR SPECIAL CASES	
	BGZ	13Z10	$D(K) > 1$: GENERAL CASE	
	BLZ	13Z9	$D(K) = 0$: RESULT = IDENT	
	LI, T	LOGL	$D(K) = 1$: RESULT = ARG	
	CW, T	RTTYPE	IF ARG LOGL, CHANGE TO WORD-LOGL,	
	BNE	13Z8	TO COPY DATA 32 BITS AT A TIME.	
	MTW, WORD	LOGL-LOGL RTTYPE		
13Z8	LI, X	-2	SET RTADR AND RSADR	
	BAL, L2	SETADRS1	FOR SEQUENTIAL ACCESS.	
	BAL, L1	GXSEGML	GEN LOAD	
	B	GXSTEXC1	GEN STORE, LOOP CODE; EXECUTE	
*				
FILIDENT	EQU	%	FILL RESULT WITH IDENT ELMTS	09-00003
*				09-00004
*				09-00005
13Z9	LI, X	-1	$D(K) = 0$: RESULT = IDENTITY	
	BAL, LX	SETADR	SET RSADR FOR SEQUENTIAL STORE	
	BAL, LX	GXSEGINI	INIT XSEG	
	LB, X	IDENTBL, OP	GET IDENT VALUE CODE	
	LW, R	LODIDTBL, X	GEN LOAD OF IDENT VALUE	
	GEN, 1, 0	R		
	B	GXSTEXC1	GEN STORE, LOOP CODE; EXECUTE	
13Z10	LI, A	1	$D(K) > 1$: GENERAL CASE	
	BAL, LX	XSETUP	SET ARG FOR INDEXED LOAD	
	LI, X	-1		
	BAL, LX	SETADR	SET RESULT FOR SEQ STORE	
	B	*CLINKTMP	RETURN TO SCALAR OP ROUTINE	
*				
*				
REDLOAD	EQU	%	SUBSTITUTE FOR GXSEGDL	
	STW, L3	CLINKTMP	(GEN CLOBS L3)	
	GEN, 0, 6	CODE20	GEN LOOP SETUP CODE,	
	LI, A	1		
	BAL, L1	GENLOAD	LOAD/CONVERT OF LAST ARG ELMT,	
REDSCN1	EQU	%	COMMON CODE FOR REDUCTION/SCAN	U09-0059
	STW, XL	LOOPLOC	(INNER LOOP START),	
	LW, T	RSTYPE		
	LW, R	STMPINST, T	STORE TO 'ACCUM',	
	AI, R	ACCUM		
	GEN, 1, 1	R, CODE21		
	LI, A	1		
	BAL, L1	GENLOAD	LOAD/CONVERT OF J'TH ARG ELMT.	
	LI, R	ACCUM	LEFT ARG IS IN REG, RIGHT ARG	
	STW, R	RTADR	IN 'ACCUM'.	
	B	*CLINKTMP	RETURN TO SCALAR OR ROUTINE, WHICH	
			NOW GENS CODE TO EVAL THE OP.	
*				
*				
*				
REDSTOR	EQU	%	SUBSTITUTE FOR 'GXSTEXEC'	
	LW, T	RSTYPE		
	LW, R	CODE22	GEN INNER LOOP CODE,	
	LW, R+1	STORINST, T	STORE TO RESULT,	
	AW, R+1	RSADR		
	GEN, 2, 2	R, CODE23	AND STORE INDEX CODE.	
	CLM, OP	RELATOP	IF OP IS RELATIONAL (THE ONLY ONES	
	BCS, 9	14Z1	WHICH HAVE RS TYPE DIFFERENT THAN	
	AWM, XL	-4, XL	ARG CONVERSION TYPE),	
	LW, T	RTTYPE	INSERT CONVERSION FROM RESULT	
	LW, R	CODETBL6, T	TYPE TO ARG TYPE.	
	GEN, 1, 1	R, CODE24		
	AI, XL	3		
14Z1	LW, R	LOOPLOC	FILL IN ADR OF INNER LOOP	
	AWM, R	-4, XL		
	B	EXECUTE	EXECUTE XSEG	
*				
*				

RELATOP	BOUND	8	
COMPAROP	DATA	DOPLESS,DOPEQUAL	RELATIONAL OPS - OP CODE RANGE
ANDNOROP	DATA	DOPNEQ,DOPEQUAL	COMPARISON OPS - OP CODE RANGE
*	DATA	DOPAND,DOPNOR	AND-TO-NOR OP CODE RANGE
PINF	OPEN	ERR,PINF,MINF	
MINF	EQU	2	IDENTITY = +INFINITY
ERR	EQU	3	IDENTITY = -INFINITY
IDENTBL	EQU	4	NO IDENTITY VALUE
	TABLE	DOPADD/4	IDENTITY VALUE TABLE - BY OP CODE:
	RES,1	DOPADD&3	
	DATA,1	0,0,1,1,1	ADD, SUB, MUL, DIV, POWER
	DATA,1	ERR,ERR,MINF,PINF	LOG, CIRCULAR, MAX, MIN
	DATA,1	0,1	RESIDUE, COMBINATORIAL
	DATA,1	0,1,0,1,0,1	RELATIONALS: <, <=, >, >=, /=, =
	DATA,1	1,0,ERR,ERR	AND, OR, NAND, NOR
	BOUND	4	
	CLOSE	ERR,PINF,MINF	
IDTYPTBL	TABLE	0	IDENTITY VALUE TYPE TABLE
	DATA,1	LOGL	0
	DATA,1	LOGL	1
	DATA,1	FLOT	+INF
	DATA,1	FLOT	-INF
	BOUND	4	
LODIDTBL	TABLE	0	LOAD IDENT INSTRUCTION TABLE
	LI,AI	0	0
	LI,AI	-1	1
	LD,AF	FLOTINF	+INF
	LCD,AF	FLOTINF	-INF
CLINKTMP	TEMP		COMPOSITE OP LINK TEMP
ACCUM	EQU	LFTEMP	TEMP TO ACCUMULATE REDUCTION VALUE
XSEG CODE:			
CODE20	LCW,N	RSSIZE	0 INIT STORE INDEX
	LW,K	OUTRSTEP	1 INIT LOAD INDEX
	STW,K	OUTRSAVE	2 OUTER LOOP: SAVE LOAD INDEX
	LW,N2	MIDLCNT	3 INIT MIDDLE LOOP COUNT
	STW,K	MIDLSAVE	4 MIDDLE LOOP: SAVE LOAD INDEX
	LW,N3	INERCNT	5 INIT INNER LOOP COUNT
	SW,K	INERSTEP	6 BUMP LOAD INDEX ALONG K'TH COORD
	LOAD/CNV	RTARG(K)	7 LOAD LAST (N'TH) VALUE TO INIT LOOP
*LOOPLOC	STORE	ACCUM	INNER LOOP: SAVE PREVIOUS RESULT
CODE21	SW,K	INERSTEP	BUMP TO NEXT (EARLIER) ARG ELMT
	LOAD/CNV	RTARG(K)	LOAD IT; USE IT AS LFARG, AND PREV
	OP	ACCUM	RESULT AS RTARG IN DOING 'OP'.
CODE22	BDR,N3	0 (LOOPLOC/CONVRS)	COUNT INNER LOOP
	STORE	RESULT(N)	STORE ACCUM'D VALUE IN RESULT
CODE23	BIR,N	CODE25	COUNT RESULT
	B	*RETURN	EXIT WHEN RESULT FILLED
*CONVRS	CONVERT	RESULT TYPE TO ARG TYPE	(ONLY NEED FOR
CODE24	B	0 (LOOPLOC)	RELATIONAL OPS).
CODE25	LW,K	MIDLSAVE	* RESTORE INDEX SAVED BY MIDDLE LOOP
	AW,K	MIDLSTEP	* BUMP IT
	BDR,N2	XSEGBASE+4	* COUNT MIDDLE LOOP
	LW,K	OUTRSAVE	* RESTORE INDEX SAVED BY OUTER LOOP
	AW,K	OUTRSTEP	* BUMP IT
	B	XSEGBASE+2	* CONTINUE OUTER LOOP
CODETBL6	TABLE	LOGL	CONVERSION FROM RESULT TO ARG TYPE
	LCW,AI	AI	L TO I (LOGL ARG DONE IN INTG MODE)
	LI,AI	-1	L TO C (-1 WONT MATCH ANY CHAR)
	LCW,AI	AI	L TO I
	LD,AF	FLOT01,AI	L TO F
	LCW,AI	AI	L TO I (ISEQ)
	PAGE		

U09-0061
U09-0062
U09-0063
U09-0064
U09-0065
U09-0066

*				U09-0067
SCAN	EQU	%	(MONADIC) SCAN	U09-0068
	BAL, LX	SETLINKS	SET DETOUR LINKS, ENTER DYADIC	U09-0071
*			SCALAR OP ROUTINE	U09-0072
	PZE	*MXRETURN	RETURN ADR / COPTRIG	U09-0073
	PZE	SCANTYPE	CTYPELNK FOR SCAN	U09-0074
	PZE	SCANSET	CSETLNK FOR SCAN	U09-0075
	PZE	SCANLOAD	CLOADLNK FOR SCAN	U09-0076
	PZE	SCANSTOR	CSTORLNK FOR SCAN	U09-0077
*				U09-0078
*				U09-0079
SCANTYPE	EQU	REDTYPE	SUBSTITUTE FOR DTYPEIF	U09-0080
*				U09-0081
*				U09-0082
SCANSET	EQU	%	SUBSTITUTE FOR DSETUP	U09-0083
	STW, L2	CLINKTMP	SAVE LINK	U09-0084
	BAL, L2	ST3LUPSN	SET LOOP PARAMS USING COORD SPEC	U09-0085
	LW, A	MIDLCNT		U09-0086
	AI, A	-1	IF D(K)=1, THEN RESULT = ARG	U09-0087
	BEZ	MNOP		U09-0088
	CLM, OP	RELATOP	HANDLE RELATIONAL OP SPECIALLY:	U09-0089
	BCS, 9	17Z1	RESULT TYPE = COMPARE TYPE INSTEAD	U09-0090
	LW, T	TYPETEMP	OF 'LOGL' (BECAUSE R(1)=B(1));	U09-0091
	STW, T	RSTYPE		U09-0092
	AI, T	-CHAR	'CHAR' ARG NOT ALLOWED FOR D(K)>1.	U09-0093
	BEZ	ERDOMAIN		U09-0094
17Z1	BAL, L1	RSLIKRT2	ALLOCATE RS WITH SAME RANK AND	U09-0095
*			SIZE AS ARG.	U09-0096
	BAL, LX	MRTDIMS	COPY ARG DIMENS TO RESULT	U09-0097
	LI, A	1		U09-0098
	BAL, LX	XSETUP	SET RTADR FOR INDEXED LOAD	U09-0099
	LI, A	2		U09-0100
	BAL, LX	XSETUP	SET RSADR FOR INDEXED STORE	U09-0101
	B	*CLINKTMP	RETURN TO DYADIC OP ROUTINE	U09-0102
*				U09-0103
*				U09-0104
SCANLOAD	EQU	%	SUBSTITUTE FOR GXSEGDL	U09-0105
	STW, L3	CLINKTMP	SAVE LINK	U09-0106
	LI, XL	XSEGBASE	DISCARD CODE GENED BY GXSEGINI	U09-0107
	LW, A	OPER	GET UNMODIFIED OP	
	LB, R	ASSOCTBL, A	CHECK IF ASSOCIATIVE	
	BNEZ	ASCNLOAD	YES, DO SPECIAL LOAD/STORE	
	GEN, 0, 8	CODE40	GEN LOOP SETUP CODE,	U09-0108
	LI, A	1		U09-0109
	BAL, L1	GENLOAD	LOAD/CONVERT OF ARG ELEMENT,	U09-0110
	GEN, 0, 3	CODE41	AND 1ST-ELEMENT TEST CODE.	U09-0111
	AWM, XL	-2, XL		U09-0112
	B	REDSN1	FINISH AS FOR REDUCTION: THIS GIVES	U09-0113
*			CORRECT XSEG CODE EXCEPT FOR ONE	U09-0114
*			ITEM: AN 'INERSTEP' ADR THAT HAS	U09-0115
*			TO BE CHANGED TO 'MIDLSTEP'. THIS	U09-0116
*			IS DONE BY SCANSTOR BELOW ...	U09-0117
*				U09-0118
*				U09-0119
SCANSTOR	EQU	%	SUBSTITUTE FOR GXSTEXEC	U09-0120
	CLM, OP	RELATOP	FOR A RELATIONAL OP,	U09-0121
	BCS, 9	17Z3	GEN CODE TO CONVERT LOGL VALUE	U09-0122
	LW, T	TYPETEMP	TO COMPARISON TYPE.	U09-0123
	STW, T	RSTYPE		U09-0124
	GENX	CONVTABL+4*LOGL, T		U09-0125
17Z3	GEN, 0, 2	CODE43	GEN LOOP CONTROL CODE	U09-0126
	LW, X	LOOPLOC	CHANGE 'INERSTEP' TO 'MIDLSTEP'	U09-0127
	MTW, 3	1, X	AS MENTIONED ABOVE.	U09-0128
	AWM, X	-2, XL		U09-0129
	AWM, XL	-1, X		U09-0130
	LW, T	RSTYPE		U09-0131
	LW, R	STORINST, T	GEN STORE,	U09-0132
	AW, R	RSADR		U09-0133
	GEN, 1, 5	R, CODE44	AND FINAL LOOP CONTROL CODE.	U09-0134
	B	EXECUTE	EXECUTE XSEG	U09-0135
*				U09-0136
*				U09-0137
*				U09-0138
*				U09-0139
		XSEG CODE:		
CODE40	LI, K	0	0 INIT INDEX TO 1ST ELEMENT	U09-0140
	LW, N1	OUTRCNT	1 INIT OUTER LOOP COUNT	U09-0141
	LW, N2	MIDLCNT	2 OUTER LOOP: INIT MIDDLE LOOP COUNT	U09-0142
	LI, R	0	3	U09-0143

	STW,R	COUNT	4	INIT 1ST-TIME COUNTER	U09-0144
	LW,N3	INERCNT	5	MIDDLE LOOP: INIT INNER LOOP COUNT	U09-0145
	MTW,1	COUNT	6	BUMP 1ST-TIME COUNTER	U09-0146
*	STW,K	KTEMP	7	INNER LOOP: REMEMBER ARG/RS INDEX	U09-0147
* CODE41	LOAD/CNV	RTARG(K)	8	GET CORRESPONDING ARG ELMT	U09-0148
	LW,N	COUNT		IF THIS IS THE FIRST TIME IN THE	U09-0149
	BDR,N	0 (%+2)		MIDDLE LOOP, WE'VE GOT THE RESULT	U09-0150
	B	0 (STORE RS)		ELMT; OTHERWISE, USING OUR	U09-0151
* LOOPLOC	STORE	ACCUM		VALUE AS RT ARG OF 'OP', PICK	U09-0152
*	SW,K	MIDLSTEP		LF ARG VALUES FROM SUCCESSIVELY	U09-0153
*	LOAD/CNV	RTARG(K)		EARLIER ARG ELMTS MOVING ALONG	U09-0154
*	OP	ACCUM		K'TH COORD; APPLY 'OP' EACH TIME.	U09-0155
*	(CONVERT	LOGL TO RS TYPE)		(RELATIONAL OPS ONLY)	U09-0156
CODE43	BDR,N	0 (LOOPLOC)		CONTINUE 'TIL WE'VE USED LEFTMOST	U09-0157
	LW,K	KTEMP		ELMT ON K'TH COORD; RESTORE K.	U09-0158
*	STORE	RESULT(K)		STORE RESULT ELEMENT	U09-0159
CODE44	AW,K	INERSTEP		BUMP ARG/RS INDEX	U09-0160
	BDR,N3	XSEGBASE+7		COUNT INNER LOOP	U09-0161
	BDR,N2	XSEGBASE+5		COUNT MIDDLE LOOP	U09-0162
	BDR,N1	XSEGBASE+2		COUNT OUTER LOOP	U09-0163
	B	*RETURN		EXIT	U09-0164
*					U09-0165
*					U09-0166
KTEMP	EQU	K1TEMP			U09-0167
	PAGE				
*					
*				SPECIAL CASE OF 'SCAN' FOR ASSOCIATIVE OPS:	
*				THE TIME REQUIRED TO PROCESS AN N-ELEMENT VECTOR	
*				IS PROPORTIONAL TO N, INSTEAD OF THE N*N/2	
*				NORMALLY REQUIRED.	
*				OPS INCLUDED ARE:	
*				STRAIGHT - ADD, MULTIPLY, MIN, MAX,	
*				AND, OR, NAND, NOR.	
*				ALTERNATING - SUBTRACT, DIVIDE.	
*					
ASCNLOAD	EQU	%		SCANLOAD FOR ASSOCIATIVE OPS	
	BAL,LX	EXCHLUPS		EXCHANGE INNER/MIDDLE LOOP PARAMS	
	GEN,0,6	CODE50		GEN LOOP INIT CODE	
	LI,A	1			
	BAL,L1	GENLOAD		GEN LOAD/CONVERT OF ARG ELMT	
	STW,XL	LOOPLOC		SAVE ADR OF 'B ...'	
	GEN,0,2	CODE51		GEN BRANCH AND INDEX-INCR	
	LW,T	RSTYPE		IF ARG TYPE DIFFERS FROM	
	CLM,OP	ANDNOROP		RESULT TYPE OR IF OP IS	
	BCR,9	ASCNL1		-AND-OR-NAND-NOR	
	CW,T	RTTYPE		WE MUST GEN CODE TO	
	BE	17Z4		SAVE ACCUM, LOAD & CONVERT	
ASCNL1	LW,LX	STMPINST,T		ARG ELEMENT,SAVE THAT IN	
	AI,LX	ACCUM		A TEMP, THEN RESTORE ACCUM VALUE.	
	GEN,1,0	LX			
	LI,A	1			
	BAL,L2	GENLOADT			
	SW,LX	=X'03000000'		(CHANGE STW/STD TO LW/LD)	
	GEN,1,0	LX			
17Z4	LW,R	RTADR		GET ADR OF ARG ELMT FOR OP ROUTINE	
	STW,XL	XSADRTMP		SAVE ADR OF MAIN INST	
	LI,A	ASCNSTOR		CHANGE STORE DETOUR ADDRESS	
	STW,A	COPLINKS+4		TO SPECIAL ROUTINE FOR ASSOC OPS.	
	B	*CLINKTMP		EXIT FROM LOAD ROUTINE: GEN OP	
*					
*					
ASCNSTOR	EQU	%		SCANSTOR FOR ASSOCIATIVE OPS	
	AWM,XL	*LOOPLOC		FILL IN BRANCH ADR	
	MTW,1	LOOPLOC		MAKE LOOPLOC POINT TO 'AW,K ...'	
	LW,T	RSTYPE			
	LW,R	STORINST,T		GEN STORE	
	AW,R	RSADR			
	GEN,1,2	R,CODE53		GEN 1ST BDR	
	LB,R	ASSOCTBL,OP		IS OP ALSO COMMUTATIVE?	
	BDR,R	17Z5		BRANCH IF SO	
	LW,X	XL		NON-COMMUTATIVE: GEN AN EXTRA	
	SW,X	LOOPLOC		COPY OF 'OP ARG(K)' CODE.	
	SCS,X	-4			
	LB,XL-1	ALTEROP,OP		CHANGE SUBTRACT (DIVIDE) OP	
	LC	X			
	LM,XL+1	*LOOPLOC			
	STM,XL+1	0,XL			
	STB,XL-1	*XSADRTMP		TO ADD (MULTIPLY) IN 1ST COPY.	

```

AWM,XL -2,XL FILL IN PREV BDR ADR TO JUMP HERE
SLS,XL 1 UPDATE XL TO INCLUDE NEW CODE
SW,XL LOOPLOC
LI,R X'E0000' SET MASK FOR COMPARISON
AND,R -5,XL STRIP ADDRESS FROM INSTRUCTION
CW,R BCR40 CHECK IF SUSPECT INSTRUCTION
* IS INTEGER OVERFLOW TEST.
* NO.
* YES,FORM CORRECTED ADDRESS
17Z5 BNE 17Z5 AND RESTORE IN XSEG.
AW,R XL
AI,R -3
STW,R -5,XL
LW,R LOOPLOC
AWM,R -2,XL FILL IN LAST BDR ADR TO GO BACK
B EXECUTE EXECUTE THE XSEG
BCR40 BCR,4 0 (USED IN COMPARISON)
*
*
DOPIADD EQU DOPADD-3 INTG ADD OP INDEX
ASSOCTBL TABLE DOPIADD/4 ASSOCIATIVE OP TABLE
RES,1 DOPIADD&3 0=ORD, 1=ASSOC, 2=ASSCO&COMMUT
DATA,1 2,1,2 INTG ADD, INTG SUB, INTG MUL
DATA,1 2,1,2,1,0 ADD, SUB, MUL, DIV, POWER
DATA,1 0,0,2,2 LOG, CIRCULAR, MAX, MIN
DATA,1 0,0 RESIDUE, COMBINATORIAL
DATA,1 0,0,0,0,0,0 < <= > >= /= =
DATA,1 2,2,0,0 AND,OR,NAND,NOR
BOUND 4
*
*
ALTEROP TABLE DOPIADD/4 ALTERNATE OP TABLE
RES,1 DOPIADD&3
DATA,1 0,X'30' INTG SUB: CHANGE SW TO AW
DATA,1 0,0,X'1D' FLOT SUB: CHANGE FSL TO FAL
DATA,1 0,X'1F' FLOT DIV: CHANGE FDL TO FML
BOUND 4
*
*
XSEG CODE FOR SCAN W/ASSOCIATIVE OPS:
*
*
CODE50 LI,K 0 0 INIT INDEX
LW,N1 OUTRCNT 1 INIT OUTER LOOP COUNT
STW,K OUTRSVAVE 2 OUTER LOOP: SAVE INDEX
LW,N2 MIDLCNT 3 INIT MIDDLE LOOP COUNT
STW,K MIDLSAVE 4 MIDDLE LOOP: SAVE INDEX
LW,N3 INERCNT 5 INIT INNER LOOP COUNT
* LOAD/CNV RTARG(K) 6 GET 1ST ELMT (ON K'TH COORD)
CODE51 B 0 (TO STORE) STORE 1ST, PROCESS FOLLOWING ELMTS
AW,K INERSTEP INNER LOOP: BUMP INDEX
* OP RTARG(K) INCLUDE NEXT ARG ELMT
* STORE RESULT(K) STORE AS NEXT RESULT ELMT
CODE53 BDR,N3 0 (TO AW,K...) GO DO NEXT ELMT
B CODE54 END INNER LOOP
*
*
ADDITIONAL CODE INCLUDED ONLY FOR ALTERNATING OPS
(SUBTRACT/DIVIDE):
*
*
AW,K INERSTEP (ALMOST THE SAME AS ABOVE,
* OP RTARG(K) EXCEPT FOR MAIN OP:
* STORE RESULT(K) HERE IT'S SUBTRACT OR
* BDR,N3 0 (1ST AW,K...) DIVIDE; ABOVE IT WAS CHANGED
* B CODE54 TO ADD OR MULTIPLY.)
*
*
CODE54 LW,K MIDLSAVE * END OF INNER LOOP: RESTORE MIDDLE
AW,K MIDLSTEP * LOOP INDEX VAL, BUMP IT.
BDR,N2 XSEGBASE+4 * COUNT MIDDLE LOOP
LW,K OUTRSVAVE * END OF MIDDLE LOOP: RESTORE OUTER
AW,K OUTRSTEP * LOOP INDEX VAL, BUMP IT.
BDR,N1 XSEGBASE+2 * COUNT OUTER LOOP
B *RETURN * END OF OUTER LOOP: EXIT
PAGE
*
*
* OUTER PRODUCT:
*
* LFARG CIRCLE.OP RTARG
*

```

OUTER	EQU	%	(DYADIC) OUTER PRODUCT
*	BAL, LX	SETLINKS	SET DETOUR LINKS; ENTER DYADIC
			SCALAR OP ROUTINE.
	PZE	*DXRETURN	RETURN ADR / COPTRIG
	PZE	OUTRTYP	CTYPELNK FOR OUTER PRODUCT
	PZE	SETOUTER	CSETLNK FOR OUTER PRODUCT
	PZE	OUTRLOAD	CLOADLNK FOR OUTER PRODUCT
	PZE	OUTRSTOR	CSTORLNK FOR OUTER PRODUCT
*			
*			
OUTRTYP	EQU	%	SUBSTITUTE FOR 'DTYPEIF'
	LI, T	CHAR	
	CB, T	*LFARG	IF EITHER ARG IS CHAR,
	BE	TESTEQ	SPECIAL CASE FOR EQUAL/
	CB, T	*RTARG	NOTEQUAL OPS.
	BE	TESTEQ	
	B	DTYPEIF1	OTHERWISE, USE 'DTYPEIF'.
*			
*			
OUTRLOAD	EQU	%	SUBSTITUTE FOR GXSEGD
	MTW, -1	XSEGBASE	CHANGE 'RSSIZE' TO 'RTSIZE'
	GEN, 0, 2	CODE30	GEN OUTER LOOP INIT,
	LI, A	1	LOAD/CONVERT OF RTARG,
	BAL, L2	GENLOADT	STORE TO RTTEMP,
	GEN, 0, 3	CODE31	INNER LOOP INIT,
	STW, XL	LOOPLOC	
	LI, A	0	LOAD/CNV LEFT ARG.
	BAL, L1	GENLOAD	
	LW, R	RTADR	WITH LFARG IN REG, GEN CODE TO
	B	*L3	EVAL OP USING RTTEMP.
*			
*			
OUTRSTOR	EQU	%	SUBSTITUTE FOR GXSTEXEC
	LW, T	RSTYPE	GEN STORE IN RESULT,
	LW, R	STORINST, T	
	AW, R	RSADR	
	GEN, 1, 2	R, CODE32	INNER LOOP CONTROL CODE,
	GEN, 0, 6	CODE36	AND OUTER LOOP CONTROL CODE.
	LW, R	LOOPLOC	
	AWM, R	-6, XL	FILL IN INNER LOOP ADR
	LW, S	RSSIZE	RECOMPUTE RTSIZE (SETTING RTADR
	DW, S	LFSIZE	LOUSES IT UP IF RTARG IS CHAR;
	STW, S	RTSIZE1	BUT LFSIZE IS OK - XSETUP DOESN'T
*			CHANGE LFSIZE).
	B	EXECUTE	EXECUTE XSEG
*			
*			
*			
*			
		XSEG CODE:	
	LCW, N	RTSIZE	0 INIT RTARG INDEX
CODE30	LI, K	0	1 INIT STORE INDEX
	STW, K	OUTRSAVE	2 OUTER LOOP: SAVE STORE INDEX
*	LOAD/CNV	RTARG(N)	3 LOAD LFARG ELEMENT
*	STORE	RTTEMP	STORE IN TEMP
CODE31	AI, N	1	BUMP RT INDEX
	LI, K1	0	INIT LFARG INDEX
	LW, N2	LFSIZE	INIT INNER LOOP COUNT
*LOOPLOC	LOAD/CNV	LFARG(K1)	INNER LOOP: FETCH LFARG TO REG
*	OP	RTTEMP	EVAL OP, USING RTARG IN RTTEMP
*	STORE	RESULT(K)	STORE ANSWER IN RESULT
CODE32	AW, K	RTSIZE1	BUMP STORE INDEX
	AI, K1	1	BUMP LFARG INDEX
CODE36	BDR, N2	0 (LOOPLOC)	COUNT INNER LOOP
	LW, K	OUTRSAVE	RESTORE STORE INDEX OUTER LOOP SAVED
	AI, K	1	BUMP IT
	CW, K	RTSIZE1	
	BL	XSEGBASE+2	COUNT OUTER LOOP
	B	*RETURN	EXIT WHEN DONE
*			
RTSIZE1	EQU	MIDLSAVE	
	PAGE		
*			
*			
*	INNER PRODUCT:		
*			
*		LFARG OP2.OP1 RTARG	
*			
INNER	EQU	%	(DYADIC) INNER PRODUCT

	SLS, OP	-8	
	AND, OP	=X'FF'	ISOLATE OUTER OP CODE (OP2)
	STW, OP	OTHEROP	SAVE IT
	LW, OP	OPER	RESTORE OP
	BAL, LX	SETLINKS	SET DETOUR LINKS; PUT INNER OP CODE
*			(OP1) IN OPER; ENTER SCALAR
*			OP ROUTINE.
INR2LNKS	EQU	%	
	PZE	*DXRETURN	RETURN ADR / COPTRIG
	PZE	INRTYPE1	CTYPELNK FOR INNER PRODUCT (OP1)
	PZE	INRSET1	CSETLNK FOR INNER PRODUCT (OP1)
	PZE	INRLOAD1	CLOADLNK FOR INNER PRODUCT (OP1)
	PZE	INRSTOR1	CSTORLNK FOR INNER PRODUCT (OP1)
*			
*			
INRTYPE1	EQU	OUTRTYP	SUBSTITUTE FOR 'DTYPEIF'
*			
*			
INRSET1	EQU	%	SUBSTITUTE FOR DSETUP
	STW, L2	CLINKTMP	SAVE LINK
	LW, X	OTHEROP	TYPE BY WHICH TO ALOC RESULT IS
	LB, T	RSTYPTBL, X	DETERMINED BY OUTER OP (OP2).
	BNEZ	15Z1	IF OP2 IS IN INTG/FLOT GROUP,
	LW, T	RSTYPE	RESULT TYPE IS MAX (T, INTG),
	CI, T	LOGL	WHERE T = OP2'S ARG TYPE,
	BNE	15Z1	= OP1'S RESULT TYPE.
	LI, T	INTG	
15Z1	STW, T	OUTRTYPE	REMEMBER OP2'S RESULT TYPE
	XW, T	RSTYPE	USE IT TO ALOC RESULT
	STW, T	INERTYPE	REMEMBER OP1'S RESULT TYPE
	BAL, L2	SETINNER	CHECK CONFORM; ALOC RS; SET LOOPS
	LW, T	INERTYPE	RESTORE OP1'S TYPE
	STW, T	RSTYPE	
	B	*CLINKTMP	RETURN TO SCALAR OP ROUTINE
*			
*			
INRLOAD1	EQU	%	SUBSTITUTE FOR GXSEGD1
	STW, L3	CLINKTMP	(GEN CLOBS L3)
	GEN, 0, 9	CODE1	GEN LOOP INIT CODE (THIS SEQUENCE
	LW, L3	CLINKTMP	
*			IS ACTUALLY ONLY 8 WORDS LONG,
*			THE 9'TH WORD IS RESERVED FOR
*			A STORE INST, TO BE FILLED IN
*			LATER).
	B	GXSEGD1	MAKE RTARG ADDRESSABLE, LOAD LFARG;
*			RETURN TO SCALAR OP ROUTINE
*			TO GEN OP1 EVAL CODE.
*			
INRSTOR1	EQU	%	SUBSTITUTE FOR GXSTEXEC
	LI, OP	1	
	CW, OP	RTCOMDIM	
	BE	15Z2	IF D(1)>1,
	GEN, 0, 1	CODE3	GEN RTARG INDEX BUMP CODE.
15Z2	CW, OP	LFCOMDIM	
	BE	15Z3	IF C(M)>1,
	GEN, 0, 1	CODE6	GEN LFARG INDEX BUMP CODE.
15Z3	CW, OP	INERCNT	IF C(M)=D(1)=1,
	BE	KILLOP2	DONT BOTHER GEN'ING OP2 CODE (OP
*			HAS BEEN CLOBBED BECAUSE OF
*			KILLOP2'S OP TEST, TO FAIL IT).
	STW, XL	XSADRTMP	SAVE XSEG LOC
	LW, OP	OPER	SAVE INNER OP
	XW, OP	OTHEROP	SET UP TO DO OUTER OP (OP2)
	BAL, LX	SETLINKS	CHANGE DETOUR LINKS; ENTER DYADIC
*			SCALAR OP ROUTINE FOR OP2.
	PZE	*DXRETURN	RETURN ADR / COPTRIG
	PZE	INRTYPE2	CTYPELNK FOR INNER PRODUCT (OP2)
	PZE	INRSET2	CSETLNK FOR INNER PRODUCT (OP2)
	PZE	INRLOAD2	CLOADLNK FOR INNER PRODUCT (OP2)
	PZE	INRSTOR2	CSTORLNK FOR INNER PRODUCT (OP2)
*			
*			
INRTYPE2	EQU	%	SUBSTITUTE FOR DTYPEIF
	LW, T	INERTYPE	OP1'S RESULT TYPE BECOMES
	STW, T	LFTYPE	OP2'S ARG TYPES.
	STW, T	RTTYPE	
	STW, T	RSTYPE	SET OP'S RESULT TYPE

	B	%+1-LOGL,T	WHAT IS RS TYPE?
	MTW,	INTG-LOGL RSTYPE	LOGL: SET RS TO INTG
	NOP		CHAR: (CAN'T BE THIS TYPE)
	B	0,LX	INTG OR LOGL: TAKE INTG EXIT
	B	1,LX	FLOT: TAKE FLOT EXIT
*			
*			
INRSET2	EQU	%	SUBSTITUTE FOR DSETUP
	B	*L2	THE SETUP HAS ALREADY BEEN DONE
*			
*			
INRLOAD2	EQU	%	SUBSTITUTE FOR GXSEGD
	LW,XL	XSADRTMP	RESTORE XSEG ADR
	LW,T	RSTYPE	= THE TYPE TO WHICH OP2'S
	LW,R	STMPINST,T	ARGS ARE TO BE CONVERTED.
	AI,R	ACCUM	FILL IN THAT STORE INST FOR WHICH
	STW,R	XSEGBASE+9	SPACE WAS RESERVED.
	LW,T	INERTYPE	
	SLS,T	2	
	AW,T	RSTYPE	GEN CONVERSION OF OP1'S RESULT
	GENX	CONVTABL,T	TO OP2'S ARG CONV MODE.
	GEN,0,4	CODE33	GEN FIRST-TIME TEST
	AWM,XL	-3,XL	
	STW,XL	LOOPLOC	SAVE OP2 LOC
	LI,R	ACCUM	SET RTADR = ACCUM
	STW,R	RTADR	
	B	*L3	GO GEN CODE TO EVAL OP2
*			
*			
INRSTOR2	EQU	%	SUBSTITUTE FOR GXSTEXEC
	LW,R	OTHEROP	PUT OPS BACK THE WAY THEY WERE
	XW,R	OPER	DURING INRTYPE1, SO THAT
	STW,R	OTHEROP	'INTGOVFL' WILL WORK OK IF CALLED.
	LCI	3	RESTORE SETUP/LOAD/STORE LINKS
	LM,R	INR2LNKS+2	AS THEY WERE FOR OP1, SO THAT
	STM,R	COPLINKS+2	'INTGOVFL' WILL WORK RIGHT.
	GEN,0,1	CODE34	GEN INNER LOOP CONTROL
	LW,X	LOOPLOC	
	AWM,XL	-1,X	FILL IN BRANCH ADR
KILL0P2	LW,T	RSTYPE	
	LW,R	STORINST,T	GEN STORE IN RESULT,
	AW,R	RSADR	
	GEN,1,2	R,CODE16	AND STORE-INDEX CONTROL CODE.
	CLM,OP	RELATOP	IF OP2 IS RELATIONAL,
	BCS,9	EXECUTE	INSERT CODE TO CONVERT ITS
	AW,XL	BDRN3INS	LOGL RESULT TO ITS ARG CONVERSION
	STW,XL	-4,XL	MODE.
	LW,T	RSTYPE	
	LW,R	CODETABL6,T	
	GEN,1,1	R,CODE35	
	B	EXECUTE	EXECUTE XSEG
*			
*			
RSTYPTBL	TABLE	DOPADD/4	RESULT TYPE TABLE - BY OP:
	RES,1	DOPADD&3	
	DATA,1	0,0,0,FLOT,0	ADD, SUB, MUL, DIV, POWER
	DATA,1	FLOT,FLOT,0,0	LOG, CIRCULAR, MAX, MIN
	DATA,1	0,0	RESIDUE, COMBINATORIAL
	DATA,1	LOGL,LOGL,LOGL	RELATIONAL OPS: <, <=, >,
	DATA,1	LOGL,LOGL,LOGL	>=, /=, =.
	DATA,1	LOGL,LOGL	AND, OR
	DATA,1	LOGL,LOGL	NAND, NOR
	BOUND	4	
*			
BDRN3INS	BDR,N3	0	
*			
OTHEROP	TEMP		OP1/OP2 ARE KEPT HERE AND IN OPER
XSADRTMP	TEMP		XSEG ADR SAVE TEMP
INERTYPE	TEMP		OP1 RESULT TYPE
OUTRTYPE	TEMP		OP2 RESULT TYPE
*			
*			
*			
		XSEG CODE:	
*			
	LCW,N	RSSIZE	0 INIT STORE INDEX
*CODE1	LW,K1	LFCDMDIM	1 INIT LFARG INDEX
*	AI,K1	-1	2
*	LW,K	RTSIZE	3 INIT RTARG INDEX

```

*      SW,K      MIDLCNT      4  OUTER LOOP: BUMP RTARG INDEX
*      STW,K1    OUTRSAVE     5  SAVE LFARG INDEX
*      LW,N2     MIDLCNT      6  INIT MIDDLE LOOP COUNT
*      STW,K     MIDLSAVE     7  MIDDLE LOOP: SAVE RTARG INDEX
*      LW,N3     INERCNT      8  INIT INNER LOOP COUNT
*      STORE     ACCUM        9  INNER LOOP: STORE PREVIOUS OP2 RS
*      LOAD/CNV RTARG(K)      LOAD NEW RTARG ELMT
*      STORE     RTTEMP       SAVE IT IN TEMP
*      LOAD/CNV LFARG(K1)     LOAD NEW LFARG ELMT
*      OP1       RTARG(K)/RTTEMP DO OP1
*CODE3  SW,K      MIDLCNT      (CONDITIONALLY GEN'D) BUMP RT INDEX
*CODE6  AI,K1     -1          (CONDITIONALLY GEN'D) BUMP LF INDEX
*      CONVERT   OP1 RESULT TO OP2 ARG TYPE
CODE33  CW,N3     INERCNT
      BNE        0          (%+3)  IF IT'S THE 1ST TIME IN INNER LOOP,
      BDR,N3     XSEGBASE+9      DON'T DO OP2 (REG=RESULT).
      B          0          (%+3)
*LOOPLOC OP2     ACCUM        NOT 1ST TIME: DO OP2
CODE34  BDR,N3   XSEGBASE+9 /CONVRS COUNT INNER LOOP (MAYBE CONV)
*      STORE     RESULT(N)      STORE ACCUM'D VALUE IN RESULT
*CODE16 BIR,N     CODE8         COUNT RESULT
*      B         *RETURN       EXIT WHEN RESULT FILLED
*CONVRS  CONVERT OP2 RESULT TO OP2 ARG TYPE (RELATIONAL OPS ONLY)
CODE35  B        XSEGBASE+9
*CODE8  LW,K1    OUTRSAVE      * RESTORE RTARG
*      LW,K     MIDLSAVE      * AND LFARG INDECES.
*      AI,K     1             * BUMP RTARG INDEX
*      BDR,N2   XSEGBASE+7     * COUNT MIDDLE LOOP
*      AW,K1    LFCOMDIM      * BUMP LFARG INDEX
*      B        XSEGBASE+4     * CONTINUE OUTER LOOP
      PAGE

```

SETUP ROUTINES FOR COMPOSITE OPS

SET DETOUR LINKS

SETS THE COMPOSITE OP DETOUR LINKS TO THE VALUES
GIVEN IN BAL+1,...,BAL+5; SETS OP AND OPER TO RIGHT
OP; ENTERS CORRESPONDING DYADIC SCALAR OP ROUTINE.
LINK IS LX.

```

SETLINKS EQU      %
AND,OP      =X'FF'      CLEAN UP OP
STW,OP      OPER       COPY TO OPER
LCI         5
LM,BUF      0,LX       COPY GIVEN DETOUR LINKS
STM,BUF     COPLINKS    AND COPTRIG VALUE.
B           DXTABLE,OP  ENTER DYADIC SCALAR OP ROUTINE

```

```

COPLINKS EQU      COPTRIG      COMP OP LINK TABLE ADR
RETURN EQU        COPTRIG      RETURN ADR / COPTRIG CELL
COPTRIG TEMP      COPTRIG      COMP OP TRIG
CTYPELNK TEMP     DETOUR LINK FOR DTYPEIF
CSETLNK TEMP      DETOUR LINK FOR DSETUP
CLOADLNK TEMP     DETOUR LINK FOR GXSEGD
CSTORLNK TEMP     DETOUR LINK FOR GXSTEXEC
      PAGE

```

SET UP FOR OUTER PRODUCT / ENCODE

1. SETS UP LF/RT/RS RANK/SIZE
 2. ALLOCATES RESULT
 3. ESTABLISHES RESULT DIMENS
 4. SETS UP LFADR FOR INDEXED LOAD USING K1,
RTADR FOR SEQUENTIAL LOAD USING N,
AND RSADR FOR INDEXED STORE USING K.
- LINK IS L2.

```

SETOUTER EQU      %
LI,A        0
BAL,LX      SETUPARG      SET UP ARG PARAMS
LI,A        1
BAL,LX      SETUPARG      *
LW,R        LFRANK

```

	AW, R	RTRANK	RESULT RANK = SUM OF ARG RANKS	
	STW, R	RSRANK		
	LW, S	LFSIZE		
	MW, S	RTSIZE	RESULT SIZE = PRODUCT OF ARG SIZES	
	STW, S	RSSIZE		
	BAL, L1	ALOCRS	ALLOCATE RESULT DB	
	AW, A	RSRANK	POINT TO LAST DIMEN -1	
	LW, X	RTRANK		
	BEZ	9Z2		
	MTW, 1	RTARG		
9Z1	LW, R	*RTARG, X	COPY RTARG DIMENS	
	STW, R	1, A	TO RESULT DIMENS.	
	AI, A	-1		
	BDR, X	9Z1		
	MTW, -1	RTARG		
9Z2	LW, X	LFRANK		
	BEZ	9Z4		
	MTW, 1	LFARG		
9Z3	LW, R	*LFARG, X	COPY LFARG DIMENS	
	STW, R	1, A	TO RESULT DIMENS.	
	AI, A	-1		
	BDR, X	9Z3		
	MTW, -1	LFARG		
9Z4	LI, A	2		
	BAL, LX	XSETUP	SET RSADR FOR INDX STORE USING K	
	LI, X	-2		
	BAL, LX	SETADR	SET RTADR FOR SEQ LOAD USING N	
	LI, A	0		
	BAL, LX	XSETUP	SET LFADR FOR INDEXED LOAD,	
	BAL, L1	SETSPEC2	USING K1.	
	B	*L2		
	PAGE			
*				
*				
*	SET UP FOR INNER PRODUCT / DECODE			
*				
*	1. SETS UP LF/RT/RS RANK/SIZE			
*	2. CHECKS CONFORMABILITY			
*	3. SETS THE FOLLOWING LOOP CONTROL PARAMS:			
*	LFCOMDIM = C(M) (LFARG'S LAST DIMEN)			
*	RTCOMDIM = D(1) (RTARG'S FIRST DIMEN)			
*	INERCNT = MAX(C(M), D(1))			
*	MIDLCNT = D(2)*...*D(N)			
*	OUTRCNT = C(1)*...*C(M-1)			
*	4. ALLOCATES RESULT			
*	5. ESTABLISHES RESULT DIMENS			
*	6. SETS UP LFADR FOR INDEXED LOAD USING K1,			
*	RTADR FOR INDEXED LOAD USING K,			
*	AND RSADR FOR SEQUENTIAL STORE USING N.			
*	THIS ROUTINE AUTOMATICALLY HANDLES THE CASES		09-00007	
*	WHERE C(M)=0 OR D(1)=0: IT FILLS THE RESULT WITH		09-00008	
*	THE APPROPRIATE IDENTITY ELEMENTS, AND		09-00009	
*	DOES NOT RETURN TO THE CALLER. IF IT		09-00010	
*	DOES RETURN, INERCNT>0.		09-00011	
*	LINK IS L2.			
*				
SETINNER	EQU	%		
	LI, A	0		
	BAL, LX	SETUPARG	SET UP ARG PARAMS	
	LI, A	1		
	BAL, LX	SETUPARG	*	
	LI, S	1		
	CW, S	LFSIZE	SET 'LEFT COMBINING DIMEN'	
	BE	10Z1	TO 1 (IF LFARG IS SCALAR OR	
	LW, X	LFRANK	1-ELMT THING), OR C(M) (LFARG'S	
	AI, X	1	LAST DIMEN).	
	LW, S	*LFARG, X		
10Z1	STW, S	LFCOMDIM		
	LI, S	1		
	CW, S	RTSIZE	SET 'RIGHT COMBINING DIMEN'	
	BE	10Z2	TO 1 (IF RTARG IS SCALAR OR	
	LI, X	2	1-ELMT THING), OR D(1) (RTARG'S	
	LW, S	*RTARG, X	FIRST DIMEN).	
10Z2	STW, S	RTCOMDIM		
	CW, S	LFCOMDIM	DENOTING THE COMBINING DIMENS	
	BE	10Z4	BY L AND R,	
	CI, S	1	WE MUST HAVE FOR CONFORMABILITY:	
	BNE	10Z3	L=R, OR L=1, OR R=1	

	LW, S	LFCOMDIM		
	B	10Z4		
10Z3	LI, R	1		
	CW, R	LFCOMDIM		
	BNE	ERLENGTH		
10Z4	STW, S	INERCNT	INNER LOOP COUNT = MAX(L, R)	
	BDR, S	10Z41		
	AI, S	1	INERCNT = 0 OR 1	09-00015
	BGZ	10Z405	BRANCH IF 1	09-00016
	LI, OP	DOPADD	INERCNT = 0: SET UP TYPE	09-00017
	LW, X	COPTRIG	OF IDENTITY ELEMENT	09-00018
	BGEZ	10Z404		09-00019
	LW, OP	OTHEROP	(LEFT OP FOR INNER PRODUCT,	09-00020
10Z404	LB, X	IDENTBL, OP	'+' FOR DECODE)	09-00021
	LB, T	IDTYPTBL, X		09-00022
	B	10Z409		09-00023
10Z405	EQU	%	INERCNT = 1	09-00024
	LW, S	COPTRIG	IF INERCNT=1 ON INNER PRODUCT,	
	BGEZ	10Z41		
	LW, T	INERTYPE	RESULT TYPE IS INNER OP TYPE,	
*			NOT OUTER OP TYPE.	09-00026
10Z409	STW, T	RSTYPE		09-00027
10Z41	LW, R	LFRANK		
	BDR, R	10Z5		
	LI, R	0		
10Z5	LW, S	RTRANK	RESULT RANK =	
	BDR, S	10Z6	MAX(LFRANK-1, 0) + MAX(RTRANK-1, 0).	
	LI, S	0		
10Z6	AW, R	S		
	STW, R	RSRANK		
	LW, S	LFSIZE		
	DW, S	LFCOMDIM	OUTER LOOP COUNT =	
	BNOV	10Z63	C(1)*C(2)*...*C(M-1).	09-00029
	LI, S	1	C(M)=0: COMPUTE OUTERCNT	09-00030
	MTW, 1	LFARG	THE HARD WAY.	09-00031
	LW, X	LFRANK		09-00032
	B	10Z62		09-00033
10Z61	MW, S	*LFARG, X		09-00034
	BOV	ERLENGTH		09-00035
10Z62	BDR, X	10Z61		09-00036
	MTW, -1	LFARG		09-00037
10Z63	EQU	%		09-00038
	STW, S	OUTRCNT	C(1)*C(2)*...*C(M-1).	
	LW, S	RTSIZE		
	DW, S	RTCOMDIM	MIDDLE LOOP COUNT =	
	BNOV	10Z66	D(2)*D(3)*...*D(N).	09-00040
	LI, S	1	D(1)=0: COMPUTE MIDLCNT	09-00041
	MTW, 2	RTARG	THE HARD WAY.	09-00042
	LW, X	RTRANK		09-00043
	B	10Z65		09-00044
10Z64	MW, S	*RTARG, X		09-00045
	BOV	ERLENGTH		09-00046
10Z65	BDR, X	10Z64		09-00047
	MTW, -2	RTARG		09-00048
10Z66	EQU	%		09-00049
	STW, S	MIDLCNT	D(2)*D(3)*...*D(N).	
	MW, S	OUTRCNT	RESULT SIZE =	
	STW, S	RSSIZE	C(1)*...*C(M-1)*D(2)*...*D(N).	
	BAL, L1	ALOCRS	ALLOCATE RESULT DATA BLOCK	
	AW, A	RSRANK	POINT TO LAST RS DIMEN -1	
	MTW, 2	RTARG		
	LW, X	RTRANK	MOVE D(2), ..., D(N) TO RS DIMENS,	
	B	10Z8	IF N>=2.	
10Z7	LW, R	*RTARG, X		
	STW, R	1, A		
	AI, A	-1		
10Z8	BDR, X	10Z7		
	MTW, -2	RTARG		
	MTW, 1	LFARG		
	LW, X	LFRANK	MOVE C(1), ..., C(M-1) TO RS DIMENS,	
	B	10Z10	IF M>=2.	
10Z9	LW, R	*LFARG, X		
	STW, R	1, A		
	AI, A	-1		
10Z10	BDR, X	10Z9		
	MTW, -1	LFARG		
	LW, X	INERCNT		09-00051
	BEZ	FILIDENT	HANDLE INERCNT=0 CASE SPECIALLY	09-00052

```

        LI,A      0
        BAL,LX    XSETUP          SET LFADR FOR INDX LOAD (K1),
        BAL,L1    SETSPEC1        RTADR FOR INDX (K), RSADR SEQ (N).
        B         *L2              RETURN
        PAGE

*
*
*  M I X E D   O P   S E T U P   R O U T I N E S
*
*  SET UP ARG PARAMETERS
*
*          SETS UP RANK, SIZE AND TYPE CELLS FOR LEFT (A=0)
*          OR RIGHT (A=1) ARG.  LINK IS LX.
*
SETUPARG EQU      %
        LW,X      LFARG,A          GET ARG PNTR
        LB,T      *X              GET ARG TYPE CODE
        CI,T      LIST            CHECK FOR LIST OR OTHER
        BGE       ERDOMAIN        ILLEGAL ARG TYPES.
        STW,T     LFTYPE,A        COPY
        LI,AI     1              INIT SIZE = 1
        LB,R      *X,AI          GET ARG RANK
        STW,R     LFRANK,A        COPY
        BEZ       4Z2            IF RANK>0,
4Z1      MW,AI     2,X              SIZE = PRODUCT OF DIMENS.
        AI,X      1
        BDR,R     4Z1
4Z2      STW,AI    LFSIZE,A        COPY
        B         0,LX          RETURN
        PAGE

*
*
*  TYPE COMPATIBILITY CHECK
*
*          CALLED WITH ARG TYPES IN LFTYPE/RTTYPE.  IF BOTH
*          ARE CHAR OR BOTH NUMERIC, IT RETURNS BAL+2 WITH
*          HIGHEST TYPE IN RSType AND T REG; IF ONE TYPE IS
*          CHAR AND THE OTHER NUMERIC, IT RETURNS TO BAL+1.
*          LINK IS LX.
*          IF HIGHEST TYPE IS 'ISEQ', TYPE 'INTG' IS SUBSTITUTED.
*
TYCOMPAT EQU      %
        LI,T      CHAR
        CW,T      LFTYPE
        BNE       8Z1
        CW,T      RTTYPE          L=CHAR: TEST R
        BE        8Z3            L=R=CHAR: SET RS=CHAR, RTN BAL+2
        B         0,LX          L=CHAR, R=NUMER: RETURN BAL+1
8Z1      CW,T      RTTYPE          L=NUMER: TEST R
        BNE       8Z2
        B         0,LX          L=NUMER, R=CHAR: RETURN BAL+1
8Z2      LW,T      LFTYPE          L=R=NUMER: SET RS=MAX(L,R)
        LB,T      TCONV,T        SUBSTITUTE INTG FOR ISEQ
        LW,X      RTTYPE
        CB,T      TCONV,X        COMPARE (CONVERTED) TYPES
        BGE       8Z3
        LB,T      TCONV,X        L<R, GET CONVERTED R TYPE
8Z3      STW,T     RSType          TYPES COMPATIBLE
        B         1,LX          RETURN BAL+2
*
TCONV    EQU      %              ISEQ-TO-INTG CONVERT TABLE
        DATA,1   0,LOGI,CHAR,INTG,FLOT,INTG
        BOUND     4
        PAGE

*
*
*  EVALUATE INTEGER SCALAR ARG
*
*          MAKE SURE THAT RTARG IS A SCALAR (OR 1-ELEMENT ARRAY)
*          WITH AN INTEGER VALUE, WHICH IS RETURNED IN AI. LINK
*          IS LZ.
*
INTSCALR EQU      %
        LB,T      *RTARG          GET ARG TYPE
        LW,X      RTARG          GET ARG PNTR
        LI,AI     1
        LB,R      *RTARG,AI      GET ARG RANK

```

2Z1	BEZ	GSCLRVAL, T	IS ARG SCALAR?	
	AI, X	1	NO, MAKE SURE ALL ITS DIMENS	
	CW, AI	1, X	ARE 1, AND LEAVE 'X' SITTING	
	BNE	ERLENGTH	TWO WORDS BELOW 1ST DATA WORD,	
	BDR, R	2Z1	AS IT WOULD BE IN THE SCALAR CASE.	
	B	GSCLRVAL, T	GO TO JUMP TABLE	
	PAGE			
*				
*				
*	ALLOCATE	INTEGER VECTOR RESULT DATA BLOCK		
*				
*		CALLLED WITH SIZE IN RSSIZE, IT ALLOCATES THE DB,		
*		SETS ITS DIMEN, AND SETS UP RSTYPE AND RSRANK.		
*		LINK IS L2. IF RSSIZE=0, IT EXITS THROUGH 'RETURN';		
*		OTHERWISE, IT RETURNS TO CALLER WITH RESULT PNTR		
*		IN 'A' AND SIZE IN 'S'.		
*		'VECTORRS' IS ALTERNATE ENTRY WHICH TAKES TYPE		
*		FROM 'T'.		
	INTVECRS	EQU	%	
		LI, T	INTG	
	VECTORRS	EQU	%	
		ISEQFIX, T	USE TYPE INTG INSTEAD OF ISEQ	
		STW, T	RS TYPE = INTG	
		LI, R	1	
		STW, R	RSRANK	RS RANK = 1 (VECTOR)
		LW, S	RSSIZE	
		BAL, L1	ALOCRS	ALLOCATE RS; PNTR TO 'A'
		LW, S	RSSIZE	GET SIZE
		STW, S	2, A	SET RS DIMEN = SIZE
		BGZ	*L2	SIZE>0: RETURN TO CALLER
		B	*RETURN	SIZE=0: EXIT DRIVER
*				
	NTEMPS	SET	ERROR, X'F', TLOC>28	'TOO MANY TEMPS' U09-0169
	18Z	END	TLOC	U09-0170
				U09-0171

*
*
* E X T E R N A L C O M M U N I C A T I O N
*
*

* DEFINITIONS
*

DEF	AXDRIVER	INDEXED ASSIGNMENT EXEC DRIVER	
DEF	COORDK	K'TH COORDINATE SPEC	
DEF	DBUFEND	DIMENSION BUFFER END	
DEF	DCATEN	DYADIC CATENATE OP ROUTINE	
DEF	DCOMPRES	DYADIC COMPRESS OP ROUTINE	
DEF	DDROP	DYADIC DROP OP ROUTINE	
DEF	DEXPAND	DYADIC EXPAND OP ROUTINE	
DEF	DROTATE	DYADIC ROTATE OP ROUTINE	
DEF	DTAKE	DYADIC TAKE OP ROUTINE	
DEF	DTRANS	DYADIC TRANSPOSE OP ROUTINE	
DEF	EXCHLUPS	EXCHANGE MIDDLE/INNER LOOPS	
DEF	GSCLRVAL	GET INTEGER SCALAR VALUE	
DEF	INDX@	START OF PROCEDURE	
DEF	INDXLDLR	INDEXED LOAD LOGICAL RIGHT	
DEF	INERCNT	INNER LOOP COUNT	
DEF	INNERSTEP	INNER LOOP STEP SIZE	
DEF	MAXDIMEN	MAXIMUM NUMBER OF DIMENSIONS	
DEF	MBUFDIMS	MOVER BUFFER DIMENSIONS TO RS	
DEF	MIDLCNT	MIDDLE LOOP COUNT	
DEF	MIDLSAVE	MIDDLE LOOP SAVE TEMP	
DEF	MIDLSTEP	MIDDLE LOOP STEP SIZE	
DEF	MREVERSE	MONADIC REVERSE OP ROUTINE	
DEF	MRTDIMS	MOVER RT ARG DIMENSIONS TO RS	
DEF	MTRANS	MONADIC TRANSPOSE OP ROUTINE	
DEF	OUTRCNT	OUTER LOOP COUNT	
DEF	OUTRSAVE	OUTER LOOP SAVE TEMP	
DEF	OUTRSTEP	OUTER LOOP STEP SIZE	
DEF	RSLIKRT1	ALOC RESULT LIKE RTARG	
DEF	RSLIKRT2	ALOC RESULT LIKE RTARG	U10-0004
DEF	SETSPEC1	SPECIAL ADR SETUP	
DEF	SETSPEC2	SPECIAL ADR SETUP	
DEF	ST3LUPSN	SET UP LOOP PARAMS, NO ALOC	
DEF	SXDRIVER	SUBSCRIPTED EXPRESSION EXEC DRIVER	
DEF	XSETUP	SET UP ADR FOR INDEXED LOAD	

*
* REFERENCES
*

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS	
REF	ALOCRS	ALLOCATE RESULT DATA BLOCK	
REF	AXRETURN	INDEXED ASSIGNMENT EXEC RETURN	
REF	BITMASK	BIT SELECTION MASK TABLE	
REF	DREF	DE-REF	
ERCOORD	EQU	COORDINATE SPECIFICATION ERROR	
REF	ERRANK	DOMAIN ERROR	
REF	ERDOMAIN	DOMAIN ERROR	
REF	ERINDEX	INDEX RANGE ERROR	
REF	ERLENGTH	LENGTH ERROR	
REF	ERRANK	RANK ERROR	
REF	EXECUTE	EXECUTE XSEG	
REF	FLOT0	FLOATING PT 0.0	
REF	F2I	CONVERT F TO I	
REF	GENLOAD	GEN LOAD BY RSTYPE	
REF	GENLOADT	GEN LOAD TO TEMP	
REF	GXSEGINI	GEN XSEG INITIALIZATION	
REF	GXSTEXEC	GEN XSEG STORE; EXECUTE XSEG	
REF	INBUF	INPUT BUFFER (USED AS TEMP BUFFER)	
REF	INDXTMPS	INDX TEMPS ARE IN WINDOW IN APLUTSI	U10-0006
REF	LBLOCK	LOOP CONTROL BLOCK PNTR	
REF	LFADR	LEFT ARG ADDRESS	
REF	LFARG	LEFT ARG PNTR	
REF	LFLGLADR	LEFT LOGICAL DATA ADR	
REF	LFLGLCNT	LEFT LOGICAL BIT COUNT	
REF	LFRANK	LEFT ARG RANK	
REF	LFSIZE	LEFT ARG SIZE	
REF	LFTYPE	LEFT ARG TYPE	
REF	LOADINST	LOAD INSTRUCTION TABLE, BY TYPE	
REF	LOOPLOC	LOOP LOCATION	
REF	MNOP	MONADIC NO OP ROUTINE	
REF	NILCK	'NIL CHECK' = SCRIPT LIST PNTR	
SCRIPT	EQU	SUBSCRIPT LIST POINTER	

REF	OPER	OPERATOR WORDS
REF	ORGADJ	ORIGIN, ADJUSTED (=1-ORG)
REF	ORIGIN	INDEX ORIGIN VALUE (0 OR 1)
REF	RESULT	RESULT DATA BLOCK POINTER
REF	RETURN	RETURN ADR CELL
REF	RSADR	RESULT ADDRESS
REF	RSRANK	RESULT RANK
REF	RSSIZE	RESULT SIZE
REF	RSTYPE	RESULT TYPE
REF	RTADR	RIGHT ARG ADDRESS
REF	RTARG	RIGHT ARG PNTR
REF	RTRANK	RIGHT ARG RANK
REF	RTSIZE	RIGHT ARG SIZE
REF	RTTYPE	RIGHT ARG TYPE
REF	SETADR	SET UP ADDRESS, SEQUENTIAL
REF	SETADRS1	SET UP ADDRESSES, SEQUENTIAL
REF	SETUPARG	SET UP ARG PARAMS
REF	STORINST	STORE INST TABLE
REF	SXRETURN	SUBSCRIPTED EXP EXEC DRIVER RETURN
REF	SYSTERR	SYSTEM ERROR
REF	TYCOMPAT	TYPE COMPATIBILITY CHECK
REF	VECTORRS	ALLOCATE VECTOR RESULT
REF	XSEGBASE	BASE OF XSEG AREA
	PAGE	

*
*
*
*
*

A S S E M B L Y P A R A M E T E R S

	SYSTEM	SIG5F	
PROGSECT	CSECT	1	
INDX@	RES	0	START OF PROCEDURE

*
*

REGISTERS

IX	EQU	0	EVEN/ODD PAIR
IX1	EQU	1	*
N	EQU	1	XSEG EXECUTION REG
N1	EQU	4	XSEG EXECUTION REG
K1	EQU	4	XSEG EXECUTION REG
N2	EQU	11	XSEG EXECUTION REG
N3	EQU	10	XSEG EXECUTION REG
X	EQU	1	SCRIPT POINTER
T	EQU	2	TYPE REG
NX	EQU	3	COUNT REG
XL	EQU	3	XSEG LOC REG
K	EQU	2	SUBSCRIPT VALUE
KB	EQU	3	B-BLOCK POINTER
KN	EQU	8	LOOP COUNT
KV	EQU	9	ARRAY LOOP OFFSET
KL	EQU	10	LINK REG
A	EQU	4	ARG ADR/INDEX
LZ	EQU	4	INDEX LINK REG
LX	EQU	5	INDEX LINK REG
OP	EQU	6	OP CODE REG
AF	EQU	6	ACCUM FOR FLOT VALUES
AF1	EQU	7	*
LX7	EQU	7	INDEX LINK REG
AI	EQU	7	ACCUM FOR LOGL/CHAR/INTG VALUES
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS
R	EQU	8	GENERAL WORK REG
BI	EQU	9	2ND ACCUM FOR L/C/I VALUES
S	EQU	11	SIZE
L3	EQU	12	LINK REG
L2	EQU	13	LINK REG
L1	EQU	14	LINK REG
R15	EQU	15	

*
*

ARG TYPE CODES

LOGL	EQU	1	LOGICAL (BIT)
CHAR	EQU	2	CHARACTER (BYTE)
INTG	EQU	3	INTEGER (WORD)
FLOT	EQU	4	FLOATING (DOUBLEWORD)
ISEQ	EQU	5	INDEX SEQUENCE VECTOR
LIST	EQU	6	LIST

*

CODESTRING DESIGNATIONS

```

*
DOPCMPRS EQU      84      DYADIC COMPRESS
DOPCATEN EQU      87      DYADIC CATENATE/LAMINATE
DOPTAKE  EQU     116      DYADIC TAKE
*
*   B-BLOCK STRUCTURE PARAMETERS
*
BN          EQU      6      B BLOCK ITEM SIZE
BSTEPVAL   EQU      0      OFFSET TO STEP VALUE
BINITOFFS  EQU      0      OFFSET TO INITIAL OFFSET VALUE
BINITCNT   EQU      1      OFFSET TO INITIAL COUNT
BNULLCNT   EQU      1      OFFSET TO NULL GROUP COUNT
BSCRIPT    EQU      2      OFFSET TO SCRIPT SAVE LOC
BOFFSET    EQU      2      OFFSET TO OFFSET SAVE LOC
BCOUNT     EQU      3      OFFSET TO COUNT SAVE LOC
BADDEND    EQU      4      OFFSET TO ADDEND VALUE
BNULTEST   EQU      4      OFFSET TO NULL TEST VALUE
BLOOPTYP   EQU      5      OFFSET TO LOOP TYPE CODE
*
SEQLOOP    EQU      0      SEQUENCE LOOP
ARRAYLOOP  EQU     -1      ARRAY LOOP
MIXLOOP    EQU     -2      MIXED SEQUENCE LONP
LOPBIAS    EQU     -2      LOOP TYPE CODE BIAS
*
*   DIMENSION BUFFER PARAMETERS
*
MAXDIMEN   EQU      63      MAXIMUM NUMBER OF DIMENSIONS
DIMENBUF   EQU      XSEGBASE DIMENSION BUFFER START
DBUFEND    EQU      DIMENBUF+MAXDIMEN DIMENSION BUFFER END
*
*   MISCELLANEOUS PARAMETERS
*
INFINITY   EQU      1**19-1 PLUS INFINITY (FOR 'LI')
LAMBIT     EQU      X'40'    LAMINATE BIT
PAGE
*
*
*   P R O C S
*
*
TLOC       SET      0      U10-0008
*
TEMP       CNAME     1
DTEMP      CNAME     2
PROC
DO1        NAME=2
TLOC       SET      TLOC+(TLOC&1) U10-0011
DISP       TLOC      U10-0012
LF         EQU      INDXTMPS+TLOC U10-0013
TLOC       SET      TLOC+NAME     U10-0014
PEND
*
*
EVEN       CNAME     0
ODD        CNAME     1
PROC
LF         EQU      %
ERROR,1,(CF(2)+NAME)&1 'REGISTER HAS WRONG PARITY'
PEND
*
*
EQUAL      CNAME
PROC
LF         EQU      %
ERROR,1,1-(CF(2)=CF(3)) 'REGISTERS MUST BE EQUAL'
PEND
*
*
EXCHANGE   CNAME
OPEN       I,K, GROUP
GROUP      EQU      LFARG, LFTYPE, LFSIZE, LFRANK, LFADR
PROC
LF         EQU      %
I          DO        NUM(AF)
K          SET      SCOR(AF(I), ARGS, TYPES, SIZES, RANKS, ADRS)
ERROR,1,K=0 'UNKNOWN GROUP INDICATOR'
LW,R      GROUP(K)
XW,R      GROUP(K)+1

```

```

        STW, R      GROUP(K)
        FIN
        PEND
        CLOSE      I, K, GROUP
*
*
ISEQFIX  CNAME
        PROC
LF        CI, CF(2)  ISEQ
        BNE        %+2
        LI, CF(2)  INTG
        PEND
        PAGE
*
*
*   XSEG GEN PROCS
*
*
GEN      OPEN      GEN
        CNAME
        OPEN      M, N, MN, I
        PROC
LF      EQU        %
        ERROR, 1, 1-(NUM(CF)=3)  'WRONG NUMBER OF CF ARGS'
M      SET        CF(2)
N      SET        CF(3)
MN      SET        M+N
        ERROR, 1, 1-(NUM(AF)=(M>0)+(N>0))  'WRONG NUMBER OF AF ARGS'
        DO        M>0
I      DO        N
        LW, AF(1)+M+I-1  AF(2)+I-1
        FIN
I      DO        MN*(MN<3)
        STW, AF(1)+I-1  I-1, XL
        ELSE
        LCI        MN
        STM, AF(1)  0, XL
        FIN
I      ELSE
        DO        MN*(MN<3)
        LW, BUF    AF(1)+I-1
        STW, BUF    I-1, XL
        ELSE
        LCI        N
        LM, BUF    AF(1)
        STM, BUF    0, XL
        FIN
        FIN
        AI, XL      MN
        PEND
        CLOSE      M, N, MN, I
        PAGE
*
*
*   TABLE BUILDING PROCS
*
*
TABLE    CNAME
        OPEN      T, N
        PROC
T      SET        %-AF(1)
LF      EQU        T
        DISP      T
        PEND
*
*
ITEM     CNAME
        PROC
N      SET        T+AF(1)-%
        ERROR, 1, N<0  'ITEM OUT OF SEQUENCE'
        RES        N*(N>0)
        DISP      AF(1)
        PEND
        CLOSE      T, N
        PAGE
*
*
*   O P E R A T O R   E X E C U T I O N   D R I V E R S

```

```

*
*
*      USECT      PROGSECT
*
*
*      SUBSCRIPTED EXPRESSION EXECUTION DRIVER
*
*      ENTERED WITH SUBSCRIPT LIST POINTER IN 'SCRIPT' AND
*      POINTER TO EXPRESSION-TO-BE-SUBSCRIPTED IN 'LFARG'.
*      IF NO ERRORS OCCUR, IT RETURNS TO 'SXRETURN' WITH
*      RESULT POINTER IN 'RESULT', AND 'SCRIPT' DE-REFFED.
*      LEFT ARG POINTER WILL HAVE BEEN MOVED TO 'RTARG',
*      WHICH, ALONG WITH 'LBLOCK', STILLS NEEDS TO
*      BE DE-REFFED.
*
*
*      SXDRIVER EQU      %      SUBSCRIPTED EXPRESSION EXEC DRIVER
*      LI,A      SXRETURN      SET UP RETURN ADDRESS
*      STW,A      RETURN
*      EXCHANGE  ARGS
*      BAL,L2     INDEXA
*
*      LB,T      *RTARG
*      CI,T      LIST
*      BGE       ERDOMAIN
*      ISEQFIX,T
*      STW,T     RSTYPE
*
*      ALSELECT EQU      %
*      LW,S      RSSIZE
*      BAL,L1     ALOCRS
*      BAL,LX     MBUFDIMS
*
*      MTSELECT EQU      %
*      LI,X      -1
*      BAL,LX     SETADR
*      BAL,LX     GXSEGINI
*      GEN,0,1    BAL1STSC
*      LI,A      1
*      BAL,LX     XSETUP
*      LB,T      *RTARG
*      AW,R      LOADINST,T
*      GEN,1,0    R
*      LI,A      NXTSCRPT
*      STW,A      LOOPLOC
*      B          GXSTEXEC
*
*      XSEG CODE:
*
*      LCW,N      RSSIZE
*      BAL,L1     1STSCRPT
*      LOAD       RTARG(K)
*      STORE      RESULT(N)
*      BIR,N      NXTSCRPT
*      B          *RETURN
*      PAGE
*
*      INDEXED ASSIGNMENT EXECUTION DRIVER
*
*      ENTERED WITH SUBSCRIPT LIST POINTER IN 'SCRIPT',
*      POINTER TO INDEXED VARIABLE (ON LEFT SIDE OF ASSIGNMENT
*      ARROW) IN 'LFARG', AND POINTER TO THE VALUE-TO-ASSIGN
*      IN 'RTARG'. IF NO ERROR OCCUR, IT RETURNS TO 'AXRETURN'
*      WITH RESULT POINTER (VALUE OF THE 'LFARG' VARIABLE,
*      AFTER THE INDEXED ASSIGNMENT) IN 'RESULT', AND 'SCRIPT'
*      DE-REFFED. 'LBLOCK' STILL NEEDS TO BE DE-REFFED.
*
*
*      AXDRIVER EQU      %      INDEXED ASSIGNMENT EXEC DRIVER
*      LI,A      AXRETURN      SET RETURN ADR
*      STW,A      RETURN
*      EXCHANGE  ARGS
*      BAL,L2     INDEXA
*
*      LI,A      0
*      BAL,LX     SETUPARG
*      LI,A      1

```

```

SUBSCRIPTED EXPRESSION EXEC DRIVER
SET UP RETURN ADDRESS
PUT ARG PNTR IN RTARG.
BUILD LOOP CONTROL BLOCK FROM
SCRIPT LIST; SET SIZE/RANK.

```

```

DISALLOW LIST INDEXING
USE TYPE INTG INSTEAD OF ISEQ
SET TYPE

```

```

ALLOCATE RESULT DATA BLOCK
MOVE DIMENS FROM BUFFER TO RS

```

```

SET RSADR FOR SEQUENTIAL STORE
GEN XSEG INIT; EXIT IF RESULT NULL
GEN:      BAL,L1  1STSCRPT
SET RTADR FOR INDEXED LOAD
GEN INDEXED LOAD
SET TO LOOP BACK TO 'NXTSCRPT'
GEN STORE/LOOP CONTROL; EXECUTE

```

```

U10-0016
U10-0017
U10-0018
U10-0019
U10-0021
U10-0022
U10-0023
U10-0024

```

	BAL, LX	SETUPARG	IND.VAR.	
	BAL, LX	TYCOMPAT	MAKE SURE THE TYPES ARE COMPATIBLE,	
	B	ERDOMAIN	PUT HIGHEST TYPE IN RSTYPE.	
	LW, S	RSSIZE	SET SCRIPT SIZE (NUMBER	
	STW, S	SCRPTCNT	OF SCRIPT VALUES).	
	LI, S	1		
	CW, S	LFSIZE	IF ASSIGN VALUE IS 1-ELMT,	
	BE	14Z2	SKIP RANK/DIMEN CHECKS.	
	LI, AI	1		10-00071
	LW, A	LFARG	SET UP TO COMPARE DIMENS OF ASSIGN	10-00072
	LW, R	LFRANK	VALUE AND SUBSCRIPT STRUCTURE.	10-00073
	AI, R	1	EXCEPT FOR 1'S (WHICH ARE SKIPPED	10-00074
	LCW, K	RSRANK	OVER), THESE DIMENS MUST MATCH.	10-00075
	AI, K	-1		10-00076
14Z10	BDR, R	14Z13	GET A DIMEN FROM ASSIGN VALUE	10-00077
14Z11	BIR, K	14Z12	NO MORE: REMAINING SCRIPT	
	B	14Z2	DIMENS MUST ALL BE 1'S.	10-00079
14Z12	CW, AI	DBUFEND, K		10-00080
	BE	14Z11		10-00081
	B	ERRANK		10-00082
14Z13	LW, S	2, A	(NEXT ASSIGN-VALUE DIMEN)	10-00083
	AI, A	1		10-00084
	CI, S	1	IF IT'S 1, SKIP IT AND GET ANOTHER	10-00085
	BE	14Z10		10-00086
14Z14	BIR, K	14Z15	GET NEXT SCRIPT DIMEN	
	B	ERRANK	(THERE HAD BETTER BE ONE)	10-00088
14Z15	CW, AI	DBUFEND, K	IF IT'S 1, SKIP IT & GET NEXT	10-00089
	BE	14Z14		10-00090
	CW, S	DBUFEND, K	BOTH DIMENS NON 1: THEY	10-00091
	BE	14Z10	ARE REQUIRED TO AGREE.	10-00092
	B	ERLENGTH		10-00093
14Z2	LW, S	SCRPTCNT	IF INDEX NULL, SET RESULT=	10-00061
	BEZ	MNOP	INDEXED VARIABLE.	
	LW, OP	LFTYPE	TEST FOR SPECIAL CASE:	U10-0026
	CW, OP	RTTYPE	ALLOW ONLY IF LFARG IS CONVERTIBLE	U10-0027
	BG	14Z21	TO RTARG'S TYPE,	U10-0028
	LI, OP	ISEQ	RT ARG IS NOT AN ISEQ,	
	CW, OP	RTTYPE		
	BE	14Z21		
	LW, A	RTARG		U10-0029
	LW, R	1, A	AND RTARG IS RE-USABLE AS RESULT	U10-0030
	AI, R	-2	(I.E., ITS REF COUNT IS 2).	U10-0031
	BNEZ	14Z21	NO GOOD	U10-0032
*			GOOD: CONDITIONS FOR THE SPECIAL	U10-0033
*			CASE HAVE BEEN MET. WE'LL NOW	U10-0034
*			USE RTARG FOR THE RESULT.	U10-0035
	LI, OP	-100	SIGNAL TO GEN SPECIAL CODE	U10-0036
	MTW, 1	1, A	CREATE NEW COPY OF RTARG PTR	U10-0037
	STW, A	RESULT	USE IT FOR RESULT PTR	U10-0038
	B	14Z22	JOIN GENERAL CASE	U10-0039
14Z21	EQU	%	GENERAL CASE: ALOC NEW RS BLOCK	U10-0040
	BAL, L1	RSLIKRT2	ALOC RS WITH SHAPE = IND.VAR.SHAPE	
	BAL, LX	MRTDIMS	MOVE IND.VAR. DIMENS TO RESULT	
14Z22	LW, S	RTSIZE	IF IND.VAR. IS NULL,	U10-0042
	BEZ	*RETURN	THEN SO IS RESULT.	
	LI, X	-3		
	BAL, LX	SETADR	ST LFADR FOR SEQ LOAD (N)	
	LI, A	1		
	BAL, LX	XSETUP	SET RTADR FOR INDX LOAD (K)	
	LI, A	2		
	BAL, LX	XSETUP	SET RSADR FOR INDX STORE (K)	
	LI, XL	XSEGBASE	PREPARE TO GEN XSEG CODE	
	LI, S	1		
	CW, S	LFSIZE	IF ASSIGN VALUE IS 1-ELMT,	
	BNE	14Z3		
	LI, A	0	GEN LOAD/CONVERT ASSIGN	
	BAL, L2	GENLOADT	VALUE TO TEMP.	
14Z3	BIR, OP	14Z4	JUMP IF SPECIAL CASE	U10-0044
	STW, XL	LOOPLOC	GENERAL CASE	U10-0045
	GEN, 0, 9	CODE7	GEN K-LOOP CONTROL	
	AWM, XL	-7, XL		
	B	14Z5	JOIN SPECIAL CASE	U10-0047
14Z4	GEN, 0, 1	BAL1STSC	SPECIAL CASE: SIMPLER K-LOOP CONTROL	U10-0048
14Z5	EQU	%	COMMON CODE	U10-0049
	LI, A	0	GEN LOAD/CONVERT OF ASSIGN-VALUE	
	BAL, L1	GENLOAD	ELMT FOR SELECTED POSITION.	
	LW, T	RSTYPE		
	LW, R	STORINST, T	SET UP STORE INST	

	AW,R	RSADR	JUMP IF SPECIAL CASE	U10-0051
	BIR,OP	14Z6		
	STW,R	TEMP		
	GEN,1,4	R, CODE8	GEN N-LOOP CONTROL	
	LW,X	LOOPLOC		
	AWM,X	-1,XL	FILL IN BRANCH ADR	
	AWM,XL	6,X		
	LI,A	1	GEN LOAD/CONVERT OF IND.VAR. TO FILL IN NON-SELECTED ELMTS.	
	BAL,L1	GENLOAD		
	LW,R	TEMP		
	GEN,1,1	R, CODE11	GEN STORE/N-LOOP CONTROL 3	
	LW,X	LOOPLOC		
	AWM,X	-1,XL	FILL IN BRACH ADR	
*				
	LI,R	-1	PREPARE FOR XSEG EXECUTION:	
	STW,R	KTEMP	'OLD K' = -1,	
	B	14Z7	JOIN SPECIAL CASE	U10-0053
14Z6	GEN,1,3	R, CODE13	SPECIAL CASE: SIMPLER STORE CONTROL	U10-0054
14Z7	EQU	%	COMMON CODE	U10-0055
	LCW,N	LFSIZE	N = ASSIGN-VALUE INDEX	
	LW,N2	SCRPTCNT	N2 = NUMBER OF SCRIPT VALUES	
	BIR,OP	XSEGBASE	IF SPECIAL CASE, DONT ALLOW BREAKS	U10-0057
	B	EXECUTE		
*				
KTEMP	EQU	TEMP		
SCRPTCNT	EQU	INERCNT		
*				
*				
*				
*				
*				
		XSEG CODE:		
	LOAD/CNV	LFARG	(ONLY IF LFARG IS	U10-0059
	STORE	LFTEMP	1-ELEMENT).	
CODE7	BAL,L1	1STSCRPT	GET INDEX (K) OF 1ST SELECTED ELMT	
	CW,K	KTEMP	IF NEW INDEX <= PREVIOUS	
	BLE	0 (LOOP1)	ONE, GO BACK & STORE IN IT.	
	XW,K	KTEMP	SWAP W/INDEX OF PREVIOUS SEL.ELMT	
	AI,K	1	IF THERE'S A GAP,	
	CW,K	KTEMP		
	BNE	0 (LOOP2)	GO FILL IN WITH IND.VAR. VALUES.	
	CW,K	RSSIZE	IF THIS IS THE FINAL FILL-OUT	
	BGE	*RETURN	PASS, STOP WHEN FULL.	
*LOOP1	LOAD/CNV	LFARG(N)	FETCH NEXT ELMT OF ASSIGN-VALUE	
*	STORE	RESULT(K)	STORE ASSIGN-VALUE ELMT	
CODE8	AI,N	1	BUMP ASSIGN-VALUE INDEX	
	BDR,N2	NXTSCRPT	GET NEXT INDEX	
	LW,K	RSSIZE	NO MORE INDECES, SET TO FILL	
	B	3 (XW,K)	OUT RESULT W/ IND.VAR. ELMTS.	
*LOOP2	LOAD/CNV	RTARG(K)	FETCH IND.VAR. ELMT TO FILL INTO	
*	STORE	RESULT(K)	NON-SELECTED RESULT POSITION.	
CODE11	B	4 (AI,K)	KEEP FILLING 'TIL WE HIT SELECTED	
*			POSITION, OR FILL UP RESULT.	
*				U10-0061
*				U10-0062
*				U10-0063
*				U10-0064
*				U10-0065
*	LOAD/CNV	LFARG	(ONLY IF LFARG IS	U10-0066
*	STORE	LFTEMP	1-ELEMENT).	U10-0067
*	BAL,L1	1STSCRPT	GET 1ST K-VALUE	U10-0068
*	LOAD/CNV	LFARG(N)	GET NEXT ELEMENT OF ASSIGN VALUE	U10-0069
*	STORE	RESULT(K)	STORE INTO SELECTED RESULT POSITION	U10-0070
CODE13	AI,N	1	BUMP LOAD INDEX	U10-0071
	BDR,N2	NXTSCRPT	GET NEXT K-VALUE	U10-0072
	B	*RETURN	EXIT WHEN ALL K-VALUES USED	
	PAGE			
*				
*				
*	M O N A D I C	I N D E X I N G	O P	R O U T I N E S
*				
*				
MTRANS	EQU	%	MONADIC TRANSPOSE	
	LI,A	1		
	BAL,LX	SETUPARG	SET UP RT ARG PARAMS	
	LW,S	RTRANK	GET ARG RANK	
	CI,S	1	IF ARG IS A SCALAR OR VECTOR,	
	BLE	MNOP	RESULT = ARG.	
	BAL,L2	INDEXB	RANK>=2: ALOC LOOP CONTROL BLOCK	
	BAL,L1	RSLIKERT	ALOC RESULT DB LIKE ARG DB	
	LW,KB	OUTRBLOK	PREPARE TO SET UP LOOP CONTROLS	

	LI, OP	SEQLOOP	LOOP TYPE = SEQUENTIAL
	LI, S	1	FIRST STEP VAL = 1
	LW, A	RESULT	INCREASING DIMEN INDEX
	LW, X	RTRANK	DECREASING DIMEN INDEX AND COUNTER
	MTW, 1	RTARG	POINT TO ARG DIMENS
*			FOR LOOP(K) ...
22Z1	STW, OP	BLOOPTYP, KB	TYPE = SEQUENTIAL
	STW, S	BSTEPVAL, KB	STEP = D(N)*D(N-1)*...*D(N+2-K)
	LW, R	*RTARG, X	
	STW, R	BINITCNT, KB	COUNT = D(N+1-K)
	STW, R	2, A	SET K'TH RESULT DIMEN = D(N+1-K)
	MW, S	*RTARG, X	UPDATE STEP VALUE: MULT BY D(N+1-K)
	AI, A	1	BUMP INCREASING DIMEN INDEX
	AI, KB	+BN	BUMP TO NEXT HIGHER LOOP BLOCK
	BDR, X	22Z1	BUMP DECREASING DIMEN INDEX, COUNT
	MTW, -1	RTARG	CORRECT ARG POINTER
	B	MTSELECT	USE LOOP CONTROL BLOCK TO PICK
			RESULT ELEMENTS IN CORRECT ORDER.
*			
	PAGE		
*			
*			
MREVERSE	EQU	%	MONADIC REVERSE
	LB, T	*RTARG	GET ARGUMENT TYPE
	CI, T	LIST	
	BGE	ERDOMAIN	DISALLOW LIST OR ABOVE
	CI, T	ISEQ	IS IT AN ISEQ
	BNE	23Z1	NO, FORGET IT.
	LB, S	OPER+1	CHECK FOR COORDINATE SPEC.
	BDR, S	ERCOORD	ERROR IF GREATER THAN 1
	LI, S	3	YES, CREATE AN ISEQ RESULT
	BAL, LX7	ALOCHNW	ALOC DATA BLOCK FOR RESULT
	STW, A	RESULT	SAVE POINTER
	LI, R	ISEQ**8+1	TYPE=ISEQ, RANK=1
	STH, R	*RESULT	
	LW, X	RTARG	GET ARG POINTER
	LW, S	2, X	COMPUTE RESULT BASE VALUE =
	AI, S	1	A+B*(N+1) WHERE A/B ARE
	ODD, S		
	MW, S	4, X	THE ARG BASE/STEP VALUES.
	AW, S	3, X	
	LCW, S+1	4, X	RESULT STEP = -B
	LW, S-1	2, X	RESULT SIZE = N
	LCI	3	STORE RESULT SIZE/BASE/STEP
	STM, S-1	2, A	IN RESULT DATA BLOCK.
	B	*RETURN	EXIT
23Z1	EQU	%	ARG NOT ISEQ: PROCEED NORMALLY
	BAL, L2	SET3LUPS	SET UP LOOP PARAMETERS
	LCW, R	MIDLSTEP	MIDDLE LOOP STEP =
	STW, R	MIDLSTEP	-D(K+1)*...*D(N).
	AW, R	OUTRSTEP	FVALUE =
	STW, R	FVALUE	(D(K)-1)*(D(K+1)*...*D(N).
INDXOP1	LI, A	1	SET UP RT ADR FOR INDEXED LOAD
	BAL, LX	XSETUP	
	LI, X	-1	SET UP RS ADR FOR SEQUENTIAL STORE
	BAL, LX	SETADR	
	LI, XL	XSEGBASE	PREPARE TO GEN XSEG
	GEN, 0, 7	CODE1	GEN OUTER/MIDDLE LOOP INIT CODE
	STW, XL	LOOPLOC	INNER LOOP LOC HERE
	LB, T	*RTARG	
	LW, R	LOADINST, T	GEN LOAD OF RIGHT ARG,
	AW, R	RTADR	
	LW, R+1	STORINST, T	STORE TO RESULT,
	AW, R+1	RSADR	
	GEN, 2, 1	R, AIN1INST	INCREMENT OF STORE INDEX.
	GEN, 0, 9	CODE3	GEN INNER LOOP CONTROL
	LW, R	LOOPLOC	
	AWM, R	-8, XL	
	B	EXECUTE	EXECUTE XSEG
*			
*			
*			
	XSEG CODE:		
CODE1	LCW, N	RSSIZE	0 INIT STORE INDEX
	LW, K	FVALUE	1 INIT INDEX
	LW, N1	OUTRCNT	2 INIT OUTER LOOP
	STW, K	OUTRSAVE	3 SAVE INDEX FOR OUTER LOOP
	LW, N2	MIDLCNT	4 INIT MIDDLE LOOP
	STW, K	MIDLSAVE	5 SAVE INDEX FOR MIDDLE LOOP
	LW, N3	INERCNT	6 INIT INNER LOOP

*	LOAD	RTARG(K)	LOAD ARG
*	STORE	RESULT(N)	STORE RESULT
AIN1INST	AI,N	1	BUMP STORE INDEX
CODE3	AW,K	INNERSTEP	BUMP BY INNER LOOP STEP
	BDR,N3	0 (INNER LOOP LOC)	COUNT INNER LOOP
CODE5	LW,K	MIDLSAVE	RESTORE INDEX FOR MIDDLE LOOP
	AW,K	MIDLSTEP	BUMP BY MIDDLE LOOP STEP
	BDR,N2	XSEGBASE+5	COUNT MIDDLE LOOP
	LW,K	OUTRSAVE	RESTORE INDEX FOR OUTER LOOP
	AW,K	OUTRSTEP	BUMP BY OUTER LOOP STEP
	BDR,N1	XSEGBASE+3	COUNT OUTER LOOP
	B	*RETURN	EXIT
	PAGE		

*
*
*
*
*

D Y A D I C I N D E X I N G O P R O U T I N E S

DROTATE	EQU	%	DYADIC ROTATE	
	BAL,LX	LISTCHK	CHECK ARGUMENTS FOR LIST	
	BAL,L2	SET3LUPS	SET UP LOOP PARAMS	
	BAL,LX	EXCHLUPS	EXCHANGE INNER/MIDDLE LOOPS	
	LI,R	0	INIT FIXED PART VALUE	
	STW,R	FVALUE		
	LI,A	1		
	BAL,LX	XSETUP	SET RIGHT ARG	
	LI,A	2	AND RESULT	
	BAL,LX	XSETUP	FOR INDICIAL ACCESS.	
	LI,XL	XSEGBASE	PREPARE TO GEN CODE	
	GEN,0,7	CODE1	GEN OUTER/MIDDLE LOOP INIT/SAVE	
	MTW,-2	XSEGBASE	CHANGE 'RSSIZE' TO 'LFSIZE'	
	LI,X	1	THERE ARE TWO CASES OF ROTATE,	
	LB,R	*LFARG,X	DEPENDING ON LEFT ARG RANK:	
	BGZ	13Z3	>0: GENERAL CASE	
	LW,X	LFARG	=0 (SCALAR): MUCH LIKE 'REVERSE'	
	LB,T	*LFARG		
	BAL,LZ	GSCLRVAL,T	GET VALUE OF LEFT ARG (AS INTG)	
13Z0	BGEZ	13Z1		
	LI,AI-1	-1	EXTEND SIGN	
	B	13Z2		
13Z1	LI,AI-1	0		
13Z2	DW,AI-1	INERCNT	COMPUTE MOD(LFARG,D(K))	
	AI,AI-1	0		
	BGEZ	13Z21	MAKE SURE SHIFT>=0	
	AW,AI-1	INERCNT		
13Z21	STW,AI-1	TESTVAL	= SPECIAL MIDL LOOP VALUE	
	LW,AI	TESTVAL		
	MW,AI	INNERSTEP	INIT DELTA =	
	STW,AI	IDELTA	SHIFT*D(K+1)*...*D(N).	
	GEN,0,2	CODE4	GEN COPY IDELTA TO DELTA	
	B	13Z9	JOIN GENERAL CASE	
13Z3	LI,A	0	GENERAL CASE	
	STW,XL	TEMP		10-00003
	BAL,LX	SETUPARG	SET UP LFARG PARAMS.	
	LW,XL	TEMP		10-00005
	AI,X	1	X=BLOCK POINTER+RANK+1	
	AI,AI	-1	CHECK IF LEFT ARGUMENT HAS 1 ELEM.	
	BNEZ	13Z31	NO-GO TO NORMAL PROCESSING	
	LI,S	1	YES-FETCH THE SOLITARY ARRAY	
	BAL,LZ	GARAYVAL,T	VALUE AND TREAT AS SCALAR.	
	BAL,R15	SYSTERR	THIS POINT SHOULDN'T BE REACHED.	
	B	13Z0		
13Z31	SW,X	LFARG	X=LFARG+1	
	CW,X	RTRANK	NORMAL CASE,CHECK FOR RANK	
	BNE	ERRANK	CONFORMABILITY.	
	LW,S	LFARG	SAVE LFARG PTR	
	MTW,1	RTARG	POINT TO 1ST DIMEN -1	
13Z4	CW,X	COORDK	IF WE'VE GOTTEN TO K'TH COORD,	
	BNE	13Z6		
	MTW,1	LFARG	LINE UP LF/RT DIMEN PTRS.	
13Z5	BDR,X	13Z4		
	B	13Z7		
13Z6	LW,R	*LFARG,X	MAKE SURE (ALL) LF DIMENS =	
	CW,R	*RTARG,X	RT DIMENS (K'TH ONE DELETED).	
	BE	13Z5		
	STW,S	LFARG	NO GOOD: RESTORE PNTRS	
	MTW,-1	RTARG		
	B	ERLENGTH	AND SIGNAL ERROR.	

13Z7	STW,S	LFARG	OK: RESTORE PNTRS
	MTW,-1	RTARG	
	LI,X	-3	SET LEFT ARG FOR
	BAL,LX	SETADR	SEQUENTIAL LOAD.
	LI,T	INTG	
	STW,T	RSTYPE	
	LI,A	0	
	BAL,L1	GENLOAD	GEN LOAD LEFT(N) IN INTG TYPE
	LW,T	LFTYPE	
	GEN,0,3	CODE2	GEN LCC TEST CODE
13Z9	STW,XL	LOOPLOC	SAVE INNER LOOP LOC
	GEN,0,1	AWKDELTA	GEN ADD OF DELTA
	LW,T	RTTYPE	
	LW,A	LOADINST,T	GEN LOAD RIGHT(K),
	AW,A	RTADR	
	LW,A+1	SWKDELTA	SUBTRACT OF DELTA,
	ISEQFIX,T		
	LW,A+2	STORINST,T	STORE RESULT(K),
	AW,A+2	RSADR	
	GEN,3,2	A, CODE6	AND LOOP CONTROL CODE.
	GEN,0,7	CODE5	
	B	EXECUTE	
*			
*			
*			
*			
*			
	SET OUTER LOOP		
	SET MIDDLE LOOP		
	LOAD/CNV LFARG(N)		GET A SHIFT COUNT
CODE2	BGEZ	ROTATE1	EXTEND SIGN(AI) INTO AI-1
	LI,AI-1	-1	
	B	ROTATE2	
CODE4	LW,R	IDELTA	RE-INIT DELTA (ONLY FOR
	STW,R	DELTA	ROTATE W/SCALAR LFARG).
ROTATE1	LI,AI-1	0	*(THIS CODE NOT IN XSEG)
ROTATE2	AI,N	1	* BUMP LFARG INDEX
	DW,AI-1	INERCNT	* COMPUTE J=MOD(SHIFT COUNT, D(K))
	AI,AI-1	0	*
	BGEZ	13Z91	* MAKE SURE J>=0
	AW,AI-1	INERCNT	*
13Z91	STW,AI-1	TESTVAL	* J
	LW,AI	TESTVAL	* DELTA (DISTANCE BETWEEN UNSHIFTED
	MW,AI	INERSTEP	* AND SHIFTED INDEX VALUES)
	STW,AI	DELTA	* = J*D(K+1)*D(K+2)*...*D(N).
	LW,N3	INERCNT	* RUN INNER LOOP D(K) TIMES
ROTATE3	CW,N3	TESTVAL	* UPON STARTING THE D(K)-J+1 'TH
	BNE	*LOOPLOC	* ITERATION OF INNER LOOP,
	LCW,R	OUTRSTEP	* DECR DELTA BY D(K)*D(K+1)*...*D(N)
	AWM,R	DELTA	* TO KEEP K'TH COORD INDEX
	B	*LOOPLOC	* IN BOUNDS.
*LOOPLOC	EQU	%	
AWKDELTA	AW,K	DELTA	SHIFT INDEX
*	LOAD	RTARG(K)	FETCH, SHIFTEDLY
SWKDELTA	SW,K	DELTA	UNSHIFT INDEX
*	STORE	RESULT(K)	STORE, UNSHIFTEDLY
CODE6	AW,K	INERSTEP	BUMP BY INNER LOOP STEP
	BDR,N3	ROTATE3	COUNT INNER LOOP
*	...		COUNT MIDDLE/OUTER LOOPS
*	...		EXIT
*			
DELTA	TEMP		DISTANCE 'TWEEN RTARG/RESULT ELMTS
IDELTA	TEMP		INITIAL VALUE FOR DELTA
TESTVAL	TEMP		LOOP COUNT SPECIAL TEST VAL
	PAGE		
*			
*			
DTRANS	EQU	%	DYADIC TRANSPOSE
	BAL,LX	LISTCHK	CHECK ARGUMENTS FOR LIST
	LI,X	1	
	CB,X	*LFARG,X	LEFT ARG MUST BE VECTOR ...
	BNE	ERRANK	
	LB,S	*RTARG,X	... OF LENGTH EQUAL RIGHT RANK.
	LW,X	LFARG	
	CW,S	2,X	
	BNE	ERLENGTH	
	STW,S	LFSIZE	
	LI,R	0	
	STW,R	RSRANK	INIT RS RANK = 0
	LI,R	INFINITY	

```

11Z1      LCW,X      LFSIZE
          BEZ        MNOF
          STW,R      DBUFEND,X
          BIR,X      11Z1
          BAL,LZ     TRANSVAL
*
*
          B          11Z21
          AW,AI      ORGADJ
          BLEZ       ERDOMAIN
          CW,AI      RSRANK
          BLE        11Z2
          STW,AI     RSRANK
11Z2      SW,AI      LFSIZE
          BGZ       ERDOMAIN
          LW,R       1,NX
          AI,NX      -1
          CW,R       DBUFEND-1,AI
          BGE       *L1
          STW,R      DBUFEND-1,AI
          B          *L1
*
11Z21     LW,X       RSRANK
*
          LI,A       DBUFEND-1
          SW,A       LFSIZE
          LI,NX      DBUFEND-1
          SW,NX      RSRANK
          LI,S       1
11Z3      LW,R       *A,X
          CI,R       INFINITY
          BE         ERDOMAIN
          STW,R      *NX,X
          MW,S       R
          BDR,X      11Z3
          STW,S      RSSIZE
          LB,T       *RTARG
          ISEQFIX,T
          STW,T      RSTYPE
*
*
          NOW SET UP THE LOOP CONTROL BLOCK
*
          LW,S       RSRANK
          BAL,L2     INDEXB
          LW,KB      OUTRBLOK
          LI,R       0
          LCW,X      RSRANK
11Z4      LW,S       DBUFEND,X
          STW,S      BINITCNT,KB
          STW,R      BSTEPVAL,KB
          EQUAL,0,SEQLOOP
          STW,R      BLOOPTYP,KB
          AI,KB      BN
          BIR,X      11Z4
          LI,BI      1
          BAL,LZ     TRANSVAL
*
          B          ELSELECT
          SW,AI      ORIGIN
          MI,AI      BN
          EQUAL,BSTEPVAL,0
          AWM,BI     *OUTRBLOK,AI
          MW,BI      1,NX
          AI,NX      -1
          B          *L1
*
*
          TRANSVAL EQU %
          LW,NX      RTARG
          AW,NX      LFSIZE (=RTRANK) NX = PTR TO LAST RT DIMEN -1
          LW,X       LFARG
          AI,X       2
          LB,T       *LFARG
          LW,S       LFSIZE
          B          GARAYVAL,T
          PAGE
*
*
          NO-OP IF RESULT SCALAR
          INIT RS DIMENS = +INF
          SET UP NX= PTR TO LAST RT DIMEN -1,
            AND INIT 'GARAYVAL' TO FETCH
            LEFT ARG VALUES.
            (DONE)
          CONVERT NEW LFARG VAL TO ORIGIN 1
          RANGE CHECK: 1<=VAL<=LFSIZE
            RSRANK=MAX(RSRANK, VAL)
          FINISH RANGE CHECK, PREPARE TO
            INDEX INTO DIMENBUF.
          GET CURRENT RT DIMEN
          DECR DIMEN PTR
          RSDIMEN(VAL) = MIN(RSDIMEN(VAL),
            RTDIMEN).
          GO FETCH NEXT VALUE FROM LFARG
          LF ARG SCAN DONE. NOW MAKE SURE
            THE VALUES FORMED A DENSE SET,
            AND COMPUTE RESULT SIZE.
          IF D(I)=INF, THEN THERE WAS
            A GAP IN LF VALUES.
          MOVE DIMENS TO END OF DIMENBUF
          SIZE = D(1)*...*D(N)
          STORE RS SIZE
          USE TYPE INTG INSTEAD OF ISEQ
          RESULT TYPE = RIGHT TYPE
          NOW SET UP THE LOOP CONTROL BLOCK
          GET LBLOCK BIG ENOUGH FOR
            N (=RSRANK) B-BLOCKS.
          INITIALIZE B-BLOCKS:
            COUNT(J) = D(J)
            STEP(J) = 0
            TYPE(J) = SEQ
          INIT WEIGHT = 1
          SET NX= PNTR TO D(N) -1; INIT
            'GARAYVAL' TO GET LFARG VALUES.
            (DONE: ALOC RS, DO SELECTION)
          CONVERT NEW LF VALUE TO ORIGIN 0
          COMPUTE INDEX TO B-BLOCK(VAL),
            AT BSTEPVAL POSITION.
          ADD WEIGHT TO STEP(VAL)
          WEIGHT = D(K)*...*D(N)
          DECR D(K) PNTR
          GO GET NEXT VALUE
          X = PTR TO LEFT ARG +RANK+1
          T = LEFT TYPE
          S = LEFT SIZE
          BEGIN FETCHING LEFT VALUES; RETURN

```

DEXPAND	EQU	%	DYADIC EXPAND
	LI, OP	DOPCMPRS+1	ALMOST LIKE COMPRESS
DCOMPRES	EQU	%	DYADIC COMPRESS
	BAL, LX	LISTCHK	CHECK ARGUMENTS FOR LIST
	LI, A	0	SET UP LEFT ARG RANK/SIZE/TYPE
	BAL, LX	SETUPARG	
	LI, X	1	
	CW, X	LFSIZE	LEFT ARG MUST BE EITHER
	BE	15Z0	A 1-ELEMENT ENTITY
	CW, X	LFRANK	OR A VECTOR.
	BNE	ERRANK	
15Z0	BAL, L2	ST3LUPSN	SET UP LOOP PARAMS FROM RTARG, BUT
*			DON'T ALOC RESULT YET.
	LI, NX	0	SET UP FOR SWEEP THRU
	LW, S	LFSIZE	LEFT ARG VALUES.
	BEZ	15Z1	IF IT'S NULL, SKIP IT (SUM=0)
	LW, T	LFTYPE	TYPE
	LW, X	LFARG	
	AW, X	LFRANK	
	AI, X	1	DATA PNTR
	STW, OP	OPTEMP	SAVE OP VALUE
	BAL, LZ	GARAYVAL, T	GET 1ST
	B	15Z01	(DONE)
	AW, NX	AI	ACCUMULATE SUM OF VALUES
	CI, AI	-2	NEXT: MAKE SURE IT'S 0 OR 1
	BAZ	*L1	GET NEXT
	B	ERDOMAIN	
15Z01	LW, OP	OPTEMP	RESTORE OP VALUE
15Z1	LI, S	1	DONE: NX = SUM OF LFARG VALUES
	CW, S	RTSIZE	IS RTARG A 1-ELEMENT THING ?
	BNE	15Z2	NO
	EXU	OPTBL1, OP	YES, SET RESULT SIZE
	STW, NX	RSSIZE	= SUM (COMPRESS), OR
	LW, T	RTTYPE	= LEFT SIZE (EXPAND).
	BAL, L2	VECTORRS	ALOC RS = VECTOR
	B	15Z9	GO GEN XSEG CODE
15Z2	CW, S	LFSIZE	RTARG NORMAL: IS LFARG
	BNE	15Z6, OP	A 1-ELEMENT THING ?
	B	15Z3, NX	YES, WHAT IS ITS VALUE ?
15Z3	B	15Z4, OP	0: SPECIAL CASE FOR COMPRESS
	B	MNOP	1: RESULT = RTARG
15Z4	TABLE	DOPCMPRS	LFARG = 0 (1-ELEMENT)
	B	15Z8	COMPRESS: TREAT AS IF LFSIZE=D(K)
15Z5	CW, NX	MIDLCNT	EXPAND: MAKE SURE THAT
	BNE	ERLENGTH	D(K) = +/(LEFT ARG)
	LW, NX	LFSIZE	AND SET M = LEFT SIZE.
	B	15Z8	
15Z6	TABLE	DOPCMPRS	BOTH ARGS NORMAL. WHAT OP ?
	B	15Z7	COMPRESS
	B	15Z5	EXPAND
15Z7	LW, S	LFSIZE	COMPRESS: MAKE SURE THAT
	CW, S	MIDLCNT	D(K) = LEFT SIZE
	BNE	ERLENGTH	AND USE M = +/(LEFT ARG).
15Z8	SW, NX	MIDLCNT	IF M=D(K),
	BEZ	MNOP	RESULT= RTARG.
	STW, NX	DELTA	SAVE M-D(K)
	LW, L2	RTSIZE	SAVE ACTUAL ARG SIZE
	MW, NX	OUTRCNT	RESULT SIZE DIFFERS FROM RT SIZE BY
	MW, NX	INERCNT	$D(1) \dots D(K-1) * (M-D(K)) * D(K+1) \dots D(N)$.
	AWM, NX	RTSIZE	
	BAL, L1	RSLIKERT	ALOC ARG LIKE RTARG (EXCEPT
	STW, L2	RTSIZE	RESTORE ARG SIZE
	BAL, LX	MRTDIMS	FOR SIZE); COPY RT DIMENS.
	LW, K	COORDK	CHANGE K'TH DIMEN OF RESULT
	AI, K	1	FROM D(K) TO M BY ADDING
	LW, R	DELTA	D(K)-M TO IT.
	AWM, R	*RESULT, K	
15Z9	BAL, L1	SETN1KN	SET UP LFADR FOR SEQ ACCESS (N1),
*			RTADR FOR IND ACCESS (K),
*			AND RSADR FOR SEQ ACCESS (N).
	BAL, LX	GXSEGINI	INIT XSEG; EXIT IF RESULT NULL
	GEN, 0, 4	CODE12	GEN LOOP INIT CODE
	LI, T	LOGL	SET TO CONVERT LFARG TO LOGL
	STW, T	RSTYPE	
	LI, A	0	GEN LOAD/CONVERT LEFT ARG ELMT
	BAL, L1	GENLOAD	
	GEN, 0, 2	CODE14	GEN TEST/INNER LOOP INIT
	STW, XL	LOOPLOC	REMEMBER INNER LOOP LOC

	LW, T	RTTYPE	
	LW, R	LOADINST, T	GEN LOAD (RIGHT),
	AW, R	RTADR	
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ
	LW, R+1	STORINST, T	STORE,
	AW, R+1	RSADR	
	GEN, 2, 2	R, CODE15	AND LOOP CONTROL CODE.
	LI, S	1	
	LW, OP	OPTEMP	RESTORE OP CODE AGAIN
	CW, S	RTSIZE	IF RTARG IS 1-ELEMENT,
	BNE	15Z10, OP	MODIFY CODE TO EXCLUDE
	MTW, 1	-2, XL	K-INCREMENTS.
	MTW, 1	-6, XL	
15Z10	B	15Z10, OP	
	TABLE	DOPCMPRS	
	B	15Z11	COMPRESS: XSEG COMPLETED
	LI, AF1	X'1FFFF'	EXPAND: CHANGE TEST INST
	LW, AF	XL	ADR FROM COMPRES2 TO HERE.
	STS, AF	-6, XL	
	LW, R	CODE17	GEN INNER LOOP
	LW, T	RTTYPE	WHICH STORES NULL ELEMENTS
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ
	LW, R+1	NULLINST, T	IN RESULT.
	LW, R+2	STORINST, T	
	AW, R+2	RSADR	
	GEN, 3, 2	R, CODE18	
*			
15Z11	LW, R	LFLGLADR	FETCH LOGL PARAMS FOR USE TO
	LW, R+1	LFLGLCNT	RE-INIT LOGL.
	B	EXECUTE	EXECUTE XSEG
*			
*			
OPTBL1	TABLE	DOPCMPRS	EXU TABLE - BY OP:
	NOP		COMPRESS
	LW, NX	LFSIZE	EXPAND
*			
OPTEMP	TEMP		TEMP FOR OP CODE
*			
*			
NULLINST	TABLE	LOGL	LOAD NULL INST - BY TYPE:
	LI, AI	0	LOGL
	LI, AI	' '	CHAR
	LI, AI	0	INTG
	LD, AF	FLOT0	FLOT
*			
*			
*			
*			
		XSEG CODE:	
*			
CODE12	LCW, N	RSSIZE	0 INIT STORE INDEX
	LI, K	0	1 INIT RTARG FETCH INDEX
	STW, R	LFLGLADR	2 RESTORE INITIAL LOGL PARAMS
	STW, R+1	LFLGLCNT	3
	LCW, N1	LFSIZE	4 INIT MIDDLE LOOP
*	LOAD/CNV	LFARG(N1)	5 LOAD LFARG VALUE (0 OR 1)
CODE14	BEZ	COMPRES2	IF ZERO, GO TO SKIP (NULL-FILL) CODE
	LW, N2	INERCNT	ONE: INIT INNER LOOP
*LOOPLOC	LOAD	RTARG(K)	COPY RTARG ELMTS TO RESULT
*	STORE	RESULT(N)	
CODE15	BIR, N	COMPRES1	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
COMPRES1	AI, K	1	* BUMP FETCH INDEX
	BDR, N2	*LOOPLOC	* COUNT INNER LOOP
	BIR, N1	XSEGBASE+5	* COUNT MIDDLE LOOP
	B	EXPAND3	* RE-START IF APPROPRIATE
*			
COMPRES2	AW, K	INERCNT	* ZERO: SKIP OVER INNER LOOP
	BIR, N1	XSEGBASE+5	* COUNT MIDDLE LOOP
	B	EXPAND3	* RE-START IF APPROPRIATE
*			
	...OR...		
CODE17	LW, N2	INERCNT	ZERO: INNER LOOP ...
*LOOP4	LOAD	NULL	COPIES NULLS
*	STORE	RESULT(N)	TO RESULT.
CODE18	BIR, N	EXPAND2	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
EXPAND2	BDR, N2	-4, XL (LOOP4)*	COUNT INNER LOOP
	BIR, N1	XSEGBASE+5	COUNT MIDDLE LOOP
EXPAND3	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.

	B PAGE	XSEGBASE+2	* RE-START	
*				
DTAKE	EQU	%	DYADIC TAKE	
DDROP	EQU	%	DYADIC DROP	
	STW, OP	OPTEMP	SAVE OP CODE	
	LI, A	1	SET UP RIGHT ARG'S RANK/SIZE/TYPE,	
	BAL, LX	SETUPARG	CONVERTING ISEQ TO INTG VECTOR.	
	LW, X	LFARG	GET LFARG POINTER	
	LI, NX	1		
	CB, NX	*LFARG, NX	LEFT ARG MUST BE EITHER	
	BL	ERRANK		
	BG	17Z0	A SCALAR,	
	AI, X	1	OR A VECTOR OF LENGTH	
	LW, NX	1, X	= RIGHT ARG RANK.	
17Z0	STW, NX	LFSIZE	SAVE LEFT ARG SIZE	10-00008
	LI, R	=1	IF RT ARG SCALAR, SET 'ARGDIMPT'	10-00009
	LW, L3	RTRANK	TO POINT TO '1' AND L3=0;	U10-0074
	BEZ	17Z01	OTHERWISE, 'ARGDIMPT' POINTS	10-00011
*			TO ARG DIMENS, L3=-1 TO BUMP IT.	U10-0076
*	CW, NX	RTRANK	RT ARG NOT SCALAR: LF ARG SIZE	10-00013
			MUST AGREE.	10-00014
	BNE	ERLENGTH		
	LW, R	RTARG	SET UP TO ACCESS RT ARG	
	AW, R	RTRANK	DIMENS D(N), ..., D(1).	10-00016
	AI, R	1		10-00017
	LI, L3	-1	..TO BUMP DIMEN PTR	10-00018
17Z01	STW, R	ARGDIMPT		10-00019
	LB, T	*LFARG		
	LI, BI	1	INIT RS SIZE	
	AI, X	1	POINT TO LF DATA -1	
	LW, S	LFSIZE	GET SIZE OF LF ARG	10-00021
	BNEZ	17Z1, OP	ENTER TAKE/DROP LOOP UNLESS LFSIZE=	
	B	MNOP	=RTRANK=0, IN WHICH CASE, DO NO-OP.	
17Z1	TABLE	DOPTAKE		
	B	17Z4	TAKE	
	BAL, LZ	GARAYVAL, T	DROP: GET A(N) VALUE	
	B	17Z6	(DONE)	
	STW, AI	DIMENBUF, NX	SAVE IT IN DIMEN BUF	
	BGEZ	17Z2	J'TH RESULT DIMEN R(J)	
	AW, AI	*ARGDIMPT	= MAX(0, D(J)-ABS(A(J))).	10-00023
	BGEZ	17Z3		10-00024
	B	17Z21		10-00025
17Z2	SW, AI	*ARGDIMPT	AI = +-R(J)	10-00026
	BLEZ	17Z3		10-00027
17Z21	LI, AI	0		10-00028
17Z3	MW, BI	AI	ACCUMULATE +-RSSIZE	
	BN OV	17Z5	= +-R(1)*...*R(N); GET NEXT A(J)	
	B	ERDOMAIN		
17Z4	BAL, LZ	GARAYVAL, T	TAKE: GET A(N) VALUE	
	B	17Z6	(DONE)	
	STW, AI	DIMENBUF, NX	SAVE IT IN DIMEN BUF	
	MW, BI	AI	R(J) = ABS(A(J)); ACCUM +-RSSIZE	
	BOV	ERDOMAIN	= +-A(1)*...*A(N).	
17Z5	AWM, L3	ARGDIMPT	BUMP DIMEN POINTER	10-00030
	BDR, NX	*L1	REDUCE J, GET NEXT A(J)	10-00031
17Z6	LW, OP	OPTEMP	DONE, RESTORE OP CODE	
	LAW, S	BI	RSSIZE = R(1)*...*R(N)	
	STW, S	RSSIZE		
	LW, T	RTTYPE	RESULT TYPE/RANK	
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ	
	STW, T	RSTYPE	= RTARG TYPE/RANK.	10-00033
	LW, R	LFSIZE		
	STW, R	RSRANK		
	BAL, L1	ALOCRS	ALOC RS DATA BLOCK	
	LW, S	RSRANK	ALOC LOOP CONTROL BLOCK	
	BAL, L2	INDEXB	AND INIT LOOP PARAMS.	
	AI, L3	0	IF RT ARG NOT SCALAR,	10-00035
	BEZ	17Z7	RE-INIT ARG DIMEN PTR.	10-00036
	LW, R	RTARG		
	AW, R	RTRANK		10-00038
	AI, R	1		10-00039
	STW, R	ARGDIMPT	THE D(J)	
17Z7	EQU	%		10-00041
	LW, R	RESULT	AND THE R(J).	
	AI, R	1		
	STW, R	DIMENPTR		

```

        LW,X      RSRANK      INIT LOOP: J=N;
        LW,KB     INNERMOST B-BLOCK;
        LI,AI     1           NULL COUNT = R(J+1)*...*R(N);
        LI,BI     1           WEIGHT = D(J+1)*...*D(N).
*
*   SET UP LOOP CONTROL BLOCK AND RESULT DIMENS
*
17Z8      LAW,S    DIMENBUF,X   GET ABS(A(J))
        B         17Z9,OP      WHICH OP?
17Z9      TABLE  DOPTAKE
        B         17Z10
        LCW,S     S           TAKE
        AW,S      *ARGDIMPT    DROP: COMPUTE R(J)=D(J)-ABS(A(J))
        BGEZ     17Z91        10-00043
        LI,S      0           IF D(J)<ABS(A(J)),
        EQU       %           R(J)=0.
        STW,S     *DIMENPTR,X   STORE R(J) IN RESULT DIMEN
        STW,S     BINITCNT,KB   COUNT OF LOOP(J) = R(J)
        LW,S      DIMENBUF,X   GET A(J); IF A(J)>0, INCREASE
        BGZ       17Z13        FIXED-PART VAL BY A(J)*WEIGHT.
        B         17Z14
17Z10     STW,S     *DIMENPTR,X   TAKE: R(J)=ABS(A(J)); STORE IT AS
        STW,S     BINITCNT,KB   RESULT DIMEN AND LOOP COUNT.
        LW,S      DIMENBUF,X   GET A(J)
        BLZ       17Z12        A(J)<0: BRANCH
        CW,S      *ARGDIMPT    A(J)>=0: COMPARE WITH D(J)
        BLE       17Z14        10-00048
        LW,S      =X'7FFFFFFF' A(J)>D(J): SPECIFY 'D(J) ELEMENTS
        SW,S      *ARGDIMPT    FOLLOWED BY A(J)-D(J) NULLS'.
        STW,S     NULTSBUF,X    SET UP 'MIXED SEQUENCE' LOOP:
        STW,AI     NULCTBUF,X   NULL-TEST VAL IN 'NULTSBUF' AND
        LI,S      MIXLOOP      NULL GROUP COUNT IN 'NULCTBUF'.
        STB,X      S           INCLUDE BUFFER INDEX IN OP CODE WORD.
        B         17Z15
17Z12     AW,S      *ARGDIMPT    A(J)<0: COMPUTE A(J)+D(J)
        BLZ       17Z11        10-00052
        A(J)<-D(J): SPECIFY '-D(J)-A(J)
*        NULLS, FOLLOWED BY D(J) ELMTS'.
17Z13     MW,S      BI          INCR FIXED PART VAL BY A(J)*WEIGHT
        AWM,S     FVALUE      (DROP) OR (A(J)+D(J))*WEIGHT (TAKE
17Z14     LI,S      SELOOP     LOOP TYPE = ORDINARY SEQUENTIAL
17Z15     STW,S     BLOOPTYP,KB
        STW,BI     BSTEPVAL,KB
        MW,BI      *ARGDIMPT    STEP VALUE =WEIGHT=D(J+1)*...*D(N)
        MW,AI      *DIMENPTR,X  UPDATE WEIGHT FACTOR (NO OVFL POSS)
        AI,KB      -BN          UPDATE NULL COUNT =R(J+1)*...*R(N)
        AWM,L3     ARGDIMPT     POINT TO NEXT OUTER B-BLOCK
        BDR,X      17Z8        BUMP DIMEN PNTR (IF NOT SCALAR)
                                10-00056
                                DECR J, DO NEXT COORD
*
*   COORD LOOP DONE.  NOW BUILD AND EXECUTE XSEG
*
        LI,A       1           SET RTARG FOR INDICIAL LOAD
        BAL,LX     XSETUP
        LI,X       -1          SET RESULT FOR SEQUENTIAL STORE
        BAL,LX     SETADR
        BAL,LX     GXSEGINI    INIT XSEG; EXIT IF RESULT NULL
        GEN,0,2    CODE20      GEN LOOP INIT CODE
        LW,T       RTTYPE
        LW,R       LOADINST,T  GEN LOAD,
        AW,R       RTADR
                                USE TYPE INTG INSTEAD OF ISEQ
        ISEQFIX,T  STORINST,T  STORE,
        LW,R+1     RSADR
        AW,R+1     R,CODE21
        GEN,2,2    -DOPTAKE    LOOP CONTROL CODE.
        AI,OP      EXECUTE     IF IT'S 'DROP', THE XSEG IS FINISHED
        BNEZ
        LW,R       NULLINST,T  TAKE: GEN ADDITIONAL CODE
        GEN,2,4    R,CODE22    TO COPY NULLS TO RESULT.
        B         EXECUTE
*
*   XSEG CODE:
*
CODE20     LCW,N     RSSIZE      0  INIT STORE INDEX
        LI,L2      XSEGBASE+7  1  SET NULL-COPY ADR (FOR 'TAKE' ONLY)
        BAL,L1     1STSCRPT    2  GET 1ST LOAD INDEX
        LOAD       RTARG(K)    3  COPY SELECTED ELEMENT
        STORE      RESULT(N)   4  TO RESULT.
CODE21     BIR,N     NXTSCRPT   5  BUMP STORE INDEX; GET NEXT LOAD INDX

```

	B	*RETURN	6	EXIT WHEN RESULT FULL
*	LOAD	NULL	7	COPY NULL ELEMENT
*	STORE	RESULT(N)	8	TO RESULT.
CODE22	BIR,N	XSEGBASE+11	9	BUMP STORE INDEX
	B	*RETURN	10	EXIT WHEN DONE
	BDR,N2	XSEGBASE+7	11	COUNT NULLS
	B	NXTSCRPT	12	RETURN FOR NEXT LOAD INDEX VALUE
*				
NULTSBUF	EQU	INBUF+0		NULL-TEST VALUE BUFFER
NULCTBUF	EQU	INBUF+64		NULL-GROUP COUNT BUFFER
	PAGE			
*				
*				
DCATEN	EQU	%		DYADIC CATENATE
	LI,X	1		
	LB,R	*LFARG,X		
	CB,R	*RTARG,X		PLACE HIGHER RANKED
	BLE	18Z0		ARG IN RTARG
	EXCHANGE	ARGS		
	AI,OP	-1**17		IF ARGS SWAPPED, SET OP'S SIGN BIT
18Z0	LI,A	0		
	BAL,LX	SETUPARG		SET LFRANK/SIZE/TYPE
	LI,A	1		
	BAL,LX	SETUPARG		SET RTRANK/SIZE/TYPE
	BAL,LX	TYCOMPAT		CHECK COMPATIBILITY; SET RS TYPE
	B	18Z02		GO RESOLVE NUMERIC VS TEXT
18Z01	LW,S	LFSIZE		
	AW,S	RTSIZE		RESULT SIZE = SUM OF ARG SIZES
	STW,S	RSSIZE		
	LW,R	RTRANK		SET RESULT RANK
	STW,R	RSRANK		= HIGHEST ARG RANK.
	BNEZ	18Z1		ARE BOTH ARGS SCALARS?
	BAL,L2	VECTORRS		YES: ALOC VECTOR RESULT (SIZE=2)
	LI,R	1		SET COUNTS TO 1
	STW,R	LFCOUNT		
	STW,R	RTCOUNT		
	B	18Z12		JOIN MAIN CASE
18Z02	LW,S	LFSIZE		IS 'LFARG' EMPTY...
	BEZ	18Z03		YES, USE TYPE OF 'RTARG'
	LW,T	LFTYPE		NO, SET FOR TYPE OF 'LFARG'
	LW,S	RTSIZE		IS 'RTARG' EMPTY...
	BEZ	18Z04		YES, USE TYPE OF 'LFARG'
	B	ERDOMAIN		NO, CONFLICT-- DOMAIN ERROR
18Z03	LW,T	RTTYPE		SET FOR TYPE OF 'RTARG'
18Z04	ISEQFIX,T			USE TYPE INTG INSTEAD OF ISEQ
	STW,T	RSTYPE		
	B	18Z01		OF (HOPEFULLY) NON-EMPTY ARG
18Z1	LB,K	OPER+1		CHECK LAMINATION BIT
	CI,K	LAMBIT		
	BAZ	18Z2		OFF (CATENATION)
	AI,OP	2		ON (LAMINATION): MODIFY OP CODE
	AI,K	-LAMBIT+1		GET RID OF LAM. BIT, BUMP COORD
	STB,K	OPER+1		SPEC TO CEILING OF GIVEN COORD.
	LW,S	RTSIZE		SET RESULT SIZE
	SLS,S	1		=2* RIGHT ARG SIZE.
	STW,S	RSSIZE		
	LW,R	LFRANK		
	BEZ	18Z15		UNLESS LFARG IS SCALAR,
	CW,R	RTRANK		ARG RANKS MUST AGREE
	BNE	ERRANK		
18Z15	MTW,1	RSRANK		RESULT RANK IS 1 HIGHER
	CW,K	RSRANK		COMPARE WITH K
	BL	18Z17		K<=ARG RANK: 'ST3LUPSN' WILL WORK
	BNE	ERCOORD		K> ARG RANK+1: ERROR
	STW,K	COORDK		K= ARG RANK+1: SIMULATE ST3LUPSN
	LI,A	1		
	STW,A	MIDLCNT		
	STW,A	MIDLSTEP		
	STW,A	OUTRSTEP		
	B	18Z6		JOIN OTHER CASE
18Z17	BAL,L2	ST3LUPSN		SET LOOP PARAMS; CHECK K
	B	18Z6		
18Z2	LW,R	LFRANK		EITHER LFARG IS SCALAR,
	BEZ	18Z3		
	CW,R	RTRANK		OR RANKS AGREE,
	BE	18Z4		
	AI,R	1		OR THEY MUST DIFFER BY 1.
	CW,R	RTRANK		

	BNE	ERRANK	
18Z3	AI, OP	1	MODIFY OP TO SAY 'DIFFERENT RANKS'
18Z4	BAL, L2	ST3LUPSN	SET LOOP PARAMS FOR RTARG; CHECK K;
	LW, R	LFRANK	IF LFARG ISN'T SCALAR,
	BNEZ	18Z6	WE'VE ALREADY GOT RSSIZE.
	LW, S	OUTRCNT	IF IT IS, WE MUST SET
	MW, S	INERCNT	RSSIZE = D(1)*...
	BOV	ERLENGTH	10-00063
	AW, S	RTSIZE	*(D(K)+1)*...*D(N).
	BOV	ERLENGTH	10-00065
	STW, S	RSSIZE	
18Z6	LW, S	RSSIZE	GENERAL CASE:
	BAL, L1	ALOCRS	ALOC RS DATA BLOCK
	LW, X	RSRANK	SET UP TO SCAN DIMENS
	B	18Z7, OP	
18Z7	TABLE	DOPCATEN	MODIFY PNTR(S) - BY OP:
	MTW, 1	LFARG	CAT, RANKS SAME: ALIGN PNTRS
	MTW, 1	RTARG	CAT, RANKS DIFF: OFFSET PNTRS
	MTW, 1	RESULT	LAM: PNTRS ALIGNED, 1 LOW
18Z8	LW, R	*RTARG, X	DIMEN LOOP: GET D(I)
	CW, X	COORDK	IS I=K ?
	BNE	18Z13	NO, GO CHECK C(I)=D(I)
	EXU	OPTBL2, OP	YES, COMPUTE VALUE OF LEFT COUNT
	MW, S	MIDLSTEP	NOW, WHILE C(K) AND D(K)
	STW, S	LFCOUNT	ARE HANDY.
	EXU	OPTBL3, OP	COMPUTE K'TH RESULT DIMEN
	B	18Z9, OP	DO APPROPRIATE PNTR MODIFY
18Z9	TABLE	DOPCATEN	
	B	18Z11	CAT, RANKS SAME: PNTRS ALIGNED
	B	18Z10	CAT, RANKS DIFF: ALIGN ARG PNTRS
	MTW, 1	RTARG	LAM: CORRECT FOR 1 EXTRA RS DIMEN
18Z10	MTW, 1	LFARG	
18Z11	STW, R	*RESULT, X	STORE I'TH RESULT DIMEN
	BDR, X	18Z8	COUNT DIMEN LOOP
	MTW, -1	LFARG	DONE: RESTORE PNTRS
	MTW, -1	RTARG	
	MTW, -1	RESULT	
18Z12	STW, OP	OPTEMP	SAVE OP CODE
	BAL, L1	SETN1KN	SET ADDRESSES
	BAL, LX	GXSEGINI	PREPARE TO GEN XSEG CODE
	LW, A	LFRANK	IF LFARG IS SCALAR
	BNEZ	18Z16	GEN LOAD/CONVERT OF
	BAL, L2	GENLOADT	LFARG TO ITS TEMP.
18Z16	GEN, 0, 5	CODE23A	GEN LOOP INIT CODE
	STW, XL	LFLOOP	SAVE LOC OF LEFT ARG LOOP
	LI, A	0	
	BAL, L1	GENLOAD	GEN LOAD/CONVERT OF LFARG(N1)
	LW, T	RSTYPE	
	LW, R	STORINST, T	GEN STORE RESULT(N)
	AW, R	RSADR	
	STW, R	TEMP	
	GEN, 1, 2	R, CODE23	GEN LOOP CONTROL
	STW, XL	RTLLOOP	SAVE LOC OF RIGHT ARG LOOP
	LI, A	1	
	BAL, L1	GENLOAD	GEN LOAD/CONVERT OF RTARG(K)
	LW, R	TEMP	GEN STORE RESULT(N),
	GEN, 1, 2	R, CODE24	LOOP CONTROL
	B	EXECUTE	EXECUTE XSEG
18Z13	CW, R	*LFARG, X	DIMEN LOOP: FOR I/=K, WE
	BE	18Z11	REQUIRE C(I)=D(I).
	LW, S	LFRANK	DONT ACTUALLY DO COMPARISON
	BEZ	18Z11	IF LFARG IS SCALAR.
	CW, X	COORDK	ERROR, IF I>K, PNTRS ARE STILL AS
	BG	18Z14, OP	INITIALIZED, AND MUST BE SELECT-
*			IVELY FIXED. IF I<K, THEY'RE
18Z14	TABLE	DOPCATEN	ALL OFF BY 1.
	MTW, -1	LFARG	
	MTW, -1	RTARG	
	MTW, -1	RESULT	
	B	ERLENGTH	REPORT ERROR
*			
*			
*			
*	LCW, N	RSSIZE	INIT STORE INDEX
*	LOAD/CNV	LFARG	PUT LFARG IN TEMP (ONLY IF
*	STORE	LFTEMP	IT'S SCALAR).
CODE23A	LCW, N1	LFSIZE	INIT LEFT FETCH INDEX
	LI, K	0	INIT RIGHT FETCH INDEX

	LW, OP	OPTEMP	WERE ARGS SWAPPED ?
	BGZ	CATEN4	NO, START WITH LEFT LOOP
	B	CATEN2	YES, START WITH RIGHT LOOP
*LFL00P	LOAD/CNV	LFARG(N1)	MOVE LEFT ARG ELMTS
*	STORE	RESULT(N)	TO RESULT.
CODE23	BIR, N	CATEN1	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
CATEN1	AI, N1	1	* BUMP AND
	BDR, N2	*LFL00P	* COUNT LEFT LOOP.
CATEN2	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.
	LW, N2	RTCOUNT	* INIT RIGHT LOOP
	BNEZ	*RTL00P	* GO TO RIGHT LOOP
	B	CATEN4	* RTCOUNT=0, GO TO LEFT LOOP
*RTL00P	LOAD/CNV	RTARG(K)	MOVE RIGHT ARG ELMTS
*	STORE	RESULT(N)	TO RESULT.
CODE24	BIR, N	CATEN3	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
CATEN3	AI, K	1	* BUMP AND
	BDR, N2	*RTL00P	* COUNT RIGHT LOOP.
CATEN4	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.
	LW, N2	LFCOUNT	* INIT LEFT LOOP
	BNEZ	*LFL00P	* GO TO LEFT LOOP
	B	CATEN2	* LFCOUNT=0, GO TO RIGHT LOOP
*			
*			
LFCOUNT	TEMP		LEFT LOOP COUNT
RTCOUNT	EQU	OUTRSTEP	RIGHT LOOP COUNT (SET BY ST3LUPSN)
LFL00P	TEMP		LEFT ARG LOOP LOC
RTL00P	EQU	LOOPLOC	RIGHT ARG LOOP LOC
*			
*			
OPTBL2	TABLE	DOPCATEN	LFCOUNT COMPUTATION - BY OP:
	LW, S	*LFARG, X	CAT, RANKS SAME: $C(K)*D(K+1)*...$
	LI, S	1	CAT, RANKS DIFF: $1*D(K+1)*...$
	LW, S	MIDLCNT	LAM: $D(K)*D(K+1)*...$
*			
OPTBL3	TABLE	DOPCATEN	K'TH RS DIMEN COMPUTATION - BY OP:
	AW, R	*LFARG, X	CAT, RANKS SAME: $C(K)+D(K)$
	AI, R	1	CAT, RANKS DIFF: $1+D(K)$
	LI, R	2	LAM: (EXTRA DIMEN)= 2
	PAGE		

INDEXING SETUP ROUTINES

BUILD L-BLOCK FROM SUBSCRIPT LIST

THIS SUBROUTINE ANALYSES THE SUBSCRIPT LIST (PNTR IN 'SCRIPT') AND BUILDS A LOOP CONTROL BLOCK SUITABLE FOR USE BY 'NXTSCRIPT'. THE STRUCTURE OF THIS BLOCK IS AS FOLLOWS:

1. B (ONE 6-WORD BLOCK PER SCRIPT). EACH B-BLOCK HAS ONE OF THE FOLLOWING FORMATS. DEPENDING ON THE SCRIPT STRUCTURE:

```

NULL:  STEP,  SIZE,  -,  -,  -,  0
ISEQ:  STEP,  SIZE,  -,  -,  -,  0
SCALAR: -----NONE-----
ARRAY:  VPNTR,  SIZE,  -,  -,  -,  -1

```

WHERE VPNTR = OFFSET INTO V-BLOCK WHERE SCRIPT VALUES ARE LOCATED.

2. V (VARIABLE LENGTH): CONTAINS SCRIPT VALUES FOR ALL ARRAY SCRIPTS.

THE RESULT DIMENSIONS (= ALL THE SCRIPT DIMENS, CATENATED) ARE PLACED IN THE DIMENSION BUFFER. EACH SCRIPT PLACES STUFF IN THE DIMEN BUFFER AS FOLLOWS:

```

NULL:  SIZE OF CURRENT DIMEN
ISEQ:  SIZE
SCALAR: NOTHING

```

			ARRAY: ARRAY DIMENS	
			THE ORIGINAL SCRIPT LIST POINTER IS DE-REFFED, AND THE	
			NEW L-BLOCK POINTER IS PLACED IN 'LBLOCK'.	
			THIS ROUTINE ALSO ESTABLISHES RESULT SIZE AND RANK	
			IN RSRANK/RSSIZE. LINK IS L2.	
INDEXA	EQU	%		
	LI,S	0		U10-0078
	STW,S	RSRANK	INIT SCRIPT RANK TO ZERO	U10-0079
				U10-0080
	TEST FOR SPECIAL CASE: VECTOR ARG WITH SCALAR SUBSCRIPT			U10-0081
				U10-0082
	STW,S	OLDLBLOK	SET OLDBLOK = LBLOCK (=0)	U10-0083
			FOR UPDATELP.	U10-0084
	STW,S	INERBLOK	SET INERBLOK<OUTRBLOK FOR 1STSCRIPT	U10-0085
	LI,A	1		U10-0086
	STW,A	OUTRBLOK		U10-0087
	STW,A	RSSIZE	INIT SCRIPT SIZE TO 1	U10-0088
	LW,X	SCRIPT	IN ORDER TO HAVE THE SPECIAL CASE,	U10-0089
	CW,A	2,X	THERE MUST BE EXACTLY 1 SCRIPT,	U10-0090
	BNE	1Z0		U10-0091
	CB,A	*RTARG,A	THE IND.VAR. MUST BE A VECTOR,	U10-0092
	BNE	ERRANK		U10-0093
	LW,X	3,X	AND THE SCRIPT	U10-0094
	BEZ	1Z0	(NOT ELIDED)	U10-0095
	LB,R	*X,A	MUST BE A SCALAR.	U10-0096
	BNEZ	1Z0		U10-0097
				U10-0098
	SPECIAL CASE: NO LOOP CONTROL BLOCK NEEDED			U10-0099
				U10-0100
	LB,T	*X		U10-0101
	BAL,LZ	GSCLRVAL,T	GET SUBSCRIPT VALUE	U10-0102
	SW,AI	ORIGIN	CONVERT TO ORIGIN-0	U10-0103
	BLZ	ERINDEX	REQUIRE:	U10-0104
	LW,A	RTARG	0<=SCRIPT<DIMEN	U10-0105
	CW,AI	2,A		U10-0106
	BGE	ERINDEX		U10-0107
	STW,AI	FVALUE	THAT'S THE WHOLE SCRIPT	U10-0108
	B	DREFSCR	DE-REF SCRIPT, EXIT	U10-0109
1Z0	EQU	%		U10-0110
				U10-0111
	GENERAL CASE: COMPUTE SIZE OF LOOP CONTROL BLOCK, ALLOCATE IT			U10-0112
				U10-0113
	STW,S	BLENGTH		
	STW,S	VLENGTH		
	LW,X	SCRIPT	CREATE POINTER TO SCRIPT LIST +2	
	AI,X	2	(WORD BEFORE 1ST SUBSCRIPT PNTR)	
	STW,X	SCRIPT2		
	LW,NX	*SCRIPT2	GET NUMBER OF SUBSCRIPTS	
1Z1	LW,X	*SCRIPT2,NX	GET SCRIPT PNTR (STARTING WITH LAST)	
	BNEZ	1Z3	IS IT ZERO (I.E., NULL) ?	
1Z2	MTW,1	RSRANK	YES, NULL/ISEQ: INCREASE RANK BY 1	
	MTW,BN	BLENGTH	AND RESERVE ONE ITEM FOR B-BLOCK.	
	BDR,NX	1Z1	SCAN NEXT SUBSCRIPT	
	B	1Z5	DONE	
1Z3	LI,T	ISEQ	NOT NULL, IS IT AN 'INDEX SEQUENCE'?	
	CB,T	*X		
	BE	1Z2	YES, RESERVE A WORD FOR D	
	LI,T	1	NO, IS IT A SCALAR ?	
	LB,R	*X,T		
	BEZ	1Z4	YES	
	AWM,R	RSRANK	NO,ARRAY: BUMP RANK BY SCRIPT RANK	
	AI,X	1		
	LI,S	1		
1Z6	MW,S	*R,X	COMPUTE SCRIPT SIZE; RESERVE	
	BDR,R	1Z6		
	AWM,S	VLENGTH	SPACE IN V FOR SCRIPT VALUES.	
	MTW,BN	BLENGTH	RESERVE SPACE FOR B-BLOCK	
1Z4	BDR,NX	1Z1	SCAN NEXT SCRIPT	
1Z5	LW,S	BLENGTH	COMPUTE TOTAL L-BLOCK SIZE:	
	AW,S	VLENGTH	L = B+V	
	BAL,LX7	ALOCHNW	ALLOCATE L DATA BLOCK; A = PNTR	
	STW,A	LBLOCK	COPY L PNTR	
	STW,A	OLDBLOK	SAVE FOR 'UPDATELP'	
	LI,R	INTG**8	SET TYPE=INTG, RANK=0	
	STH,R	*LBLOCK	(TYPE FIELD MUST BE LEGIT)	

	AI, A	2	BUMP PAST HEADER WORDS
	AW, A	BLENGTH	INIT B AND V PNTRS AT 1 + THE
	STW, A	BPNTR	ENDS OF THEIR RESPECTIVE
	AW, A	VLENGTH	REGIONS.
	STW, A	VPNTR	
	LI, A	DBUFEND	INIT DIMEN PNTR
	STW, A	DIMENPTR	
*			
*	SCAN SUBSCRIPTS, SET UP LOOP CONTROL DATA IN L-BLOCK		
*	U10-0116		
	LW, X	SCRIPT	RE-CREATE SCRIPT LIST PNTR +2
	AI, X	2	(ALOCHNW MAY HAVE MOVED
	STW, X	SCRIPT2	THE SCRIPT LIST).
	LW, X	RTARG	CREATE POINTER TO RTARG DIMENS
	AI, X	1	
	STW, X	ARGDIMPT	
	LW, X	BPNTR	CREATE POINTER TO INNERMOST
	AI, X	-BN	B-BLOCK
	STW, X	INERBLOK	
	LI, X	1	
	LB, NX	*RTARG, X	GET RIGHT ARG'S RANK
	CW, NX	*SCRIPT2	MAKE SURE IT AGREES WITH
	BNE	ERRANK	NUMBER OF SUBSCRIPTS.
	LI, R	0	INITIALIZE:
	STW, R	FVALUE	FIXED PART VALUE = 0
	LI, S	1	
	STW, S	ISEQFLAG	ISEQ FLAG = .FALSE.
	STW, S	RSSIZE	RESULT SIZE = 1
	STW, S	WEIGHT	WEIGHT FACTOR = 1
2Z1	LW, X	*SCRIPT2, NX	GET NEXT SCRIPT PNTR
	BNEZ	2Z2	IS IT NULL ?
	LW, S	*ARGDIMPT, NX	YES, SIZE= CORRESPONDING DIMEN
	LW, AI	WEIGHT	WEIGHTED STEP = 1*WEIGHT
	B	3Z2	CONTINUE LIKE ISEQ
2Z2	LB, T	*X	GET TYPE CODE
	CI, T	ISEQ	HANDLE ISEQ SPECIALLY
	BE	3Z1	
	LI, AI	1	
	LB, R	*X, AI	GET RANK; IS IT SCALAR ?
	BEZ	2Z16	YES
	STW, R	XRANK	REMEMBER RANK
	LI, S	1	NO, INIT SIZE = 1
	AI, X	1	POINT TO WORD BEFORE 1ST DIMEN
2Z3	LW, AI	*R, X	COPY NEXT SCRIPT DIMEN
	MTW, -1	DIMENPTR	INTO DIMEN BUF,
	STW, AI	*DIMENPTR	
	ODD, S		
	MW, S	*DIMENPTR	AND ACCUMULATE SIZE.
	BDR, R	2Z3	PROCESS ALL DIMENS
	AW, X	XRANK	POINT TO WORD BEFORE 1ST VALUE WORD
	STW, S	XSIZE	SAVE SCRIPT SIZE
	AI, S	0	
	BEZ	3Z5	IF SIZE=0; NO B-BLOCK, DO NEXT SCRPT
	BAL, LZ	GARAYVAL, T	SIZE>0, GET ARRAY VALUES (INTG)
	B	2Z11	(DONE)
	BAL, LX	CHEKSUBS	OFFSET/CHECK SUBSCRIPT
	MW, AI	WEIGHT	APPLY WEIGHT FACTOR
	MTW, -1	VPNTR	STORE IT IN V-BLOCK
	STW, AI	*VPNTR	
	B	*L1	GET NEXT VALUE
2Z11	LW, S	XSIZE	FINISH: GET SCRIPT SIZE
	STW, LZ	ISEQFLAG	RESET ISEQ FLAG (>0)
	LW, AI	VPNTR	GET V-POINTER
	SW, AI	LBLOCK	CONVERT TO L-BLOCK OFFSET VALUE
	LI, L1	ARAYLOOP	SET LOOP TYPE = ARRAY
	B	3Z4	GO CREATE B-BLOCK ITEM
*			
2Z16	BAL, LZ	GSCLRVAL, T	GET SCALAR VALUE
	BAL, LX	CHEKSUBS	OFFSET/CHECK SUBSCRIPT VALUE
	STW, LX	ISEQFLAG	RESET ISEQ FLAG (>0)
	ODD, AI		
	MW, AI	WEIGHT	INCR FIXED PART VALUE
	AWM, AI	FVALUE	BY WEIGHTED SCRIPT VALUE.
	B	3Z6	PROCESS NEXT SCRIPT
*			
3Z1	LW, AI	2, X	ISEQ: CHECK RANGE OF LAST VALUE:
	BEZ	3Z7	
	MW, AI	4, X	= A+B*N

	AW, AI	3, X	
	BAL, LX	CHEKSUBS	
	LW, AI	3, X	OFFSET/CHECK FIRST VALUE:
	AW, AI	4, X	= A+B
	BAL, LX	CHEKSUBS	
	MW, AI	WEIGHT	INCR FIXED PART BY WEIGHTED
	AWM, AI	FVALUE	1ST SCRIPT VALUE.
3Z7	LW, S	2, X	SIZE = N
	LW, AI	4, X	STEP = B * WEIGHT
	MW, AI	WEIGHT	
3Z2	MTW, -1	DIMENPTR	ISEQ/NULL: PUT SIZE IN DIMEN BUF
	STW, S	*DIMENPTR	
	LI, L1	SEQLOOP	SET LOOP TYPE = SEQ
	LCW, R	ISEQFLAG	TEST ISEQ FLAG: WAS LAST SCRIPT
	BLZ	3Z3	AN ISEQ?
	LW, T	BPNTR	YES, SEE IF WE CAN COMBINE THIS
	LW, BI	BSTEPVAL, T	ONE WITH THE LAST; THIS IS
	ODD, BI		
	MW, BI	BINITCNT, T	POSSIBLE IFF (LAST SIZE)*(LAST
	CW, BI	AI	STEP) = (THIS STEP).
	BNE	3Z4	RATS.
	LW, AI	S	OH GOOD; MODIFY LAST B-BLOCK BY
	MW, AI	BINITCNT, T	SETTING ITS SIZE =
	STW, AI	BINITCNT, T	(LAST SIZE)*(THIS SIZE).
	B	3Z5	
3Z3	STW, R	ISEQFLAG	ISEQ WAS RESET: SET IT NOW (<0)
3Z4	MTW, -BN	BPNTR	
	LW, T	BPNTR	BUILD NEW B-BLOCK:
	STW, AI	BSTEPVAL, T	INCLUDE ITS WEIGHTED STEP VALUE
	STW, S	BINITCNT, T	(OR V-OFFSET) AND SIZE.
	STW, L1	BLOOPTYP, T	AND TYPE.
3Z5	MW, S	RSSIZE	UPDATE RESULT SIZE
	STW, S	RSSIZE	
3Z6	LW, AI	*ARGDIMPT, NX	UPDATE WEIGHT FACTOR
	MW, AI	WEIGHT	
	STW, AI	WEIGHT	
	BDR, NX	2Z1	DO NEXT SCRIPT
*			
*	DE-REF SCRIPT LIST DATA BLOCK		U10-0118
*			
DREFSCR	EQU	%	U10-0120
	LI, A	0	
	XW, A	SCRIPT	DISCARD SCRIPT LIST PNTR
	BAL, LX7	DREF	DE-REF SCRIPT LIST DATA BLOCK
	B	*L2	RETURN
	PAGE		
*			
*			
*	ALLOCATE LOOP CONTROL BLOCK		
*			
*		CALLED WITH THE NUMBER OF B-BLOCKS NEEDED IN 'S'.	
*		ALLOCATES AN LBLOCK BIG ENOUGH FOR THAT MANY B-BLOCKS	
*		AND AN EMPTY V-BLOCK. SETS UP LBLOCK, OLDBLOCK,	
*		INNERBLOCK, OUTRBLOCK, FVALUE. LINK IS L2.	
*			
INDEXB	EQU	%	
	MTW, -4	RETURN	ALTER RETURN ADR SO THAT, UPON
*			EXIT FROM OP DRIVER, 'LBLOCK'
*			WILL BE DE-REFED
	MI, S	BN	COMPUTE TOTAL SIZE OF B-BLOCKS
	STW, S	TEMP	SAVE IT
	BAL, LX7	ALOCHNW	ALLOCATE L-BLOCK
	STW, A	LBLOCK	SAVE POINTER
	STW, A	OLDBLOCK	(FOR UPDATLTP)
	LI, R	INTG**8	SET TYPE (MUSTN'T LEAVE IT 0)
	STH, R	*LBLOCK	
	AI, A	2	
	STW, A	OUTRBLOCK	PNTR TO OUTER B-BLOCK
	AW, A	TEMP	
	AI, A	-BN	
	STW, A	INNERBLOK	PNTR TO INNER B-BLOCK
	LI, R	0	INITIALIZE FIXED PART VALUE
	STW, R	FVALUE	
	B	*L2	RETURN
*			
TEMP	TEMP		
	PAGE		
*			

```

*
* SET UP PARAMETERS FOR THREE NESTED LOOPS
*
*      USES THE COORDINATE SPECIFICATION VALUE (OR RTRANK,
*      IF NONE) TO SET UP THREE NESTED LOOPS: THE OUTER,
*      CORRESPONDING TO COORDS 1,2,...,K-1; THE MIDDLE,
*      FOR COORD K; AND THE INNER FOR COORDS K+1,...,N
*      (WHERE K= COORD SPEC, N = RT RANK). THIS ROUTINE
*      SETS UP RANK/SIZE/TYPE CELLS IDENTICALLY FOR RT ARG
*      AND RESULT, ALLOCATES RESULT DB, COPIES RT DIMENS
*      TO RESULT, AND SETS UP LOOP CONTROL PARAMETERS AS
*      FOLLOWS:
*
*      COORDK   = K
*      OUTRCNT  = D(1)*D(2)*...*D(K-1)
*      OUTRSTEP = D(K)*D(K+1)*...*D(N)
*      MIDLCNT  = D(K)
*      MIDLSTEP = D(K+1)*D(K+2)*...*D(N)
*      INERCNT  = D(K+1)*D(K+2)*...*D(N)
*      INNERSTEP = 1
*
*      IF THE RESULT SIZE IS 0 OR 1, THIS ROUTINE EXITS
*      TO 'MNOP' WHICH SETS RESULT = RTARG; OTHERWISE IT
*      RETURNS TO CALLER. LINK IS L2.
*
*      'ST3LUPSK' IS AN ALTERNATE ENTRY WHICH USES THE VALUE
*      IN REG 'K' AS COORDINATE SPEC, INSTEAD OF VALUE IN
*      OPER WORDS. 'ST3LUPSN' IS AN ALTERNATE ENTRY WHICH
*      DOESN'T ALOC RESULT, COPY DIMENS, OR EXIT TO 'MNOP'.
*
ST3LUPSN EQU      %
AI,L2      -1**17          SET FLAG BIT
SET3LUPS EQU      %
LB,T       *RTARG          GET ARG TYPE
STW,T      RTTYPE
LI,X       1
LB,R       *RTARG,X        GET ARG RANK
STW,R      RTRANK
BEZ        10Z0            IF SCALAR, CHECK FOR NO COORD SPEC
LB,K       OPER+1          GET COORDINAT SPECIFICATION VALUE
BEZ        10Z1            IF ZERO, NONE WAS GIVEN
CW,K       RTRANK          IF GIVEN, MAKE SURE COORD<=RANK
BLE        10Z2
B          ERCOORD
10Z0 LB,K       OPER+1          RTARG IS SCALAR: NO COORDINATE
BNEZ       ERCOORD        MAY BE SPECIFIED.
LI,S       1
STW,S      INNERSTEP      SET ALL LOOP PARAMS = 1
STW,S      INERCNT
STW,S      MIDLSTEP
STW,S      MIDLCNT
STW,S      OUTRSTEP
B          10Z6
10Z1 LW,K     RTRANK        NO COORD: USE ARG RANK (=LAST COORD)
*
ST3LUPSK EQU      %
10Z2 STW,K     COORDK        SET LOOPS, USING K-REG VAL
MTW,1      RTARG           SET K'TH COORD
LW,X       RTRANK          POINT TO 1ST DIMEN -1
LI,S       1               PREPARE TO MULTIPLY ALL DIMENS
STW,S      INNERSTEP      INIT INNER COUNT
10Z3 CW,X     COORDK        INNER STEP = 1
BNE        10Z4            WHEN THE K'TH COORD IS REACHED,
STW,S      INERCNT        SET INNER COUNT AND MIDDLE STEP
STW,S      MIDLSTEP      = D(K+1)*D(K+2)*...*D(N),
MW,S       *RTARG,X        OUTER STEP
STW,S      OUTRSTEP      = D(K)*D(K+1)*...*D(N),
LW,S       *RTARG,X        AND MIDDLE COUNT = D(K);
STW,S      MIDLCNT
LI,S       1               INIT S FOR OUTER COUNT,
B          10Z5            SKIP DIMEN K.
10Z4 MW,S     *RTARG,X      ACCUM SIZE
BOV        ERLNGTH
10Z5 BDR,X    10Z3
MTW,-1     RTARG           DONE: RESTORE ARG PNTR
10Z6 STW,S    OUTRCNT      OUTER COUNT = D(1)*D(2)*...*D(K-1)
MW,S      OUTRSTEP
BOV        ERLNGTH

```

10-00067

10-00069

```

        STW,S      RTSIZE      SIZE = D(1)*D(2)*...*D(N)
        AI,L2      0           IF SIGN BIT SET, ENTRY WAS
        BLZ        *L2        'ST3LUPSN': DONT ALOC RS.
        BDR,S      10Z7       IF RESULT IS 1 ELEMENT,
        B          MNOP        RESULT = RTARG.
10Z7      BAL,L1    RSLIKERT    ALOC RESULT DB LIKE RT ARG
        BAL,LX     MRTDIMS     COPY ARG DIMENS TO RESULT
        B          *L2        RETURN

*
*
INERCNT  TEMP      INNER  LOOP  COUNT
INERSTEP TEMP      INNER  LOOP  STEP
MIDLCNT  TEMP      MIDDLE LOOP COUNT
MIDLSAVE TEMP      MIDDLE LOOP STEP
MIDLSTEP TEMP      MIDDLE LOOP SAVE CELL
OUTRCNT  TEMP      OUTER  LOOP COUNT
OUTRSAVE TEMP      OUTER  LOOP STEP
OUTRSTEP TEMP      OUTER  LOOP SAVE CELL
COORDK   TEMP      K'TH COORDINATE SPEC
*
*
*   EXCHANGE INNER/MIDDLE LOOPS
*
*           EXCHANGES INNER AND MIDDLE LOOP PARAMETRS, AND
*           SETS FVALUE TO ZERO.  LINK IS LX.
*
EXCHLUPS EQU      %           SWAP COUNTS
        LW,R      INERCNT
        XW,R      MIDLCNT
        STW,R     INERCNT
        LW,R      INERSTEP    SWAP STEPS
        XW,R      MIDLSTEP
        STW,R     INERSTEP
        LI,R      0           ZERO FVALUE
        STW,R     FVALUE
        B          0,LX       RETURN
        PAGE

*
*
*   CHECK SUBSCRIPT
*
*           THE SUBSCRIPT VALUE IN 'AI' IS OFFSET TO ORIGIN 0
*           AND RANGE-CHECKED. OFFSET VALUE IS RETURNED IN 'AI'.
*           LINK IS LX.
*
CHEKSUBS SW,AI     ORIGIN      OFFSET TO ORIGIN 0
        BLZ        ERINDEX     MAKE SURE ITS VALUE IS
        CW,AI      *ARGDIMPT,NX >=0 AND
        BL         0,LX        < (CURRENT DIMEN).
        B          ERINDEX
        PAGE

*
*
*   GET INTEGER ARRAY VALUES
*
*           'GARAYVAL' IS USED TO FETCH, ONE BY ONE (FROM LAST
*           TO FIRST), THE ELEMENTS OF AN ARRAY, CONVERTING THEM
*           TO THE INTEGER DOMAIN.  IT IS CALLED WITH THE DATA
*           BLOCK POINTER +RANK+1 IN X, SIZE (ELEMENTS) IN S,
*           AND TYPE IN T; LINK IS LZ.  THE CALLING SEQUENCE
*           IS AS FOLLOWS:
*
*           BAL,LZ    GARAYVAL,T
*           B          ENDLOC
*           (PROCESS ONE VALUE)
*
*           ...
*           B          *L1
*
*           'GARAYVAL' WILL RETURN TO BAL+2 ONCE FOR EACH ARRAY
*           VALUE, WITH ITS INTG VAL IN AI AND LCC SET; VALUES ARE
*           PASSED IN REVERSE RAVEL ORDER.  THE CALLER MUST NOT
*           CLOBBER THE FOLLOWING REGS: S, T, X, LZ.
*           AFTER ALL VALUES HAVE BEEN PASSED, 'GARAYVAL' WILL
*           RETURN TO BAL+1.
*
GARAYVAL EQU      %-LOGL      GET ARRAY VALUE: JUMP TABLE
        B          7Z5        L VALUES: CONVERT TO I
        B          ERDOMAIN    C VALUES: WRONG

```

	B	7Z10	I VALUES: OK
	B	7Z8	F VALUES: CONVERT TO I
	B	7Z1	ISEQ: GEN VALUES
	B	ERDOMAIN	LIST: WRONG
*			
7Z1	LW, AI	0, X	ISEQ: INIT TO LAST VAL
	MW, AI	2, X	
	AW, AI	1, X	= BASE+STEP*SIZE
7Z2	STW, AI	AVALTEMP	SAVE VALUE (USER CLOBBERS)
	BAL, L1	1, LZ	PASS IT TO CALLER
	LW, AI	AVALTEMP	RESTORE VALUE
	SW, AI	2, X	SUBTRACT STEP VAL
	BDR, S	7Z2	DO NEXT
	B	0, LZ	END: RETURN TO ENDLOC
*			
7Z5	LCW, T	S	SPLIT UP ARRAY SIZE INTO
	AI, S	31	WORD AND BIT COUNT.
	SLS, S	-5	S = WORD COUNT
	OR, T	=-32	T = - BIT COUNT
7Z6	LW, AI	*S, X	GET CURRENT WORD
	AND, AI	BITMASK+33, T	EXTRACT CURRENT BIT
	BEZ	7Z7	CONVERT TO INTG FORM:
	LI, AI	1	0 OR 1.
7Z7	BAL, L1	1, LZ	PROCESS ARRAY VALUE
	BIR, T	7Z6	ALL BITS
	LI, T	-32	RESET BIT COUNT
	BDR, S	7Z6	ALL WORDS
	B	0, LZ	FINISH ARRAY PROCESSING
*			
7Z8	AND, X	=-2	FLOT: CONVERT PNTR TO DOUBLEWORD ADR
	XW, S	X	COUNT MUST BE IN INDEX
7Z9	LD, AF	*S, X	GET NEXT VALUE
	BAL, LX	F2I	CONVERT TO INTEGER
	B	ERDOMAIN	(IF POSSIBLE)
	BAL, L1	1, LZ	PROCESS IT
	BDR, X	7Z9	ALL VALUES
	B	0, LZ	FINISH
*			
7Z10	LW, AI	*S, X	INTG: GET NEXT VALUE
	BAL, L1	1, LZ	PROCESS IT
	BDR, S	7Z10	ALL VALUES
	B	0, LZ	FINISH
*			
AVALTEMP	TEMP		ARRAY VALUE TEMP
	PAGE		
*			
*			
*	GET INTEGER	SCALAR VALUE	
*			
*		'GSCLRVAL' IS USED TO OBTAIN THE VALUE OF A	
*		SCALAR, CONVERTING IT TO THE INTEGER DOMAIN IF	
*		NECESSARY. IT IS CALLED WITH THE DATA BLOCK	
*		POINTER IN X AND TYPE IN T; LINK IS LZ.	
*		THE CALLING SEQUENCE IS:	
*			
*		BAL, LZ GSCLRVAL, T	
*			
*		THE VALUE IS RETURNED IN 'AI' WITH LCC SET ACCORDING	
*		TO VALUE'S SIGNUM.	
*			
GSCLRVAL	EQU	%-LOGL	GET SCALAR VALUE: JUMP TABLE
	B	8Z3	L VALUE: CONVERT TO I
	B	ERDOMAIN	C VALUE: WRONG
	B	8Z5	I VALUE: OK
	B	8Z4	F VALUE: CONVERT TO I
	B	8Z1	ISEQ: (TO HANDLE 1-ELMT ARRAY)
	B	ERDOMAIN	LIST: WRONG
*			
8Z1	LW, AI	2, X	ISEQ: VALUE = BASE+STEP
	AW, AI	3, X	
	B	0, LZ	RETURN
*			
8Z3	LW, AI	2, X	LOGL SCALAR: GET VALUE
	AND, AI	=X'80000000'	
	BEZ	0, LZ	
	LI, AI	1	CONVERT TO I-FORM
	B	0, LZ	RETURN
*			


```

8Z4      AI,X      1      FLOT SCALAR: GET VALUE
          SLS,X    -1
          LD,AF    2,X
          BAL,LX   F2I      CONVERT TO INTO
          B        ERDOMAIN (IF POSSIBLE)
          B        0,LZ     RETURN

```

```

*
8Z5      LW,AI     2,X      INTG: GET VALUE
          B        0,LZ     RETURN
          PAGE

```

GET SUBSCRIPT VALUES

THESE SUBROUTINES EXECUTE THE LOOPS SPECIFIED BY THE L-BLOCK, DELIVERING ONE ZERO-ORIGIN SUBSCRIPT VALUE PER CALL. '1STSCRPT' INITIALIZES THE LOOPS AND RETURNS THE FIRST SUBSCRIPT IN 'K'. 'NXTSCRPT' PRODUCES ONE ADDITIONAL SUBSCRIPT IN 'K' EACH TIME IT IS CALLED. LINK IS L1. THE FOLLOWING REGISTERS MUST NOT BE Clobbered BY THE CALLER, ONCE THEY HAVE BEEN SET UP BY '1STSCRPT': K, KB, KV, KN, KL.

THE LOOP CONTROL BLOCK CONTAINS THE PARAMETERS NEEDED TO INITIALIZE EACH LOOP, AND SPACE TO HOLD RUNNING LOOP VALUES. FOR A NESTING DEPTH OF N, THE BLOCK CONSISTS OF N 6-WORD 'B-BLOCKS', EACH DESCRIBING ONE LOOP, AND A 'V-BLOCK' LARGE ENUF TO CONTAIN ALL THE SUBSCRIPTS REQUIRED FOR 'ARRAY' LOOPS AS DESCRIBED BELOW. LET YJ(IJ) (FOR IJ=1,2,...,MJ) BE THE VALUES PRODUCED BY THE J'TH LOOP; THEN THE VALUES PRODUCED BY 1STSCRPT/NXTSCRPT ARE =FVALUE+Y1(I1)+Y2(I2)+...+YN(IN), WITH 'IN' RUNNING FASTEST AND 'I1' SLOWEST.

EACH LOOP IS OF ONE OF THE FOLLOWING TYPES:

1. SEQUENCE LOOP: PRODUCES $YJ(IJ) = (IJ-1) * STEPJ$ FOR $IJ = 1, 2, \dots, MJ$. B-BLOCK CONTENTS:

```

WORD 0 = STEPJ
WORD 1 = MJ
WORD 2 = FVALUE+Y1(I1)+...+YJ(IJ)
WORD 3 = MJ-IJ
WORD 4 = (UNUSED)
WORD 5 = 0 (OP CODE FOR 'SEQLOOP')

```

EACH ITERATION REDUCES WORD 3 BY 1, AND ADDS STEPJ TO WORD 2.

2. ARRAY LOOP: PRODUCES $YJ(IJ) = L(VJ+IJ-1)$ FOR $IJ = 1, 2, \dots, MJ$. B-BLOCK CONTENTS:

```

WORD 0 = VJ (OFFSET INTO V-BLOCK)
WORD 1 = MJ
WORD 2 = LOC(L)+VJ+IJ-1
WORD 3 = MJ-IJ
WORD 4 = FVALUE+Y1(I1)+...+YK(IK) (K=J-1)
WORD 5 = -1 (OP CODE FOR 'ARAYLOOP')

```

EACH ITERATION REDUCES WORD 3 BY 1, INCREASES WORD 2 BY 1 AND ACCESSES NEW YJ VALUE FROM V-BLOCK LOCATION GIVEN BY WORD 2.

3. MIXED SEQUENCE LOOP: USED BY 'TAKE' OPERATOR, THIS LOOP PRODUCES PJ 'NULL GROUPS' FOLLOWED BY MJ-PJ VALUES OF THE 'SEQUENCE' LOOP VARIETY - OR, IT FIRST PRODUCES QJ VALUES, THEN MJ-QJ NULL GROUPS.

```

CASE 1: YJ(IJ) = 'NULL GROUP' FOR IJ=1,...,PJ
        = (IJ-PJ-1)*STEPJ FOR IJ=PJ+1,...,MJ.
CASE 2: YJ(IJ) = (IJ-1)*STEPJ FOR IJ=1,...,QJ
        = 'NULL GROUP' FOR IJ=QJ+1,...,MJ.

```

B-BLOCK CONTENTS:

```

WORD 0 = STEPJ
WORD 1 = MJ
WORD 2 = FVALUE+Y1(I1)+...+YJ(IJ)
WORD 3 = MJ-IJ

```

```

*
*          WORD 4 = T+IJ (T= NULL TEST VAL, SEE BELOW)
*          WORD 5 = (X,-2) X (IN THE UPPER 8 BITS)
*
*          IS THE INDEX INTO NULTSBUF FOR 'T'
*          AND INTO NULCTBUF FOR 'G'. THE
*          LOWER 24 BITS CONTAIN -2, THE OP
*          CODE FOR 'MIXLOOP'.
*
*          NULTSBUF(X) CONTAINS 'T', THE NULL TEST VALUE: IN
*          CASE 1, T=-PJ<0; IN CASE 2, T=X'7FFFFFFF'-QJ>0.
*          NULCTBUF(X) CONTAINS 'G', THE NULL-GROUP COUNT:
*          G=MK*...*MN (K=J+1).
*          EACH ITERATION LOWERS WORD 3 BY 1 AND RAISES
*          WORD 4 BY 1 TESTING ITS SIGN: IF T+IJ>0, THE LOOP
*          ACTS LIKE A 'SEQUENCE' LOOP, ADDING STEPJ TO
*          WORD 2; IF T+IJ<=0, A NULL GROUP OF LENGTH G IS
*          PASSED TO THE RESULT, AND THE EXECUTION OF LOOPS
*          J+1,...,N IS SKIPPED. NXTSCRPT PASSES THE NULL
*          GROUP BY PUTTING G IN REG 'N2' AND EXITING ON
*          LINK L2, INSTEAD OF L1.
*
*
1STSCRPT EQU      %          GET 1ST SCRIPT
          BAL,LX      UPDATELP
          LW,KB       OUTRBLOK
          LW,K        FVALUE
          CW,KB       INERBLOK
          BLE         4Z6
          LI,KN       1
          BAL,KL      *L1
*
*
*
*
NXTSCRPT EQU      %          GET NEXT SCRIPT
          BDR,KN      *KL
*
*          BRANCH IF CURRENT LOOP NOT DONE
*          (KL HAS BEEN SET BY ONE OF
*          SEVERAL 'BAL,KL ...' OPS).
4Z1      AI,KB       -BN
          CW,KB       OUTRBLOK
          BL          *RETURN
          LW,KN       BCOUNT,KB
          LW,LX       BLOOPTYP,KB
          BDR,KN      4Z3,LX
          B           4Z1
*
4Z21     LW,KV       BOFFSET,KB
*
          AI,KV       1
          LW,K        *KV
          AW,K        BADDEND,KB
          STW,KV      BOFFSET,KB
          B           4Z5
*
4Z3      TABLE     LOPBIAS
          B           4Z13
          B           4Z21
4Z31     LW,K        BSCRIPT,KB
          AW,K        BSTEPVAL,KB
4Z4      STW,K       BSCRIPT,KB
4Z5      STW,KN      BCOUNT,KB
          AI,KB       BN
4Z6      LW,KN      BINITCNT,KB
          LW,LX       BLOOPTYP,KB
          B           4Z7,LX
4Z61     LW,KV      BINITOFS,KB
          AW,KV       LBLOCK
          STW,K       BADDEND,KB
          AW,K        *KV
          CW,KB       INERBLOK
          BL          4Z2
          BAL,KL      *L1
*
          AI,KV       1
          LW,K        *KV
          AW,K        BADDEND,KB
          B           *L1
*
4Z7      TABLE     LOPBIAS
          B           4Z9

```

U10-0122

```

*      B      4Z61      ARRAY LOOP
*      CW,KB   INERBLOK  SEQ LOOP
*      BL      4Z4      IF THIS ISN'T THE INNERMOST LOOP,
*      BAL,KL  *L1      SAVE ITS STUFF & INIT INNER LOOPS.
*      4Z8     AW,K     BSTEPVAL,KB  SET KL TO INNER-SEQUENCE-LOOP LOC,
*      B      *L1      AND RETURN K TO CALLER.
*      4Z9     LB,LX    LX          BUMP INNER SEQUENCE LOOP BY SIMPLY
*      LW,LX    NULTSBUF,LX        ADDING THE STEP VAL; RETURN.
*      STW,LX   BNULTEST,KB        INIT MIXLOOP:
*      4Z10     MTW,1    BNULTEST,KB  MIXLOOP: GET NULTSBUF INDEX
*      BGZ      4Z11    >0 TREAT LIKE SEQLOOP
*      LI,KL    4Z10    <=0 FIRST, SEND A NULL GROUP,
*      B        4Z12    THEN TRY AGAIN.
*      4Z11     CW,KB   INERBLOK    IS THIS THE INNERMOST LOOP ?
*      BL      4Z4      NO, SAVE STUFF & INIT NEXT INNER LUP
*      BAL,KL  *L1      YES, SEND SCRIPT (K) TO CALLER,
*      MTW,1    BNULTEST,KB        AND ADD STEP TO SCRIPT, AS LONG
*      BGZ      4Z8      AS NULL TEST >0.
*      4Z12     LW,LX    BLOOPTYP,KB  GET NULCTBUF INDEX
*      LB,LX    LX
*      LW,N2    NULCTBUF,LX        GET NULL-GROUP COUNT
*      B        *L2      GO TO SEND-NULL ROUTINE
*      4Z13     MTW,1    BNULTEST,KB  BUMP NON-INNER MIXLOOP
*      BGZ      4Z31    IF NULL TEST >0, TREAT LIKE SEQLOOP
*      LI,KL    4Z13    IF <=0, SEND NULL GROUPS UNTIL
*      B        4Z12    IT GOES >0 (OR LOOP COUNTS OUT).
*      PAGE
*
*
*      MOVE BUFFER DIMENSIONS
*
*      COPIES DIMENSION BUFFER CONTENTS TO RESULT DIMENS.
*      LINK IS LX.
*
*      MBUFDIMS EQU      %
*      LCW,X     RSRANK   GET -RANK
*      BEZ      0,LX     IF RANK=0, NO DIMENS TO MOVE
*      LW,A      RESULT   GET PNTR TO 1ST RS DIMEN -2
*      9Z1      LW,R     DBUFEND,X  GET DIMEN FROM BUFFER
*      STW,R     2,A      STORE IT IN RESULT
*      AI,A      1
*      BIR,X     9Z1      DO 'EM ALL
*      B        0,LX     RETURN
*
*
*      MOVE ARG DIMENSIONS
*
*      COPIES RIGHT ARG DIMENSIONS TO RESULT DIMENS.
*      LINK IS LX.
*
*      MRTDIMS EQU      %
*      LW,X     RSRANK   GET RANK
*      BEZ      0,LX     IF RANK=0, NO DIMENS TO MOVE
*      MTW,1    RTARG    POINT TO 1ST DIMEN -1
*      9Z2      MTW,1    RESULT
*      LW,R     *RTARG,X  GET A DIMEN FROM ARG
*      STW,R     *RESULT,X  STORE IT IN RESULT
*      BDR,X     9Z2      DO 'EM ALL
*      MTW,-1    RTARG    RESTORE POINTERS TO NORMALCY
*      MTW,-1    RESULT
*      B        0,LX     RETURN
*      PAGE
*
*
*      SET UP ADDRESS FOR INDEXED LOAD/STORE
*
*      SETS UP LFADR IF A=0; RTADR IF A=1; RSADR IF A=2.
*      FOR CHAR/INTG/FLOT TYPE, SETS UP ADDRESS OF 1ST
*      DATA VALUE, WITH 'K' IN INDEX FIELD; FOR LOGL,
*      SETS UP ADR OF THE INDEXED LOAD-STORE LOGL ROUTINE.
*      LINK IS LX. ADDRESS IS ALSO RETURNED IN 'R'.
*
*      XSETUP EQU      %
*      LW,X     LFARG,A   GET ARG PNTR
*      LI,T     1

```

	LB, R	*X, T	GET LEFT ARG RANK
	AW, R	LFARG, A	
	AI, R	K**17+2	INIT ADR = ARG+RANK+2, K
	LB, T	*X	GET TYPE CODE
	B	%+1-LOGL, T	GO INTO JUMP TABLE - BY TYPE:
	B	6Z1	LOGL
	B	6Z2	CHAR
	B	6Z2	INTG
	B	6Z3	FLOT
	STW, R	XSEQLADR, A	ISEQ: SAVE BASE VALUE ADR
	LW, R	XSQADRTB, A	GET ADR OF ISEQ CALC ROUTINE
	B	6Z2	STORE ADR, EXIT
*			
6Z1	STW, R	XLGLLADR, A	LOGL: SAVE ADR OF LOGL DATA
	LW, R	XLGADRTB, A	ADR = LOC OF LOGL COPY SUBR
6Z2	STW, R	LFADR, A	STORE ADR
	B	0, LX	RETURN
6Z3	AI, R	1	FLOT: ADR =
	AND, R	=-2	ARG+RANK+(2 OR 3), K
	STW, R	LFADR, A	STORE ADR
	B	0, LX	RETURN
*			
XLGADRTB	TABLE	0	LOGL LOAD/STORE SUBR ADR TABLE
	PZE	INDXLDLL	INDEXED LOAD LOGICAL LEFT
	PZE	INDXDLR	INDEXED LOAD LOGICAL RIGHT
	PZE	INDXSTLG	INDEXED STORE LOGICAL RESULT
*			
XSQADRTB	TABLE	0	ISEQ CALC ROUTINE ADR TABLE
	PZE	INDXLSQL	INDEXED LOAD ISEQ LEFT
	PZE	INDXLSQR	INDEXED ISEQ LOAD RIGHT
	PAGE		
*			
*			
*	SET UP ARG ADRS IN SPECIAL WAY		
*			
*		SETS LFADR FOR SEQUENTIAL LOAD USING INDEX N1,	
*		RTADR FOR INDICIAL LOAD USING INDEX K, AND RSADR	
*		FOR SEQUENTIAL STORE USING INDEX N. LINK IS L1.	
*		ALSO SETS 'DELRSIZE' = (OLD RSSIZE)-(NEW RSSIZE).	
*		THIS VALUE IS NONZERO ONLY FOR CERTAIN CHARACTER RESULTS,	
*		IN WHICH CASE IT MAY EQUAL -1, -2, OR -3.	
*			
SETN1KN	EQU	%	
	LI, X	-3	SET UP LEFT ADR CELL FOR
	BAL, LX	SETADR	SEQUENTIAL LOAD.
SETSPEC1	EQU	%	
	LI, A	1	SET UP RTADR FOR
	BAL, LX	XSETUP	INDEXED LOAD: INDEX=K.
	LW, BI	RSSIZE	KEEP OLD RSSIZE
	LI, X	-1	SET UP RSADR FOR SEQUENTIAL
	BAL, LX	SETADR	STORE: INDEX(IF ANY) =N.
	SW, BI	RSSIZE	COMPUTE DELRSIZE = OLD RSSIZE
	STW, BI	DELRSIZE	- NEW RSSIZE.
SETSPEC2	EQU	%	
	LD, IX	N1REGMSK	IF IT HAS AN INDEX (=N, IF ANY),
	CW, IX1	LFADR	CHANGE IT TO N1.
	BAZ	SETLK1	
	STS, IX	LFADR	
	B	*L1	RETURN
SETLK1	LW, IX	LFADR	IF LFARG IS LOGL OR ISEQ,
	CI, IX	INDXLDLL	CHANGE ADR OF INDX-LOAD
	BE	20Z1	SUBROUTINE TO ADR OF ONE
	CI, IX	INDXLSQL	THAT USES K1 AS THE INDEX,
	BNE	*L1	INSTEAD OF K.
	LI, IX	INDSQLL1	
	B	20Z2	
20Z1	LI, IX	INDLDLL1	
20Z2	STW, IX	LFADR	
	B	*L1	
*			
*			
N1REGMSK	BOUND	8	
	PZE	0, N1	N1 INDEX VALUE
	PZE	0, -1	AND MASK.
*			
DELRSIZE	TEMP		CHANGE IN RSSIZE
	PAGE		
*			

```

*
* INDEXED LOAD LOGICAL
*
*          LOADS INTO AI THE K'TH BIT (ORIGIN 0) OF
*          LFARG/RTARG.
*          LINK IS LX. K IS NOT ALTERED.
*
INDXLDLL EQU      %          INDEXED LOAD LOGICAL LEFT
        LW,AF      XLGLLADR   GET LFARG DATA ADR
        B          16Z1
INDXDLR EQU      %          INDEXED LOAD LOGICAL RIGHT
        LW,AF      XLGLRADR   GET RTARG DATA ADR
*          SPLIT SUBSCRIPT INTO
16Z1      LCW,AI      K          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K      -5          AND WORD INDEX IN K.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        AND,AI      *AF,K      FROM APPROPRIATE WORD.
        SCS,K      5          RESTORE K
        BEZ        0,LX        CONVERT TO 0 OR -1 REPRESENTATION
        LI,AI      -1
        B          0,LX
*
*
* INDEXED LOAD LOGICAL, INDEX K1
*
*          LOADS INTO AI THE K1'TH BIT (ORIGIN 0) OF LFARG
*          LINK IS LX. K1 IS NOT ALTERED.
*
INDLDLL1 EQU      %          INDEXED LOAD LOGICAL LEFT (K1)
*          SPLIT SUBSCRIPT INTO
        LCW,AI      K1          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K1     -5          AND WORD INDEX IN K1.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        AND,AI      *XLGLLADR,K1 FROM APPROPRIATE WORD.
        SCS,K1     5          RESTORE K1
        BEZ        0,LX        CONVERT TO 0 OR -1 REPRESENTATION
        LI,AI      -1
        B          0,LX
*
*
* INDEXED STORE LOGICAL
*
*          STORES AI (0 OR -1) INTO K'TH BIT (ORIGIN 0) OF RESULT.
*          LINK IS LX. K IS NOT ALTERED.
*
INDXSTLG EQU      %
        LW,AI-1    AI
*          SPLIT SUBSCRIPT INTO
        LCW,AI      K          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K      -5          AND WORD INDEX IN K.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        STS,AI-1    *XLGSTADR,K STORE DATA BIT INTO APPROP WORD
        SCS,K      5          RESTORE INDEX VALUE
        LW,AI      AI-1        RESTORE VALUE OF AI
        B          0,LX
*
XLGLLADR TEMP      INDEXED LOGICAL LOAD LEFT ADR
XLGLRADR TEMP      INDEXED LOGICAL LOAD RIGHT ADR
XLGSTADR TEMP      INDEXED LOGICAL STORE ADR
*
* INDEXED LOAD ISEQ
*
*          COMPUTES K'TH ELEMENT OF ISEQ LFARG/RTARG
*          AND PLACES IT IN AI. LINK IS LX. K IS
*          NOT ALTERED.
*
INDXLSQL EQU      %          INDEXED LOAD ISEQ LEFT
        LW,AF      XSEQLADR   GET ISEQ BASE ADR
21Z1      LW,AI      K          GET INDEX
21Z2      ODD,AI
        MW,AI      1,AF        ISEQ ELEMENT = STEP*(K+1)+BASE
        AW,AI      1,AF        = STEP*K+STEP+BASE.
        AW,AI      0,AF
        B          0,LX        RETURN

```

```

*
INDXLSQR EQU      %      INDEXED LOAD ISEQ RIGHT
          LW,AF    XSEQRADR GAE RIGHT ISEQ BASE ADR
          B        21Z1    CONTINUE AS FOR LFARG
*
INDSQLL1 EQU      %      INDEXED LOAD ISEQ LEFT, K1 INDEX
          LW,AF    XSEQLADR GET BASE ADR
          LW,AI    K1       GET INDEX FROM K1, NOT K
          B        21Z2    CONTINUE AS FOR LF LOAD FROM K
*
XSEQLADR TEMP      ISEQ BASE ADR FOR LFARG
XSEQRADR TEMP      ISEQ BASE ADR FOR RTADR
          PAGE
*
*
*  ALLOCATE RESULT LIKE RIGHT ARG
*
*          SETS UP RESULT RANK/SIZE/TYPE CELLS TO MATCH
*          THOSE OF RTARG, AND ALLOCATES THE RESULT DATA
*          BLOCK. LINK IS L1. PNTR RETURNED IN 'A' AND RESULT.
*          'RSLIKRT1' IS AN ALTERNATE ENTRY WHICH DOES
*          THE SAME THING, EXCEPT IT TAKES RSTYPE FROM 'T'.
*          'RSLIKRT2' DOESN'T SET UP RSTYPE AT ALL.
*
RSLIKERT EQU      %
          LW,T      RTTYPE
RSLIKRT1 EQU      %
          ISEQFIX,T USE TYPE INTG INSTEAD OF ISEQ
          STW,T      RSTYPE RESULT TYPE = ARG TYPE
RSLIKRT2 EQU      %
          LW,R      RTRANK
          STW,R      RSRANK RESULT RANK = ARG RANK
          LW,S      RTSIZE
          STW,S      RSSIZE RESULT SIZE = ARG SIZE
          B          ALOCRS ALLOCATE RESULT; RETURN TO CALLER
          PAGE
*
*  UPDATE L-BLOCK POINTERS
*
*          IF THE L-BLOCK MAY HAVE BEEN MOVED (DUE TO AN ALLOCATION
*          OR DE-REF OPERATION), A CALL UPON THIS ROUTINE WILL
*          ASSURE THAT ALL NECESSARY L-BLOCK POINTERS REMAIN VALID.
*          LINK IS LX.
*
UPDATELP EQU      %      UPDATE L-BLOCK POINTERS
          LW,K      LBLOCK NEW L-BLOCK LOC
          SW,K      OLDLBLOK OLD L-BLOCK LOC
          BEZ       0,LX    RETURN, IF THEY ARE EQUAL
          AWM,K      OLDLBLOK NOT EQUAL: ADD DIFFERENCE TO
          AWM,K      INERBLOK ALL POINTERS.
          AWM,K      OUTRBLOK
          B          0,LX    RETURN
*
*  LISTCHK-CHECKS LFARG AND RTARG-ERROR IF TYPE IS LIST OR HIGHER
*
LISTCHK  LI,A      LIST
          CB,A      *LFARG
          BLE       ERDOMAIN
          CB,A      *RTARG
          BG        0,LX
          B          ERDOMAIN
          PAGE
*
*
*  TEMPS FOR INDEX ROUTINES
*
ARGDIMPT TEMP      ARG DIMEN POINTER
BLENGTH TEMP      B BLOCK LENGTH
VLENGTH TEMP      V BLOCK LENGTH
BPNTR EQU          OUTRBLOK B BLOCK POINTER
VPNTR TEMP      V BLOCK POINTER
DIMENPTR TEMP      DIMENSION POINTER
SCRIPT2 TEMP      SUBSCRIPT LIST PNTR +2
ISEQFLAG TEMP      ISEQ FLAG (<0 FOR .TRUE.)
XRANK TEMP      SUBSCRIPT RANK
XSIZE TEMP      SUBSCRIPT SIZE
WEIGHT TEMP      WEIGHT FACTOR
FVALUE TEMP      FIXED PART VALUE

```

OLDLBLOK TEMP
INERBLOK TEMP
OUTRBLOK TEMP
*

OLD L-BLOCK POINTER
INNERMOST B-BLOCK POINTER
OUTERMOST B-BLOCK POINTER

NTEMPS SET
23Z END

ERROR,X'F',TLOC>38 'TOO MANY TEMPS'
TLOC

U10-0124
U10-0125

*
*
* E X T E R N A L C O M M U N I C A T I O N
*
*
* DEFINITIONS
*

DEF	AXDRIVER	INDEXED ASSIGNMENT EXEC DRIVER	
DEF	COORDK	K'TH COORDINATE SPEC	
DEF	DBUFEND	DIMENSION BUFFER END	
DEF	DCATEN	DYADIC CATENATE OP ROUTINE	
DEF	DCOMPRES	DYADIC COMPRESS OP ROUTINE	
DEF	DDROP	DYADIC DROP OP ROUTINE	
DEF	DEXPAND	DYADIC EXPAND OP ROUTINE	
DEF	DROTATE	DYADIC ROTATE OP ROUTINE	
DEF	DTAKE	DYADIC TAKE OP ROUTINE	
DEF	DTRANS	DYADIC TRANSPOSE OP ROUTINE	
DEF	EXCHLUPS	EXCHANGE MIDDLE/INNER LOOPS	
DEF	GSCLRVAL	GET INTEGER SCALAR VALUE	
DEF	INDX@	START OF PROCEDURE	
DEF	INDXLDLR	INDEXED LOAD LOGICAL RIGHT	
DEF	INERCNT	INNER LOOP COUNT	
DEF	INNERSTEP	INNER LOOP STEP SIZE	
DEF	MAXDIMEN	MAXIMUM NUMBER OF DIMENSIONS	
DEF	MBUFDIMS	MOVER BUFFER DIMENSIONS TO RS	
DEF	MIDLCNT	MIDDLE LOOP COUNT	
DEF	MIDLSAVE	MIDDLE LOOP SAVE TEMP	
DEF	MIDLSTEP	MIDDLE LOOP STEP SIZE	
DEF	MREVERSE	MONADIC REVERSE OP ROUTINE	
DEF	MRTDIMS	MOVER RT ARG DIMENSIONS TO RS	
DEF	MTRANS	MONADIC TRANSPOSE OP ROUTINE	
DEF	OUTRCNT	OUTER LOOP COUNT	
DEF	OUTRSAVE	OUTER LOOP SAVE TEMP	
DEF	OUTRSTEP	OUTER LOOP STEP SIZE	
DEF	RSLIKRT1	ALOC RESULT LIKE RTARG	
DEF	RSLIKRT2	ALOC RESULT LIKE RTARG	U10-0004
DEF	SETSPEC1	SPECIAL ADR SETUP	
DEF	SETSPEC2	SPECIAL ADR SETUP	
DEF	ST3LUPSN	SET UP LOOP PARAMS, NO ALOC	
DEF	SXDRIVER	SUBSCRIPTED EXPRESSION EXEC DRIVER	
DEF	XSETUP	SET UP ADR FOR INDEXED LOAD	

*
* REFERENCES
*

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS	
REF	ALOCRS	ALLOCATE RESULT DATA BLOCK	
REF	AXRETURN	INDEXED ASSIGNMENT EXEC RETURN	
REF	BITMASK	BIT SELECTION MASK TABLE	
REF	DREF	DE-REF	
ERCOORD	EQU	COORDINATE SPECIFICATION ERROR	
REF	ERRANK	DOMAIN ERROR	
REF	ERDOMAIN	DOMAIN ERROR	
REF	ERINDEX	INDEX RANGE ERROR	
REF	ERLENGTH	LENGTH ERROR	
REF	ERRANK	RANK ERROR	
REF	EXECUTE	EXECUTE XSEG	
REF	FLOT0	FLOATING PT 0.0	
REF	F2I	CONVERT F TO I	
REF	GENLOAD	GEN LOAD BY RSTYPE	
REF	GENLOADT	GEN LOAD TO TEMP	
REF	GXSEGINI	GEN XSEG INITIALIZATION	
REF	GXSTEXEC	GEN XSEG STORE; EXECUTE XSEG	
REF	INBUF	INPUT BUFFER (USED AS TEMP BUFFER)	
REF	INDXTMPS	INDX TEMPS ARE IN WINDOW IN APLUTSI	U10-0006
REF	LBLOCK	LOOP CONTROL BLOCK PNTR	
REF	LFADR	LEFT ARG ADDRESS	
REF	LFARG	LEFT ARG PNTR	
REF	LFLGLADR	LEFT LOGICAL DATA ADR	
REF	LFLGLCNT	LEFT LOGICAL BIT COUNT	
REF	LFRANK	LEFT ARG RANK	
REF	LFSIZE	LEFT ARG SIZE	
REF	LFTYPE	LEFT ARG TYPE	
REF	LOADINST	LOAD INSTRUCTION TABLE, BY TYPE	
REF	LOOPLOC	LOOP LOCATION	
REF	MNOP	MONADIC NO OP ROUTINE	
REF	NILCK	'NIL CHECK' = SCRIPT LIST PNTR	
SCRIPT	EQU	SUBSCRIPT LIST POINTER	

REF	OPER	OPERATOR WORDS
REF	ORGADJ	ORIGIN, ADJUSTED (=1-ORG)
REF	ORIGIN	INDEX ORIGIN VALUE (0 OR 1)
REF	RESULT	RESULT DATA BLOCK POINTER
REF	RETURN	RETURN ADR CELL
REF	RSADR	RESULT ADDRESS
REF	RSRANK	RESULT RANK
REF	RSSIZE	RESULT SIZE
REF	RSTYPE	RESULT TYPE
REF	RTADR	RIGHT ARG ADDRESS
REF	RTARG	RIGHT ARG PNTR
REF	RTRANK	RIGHT ARG RANK
REF	RTSIZE	RIGHT ARG SIZE
REF	RTTYPE	RIGHT ARG TYPE
REF	SETADR	SET UP ADDRESS, SEQUENTIAL
REF	SETADRS1	SET UP ADDRESSES, SEQUENTIAL
REF	SETUPARG	SET UP ARG PARAMS
REF	STORINST	STORE INST TABLE
REF	SXRETURN	SUBSCRIPTED EXP EXEC DRIVER RETURN
REF	SYSTERR	SYSTEM ERROR
REF	TYCOMPAT	TYPE COMPATIBILITY CHECK
REF	VECTORRS	ALLOCATE VECTOR RESULT
REF	XSEGBASE	BASE OF XSEG AREA
	PAGE	

*
*
*
*
*

A S S E M B L Y P A R A M E T E R S

	SYSTEM	SIG5F	
PROGSECT	CSECT	1	
INDX@	RES	0	START OF PROCEDURE

*
*

REGISTERS

IX	EQU	0	EVEN/ODD PAIR
IX1	EQU	1	*
N	EQU	1	XSEG EXECUTION REG
N1	EQU	4	XSEG EXECUTION REG
K1	EQU	4	XSEG EXECUTION REG
N2	EQU	11	XSEG EXECUTION REG
N3	EQU	10	XSEG EXECUTION REG
X	EQU	1	SCRIPT POINTER
T	EQU	2	TYPE REG
NX	EQU	3	COUNT REG
XL	EQU	3	XSEG LOC REG
K	EQU	2	SUBSCRIPT VALUE
KB	EQU	3	B-BLOCK POINTER
KN	EQU	8	LOOP COUNT
KV	EQU	9	ARRAY LOOP OFFSET
KL	EQU	10	LINK REG
A	EQU	4	ARG ADR/INDEX
LZ	EQU	4	INDEX LINK REG
LX	EQU	5	INDEX LINK REG
OP	EQU	6	OP CODE REG
AF	EQU	6	ACCUM FOR FLOT VALUES
AF1	EQU	7	*
LX7	EQU	7	INDEX LINK REG
AI	EQU	7	ACCUM FOR LOGL/CHAR/INTG VALUES
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS
R	EQU	8	GENERAL WORK REG
BI	EQU	9	2ND ACCUM FOR L/C/I VALUES
S	EQU	11	SIZE
L3	EQU	12	LINK REG
L2	EQU	13	LINK REG
L1	EQU	14	LINK REG
R15	EQU	15	

*
*

ARG TYPE CODES

LOGL	EQU	1	LOGICAL (BIT)
CHAR	EQU	2	CHARACTER (BYTE)
INTG	EQU	3	INTEGER (WORD)
FLOT	EQU	4	FLOATING (DOUBLEWORD)
ISEQ	EQU	5	INDEX SEQUENCE VECTOR
LIST	EQU	6	LIST

*

CODESTRING DESIGNATIONS

```

*
DOPCMPRS EQU      84          DYADIC COMPRESS
DOPCATEN EQU      87          DYADIC CATENATE/LAMINATE
DOPTAKE  EQU     116          DYADIC TAKE
*
*   B-BLOCK STRUCTURE PARAMETERS
*
BN          EQU      6          B BLOCK ITEM SIZE
BSTEPVAL   EQU      0          OFFSET TO STEP VALUE
BINITOFFS  EQU      0          OFFSET TO INITIAL OFFSET VALUE
BINITCNT   EQU      1          OFFSET TO INITIAL COUNT
BNULLCNT   EQU      1          OFFSET TO NULL GROUP COUNT
BSCRIPT    EQU      2          OFFSET TO SCRIPT SAVE LOC
BOFFSET    EQU      2          OFFSET TO OFFSET SAVE LOC
BCOUNT     EQU      3          OFFSET TO COUNT SAVE LOC
BADDEND    EQU      4          OFFSET TO ADDEND VALUE
BNULTEST   EQU      4          OFFSET TO NULL TEST VALUE
BLOOPTYP   EQU      5          OFFSET TO LOOP TYPE CODE
*
SEQLOOP    EQU      0          SEQUENCE LOOP
ARRAYLOOP  EQU     -1          ARRAY LOOP
MIXLOOP    EQU     -2          MIXED SEQUENCE LONP
LOPBIAS    EQU     -2          LOOP TYPE CODE BIAS
*
*   DIMENSION BUFFER PARAMETERS
*
MAXDIMEN   EQU      63          MAXIMUM NUMBER OF DIMENSIONS
DIMENBUF   EQU      XSEGBASE    DIMENSION BUFFER START
DBUFEND    EQU      DIMENBUF+MAXDIMEN DIMENSION BUFFER END
*
*   MISCELLANEOUS PARAMETERS
*
INFINITY   EQU      1**19-1     PLUS INFINITY (FOR 'LI')
LAMBIT     EQU      X'40'       LAMINATE BIT
PAGE
*
*
*   P R O C S
*
*
TLOC       SET      0                               U10-0008
*
TEMP       CNAME     1
DTEMP      CNAME     2
PROC
DO1        NAME=2
TLOC       SET      TLOC+(TLOC&1)                   U10-0011
DISP       TLOC      U10-0012
LF         EQU      INDXTMPS+TLOC                    U10-0013
TLOC       SET      TLOC+NAME                        U10-0014
PEND
*
*
EVEN       CNAME     0
ODD        CNAME     1
PROC
LF         EQU      %
ERROR,1,(CF(2)+NAME)&1  'REGISTER HAS WRONG PARITY'
PEND
*
*
EQUAL      CNAME
PROC
LF         EQU      %
ERROR,1,1-(CF(2)=CF(3)) 'REGISTERS MUST BE EQUAL'
PEND
*
*
EXCHANGE   CNAME
OPEN       I,K, GROUP
GROUP      EQU      LFARG, LFTYPE, LFSIZE, LFRANK, LFADR
PROC
LF         EQU      %
I          DO       NUM(AF)
K          SET      SCOR(AF(I), ARGS, TYPES, SIZES, RANKS, ADRS)
ERROR,1,K=0  'UNKNOWN GROUP INDICATOR'
LW,R      GROUP(K)
XW,R      GROUP(K)+1

```

```

        STW, R      GROUP(K)
        FIN
        PEND
        CLOSE      I, K, GROUP
*
*
ISEQFIX  CNAME
        PROC
LF        CI, CF(2)  ISEQ
        BNE        %+2
        LI, CF(2)  INTG
        PEND
        PAGE
*
*
*   XSEG GEN PROCS
*
*
GEN      OPEN      GEN
        CNAME
        OPEN      M, N, MN, I
        PROC
LF        EQU      %
        ERROR, 1, 1-(NUM(CF)=3)  'WRONG NUMBER OF CF ARGS'
M        SET      CF(2)
N        SET      CF(3)
MN       SET      M+N
        ERROR, 1, 1-(NUM(AF)=(M>0)+(N>0))  'WRONG NUMBER OF AF ARGS'
        DO      M>0
I        DO      N
        LW, AF(1)+M+I-1  AF(2)+I-1
        FIN
I        DO      MN*(MN<3)
        STW, AF(1)+I-1  I-1, XL
        ELSE
        LCI      MN
        STM, AF(1)  0, XL
        FIN
I        ELSE
        DO      MN*(MN<3)
        LW, BUF  AF(1)+I-1
        STW, BUF  I-1, XL
        ELSE
        LCI      N
        LM, BUF  AF(1)
        STM, BUF  0, XL
        FIN
        FIN
        AI, XL  MN
        PEND
        CLOSE  M, N, MN, I
        PAGE
*
*
*   TABLE BUILDING PROCS
*
*
TABLE    CNAME
        OPEN      T, N
        PROC
T        SET      %-AF(1)
LF       EQU      T
        DISP     T
        PEND
*
*
ITEM     CNAME
        PROC
N        SET      T+AF(1)-%
        ERROR, 1, N<0  'ITEM OUT OF SEQUENCE'
        RES      N*(N>0)
        DISP     AF(1)
        PEND
        CLOSE  T, N
        PAGE
*
*
*   O P E R A T O R      E X E C U T I O N      D R I V E R S

```

```

*
*
*      USECT      PROGSECT
*
*
*      SUBSCRIPTED EXPRESSION EXECUTION DRIVER
*
*      ENTERED WITH SUBSCRIPT LIST POINTER IN 'SCRIPT' AND
*      POINTER TO EXPRESSION-TO-BE-SUBSCRIPTED IN 'LFARG'.
*      IF NO ERRORS OCCUR, IT RETURNS TO 'SXRETURN' WITH
*      RESULT POINTER IN 'RESULT', AND 'SCRIPT' DE-REFFED.
*      LEFT ARG POINTER WILL HAVE BEEN MOVED TO 'RTARG',
*      WHICH, ALONG WITH 'LBLOCK', STILLS NEEDS TO
*      BE DE-REFFED.
*
*
*      SXDRIVER EQU      %      SUBSCRIPTED EXPRESSION EXEC DRIVER
*      LI,A      SXRETURN      SET UP RETURN ADDRESS
*      STW,A      RETURN
*      EXCHANGE  ARGS
*      BAL,L2     INDEXA
*
*      LB,T      *RTARG
*      CI,T      LIST
*      BGE       ERDOMAIN
*      ISEQFIX,T
*      STW,T     RSTYPE
*
*      ALSELECT EQU      %
*      LW,S      RSSIZE
*      BAL,L1     ALOCRS
*      BAL,LX     MBUFDIMS
*
*      MTSELECT EQU      %
*      LI,X      -1
*      BAL,LX     SETADR
*      BAL,LX     GXSEGINI
*      GEN,0,1    BAL1STSC
*      LI,A      1
*      BAL,LX     XSETUP
*      LB,T      *RTARG
*      AW,R      LOADINST,T
*      GEN,1,0    R
*      LI,A      NXTSCRPT
*      STW,A      LOOPLOC
*      B          GXSTEXEC
*
*      XSEG CODE:
*
*      LCW,N      RSSIZE
*      BAL,L1     1STSCRPT
*      LOAD       RTARG(K)
*      STORE      RESULT(N)
*      BIR,N      NXTSCRPT
*      B          *RETURN
*      PAGE
*
*      INDEXED ASSIGNMENT EXECUTION DRIVER
*
*      ENTERED WITH SUBSCRIPT LIST POINTER IN 'SCRIPT',
*      POINTER TO INDEXED VARIABLE (ON LEFT SIDE OF ASSIGNMENT
*      ARROW) IN 'LFARG', AND POINTER TO THE VALUE-TO-ASSIGN
*      IN 'RTARG'. IF NO ERROR OCCUR, IT RETURNS TO 'AXRETURN'
*      WITH RESULT POINTER (VALUE OF THE 'LFARG' VARIABLE,
*      AFTER THE INDEXED ASSIGNMENT) IN 'RESULT', AND 'SCRIPT'
*      DE-REFFED. 'LBLOCK' STILL NEEDS TO BE DE-REFFED.
*
*
*      AXDRIVER EQU      %      INDEXED ASSIGNMENT EXEC DRIVER
*      LI,A      AXRETURN      SET RETURN ADR
*      STW,A      RETURN
*      EXCHANGE  ARGS
*      BAL,L2     INDEXA
*
*      LI,A      0
*      BAL,LX     SETUPARG
*      LI,A      1

```

```

SUBSCRIPTED EXPRESSION EXEC DRIVER
SET UP RETURN ADDRESS
PUT ARG PNTR IN RTARG.
BUILD LOOP CONTROL BLOCK FROM
SCRIPT LIST; SET SIZE/RANK.

```

```

DISALLOW LIST INDEXING
USE TYPE INTG INSTEAD OF ISEQ
SET TYPE

```

```

ALLOCATE RESULT DATA BLOCK
MOVE DIMENS FROM BUFFER TO RS

```

```

SET RSADR FOR SEQUENTIAL STORE
GEN XSEG INIT; EXIT IF RESULT NULL
GEN:      BAL,L1  1STSCRPT
SET RTADR FOR INDEXED LOAD
GEN INDEXED LOAD
SET TO LOOP BACK TO 'NXTSCRPT'
GEN STORE/LOOP CONTROL; EXECUTE

```

```

U10-0016
U10-0017
U10-0018
U10-0019
U10-0021
U10-0022
U10-0023
U10-0024

```

	BAL, LX	SETUPARG	IND.VAR.	
	BAL, LX	TYCOMPAT	MAKE SURE THE TYPES ARE COMPATIBLE,	
	B	ERDOMAIN	PUT HIGHEST TYPE IN RSTYPE.	
	LW, S	RSSIZE	SET SCRIPT SIZE (NUMBER	
	STW, S	SCRPTCNT	OF SCRIPT VALUES).	
	LI, S	1		
	CW, S	LFSIZE	IF ASSIGN VALUE IS 1-ELMT,	
	BE	14Z2	SKIP RANK/DIMEN CHECKS.	
	LI, AI	1		10-00071
	LW, A	LFARG	SET UP TO COMPARE DIMENS OF ASSIGN	10-00072
	LW, R	LFRANK	VALUE AND SUBSCRIPT STRUCTURE.	10-00073
	AI, R	1	EXCEPT FOR 1'S (WHICH ARE SKIPPED	10-00074
	LCW, K	RSRANK	OVER), THESE DIMENS MUST MATCH.	10-00075
	AI, K	-1		10-00076
14Z10	BDR, R	14Z13	GET A DIMEN FROM ASSIGN VALUE	10-00077
14Z11	BIR, K	14Z12	NO MORE: REMAINING SCRIPT	
	B	14Z2	DIMENS MUST ALL BE 1'S.	10-00079
14Z12	CW, AI	DBUFEND, K		10-00080
	BE	14Z11		10-00081
	B	ERRANK		10-00082
14Z13	LW, S	2, A	(NEXT ASSIGN-VALUE DIMEN)	10-00083
	AI, A	1		10-00084
	CI, S	1	IF IT'S 1, SKIP IT AND GET ANOTHER	10-00085
	BE	14Z10		10-00086
14Z14	BIR, K	14Z15	GET NEXT SCRIPT DIMEN	
	B	ERRANK	(THERE HAD BETTER BE ONE)	10-00088
14Z15	CW, AI	DBUFEND, K	IF IT'S 1, SKIP IT & GET NEXT	10-00089
	BE	14Z14		10-00090
	CW, S	DBUFEND, K	BOTH DIMENS NON 1: THEY	10-00091
	BE	14Z10	ARE REQUIRED TO AGREE.	10-00092
	B	ERLENGTH		10-00093
14Z2	LW, S	SCRPTCNT	IF INDEX NULL, SET RESULT=	10-00061
	BEZ	MNOP	INDEXED VARIABLE.	
	LW, OP	LFTYPE	TEST FOR SPECIAL CASE:	U10-0026
	CW, OP	RTTYPE	ALLOW ONLY IF LFARG IS CONVERTIBLE	U10-0027
	BG	14Z21	TO RTARG'S TYPE,	U10-0028
	LI, OP	ISEQ	RT ARG IS NOT AN ISEQ,	
	CW, OP	RTTYPE		
	BE	14Z21		
	LW, A	RTARG		U10-0029
	LW, R	1, A	AND RTARG IS RE-USABLE AS RESULT	U10-0030
	AI, R	-2	(I.E., ITS REF COUNT IS 2).	U10-0031
	BNEZ	14Z21	NO GOOD	U10-0032
*			GOOD: CONDITIONS FOR THE SPECIAL	U10-0033
*			CASE HAVE BEEN MET. WE'LL NOW	U10-0034
*			USE RTARG FOR THE RESULT.	U10-0035
	LI, OP	-100	SIGNAL TO GEN SPECIAL CODE	U10-0036
	MTW, 1	1, A	CREATE NEW COPY OF RTARG PTR	U10-0037
	STW, A	RESULT	USE IT FOR RESULT PTR	U10-0038
	B	14Z22	JOIN GENERAL CASE	U10-0039
14Z21	EQU	%	GENERAL CASE: ALOC NEW RS BLOCK	U10-0040
	BAL, L1	RSLIKRT2	ALOC RS WITH SHAPE = IND.VAR.SHAPE	
	BAL, LX	MRTDIMS	MOVE IND.VAR. DIMENS TO RESULT	
14Z22	LW, S	RTSIZE	IF IND.VAR. IS NULL,	U10-0042
	BEZ	*RETURN	THEN SO IS RESULT.	
	LI, X	-3		
	BAL, LX	SETADR	ST LFADR FOR SEQ LOAD (N)	
	LI, A	1		
	BAL, LX	XSETUP	SET RTADR FOR INDX LOAD (K)	
	LI, A	2		
	BAL, LX	XSETUP	SET RSADR FOR INDX STORE (K)	
	LI, XL	XSEGBASE	PREPARE TO GEN XSEG CODE	
	LI, S	1		
	CW, S	LFSIZE	IF ASSIGN VALUE IS 1-ELMT,	
	BNE	14Z3		
	LI, A	0	GEN LOAD/CONVERT ASSIGN	
	BAL, L2	GENLOADT	VALUE TO TEMP.	
14Z3	BIR, OP	14Z4	JUMP IF SPECIAL CASE	U10-0044
	STW, XL	LOOPLOC	GENERAL CASE	U10-0045
	GEN, 0, 9	CODE7	GEN K-LOOP CONTROL	
	AWM, XL	-7, XL		
	B	14Z5	JOIN SPECIAL CASE	U10-0047
14Z4	GEN, 0, 1	BAL1STSC	SPECIAL CASE: SIMPLER K-LOOP CONTROL	U10-0048
14Z5	EQU	%	COMMON CODE	U10-0049
	LI, A	0	GEN LOAD/CONVERT OF ASSIGN-VALUE	
	BAL, L1	GENLOAD	ELMT FOR SELECTED POSITION.	
	LW, T	RSTYPE		
	LW, R	STORINST, T	SET UP STORE INST	

	AW,R	RSADR	JUMP IF SPECIAL CASE	U10-0051
	BIR,OP	14Z6		
	STW,R	TEMP		
	GEN,1,4	R,CODE8	GEN N-LOOP CONTROL	
	LW,X	LOOPLOC		
	AWM,X	-1,XL	FILL IN BRANCH ADR	
	AWM,XL	6,X		
	LI,A	1	GEN LOAD/CONVERT OF IND.VAR. TO FILL IN NON-SELECTED ELMTS.	
	BAL,L1	GENLOAD		
	LW,R	TEMP		
	GEN,1,1	R,CODE11	GEN STORE/N-LOOP CONTROL 3	
	LW,X	LOOPLOC		
	AWM,X	-1,XL	FILL IN BRACH ADR	
*				
	LI,R	-1	PREPARE FOR XSEG EXECUTION:	
	STW,R	KTEMP	'OLD K' = -1,	
	B	14Z7	JOIN SPECIAL CASE	U10-0053
14Z6	GEN,1,3	R,CODE13	SPECIAL CASE: SIMPLER STORE CONTROL	U10-0054
14Z7	EQU	%	COMMON CODE	U10-0055
	LCW,N	LFSIZE	N = ASSIGN-VALUE INDEX	
	LW,N2	SCRPTCNT	N2 = NUMBER OF SCRIPT VALUES	
	BIR,OP	XSEGBASE	IF SPECIAL CASE, DONT ALLOW BREAKS	U10-0057
	B	EXECUTE		
*				
KTEMP	EQU	TEMP		
SCRPTCNT	EQU	INERCNT		
*				
*				
*				
*				
*				
		XSEG CODE:		
	LOAD/CNV	LFARG	(ONLY IF LFARG IS	U10-0059
	STORE	LFTEMP	1-ELEMENT).	
CODE7	BAL,L1	1STSCRPT	GET INDEX (K) OF 1ST SELECTED ELMT	
	CW,K	KTEMP	IF NEW INDEX <= PREVIOUS	
	BLE	0 (LOOP1)	ONE, GO BACK & STORE IN IT.	
	XW,K	KTEMP	SWAP W/INDEX OF PREVIOUS SEL.ELMT	
	AI,K	1	IF THERE'S A GAP,	
	CW,K	KTEMP		
	BNE	0 (LOOP2)	GO FILL IN WITH IND.VAR. VALUES.	
	CW,K	RSSIZE	IF THIS IS THE FINAL FILL-OUT	
	BGE	*RETURN	PASS, STOP WHEN FULL.	
*LOOP1	LOAD/CNV	LFARG(N)	FETCH NEXT ELMT OF ASSIGN-VALUE	
*	STORE	RESULT(K)	STORE ASSIGN-VALUE ELMT	
CODE8	AI,N	1	BUMP ASSIGN-VALUE INDEX	
	BDR,N2	NXTSCRPT	GET NEXT INDEX	
	LW,K	RSSIZE	NO MORE INDECES, SET TO FILL	
	B	3 (XW,K)	OUT RESULT W/ IND.VAR. ELMTS.	
*LOOP2	LOAD/CNV	RTARG(K)	FETCH IND.VAR. ELMT TO FILL INTO	
*	STORE	RESULT(K)	NON-SELECTED RESULT POSITION.	
CODE11	B	4 (AI,K)	KEEP FILLING 'TIL WE HIT SELECTED	
*			POSITION, OR FILL UP RESULT.	
*				U10-0061
*				U10-0062
*				U10-0063
*				U10-0064
*				U10-0065
*	LOAD/CNV	LFARG	(ONLY IF LFARG IS	U10-0066
*	STORE	LFTEMP	1-ELEMENT).	U10-0067
*	BAL,L1	1STSCRPT	GET 1ST K-VALUE	U10-0068
*	LOAD/CNV	LFARG(N)	GET NEXT ELEMENT OF ASSIGN VALUE	U10-0069
*	STORE	RESULT(K)	STORE INTO SELECTED RESULT POSITION	U10-0070
CODE13	AI,N	1	BUMP LOAD INDEX	U10-0071
	BDR,N2	NXTSCRPT	GET NEXT K-VALUE	U10-0072
	B	*RETURN	EXIT WHEN ALL K-VALUES USED	
	PAGE			
*				
*				
*	M O N A D I C	I N D E X I N G	O P	R O U T I N E S
*				
*				
MTRANS	EQU	%	MONADIC TRANSPOSE	
	LI,A	1		
	BAL,LX	SETUPARG	SET UP RT ARG PARAMS	
	LW,S	RTRANK	GET ARG RANK	
	CI,S	1	IF ARG IS A SCALAR OR VECTOR,	
	BLE	MNOP	RESULT = ARG.	
	BAL,L2	INDEXB	RANK>=2: ALOC LOOP CONTROL BLOCK	
	BAL,L1	RSLIKERT	ALOC RESULT DB LIKE ARG DB	
	LW,KB	OUTRBLOK	PREPARE TO SET UP LOOP CONTROLS	

	LI, OP	SEQLOOP	LOOP TYPE = SEQUENTIAL
	LI, S	1	FIRST STEP VAL = 1
	LW, A	RESULT	INCREASING DIMEN INDEX
	LW, X	RTRANK	DECREASING DIMEN INDEX AND COUNTER
	MTW, 1	RTARG	POINT TO ARG DIMENS
*			FOR LOOP(K) ...
22Z1	STW, OP	BLOOPTYP, KB	TYPE = SEQUENTIAL
	STW, S	BSTEPVAL, KB	STEP = D(N)*D(N-1)*...*D(N+2-K)
	LW, R	*RTARG, X	
	STW, R	BINITCNT, KB	COUNT = D(N+1-K)
	STW, R	2, A	SET K'TH RESULT DIMEN = D(N+1-K)
	MW, S	*RTARG, X	UPDATE STEP VALUE: MULT BY D(N+1-K)
	AI, A	1	BUMP INCREASING DIMEN INDEX
	AI, KB	+BN	BUMP TO NEXT HIGHER LOOP BLOCK
	BDR, X	22Z1	BUMP DECREASING DIMEN INDEX, COUNT
	MTW, -1	RTARG	CORRECT ARG POINTER
	B	MTSELECT	USE LOOP CONTROL BLOCK TO PICK
			RESULT ELEMENTS IN CORRECT ORDER.
*			
	PAGE		
*			
*			
MREVERSE	EQU	%	MONADIC REVERSE
	LB, T	*RTARG	GET ARGUMENT TYPE
	CI, T	LIST	
	BGE	ERDOMAIN	DISALLOW LIST OR ABOVE
	CI, T	ISEQ	IS IT AN ISEQ
	BNE	23Z1	NO, FORGET IT.
	LB, S	OPER+1	CHECK FOR COORDINATE SPEC.
	BDR, S	ERCOORD	ERROR IF GREATER THAN 1
	LI, S	3	YES, CREATE AN ISEQ RESULT
	BAL, LX7	ALOCHNW	ALLOC DATA BLOCK FOR RESULT
	STW, A	RESULT	SAVE POINTER
	LI, R	ISEQ**8+1	TYPE=ISEQ, RANK=1
	STH, R	*RESULT	
	LW, X	RTARG	GET ARG POINTER
	LW, S	2, X	COMPUTE RESULT BASE VALUE =
	AI, S	1	A+B*(N+1) WHERE A/B ARE
	ODD, S		
	MW, S	4, X	THE ARG BASE/STEP VALUES.
	AW, S	3, X	
	LCW, S+1	4, X	RESULT STEP = -B
	LW, S-1	2, X	RESULT SIZE = N
	LCI	3	STORE RESULT SIZE/BASE/STEP
	STM, S-1	2, A	IN RESULT DATA BLOCK.
	B	*RETURN	EXIT
23Z1	EQU	%	ARG NOT ISEQ: PROCEED NORMALLY
	BAL, L2	SET3LUPS	SET UP LOOP PARAMETERS
	LCW, R	MIDLSTEP	MIDDLE LOOP STEP =
	STW, R	MIDLSTEP	-D(K+1)*...*D(N).
	AW, R	OUTRSTEP	FVALUE =
	STW, R	FVALUE	(D(K)-1)*(D(K+1)*...*D(N).
INDXOP1	LI, A	1	SET UP RT ADR FOR INDEXED LOAD
	BAL, LX	XSETUP	
	LI, X	-1	SET UP RS ADR FOR SEQUENTIAL STORE
	BAL, LX	SETADR	
	LI, XL	XSEGBASE	PREPARE TO GEN XSEG
	GEN, 0, 7	CODE1	GEN OUTER/MIDDLE LOOP INIT CODE
	STW, XL	LOOPLOC	INNER LOOP LOC HERE
	LB, T	*RTARG	
	LW, R	LOADINST, T	GEN LOAD OF RIGHT ARG,
	AW, R	RTADR	
	LW, R+1	STORINST, T	STORE TO RESULT,
	AW, R+1	RSADR	
	GEN, 2, 1	R, AIN1INST	INCREMENT OF STORE INDEX.
	GEN, 0, 9	CODE3	GEN INNER LOOP CONTROL
	LW, R	LOOPLOC	
	AWM, R	-8, XL	
	B	EXECUTE	EXECUTE XSEG
*			
*			
*			
	XSEG CODE:		
CODE1	LCW, N	RSSIZE	0 INIT STORE INDEX
	LW, K	FVALUE	1 INIT INDEX
	LW, N1	OUTRCNT	2 INIT OUTER LOOP
	STW, K	OUTRSAVE	3 SAVE INDEX FOR OUTER LOOP
	LW, N2	MIDLCNT	4 INIT MIDDLE LOOP
	STW, K	MIDLSAVE	5 SAVE INDEX FOR MIDDLE LOOP
	LW, N3	INERCNT	6 INIT INNER LOOP

*	LOAD	RTARG(K)	LOAD ARG
*	STORE	RESULT(N)	STORE RESULT
AIN1INST	AI,N	1	BUMP STORE INDEX
CODE3	AW,K	INNERSTEP	BUMP BY INNER LOOP STEP
	BDR,N3	0 (INNER LOOP LOC)	COUNT INNER LOOP
CODE5	LW,K	MIDLSAVE	RESTORE INDEX FOR MIDDLE LOOP
	AW,K	MIDLSTEP	BUMP BY MIDDLE LOOP STEP
	BDR,N2	XSEGBASE+5	COUNT MIDDLE LOOP
	LW,K	OUTRSAVE	RESTORE INDEX FOR OUTER LOOP
	AW,K	OUTRSTEP	BUMP BY OUTER LOOP STEP
	BDR,N1	XSEGBASE+3	COUNT OUTER LOOP
	B	*RETURN	EXIT
	PAGE		

*
*
*
*
*

D Y A D I C I N D E X I N G O P R O U T I N E S

DROTATE	EQU	%	DYADIC ROTATE	
	BAL,LX	LISTCHK	CHECK ARGUMENTS FOR LIST	
	BAL,L2	SET3LUPS	SET UP LOOP PARAMS	
	BAL,LX	EXCHLUPS	EXCHANGE INNER/MIDDLE LOOPS	
	LI,R	0	INIT FIXED PART VALUE	
	STW,R	FVALUE		
	LI,A	1		
	BAL,LX	XSETUP	SET RIGHT ARG	
	LI,A	2	AND RESULT	
	BAL,LX	XSETUP	FOR INDICIAL ACCESS.	
	LI,XL	XSEGBASE	PREPARE TO GEN CODE	
	GEN,0,7	CODE1	GEN OUTER/MIDDLE LOOP INIT/SAVE	
	MTW,-2	XSEGBASE	CHANGE 'RSSIZE' TO 'LFSIZE'	
	LI,X	1	THERE ARE TWO CASES OF ROTATE,	
	LB,R	*LFARG,X	DEPENDING ON LEFT ARG RANK:	
	BGZ	13Z3	>0: GENERAL CASE	
	LW,X	LFARG	=0 (SCALAR): MUCH LIKE 'REVERSE'	
	LB,T	*LFARG		
	BAL,LZ	GSCLRVAL,T	GET VALUE OF LEFT ARG (AS INTG)	
13Z0	BGEZ	13Z1		
	LI,AI-1	-1	EXTEND SIGN	
	B	13Z2		
13Z1	LI,AI-1	0		
13Z2	DW,AI-1	INERCNT	COMPUTE MOD(LFARG,D(K))	
	AI,AI-1	0		
	BGEZ	13Z21	MAKE SURE SHIFT>=0	
	AW,AI-1	INERCNT		
13Z21	STW,AI-1	TESTVAL	= SPECIAL MIDL LOOP VALUE	
	LW,AI	TESTVAL		
	MW,AI	INNERSTEP	INIT DELTA =	
	STW,AI	IDELTA	SHIFT*D(K+1)*...*D(N).	
	GEN,0,2	CODE4	GEN COPY IDELTA TO DELTA	
	B	13Z9	JOIN GENERAL CASE	
13Z3	LI,A	0	GENERAL CASE	
	STW,XL	TEMP		10-00003
	BAL,LX	SETUPARG	SET UP LFARG PARAMS.	
	LW,XL	TEMP		10-00005
	AI,X	1	X=BLOCK POINTER+RANK+1	
	AI,AI	-1	CHECK IF LEFT ARGUMENT HAS 1 ELEM.	
	BNEZ	13Z31	NO-GO TO NORMAL PROCESSING	
	LI,S	1	YES-FETCH THE SOLITARY ARRAY	
	BAL,LZ	GARAYVAL,T	VALUE AND TREAT AS SCALAR.	
	BAL,R15	SYSTERR	THIS POINT SHOULDN'T BE REACHED.	
	B	13Z0		
13Z31	SW,X	LFARG	X=LFARG+1	
	CW,X	RTRANK	NORMAL CASE,CHECK FOR RANK	
	BNE	ERRANK	CONFORMABILITY.	
	LW,S	LFARG	SAVE LFARG PTR	
	MTW,1	RTARG	POINT TO 1ST DIMEN -1	
13Z4	CW,X	COORDK	IF WE'VE GOTTEN TO K'TH COORD,	
	BNE	13Z6		
	MTW,1	LFARG	LINE UP LF/RT DIMEN PTRS.	
13Z5	BDR,X	13Z4		
	B	13Z7		
13Z6	LW,R	*LFARG,X	MAKE SURE (ALL) LF DIMENS =	
	CW,R	*RTARG,X	RT DIMENS (K'TH ONE DELETED).	
	BE	13Z5		
	STW,S	LFARG	NO GOOD: RESTORE PNTRS	
	MTW,-1	RTARG		
	B	ERLENGTH	AND SIGNAL ERROR.	

13Z7	STW,S	LFARG	OK: RESTORE PNTRS
	MTW,-1	RTARG	
	LI,X	-3	SET LEFT ARG FOR
	BAL,LX	SETADR	SEQUENTIAL LOAD.
	LI,T	INTG	
	STW,T	RSTYPE	
	LI,A	0	
	BAL,L1	GENLOAD	GEN LOAD LEFT(N) IN INTG TYPE
	LW,T	LFTYPE	
	GEN,0,3	CODE2	GEN LCC TEST CODE
13Z9	STW,XL	LOOPLOC	SAVE INNER LOOP LOC
	GEN,0,1	AWKDELTA	GEN ADD OF DELTA
	LW,T	RTTYPE	
	LW,A	LOADINST,T	GEN LOAD RIGHT(K),
	AW,A	RTADR	
	LW,A+1	SWKDELTA	SUBTRACT OF DELTA,
	ISEQFIX,T		
	LW,A+2	STORINST,T	STORE RESULT(K),
	AW,A+2	RSADR	
	GEN,3,2	A, CODE6	AND LOOP CONTROL CODE.
	GEN,0,7	CODE5	
	B	EXECUTE	
*			
*			
*			
*			
*			
	SET OUTER LOOP		
	SET MIDDLE LOOP		
	LOAD/CNV LFARG(N)		GET A SHIFT COUNT
CODE2	BGEZ	ROTATE1	EXTEND SIGN(AI) INTO AI-1
	LI,AI-1	-1	
	B	ROTATE2	
CODE4	LW,R	IDELTA	RE-INIT DELTA (ONLY FOR
	STW,R	DELTA	ROTATE W/SCALAR LFARG).
ROTATE1	LI,AI-1	0	*(THIS CODE NOT IN XSEG)
ROTATE2	AI,N	1	* BUMP LFARG INDEX
	DW,AI-1	INERCNT	* COMPUTE J=MOD(SHIFT COUNT, D(K))
	AI,AI-1	0	*
	BGEZ	13Z91	* MAKE SURE J>=0
	AW,AI-1	INERCNT	*
13Z91	STW,AI-1	TESTVAL	* J
	LW,AI	TESTVAL	* DELTA (DISTANCE BETWEEN UNSHIFTED
	MW,AI	INERSTEP	* AND SHIFTED INDEX VALUES)
	STW,AI	DELTA	* = J*D(K+1)*D(K+2)*...*D(N).
	LW,N3	INERCNT	* RUN INNER LOOP D(K) TIMES
ROTATE3	CW,N3	TESTVAL	* UPON STARTING THE D(K)-J+1 'TH
	BNE	*LOOPLOC	* ITERATION OF INNER LOOP,
	LCW,R	OUTRSTEP	* DECR DELTA BY D(K)*D(K+1)*...*D(N)
	AWM,R	DELTA	* TO KEEP K'TH COORD INDEX
	B	*LOOPLOC	* IN BOUNDS.
*LOOPLOC	EQU	%	
AWKDELTA	AW,K	DELTA	SHIFT INDEX
*	LOAD	RTARG(K)	FETCH, SHIFTEDLY
SWKDELTA	SW,K	DELTA	UNSHIFT INDEX
*	STORE	RESULT(K)	STORE, UNSHIFTEDLY
CODE6	AW,K	INERSTEP	BUMP BY INNER LOOP STEP
	BDR,N3	ROTATE3	COUNT INNER LOOP
*	...		COUNT MIDDLE/OUTER LOOPS
*	...		EXIT
*			
DELTA	TEMP		DISTANCE 'TWEEN RTARG/RESULT ELMTS
IDELTA	TEMP		INITIAL VALUE FOR DELTA
TESTVAL	TEMP		LOOP COUNT SPECIAL TEST VAL
	PAGE		
*			
*			
DTRANS	EQU	%	DYADIC TRANSPOSE
	BAL,LX	LISTCHK	CHECK ARGUMENTS FOR LIST
	LI,X	1	
	CB,X	*LFARG,X	LEFT ARG MUST BE VECTOR ...
	BNE	ERRANK	
	LB,S	*RTARG,X	... OF LENGTH EQUAL RIGHT RANK.
	LW,X	LFARG	
	CW,S	2,X	
	BNE	ERLENGTH	
	STW,S	LFSIZE	
	LI,R	0	
	STW,R	RSRANK	INIT RS RANK = 0
	LI,R	INFINITY	

```

11Z1      LCW,X      LFSIZE
          BEZ        MNOF
          STW,R      DBUFEND,X
          BIR,X      11Z1
          BAL,LZ     TRANSVAL
          *
          *
          B          11Z21
          AW,AI      ORGADJ
          BLEZ       ERDOMAIN
          CW,AI      RSRANK
          BLE        11Z2
          STW,AI     RSRANK
11Z2      SW,AI      LFSIZE
          BGZ       ERDOMAIN
          LW,R       1,NX
          AI,NX      -1
          CW,R       DBUFEND-1,AI
          BGE       *L1
          STW,R      DBUFEND-1,AI
          B          *L1
          *
11Z21     LW,X       RSRANK
          *
          LI,A       DBUFEND-1
          SW,A       LFSIZE
          LI,NX      DBUFEND-1
          SW,NX      RSRANK
          LI,S       1
11Z3      LW,R       *A,X
          CI,R       INFINITY
          BE         ERDOMAIN
          STW,R      *NX,X
          MW,S       R
          BDR,X      11Z3
          STW,S      RSSIZE
          LB,T       *RTARG
          ISEQFIX,T
          STW,T      RSTYPE
          *
          *
          *          NOW SET UP THE LOOP CONTROL BLOCK
          *
          LW,S       RSRANK
          BAL,L2     INDEXB
          LW,KB      OUTRBLOK
          LI,R       0
          LCW,X      RSRANK
11Z4      LW,S       DBUFEND,X
          STW,S      BINITCNT,KB
          STW,R      BSTEPVAL,KB
          EQUAL,0,SEQLOOP
          STW,R      BLOOPTYP,KB
          AI,KB      BN
          BIR,X      11Z4
          LI,BI      1
          BAL,LZ     TRANSVAL
          *
          B          ELSELECT
          SW,AI      ORIGIN
          MI,AI      BN
          EQUAL,BSTEPVAL,0
          AWM,BI     *OUTRBLOK,AI
          MW,BI      1,NX
          AI,NX      -1
          B          *L1
          *
          *
          TRANSVAL  EQU      %
          LW,NX      RTARG
          AW,NX      LFSIZE  (=RTRANK) NX = PTR TO LAST RT DIMEN -1
          LW,X       LFARG
          AI,X       2
          LB,T       *LFARG
          LW,S       LFSIZE
          B          GARAYVAL,T
          PAGE
          *
          *

```

NO-OP IF RESULT SCALAR
INIT RS DIMENS = +INF

SET UP NX= PTR TO LAST RT DIMEN -1,
AND INIT 'GARAYVAL' TO FETCH
LEFT ARG VALUES.
(DONE)
CONVERT NEW LFARG VAL TO ORIGIN 1
RANGE CHECK: 1<=VAL<=LFSIZE
RSRANK=MAX(RSRANK, VAL)

FINISH RANGE CHECK, PREPARE TO
INDEX INTO DIMENBUF.
GET CURRENT RT DIMEN
DECR DIMEN PTR
RSDIMEN(VAL) = MIN(RSDIMEN(VAL),
RTDIMEN).
GO FETCH NEXT VALUE FROM LFARG

LF ARG SCAN DONE. NOW MAKE SURE
THE VALUES FORMED A DENSE SET,
AND COMPUTE RESULT SIZE.

IF D(I)=INF, THEN THERE WAS
A GAP IN LF VALUES.
MOVE DIMENS TO END OF DIMENBUF
SIZE = D(1)*...*D(N)
STORE RS SIZE
USE TYPE INTG INSTEAD OF ISEQ
RESULT TYPE = RIGHT TYPE

INIT WEIGHT = 1
SET NX= PNTR TO D(N) -1; INIT
'GARAYVAL' TO GET LFARG VALUES.
(DONE: ALOC RS, DO SELECTION)
CONVERT NEW LF VALUE TO ORIGIN 0
COMPUTE INDEX TO B-BLOCK(VAL),
AT BSTEPVAL POSITION.
ADD WEIGHT TO STEP(VAL)
WEIGHT = D(K)*...*D(N)
DECR D(K) PNTR
GO GET NEXT VALUE

X = PTR TO LEFT ARG +RANK+1
T = LEFT TYPE
S = LEFT SIZE
BEGIN FETCHING LEFT VALUES; RETURN

DEXPAND	EQU	%	DYADIC EXPAND
	LI,OP	DOPCMPRS+1	ALMOST LIKE COMPRESS
DCOMPRES	EQU	%	DYADIC COMPRESS
	BAL,LX	LISTCHK	CHECK ARGUMENTS FOR LIST
	LI,A	0	SET UP LEFT ARG RANK/SIZE/TYPE
	BAL,LX	SETUPARG	
	LI,X	1	
	CW,X	LFSIZE	LEFT ARG MUST BE EITHER
	BE	15Z0	A 1-ELEMENT ENTITY
	CW,X	LFRANK	OR A VECTOR.
	BNE	ERRANK	
15Z0	BAL,L2	ST3LUPSN	SET UP LOOP PARAMS FROM RTARG, BUT
*			DON'T ALOC RESULT YET.
	LI,NX	0	SET UP FOR SWEEP THRU
	LW,S	LFSIZE	LEFT ARG VALUES.
	BEZ	15Z1	IF IT'S NULL, SKIP IT (SUM=0)
	LW,T	LFTYPE	TYPE
	LW,X	LFARG	
	AW,X	LFRANK	
	AI,X	1	DATA PNTR
	STW,OP	OPTEMP	SAVE OP VALUE
	BAL,LZ	GARAYVAL,T	GET 1ST
	B	15Z01	(DONE)
	AW,NX	AI	ACCUMULATE SUM OF VALUES
	CI,AI	-2	NEXT: MAKE SURE IT'S 0 OR 1
	BAZ	*L1	GET NEXT
	B	ERDOMAIN	
15Z01	LW,OP	OPTEMP	RESTORE OP VALUE
15Z1	LI,S	1	DONE: NX = SUM OF LFARG VALUES
	CW,S	RTSIZE	IS RTARG A 1-ELEMENT THING ?
	BNE	15Z2	NO
	EXU	OPTBL1,OP	YES, SET RESULT SIZE
	STW,NX	RSSIZE	= SUM (COMPRESS), OR
	LW,T	RTTYPE	= LEFT SIZE (EXPAND).
	BAL,L2	VECTORRS	ALOC RS = VECTOR
	B	15Z9	GO GEN XSEG CODE
15Z2	CW,S	LFSIZE	RTARG NORMAL: IS LFARG
	BNE	15Z6,OP	A 1-ELEMENT THING ?
	B	15Z3,NX	YES, WHAT IS ITS VALUE ?
15Z3	B	15Z4,OP	0: SPECIAL CASE FOR COMPRESS
	B	MNOP	1: RESULT = RTARG
15Z4	TABLE	DOPCMPRS	LFARG = 0 (1-ELEMENT)
	B	15Z8	COMPRESS: TREAT AS IF LFSIZE=D(K)
15Z5	CW,NX	MIDLCNT	EXPAND: MAKE SURE THAT
	BNE	ERLENGTH	D(K) = +/(LEFT ARG)
	LW,NX	LFSIZE	AND SET M = LEFT SIZE.
	B	15Z8	
15Z6	TABLE	DOPCMPRS	BOTH ARGS NORMAL. WHAT OP ?
	B	15Z7	COMPRESS
	B	15Z5	EXPAND
15Z7	LW,S	LFSIZE	COMPRESS: MAKE SURE THAT
	CW,S	MIDLCNT	D(K) = LEFT SIZE
	BNE	ERLENGTH	AND USE M = +/(LEFT ARG).
15Z8	SW,NX	MIDLCNT	IF M=D(K),
	BEZ	MNOP	RESULT= RTARG.
	STW,NX	DELTA	SAVE M-D(K)
	LW,L2	RTSIZE	SAVE ACTUAL ARG SIZE
	MW,NX	OUTRCNT	RESULT SIZE DIFFERS FROM RT SIZE BY
	MW,NX	INERCNT	$D(1) \dots D(K-1) * (M-D(K)) * D(K+1) \dots D(N)$.
	AWM,NX	RTSIZE	
	BAL,L1	RSLIKERT	ALOC ARG LIKE RTARG (EXCEPT
	STW,L2	RTSIZE	RESTORE ARG SIZE
	BAL,LX	MRTDIMS	FOR SIZE); COPY RT DIMENS.
	LW,K	COORDK	CHANGE K'TH DIMEN OF RESULT
	AI,K	1	FROM D(K) TO M BY ADDING
	LW,R	DELTA	D(K)-M TO IT.
	AWM,R	*RESULT,K	
15Z9	BAL,L1	SETN1KN	SET UP LFADR FOR SEQ ACCESS (N1),
*			RTADR FOR IND ACCESS (K),
*			AND RSADR FOR SEQ ACCESS (N).
	BAL,LX	GXSEGINI	INIT XSEG; EXIT IF RESULT NULL
	GEN,0,4	CODE12	GEN LOOP INIT CODE
	LI,T	LOGL	SET TO CONVERT LFARG TO LOGL
	STW,T	RSTYPE	
	LI,A	0	GEN LOAD/CONVERT LEFT ARG ELMT
	BAL,L1	GENLOAD	
	GEN,0,2	CODE14	GEN TEST/INNER LOOP INIT
	STW,XL	LOOPLOC	REMEMBER INNER LOOP LOC

	LW, T	RTTYPE	
	LW, R	LOADINST, T	GEN LOAD (RIGHT),
	AW, R	RTADR	
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ
	LW, R+1	STORINST, T	STORE,
	AW, R+1	RSADR	
	GEN, 2, 2	R, CODE15	AND LOOP CONTROL CODE.
	LI, S	1	
	LW, OP	OPTEMP	RESTORE OP CODE AGAIN
	CW, S	RTSIZE	IF RTARG IS 1-ELEMENT,
	BNE	15Z10, OP	MODIFY CODE TO EXCLUDE
	MTW, 1	-2, XL	K-INCREMENTS.
	MTW, 1	-6, XL	
15Z10	B	15Z10, OP	
	TABLE	DOPCMPRS	
	B	15Z11	COMPRESS: XSEG COMPLETED
	LI, AF1	X'1FFFF'	EXPAND: CHANGE TEST INST
	LW, AF	XL	ADR FROM COMPRES2 TO HERE.
	STS, AF	-6, XL	
	LW, R	CODE17	GEN INNER LOOP
	LW, T	RTTYPE	WHICH STORES NULL ELEMENTS
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ
	LW, R+1	NULLINST, T	IN RESULT.
	LW, R+2	STORINST, T	
	AW, R+2	RSADR	
	GEN, 3, 2	R, CODE18	
*			
15Z11	LW, R	LFLGLADR	FETCH LOGL PARAMS FOR USE TO
	LW, R+1	LFLGLCNT	RE-INIT LOGL.
	B	EXECUTE	EXECUTE XSEG
*			
*			
OPTBL1	TABLE	DOPCMPRS	EXU TABLE - BY OP:
	NOP		COMPRESS
	LW, NX	LFSIZE	EXPAND
*			
OPTEMP	TEMP		TEMP FOR OP CODE
*			
*			
NULLINST	TABLE	LOGL	LOAD NULL INST - BY TYPE:
	LI, AI	0	LOGL
	LI, AI	' '	CHAR
	LI, AI	0	INTG
	LD, AF	FLOT0	FLOT
*			
*			
*			
*			
		XSEG CODE:	
*			
CODE12	LCW, N	RSSIZE	0 INIT STORE INDEX
	LI, K	0	1 INIT RTARG FETCH INDEX
	STW, R	LFLGLADR	2 RESTORE INITIAL LOGL PARAMS
	STW, R+1	LFLGLCNT	3
	LCW, N1	LFSIZE	4 INIT MIDDLE LOOP
*	LOAD/CNV	LFARG(N1)	5 LOAD LFARG VALUE (0 OR 1)
CODE14	BEZ	COMPRES2	IF ZERO, GO TO SKIP (NULL-FILL) CODE
	LW, N2	INERCNT	ONE: INIT INNER LOOP
*LOOPLOC	LOAD	RTARG(K)	COPY RTARG ELMTS TO RESULT
*	STORE	RESULT(N)	
CODE15	BIR, N	COMPRES1	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
COMPRES1	AI, K	1	* BUMP FETCH INDEX
	BDR, N2	*LOOPLOC	* COUNT INNER LOOP
	BIR, N1	XSEGBASE+5	* COUNT MIDDLE LOOP
	B	EXPAND3	* RE-START IF APPROPRIATE
*			
COMPRES2	AW, K	INERCNT	* ZERO: SKIP OVER INNER LOOP
	BIR, N1	XSEGBASE+5	* COUNT MIDDLE LOOP
	B	EXPAND3	* RE-START IF APPROPRIATE
*			
	...OR...		
CODE17	LW, N2	INERCNT	ZERO: INNER LOOP ...
*LOOP4	LOAD	NULL	COPIES NULLS
*	STORE	RESULT(N)	TO RESULT.
CODE18	BIR, N	EXPAND2	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
EXPAND2	BDR, N2	-4, XL (LOOP4)*	COUNT INNER LOOP
	BIR, N1	XSEGBASE+5	COUNT MIDDLE LOOP
EXPAND3	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.

	B PAGE	XSEGBASE+2	* RE-START	
*				
DTAKE	EQU	%	DYADIC TAKE	
DDROP	EQU	%	DYADIC DROP	
	STW, OP	OPTEMP	SAVE OP CODE	
	LI, A	1	SET UP RIGHT ARG'S RANK/SIZE/TYPE,	
	BAL, LX	SETUPARG	CONVERTING ISEQ TO INTG VECTOR.	
	LW, X	LFARG	GET LFARG POINTER	
	LI, NX	1		
	CB, NX	*LFARG, NX	LEFT ARG MUST BE EITHER	
	BL	ERRANK		
	BG	17Z0	A SCALAR,	
	AI, X	1	OR A VECTOR OF LENGTH	
	LW, NX	1, X	= RIGHT ARG RANK.	
17Z0	STW, NX	LFSIZE	SAVE LEFT ARG SIZE	10-00008
	LI, R	=1	IF RT ARG SCALAR, SET 'ARGDIMPT'	10-00009
	LW, L3	RTRANK	TO POINT TO '1' AND L3=0;	U10-0074
	BEZ	17Z01	OTHERWISE, 'ARGDIMPT' POINTS	10-00011
*			TO ARG DIMENS, L3=-1 TO BUMP IT.	U10-0076
*	CW, NX	RTRANK	RT ARG NOT SCALAR: LF ARG SIZE	10-00013
			MUST AGREE.	10-00014
	BNE	ERLENGTH		
	LW, R	RTARG	SET UP TO ACCESS RT ARG	
	AW, R	RTRANK	DIMENS D(N), ..., D(1).	10-00016
	AI, R	1		10-00017
	LI, L3	-1	..TO BUMP DIMEN PTR	10-00018
17Z01	STW, R	ARGDIMPT		10-00019
	LB, T	*LFARG		
	LI, BI	1	INIT RS SIZE	
	AI, X	1	POINT TO LF DATA -1	
	LW, S	LFSIZE	GET SIZE OF LF ARG	10-00021
	BNEZ	17Z1, OP	ENTER TAKE/DROP LOOP UNLESS LFSIZE=	
	B	MNOP	=RTRANK=0, IN WHICH CASE, DO NO-OP.	
17Z1	TABLE	DOPTAKE		
	B	17Z4	TAKE	
	BAL, LZ	GARAYVAL, T	DROP: GET A(N) VALUE	
	B	17Z6	(DONE)	
	STW, AI	DIMENBUF, NX	SAVE IT IN DIMEN BUF	
	BGEZ	17Z2	J'TH RESULT DIMEN R(J)	
	AW, AI	*ARGDIMPT	= MAX(0, D(J)-ABS(A(J))).	10-00023
	BGEZ	17Z3		10-00024
	B	17Z21		10-00025
17Z2	SW, AI	*ARGDIMPT	AI = +-R(J)	10-00026
	BLEZ	17Z3		10-00027
17Z21	LI, AI	0		10-00028
17Z3	MW, BI	AI	ACCUMULATE +-RSSIZE	
	BN OV	17Z5	= +-R(1)*...*R(N); GET NEXT A(J)	
	B	ERDOMAIN		
17Z4	BAL, LZ	GARAYVAL, T	TAKE: GET A(N) VALUE	
	B	17Z6	(DONE)	
	STW, AI	DIMENBUF, NX	SAVE IT IN DIMEN BUF	
	MW, BI	AI	R(J) = ABS(A(J)); ACCUM +-RSSIZE	
	BOV	ERDOMAIN	= +-A(1)*...*A(N).	
17Z5	AWM, L3	ARGDIMPT	BUMP DIMEN POINTER	10-00030
	BDR, NX	*L1	REDUCE J, GET NEXT A(J)	10-00031
17Z6	LW, OP	OPTEMP	DONE, RESTORE OP CODE	
	LAW, S	BI	RSSIZE = R(1)*...*R(N)	
	STW, S	RSSIZE		
	LW, T	RTTYPE	RESULT TYPE/RANK	
	ISEQFIX, T		USE TYPE INTG INSTEAD OF ISEQ	
	STW, T	RSTYPE	= RTARG TYPE/RANK.	10-00033
	LW, R	LFSIZE		
	STW, R	RSRANK		
	BAL, L1	ALOCRS	ALOC RS DATA BLOCK	
	LW, S	RSRANK	ALOC LOOP CONTROL BLOCK	
	BAL, L2	INDEXB	AND INIT LOOP PARAMS.	
	AI, L3	0	IF RT ARG NOT SCALAR,	10-00035
	BEZ	17Z7	RE-INIT ARG DIMEN PTR.	10-00036
	LW, R	RTARG		
	AW, R	RTRANK		10-00038
	AI, R	1		10-00039
	STW, R	ARGDIMPT	THE D(J)	
17Z7	EQU	%		10-00041
	LW, R	RESULT	AND THE R(J).	
	AI, R	1		
	STW, R	DIMENPTR		

```

        LW,X      RSRANK      INIT LOOP: J=N;
        LW,KB     INNERMOST B-BLOCK;
        LI,AI     1           NULL COUNT = R(J+1)*...*R(N);
        LI,BI     1           WEIGHT = D(J+1)*...*D(N).
*
*   SET UP LOOP CONTROL BLOCK AND RESULT DIMENS
*
17Z8      LAW,S    DIMENBUF,X   GET ABS(A(J))
        B        17Z9,OP      WHICH OP?
17Z9      TABLE  DOPTAKE
        B        17Z10
        LCW,S     S           TAKE
        AW,S      *ARGDIMPT    DROP: COMPUTE R(J)=D(J)-ABS(A(J))
        BGEZ     17Z91        10-00043
        LI,S      0           IF D(J)<ABS(A(J)),
        EQU      %           R(J)=0.
        STW,S     *DIMENPTR,X   STORE R(J) IN RESULT DIMEN
        STW,S     BINITCNT,KB   COUNT OF LOOP(J) = R(J)
        LW,S      DIMENBUF,X   GET A(J); IF A(J)>0, INCREASE
        BGZ      17Z13        FIXED-PART VAL BY A(J)*WEIGHT.
        B        17Z14
17Z10     STW,S     *DIMENPTR,X   TAKE: R(J)=ABS(A(J)); STORE IT AS
        STW,S     BINITCNT,KB   RESULT DIMEN AND LOOP COUNT.
        LW,S      DIMENBUF,X   GET A(J)
        BLZ      17Z12        A(J)<0: BRANCH
        CW,S      *ARGDIMPT    A(J)>=0: COMPARE WITH D(J)
        BLE      17Z14        10-00048
        LW,S      =X'7FFFFFFF'  A(J)>D(J): SPECIFY 'D(J) ELEMENTS
        SW,S      *ARGDIMPT    FOLLOWED BY A(J)-D(J) NULLS'.
        STW,S     NULTSBUF,X    SET UP 'MIXED SEQUENCE' LOOP:
        STW,AI    NULCTBUF,X    NULL-TEST VAL IN 'NULTSBUF' AND
        LI,S      MIXLOOP      NULL GROUP COUNT IN 'NULCTBUF'.
        STB,X     S           INCLUDE BUFFER INDEX IN OP CODE WORD.
        B        17Z15
17Z12     AW,S      *ARGDIMPT    A(J)<0: COMPUTE A(J)+D(J)
        BLZ      17Z11        10-00052
        A(J)<-D(J): SPECIFY '-D(J)-A(J)
*        NULLS, FOLLOWED BY D(J) ELMTS'.
17Z13     MW,S      BI          INCR FIXED PART VAL BY A(J)*WEIGHT
        AWM,S     FVALUE      (DROP) OR (A(J)+D(J))*WEIGHT (TAKE
17Z14     LI,S      SELOOP      LOOP TYPE = ORDINARY SEQUENTIAL
17Z15     STW,S     BLOOPTYP,KB
        STW,BI    BSTEPVAL,KB
        MW,BI     *ARGDIMPT    STEP VALUE =WEIGHT=D(J+1)*...*D(N)
        MW,AI     *DIMENPTR,X  UPDATE WEIGHT FACTOR (NO OVFL POSS)
        AI,KB     -BN          UPDATE NULL COUNT =R(J+1)*...*R(N)
        AWM,L3    ARGDIMPT    POINT TO NEXT OUTER B-BLOCK
        BDR,X     17Z8        BUMP DIMEN PNTR (IF NOT SCALAR)
                                10-00056
                                DECR J, DO NEXT COORD
*
*   COORD LOOP DONE.  NOW BUILD AND EXECUTE XSEG
*
        LI,A      1           SET RTARG FOR INDICIAL LOAD
        BAL,LX    XSETUP
        LI,X      -1          SET RESULT FOR SEQUENTIAL STORE
        BAL,LX    SETADR
        BAL,LX    GXSEGINI    INIT XSEG; EXIT IF RESULT NULL
        GEN,0,2   CODE20      GEN LOOP INIT CODE
        LW,T      RTTYPE
        LW,R      LOADINST,T  GEN LOAD,
        AW,R      RTADR
        ISEQFIX,T  USE TYPE INTG INSTEAD OF ISEQ
        LW,R+1    STORINST,T  STORE,
        AW,R+1    RSADR
        GEN,2,2   R,CODE21    LOOP CONTROL CODE.
        AI,OP     -DOPTAKE    IF IT'S 'DROP', THE XSEG IS FINISHED
        BNEZ     EXECUTE
        LW,R      NULLINST,T  TAKE: GEN ADDITIONAL CODE
        GEN,2,4   R,CODE22    TO COPY NULLS TO RESULT.
        B        EXECUTE
*
*   XSEG CODE:
*
CODE20     LCW,N    RSSIZE      0  INIT STORE INDEX
        LI,L2     XSEGBASE+7   1  SET NULL-COPY ADR (FOR 'TAKE' ONLY)
        BAL,L1    1STSCRPT     2  GET 1ST LOAD INDEX
        LOAD      RTARG(K)     3  COPY SELECTED ELEMENT
        STORE     RESULT(N)    4  TO RESULT.
CODE21     BIR,N   NXTSCRPT     5  BUMP STORE INDEX; GET NEXT LOAD INDX

```

	B	*RETURN	6	EXIT WHEN RESULT FULL
*	LOAD	NULL	7	COPY NULL ELEMENT
*	STORE	RESULT(N)	8	TO RESULT.
CODE22	BIR,N	XSEGBASE+11	9	BUMP STORE INDEX
	B	*RETURN	10	EXIT WHEN DONE
	BDR,N2	XSEGBASE+7	11	COUNT NULLS
	B	NXTSCRPT	12	RETURN FOR NEXT LOAD INDEX VALUE
*				
NULTSBUF	EQU	INBUF+0		NULL-TEST VALUE BUFFER
NULCTBUF	EQU	INBUF+64		NULL-GROUP COUNT BUFFER
	PAGE			
*				
*				
DCATEN	EQU	%		DYADIC CATENATE
	LI,X	1		
	LB,R	*LFARG,X		
	CB,R	*RTARG,X		PLACE HIGHER RANKED
	BLE	18Z0		ARG IN RTARG
	EXCHANGE	ARGS		
	AI,OP	-1**17		IF ARGS SWAPPED, SET OP'S SIGN BIT
18Z0	LI,A	0		
	BAL,LX	SETUPARG		SET LFRANK/SIZE/TYPE
	LI,A	1		
	BAL,LX	SETUPARG		SET RTRANK/SIZE/TYPE
	BAL,LX	TYCOMPAT		CHECK COMPATIBILITY; SET RS TYPE
	B	18Z02		GO RESOLVE NUMERIC VS TEXT
18Z01	LW,S	LFSIZE		
	AW,S	RTSIZE		RESULT SIZE = SUM OF ARG SIZES
	STW,S	RSSIZE		
	LW,R	RTRANK		SET RESULT RANK
	STW,R	RSRANK		= HIGHEST ARG RANK.
	BNEZ	18Z1		ARE BOTH ARGS SCALARS?
	BAL,L2	VECTORRS		YES: ALOC VECTOR RESULT (SIZE=2)
	LI,R	1		SET COUNTS TO 1
	STW,R	LFCOUNT		
	STW,R	RTCOUNT		
	B	18Z12		JOIN MAIN CASE
18Z02	LW,S	LFSIZE		IS 'LFARG' EMPTY...
	BEZ	18Z03		YES, USE TYPE OF 'RTARG'
	LW,T	LFTYPE		NO, SET FOR TYPE OF 'LFARG'
	LW,S	RTSIZE		IS 'RTARG' EMPTY...
	BEZ	18Z04		YES, USE TYPE OF 'LFARG'
	B	ERDOMAIN		NO, CONFLICT-- DOMAIN ERROR
18Z03	LW,T	RTTYPE		SET FOR TYPE OF 'RTARG'
18Z04	ISEQFIX,T			USE TYPE INTG INSTEAD OF ISEQ
	STW,T	RSTYPE		
	B	18Z01		OF (HOPEFULLY) NON-EMPTY ARG
18Z1	LB,K	OPER+1		CHECK LAMINATION BIT
	CI,K	LAMBIT		
	BAZ	18Z2		OFF (CATENATION)
	AI,OP	2		ON (LAMINATION): MODIFY OP CODE
	AI,K	-LAMBIT+1		GET RID OF LAM. BIT, BUMP COORD
	STB,K	OPER+1		SPEC TO CEILING OF GIVEN COORD.
	LW,S	RTSIZE		SET RESULT SIZE
	SLS,S	1		=2* RIGHT ARG SIZE.
	STW,S	RSSIZE		
	LW,R	LFRANK		
	BEZ	18Z15		UNLESS LFARG IS SCALAR,
	CW,R	RTRANK		ARG RANKS MUST AGREE
	BNE	ERRANK		
18Z15	MTW,1	RSRANK		RESULT RANK IS 1 HIGHER
	CW,K	RSRANK		COMPARE WITH K
	BL	18Z17		K<=ARG RANK: 'ST3LUPSN' WILL WORK
	BNE	ERCOORD		K> ARG RANK+1: ERROR
	STW,K	COORDK		K= ARG RANK+1: SIMULATE ST3LUPSN
	LI,A	1		
	STW,A	MIDLCNT		
	STW,A	MIDLSTEP		
	STW,A	OUTRSTEP		
	B	18Z6		JOIN OTHER CASE
18Z17	BAL,L2	ST3LUPSN		SET LOOP PARAMS; CHECK K
	B	18Z6		
18Z2	LW,R	LFRANK		EITHER LFARG IS SCALAR,
	BEZ	18Z3		
	CW,R	RTRANK		OR RANKS AGREE,
	BE	18Z4		
	AI,R	1		OR THEY MUST DIFFER BY 1.
	CW,R	RTRANK		

	BNE	ERRANK	
18Z3	AI, OP	1	MODIFY OP TO SAY 'DIFFERENT RANKS'
18Z4	BAL, L2	ST3LUPSN	SET LOOP PARAMS FOR RTARG; CHECK K;
	LW, R	LFRANK	IF LFARG ISN'T SCALAR,
	BNEZ	18Z6	WE'VE ALREADY GOT RSSIZE.
	LW, S	OUTRCNT	IF IT IS, WE MUST SET
	MW, S	INERCNT	RSSIZE = D(1)*...
	BOV	ERLENGTH	10-00063
	AW, S	RTSIZE	*(D(K)+1)*...*D(N).
	BOV	ERLENGTH	10-00065
	STW, S	RSSIZE	
18Z6	LW, S	RSSIZE	GENERAL CASE:
	BAL, L1	ALOCRS	ALOC RS DATA BLOCK
	LW, X	RSRANK	SET UP TO SCAN DIMENS
	B	18Z7, OP	
18Z7	TABLE	DOPCATEN	MODIFY PNTR(S) - BY OP:
	MTW, 1	LFARG	CAT, RANKS SAME: ALIGN PNTRS
	MTW, 1	RTARG	CAT, RANKS DIFF: OFFSET PNTRS
	MTW, 1	RESULT	LAM: PNTRS ALIGNED, 1 LOW
18Z8	LW, R	*RTARG, X	DIMEN LOOP: GET D(I)
	CW, X	COORDK	IS I=K ?
	BNE	18Z13	NO, GO CHECK C(I)=D(I)
	EXU	OPTBL2, OP	YES, COMPUTE VALUE OF LEFT COUNT
	MW, S	MIDLSTEP	NOW, WHILE C(K) AND D(K)
	STW, S	LFCOUNT	ARE HANDY.
	EXU	OPTBL3, OP	COMPUTE K'TH RESULT DIMEN
	B	18Z9, OP	DO APPROPRIATE PNTR MODIFY
18Z9	TABLE	DOPCATEN	
	B	18Z11	CAT, RANKS SAME: PNTRS ALIGNED
	B	18Z10	CAT, RANKS DIFF: ALIGN ARG PNTRS
	MTW, 1	RTARG	LAM: CORRECT FOR 1 EXTRA RS DIMEN
18Z10	MTW, 1	LFARG	
18Z11	STW, R	*RESULT, X	STORE I'TH RESULT DIMEN
	BDR, X	18Z8	COUNT DIMEN LOOP
	MTW, -1	LFARG	DONE: RESTORE PNTRS
	MTW, -1	RTARG	
	MTW, -1	RESULT	
18Z12	STW, OP	OPTEMP	SAVE OP CODE
	BAL, L1	SETN1KN	SET ADDRESSES
	BAL, LX	GXSEGINI	PREPARE TO GEN XSEG CODE
	LW, A	LFRANK	IF LFARG IS SCALAR
	BNEZ	18Z16	GEN LOAD/CONVERT OF
	BAL, L2	GENLOADT	LFARG TO ITS TEMP.
18Z16	GEN, 0, 5	CODE23A	GEN LOOP INIT CODE
	STW, XL	LFLOOP	SAVE LOC OF LEFT ARG LOOP
	LI, A	0	
	BAL, L1	GENLOAD	GEN LOAD/CONVERT OF LFARG(N1)
	LW, T	RSTYPE	
	LW, R	STORINST, T	GEN STORE RESULT(N)
	AW, R	RSADR	
	STW, R	TEMP	
	GEN, 1, 2	R, CODE23	GEN LOOP CONTROL
	STW, XL	RTLLOOP	SAVE LOC OF RIGHT ARG LOOP
	LI, A	1	
	BAL, L1	GENLOAD	GEN LOAD/CONVERT OF RTARG(K)
	LW, R	TEMP	GEN STORE RESULT(N),
	GEN, 1, 2	R, CODE24	LOOP CONTROL
	B	EXECUTE	EXECUTE XSEG
18Z13	CW, R	*LFARG, X	DIMEN LOOP: FOR I/=K, WE
	BE	18Z11	REQUIRE C(I)=D(I).
	LW, S	LFRANK	DONT ACTUALLY DO COMPARISON
	BEZ	18Z11	IF LFARG IS SCALAR.
	CW, X	COORDK	ERROR, IF I>K, PNTRS ARE STILL AS
	BG	18Z14, OP	INITIALIZED, AND MUST BE SELECT-
*			IVELY FIXED. IF I<K, THEY'RE
18Z14	TABLE	DOPCATEN	ALL OFF BY 1.
	MTW, -1	LFARG	
	MTW, -1	RTARG	
	MTW, -1	RESULT	
	B	ERLENGTH	REPORT ERROR
*			
*			
*			
*	LCW, N	RSSIZE	INIT STORE INDEX
*	LOAD/CNV	LFARG	PUT LFARG IN TEMP (ONLY IF
*	STORE	LFTEMP	IT'S SCALAR).
CODE23A	LCW, N1	LFSIZE	INIT LEFT FETCH INDEX
	LI, K	0	INIT RIGHT FETCH INDEX

	LW, OP	OPTEMP	WERE ARGS SWAPPED ?
	BGZ	CATEN4	NO, START WITH LEFT LOOP
	B	CATEN2	YES, START WITH RIGHT LOOP
*LFL00P	LOAD/CNV	LFARG(N1)	MOVE LEFT ARG ELMTS
*	STORE	RESULT(N)	TO RESULT.
CODE23	BIR, N	CATEN1	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
CATEN1	AI, N1	1	* BUMP AND
	BDR, N2	*LFL00P	* COUNT LEFT LOOP.
CATEN2	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.
	LW, N2	RTCOUNT	* INIT RIGHT LOOP
	BNEZ	*RTL00P	* GO TO RIGHT LOOP
	B	CATEN4	* RTCOUNT=0, GO TO LEFT LOOP
*RTL00P	LOAD/CNV	RTARG(K)	MOVE RIGHT ARG ELMTS
*	STORE	RESULT(N)	TO RESULT.
CODE24	BIR, N	CATEN3	COUNT RESULT
	B	*RETURN	EXIT WHEN FILLED
CATEN3	AI, K	1	* BUMP AND
	BDR, N2	*RTL00P	* COUNT RIGHT LOOP.
CATEN4	CW, N	DELRSIZE	* IF RESULT IS CHAR, DON'T TRY
	BGE	*RETURN	* TO FILL OUT TRAILING GAP.
	LW, N2	LFCOUNT	* INIT LEFT LOOP
	BNEZ	*LFL00P	* GO TO LEFT LOOP
	B	CATEN2	* LFCOUNT=0, GO TO RIGHT LOOP
*			
*			
LFCOUNT	TEMP		LEFT LOOP COUNT
RTCOUNT	EQU	OUTRSTEP	RIGHT LOOP COUNT (SET BY ST3LUPSN)
LFL00P	TEMP		LEFT ARG LOOP LOC
RTL00P	EQU	LOOPLOC	RIGHT ARG LOOP LOC
*			
*			
OPTBL2	TABLE	DOPCATEN	LFCOUNT COMPUTATION - BY OP:
	LW, S	*LFARG, X	CAT, RANKS SAME: C(K)*D(K+1)*...
	LI, S	1	CAT, RANKS DIFF: 1*D(K+1)*...
	LW, S	MIDLCNT	LAM: D(K)*D(K+1)*...
*			
OPTBL3	TABLE	DOPCATEN	K'TH RS DIMEN COMPUTATION - BY OP:
	AW, R	*LFARG, X	CAT, RANKS SAME: C(K)+D(K)
	AI, R	1	CAT, RANKS DIFF: 1+D(K)
	LI, R	2	LAM: (EXTRA DIMEN)= 2
	PAGE		

INDEXING SETUP ROUTINES

BUILD L-BLOCK FROM SUBSCRIPT LIST

THIS SUBROUTINE ANALYSES THE SUBSCRIPT LIST (PNTR IN 'SCRIPT') AND BUILDS A LOOP CONTROL BLOCK SUITABLE FOR USE BY 'NXTSCRIPT'. THE STRUCTURE OF THIS BLOCK IS AS FOLLOWS:

1. B (ONE 6-WORD BLOCK PER SCRIPT). EACH B-BLOCK HAS ONE OF THE FOLLOWING FORMATS. DEPENDING ON THE SCRIPT STRUCTURE:

```

NULL:  STEP,  SIZE,  -,  -,  -,  0
ISEQ:  STEP,  SIZE,  -,  -,  -,  0
SCALAR: -----NONE-----
ARRAY:  VPNTR,  SIZE,  -,  -,  -,  -1

```

WHERE VPNTR = OFFSET INTO V-BLOCK WHERE SCRIPT VALUES ARE LOCATED.

2. V (VARIABLE LENGTH): CONTAINS SCRIPT VALUES FOR ALL ARRAY SCRIPTS.

THE RESULT DIMENSIONS (= ALL THE SCRIPT DIMENS, CATENATED) ARE PLACED IN THE DIMENSION BUFFER. EACH SCRIPT PLACES STUFF IN THE DIMEN BUFFER AS FOLLOWS:

```

NULL:  SIZE OF CURRENT DIMEN
ISEQ:  SIZE
SCALAR: NOTHING

```

			ARRAY: ARRAY DIMENS	
			THE ORIGINAL SCRIPT LIST POINTER IS DE-REFFED, AND THE	
			NEW L-BLOCK POINTER IS PLACED IN 'LBLOCK'.	
			THIS ROUTINE ALSO ESTABLISHES RESULT SIZE AND RANK	
			IN RSRANK/RSSIZE. LINK IS L2.	
INDEXA	EQU	%		
	LI,S	0		U10-0078
	STW,S	RSRANK	INIT SCRIPT RANK TO ZERO	U10-0079
				U10-0080
	TEST FOR SPECIAL CASE: VECTOR ARG WITH SCALAR SUBSCRIPT			U10-0081
				U10-0082
	STW,S	OLDLBLOK	SET OLDLBLOK = LBLOCK (=0)	U10-0083
			FOR UPDATELP.	U10-0084
	STW,S	INERBLOK	SET INERBLOK<OUTRBLOK FOR 1STSCRIPT	U10-0085
	LI,A	1		U10-0086
	STW,A	OUTRBLOK		U10-0087
	STW,A	RSSIZE	INIT SCRIPT SIZE TO 1	U10-0088
	LW,X	SCRIPT	IN ORDER TO HAVE THE SPECIAL CASE,	U10-0089
	CW,A	2,X	THERE MUST BE EXACTLY 1 SCRIPT,	U10-0090
	BNE	1Z0		U10-0091
	CB,A	*RTARG,A	THE IND.VAR. MUST BE A VECTOR,	U10-0092
	BNE	ERRANK		U10-0093
	LW,X	3,X	AND THE SCRIPT	U10-0094
	BEZ	1Z0	(NOT ELIDED)	U10-0095
	LB,R	*X,A	MUST BE A SCALAR.	U10-0096
	BNEZ	1Z0		U10-0097
				U10-0098
	SPECIAL CASE: NO LOOP CONTROL BLOCK NEEDED			U10-0099
				U10-0100
	LB,T	*X		U10-0101
	BAL,LZ	GSCLRVAL,T	GET SUBSCRIPT VALUE	U10-0102
	SW,AI	ORIGIN	CONVERT TO ORIGIN-0	U10-0103
	BLZ	ERINDEX	REQUIRE:	U10-0104
	LW,A	RTARG	0<=SCRIPT<DIMEN	U10-0105
	CW,AI	2,A		U10-0106
	BGE	ERINDEX		U10-0107
	STW,AI	FVALUE	THAT'S THE WHOLE SCRIPT	U10-0108
	B	DREFSCR	DE-REF SCRIPT, EXIT	U10-0109
1Z0	EQU	%		U10-0110
				U10-0111
	GENERAL CASE: COMPUTE SIZE OF LOOP CONTROL BLOCK, ALLOCATE IT			U10-0112
				U10-0113
	STW,S	BLENGTH		
	STW,S	VLENGTH		
	LW,X	SCRIPT	CREATE POINTER TO SCRIPT LIST +2	
	AI,X	2	(WORD BEFORE 1ST SUBSCRIPT PNTR)	
	STW,X	SCRIPT2		
	LW,NX	*SCRIPT2	GET NUMBER OF SUBSCRIPTS	
1Z1	LW,X	*SCRIPT2,NX	GET SCRIPT PNTR (STARTING WITH LAST)	
	BNEZ	1Z3	IS IT ZERO (I.E., NULL) ?	
1Z2	MTW,1	RSRANK	YES, NULL/ISEQ: INCREASE RANK BY 1	
	MTW,BN	BLENGTH	AND RESERVE ONE ITEM FOR B-BLOCK.	
	BDR,NX	1Z1	SCAN NEXT SUBSCRIPT	
	B	1Z5	DONE	
1Z3	LI,T	ISEQ	NOT NULL, IS IT AN 'INDEX SEQUENCE'?	
	CB,T	*X		
	BE	1Z2	YES, RESERVE A WORD FOR D	
	LI,T	1	NO, IS IT A SCALAR ?	
	LB,R	*X,T		
	BEZ	1Z4	YES	
	AWM,R	RSRANK	NO,ARRAY: BUMP RANK BY SCRIPT RANK	
	AI,X	1		
	LI,S	1		
1Z6	MW,S	*R,X	COMPUTE SCRIPT SIZE; RESERVE	
	BDR,R	1Z6		
	AWM,S	VLENGTH	SPACE IN V FOR SCRIPT VALUES.	
	MTW,BN	BLENGTH	RESERVE SPACE FOR B-BLOCK	
1Z4	BDR,NX	1Z1	SCAN NEXT SCRIPT	
1Z5	LW,S	BLENGTH	COMPUTE TOTAL L-BLOCK SIZE:	
	AW,S	VLENGTH	L = B+V	
	BAL,LX7	ALOCHNW	ALLOCATE L DATA BLOCK; A = PNTR	
	STW,A	LBLOCK	COPY L PNTR	
	STW,A	OLDLBLOK	SAVE FOR 'UPDATELP'	
	LI,R	INTG**8	SET TYPE=INTG, RANK=0	
	STH,R	*LBLOCK	(TYPE FIELD MUST BE LEGIT)	

	AI, A	2	BUMP PAST HEADER WORDS
	AW, A	BLENGTH	INIT B AND V PNTRS AT 1 + THE
	STW, A	BPNTR	ENDS OF THEIR RESPECTIVE
	AW, A	VLENGTH	REGIONS.
	STW, A	VPNTR	
	LI, A	DBUFEND	INIT DIMEN PNTR
	STW, A	DIMENPTR	
*			
*	SCAN SUBSCRIPTS, SET UP LOOP CONTROL DATA IN L-BLOCK		U10-0116
*			
	LW, X	SCRIPT	RE-CREATE SCRIPT LIST PNTR +2
	AI, X	2	(ALOCHNW MAY HAVE MOVED
	STW, X	SCRIPT2	THE SCRIPT LIST).
	LW, X	RTARG	CREATE POINTER TO RTARG DIMENS
	AI, X	1	
	STW, X	ARGDIMPT	
	LW, X	BPNTR	CREATE POINTER TO INNERMOST
	AI, X	-BN	B-BLOCK
	STW, X	INERBLOK	
	LI, X	1	
	LB, NX	*RTARG, X	GET RIGHT ARG'S RANK
	CW, NX	*SCRIPT2	MAKE SURE IT AGREES WITH
	BNE	ERRANK	NUMBER OF SUBSCRIPTS.
	LI, R	0	INITIALIZE:
	STW, R	FVALUE	FIXED PART VALUE = 0
	LI, S	1	
	STW, S	ISEQFLAG	ISEQ FLAG = .FALSE.
	STW, S	RSSIZE	RESULT SIZE = 1
	STW, S	WEIGHT	WEIGHT FACTOR = 1
2Z1	LW, X	*SCRIPT2, NX	GET NEXT SCRIPT PNTR
	BNEZ	2Z2	IS IT NULL ?
	LW, S	*ARGDIMPT, NX	YES, SIZE= CORRESPONDING DIMEN
	LW, AI	WEIGHT	WEIGHTED STEP = 1*WEIGHT
	B	3Z2	CONTINUE LIKE ISEQ
2Z2	LB, T	*X	GET TYPE CODE
	CI, T	ISEQ	HANDLE ISEQ SPECIALLY
	BE	3Z1	
	LI, AI	1	
	LB, R	*X, AI	GET RANK; IS IT SCALAR ?
	BEZ	2Z16	YES
	STW, R	XRANK	REMEMBER RANK
	LI, S	1	NO, INIT SIZE = 1
	AI, X	1	POINT TO WORD BEFORE 1ST DIMEN
2Z3	LW, AI	*R, X	COPY NEXT SCRIPT DIMEN
	MTW, -1	DIMENPTR	INTO DIMEN BUF,
	STW, AI	*DIMENPTR	
	ODD, S		
	MW, S	*DIMENPTR	AND ACCUMULATE SIZE.
	BDR, R	2Z3	PROCESS ALL DIMENS
	AW, X	XRANK	POINT TO WORD BEFORE 1ST VALUE WORD
	STW, S	XSIZE	SAVE SCRIPT SIZE
	AI, S	0	
	BEZ	3Z5	IF SIZE=0; NO B-BLOCK, DO NEXT SCRPT
	BAL, LZ	GARAYVAL, T	SIZE>0, GET ARRAY VALUES (INTG)
	B	2Z11	(DONE)
	BAL, LX	CHEKSUBS	OFFSET/CHECK SUBSCRIPT
	MW, AI	WEIGHT	APPLY WEIGHT FACTOR
	MTW, -1	VPNTR	STORE IT IN V-BLOCK
	STW, AI	*VPNTR	
	B	*L1	GET NEXT VALUE
2Z11	LW, S	XSIZE	FINISH: GET SCRIPT SIZE
	STW, LZ	ISEQFLAG	RESET ISEQ FLAG (>0)
	LW, AI	VPNTR	GET V-POINTER
	SW, AI	LBLOCK	CONVERT TO L-BLOCK OFFSET VALUE
	LI, L1	ARAYLOOP	SET LOOP TYPE = ARRAY
	B	3Z4	GO CREATE B-BLOCK ITEM
*			
2Z16	BAL, LZ	GSCLRVAL, T	GET SCALAR VALUE
	BAL, LX	CHEKSUBS	OFFSET/CHECK SUBSCRIPT VALUE
	STW, LX	ISEQFLAG	RESET ISEQ FLAG (>0)
	ODD, AI		
	MW, AI	WEIGHT	INCR FIXED PART VALUE
	AWM, AI	FVALUE	BY WEIGHTED SCRIPT VALUE.
	B	3Z6	PROCESS NEXT SCRIPT
*			
3Z1	LW, AI	2, X	ISEQ: CHECK RANGE OF LAST VALUE:
	BEZ	3Z7	
	MW, AI	4, X	= A+B*N

	AW, AI	3, X	
	BAL, LX	CHEKSUBS	
	LW, AI	3, X	OFFSET/CHECK FIRST VALUE:
	AW, AI	4, X	= A+B
	BAL, LX	CHEKSUBS	
	MW, AI	WEIGHT	INCR FIXED PART BY WEIGHTED
	AWM, AI	FVALUE	1ST SCRIPT VALUE.
3Z7	LW, S	2, X	SIZE = N
	LW, AI	4, X	STEP = B * WEIGHT
	MW, AI	WEIGHT	
3Z2	MTW, -1	DIMENPTR	ISEQ/NULL: PUT SIZE IN DIMEN BUF
	STW, S	*DIMENPTR	
	LI, L1	SEQLOOP	SET LOOP TYPE = SEQ
	LCW, R	ISEQFLAG	TEST ISEQ FLAG: WAS LAST SCRIPT
	BLZ	3Z3	AN ISEQ?
	LW, T	BPNTR	YES, SEE IF WE CAN COMBINE THIS
	LW, BI	BSTEPVAL, T	ONE WITH THE LAST; THIS IS
	ODD, BI		
	MW, BI	BINITCNT, T	POSSIBLE IFF (LAST SIZE)*(LAST
	CW, BI	AI	STEP) = (THIS STEP).
	BNE	3Z4	RATS.
	LW, AI	S	OH GOOD; MODIFY LAST B-BLOCK BY
	MW, AI	BINITCNT, T	SETTING ITS SIZE =
	STW, AI	BINITCNT, T	(LAST SIZE)*(THIS SIZE).
	B	3Z5	
3Z3	STW, R	ISEQFLAG	ISEQ WAS RESET: SET IT NOW (<0)
3Z4	MTW, -BN	BPNTR	
	LW, T	BPNTR	BUILD NEW B-BLOCK:
	STW, AI	BSTEPVAL, T	INCLUDE ITS WEIGHTED STEP VALUE
	STW, S	BINITCNT, T	(OR V-OFFSET) AND SIZE.
	STW, L1	BLOOPTYP, T	AND TYPE.
3Z5	MW, S	RSSIZE	UPDATE RESULT SIZE
	STW, S	RSSIZE	
3Z6	LW, AI	*ARGDIMPT, NX	UPDATE WEIGHT FACTOR
	MW, AI	WEIGHT	
	STW, AI	WEIGHT	
	BDR, NX	2Z1	DO NEXT SCRIPT
*			
*	DE-REF SCRIPT LIST DATA BLOCK		U10-0118
*			
DREFSCR	EQU	%	U10-0120
	LI, A	0	
	XW, A	SCRIPT	DISCARD SCRIPT LIST PNTR
	BAL, LX7	DREF	DE-REF SCRIPT LIST DATA BLOCK
	B	*L2	RETURN
	PAGE		
*			
*			
*	ALLOCATE LOOP CONTROL BLOCK		
*			
*		CALLED WITH THE NUMBER OF B-BLOCKS NEEDED IN 'S'.	
*		ALLOCATES AN LBLOCK BIG ENOUGH FOR THAT MANY B-BLOCKS	
*		AND AN EMPTY V-BLOCK. SETS UP LBLOCK, OLDBLOCK,	
*		INNERBLOCK, OUTRBLOCK, FVALUE. LINK IS L2.	
*			
INDEXB	EQU	%	
	MTW, -4	RETURN	ALTER RETURN ADR SO THAT, UPON
*			EXIT FROM OP DRIVER, 'LBLOCK'
*			WILL BE DE-REFED
	MI, S	BN	COMPUTE TOTAL SIZE OF B-BLOCKS
	STW, S	TEMP	SAVE IT
	BAL, LX7	ALOCHNW	ALLOCATE L-BLOCK
	STW, A	LBLOCK	SAVE POINTER
	STW, A	OLDBLOCK	(FOR UPDATERP)
	LI, R	INTG**8	SET TYPE (MUSTN'T LEAVE IT 0)
	STH, R	*LBLOCK	
	AI, A	2	
	STW, A	OUTRBLOCK	PNTR TO OUTER B-BLOCK
	AW, A	TEMP	
	AI, A	-BN	
	STW, A	INNERBLOK	PNTR TO INNER B-BLOCK
	LI, R	0	INITIALIZE FIXED PART VALUE
	STW, R	FVALUE	
	B	*L2	RETURN
*			
TEMP	TEMP		
	PAGE		
*			

```

*
* SET UP PARAMETERS FOR THREE NESTED LOOPS
*
*      USES THE COORDINATE SPECIFICATION VALUE (OR RTRANK,
*      IF NONE) TO SET UP THREE NESTED LOOPS: THE OUTER,
*      CORRESPONDING TO COORDS 1,2,...,K-1; THE MIDDLE,
*      FOR COORD K; AND THE INNER FOR COORDS K+1,...,N
*      (WHERE K= COORD SPEC, N = RT RANK). THIS ROUTINE
*      SETS UP RANK/SIZE/TYPE CELLS IDENTICALLY FOR RT ARG
*      AND RESULT, ALLOCATES RESULT DB, COPIES RT DIMENS
*      TO RESULT, AND SETS UP LOOP CONTROL PARAMETERS AS
*      FOLLOWS:
*
*      COORDK   = K
*      OUTRCNT  = D(1)*D(2)*...*D(K-1)
*      OUTRSTEP = D(K)*D(K+1)*...*D(N)
*      MIDLCNT  = D(K)
*      MIDLSTEP = D(K+1)*D(K+2)*...*D(N)
*      INERCNT  = D(K+1)*D(K+2)*...*D(N)
*      INERSTEP = 1
*
*      IF THE RESULT SIZE IS 0 OR 1, THIS ROUTINE EXITS
*      TO 'MNOP' WHICH SETS RESULT = RTARG; OTHERWISE IT
*      RETURNS TO CALLER. LINK IS L2.
*
*      'ST3LUPSK' IS AN ALTERNATE ENTRY WHICH USES THE VALUE
*      IN REG 'K' AS COORDINATE SPEC, INSTEAD OF VALUE IN
*      OPER WORDS. 'ST3LUPSN' IS AN ALTERNATE ENTRY WHICH
*      DOESN'T ALOC RESULT, COPY DIMENS, OR EXIT TO 'MNOP'.
*
ST3LUPSN EQU      %
AI,L2      -1**17          SET FLAG BIT
SET3LUPS EQU      %
LB,T       *RTARG          GET ARG TYPE
STW,T      RTTYPE
LI,X       1
LB,R       *RTARG,X        GET ARG RANK
STW,R      RTRANK
BEZ        10Z0            IF SCALAR, CHECK FOR NO COORD SPEC
LB,K       OPER+1          GET COORDINAT SPECIFICATION VALUE
BEZ        10Z1            IF ZERO, NONE WAS GIVEN
CW,K       RTRANK          IF GIVEN, MAKE SURE COORD<=RANK
BLE        10Z2
B          ERCOORD
10Z0 LB,K       OPER+1          RTARG IS SCALAR: NO COORDINATE
BNEZ       ERCOORD        MAY BE SPECIFIED.
LI,S       1
STW,S      INERSTEP        SET ALL LOOP PARAMS = 1
STW,S      INERCNT
STW,S      MIDLSTEP
STW,S      MIDLCNT
STW,S      OUTRSTEP
B          10Z6
10Z1 LW,K     RTRANK        NO COORD: USE ARG RANK (=LAST COORD)
*
ST3LUPSK EQU      %
10Z2 STW,K     COORDK        SET LOOPS, USING K-REG VAL
MTW,1      RTARG          SET K'TH COORD
LW,X       RTRANK        POINT TO 1ST DIMEN -1
LI,S       1              PREPARE TO MULTIPLY ALL DIMENS
STW,S      INERSTEP        INIT INNER COUNT
10Z3 CW,X     COORDK        INNER STEP = 1
BNE        10Z4            WHEN THE K'TH COORD IS REACHED,
STW,S      INERCNT        SET INNER COUNT AND MIDDLE STEP
STW,S      MIDLSTEP        = D(K+1)*D(K+2)*...*D(N),
MW,S       *RTARG,X        OUTER STEP
STW,S      OUTRSTEP        = D(K)*D(K+1)*...*D(N),
LW,S       *RTARG,X        AND MIDDLE COUNT = D(K);
STW,S      MIDLCNT
LI,S       1              INIT S FOR OUTER COUNT,
B          10Z5            SKIP DIMEN K.
10Z4 MW,S     *RTARG,X        ACCUM SIZE
BOV        ERLNGTH
10Z5 BDR,X    10Z3
MTW,-1     RTARG          DONE: RESTORE ARG PNTR
10Z6 STW,S    OUTRCNT        OUTER COUNT = D(1)*D(2)*...*D(K-1)
MW,S      OUTRSTEP
BOV        ERLNGTH

```

10-00067

10-00069

```

        STW,S      RTSIZE      SIZE = D(1)*D(2)*...*D(N)
        AI,L2      0           IF SIGN BIT SET, ENTRY WAS
        BLZ        *L2        'ST3LUPSN': DONT ALOC RS.
        BDR,S      10Z7       IF RESULT IS 1 ELEMENT,
        B          MNOP        RESULT = RTARG.
10Z7      BAL,L1    RSLIKERT    ALOC RESULT DB LIKE RT ARG
        BAL,LX     MRTDIMS     COPY ARG DIMENS TO RESULT
        B          *L2        RETURN

*
*
INERCNT  TEMP      INNER  LOOP  COUNT
INERSTEP TEMP      INNER  LOOP  STEP
MIDLCNT  TEMP      MIDDLE LOOP COUNT
MIDLSAVE TEMP      MIDDLE LOOP STEP
MIDLSTEP TEMP      MIDDLE LOOP SAVE CELL
OUTRCNT  TEMP      OUTER  LOOP COUNT
OUTRSAVE TEMP      OUTER  LOOP STEP
OUTRSTEP TEMP      OUTER  LOOP SAVE CELL
COORDK   TEMP      K'TH COORDINATE SPEC
*
*
*   EXCHANGE INNER/MIDDLE LOOPS
*
*           EXCHANGES INNER AND MIDDLE LOOP PARAMETRS, AND
*           SETS FVALUE TO ZERO.  LINK IS LX.
*
EXCHLUPS EQU      %           SWAP COUNTS
        LW,R      INERCNT
        XW,R      MIDLCNT
        STW,R     INERCNT
        LW,R      INERSTEP    SWAP STEPS
        XW,R      MIDLSTEP
        STW,R     INERSTEP
        LI,R      0           ZERO FVALUE
        STW,R     FVALUE
        B          0,LX       RETURN
        PAGE

*
*
*   CHECK SUBSCRIPT
*
*           THE SUBSCRIPT VALUE IN 'AI' IS OFFSET TO ORIGIN 0
*           AND RANGE-CHECKED. OFFSET VALUE IS RETURNED IN 'AI'.
*           LINK IS LX.
*
CHEKSUBS SW,AI     ORIGIN      OFFSET TO ORIGIN 0
        BLZ        ERINDEX     MAKE SURE ITS VALUE IS
        CW,AI      *ARGDIMPT,NX >=0 AND
        BL         0,LX        < (CURRENT DIMEN).
        B          ERINDEX
        PAGE

*
*
*   GET INTEGER ARRAY VALUES
*
*           'GARAYVAL' IS USED TO FETCH, ONE BY ONE (FROM LAST
*           TO FIRST), THE ELEMENTS OF AN ARRAY, CONVERTING THEM
*           TO THE INTEGER DOMAIN.  IT IS CALLED WITH THE DATA
*           BLOCK POINTER +RANK+1 IN X, SIZE (ELEMENTS) IN S,
*           AND TYPE IN T; LINK IS LZ.  THE CALLING SEQUENCE
*           IS AS FOLLOWS:
*
*           BAL,LZ    GARAYVAL,T
*           B          ENDLOC
*           (PROCESS ONE VALUE)
*
*           ...
*           B          *L1
*
*           'GARAYVAL' WILL RETURN TO BAL+2 ONCE FOR EACH ARRAY
*           VALUE, WITH ITS INTG VAL IN AI AND LCC SET; VALUES ARE
*           PASSED IN REVERSE RAVEL ORDER.  THE CALLER MUST NOT
*           CLOBBER THE FOLLOWING REGS: S, T, X, LZ.
*           AFTER ALL VALUES HAVE BEEN PASSED, 'GARAYVAL' WILL
*           RETURN TO BAL+1.
*
GARAYVAL EQU      %-LOGL      GET ARRAY VALUE: JUMP TABLE
        B          7Z5        L VALUES: CONVERT TO I
        B          ERDOMAIN    C VALUES: WRONG

```

	B	7Z10	I VALUES: OK
	B	7Z8	F VALUES: CONVERT TO I
	B	7Z1	ISEQ: GEN VALUES
	B	ERDOMAIN	LIST: WRONG
*			
7Z1	LW, AI	0, X	ISEQ: INIT TO LAST VAL
	MW, AI	2, X	
	AW, AI	1, X	= BASE+STEP*SIZE
7Z2	STW, AI	AVALTEMP	SAVE VALUE (USER CLOBBERS)
	BAL, L1	1, LZ	PASS IT TO CALLER
	LW, AI	AVALTEMP	RESTORE VALUE
	SW, AI	2, X	SUBTRACT STEP VAL
	BDR, S	7Z2	DO NEXT
	B	0, LZ	END: RETURN TO ENDLOC
*			
7Z5	LCW, T	S	SPLIT UP ARRAY SIZE INTO
	AI, S	31	WORD AND BIT COUNT.
	SLS, S	-5	S = WORD COUNT
	OR, T	=-32	T = - BIT COUNT
7Z6	LW, AI	*S, X	GET CURRENT WORD
	AND, AI	BITMASK+33, T	EXTRACT CURRENT BIT
	BEZ	7Z7	CONVERT TO INTG FORM:
	LI, AI	1	0 OR 1.
7Z7	BAL, L1	1, LZ	PROCESS ARRAY VALUE
	BIR, T	7Z6	ALL BITS
	LI, T	-32	RESET BIT COUNT
	BDR, S	7Z6	ALL WORDS
	B	0, LZ	FINISH ARRAY PROCESSING
*			
7Z8	AND, X	=-2	FLOT: CONVERT PNTR TO DOUBLEWORD ADR
	XW, S	X	COUNT MUST BE IN INDEX
7Z9	LD, AF	*S, X	GET NEXT VALUE
	BAL, LX	F2I	CONVERT TO INTEGER
	B	ERDOMAIN	(IF POSSIBLE)
	BAL, L1	1, LZ	PROCESS IT
	BDR, X	7Z9	ALL VALUES
	B	0, LZ	FINISH
*			
7Z10	LW, AI	*S, X	INTG: GET NEXT VALUE
	BAL, L1	1, LZ	PROCESS IT
	BDR, S	7Z10	ALL VALUES
	B	0, LZ	FINISH
*			
AVALTEMP	TEMP		ARRAY VALUE TEMP
	PAGE		
*			
*			
*	GET INTEGER SCALAR VALUE		
*			
*	'GSCLRVAL' IS USED TO OBTAIN THE VALUE OF A		
*	SCALAR, CONVERTING IT TO THE INTEGER DOMAIN IF		
*	NECESSARY. IT IS CALLED WITH THE DATA BLOCK		
*	POINTER IN X AND TYPE IN T; LINK IS LZ.		
*	THE CALLING SEQUENCE IS:		
*			
*	BAL, LZ GSCLRVAL, T		
*			
*	THE VALUE IS RETURNED IN 'AI' WITH LCC SET ACCORDING		
*	TO VALUE'S SIGNUM.		
*			
GSCLRVAL	EQU	%-LOGL	GET SCALAR VALUE: JUMP TABLE
	B	8Z3	L VALUE: CONVERT TO I
	B	ERDOMAIN	C VALUE: WRONG
	B	8Z5	I VALUE: OK
	B	8Z4	F VALUE: CONVERT TO I
	B	8Z1	ISEQ: (TO HANDLE 1-ELMT ARRAY)
	B	ERDOMAIN	LIST: WRONG
*			
8Z1	LW, AI	2, X	ISEQ: VALUE = BASE+STEP
	AW, AI	3, X	
	B	0, LZ	RETURN
*			
8Z3	LW, AI	2, X	LOGL SCALAR: GET VALUE
	AND, AI	=X'80000000'	
	BEZ	0, LZ	
	LI, AI	1	CONVERT TO I-FORM
	B	0, LZ	RETURN
*			


```

8Z4      AI,X      1      FLOT SCALAR: GET VALUE
          SLS,X    -1
          LD,AF    2,X
          BAL,LX   F2I      CONVERT TO INTO
          B        ERDOMAIN (IF POSSIBLE)
          B        0,LZ     RETURN

```

```

*
8Z5      LW,AI     2,X      INTG: GET VALUE
          B        0,LZ     RETURN
          PAGE

```

GET SUBSCRIPT VALUES

THESE SUBROUTINES EXECUTE THE LOOPS SPECIFIED BY THE L-BLOCK, DELIVERING ONE ZERO-ORIGIN SUBSCRIPT VALUE PER CALL. '1STSCRPT' INITIALIZES THE LOOPS AND RETURNS THE FIRST SUBSCRIPT IN 'K'. 'NXTSCRPT' PRODUCES ONE ADDITIONAL SUBSCRIPT IN 'K' EACH TIME IT IS CALLED. LINK IS L1. THE FOLLOWING REGISTERS MUST NOT BE Clobbered BY THE CALLER, ONCE THEY HAVE BEEN SET UP BY '1STSCRPT': K, KB, KV, KN, KL.

THE LOOP CONTROL BLOCK CONTAINS THE PARAMETERS NEEDED TO INITIALIZE EACH LOOP, AND SPACE TO HOLD RUNNING LOOP VALUES. FOR A NESTING DEPTH OF N, THE BLOCK CONSISTS OF N 6-WORD 'B-BLOCKS', EACH DESCRIBING ONE LOOP, AND A 'V-BLOCK' LARGE ENUF TO CONTAIN ALL THE SUBSCRIPTS REQUIRED FOR 'ARRAY' LOOPS AS DESCRIBED BELOW. LET YJ(IJ) (FOR IJ=1,2,...,MJ) BE THE VALUES PRODUCED BY THE J'TH LOOP; THEN THE VALUES PRODUCED BY 1STSCRPT/NXTSCRPT ARE =FVALUE+Y1(I1)+Y2(I2)+...+YN(IN), WITH 'IN' RUNNING FASTEST AND 'I1' SLOWEST.

EACH LOOP IS OF ONE OF THE FOLLOWING TYPES:

1. SEQUENCE LOOP: PRODUCES $YJ(IJ) = (IJ-1) * STEPJ$ FOR $IJ = 1, 2, \dots, MJ$. B-BLOCK CONTENTS:

```

WORD 0 = STEPJ
WORD 1 = MJ
WORD 2 = FVALUE+Y1(I1)+...+YJ(IJ)
WORD 3 = MJ-IJ
WORD 4 = (UNUSED)
WORD 5 = 0 (OP CODE FOR 'SEQLOOP')

```

EACH ITERATION REDUCES WORD 3 BY 1, AND ADDS STEPJ TO WORD 2.

2. ARRAY LOOP: PRODUCES $YJ(IJ) = L(VJ+IJ-1)$ FOR $IJ = 1, 2, \dots, MJ$. B-BLOCK CONTENTS:

```

WORD 0 = VJ (OFFSET INTO V-BLOCK)
WORD 1 = MJ
WORD 2 = LOC(L)+VJ+IJ-1
WORD 3 = MJ-IJ
WORD 4 = FVALUE+Y1(I1)+...+YK(IK) (K=J-1)
WORD 5 = -1 (OP CODE FOR 'ARAYLOOP')

```

EACH ITERATION REDUCES WORD 3 BY 1, INCREASES WORD 2 BY 1 AND ACCESSES NEW YJ VALUE FROM V-BLOCK LOCATION GIVEN BY WORD 2.

3. MIXED SEQUENCE LOOP: USED BY 'TAKE' OPERATOR, THIS LOOP PRODUCES PJ 'NULL GROUPS' FOLLOWED BY MJ-PJ VALUES OF THE 'SEQUENCE' LOOP VARIETY - OR, IT FIRST PRODUCES QJ VALUES, THEN MJ-QJ NULL GROUPS.

```

CASE 1: YJ(IJ) = 'NULL GROUP' FOR IJ=1,...,PJ
          = (IJ-PJ-1)*STEPJ FOR IJ=PJ+1,...,MJ.
CASE 2: YJ(IJ) = (IJ-1)*STEPJ FOR IJ=1,...,QJ
          = 'NULL GROUP' FOR IJ=QJ+1,...,MJ.

```

B-BLOCK CONTENTS:

```

WORD 0 = STEPJ
WORD 1 = MJ
WORD 2 = FVALUE+Y1(I1)+...+YJ(IJ)
WORD 3 = MJ-IJ

```

```

*
*          WORD 4 = T+IJ (T= NULL TEST VAL, SEE BELOW)
*          WORD 5 = (X,-2) X (IN THE UPPER 8 BITS)
*
*          IS THE INDEX INTO NULTSBUF FOR 'T'
*          AND INTO NULCTBUF FOR 'G'. THE
*          LOWER 24 BITS CONTAIN -2, THE OP
*          CODE FOR 'MIXLOOP'.
*
*          NULTSBUF(X) CONTAINS 'T', THE NULL TEST VALUE: IN
*          CASE 1, T=-PJ<0; IN CASE 2, T=X'7FFFFFFF'-QJ>0.
*          NULCTBUF(X) CONTAINS 'G', THE NULL-GROUP COUNT:
*          G=MK*...*MN (K=J+1).
*          EACH ITERATION LOWERS WORD 3 BY 1 AND RAISES
*          WORD 4 BY 1 TESTING ITS SIGN: IF T+IJ>0, THE LOOP
*          ACTS LIKE A 'SEQUENCE' LOOP, ADDING STEPJ TO
*          WORD 2; IF T+IJ<=0, A NULL GROUP OF LENGTH G IS
*          PASSED TO THE RESULT, AND THE EXECUTION OF LOOPS
*          J+1,...,N IS SKIPPED. NXTSCRPT PASSES THE NULL
*          GROUP BY PUTTING G IN REG 'N2' AND EXITING ON
*          LINK L2, INSTEAD OF L1.
*
*
1STSCRPT EQU      %          GET 1ST SCRIPT
          BAL,LX      UPDATELP
          LW,KB       OUTRBLOK
          LW,K        FVALUE
          CW,KB       INERBLOK
          BLE         4Z6
          LI,KN       1
          BAL,KL      *L1
*
*
*
*
NXTSCRPT EQU      %          GET NEXT SCRIPT
          BDR,KN      *KL
*
*          BRANCH IF CURRENT LOOP NOT DONE
*          (KL HAS BEEN SET BY ONE OF
*          SEVERAL 'BAL,KL ...' OPS).
4Z1      AI,KB       -BN
          CW,KB       OUTRBLOK
          BL          *RETURN
          LW,KN       BCOUNT,KB
          LW,LX       BLOOPTYP,KB
          BDR,KN      4Z3,LX
          B           4Z1
*          4Z21      LW,KV       BOFFSET,KB
*
*          ARRAY LOOP: RESTORE OFFSET VAL
*          LOC, THEN GO COUNT THIS LOOP.
          AI,KV       1
          LW,K        *KV
          AW,K        BADDEND,KB
          STW,KV      BOFFSET,KB
          B           4Z5
4Z2
*          SAVE THIS LOOP'S OFFSET AND COUNT,
*          INIT INNER LOOPS.
4Z3      TABLE     LOPBIAS
          B           4Z13
          B           4Z21
4Z31     LW,K        BSCRIPT,KB
          AW,K        BSTEPVAL,KB
4Z4      STW,K       BSCRIPT,KB
4Z5      STW,KN      BCOUNT,KB
          AI,KB       BN
4Z6      LW,KN      BINITCNT,KB
          LW,LX       BLOOPTYP,KB
          B           4Z7,LX
4Z61     LW,KV      BINITOFS,KB
          AW,KV      LBLOCK
          STW,K       BADDEND,KB
          AW,K        *KV
          CW,KB       INERBLOK
          BL          4Z2
          BAL,KL      *L1
*
*          AI,KV      1
          LW,K        *KV
          AW,K        BADDEND,KB
          B           *L1
*
*          4Z7      TABLE     LOPBIAS
          B           4Z9
          LOOP INIT ROUTINE TBL
          MIXED SEQUENCE LOOP

```

U10-0122

```

*      B      4Z61      ARRAY LOOP
*      CW,KB    INERBLOK  SEQ LOOP
*      BL      4Z4      IF THIS ISN'T THE INNERMOST LOOP,
*      BAL,KL  *L1      SAVE ITS STUFF & INIT INNER LOOPS.
*      4Z8      AW,K      BSTEPVAL,KB  SET KL TO INNER-SEQUENCE-LOOP LOC,
*      B      *L1      AND RETURN K TO CALLER.
*      4Z9      LB,LX      LX          BUMP INNER SEQUENCE LOOP BY SIMPLY
*      LW,LX      NULTSBUF,LX        ADDING THE STEP VAL; RETURN.
*      STW,LX      BNULTEST,KB      INIT MIXLOOP:
*      4Z10      MTW,1      BNULTEST,KB  MIXLOOP: GET NULTSBUF INDEX
*      BGZ      4Z11      >0 TREAT LIKE SEQLOOP
*      LI,KL      4Z10      <=0 FIRST, SEND A NULL GROUP,
*      B      4Z12      THEN TRY AGAIN.
*      4Z11      CW,KB      INERBLOK    IS THIS THE INNERMOST LOOP ?
*      BL      4Z4      NO, SAVE STUFF & INIT NEXT INNER LUP
*      BAL,KL  *L1      YES, SEND SCRIPT (K) TO CALLER,
*      MTW,1      BNULTEST,KB        AND ADD STEP TO SCRIPT, AS LONG
*      BGZ      4Z8      AS NULL TEST >0.
*      4Z12      LW,LX      BLOOPTYP,KB  GET NULCTBUF INDEX
*      LB,LX      LX
*      LW,N2      NULCTBUF,LX        GET NULL-GROUP COUNT
*      B      *L2      GO TO SEND-NULL ROUTINE
*      4Z13      MTW,1      BNULTEST,KB  BUMP NON-INNER MIXLOOP
*      BGZ      4Z31      IF NULL TEST >0, TREAT LIKE SEQLOOP
*      LI,KL      4Z13      IF <=0, SEND NULL GROUPS UNTIL
*      B      4Z12      IT GOES >0 (OR LOOP COUNTS OUT).
*      PAGE
*
*
*      MOVE BUFFER DIMENSIONS
*
*      COPIES DIMENSION BUFFER CONTENTS TO RESULT DIMENS.
*      LINK IS LX.
*
*      MBUFDIMS EQU      %
*      LCW,X      RSRANK      GET -RANK
*      BEZ      0,LX      IF RANK=0, NO DIMENS TO MOVE
*      LW,A      RESULT      GET PNTR TO 1ST RS DIMEN -2
*      9Z1      LW,R      DBUFEND,X  GET DIMEN FROM BUFFER
*      STW,R      2,A      STORE IT IN RESULT
*      AI,A      1
*      BIR,X      9Z1      DO 'EM ALL
*      B      0,LX      RETURN
*
*
*      MOVE ARG DIMENSIONS
*
*      COPIES RIGHT ARG DIMENSIONS TO RESULT DIMENS.
*      LINK IS LX.
*
*      MRTDIMS EQU      %
*      LW,X      RSRANK      GET RANK
*      BEZ      0,LX      IF RANK=0, NO DIMENS TO MOVE
*      MTW,1      RTARG      POINT TO 1ST DIMEN -1
*      MTW,1      RESULT
*      9Z2      LW,R      *RTARG,X  GET A DIMEN FROM ARG
*      STW,R      *RESULT,X      STORE IT IN RESULT
*      BDR,X      9Z2      DO 'EM ALL
*      MTW,-1      RTARG      RESTORE POINTERS TO NORMALCY
*      MTW,-1      RESULT
*      B      0,LX      RETURN
*      PAGE
*
*
*      SET UP ADDRESS FOR INDEXED LOAD/STORE
*
*      SETS UP LFADR IF A=0; RTADR IF A=1; RSADR IF A=2.
*      FOR CHAR/INTG/FLOT TYPE, SETS UP ADDRESS OF 1ST
*      DATA VALUE, WITH 'K' IN INDEX FIELD; FOR LOGL,
*      SETS UP ADR OF THE INDEXED LOAD-STORE LOGL ROUTINE.
*      LINK IS LX. ADDRESS IS ALSO RETURNED IN 'R'.
*
*      XSETUP EQU      %
*      LW,X      LFARG,A      GET ARG PNTR
*      LI,T      1

```

	LB, R	*X, T	GET LEFT ARG RANK
	AW, R	LFARG, A	
	AI, R	K**17+2	INIT ADR = ARG+RANK+2, K
	LB, T	*X	GET TYPE CODE
	B	%+1-LOGL, T	GO INTO JUMP TABLE - BY TYPE:
	B	6Z1	LOGL
	B	6Z2	CHAR
	B	6Z2	INTG
	B	6Z3	FLOT
	STW, R	XSEQLADR, A	ISEQ: SAVE BASE VALUE ADR
	LW, R	XSQADRTB, A	GET ADR OF ISEQ CALC ROUTINE
	B	6Z2	STORE ADR, EXIT
*			
6Z1	STW, R	XLGLLADR, A	LOGL: SAVE ADR OF LOGL DATA
	LW, R	XLGADRTB, A	ADR = LOC OF LOGL COPY SUBR
6Z2	STW, R	LFADR, A	STORE ADR
	B	0, LX	RETURN
6Z3	AI, R	1	FLOT: ADR =
	AND, R	=-2	ARG+RANK+(2 OR 3), K
	STW, R	LFADR, A	STORE ADR
	B	0, LX	RETURN
*			
XLGADRTB	TABLE	0	LOGL LOAD/STORE SUBR ADR TABLE
	PZE	INDXLDLL	INDEXED LOAD LOGICAL LEFT
	PZE	INDXDLR	INDEXED LOAD LOGICAL RIGHT
	PZE	INDXSTLG	INDEXED STORE LOGICAL RESULT
*			
XSQADRTB	TABLE	0	ISEQ CALC ROUTINE ADR TABLE
	PZE	INDXLSQL	INDEXED LOAD ISEQ LEFT
	PZE	INDXLSQR	INDEXED ISEQ LOAD RIGHT
	PAGE		
*			
*			
*	SET UP ARG ADRS IN SPECIAL WAY		
*			
*		SETS LFADR FOR SEQUENTIAL LOAD USING INDEX N1,	
*		RTADR FOR INDICIAL LOAD USING INDEX K, AND RSADR	
*		FOR SEQUENTIAL STORE USING INDEX N. LINK IS L1.	
*		ALSO SETS 'DELRSIZE' = (OLD RSSIZE)-(NEW RSSIZE).	
*		THIS VALUE IS NONZERO ONLY FOR CERTAIN CHARACTER RESULTS,	
*		IN WHICH CASE IT MAY EQUAL -1, -2, OR -3.	
*			
SETN1KN	EQU	%	
	LI, X	-3	SET UP LEFT ADR CELL FOR
	BAL, LX	SETADR	SEQUENTIAL LOAD.
SETSPEC1	EQU	%	
	LI, A	1	SET UP RTADR FOR
	BAL, LX	XSETUP	INDEXED LOAD: INDEX=K.
	LW, BI	RSSIZE	KEEP OLD RSSIZE
	LI, X	-1	SET UP RSADR FOR SEQUENTIAL
	BAL, LX	SETADR	STORE: INDEX(IF ANY) =N.
	SW, BI	RSSIZE	COMPUTE DELRSIZE = OLD RSSIZE
	STW, BI	DELRSIZE	- NEW RSSIZE.
SETSPEC2	EQU	%	
	LD, IX	N1REGMSK	IF IT HAS AN INDEX (=N, IF ANY),
	CW, IX1	LFADR	CHANGE IT TO N1.
	BAZ	SETLK1	
	STS, IX	LFADR	
	B	*L1	RETURN
SETLK1	LW, IX	LFADR	IF LFARG IS LOGL OR ISEQ,
	CI, IX	INDXLDLL	CHANGE ADR OF INDX-LOAD
	BE	20Z1	SUBROUTINE TO ADR OF ONE
	CI, IX	INDXLSQL	THAT USES K1 AS THE INDEX,
	BNE	*L1	INSTEAD OF K.
	LI, IX	INDSPLL1	
	B	20Z2	
20Z1	LI, IX	INDLDLL1	
20Z2	STW, IX	LFADR	
	B	*L1	
*			
*			
N1REGMSK	BOUND	8	
	PZE	0, N1	N1 INDEX VALUE
	PZE	0, -1	AND MASK.
*			
DELRSIZE	TEMP		CHANGE IN RSSIZE
	PAGE		
*			

```

*
* INDEXED LOAD LOGICAL
*
*          LOADS INTO AI THE K'TH BIT (ORIGIN 0) OF
*          LFARG/RTARG.
*          LINK IS LX. K IS NOT ALTERED.
*
INDXLDLL EQU      %          INDEXED LOAD LOGICAL LEFT
        LW,AF      XLGLLADR   GET LFARG DATA ADR
        B          16Z1
INDXDLR EQU      %          INDEXED LOAD LOGICAL RIGHT
        LW,AF      XLGLRADR   GET RTARG DATA ADR
*          SPLIT SUBSCRIPT INTO
16Z1      LCW,AI      K          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K      -5          AND WORD INDEX IN K.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        AND,AI      *AF,K      FROM APPROPRIATE WORD.
        SCS,K      5          RESTORE K
        BEZ        0,LX        CONVERT TO 0 OR -1 REPRESENTATION
        LI,AI      -1
        B          0,LX
*
*
* INDEXED LOAD LOGICAL, INDEX K1
*
*          LOADS INTO AI THE K1'TH BIT (ORIGIN 0) OF LFARG
*          LINK IS LX. K1 IS NOT ALTERED.
*
INDLDLL1 EQU      %          INDEXED LOAD LOGICAL LEFT (K1)
*          SPLIT SUBSCRIPT INTO
        LCW,AI      K1          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K1      -5          AND WORD INDEX IN K1.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        AND,AI      *XLGLLADR,K1 FROM APPROPRIATE WORD.
        SCS,K1      5          RESTORE K1
        BEZ        0,LX        CONVERT TO 0 OR -1 REPRESENTATION
        LI,AI      -1
        B          0,LX
*
*
* INDEXED STORE LOGICAL
*
*          STORES AI (0 OR -1) INTO K'TH BIT (ORIGIN 0) OF RESULT.
*          LINK IS LX. K IS NOT ALTERED.
*
INDXSTLG EQU      %
        LW,AI-1    AI
*          SPLIT SUBSCRIPT INTO
        LCW,AI      K          -BIT INDEX IN AI
        OR,AI      =-X'20'
        SCS,K      -5          AND WORD INDEX IN K.
        LW,AI      BITMASK+32,AI SELECT APPROPRIATE BIT
        STS,AI-1    *XLGSTADR,K STORE DATA BIT INTO APPROP WORD
        SCS,K      5          RESTORE INDEX VALUE
        LW,AI      AI-1        RESTORE VALUE OF AI
        B          0,LX
*
XLGLLADR TEMP      INDEXED LOGICAL LOAD LEFT ADR
XLGLRADR TEMP      INDEXED LOGICAL LOAD RIGHT ADR
XLGSTADR TEMP      INDEXED LOGICAL STORE ADR
*
* INDEXED LOAD ISEQ
*
*          COMPUTES K'TH ELEMENT OF ISEQ LFARG/RTARG
*          AND PLACES IT IN AI. LINK IS LX. K IS
*          NOT ALTERED.
*
INDXLSQL EQU      %          INDEXED LOAD ISEQ LEFT
        LW,AF      XSEQLADR   GET ISEQ BASE ADR
21Z1      LW,AI      K          GET INDEX
21Z2      ODD,AI
        MW,AI      1,AF        ISEQ ELEMENT = STEP*(K+1)+BASE
        AW,AI      1,AF        = STEP*K+STEP+BASE.
        AW,AI      0,AF
        B          0,LX        RETURN

```

```

*
INDXLSQR EQU      %      INDEXED LOAD ISEQ RIGHT
          LW,AF    XSEQRADR GAE RIGHT ISEQ BASE ADR
          B        21Z1   CONTINUE AS FOR LFARG
*
INDSQLL1 EQU      %      INDEXED LOAD ISEQ LEFT, K1 INDEX
          LW,AF    XSEQLADR GET BASE ADR
          LW,AI    K1      GET INDEX FROM K1, NOT K
          B        21Z2   CONTINUE AS FOR LF LOAD FROM K
*
XSEQLADR TEMP      ISEQ BASE ADR FOR LFARG
XSEQRADR TEMP      ISEQ BASE ADR FOR RTADR
          PAGE
*
*
*  ALLOCATE RESULT LIKE RIGHT ARG
*
*      SETS UP RESULT RANK/SIZE/TYPE CELLS TO MATCH
*      THOSE OF RTARG, AND ALLOCATES THE RESULT DATA
*      BLOCK. LINK IS L1. PNTR RETURNED IN 'A' AND RESULT.
*      'RSLIKRT1' IS AN ALTERNATE ENTRY WHICH DOES
*      THE SAME THING, EXCEPT IT TAKES RSTYPE FROM 'T'.
*      'RSLIKRT2' DOESN'T SET UP RSTYPE AT ALL.
*
RSLIKERT EQU      %
          LW,T      RTTYPE
RSLIKRT1 EQU      %
          ISEQFIX,T USE TYPE INTG INSTEAD OF ISEQ
          STW,T      RSTYPE RESULT TYPE = ARG TYPE
RSLIKRT2 EQU      %
          LW,R      RTRANK
          STW,R      RSRANK RESULT RANK = ARG RANK
          LW,S      RTSIZE
          STW,S      RSSIZE RESULT SIZE = ARG SIZE
          B          ALOCRS ALLOCATE RESULT; RETURN TO CALLER
          PAGE
*
*  UPDATE L-BLOCK POINTERS
*
*      IF THE L-BLOCK MAY HAVE BEEN MOVED (DUE TO AN ALLOCATION
*      OR DE-REF OPERATION), A CALL UPON THIS ROUTINE WILL
*      ASSURE THAT ALL NECESSARY L-BLOCK POINTERS REMAIN VALID.
*      LINK IS LX.
*
UPDATELP EQU      %      UPDATE L-BLOCK POINTERS
          LW,K      LBLOCK NEW L-BLOCK LOC
          SW,K      OLDLBLOK OLD L-BLOCK LOC
          BEZ       0,LX   RETURN, IF THEY ARE EQUAL
          AWM,K      OLDLBLOK NOT EQUAL: ADD DIFFERENCE TO
          AWM,K      INERBLOK ALL POINTERS.
          AWM,K      OUTRBLOK
          B          0,LX   RETURN
*
*  LISTCHK-CHECKS LFARG AND RTARG-ERROR IF TYPE IS LIST OR HIGHER
*
LISTCHK  LI,A      LIST
          CB,A      *LFARG
          BLE       ERDOMAIN
          CB,A      *RTARG
          BG        0,LX
          B          ERDOMAIN
          PAGE
*
*
*  TEMPS FOR INDEX ROUTINES
*
ARGDIMPT TEMP      ARG DIMEN POINTER
BLENGTH TEMP      B BLOCK LENGTH
VLENGTH TEMP      V BLOCK LENGTH
BPNTR EQU          OUTRBLOK B BLOCK POINTER
VPNTR TEMP      V BLOCK POINTER
DIMENPTR TEMP      DIMENSION POINTER
SCRIPT2 TEMP      SUBSCRIPT LIST PNTR +2
ISEQFLAG TEMP      ISEQ FLAG (<0 FOR .TRUE.)
XRANK TEMP      SUBSCRIPT RANK
XSIZE TEMP      SUBSCRIPT SIZE
WEIGHT TEMP      WEIGHT FACTOR
FVALUE TEMP      FIXED PART VALUE

```

OLDLBLOK TEMP
INERBLOK TEMP
OUTRBLOK TEMP
*

OLD L-BLOCK POINTER
INNERMOST B-BLOCK POINTER
OUTERMOST B-BLOCK POINTER

NTEMPS SET
23Z END

ERROR,X'F',TLOC>38 'TOO MANY TEMPS'
TLOC

U10-0124
U10-0125

*
*
* E X T E R N A L C O M M U N I C A T I O N

*

*

*

*

*

✱

*

*

*

*

```

TEMP    CNAME    1
DTEMP   CNAME    2
        PROC
        DO1      NAME=2
TLOC     SET      TLOC+(TLOC&1)          U11-0009
        DISP     TLOC                    U11-0010
LF        EQU     EVALTMP+TLOC           U11-0011
TLOC     SET      TLOC+NAME              U11-0012
        PEND

```

*
*

```

EVEN     CNAME    0
ODD      CNAME    1
        PROC
LF        EQU      %
        ERROR,1,(CF(2)+NAME)&1    'REGISTER HAS WRONG PARITY'
        PEND

```

*
*

```

EQUAL    CNAME
        PROC
LF        EQU      %
        ERROR,1,1-(CF(2)=CF(3))  'REGISTERS MUST BE EQUAL'
        PEND
        PAGE

```

*
*

* FUZZ APPLYING PROCS

*
*

```

FUZZ     OPEN      R
        CNAME
        PROC
        ERROR,1,1-((NUM(AF)>0)&(NUM(CF)=2)) 'BAD AF OR CF'
R        SET      CF(2)          SET REGISTER
LF       CLM,R     FUZZLIMS      IF VALUE WITHIN FUZZ (ABSOLUTE)
        BCR,9     AF             OF ZERO, GO TO GIVEN LOC.
        AD,R      FUZZBIT       PERTURB VALUE SLIGHTLY UPWARDS
        CW,R      =X'00F00000'   TEST SIGN AND FRACTION OVFL
        BCS,5     %+2           IF POSITIVE, AND FRACTION OVFL,
                                LEAVE EXPON INCREASED BY 1,
                                AND SET FRACTION = .1.
        AW,R      =X'00100000'   DISCARD A FEW FRACTION BITS
        AND,R+1   FUZZMASK      GUARANTEE VALUE PROPERLY NORMALIZED
        FAL,R     FLOT0
        PEND
        CLOSE     R
        PAGE

```

*
*

* FUNCTION EVALUATORS

*
*

* THE FUNCTION EVALUATION SUBROUTINES MUST NOT CLOBBER
* REGISTERS 1, 2, 4, 10, 11 (KNOWN BY XSEG-GENERATING
* MODULES AS N, K, N1, N2, AND N3).

*
*

USECT PROGSECT

*
*

* FLOATING TO INTEGER CONVERSION

*
*

* CONVERTS FLOATING VALUE IN AF/AF1 TO INTEGER VALUE
* IN AI, IF POSSIBLE. LINK IS LX AND THERE ARE TWO
* RETURNS:
* BAL+1: THE VALUE IS WAY TOO BIG IN MAGNITUDE
* (A NEGATIVE AI IS RETURNED), OR IT WAS
* IN RANGE BUT NOT WITHIN 'FUZZ' OF AN
* INTEGER (ITS 'FLOOR' IS RETURNED IN AI).
* BAL+2: THE VALUE IS WITHIN 'FUZZ' OF AN INTEGER,
* WHICH IS RETURNED IN AI, WITH LCC SET.

*
*

```

F2I      EQU      %
        CLM,AF    INTGLIMS      IS VALUE IN RANGE ?
        BCR,9     7Z2
        LI,AI     -1            NO, SET AI NEGATIVE
        B         0,LX          AND RETURN TO BAL+1.
7Z2      FUZZ,AF   7Z4          APPLY FUZZ TO VALUE

```

```

        STD,AF      X
        FAL,AF      BIG
        STW,AF1     NUMER
        FSL,AF      BIG
        CD,AF       X
        BE          7Z3
        LW,AI       NUMER
        B           0,LX
7Z3      LW,AI       NUMER
        B           1,LX
7Z4      LI,AI      0
        STD,AI      X
        B           1,LX
*
*
        BOUND      8
INTGLIMS DATA      X'B7800000',X'487FFFFF'
BIG        DATA      X'4E200000',0
EXPON4E    DATA      X'4E000000',0
FLOT1X     DATA      X'40FFFFFF',X'FFFFFFFF'
        PAGE
*
*
*   INTEGER FLOOR
*
*       COMPUTES THE INTEGER FLOOR OF A FLOATING ARG, IF
*       WITHIN RANGE.  LINK IS LX.
*
IFLOOR     EQU      %
        FSL,AF      FUZZNEG
        BGEZ        %+2
        FSL,AF      FLOT1X
        CLM,AF      INTGLIMS
        BCS,9       INTGOVFL
        FAL,AF      BIG
        B           0,LX
*
*
*   INTEGER CEILING
*
*       COMPUTES THE INTEGER CEILING OF A FLOATING ARG, IF
*       WITHIN RANGE.  LINK IS L1.
*
ICEILING   EQU      %
        LCD,AF      AF
        BAL,LX      IFLOOR
        LCW,AI      AI
        BNOV        *L1
        B           INTGOVFL
        PAGE
*
*
*   FLOATING FLOOR
*
*       COMPUTES THE FLOATING FLOOR OF A FLOATING ARG.
*       LINK IS LX.
*
FFLOOR     EQU      %
        FSL,AF      FUZZNEG
        BGEZ        %+2
        FSL,AF      FLOT1X
        FAL,AF      EXPON4E
        B           0,LX
*
*
*   FLOATING CEILING
*
*       COMPUTES THE FLOATING CEILING OF A FLOATING ARG.
*       LINK IS L1.
*
FCEILING   EQU      %
        LCD,AF      AF
        BAL,LX      FFLOOR
        LCD,AF      AF
        B           *L1
        PAGE

```

YES, PUT INTG PART IN AF1 (=AI).
 SAVE INTG PART
 SEE IF FRACTION WAS ZERO

NO, GET FLOOR
 AND TAKE BAL+1 RETURN.
 YES, GET VALUE
 AND TAKE BAL+2 RETURN.
 VALUE NEAR ZERO: SET
 SAVE DW ZERO FOR FL. PT. USE
 RESULT =0, TAKE OK EXIT.

APPLY FUZZ (ABSOLUTE)
 SUBTRACT 1 FROM NEGATIVE VALUE
 SINCE FAL TRUNCATES TOWARDS ZERO.
 IF PERTURBED VALUE IS IN INTG RANGE,
 PUT INTG PART IN AF1 (=AI).
 RETURN

CEILING(X)
 = -FLOOR(-X)

APPLY FUZZ (ABSOLUTE)
 SUBTRACT 1 FROM NEGATIVE VALUE
 SINCE FAL TRUNCATES TOWARDS ZERO.
 TRUNCATE FRACTION
 RETURN

CEILING(X)
 = -FLOOR(-X)

* FLOATING COMPARISON SETUP

*
* APPLIES FUZZ TO THE TWO FLOATING POINT
* VALUES IN 'AF' AND 'BF'. LINK IS LX.
*

```
FFCOMPAR EQU %
FUZZ,BF 8Z2 APPLY FUZZ TO BF
8Z1 FUZZ,AF FZERORTN APPLY FUZZ TO AF
B 0,LX RETURN
8Z2 LD,BF FLOT0 BF NEAR ZERO: SET IT TO 0.0
B 8Z1 DO AF
*
FZERORTN LI,AF 0 FLOT ZERO: BOTH AF WORDS =0
IZERORTN LI,AI 0 INTG ZERO: AI=0
B 0,LX RETURN
PAGE
```

*
*
* INTEGER ROLL
*

* COMPUTES A RANDOM INTEGER N IN THE RANGE
* ORIGIN <= N <= (AI)+ORIGIN-1. IT IS RETURNED
* IN AI. LINK IS LX. RANDOM SEED IS UPDATED.
*

```
IROLL EQU %
AI,AI 0 MAKE SURE IT'S POSSIBLE TO PICK N;
BLEZ ERDOMAIN I.E., WE NEED AI>0.
XW,AI RANDOM SAVE ARG, GET RANDOM SEED VALUE
ODD,AI
MI,AI 65539 UPDATED SEED =
AND,AI =X'7FFFFFFF' SEED*65539 (MOD 2**31).
XW,AI RANDOM STORE UPDATED SEED, GET ARG
MW,AI-1 RANDOM N = FLOOR(ARG*SEED/(2**31))+ORIGIN
SLD,AI-1 -31
AW,AI ORIGIN <= ARG, SO NO OVFL MAY OCCUR.
B 0,LX
PAGE
```

*
*
* SET FUZZ VALUE
*

* GIVEN 'K' (THE NUMBER OF BITS TO IGNORE) IN AI, WITH
* $0 \leq K \leq 31$, THIS ROUTINE SETS UP ALL FUZZ-DEPENDENT
* PARAMETERS AND RETURNS TO BAL+2 WITH THE OLD K VALUE
* IN AI. IF K IS NOT IN RANGE, IT RETURNS TO BAL+1
* WITHOUT CHANGING FUZZ. LINK IS LX.
*

```
SETFUZZ EQU %
CI,AI -32 IF K<0 OR K>31,
BANZ 0,LX RETURN TO BAL+1.
LI,BF 0 SET FUZZBIT = 2**(K-1), A DOUBLEWORD
LI,BF1 1 QUANTITY CONTAINING EXACTLY ONE
SLD,BF -1,AI 1-BIT, FOLLOWED BY K-1 ZEROS.
STD,BF FUZZBIT
LW,BF =X'40000000' INSTALL EXPONENT TO FLOAT
SFL,BF 14 FUZZBIT VALUE.
STW,BF FUZZLIMS+1 +FUZZBIT(FLOATED) IS UPPER LIM
LCW,BF BF -FUZZBIT(FLOATED) IS LOWER LIM
STW,BF FUZZLIMS
LI,BF1 1
SLS,BF1 0,AI
LCW,BF1 BF1 SET FUZZMASK = -2**K, A SINGLE
STW,BF1 FUZZMASK WORD MASK OF 32-K ONES, K ZEROS.
XW,AI FUZZCNT STORE NEW K, GET OLD K
B 1,LX RETURN TO BAL+2
```

*
*
* SET ORIGIN VALUE
*

* GIVEN 'G' (THE INDEX ORIGIN) IN AI, WITH $0 \leq G \leq 1$,
* THIS ROUTINE SETS UP ALL ORIGIN-DEPENDENT PARAMETERS
* AND RETURNS TO BAL+2 WITH THE OLD G VALUE IN AI.
* IF G IS NOT IN RANGE, IT RETURNS TO BAL+1 WITHOUT
* CHANGING ORIGIN. LINK IS LX.
*

```
SETORG EQU %
CI,AI -2 IF G<0 OR G>1,
BANZ 0,LX RETURN TO BAL+1.
```

EOR, AI	=1	1-G
STW, AI	ORGADJ	SET NEW 'ADJUSTED ORIGIN'
EOR, AI	=1	RESTORE G
XW, AI	ORIGIN	SAVE NEW G, GET OLD G
B	1, LX	RETURN TO BAL+2
PAGE		

*
*
* INTEGER RESIDUE

*
* COMPUTES R = RESIDUE(A,B) = B-A*K >= 0 FOR INTEGER
* A AND B. LINK IS LX. NO OVERFLOW IS POSSIBLE SINCE
* EITHER 0<=R<ABS(A) OR R=B.
*
IIRESIDU EQU %
* LAW, CI AI COPY ABS(A) TO CI (OVFL MAY BE
* BNEZ 5Z1 IGNORED; CODE WORKS IN ALL CASES).
* LW, RI BI IS A ZERO ?
* BGEZ 0, LX A=0. IF B>=0, R=B.
* B ERDOMAIN ELSE, R UNDEFINED.
5Z1 LW, RI BI A NONZERO, EXTEND B'S SIGN
* BGEZ 5Z2 SO WE CAN DIVIDE B BY ABS(A)
* LI, RI-1 -1 AND GET THE REMAINDER.
* B 5Z3
5Z2 LI, RI-1 0
5Z3 DW, RI-1 CI DIVIDE: REM=B-A*J IS IN RI-1
* BNOV 5Z4 THE ONLY WAY IT CAN OVFL IS FOR
* LI, RI 0 A=+-1, B=-2**31; WHENCE R=0.
* B 0, LX
5Z4 LW, RI RI-1 R=B-A*J; ABS(R)<ABS(A)
* BGEZ 0, LX IF R>=0, WE'RE DONE
* AW, RI CI ELSE R:=R+ABS(A) >0
* B 0, LX
* PAGE

*
*
* FLOATING RESIDUE

*
* COMPUTES R = RESIDUE(A,B) = B-A*K >= 0 FOR FLOATING
* A AND B. LINK IS LX.
*
FFRESIDU EQU %
* LAD, CF AF C:=ABS(A)
* BNEZ 6Z1
* LD, RF BF A=0; IF B>=0, R=B.
* BGEZ 0, LX
* B ERDOMAIN ELSE, R UNDEFINED.
6Z1 LD, RF BF A NON ZERO, SAVE B
* FDL, BF CF B:= B/ABS(A)
* FAL, BF EXPON4E B:= FLOOR(B/ABS(A)) = J
* FML, BF CF
* FSL, RF BF R:= B-J*ABS(A); ABS(R)<ABS(A)
6Z2 CLM, RF FUZZLIMS IF R IS WITHIN FUZZ OF ZERO,
* BCR, 9 FZERORTN SET R=0 AND EXIT.
* BGZ 6Z3
* FAL, RF CF R<0: SET R:=R+ABS(A)
* B 6Z2 AND TEST AGAIN.
6Z3 CD, RF CF R>0: SEE IF R<ABS(A)
* BL 6Z4 R<ABS(A), SEE IF VERY NEAR ABS(A)
* FSL, RF CF R>=ABS(A): SET R:=R-ABS(A)
* B 6Z2 AND TEST AGAIN.
6Z4 FSL, CF RF R<ABS(A): SEE IF R IS WITHIN FUZZ
* CLM, CF FUZZLIMS OF ABS(A).
* BCS, 9 0, LX NO: EXIT.
* B FZERORTN YES: SET R:=0 AND EXIT.
* PAGE

*
*
* INTEGER FACTORIAL

*
* 'IFACT' COMPUTES R=FACTORIAL(A) FOR AN INTEGER ARG.
* LINK IS LX.
*
IFACT LW, N AI COPY ARG (AI=RI)
* BGZ 3Z2 IF A>0, GO INTO MULTIPLY LOOP
* BLZ ERDOMAIN IF A<0, DOMAIN ERROR

```

        LI,RI      1          A=0; FACT(0)=1
        B          0,LX      RETURN
        EQUAL,AI,RI  INITIALLY, R=A
3Z1     ODD,RI
        MW,RI      N          R:=R*N, N>=1
        BNOV      3Z2        CONTINUE IF PRODUCT OK
        B          INTGOVFL   ELSE, INTEGER OVERFLOW
3Z2     BDR,N      3Z1        N:=N-1
        B          0,LX      N=0, R=FACT(A); RETURN
        PAGE

```

```

*
*
*
*
*
*
*
*
*
*
*
*

```

FLOATING FACTORIAL

'FFACT' COMPUTES $R=FACT(A)$ FOR A FLOATING ARG.
LINK IS LX. DOMAIN ERRORS RESULT IN DIVISION BY ZERO.

NOTE: (1) 'CW' COMMANDS ARE USED INSTEAD OF 'CD'; THEY
HAVE THE SAME EFFECT FOR THE GIVEN DATA
AND ARE FASTER.
(2) INTERMEDIATE RESULTS ARE KEPT IN REGS INSTEAD
OF CORE; THE EXTRA TIME REQ'D BY FML/FDL
IS LESS THAN THAT REQ'D BY EXTRA LD AND STD.

```

FFACT   EQU        %
        STW,LX     LINKTEMP   SAVE LINK
        BAL,L2     FIXIT      IF ARG NEAR INTG, MAKE IT EXACT
        BLZ        ERDOMAIN   IF ARG = NEG INTG, ERROR
        LD,PF      FLOT1      P:=1
        AI,AF      0          TEST ARGUMENT
        BGEZ       1Z3
1Z1     FAL,AF      FLOT1      A<0,  A:=A+1
        FDL,PF      AF         P:=P/A
        AI,AF      0          CONTINUE UNTIL A>=0
        BLZ        1Z1
        B          1Z4
1Z2     FML,PF      AF         A>=1, P:=P*A
        FSL,AF      FLOT1      A:=A-1
1Z3     CW,AF       FLOT1      CONTINUE UNTIL A<1
        BGE        1Z2
1Z4     LW,LX      LINKTEMP   RESTORE LINK

```

```

*
*
*
*
*
*
*
*

```

WE NOW HAVE $0 \leq A < 1$, AND MAY EVALUATE $FACT(A)$
 $= FACT(A+1)/(A+1)$, WHERE $FACT(A+1)$ IS EVALUATED
AS A DEGREE 15 POLYNOMIAL IN 'A'.

'FACTPOLY' IS ALSO CALLED BY THE COMBINATORIAL ROUTINE
TO PERFORM $R:=P*FACT(A)$ WHERE $0 \leq A < 1$.

```

FACTPOLY EQU        %
        STD,AF      X          X:=A
        FAL,AF      FLOT1
        STD,AF      Y          Y:=A+1
        LD,RF       COEFF15
        LI,N        15
1Z5     FML,RF       X          R:=SUM(I=0,15) OF C(I)*X**I
        FAL,RF      COEFF0-2,N  =FACT(A+1)
        BDR,N       1Z5
        FDL,AF      Y          R:= FACT(A+1)/(A+1) = FACT(A)
        FML,RF      PF         R:=P*FACT(A)
        B          0,LX      RETURN
        PAGE

```

```

*
*
*
*
*
*
*

```

INTEGER COMBINATORIAL

'IICOMB' COMPUTES $R=COMBINATORIAL(A,B)$
 $=FACT(B)/(FACT(A)*FACT(B-A))$
FOR INTEGER ARGS. LINK IS LX.

```

IICOMB  EQU        %
        LW,CI       BI          C=B-A
        SW,CI       AI
        BGZ         4Z9
        BLZ         4Z5
4Z4     LI,RI       1          A=0 OR C=0:      R= 1
        B          0,LX
4Z5     AI,AI       0          B<A

```

```

      BGZ      4Z8
      BEZ      4Z4      IF A=0, R=1
4Z7      LI,RI  0        B<A<0 OR ...:      R= 0
      B        0,LX
4Z8      XW,AI  CI
*
4Z9      AI,BI  0
      BLZ      4Z15
      AI,AI    0
      BGZ      4Z10
      BLZ      4Z7      A<0<=B:          R= 0
      B        4Z4      A=0:          R= 1
4Z10     CW,CI  AI      0<A<B: SET C:=MIN(A,C), R= B- / C-
      BLE      4Z11
      LW,CI    AI
4Z11     LW,AI  BI      INIT NUMERATOR TO 1ST FACTOR (=B)
      LW,N     CI      INIT DENOM AND COUNT = C (>0)
4Z12     BDR,N  4Z13    COUNT LOOP, DECR DENOM FACTOR
      DW,AI    CI      DONE: R = NUMER/DENOM
      B        0,LX    RETURN
4Z13     AI,BI  -1      DECR NUMER FACTOR
      MW,AI    BI      INCLUDE NEW FACTOR IN NUMER
      BNOV     4Z14
      B        INTGOVFL
4Z14     MW,CI  N        INCLUDE NEW FACTOR IN NUMER
      BNOV     4Z12    CONTINUE
      B        INTGOVFL
4Z15     AI,AI  1        R = A+ / C-
      LW,BI    AI      INIT NUMER TO 1ST FACTOR (=A+1)
      LW,N     CI      INIT DENOM AND COUNT = C (>0)
4Z16     BDR,N  4Z17    COUNT LOOP, DECR DENOM FACTOR
      DW,AI    CI      DONE: R = NUMER/DENOM
      B        0,LX    RETURN
4Z17     AI,BI  1        INCR NUMER FACTOR
      MW,AI    BI      INCLUDE NEW FACTOR IN NUMER
      BNOV     4Z18
      B        INTGOVFL
4Z18     MW,CI  N        INCLUDE NEW FACTOR IN DENOM
      BNOV     4Z16
      B        INTGOVFL
      PAGE
*
*
*   FLOATING COMBINATORIAL
*
*       'FFCOMB' COMPUTES R=COMBINATORIAL(A,B)
*       =FACT(B)/(FACT(A)*FACT(B-A))
*       FOR FLOATING ARGS. LINK IS L1.
*
FFCOMB   EQU      %
          STW,L1   LINKTEMP      SAVE LINK (=CF)
          LD,CF    BF
          FSL,CF   AF            C:=B-A
*
*   PART 1 - DETERMINE WHICH ONES AMONG A,B,C ARE WITHIN
*   FUZZ OF INTEGERS.
*
      LI,N      0      INIT CODE: NEITHER A/C INTEGER
      BAL,L2    FIXIT  IF 'A' NEAR INTEGER, MAKE IT EXACT,
      AI,N      1      AND SET 'A INTG' BIT.
      XW,AF     CF      SWAP A/C (BIT 31 OF N IS REALLY
      XW,AF1    CF1     'C INTG' BIT).
      BAL,L2    FIXIT  IF 'A' NEAR INTEGER, MAKE IT EXACT,
      AI,N      2      AND SET 'A INTG' BIT.
      B        2Z1,N
2Z1      B      2Z2
      B        2Z3
      B        2Z3
      LD,BF     AF      NEITHER A/C INTG: TRY 'B'
      FAL,BF    CF      ONLY 'C' INTG: GO TEST SIGNS
      B        2Z3     ONLY 'A' INTG: GO TEST SIGNS
      XW,AF     BF      BOTH A/C INTG: MAKE 'B' EXACTLY
      XW,AF1    BF1     = A+C.
      BAL,L2    FIXIT  GO TEST SIGNS
      BLZ      ERDOMAIN NEITHER A/C INTG: TEMPORARILY
      XW,AF     BF      SWAP A/B TO TEST 'B'.
      XW,AF1    BF1     IF 'B' NEAR INTG, MAKE IT EXACT,
                        AND REQUIRE B>=0.
                        UPDATE 'B' AND RESTORE 'A'
*

```

```

*      PART 2 - CASE TESTING, ACCORDING TO SIGNS OF A,B,C
*
2Z3      LD,PF      FLOT1      P:=1
          AI,CF      0          TEST C (=B-A)
          BGZ      2Z9
          BLZ      2Z5
2Z4      LD,RF      FLOT1      A=0 OR C=0:      R= 1
          B          *LINKTEMP
2Z5      AI,AF      0          B<A
          BGZ      2Z8
          BEZ      2Z4      IF A=0, R=1
          B          2Z6,N
2Z6      B          2Z31      B<A<0, A&C FRAC:      R= A+ C+ / B+
          B          2Z7      B<A<0, C INTG:      R= 0
          B          2Z7      B<A<0, A INTG:      R= 0
2Z7      LD,RF      FLOT0      B<A<0, A&C INTG:      R= 0
          B          *LINKTEMP
2Z8      XW,AF      CF          B<A, 0<A: SWAP A/C (C<0<A, SO THAT
          XW,AF1     CF1          HEREAFTER, A<0<C, B>A).
          LB,N      =X'00020103',N SWAP      A/C INTG BITS
2Z9      AI,BF      0          A<B
          BLZ      2Z12,N
          AI,AF      0          A<B, 0<=B
          BGZ      2Z11,N
          BLZ      2Z10,N
          B          2Z4
2Z10     B          2Z39      A=0:      R= 1
          B          2Z22      A<0<=B, A&C FRAC:      R= A+ B- / C-
          B          2Z7      A<0<=B, C INTG:      R= B- / C-
          B          2Z7      A<0<=B, A INTG:      R= 0
          B          2Z7      A<0<=B, A&C INTG:      R= 0
2Z11     B          2Z42      0<A<B, A&C FRAC:      R= B- / A- C-
          B          2Z22      0<A<B, C INTG:      R= B- / C-
          B          2Z21      0<A<B, A INTG:      R= B- / A-
          CD,CF      AF          0<A<B, A&C INTG:      SET C=MIN(A,C),
          BLE      2Z22          THEN R= B- / C-
          B          2Z21
2Z12     B          2Z35      A<B<0, A&C FRAC:      R= A+ / B+ C-
          B          2Z23      A<B<0, C INTG:      R= A+ / C-
          B          2Z7      A<B<0, A INTG:      R= 0
          B          2Z23      A<B<0, A&C INTG:      R= A+ / C-
*
*      PART 3 - RUN ONE OR TWO LOOPS TO GET A,B,C DOWN TO VALUES
*      BETWEEN 0 AND 1, SO THAT THEIR FACTORIALS ARE COMPUTABLE.
*
*      INTEGER CASES (C IS A POSITIVE INTEGER, SO R MAY BE
*      COMPUTED DIRECTLY BY A MULTIPLY-DIVIDE LOOP; FRACTIONAL
*      FACTORIALS CANCEL):
*
2Z21     LD,CF      AF          A>0, A INTG
2Z22     FML,PF      BF          C>0, C INTG:      P:=P*B/C
          FDL,PF      CF
          FSL,BF      FLOT1      B:=B-1
          FSL,CF      FLOT1      C:=C-1
          BGZ      2Z22      CONTINUE UNTIL C=0
          LD,RF      PF          R = P
          B          *LINKTEMP      RETURN
2Z23     FAL,AF      FLOT1      C>0, C INTG:      A:=A+1
          FML,PF      AF          P:=P*A/C
          FDL,PF      CF
          FSL,CF      FLOT1      C:=C-1
          BGZ      2Z23      CONTINUE UNTIL C=0
          LD,RF      PF          R = P
          B          *LINKTEMP      RETURN
*
*      FRACTIONAL CASES:
*
2Z31     FAL,AF      FLOT1      B<A<0,      A:=A+1
          FAL,BF      FLOT1      B:=B+1
          FML,PF      AF          P:=P*A/B
          FDL,PF      BF
          AI,AF      0          B<A<1
          BLZ      2Z31
          B          2Z33
2Z32     FAL,BF      FLOT1      A OK, B<A
          FAL,CF      FLOT1      A OK, B<0,      B:=B+1
          FML,PF      CF          C:=C+1
          FDL,PF      BF          P:=P*C/B
          AI,BF      0
2Z33

```



```

2Z35    BLZ      2Z32
        B        2Z45      A, B OK
        FAL, AF  FLOT1    A<=B<0,      A:=A+1
        FAL, BF  FLOT1    B:=B+1
        FML, PF  AF       P:=P*A/B
        FDL, PF  BF
        AI, BF   0        A<=B<1
        BLZ      2Z35
2Z36    B        2Z37      B OK, A<=B
        FAL, AF  FLOT1    B OK, A<0,    A:=A+1
        FML, PF  AF       P:=P*A/C
        FDL, PF  CF
        FSL, CF  FLOT1    C:=C-1
2Z37    AI, AF   0        B OK, A<1
        BLZ      2Z36
2Z39    B        2Z45      A, B OK
        FAL, AF  FLOT1    A<0<=B,    A:=A+1
        FML, PF  AF       P:=P*A/C
        FDL, PF  CF
        FSL, CF  FLOT1    C:=C-1
        AI, AF   0        A<1, B>=0
        BLZ      2Z39
2Z41    B        2Z44      A OK, B>=0
        FML, PF  BF       1<=A<=B,    P:=P*B/A
        FDL, PF  AF
        FSL, AF  FLOT1    A:=A-1
        FSL, BF  FLOT1    B:=B-1
2Z42    CW, AF   FLOT1    0<=A<=B
        BGE      2Z41
2Z43    B        2Z44      A OK, B>=A
        FML, PF  BF       A OK, B>=1,  P:=P*B/C
        FDL, PF  CF
        FSL, BF  FLOT1    B:=B-1
        FSL, CF  FLOT1    C:=C-1
2Z44    CW, BF   FLOT1    A OK, B>=0
        BGE      2Z43
2Z45    EQU      %        A, B OK
*
*          WE NOW HAVE 0<=A<1, 0<=B<1, AND C=B-A;  THUS, -1<C<1.
*          R = P*FACT(B)/(FACT(A)*FACT(C)) MAY BE EVALUATED
*          BY USING THE FACTORIAL POLYNOMIAL FOR A AND B;
*          C MAY REQUIRE AN UPWARD SHIFT OF 1.
*
2Z46    AI, CF   0        A, B OK, -1<C<1
        BGEZ     2Z46
        FAL, CF  FLOT1    A, B OK, -1<C<0,  C:=C+1
        FML, PF  CF       P:=P*C
        STD, PF  NUMER    A, B, C OK,      SAVE P
        LD, PF   FLOT1    P:=1
        BAL, LX  FACTPOLY R:=FACT(A)
        LD, PF   RF
        LD, AF   CF
        BAL, LX  FACTPOLY R:=FACT(A)*FACT(C)
        STD, RF  DENOM    SAVE IT
        LD, AF   BF
        LD, PF   NUMER
        BAL, LX  FACTPOLY R:=P*FACT(B)
        FDL, RF  DENOM    R:=P*FACT(B)/(FACT(A)*FACT(C))
        B        *LINKTEMP =COMB(A,B);  RETURN
        PAGE
*
*
*    CHECK FOR NEARNESS OF F.P. VALUE TO INTEGER
*
*          IF 'AF' IS WITHIN FUZZ OF AN INTEGER, 'FIXIT' RETURNS
*          TO BAL+1 WITH THE EXACT F.P. REPRESENTATION OF THAT
*          INTEGER IN 'AF'. OTHERWISE, IT RETURNS TO BAL+2 WITH
*          'AF' UNCHANGED. IN EITHER CASE, SIGN INFO IS RETURNED
*          IN LCC. LINK IS L2. BF/CF ARE NOT CLOBBED.
*
FIXIT   EQU      %
        STD, AF  Y        SAVE THE GIVEN VALUE
        BAL, LX  F2I      ATTEMPT TO CONVERT TO INTEGER
        B        9Z1      NO WAY: BRANCH
        LD, AF   X        YUP: GET F.P. VERSION CREATED
        B        *L2      BY 'F2I' AND RETURN TO BAL+1.
9Z1     AI, L2   1        FRACTIONAL: BUMP RETURN ADR
        LD, AF   Y        RESTORE ORIGINAL VALUE, SET LCC

```

```

      B      *L2      RETURN TO BAL+2
      PAGE

*
*
*  LOCAL DATA/TEMPS
*
*
X      DTEMP      FACT: ARG FOR POLYNOMIAL EVAL
Y      DTEMP      FACT: DIVISOR FOR POLYNOMIAL EVAL
NUMER  DTEMP      COMB: NUMERATOR
DENOM  DTEMP      COMB: DENOMINATOR
LINKTEMP TEMP      LINK REG SAVE TEMP
*
*  FACTORIAL POLYNOMIAL COEFFICIENTS
*
      BOUND      8
COEFF0  DATA,8  FL'+0.9999999999999999032'
      DATA,8  FL'+0.4227843350985151178'
      DATA,8  FL'+0.4118403304219814831'
      DATA,8  FL'+0.0815769194013886786'
      DATA,8  FL'+0.0742490079434012692'
      DATA,8  FL'-0.0002669510287555266'
      DATA,8  FL'+0.0111538196719066992'
      DATA,8  FL'-0.0028515012430346494'
      DATA,8  FL'+0.0020997590350770629'
      DATA,8  FL'-0.0009083465574200521'
      DATA,8  FL'+0.0004677678114964956'
      DATA,8  FL'-0.0002064476319159326'
      DATA,8  FL'+0.0000815530498066373'
      DATA,8  FL'-0.0000248410053848712'
      DATA,8  FL'+0.0000051063592072582'
COEFF15 DATA,8  FL'-0.0000005113262726698'
      PAGE
*
*
*  GLOBAL DATA
*
*
*  THE FOLLOWING IS INCLUDED TO INDICATE THE NAMES,
*  STRUCTURE, AND DEFAULT VALUES OF THE VARIOUS GLOBAL
*  PARAMETERS PERTINENT TO FUNCTION EVALUATION.
*
      DO      0
*
      CSECT    0
      BOUND    8
FUZZBIT DATA 0,X'200'      1-BIT IN LEAST SIGNIFICANT POSITION
FUZZLIMS DATA -X'35200000',; LIMITS FOR TESTING ABSOLUTE
      DATA  +X'35200000'      DISTANCE TO ZERO.
FUZZMASK DATA X'FFFFFFC0'    RIGHT-HALF SIGNIFICANCE MASK
FUZZCNT DATA 10              COUNT OF BITS TO BE IGNORED (0...31)
ORGADJ DATA 0                1-ORIGIN
ORIGIN DATA 1                INDEX ORIGIN (0 OR 1)
RANDOM DATA 123456789         RANDOM SEED (MUST BE ODD)
*
      FIN
FUZZNEG EQU FUZZLIMS          =-FUZZ VAL (ALMOST) IN F.L. FORM
*
      ERROR,X'F',TLOC>10      'TOO MANY TEMPS'
9Z      END

```

U11-0014

TITLE	'APLINP-B00,08/20/73,DWG702985'		
SYSTEM	SIG7F		
SYSTEM	BPM		

*
* REF'S:

REF	BCONTOFF	DOES CONTINUE EXIT	
REF	B4CLOSE	DOES FORCED CLOSE	
REF	BLANKS	X'40404040'	
REF	BREAKFLG	BREAK FLAG	
REF	BSPFLAG	BACKSPACE VALIDITY FLAG	
REF	CALWROUT	APL WRITE OUTPUT RECORD	
REF	CS1	CODESTRING GENERATOR	
REF	CX	CODESTRING EXECUTION	
REF	DUMPLINP	DUMP IMAGE-PROBABLY AS PROMPT	
REF	ENDIMAGE	END OF IMAGE IN USE	
REF	ERRORCHR	ERROR-EVAL INP PROMPT-CHARACTERS	
REF	ERXEQ	ERROR RETURN FOR 'EXECUTE'	12-00010
REF	FIHANDLR	FUNCTION INPUT HANDLER	
REF	HICOL		
REF	IMAGE	APL 'IMAGE' I/O BUFFER	
REF	INBUF	APL INPUT-OUTPUT BUFFER	
REF	INTRANS		
REF	MAXCOL	MAX COL POS FOR INPUT	
REF	MERGECOL	MERGE COLUMN	
REF	MODE	MODE FLAG	
REF	OBSFLAG	OBSERVE FLAG.	
REF	ON%OFF	ON-OFF LINE FLAG	
REF	OUTSIZ	SIZE OF OUTPUT RECORD(5)	
REF	RDAPL	READ APL INPUT RECORD	
REF	RELEASER		
REF	RESQQUAD	RESUME QUOTE-QUAD PROCESSING.	
REF	TABPNTR	TAB INDICATOR	
REF	TABVALS	TAB VALUES	
REF	TERMTYPE	TERMINAL TYPE	
REF	WROUT	APL WRITE OUTPUT RECORD	
REF	WROUTWB	WRITE-WORD BOUND	
REF	RESXTEND,CMDERRH,EDUMPLIN		
REF	IDBADCH,IDOPENQT,IDTRUNC		

* DEF'S:

DEF	APLINP@	START OF PROCEDURE	
DEF	APLINP0	START OF CONTEXT	
DEF	APLINPUT		
DEF	BCX	BREF TO CX-CODESTRING EXEC.	
DEF	DWSIZIM	IMAGE DW SIZE	
DEF	EDITCK	EDIT CHECK COL. NO.	
DEF	INPDIR	DIRECT INPUT HANDLER	
DEF	INPEVAL		
DEF	INPF	INPUT FUNCTION LINE	
DEF	INPFAPND	INPUT FUN.LINE, APPEND.	
DEF	INP4MERG	INPUT LINE FOR MERGE	
DEF	INPLSCER	ENTRY TO INPUT FROM LINE-SCAN ERROR	
DEF	INPQUAD		
DEF	INPRET		
DEF	INPXEQ	'INPUT' FOR 'EXECUTE'	12-00012
DEF	INPXTEND	INPUT LINE EXTENSION	
DEF	LETTERS	DW 'A','Z'	
DEF	MNEMT1	MNEMONIC TABLE 1-CHAR	
DEF	MNEMT2	MNEMONIC TABLE 2-CHAR	
DEF	MNEMT3	MNEMONIC TABLE 3-CHAR	
DEF	OUTFLAG	O-U-T FLAG	
DEF	OVHWTABL	OVERSTRIKE CHAR TABLE	
DEF	PRMTIMAG	PROMPT WITH CONTENTS OF IMAGE	
DEF	QCNT	QUOTE COUNT	
DEF	SAVE312	SAVE AREA FOR APLINPUT	
DEF	SQUEEZER	ENTRY TO APLINPUT-SQUEEZE MNEM'S	

* STANDARD EQU'S

R0	EQU	0
R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6
R7	EQU	7
R8	EQU	8
R9	EQU	9
R10	EQU	10
R11	EQU	11

```

R12      EQU      12
R13      EQU      13
R14      EQU      14
R15      EQU      15
NBIO     EQU      3          NO. OF BLIND I/O CHANNELS+1
BLANK    EQU      X'40'
IDLE     EQU      X'16'
BYSIZIM  EQU      MAXCOL    HIGH END OF IMAGE TO BE BYTE-BLANKED
DWSIZIM  EQU      17        NO OF DW'S IN IMAGE TO BE BLANKED
PAGE

```

```

*
*  MODULE DESCRIPTION:
*
*  THIS MODULE INCLUDES THE INPUT ROUTINES AND PROCESSES FOR
*  APL, EXCEPT APL FILE I/O AND WORKSPACE MANAGEMENT I/O
*
*  INPUT OPERATIONS INCLUDED: BLIND INPUT HANDLER
*                               :QUOTE-QUAD INPUT HANDLER
*                               :EVALUATED INPUT HANDLER
*                               :DIRECT INPUT HANDLER
*                               :FUNCTION EDIT INPUT HANDLER
*  THE COMMON INPUT ROUTINE, APLINPUT, IS USED BY ALL EXCEPT
*  THE BLIND INPUT HANDLER
*
*  THE DIRECT INPUT HANDLER MAY ALSO BE CONSIDERED THE
*  DRIVER FOR APL-IT PERFORMS-VIA OTHER MODULES-NECESSARY
*  MEMORY MANAGEMENT PROCESSES PRIOR TO CALLING FOR INPUT
*
*  THIS MODULE INTERFACES WITH: THE CODESTRING GENERATION MODULE,
*                               THE MONITOR INTERFACE MODULE (APLUTSI),
*                               THE OUTPUT MODULE (APLOUT)
*                               THE CODESTRING EXECUTION MODULE
*                               THE FUNCTION DEF & EDIT MODULE
*                               THE WORKSPACE MANAGEMENT MODULE
*

```

```

PAGE
APLINP0  CSECT      0

```

```

*  FLAGS AND TEMPS IN APLINP
*

```

```

          BOUND      8
EDITCK    DATA      0,0          EDIT CHECK FLAG          U12-0004
OUTFLAG   EQU        EDITCK+1    0-U-T FLAG
QCNT      RES        1          QUOTE COUNT
SAVE312   RES        10         SAVE R3-R12 COMMON INPUT ROUTINE
BCHAR     RES        1          BAD CHAR FLAG
MNEMFLG   RES        1          MNEMONIC '%' FLAG-INPUT ROUTINE
INSWITCH  DATA      INPREAL    INPUT FETCH SWITCH
PAGE

```

```

APLINP@  CSECT      1

```

```

*  INPUT MNEMONIC TABLES
*  1. MNEMT3 - TABLE OF 3 CHARACTER MNEMONICS-4TH CHARACTER IS
*             THE CHARACTER REPRESENTED.
*
*             TABLE IS ORDERED. INSERTIONS MUST BE IN ORDER
*             MNEMT3SZ IS TABLE SIZE
*
*  2. MNEMT2 - HALFWORD TABLE OF 2 CHARACTER MNEMONICS.
*
*             TABLE IS ORDERED. INSERTIONS MUST BE IN ORDER
*             MNEMT2SZ IS TABLE SIZE
*
*  2.1 MNEMC2 - BYTE TABLE OF CHARACTERS REPRESENTED BY THE
*              MNEMONICS IN MNEMT2. ENTRIES MUST BE ORDERED TO
*              PAIR PROPERLY WITH MNEMT2 ENTRIES
*
*  3. MNEMT1 - BYTE TABLE OF SINGLE CHARACTER MNEMONICS
*
*             TABLE IS ORDERED
*             MNEMT1SZ IS SIZE
*
*  3.1 MNEMC1 - IS CORRESPONDING CHARACTER TABLE, ENTRIES
*              PAIRED WITH MNEMT1
*

```

```

*  **
*  * NOTE: IF THIS TABLE IS CHANGED, THE OUTMNEMT AND OUTTRANST TABLES

```

* ** MUST ALSO BE CHANGED

*

MNEMT3	DATA	X'C1C2E24F'	ABS	ABSOLUTE
	DATA	X'C3C1D750'	CAP	
	DATA	X'C3D6D459'	COM	COMMENT
	DATA	X'C3D7D34A'	CPL	CUP-LEFT
	DATA	X'C3D7D967'	CPR	CUP-RIGHT
	DATA	X'C3E4D75B'	CUP	
	DATA	X'C4C3C442'	DCD	DECODE
	DATA	X'C4C5D37F'	DEL	
	DATA	X'C4D3E348'	DLT	DELTA
	DATA	X'C4D9D764'	DRP	DROP
	DATA	X'C4D9E271'	DRS	DIERESIS
	DATA	X'C5C3C455'	ECD	ENCODE
	DATA	X'C6C3E341'	FCT	FACTORIAL
	DATA	X'D3D6C757'	LOG	
	DATA	X'D3D6D275'	LOK	LOCKED FUNCT
	DATA	X'D4C1E762'	MAX	MAXIMUM
	DATA	X'D4C4E5FE'	MDV	MATRIX DIVIDE
	DATA	X'D4C9D544'	MIN	MINIMUM
	DATA	X'D5D5C477'	NND	NAND
	DATA	X'D5D6D978'	NOR	
	DATA	X'D5D6E35F'	NOT	
	DATA	X'D6E4E300'	OUT	
	DATA	X'D9C4F163'	RD1	REDUCTION-1ST COORD
	DATA	X'D9C5E558'	REV	REVERSE
	DATA	X'D9D5C46F'	RND	RANDOM
	DATA	X'D9E5F168'	RV1	REVERSE-1ST COORD
	DATA	X'E3C1C205'	TAB	HORIZONTAL TAB
	DATA	X'E3C1D26A'	TAK	TAKE
	DATA	X'E3C2D954'	TBR	T-BAR
	DATA	X'E3D7E2B2'	TPS	TRANSPOSE
	DATA	X'E4C4D347'	UDL	UNDERSCORED DELTA
	DATA	X'E7D7C4B1'	XPD	EXPAND
	DATA	X'E7D7F1B0'	XP1	EXPAND-1ST COORD

*

MNEMT3X	RES	0
MNEMT3SZ	EQU	MNEMT3X-MNEMT3

*

* **

* * NOTE:IF THIS TABLE IS CHANGED,THE OUTMNEMT AND OUTRANST TABLES
* ** MUST ALSO BE CHANGED

*

*

		OVHWTABL. OVBYTABL IS NOT ORDERED IN ITSELF!	
	BOUND	4	
MNEMT2	DATA,2	0	OFFSET HALFWORD FOR BDR LOOP
	DATA,2	X'C2E2'	BS BACKSPACE
	DATA,2	X'C7C4'	GD GRADE DOWN
	DATA,2	X'C7C5'	GE GREATER THAN OR EQUAL
	DATA,2	X'C7D6'	GO BRANCH
	DATA,2	X'C7E3'	GT GREATER THAN
	DATA,2	X'C7E4'	GU GRADE UP
	DATA,2	X'C9C2'	IB I-BEAM
	DATA,2	X'C9E2'	IS ASSIGNMENT
	DATA,2	X'D3C5'	LE LESS THAN OR EQUAL
	DATA,2	X'D3E3'	LT LESS THAN
	DATA,2	X'D5C5'	NE NOT EQUAL
	DATA,2	X'D6D9'	OR
	DATA,2	X'D8D8'	QQ QUOTE QUAD
Q1	WHILE	Q1<NBIO	
	DATA,2	X'D8F0'+Q1-1	QI GRAPHIC AND BLIND I/O
	FIN		
	DATA,2	X'E2C3'	SC SMALL CIRCLE
	DATA,2	X'E4C1',X'E4C2',X'E4C3',X'E4C4',X'E4C5'	
	DATA,2	X'E4C6',X'E4C7',X'E4C8',X'E4C9'	
	DATA,2	X'E4D1',X'E4D2',X'E4D3',X'E4D4',X'E4D5'	
	DATA,2	X'E4D6',X'E4D7',X'E4D8',X'E4D9'	
	DATA,2	X'E4E2',X'E4E3',X'E4E4',X'E4E5'	
	DATA,2	X'E4E6',X'E4E7',X'E4E8',X'E4E9'	

* THE ENTRIES ABOVE ARE FOR UNDERSCORED A TO Z **

MNEMT2X	RES,2	0
MNEMT2SZ	EQU	HA(MNEMT2X)-HA(MNEMT2)-1

*

	BOUND	4	
MNEMC2	DATA,1	0	OFFSET BYTE FOR BDR LOOP
	DATA,1	X'08'	BS
	DATA,1	X'73'	GD
	DATA,1	X'76'	GE

```

DATA,1 X'FC' GO
DATA,1 X'6E' GT
DATA,1 X'46' GU
DATA,1 X'51' IB
DATA,1 X'FD' IS
DATA,1 X'74' LE
DATA,1 X'4C' LT
DATA,1 X'7B' NE
DATA,1 X'79' OR
DATA,1 X'52' QQ
Q2 WHILE Q2<NBIO
DATA,1 X'B6'+Q2-1 QI GRAPHIC AND BLIND I/O
FIN
DATA,1 X'5A' SC SMALL CIRCLE
DATA,1 X'81',X'82',X'83',X'84',X'85',X'86',X'87',X'88',X'89'
DATA,1 X'91',X'92',X'93',X'94',X'95',X'96',X'97',X'98',X'99'
DATA,1 X'A2',X'A3',X'A4',X'A5',X'A6',X'A7',X'A8',X'A9'

```

*

* **

* * NOTE:IF THIS TABLE IS CHANGED,THE OUTMNEMT AND OUTRANST TABLES

* ** MUST ALSO BE CHANGED

*

*

```

BOUND 4
MNEMT1 DATA,1 0 OFFSET BYTE FOR BDR LOOP
DATA,1 X'4D' ( LEFT BRACKET
DATA,1 X'5D' ) RIGHT BRACKET
DATA,1 X'72' - MINUS OPERATOR
DATA,1 X'C5' E EPSILON
DATA,1 X'C9' I IOTA
DATA,1 X'D6' O CIRCLE
DATA,1 X'D8' Q QUAD
DATA,1 X'D9' R RHO
DATA,1 X'E4' U UNDERSCORE
DATA,1 X'E6' W OMEGA

```

MNEMT1X RES,1

MNEMT1SZ EQU BA(MNEMT1X)-BA(MNEMT1)-1

*

```

BOUND 4
MNEMC1 DATA,1 0 OFFSET BYDE FOR BDR LOOP
DATA,1 X'B4' (
DATA,1 X'B5' )
DATA,1 X'60' -
DATA,1 X'45' E
DATA,1 X'49' I
DATA,1 X'56' O
DATA,1 X'53' Q
DATA,1 X'6C' R
DATA,1 X'6D' U
DATA,1 X'66' W

```

*

PAGE

*

* OVERSTRIKE TABLES-INPUT FORMATION OF OVERSTRIKES

* 1. OVHWTABL-TABLE OF HALFWORDS REPRESENTING VALID OVERSTRIKE
COMBINATIONS. LEFTMOST CHARACTER IS ALWAYS SMALLER.

* TABLE IS ORDERED-SMALLER TO LARGER VALUES

* INSERTIONS MUST BE IN ORDER

* 2. OVBYTABL-TABLE OF BYTE VALUES FOR VALID OVERSTRIKES

* MUST BE ORDERED TO MATCH TABLE POSITIONS WITH

* OVHWTABL ENTRIES

* 3. TABLSIZ IS NUMBER OF ENTRIES IN OVHWTABL(AND OVBYTABL)

* 4. PWR2 IS POWER OF 2 USED IN BINARY SEARCH.

* IF TABLSIZ IS = OR GREATER THAN 2EXP(N AND LESS THAN
2EXP(N+1),THEN PWR2 SHOULD BE EQUATED TO N

*

*

*

*

*

*

* **

* * NOTE:IF THIS TABLE IS CHANGED,THE OUTMNEMT AND OUTRANST TABLES

* ** MUST ALSO BE CHANGED

*

*

```

BOUND 4
OVHWTABL DATA,2 X'414B' *BANG DOT BANG1

```

	DATA, 2	X'417D'	*BANG	QUOTE	BANG2
	DATA, 2	X'4251'	*BASE	IBEAM	IBEAM1
	DATA, 2	X'4255'	BASE	TEE	IBEAM2
	DATA, 2	X'434F'	*BUCKS	ABS	BUCKS1
	DATA, 2	X'4361'	*BUCKS	COMPRESS	BUCKS2
	DATA, 2	X'43E2'	*BUCKS	S	BUCKS3
	DATA, 2	X'4648'	*GRADEUP	DELTA	GRADEUP1
	DATA, 2	X'464F'	*GRADEUP	ABS	GRADEUP2
	DATA, 2	X'4748'	*UDELTA	DELTA	UDELTA1
	DATA, 2	X'476D'	*UDELTA	USCORE	UDELTA2
	DATA, 2	X'484F'	DELTA	ABS	GRADEUP3
	DATA, 2	X'486D'	DELTA	USCORE	UDELTA3
	DATA, 2	X'4B5E'	*DOT	SEMI	SEMI1
	DATA, 2	X'4B6B'	*DOT	COMMA	COMMA1 (OF 1)
	DATA, 2	X'4B6F'	*DOT	QUERY	QUERY1 (OF 1)
	DATA, 2	X'4B7A'	*DOT	COLON	COLON1 (OF 1)
	DATA, 2	X'4B7D'	DOT	QUOTE	BANG3
	DATA, 2	X'4F56'	ABS	CIRC	ROTATE1
	DATA, 2	X'4F58'	*ABS	ROTATE	ROTATE2
	DATA, 2	X'4F73'	*ABS	GRADOWN	GRADOWN1
	DATA, 2	X'4F7F'	ABS	DEL	GRADOWN2
	DATA, 2	X'4FE2'	ABS	S	BUCKS4
	DATA, 2	X'5059'	*DOWNU	LAMP	LAMP1
	DATA, 2	X'505A'	DOWNU	OUTER	LAMP2
	DATA, 2	X'5155'	*IBEAM	TEE	IBEAM3
	DATA, 2	X'5253'	*QUAD	QUAD	QUAD1
	DATA, 2	X'527D'	*QUAD	QUOTE	QUAD2
	DATA, 2	X'537D'	QUAD	QUOTE	QUAD3
L1	WHILE	L1<NBIO			
	DATA, 2	X'53B6'+L1-1	*QUAD	QUADI	QUADI1
	FIN				
L2	WHILE	L2<NBIO			
	DATA, 2	X'53F0'+L2-1	QUAD	QUADI	QUADI2
	FIN				
	DATA, 2	X'53FB'	QUAD	DIVIDE	DOMINO1
	DATA, 2	X'53FE'	*QUAD	DOMINO	DOMINO2
	DATA, 2	X'5455'	*TBAR	TEE	TBAR1
	DATA, 2	X'5472'	*TBAR	NEGSIGN	TBAR2
	DATA, 2	X'5572'	TEE	NEGSIGN	TBAR3
	DATA, 2	X'5657'	*CIRC	LOG	LOG1
	DATA, 2	X'5658'	*CIRC	ROTATE	ROTATE3
	DATA, 2	X'565C'	CIRC	STAR	LOG2
	DATA, 2	X'5660'	CIRC	MINUS	FLIP1
	DATA, 2	X'5668'	*CIRC	FLIP	FLIP2
	DATA, 2	X'56B1'	CIRC	EXPAND	TRANSPOSE1
	DATA, 2	X'56B2'	*CIRC	TRANSPOSE	TRANSPOSE2
	DATA, 2	X'575C'	*LOG	STAR	LOG3
	DATA, 2	X'595A'	*LAMP	OUTER	LAMP3
	DATA, 2	X'5E6B'	*SEMI	COMMA	SEMI2
	DATA, 2	X'5E7A'	*SEMI	COLON	SEMI3
	DATA, 2	X'5F70'	NOT	AND	NAND1
	DATA, 2	X'5F75'	*NOT	LOCK	LOCK1
	DATA, 2	X'5F77'	*NOT	NAND	NAND2
	DATA, 2	X'5F78'	*NOT	NOR	NOR1
	DATA, 2	X'5F79'	NOT	OR	NOR2
	DATA, 2	X'5F7F'	NOT	DEL	LOCK2
	DATA, 2	X'6061'	MINUS	COMPRESS	MCOMP1
	DATA, 2	X'6063'	*MINUS	MCOMP	MCOMP2
	DATA, 2	X'6068'	*MINUS	FLIP	FLIP3
	DATA, 2	X'60B0'	*MINUS	MEXPAND	MEXPAND1
	DATA, 2	X'60B1'	MINUS	EXPAND	MEXPAND2
	DATA, 2	X'6163'	*COMPRESS	MCOMP	MCOMP3
	DATA, 2	X'61E2'	COMPRESS	S	BUCKS5
	DATA, 2	X'6B7A'	*COMMA	COLON	SEMI4
	DATA, 2	X'7077'	*AND	NAND	NAND3
	DATA, 2	X'737F'	*GRADOWN	DEL	GRADOWN3
	DATA, 2	X'757F'	*LOCK	DEL	LOCK3
	DATA, 2	X'7879'	*NOR	OR	NOR3
ENDOVHW	EQU	HA(ENDOVHW)-HA(OVHWTABL)+1			
TABLSIZ	WHILE	I<8			
I	DO	((1**I)<=TABLSIZ)&(1** (1+I)>TABLSIZ)			
PWR2	SET	I			
	FIN				
	FIN				
*					
*					
*	*NEGHWTBL-TABLE OF HALFWORDS WHICH 'CH' TREATS AS NEGATIVE				
*	USED FOR OVERSTRIKE VALIDITY TEST				

	BOUND	4	
NEGHWTBL	DATA, 2	0	OFFSET FOR BDR LOOP
	DATA, 2	X'B0B1'	*MEXPAND EXPAND MEXPAND3
	DATA, 2	X'B1B2'	*EXPAND TRANSPOSE TRANSPOSE3
L3	WHILE	L3<NBIO	
	DATA, 2	X'B6F0'+X'101'*(L3-1)	QUAD QUADI *QUADI3
	FIN		
	DATA, 2	X'FBFE'	*DIVIDE DOMINO DOMINO3
NEGHWTX	RES, 2	0	
NEGHWTSZ	EQU	HA(NEGHWTX)-HA(NEGHWTBL)-1	
	BOUND	4	
OVBYTABL	DATA, 1	X'41'	*BANG1
	DATA, 1	X'41'	*BANG2
	DATA, 1	X'51'	*IBEAM1
	DATA, 1	X'51'	IBEAM2
	DATA, 1	X'43'	*BUCKS1
	DATA, 1	X'43'	*BUCKS2
	DATA, 1	X'43'	*BUCKS3
	DATA, 1	X'46'	*GRADEUP1
	DATA, 1	X'46'	*GRADEUP2
	DATA, 1	X'47'	*UDELTA1
	DATA, 1	X'47'	*UDELTA2
	DATA, 1	X'46'	GRADEUP3
	DATA, 1	X'47'	UDELTA3
	DATA, 1	X'5E'	*SEMI1
	DATA, 1	X'6B'	*COMMA1
	DATA, 1	X'6F'	*QUERY1
	DATA, 1	X'7A'	*COLON1
	DATA, 1	X'41'	BANG3
	DATA, 1	X'58'	ROTATE1
	DATA, 1	X'58'	*ROTATE2
	DATA, 1	X'73'	*GRADOWN1
	DATA, 1	X'73'	GRADOWN2
	DATA, 1	X'43'	BUCKS4
	DATA, 1	X'59'	*LAMP1
	DATA, 1	X'59'	LAMP2
	DATA, 1	X'51'	*IBEAM3
	DATA, 1	X'52'	*QQUAD1
	DATA, 1	X'52'	*QQUAD2
	DATA, 1	X'52'	QQUAD3
B1	WHILE	B1<NBIO	
	DATA, 1	X'B6'+B1-1	*QUADI1
	FIN		
B2	WHILE	B2<NBIO	
	DATA, 1	X'B6'+B2-1	QUADI2
	FIN		
	DATA, 1	X'FE'	DOMINO1
	DATA, 1	X'FE'	*DOMINO2
	DATA, 1	X'54'	*TBAR1
	DATA, 1	X'54'	*TBAR2
	DATA, 1	X'54'	TBAR3
	DATA, 1	X'57'	*LOG1
	DATA, 1	X'58'	*ROTATE3
	DATA, 1	X'57'	LOG2
	DATA, 1	X'68'	FLIP1
	DATA, 1	X'68'	*FLIP2
	DATA, 1	X'B2'	TRANSPOSE1
	DATA, 1	X'B2'	*TRANSPOSE2
	DATA, 1	X'57'	*LOG3
	DATA, 1	X'59'	*LAMP3
	DATA, 1	X'5E'	*SEMI2
	DATA, 1	X'5E'	*SEMI3
	DATA, 1	X'77'	NAND1
	DATA, 1	X'75'	*LOCK1
	DATA, 1	X'77'	*NAND2
	DATA, 1	X'78'	*NOR1
	DATA, 1	X'78'	NOR2
	DATA, 1	X'75'	LOCK2
	DATA, 1	X'63'	MCOMP1
	DATA, 1	X'63'	*MCOMP2
	DATA, 1	X'68'	*FLIP3
	DATA, 1	X'B0'	*MEXPAND1
	DATA, 1	X'B0'	MEXPAND2
	DATA, 1	X'63'	MCOMP3
	DATA, 1	X'43'	BUCKS5
	DATA, 1	X'5E'	*SEMI4
	DATA, 1	X'77'	*NAND3
	DATA, 1	X'73'	*GRANDOWN3


```

        DATA,1  X'75'          *LOCK3
        DATA,1  X'78'          *NOR3
*
        BOUND    4
NEGBYTBL DATA,1  0              OFFSET FOR BDR LOOP
        DATA,1  X'B0'          *MEXPAND3
        DATA,1  X'B2'          *TRANPOSE3
B3        WHILE   B3<NBIO
        DATA,1  X'B6'+B3-1     *QUADI3
        FIN
ENDOVBY   DATA,1  X'FE'        *DOMINO3
*
        PAGE
*
*   CONSTANT REGION
*
        BOUND    8
LETTERS   DATA   X'000000C1'    A
        DATA   X'000000E9'    Z
ULETTERS  DATA   X'00000081'    A-UNDERSCORE
        DATA   X'000000A9'    Z-UNDERSCORE
UAUZ      DATA   X'FFFFFFE4C1'  UA +LEADING 1'S
        DATA   X'FFFFFFE4E9'  UZ +LEADING 1'S
FFFFFF00  DATA   X'FFFFFFF00'
FFFF0000  DATA   X'FFFFF0000'
        PAGE
*
*   APL INPUT DRIVERS:
*
*   INPDIR-DIRECT INPUT
*   INPEVAL-EVALUATED INPUT
*
*   THESE ROUTINES GENERATE APPROPRIATE PROMPTS
*   CALL RELEASER TO FREE UNUSED MEMORY
*   SET FLAGS
*   CALL APLINPUT TO INPUT A SOURCE LINE
*   CALL CS1 TO CODESTRING THE INPUT
*   BRANCH TO CX TO EXECUTE THE INPUT
*
INPDIR    LI,R11  1
        STW,R11  MODE           SET DIRECT MODE
        MTW,1    OBSFLAG        INCR. OBSERVE-FLAG (-1 = YES).
        BLEZ     INPSOURC
        LI,R11   0              SET IT ZERO IF GREATER.
        STW,R11  OBSFLAG
        B        INPSOURC
INPEVAL   LW,R8   ON%OFF         CHECK IF OFF LINE
        BEZ     INPSOURC        YES-SKIP PROMPT
        LI,R8   ERRORCHR        WA(EVALUATED INPUT PROMPT)
        LI,R9   2               SIZE (2 BYTES)
        STW,R9  MODE            ALSO-SET EVALUATED INPUT MODE
        LI,R10  1               BYTE DISP (BYTES 1&2 OF ERRORCHR)
        BAL,R7  WROUT           GENERATE PROMPT
INPSOURC  BAL,R11 PRMTBLNK       GENERATE BLANK PROMPT
        BAL,R11 RELEASER        RELEASE AVAILABLE CORE
INPLSCER  LI,R11  0             ENTRY FROM LINE-SCAN ERROR
        STW,R11 QCNT            RESET QUOTE COUNT
        MTW,0    BREAKFLG       CHECK FOR HANGUP
        BLZ     BCONTOFF        YES-DO CONTINUE AND OFF
        STW,R11 BREAKFLG       AND BREAK FLAG
INPRET    BAL,R11 APLINPUT       INPUT A LINE
        LI,R15  %              NOTE POS. IN R15 FOR ERR ANALYSIS.
BCSX      BAL,R12 CS1           CODESTRING THE LINE.
BCX       B      CX            COMMENCE FIRING
*
*   INPXTEND-EXTEND INPUT LINE (UNBALANCED QUOTES)
*
*   R11 USED.
*
INPXTEND  MTW,0    BREAKFLG      CHECK FOR HANGUP
        BLZ     BRESXTND        YES, HANGUP CURRENT CODESTRING.
        BAL,R11 APLINPUT        GET INPUT LINE -- NO PROMPT.
BRESXTND  B      RESXTEND      *BREF* RESUME FOR EXTENSION OF CURR. CS.
*
*   INPQUAD-ROUTINE TO INPUT 'TEXT' WHEN CALLED BY EX. OF QUOTE-QUAD
*
*   ON EXIT,R11=CHARACTER COUNT(INCLUDING CR)

```

```

*           R13=0 IF NOT 0-U-T EXIT
*           R13<0 IF HANG-UP DETECTED.
*   ALL OTHER REGISTERS SAVED AND RESTORED
*
*
*   LOCAL      NOPROMPT
INPQQUAD  LI,R11  0          SET TO INDICATE THAT
          STW,R11 OUTSIZ    INPUT IS NOT PREDECEDED BY PROMPT
          LW,R13  BREAKFLG  CHECK FOR HANG-UP...
          BLZ     BRESQQAD  YES, RESUME QQUAD PROCESSING NOW.
          LI,R11  255       BYTE MASK
          AND,R11 ERRORCHR  CHECK IF PROMPT REQUIRED
          BEZ     NOPROMPT   NO
          LCI     4          YES-SAVE R7-R10
          STM,R7  SAVE312
          LI,R8   ERRORCHR  WA(COLON)
          LI,R9   1          SIZE-1 BYTE
          LI,R10  2          OFFSET- COLON IS BYTE 2
          BAL,R7  CALWROUT  OUTPUT THE BEAST
          LCI     4
          LM,R7   SAVE312    RESTORE R7-R10
NOPROMPT  BAL,R11 APLINPUT  INPUT A LINE
QQINRET   LW,R11  HICOL
          AI,R11  -BA(IMAGE)+1 GET BYTE COUNT (INCLUDING CR)
          LW,R13  OUTFLAG    INDICATE 0-U-T IF PRESENT
BRESQQAD  B       RESQQUAD  *BREF* RESUME QUOTE-QUAD PROCESSING.
*
* INPF-INPUT A FUNCTION LINE,AFTER PROMPTING WITH LINE #
*
* ON ENTRY, LINE # IS IN IMAGE R3=CHARACTER COUNT
*
* EXIT IS TO FIHANDLER WITH INPUT LINE IN IMAGE AND HICOL SET
*
* MOST REGISTERS USED-NONE SAVED
*
INPF      LCW,R11 BREAKFLG    CHECK FOR HANGUP                      U12-0006
          BGZ     BCONTOFF    YES-- CONTINUE-OFF                    U12-0007
          AWM,R11 BREAKFLG    NO-- RESET BREAK IF SET                U12-0008
          BAL,R11 PRMTIMAG    GENERATE PROMPT                       U12-0009
INPFAPIN  BAL,R11 APLINPUT    INPUT LINE
BFIHNDLR  B       FIHANDLR   *BREF* GO TO FIHANDLR TO PROCESS
*
* INPFAPND--INPUT FUNCTION LINE APPENDAGE (SEE INPF)
*
INPFAPND  LCW,R11 BREAKFLG    CK FOR HANGUP...
          BGZ     BCONTOFF    YEP.
          AWM,R11 BREAKFLG    NOPE, CLEAR IF BREAK.
          B       INPFAPIN
*
* INP4MERG-ROUTINE TO MERGE INPUT FOR FUNCTION LINE EDIT
*
* USES MOST REGISTERS-SAVES NONE
*
* EXITS TO FIHANDLER
*
*
*   LOCAL      SINBUF,SINDEX1,SINDEX2
INP4MERG  STW,R3  HICOL      SAVE # OF CHARS
          LW,R6   ON%OFF     CHECK IF OFF-LINE
          BEZ     NOPROMPT   YES-SKIP THIS JAZZ
          BAL,R11 PRMTIMAG    NO-ISSUE PROMPT
          LW,R3   HICOL      GET NO. OF CHARS                      12-00001
          LI,R11  X'40'      SET BLANK                             12-00002
BLNKRT    STB,R11 IMAGE,R3    REPLACE 'IDLE' AND RIGHT END OF      12-00003
          AI,R3   1          IMAGE BUFFER WITH BLANKS               12-00004
          CI,R3   MAXCOL
          BL      BLNKRT     12-00005
          LW,R7   BSPFLAG    GET BSP FLAG 1=YES 0=NO                12-00006
          LW,R9   BLNKBKSP,R7 GET WORD OF BLANKS OR BSP'S
          LI,R6   DWSIZIM
SINBUF    STD,R9  INBUF-2,R6  FILL INBUF
          BDR,R6  SINBUF
          LB,R8   CRORBS,R7  SET CR OR BS AS 1ST CHAR.
          STB,R8  INBUF
          EXU     SININDEX1,R7 ESTABLISH INDEX
          EXU     SININDEX2,R7 FOR IDLE CHARACTER
          LI,R8   X'16'      IDLE
          STB,R8  INBUF,R5
          LW,R9   R5          SET SIZE

```

```

        AI,R9      1
        LI,R8      INBUF      ADDRESS
        BAL,R7     WROUTWB    -OUTPUT
NOPROMPT STW,R6     OUTSIZ     RESET OUTSIZ FOR APLINPUT
        LCW,R6     BREAKFLG   CHECK FOR HANGUP
        BGZ        BCONTOFF   YES-- CONTINUE-OFF
        LW,R6      MERGECOL
        AI,R6      -1         SET START POINT-1 IN IMAGE
        BAL,R11    APLINMRG    MERGE INPUT
        B          BFIHNDLR    GO TO FIHANDLR TO PROCESS
BLNKBKSP DATA     X'40404040',X'08080808'
CRORBS  DATA     X'15080000'
SINDEX1 LW,R5      MERGECOL    GET MERGECOL (BLANK COUNT)
        LW,R5      HICOL      GET HICOL
SINDEX2 AI,R5      1          ADD 1 FOR CR
        SW,R5      MERGECOL    SUBTRACT MERGECOL,FORM # OF BSP'S
*
* INPXEQ-FAKE INPUT FOR 'EXECUTE'
*
*      MAKES SPECIAL CALL TO APLINPUT
*      REGISTERS ALL VOLATILE
*      NORMAL EXIT TO CX      ERROR EXIT TO ERXEQ
*
INPXEQ  LI,R9      2
        STW,R9     MODE        SET EVALUATED INPUT MODE
        STW,R9     QCNT        SET QUOTE COUNT EVEN
        BAL,R11    APLINPUT    CALL APLINPUT
INPXEQR LW,R11     QCNT
        CI,R11     1          CHECK FOR UNBALANCED QUOTES
        BANZ       INPXEQUQ    YES
        BAL,R15    BCSX        NO, CODESTR. & XEQ THE LINE.
INPXEQUQ LI,R8      IDOPENQT    = ERROR I.D. FOR 'OPEN QUOTE'.
        BAL,R7     CMDERRH     HANDLE LIKE A CMD ERROR.
        LI,R3      -BA(IMAGE)
        AWM,R3     HICOL      READJUST HICOL
INPXEQBC LW,R3      HICOL
        BAL,R12    EDUMPLIN    DISPLAY OFFENDING LINE.
        B          ERXEQ      GO TO EXECUTE-OPERATOR ERROR RETURN.
        PAGE
*
* COMMON INPUT ROUTINE-HANDLES ALL TERMINAL INPUT EXCEPT 'BLIND' INPUT
*
*      INPUT IS READ INTO INBUF BY CALRDAPL
*
*      AN INPUT TRANSLATION TABLE, INTRANS, IS USED TO TRANSLATE
*      SINGLE CHARACTERS,AS REQUIRED,TO APL MAPPING. THIS
*      TABLE IS A FUNCTION OF TERMINAL TYPE
*
*      SPECIAL CHARACTERS ARE RECOGNIZED AND PROCESSED:
*      BADCHAR      ILLEGAL CHARACTER
*      NEWLINE      CR OR LF
*      MNEMONIC     %
*      FORMFEED     CONTROL-L ON TTY
*      EOTATTN      ATTN ON 2741
*      HTAB         HORIZONTAL TAB
*      IDLE         IDLE CHARACTER
*      INDEX        2741 LINE FEED(NO CR)
*      BACKSP       BACKSPACE
*      QUOTE
*
*      PROCESSED INPUT IS BUILT IN 'IMAGE',WHICH IS PRESET TO BLANKS
*
*      ORDINARY CHARACTERS ARE PROCESSED AS FOLLOWS:
*      BLANK        SKIP IT
*      IMAGE POSITION BLANK    STORE NEW CHARACTER
*      IMAGE POSITION QUOTE    DECREMENT QUOTE COUNT
*      IMAGE POSITION SAME
*      AS NEW CHARACTER      STOOE NEW CHARACTER
*      IMAGE POSITION DIFFERS
*      FROM NEW CHAR         DO OVERSTRIKE PROCESSING
*
*      OVERSTRIKE PROCESSOR SEARCHES FOR VALID OVERSTRIKE
*      LETTER-UNDERSCORE OVERSTRIKES ARE CHECKED AT ALPHTEST
*      OTHER OVERSTRIKES ARE CHECKED AT TABLTEST
*      OVHWTABL AND OVBYTABL TABLES ARE USED FOR SEARCH
*      BINARY SEARCH IS USED
*      CKOUT IS FINAL CHECK FOR SPECIAL OVERSTRIKE
*      0-BS-U-BS-T

```

```

*
*          BADCHAR IS REACHED IF NO VALID OVERSTRIKE IS FOUND
*
*
* BADCHAR PROCESSOR
*          FLAGS LOWEST BAD-CHARACTER IN 'IMAGE'
*          **** PROCESSING CONTINUES BECAUSE INPUT LINE MAY BE
*          **** COMPOSED OF MORE THAN 1 RECORD AND BECAUSE SUBSEQUENT
*          **** BS SEQUENCE AND ATTN MAY EXPUNGE BAD CHARACTERS
*
* HTAB PROCESSOR
*          SPACES INPUT ON 'IMAGE' AS INDICATED BY TAB SETTINGS
*
* BACKSP PROCESSOR
*          ADJUSTS IMAGE COLUMN POSITION FOR BACKSPACE
*
* MNEMONIC PROCESSOR
*          PROCESSES FORMS %XXX-GENERATES APL CHARS.
*          ACTUAL PROCESSING DEFERRED UNTIL END OF INPUT LINE
*          FIRST PASS RECORDS PRESENCE OF MNEMONIC START CHAR.
*
*          PROCESSING IS PRIMARILY LINEAR SEARCH OF ORDERED
*          TABLES: MNEMT3 FOR 3CHAR
*                   MNEMT2 FOR 2CHAR (RESULT IN MNEMC2)
*                   MNEMT1 FOR 1CHAR (RESULT IN MNEMC1)
*          % IS SET IF MNEMONIC NOT FOUND
*          %OUT SETS OUTFLAG AND BCHAR
*
* FORMFEED,EOTATTN,AND INDEX PROCESSORS
*          HANDLE LINE EDIT OPERATIONS
*          BLANK TO RIGHT OF EDIT (OR BCHAR) POSITION
*          DECREMENT QUOTE COUNT FOR DELETED QUOTES
*          ISSUE PROMPT AND CONTINUE INPUT AT EDIT POSITION
*
* NEWLINE PROCESSOR
*          CHECKS FOR BAD CHAR OR OUT
*          BAD CHAR RESULTS IN RIGHT END EDIT, BAD CHAR MESSAGE,
*          PROMPT WITH GOOD PART OF LINE, AND RETURN TO INPUT
*
*          OUT RESULTS IN SETTING LINE EMPTY AND LEAVING OUTFLAG
*          SET
*
*          NORMAL EXIT ADDS CR, SETS HICOL=SIZE, AND EXITS
*
* R11 IS LINK
*
* R3 TO R12 ARE SAVED,USED,AND RESTORED
*
* INTERNALLY,R6 POINTS TO POSITION IN IMAGE, R7(USUALLY) TO
* POSITION IN INBUF
*
* ROUTINES CALLED: CALWROUT -OUTPUT A FIXED PROMPT
*                   USES R7-R10
*                   BCHAROUT -CONVERT & OUTPUT IMAGE -
*                   MAY USE R3 TO R12
*
* APLINMRG  LOCAL  BLANKIT,RESEDIT
*           LCI      10          ENTRY FOR MERGED INPUT (LINE EDIT)
*           STM,R3    SAVE312
*           LW,R7     TERMTYPE    SET MNEMONIC FLAG IF          12-00038
*           AI,R7     -1          TERMINAL TYPE                12-00039
*           STW,R7    MNEMFLG     IS NOT 1                     12-00040
*           LI,R7     0
*           B         RESEDIT
* APLINPUT  LCI      10
*           STM,R3    SAVE312
*           LI,R6     -1          SAVE REGISTERS
*           STW,R6    HICOL       R6=COL
*           LI,R7     DWSIZIM     PRESET HICOL
*           LW,R9     BLANKS
* BLANKIT   STD,R9    IMAGE-2,R7   BLANK THE IMAGE BUFFER
*           BDR,R7    BLANKIT
*           STW,R7    MNEMFLG     SET MNEMONIC FLAG 'OFF'      12-00042
* RESEDIT   STD,R7    EDITCK      RESET EDITCK AND OUTFLAG
*           LI,R8     1000
*           STW,R8    BCHAR       PRESET BAD CHAR POSITION-TOO HIGH
*           LI,R8     INPREAL     SET INPUT SWITCH TO REAL INPUT
*           STW,R8    INSWTCH
*           CI,R11    INPXEQR     CHECK IF 'EXECUTE' ENTRY      12-00045
*           BNE      SETCAL      NO                               12-00046
*           MTW,INPFAKE-INPREAL  INSWTCH    SWITCH TO FAKE INPUT

```

```

        LI,R7      -1
        B          NEXTCOL
                                AND SKIP ACTUAL INPUT.
                                START AT INBUF.
*
* READ AN INPUT RECORD *
*
SETCAL   LW,R11    BREAKFLG      CHECK FOR 'HANGUP'
        BLZ        CRONLY        YES-DON'T TRY INPUT
        BAL,R11    RDAPL
*
* ACCESS A BYTE AND OPERATE ON IT *
*
NEXTCOL   LOCAL    OK,CHKTYPE
        BAL,R11    NEXTBYTE      GET & TRANSLATE BYTE
        AI,R6      1             UPDATE COL
        CI,R6      MAXCOL
        BL         OK
        LI,R8      IDTRUNC        = ERROR I.D. FOR 'TRUNCATED INPUT'.
        BAL,R7     CMDERRH        HANDLE LIKE A CMD ERROR.
        LW,R11     SAVE312+8      RESTORE LINK REGISTER      12-00060
        CI,R11     INPXEQR        CHECK FOR 'EXECUTE'          12-00061
        BNE        CRONLY        NO-SCRAP LINE                12-00062
        B          INPXEQBC       YES-DISPLAY THE BEAST        12-00063
OK        CW,R6     HICOL
        BLE        CHKTYPE
        STW,R6     HICOL          UPDATE HICOL AS INDICATED
CHKTYPE   CI,R5     BLANK          3 TYPES-SPECIAL,BLANK,ORDINARY
        BG         ORDINARY
        BL         SPECIAL,R5     BELOW BLANK IS A BRANCH VECTOR
        B          NEXTCOL        BLANK-NO CHANGE TO IMAGE
*
* ACCESS A BYTE AND TRANSLATE IT *
*
NEXTBYTE  AI,R7     1             UPDATE POINTER
        LB,R5      INBUF,R7      GET UNTRANSLATED BYTE
        B          *INSWITCH     (INPREAL OR INPFAKE)
INPREAL   BEZ       SETCAL        END OF RECORD BUT NOT OF LINE
INPXlate  LB,R5     INTRANS,R5    GET TRANSLATED VALUE OF BYTE
        B          *R11
INPFAKE   CI,R5     BLANK          CHECK VS BLANK
        BLE        INPXlate      LOW-NORMAL TRANSLATE IS OK
        CI,R5      X'7D'         CHECK VS QUOTE
        BNE        INPQD         NO
        LI,R5      9             YES-SET SPECIAL VALUE FOR QUOTE
        B          *R11          AND RETURN
INPQD     CI,R5     X'43'         CHECK VS DOLLAR SIGN
        BNE        *R11          NO-RETURN WITH NO TRANSLATION
        LI,R5      2             YES-SET SPECIAL VALUE FOR
        B          *R11          DOLLAR SIGN AND RETURN
*
* SPECIAL BYTE-BRANCH TO APPROPRIATE ROUTINE *
*
SPECIAL   B         BADCHAR      0
        B          NEWLINE      1 CR OR LF
        B          MNEMONIC     2 %
        B          FORMFEED     3 CONTROL L
        B          EOTATTN      4 ATTN
        B          HTAB         5 TAB
        B          IDLEPR       6 IDLE
        B          INDEX        7 INDEX
        B          BACKSP       8 BS
        LI,R5      X'7D'        9 QUOTE
        LB,R4      IMAGE,R6
        CI,R4      BLANK          CHECK FOR OVERSTRIKE
        BNE        CMPRBYTE
        MTW,1      QCNT          UPDATE QUOTE COUNT FOR TRUE QUOTE
        B          SETBYTE
*
* PROCESS ORDINARY BYTE *
*
ORDINARY  LOCAL
        LB,R4      IMAGE,R6
        CI,R4      BLANK          CHECK FOR OVERSTRIKE
        BE         SETBYTE        NO
        CI,R4      X'7D'         YES-CHECK FOR OLD QUOTE
        BNE        CMPRBYTE        NO
        MTW,-1     QCNT          YES-DECREMENT QUOTE COUNT
CMPRBYTE  CW,R5     R4            CHECK FOR DUPLICATION
        BNE        OVERSTRK       NO-OVERSTRIKE PROCESSING REQUIRED

```

```

SETBYTE STB,R5 IMAGE,R6 SET BYTE
B NEXTCOL AND CONTINUE
*
* OVERSTRIKE PROCESSING *
*
LOCAL ALPHTEST,UATOZ,UNDRSCR,CMPRHW,FOUND,UPPER,PLUS,MINUS
LOCAL NXTCMP, LASTCASE
LOCAL NEGHW, FOUNDNEG
OVERSTRK BG ALPHTEST CHECK R4 VS R5
XW,R4 R5 MAKE R4 THE LESSER
ALPHTEST CI,R4 X'6D' UNDERSCORE
BNE UATOZ NO
CLM,R5 LETTERS YES-CHECK LETTER RANGE
BCR,9 UNDRSCR OK
CLM,R5 ULETTERS NO-CHECK REDUNDANT UNDERSCORE
BCR,9 SETBYTE YES-SET IT
UATOZ CLM,R4 ULETTERS CHECK FOR UNDERSCORED LETTER
BCS,9 TABLTEST NO-GO TO TABLE TEST
AI,R4 BLANK YES-SWITCH TO LETTER
CW,R4 R5 COMPARE LETTERS
BNE BADCHAR NO-GO
UNDRSCR AI,R5 -BLANK SET UNDERSCORE
B SETBYTE
*
* TABLE LOOKUP PART OF OVERSTRIKE PROCESSING-BINARY SEARCH *
*
TABLTEST STB,R4 R4 R4 BYTE 0 SET TO LESSER BYTE
LI,R3 1
STB,R5 R4,R3 R4 BYTE 1 SET TO GREATER BYTE
LH,R4 R4 R4 SET FOR 'CH' WITH SIGN EXTENSION
BLZ NEGHW DAMN-NEGATIVE AW
LI,R3 (TABLSIZ-1)/2 HALFWAY POINT
LI,R8 1** (PWR2-2) BINARY INCREMENT
CH,R4 OVHWTABL,R3 CHECK WHICH HALF
BE FOUND SCORE!!
BG UPPER
LI,R3 (1** (PWR2-1)) -1 LOWER
B CMPRHW
MINUS SW,R3 R8 MOVE NEGATIVE
NXTCMP SLS,R8 -1 SHIFT POWER OF 2
AI,R8 0 CHECK IF LAST STEP
BE LASTCASE YES
CMPRHW CH,R4 OVHWTABL,R3 NO-COMPARE WITH TABLE ENTRY
BL MINUS
BG PLUS
FOUND LB,R5 OVBYTABL,R3 GET OVERSTRIKE BYTE
B SETBYTE
FOUNDNEG LB,R5 NEGBYTBL,R3
B SETBYTE
UPPER LI,R3 TABLSIZ-1** (PWR2-1)
B CMPRHW
PLUS AW,R3 R8 MOVE POSITIVE
B NXTCMP
LASTCASE CH,R4 OVHWTABL,R3 LAST CHANCE
BE FOUND OK
BNE BADCHAR NO DICE
NEGHW LI,R3 NEGHWTSZ CHECK IN THE
CH,R4 NEGHWTBL,R3 NEGATIVE HALFWORD
BE FOUNDNEG TABLE
BDR,R3 NEGHW+1 -SMALL TABLE-LINEAR SEARCH
LW,R3 SAVE312+8 GET LINK (R11)
CI,R3 QQINRET CHECK IF 0-U-T TEST IS RELEVANT
BNE BADCHAR NO
AI,R4 X'291C' CHECK FOR 'OU'-POSSIBLE 0-U-T
BNE BADCHAR NO CHANCE
BAL,R11 NEXTBYTE GET NEXT BYTE
CI,R5 8 BACKSPACE
BNE BADCHRM1 NO
BAL,R11 NEXTBYTE GET NEXT BYTE
CI,R5 X'E3' T
BNE BADCHAR NO-BAD CHARACTER
OUTFSET MTW,1 OUTFLAG YES-SET FLAG
B BADCHAR SCAN TO END OF INPUT
*
* BAD CHARACTER PROCESSOR *
*
LOCAL
BADCHRM1 AI,R7 -1

```

```

BADCHAR  CW,R6    BCHAR      CHECK BCHAR COLUMN
          BGE      NEXTCOL
          STW,R6   BCHAR
          B        NEXTCOL    SET BCHAR FLAG TO LOWER COLUMN
                                IF INDICATED-CONTINUE LINE SCAN
*
*  MNEMONIC  PROCESSOR      *
*
MNEMONIC  MTW,1    MNEMFLG    SET MNEMONIC CHECK FLAG
          LI,R5    X'43'      APL %
          B        ORDINARY
*
*  BACKSPACE PROCESSOR      *
*
BACKSP    AI,R6    -2         MOVE BACK 2 (FOR NET OF -1)
          BGEZ     NEXTCOL    CHECK UNDERFLOW
          LI,R6    -1         YES-SET TO START ON COL 0
          B        NEXTCOL
*
*  TAB       PROCESSOR      *
*
HTAB      LOCAL    ADDINT,SETABPOS,TABIND,TABNXT
          LW,R8    TABPNTR    CHECK IF TABS SET
          BEZ      NEXTCOL    NO-QUICK OUT
          CI,R8    128        CHECK INDIVIDUAL VS EQUIDISTANT TABS
          BGE      TABIND     INDIVIDUAL
          LI,R8    -1         EQUIDISTANT-SET LOOP
ADDINT    AW,R8    TABPNTR    GET NEXT SETTING
          CW,R6    R8         CHECK POSITION
          BGE      ADDINT     NOT YET
SETABPOS  LW,R6    R8         NOW!
          B        NEXTCOL
TABIND    LI,R3    0         INDIVIDUAL TABS
TABNXT    AI,R3    1         POINT TO NEXT SETTING
          LB,R8    TABVALS,R3 GET VALUE
          BEZ      NEXTCOL    QUIT ON ZERO
          AI,R8    -1         OFFSET FOR 0-INDEXING ON IMAGE
          CW,R6    R8         CHECK POSITION
          BL       SETABPOS   SCORE!
          B        TABNXT     LOOP
*
*  ERASERT-ROUTINE USED TO ERASE RIGHT END OF IMAGE AND CORRECT QCNT *
*
ERASERT   LOCAL    LOOP,NQUOTE
          LI,R10   BLANK
          LI,R4    BYSIZIM    SET END OF IMAGE
LOOP      LB,R5    IMAGE,R4   GET BYTE
          CI,R5    X'7D'      QUOTE
          BNE      NQUOTE     NO
          MTW,-1    QCNT       YES-DECREMENT COUNT
NQUOTE    STB,R10  IMAGE,R4   SET BLANK
          AI,R4    -1
          CW,R4    R6         CHECK IF COL PASSED
          BGE      LOOP       NO-LOOP
          STW,R6   HICOL      SAVE POSITION
          STW,R6   EDITCK     FLAG EDIT OPERATION
          AI,R6    -1         SET FOR NEXT ACCESS
          CW,R6    BCHAR      CHECK IF BAD CHAR INDICATOR
          BG       *R11       SHOULD BE RESET
          LI,R10   1000       YES(AT LEAST MAYBE)
          STW,R10  BCHAR
          B        *R11       YES-RETURN
*
*  FORMFEED PROCESSOR      *
*
FORMFEED  LOCAL    ATTNAPL,FRMFEED1
          LI,R8    LFEED
          LI,R9    2         PROMPT ADDRESS
          B        FRMFEED1   AND SIZE
                                USE EOTATTN ROUTINE
*
*  EOTATTN PROCESSOR      *
*
EOTATTN   LI,R8    EOTAPL     PROMPT WORD ADDRESS
          LW,R5    TERMTYPE    CHECK IF
          CI,R5    1         TERMINAL 1
          BE       ATTNAPL     YES
          LI,R8    EOTNAPL     NO-NEW PROMPT ADDRESS
ATTNAPL   LI,R9    5         PROMPT SIZE
FRMFEED1  BAL,R11  ERASERT    ERASE BEYOND ATTN

```

	LW, R11	SAVE312+8	RESTORE LINK REGISTER	12-00050
	CI, R11	INPXEQR	CHECK IF 'EXECUTE' CALL	12-00051
	BE	NEXTCOL	YES-NO INPUT PROMPT	12-00052
	LI, R10	0	BYTE DISPLACEMENT	
	BAL, R7	CALWROUT	OUTPUT EDIT PROMPT	
	STW, R10	OUTSIZ	ZERO OUTSIZ	
	B	SETCAL	GET NEXT INPUT RECORD	
* INDEX PROCESSOR *				
INDEX	BAL, R11	ERASERT	ERASE BEYOND INDEX	
	B	NEXTCOL		
* IDLE PROCESSOR *				
IDLEPR	AI, R6	-1	AVOID ADVANCING IMAGE POS	
	B	NEXTCOL		
* NEWLINE PROCESSOR *				
	LOCAL	MNEM1, MNEM2, MNEM3, MNEM4, TRY2, MNEM5, TRY1, MNEM6		
	LOCAL	TRYU, FOUND3, FOUND2, FOUND1, DOLLAR		
	LOCAL	NWLIN1, QOUT, QBCHAR		
NEWLINE	LI, R7	0		
	CW, R6	EDITCK	CHECK FOR ATTN-CR SEQUENCE	
	BE	QBCHAR	PROBABLY	
	STW, R7	EDITCK	NO-RESET EDITCK	
QBCHAR	LW, R6	BCHAR		
	CI, R6	1000		
	BNE	QOUT	YES-CHECK FOR OUT	
	MTW, 0	MNEMFLG	CHECK FOR POSSIBLE MNEMONIC	
	BEZ	NWLIN1	NO, THANK GOD	
SQUEEZER	LI, R6	0	YES, ONE MORE PASS (ENTRY-FOROPEN)	
	LI, R7	0	PRESET INDEXES FOR PASS	
MNEM1	LB, R5	IMAGE, R6	GET BYTE	
	CI, R5	X'43'	CHECK FOR APL %	
	BE	MNEM2	YES-CHECK FOR MNEMONIC	
SETCHAR	STB, R5	IMAGE, R7	SET CHARACTER (MAY OVERLAY ITSELF)	
	AI, R7	1	UPDATE	
	AI, R6	1	INDEXES	
	CW, R6	HICOL	CHECK IF DONE	
	BL	MNEM1	NO	
	STW, R7	HICOL	CORRECT HICOL FOR MNEMONIC SQUEEZE	
	B	NWLIN1	YES	
MNEM2	LI, R3	0	SET INDEX	
	LI, R10	3	AND LOOP	
MNEM3	AI, R6	1		
	LB, R5	IMAGE, R6		
	STB, R5	R4, R3	FORM C1, C2, C3, X IN R4	
	AI, R3	1		
	BDR, R10	MNEM3		
	LW, R5	FFFFFF00	SET MASK FOR COMPARISON	
	LI, R3	MNEMT3SZ	SET LOOP FOR SEARCH OF 3-CHAR MNEMS	
MNEM4	CS, R4	MNEMT3-1, R3	CHECK	
	BG	TRY2	TOO FAR-TRY 2-CHAR MNEMONICS	
	BE	FOUND3	FOUND!	
	BDR, R3	MNEM4	LOOP	
TRY2	AI, R6	-1	BACK UP 1	
	SLS, R4	-16	R4=0, 0, C1, C2	
	OR, R4	FFFF0000	SET LEFT HALF FOR 'CH'	
	LI, R3	MNEMT2SZ	SET LOOP	
MNEM5	CH, R4	MNEMT2, R3	SEARCH 2 CHAR MNEMONIC TABLE	
	BG	TRYU	NO CHANCE	
	BE	FOUND2	FOUND!	
	BDR, R3	MNEM5	LOOP	
TRY1	AI, R6	-1	BACK UP 1	
	EOR, R4	FFFF0000	RESET LEFT HALF TO 0	
	SLS, R4	-8	R4=0, 0, 0, C1	
	LI, R3	MNEMT1SZ	SET LOOP	
MNEM6	CB, R4	MNEMT1, R3	SEARCH SINGLE CHAR MNEMONIC TABLE	
	BG	DOLLAR	NOT A MNEMONIC	
	BE	FOUND1	FOUND!	
	BDR, R3	MNEM6	LOOP	
DOLLAR	AI, R6	-1	BACK UP 1	
	LI, R5	X'43'	APL %	
	B	SETCHAR	SET CHARACTER	
TRYU	CLM, R4	UAUZ	CHECK UA TO UZ	
	BCS, 9	TRY1	NO	


```

        AI,R4      X'1BC0'      YES-FORM LOWER CASE LETTER
        LW,R5      R4
        B          SETCHAR      SET CHARACTER
FOUND3  LI,R5      X'FF'        SET MASK
        AND,R5     MNEMT3-1,R3  GET BYTE-MATCH FOR 3-CHAR MNEM.
        BNEZ      SETCHAR      CHECK IF %OUT
        MTW,1      OUTFLAG      YES
        B          QOUT        TAKE OUT EXIT
FOUND2  LB,R5      MNEMC2,R3    GET BYTE-MATCH FOR 2-CHAR MNEM.
        B          SETCHAR      SET CHARACTER
FOUND1  LB,R5      MNEMC1,R3    GET BYTE-MATCH FOR 1-CHAR MNEM.
        B          SETCHAR      SET CHARACTER
NWLIN1  LW,R6      HICOL        NO-NORMAL EXIT
        LI,R5      X'15'
        STB,R5     IMAGE,R6     SET CR
        LI,R3      BA(IMAGE)
        AWM,R3     HICOL        ON EXIT, HICOL=BA(CARRIAGE RETURN)
        LCI        10
        LM,R3      SAVE312
        B          *R11
QOUT    MTW,0      OUTFLAG      RESTORE REGISTERS R3-R12
        BEZ        BCHNEWL     R11 IS LINK
        LI,R6      0           CHECK FOR OUT
CRONLY  STW,R6     HICOL        NO
        B          NWLIN1      SET LINE
                                TO 'CR'
                                ONLY
*
*   BAD CHARACTER RESOLUTION   *
*
BCHNEWL LI,R8      IDBADCH      = ERROR I.D. FOR 'BAD CHAR'.
        BAL,R7     CMDERRH      HANDLE LIKE A CMD ERROR.
        LW,R6      ON%OFF       CHECK IF OFF-LINE
        BEZ        BCONTOFF     YES-QUIT
        LW,R6      BCHAR        GET BAD CHAR
        BAL,R11    ERASERT
        LW,R11     SAVE312+8     RESTORE LINK REGISTER          12-00054
        CI,R11     INPXEQR       CHECK IF 'EXECUTE' CALL          12-00055
        BE         INPXEQBC      YES-DUMP NON-PROMPT PART-LINE    12-00056
        LI,R11     0
        STW,R11    EDITCK       RESET EDITCK
        AI,R6      1
        LI,R3      IDLE        SET IDLE
        STB,R3     IMAGE,R6     IN IMAGE FOR PROMPT
        LW,R3      R6
        BAL,R12    DUMPLINP     AS A PROMPT-LEAVE IMAGE INTACT
        LW,R6      ENDIMAGE     RESTORE IMAGE
        LI,R5      BLANK
        STB,R5     IMAGE,R6     REPLACE IDLE WITH BLANK
        AI,R6      -1          BACK UP-NEW INPUT APPENDS TO IMAGE
        LI,R5      0
        STW,R5     OUTSIZ       RESET TO IGNOR PROMPT
        B          SETCAL       RESUME INPUT
*
* TEXT FOR FIXED PROMPT MESSAGES *
*
LFEEED  DATA      X'20160000'   INDEX-IDLE
EOTAPL  DATA      X'20792008',X'16000000' IX-OR-IX-BS-IDLE
EOTNAPL DATA      X'207C2008',X'16000000' IX-@-IX-BS-IDLE
        PAGE
*
* PRMTIMAG-ROUTINE TO PROMPT WITH CHARACTERS IN IMAGE
*   ON ENTRY,R3=CHAR COUNT
*
* PRMTBLNK-ROUTINE TO PROMPT WITH 6 BLANKS
*
*
* R11-IS LINK
*   R3,R9,R12 USED
*
*   SUBROUTINE DUMPLINP USED(USES R12,R3,SAVES ALL OTHERS)
*
* PROMPT GENERATED ONLY IF ON-LINE (FAKED IF OFF-LINE)
*   EXIT TO CONTINUE ON 'HANG UP', AFTER FORCED FUNCT CLOSE IF NECESSARY
*
PRMTBLNK LW,R9     BLANKS       SET BLANKS
        STD,R9     IMAGE        IN BYTES 0-7
        LI,R3      6           OFFSET FOR 'IDLE'
PRMTIMAG LW,R9     BREAKFLG     CHECK FOR HANG-UP
        BLZ        HANGUP      YES-QUIT

```

	LCW, R9	ON%OFF	ON-LINE=-1, OFFLINE=0
	AI, R9	1	0 +1
	STW, R9	BREAKFLG	FAKE BREAK IF OFF-LINE (NO PROMPT)
	LI, R9	X'16'	SET 'IDLE'
	STB, R9	IMAGE, R3	
	BAL, R12	DUMPLINP	DUMP LINE WITHOUT CR
	LCW, R3	BREAKFLG	CHECK FOR HANGUP
	BGZ	BCONTOFF	YES
	AWM, R3	BREAKFLG	NO-RESET BREAK FLAG
	B	*R11	RETURN
HANGUP	LI, R14	BCONTOFF	HANG-UP DETECTED ON PROMPT
	LW, R13	MODE	CHECK MODE
	BEZ	B4CLOSE	FN OPEN FORCE CLOSE AND QUIT
	B	BCONTOFF	NO FN OPEN-QUIT
	END		

TITLE 'APLOUT-B00,08/21/73,DWG702985'
SYSTEM SIG7F
SYSTEM BPM
PAGE

* REF'S:

REF	APLINPUT	COMMON INPUT ROUTINE
REF	BITPOS	BIT TABLE
REF	BLANKS	X'40404040'
REF	BREAKFLG	BREAK OR 'HANGUP' FLAG
REF	BREPROMT	BRANCH TO REPROMPT.
REF	BRFUNDEF	RESUME IN FUNDEF MODULE.
REF	BSPFLAG	BACKSPACE VALIDITY FLAG
REF	CURRCS	CURRENT CS BLOCK POINTER
REF	DIGITS	
REF	DWSIZIM	DW'S IN IMAGE
REF	ERRORCHR	ERROR SIGNAL CHARACTER
REF	IMAGE	APL 'IMAGE' I/O BUFFER
REF	INBUF	APL INPUT-OUTPUT BUFFER
REF	INPDIR	DIRECT INPUT DRIVER
REF	INPF	FUNCTION EDIT INPUT DRIVER
REF	INPFAPND	INPUT FUN.LINE APPEND.
REF	MAXCOL	MAX COL POS FOR INPUT
REF	MERGECOL	MERGE OR EDIT COLUMN # (FROM 1)
REF	MNEMT1	MNEMONIC TABLE 1-CHAR
REF	MNEMT2	MNEMONIC TABLE 2-CHAR
REF	MNEMT3	MNEMONIC TABLE 3-CHAR
REF	MODE	MODE FLAG
REF	OFFSET	CODESTRING ERROR BYTE OFFSET
REF	ON%OFF	ONLINE-OFFLINE FLAG
REF	OUTMNEMT	OUTPUT MNEMONIC-OVST FLAG TABLE
REF	OUTRANST	OUTPUT TRANSLATION TABLE
REF	OUTMAXSZ	OUTPUT RECORD BLOCK SIZE
REF	OUTSIZ	SIZE OF OUTPUT RECORD(5)
REF	OVHWTABL	OVERSTRIKE HW TABLE
REF	PRMTIMAG	PROMPT WITH CURRENT IMAGE
REF	RDAPL	READ A RECORD
REF	SYMT	ADDRESS OF ADDRESS OF SYMBOL TABLE
REF	SYSTERR	SYSTEM ERROR
REF	TABVALS	TAB VALUES
REF	UNREF	END DECODE OPS FOR ERROR DIAG.
REF	VISIMAGE	BUFFER TO SAVE IMAGE DURING EDIT
REF	WIDTH	
REF	WROUTWB	WRITE OUTPUT-WORD BOUND
REF	WRTEXTC	WRITE OUTPUT-TEXTC FORM
REF	X4E1	X'4E100000'
REF	X1FFFF	ADDRESS MASK
REF	ZEROZERO	

* DEF'S:

DEF	APLOUT@	START OF PROCEDURE
DEF	APLOUT0	START OF CONTEXT
DEF	CHEKWID	CHECK WIDTH
DEF	CINTIM	CONVERT INTEGER INTO IMAGE
DEF	CREALBIN	CONVERT REAL TO BINARY.
DEF	DECODOPS	DECODESTRINGER
DEF	ENDIMAGE	END OF IMAGE IN USE
DEF	DUMLINE	
DEF	DUMPLING	LINE DUMP ROUTINE
DEF	DUMPLINP	DUMP IMAGE-PROBABLY AS PROMPT
DEF	EDECODOP	ERROR CALL TO: DECODOPS
DEF	EDUMPLIG	DUMPLING
DEF	EDUMPLIN	DUMLINE
DEF	FFFFFFFFE	
DEF	FLHALF	
DEF	FLONE	
DEF	FLTEN	
DEF	FUNLDISP	FUNCTION LINE DISPLAY
DEF	FUNLDIS%	
DEF	GENCHAR	
DEF	GENDIGS	GEN. EBCDIC DIGITS.
DEF	GENDIGSE	GEN. E-FORMAT
DEF	GENEXP	GEN. EXPONENT FORM.
DEF	GENNAME	
DEF	GENNAME0	GENNAME ENTRY FOR NO INDENT ON VFL
DEF	GENTEXT	TEXT GENERATOR ROUTINE
DEF	IDBUF	ID RECORD BUFFER
DEF	IMAGEPOS	IMAGE POSITION
DEF	IMAGES	
DEF	MAXREAL	MAX REAL #

13-00006

```

DEF      MIXEDOUT      COMPOUND STATEMENT OUTPUT
DEF      MTEMPIM2      MOVE TO IMAGE
DEF      NAMEBUF        NAME PTR BUFFER
DEF      PLUSREAL
DEF      RANKARR        ARRAY RANK
DEF      SAVESIX        SAVE TEMP-USED BY )COPY-SHARED
DEF      SETCHAR        STORES A CHAR.
DEF      SHOWFL        DISPLAY-APPEND-OR EDIT-FUNCTION LN.
DEF      SHOWSTOP
DEF      SIDR          SI DAMAGE WARNING REPORT
DEF      SINGOUT        SINGLE DATA BLOCK OUTPUT
DEF      STASHBL        STASH BLANK IN IMAGE
DEF      TABPNTR        TAB INDICATOR
DEF      TABPNTR1       TAB POINTER TEMP
DEF      TENSTBL        POWERS OF 10 (FIXED POINT)
DEF      TEXTC2I        MOVE TEXTC MSG TO IMAGE.

*      STANDARD EQU'S:
R0      EQU      0
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
BLANK   EQU      X'40'
IDLE    EQU      X'16'
*
*      CODESTRING PARAMETERS
*
CSBLOCK EQU      7      TYPE FOR CODESTRING DATA BLOCK
CSEND   EQU      139    OFFSET OF LAST CS VALUE
FNOFF   EQU      5      FUNCT NAME OFFSE1
PAGE
*      MODULE DESCRIPTION
*
*      THIS MODULE INCLUDES THE OUTPUT ROUTINES AND PROCESSORS
*      FOR APL-EXCEPT WORKSPACE FILE OUTPUT AND THE FILE OUTPUT SYSTEM
*      INCLUDED ARE DIRECT AND QUAD OUTPUT,BLIND OUTPUT, AND
*      OUTPUT GENERATED AS A RESULT OF SOME COMMAND ACTIVITY
*      OUTPUT CONVERSION ROUTINTES AS WELL AS OUTPUT IMAGE FORMATTING
*      IS INCLUDED
*
PAGE
APLOUT0 CSECT      0
*
*      VARIABLE PARAMETERS AND TEMPS
*
TABPNTR  DATA      0      TAB VALUE FLAG
TABPNTR1 RES        1      TEMP POINTER TO TABVALS
*      PROGRAM SWITCH IN DUMPLINE ROUTINE
GBYTOTAB LB,R4      IMAGE,R3      (OR B CHEKTABS)
*
ENDIMAGE RES        1      END OF IMAGE POINTER
CSERRBYT RES        1      CODESTRING ERROR BYTE
BOUND    8
IMAGES   DATA      0,0      NO. OF IMAGE LINES OUTPUT
IMERRBYT EQU        IMAGES+1  IMAGE ERROR BYTE POSITION
IMAGEPOS RES        1      IMAGE POSITION
LASTSCAN RES        1
SAVENDS  RES        1      SAVE BYTES ENCLOSING CODESTRING
CURRTAB  RES        1      CURRENT TAB-TEMP
OUTBUF   RES        1      INBUF-OUTBUF SIZE POINTER
BLCOUNT  RES        1      BLANK COUNT-TEMP
SAVE8    RES        1      TEMP
*
*      SAVE AND TEMP AREA FOR SINGOUT AND MIXEDOUT
*
*
*

```

* CONVRTR-PROGRAM SWITCH FOR OUTPUT CONVERSION ROUTINES SINGOUT-MIXEDOUT

*

```
CONVRTR  B      INTARRAY      (EXAMPLE)
        BOUND      8          @@@@@@@@
HIREAL   RES      2          @
DMAX     EQU      HIREAL      @
EFLAG    EQU      HIREAL+1    @
LOREAL   RES      2          @
DMINP5   EQU      LOREAL      @
BITCOUNT EQU      LOREAL+1    @@@@@@@@
        BOUND      8
```

```
IDBUF    RES      16          STORAGE FOR WS OR FIO ID RECORD
SAVE114   EQU      IDBUF      REG SAVE FOR MIXEDOUT-SINGOUT
ROWSIZE   EQU      IDBUF+14    ROW LENGTH-OUTPUT ROUTINES
RANKARR    EQU      IDBUF+15    RANK OF ARRAY-OUTPUT ROUTINES
        BOUND      8
NAMETEMP  RES      64          BUFFER TO BUILD OUTPUT NAMES
NAMEBUF   EQU      NAMETEMP    NAME POINTER BUFFER
BLOCKNUM  RES      1          NUMBER OF DATA BLOCKS TO OUTPUT
DBASE     RES      1          DATA BASE-OUTPUT ROUTINES
DSIZE     RES      1          DATA SIZE-OUTPUT ROUTINES
EVALUE    RES      1          EXPONENT VALUE-TEMP
LBASEM1   RES      1          BASE-1 OF LENGTH WORDS FOR ARRAY
PTRPTR    RES      1          DATA POINTER-POINTER
SAVESIX   EQU      BLOCKNUM    BLOCKNUM TO PTRPTR-SHARED WITH
*          SAVESIX. SAVESIX IS USED BY
*          WRACTIV IN MODULE APLUTSC,DURING
*          EXECUTION OF )COPY
```

```
SAVE212  RES      11          SAVE R2-R12 -BY-DUMPLINE
```

```
APLOUT@  CSECT      1
```

```
*          BOUND      8
* PRANGE-TABLE OF FLOATING POINT-LONG VALUES:1E0 TO 1E16
*
```

```
PRANGE   DATA,8  FL'1E-1'      ** PRANGE-2
          DATA,8  FL'1E0',FL'1E1',FL'1E2',FL'1E3',FL'1E4',FL'1E5'
          DATA,8  FL'1E6',FL'1E7',FL'1E8',FL'1E9',FL'1E10',FL'1E11'
          DATA,8  FL'1E12',FL'1E13',FL'1E14',FL'1E15',FL'1E16'
FLONE     EQU      PRANGE
FLTEN     EQU      PRANGE+2
FLHALF    DATA,8  FL'0.5'
FLTENS    DATA,8  FL'10',FL'10000',FL'1000000000000000'
MAXREAL   DATA    X'7FFFFFFF',X'FFFFFFFF'      MAX FL #
POWERS    DATA    1,4,13
```

```
* TENSTBL-TABLE OF POWERS OF 10 AS INTEGERS
```

```
TENSTBL  DATA    1,10,100,1000,10000,100000,1000000
          DATA    10000000,100000000,1000000000
```

```
FFFFFFFFE DATA    X'FFFFFFFFE'      MASK
```

```
*          PAGE
```

```
* DUMPLINE-ROUTINE TO DUMP A LINE FROM 'IMAGE' VIA F:OUT
```

```
* ENTRIES:DUMPLINE-NOT A PROMPT-LINE FOLLOWED BY CR
* DUMPLINP-LINE IS FUNCTION EDIT PROMPT-ENDS WITH 'IDLE'
```

```
* IF ENTRY IS FROM DUMPLINE,R3 IS THE BYTE OFFSET TO
* 1 CHARACTER PAST END OF IMAGE. ENDIMAGE OFFSET IS
* SAVED AND R3 IS SET TO 'BLANKS'. ON EXIT, R3 IS SET
* TO 6 AND THE FIRST 8 BYTES OF IMAGE ARE BLANKED
```

```
* IF ENTRY IS FROM DUMPLINP,THE LAST CHARACTER IS AN IDLE,
* INDICATING NO CR( TO COC HANDLER). ENDIMAGE IS SAVED,
* AND R3 IS SET 0. ON EXIT, IMAGE IS INTACT
```

```
* IMERRBYT IS TESTED AND,IF SET (>0) THE IMAGE OUTPUT IS
* FOLLOWED BY AN ERROR POINTER
```

```
* R12 IS LINK
```

```
* REGISTERS R2 TO R12 ARE SAVED AND RESTORED PRIOR TO EXIT
```

```
* INTERNALLY:R3 IS POINTER IN IMAGE
* R6 IS POINTER IN INBUF(OUTPUT BUFFER)
```

*
*
*
*
*
*
*

CALWR0UT IS CALLED FOR ACTUAL OUTPUT OF LINE
SEGMENTS:LINK R7,REGISTERS USED,R8,R9,R10

OUTPUT LINE SEGMENTS ARE BUILT SEQUENTIALLY IN 'INBUF' AND
LEFT IN PLACE FOR POSSIBLE SUBSEQUENT USE

	LOCAL	LCI,TABN,TABLESS,GETIMBYT,NOTABS,CHEKTABS,BUMPTAB	
	LOCAL	CHKBLNK,SETTAB,VARTABS,NOTYET,OUTBLNKS,BLOOP,UPBLCT	
	LOCAL	USCL,SINGLOUT,BSOVST,NEXTMOVE,FORWARD,BSSET,BLANKIT	
	LOCAL	LASTSEG,NOMOROUT,PUSHOUT1,PUSHOUT2	
	LOCAL	CARETSET	
EDUMLIG	EQU	DUMPLING	ERROR CALL TO DUMPLING.
DUMPLING	CI,R3	0	CHECK FOR NULL LINE
	BLE	*R12	YES-EXIT
	AI,R3	-1	ADJUST POINTER
DUMPLINP	STW,R3	ENDIMAGE	SAVE END OF IMAGE
	LI,R3	0	SET R3=0-FLAG FOR EXIT
	B	LCI	
EDUMLIN	EQU	DUMPLINE	ERROR CALL TO DUMPLINE.
DUMPLINE	AI,R3	-1	ADJUST IMAGE POINTER
	STW,R3	ENDIMAGE	SAVE IT
	LW,R3	BLANKS	SET R3= BLANKS-USED IN EXIT
LCI	LCI	11	
	STM,R2	SAVE212	
	MTW,1	IMAGES	KICK IMAGE COUNT
	LW,R3	ENDIMAGE	RESTORE R3 IMAGE END
	BLEZ	CKIMERR	IF ZERO,SKIP NEXT ITEM
	LB,R2	IMAGE,R3	CHECK FOR AND DELETE
	CI,R2	X'40'	TRAILING BLANKS
	BNE	%+2	
	BDR,R3	%-3	
	STW,R3	ENDIMAGE	SET NEW END OF IMAGE
CKIMERR	LCW,R6	IMERRBYT	CHECK IF IMAGE IS ERROR FLAGGED
	BGEZ	FORWARD	NO
	MTW,0	BSPFLAG	CHECK FOR BS CAPABILITY ON TERMINAL
	BEZ	FORWARD	NO
	CI,R3	MAXCOL	CHECK FOR WIDE IMAGE
	BG	FORWARD	YES
	SW,R3	IMERRBYT	GET DISTANCE FROM HI-END
	CW,R3	IMERRBYT	COMPARE WITH DISTANCE FROM LOW END
	BGE	FORWARD	
	AI,R3	1	
	LI,R5	8	BS
	LW,R4	ENDIMAGE	
BSSET	AI,R4	1	
	STB,R5	IMAGE,R4	LOOP-BS'S
	BDR,R3	BSSET	
	LI,R5	X'20'	IX
	AI,R4	1	
	STB,R5	IMAGE,R4	
	LB,R5	ERRORCHR	ERROR CARET-TERMINAL DEPENDANT
CARETSET	AI,R4	1	
	STB,R5	IMAGE,R4	
	STW,R6	IMERRBYT	SET IMERRBYT NEGATIVE
	STW,R4	ENDIMAGE	RESET ENDIMAGE
FORWARD	LW,R6	NOTABSET	SET SWITCH TO SKIP TAB CHECKS
	LW,R3	TABPNTR	CHECK FOR TAB SETS
	BEZ	TABLESS	NO
	LW,R6	TABSET	SET SWITCH TO USE TAB CHECKS
	CI,R3	128	YES-CHECK EQUIDISTANT VS INDIVIDUAL
	BL	TABN	EQUIDISTANT
	LI,R3	1	INDIVIDUAL
	STW,R3	TABPNTR1	GET FIRST SET
	LB,R3	TABVALS,R3	
TABN	STW,R3	CURRTAB	SET CURRENT TAB
	MTW,-1	CURRTAB	DECREMENT (FOR 0 INDEXING)
TABLESS	LI,R3	0	SET IMAGE POSITION AT START
	STW,R6	GBYTOTAB	PROGRAM SWITCH (TABS OR NOTABS)
	LI,R4	INBUF	
	STW,R4	OUTBUF	SET START OF OUTBUF=INBUF
	LI,R6	0	SET INBUF POSITION AT START
	STW,R6	BLCOUNT	ZERO BLANK COUNT
	LW,R12	OUTMAXSZ	
	AI,R12	-1	SET LOOP FOR LINE SEGMENT OUTPUT
GETIMBYT	EXU	GBYTOTAB	LB,R4 IMAGE,R3 OR B CHEKTABS
NOTABS	LB,R5	OUTRANST,R4	GET OUTPUT CHAR OR OFFSET
	LB,R7	OUTNMNEMT,R4	GET TYPE FLAG

	BEZ	SINGLOUT	SINGLE CHARACTER	
	CI, R7	4	CHECK TYPE FLAG	
	BE	BSOVST	OVERSTRIKE-GENERAL	
	BG	USCL	UNDERSCORED LETTER	
	AW, R5	MNEMTN-1, R7	** THIS POINT SHOULD BE REACHED ONLY	
	LW, R2	R5	** FROM LINE-SCAN ERROR-WIDTH SETTING	
	LW, R4	R7	** MAY BE EXCEEDED.	
	LI, R5	X'5B'	TRUE (NOT APL.) %	
	BAL, R11	PUSHOUT1	OUTPUT IT	
	LI, R11	%+1	SET LOOP	
	LB, R5	0, R2	GET MNEMONIC CHARACTER	
	AI, R2	1		
	BDR, R4	PUSHOUT1	OUTPUT MNEMONIC CHARACTERS	
	B	SINGLOUT		
CHEKTABS	LB, R4	IMAGE, R3	GET CHARACTER	
	CW, R3	CURRTAB	CHECK IF TAB SET REACHED	
	BL	NOTYET	NO	
	LW, R5	TABPNTR	CHECK TYPE OF TAB OPTION	
	CI, R5	128		
	BGE	VARTABS	INDIVIDUAL SETS	
BUMPTAB	AWM, R5	CURRTAB	VECTOR-KICK BY INTERVAL	
CHKBLNK	CI, R4	BLANK		
	BNEZ	OUTBLNKS	NOT BLANK-DUMP ANY PENDING	
	LI, R5	0		
	XW, R5	BLCOUNT	RESET BLANK COUNT	
	BEZ	NOTABS	IF IT WAS 0, OUTPUT BLANK	
SETTAB	LI, R4	5	SET TAB CHAR (INTERNAL) IN R4	
	B	NOTABS	GENERATE IT	
VARTABS	MTW, 1	TABPNTR1	POINT TO NEXT TAB	
	LW, R7	TABPNTR1		
	LB, R7	TABVALS, R7	GET NEXT TAB	
	BEZ	BUMPTAB	TOO FAR-SET TO ACCUMULATE BLANKS	
	AI, R7	-1	OFFSET	
	STW, R7	CURRTAB	SET NEW CURRTAB	
	B	CHKBLNK		
NOTYET	CI, R4	BLANK		
	BE	UPBLCT		
OUTBLNKS	LW, R2	BLCOUNT	CHECK FOR PENDING BLANKS	
	BEZ	NOTABS	NONE	
	LI, R5	BLANK	SOME-SET BLANK	
BLOOP	BAL, R11	PUSHOUT1	OUTPUT BLANK	
	BDR, R2	BLOOP	LOOP UNTIL BLCOUNT=0	
	STW, R2	BLCOUNT	REJET BLANK COUNT	
	B	NOTABS	GET OUTPUT CHAR	
UPBLCT	MTW, 1	BLCOUNT	KICK BLANK COUNT	
	B	NEXTMOVE		
*				
USCL	BAL, R11	PUSHOUT1	LETTER	
	LI, R5	8	BACKSPACE	
	BAL, R11	PUSHOUT1		
	LI, R5	X'6D'	UNDERSCORE	
	LB, R5	OUTRANST, R5	TRANSLATE IT	
SINGLOUT	LI, R11	NEXTMOVE		
	B	PUSHOUT1		
*				
BSOVST	LH, R8	OVHWTABL, R5	GET FIRST & SECOND CHARACTER IN R8	
	STW, R8	SAVE8	SAVE	
	LI, R5	2	SET INDEX	
	LB, R5	R8, R5	GET FIRST CHAR IN R5	
	LB, R5	OUTRANST, R5	TRANSLATE SINGLE CHARACTER	
	BAL, R11	PUSHOUT1	PUSH IT	
	LI, R5	8	BS	
	BAL, R11	PUSHOUT1		
	LI, R5	X'FF'	BYTE MASK	
	AND, R5	SAVE8	GET 2ND BYTE	
	LB, R5	OUTRANST, R5	AND TRANSLATE IT	
	B	SINGLOUT		
*				
NEXTMOVE	AI, R3	1		
	CW, R3	ENDIMAGE	CHECK IF IMAGE PROCESS COMPLETE	
	BLE	GETIMBYT	NO	
	MTW, 0	BLCOUNT	CHECK FOR TRAILING BLANKS	13-00001
	BEZ	LASTSEG	NO	13-00002
	LI, R5	BLANK	YES-INSERT 1 BLANK	13-00003
	BAL, R11	PUSHOUT1		13-00004
LASTSEG	SW, R12	OUTMAXSZ	YES	
	AI, R12	1	CHECK IF OUTPUT REMAINS	
	BEZ	NOMOROUT	NO	

	LCW, R9	R12	YES-SIZE IN BYTES
	BAL, R11	PUSHOUT2	GENERATE RECORD
	LW, R4	IMERRBYT	CHECK IF ERROR LINE FOLLOWS
	BLEZ	NOMOROUT	NO
	LW, R5	BLANKS	YES-BLANK IMAGE
	LI, R7	DWSIZIM	
BLANKIT	STD, R5	IMAGE-2, R7	
	BDR, R7	BLANKIT	
	LB, R5	ERRORCHR	GET ERROR CARET
	LI, R6	-1	FLAG -NEGATIVE- FOR IMERRBYT
	B	CARETSET	
NOMOROUT	STW, R6	OUTSIZ	SET OUTSIZ-IN CASE OUTPUT IS PROMPT
	LCI	11	
	LM, R2	SAVE212	RESTORE REGISTERS
	AI, R3	0	CHECK TYPE OF ENTRY
	BEZ	*R12	DUMPLINP-LEAVE IMAGE INTACT
	STD, R3	IMAGE	SET BYTES 0-7 OF IMAGE BLANK
	LI, R3	6	SET R3=6
	B	*R12	EXIT
*			
PUSHOUT1	STB, R5	INBUF, R6	STASH BYTE FOR OUTPUT
	AI, R6	1	
	BDR, R12	*R11	COUNT DOWN
	CW, R3	ENDIMAGE	CHECK COINCIDENT END OF IMAGE
	BE	LASTSEG	YES-OUTPUT LAST LINE SEGMENT
	LI, R12	IDLE	NO-SET IDLE IN LINE
	STB, R12	INBUF, R6	THIS SIGNALS -NO CR-
	AI, R6	1	
	LW, R9	OUTMAXSZ	MAX OUTPUT SIZE IN BYTES
PUSHOUT2	LW, R8	OUTBUF	OUTPUT BUFFER ADDRESS
	BAL, R7	WROUTWB	OUTPUT A LINE SEGMENT (OR LINE)
	SLS, R9	-2	SIZE IN WORDS
	AWM, R9	OUTBUF	SET NEW OUTBUF START
	LW, R12	OUTMAXSZ	RESET COUNTDOWN
	BDR, R12	*R11	DECREMENT (FOR IDLE) AND RETURN
*	PROGRAM SWITCHES		
NOTABSET	LB, R4	IMAGE, R3	GET BYTE FROM IMAGE
TABSET	B	CHEKTABS	CHECK TABS
	PAGE		
*			
*	STASH ROUTINES		
*	R11 IS LINK		
*	R3 UPDATED R8 USED		
*	R12 USED IF DUMPLINE CALLED		
*	DUMPLINE SAVES & RESTORES REGS. EXCEPT R3-UPDATED		
*			
QSTASHBL	LI, R8	1	CHECK OLD LASTSCAN-SET NEW
	XW, R8	LASTSCAN	LASTSCAN NON-ZERO
	BEZ	*R11	EXIT IF OLD LASTSCAN=0
	CW, R3	IMERRBYT	CHECK IF THIS IS ERROR POINT
	BNE	STASHBL	NO
	MTW, 1	IMERRBYT	YES-KICK POINTER PAST BLANK
STASHBL	LI, R8	BLANK	SET BLANK
STASHIM	CW, R3	WIDTH	CHECK WIDTH OVERFLOW
	BL	SETR8	NO
	BAL, R12	DUMPLINE	YES-DUMP LINE FIRST
SETR8	STB, R8	IMAGE, R3	STASH BYTE
	AI, R3	1	UPDATE POSITION
	B	*R11	EXIT
	PAGE		
*			
*	CHECKWID-CHECK IF WIDTH PERMITS PRINTING MULTI-CHARACTER FIELD		
*	IF NOT, DUMP LINE-R7 IS NOT RESTORED		
*			
CHEKWID	AW, R7	R3	ADD POSITION TO FIELD WIDTH-1
	CW, R7	WIDTH	
	BL	*R11	OK
	BAL, R12	DUMPLINE	NO-DUMP LINE
	B	*R11	
	PAGE		
*			
*	GENMNEM-GENERATE OUTPUT MNEMONIC IN IMAGE		
*			
*			
*	ON ENTRY TO GENMNEM:		
*	R3 POINTS TO IMAGE BYTE FOR STASHING OUTPUT		
*	R4 CONTAINS THE INTERNAL APL CHARACTER TO BE REPLACED BY MNEM.		
*	R5 IS LINK		


```

*          R6 POINTS TO SOURCE-CODESTRING OR TEXT
*          R7 CONTAINS MNEMONIC TYPE 1CHAR,2CHAR OR 3CHAR.
*
* GENMNEM USES:
*   R2,R5,R7,R8,R11,R12  CHANGED
*   R3-UPDATED
*   R4-UNCHANGED
*
* GENMNEM CALLS:
*   CHECKWID (WHICH USES R3,R7,R11,R12)
*   SETR8    (WHICH USES R3,R8,R11)
*   DUMPLINE (WHICH USES R12-SAVES & RESTORES OTHER REGISTERS
*               CHANGES R3)
*
*
* GENMNEM LOCAL LOOP%
*          BAL,R11 CHEKWID          CHECK IF MNEMONIC WILL FIT
*          LB,R7   OUTMNEMT,R4      RESTORE MNEMONIC TYPE IN R7
*          LB,R2   OUTRANST,R4      GET TABLE OFFSET
*          AW,R2   MNEMTN-1,R7      GET ADDRESS OF MNEMONIC
*          LI,R8   X'43'           %
*          BAL,R11 SETR8            STASH IT
*          AI,R7   1
*          LI,R11  LOOP%           SET LOOP
* LOOP%     LB,R8   0,R2           GET MNEMONIC BYTE(S)
*          AI,R2   1
*          BDR,R7  SETR8            STASH
*          B       0,R5           RETURN
*
* * MNEMTN-BYTE ADDRESSES OF MNEMONIC TABLES
*
* MNEMTN   DATA    BA(MNEMT1)
*          DATA    BA(MNEMT2)
*          DATA    BA(MNEMT3)
*          PAGE
*
* * GENTEXT-ROUTINE TO GENERATE TEXT IN IMAGE BUFFER
* * MNEMONICS ARE EXPANDED,OVERSTRIKES ARE NOT
* * IF ENTRY FROM CODESTRING TEXT, QUOTES ARE DOUBLED
*
*          R13=LINK
*          R10=BYTE COUNT-DECREMENTED TO 0
*          R6  POINTS TO FIRST SOURCE BYTE-1 ON ENTRY
*              POINTS TO LAST SOURCE BYTE ON EXIT
*          R3  POINTS TO IMAGE BYTE OFFSET ON ENTRY
*              POINTS TO NEXT IMAGE BYTE OFFSET ON EXIT
*          R4,R5,R7,R8 USED INTERNALLY
*
*          ROUTINES CALLED:
*              STASHIM (USES R3,R8,R11,R12)
*              DUMPLINE(USES R12-USES&SAVES OTHER REGISTERS-CHANGES
*                      R3)
*              GENMNEM (USES R2,3,4,5,7,8,AND 11)
*
*
*          LOCAL    GENTEXT2,SETR4,GENTEXT3,GENTEXT4
*          LOCAL    LOOP
* GENTEXT1  LI,R10   1              SET LOOP SIZE 1
*          B       GENTEXTC        BYPASS SET-UP OF R4
* GENTEXT  AI,R10   0              CHECK FOR EMPTY TEXT
*          BEZ     *R13            YES-QUICK EXIT
* LOOP     AI,R6    1              POINT TO SOURCE BYTE
* GENTEXTM LB,R4    0,R6           GET BYTE
* GENTEXTC LB,R7    OUTMNEMT,R4    CHECK FOR MNEMONIC
*          BEZ     SETR4           NO
*          CI,R7    4              MAYBE
*          BGE     SETR4           NO-(DON'T EXPAND OVERSTRIKES)
*          BAL,R5   GENMNEM       YES-GENERATE MNEMONIC
* GENTEXT2 BDR,R10  LOOP
*          B       *R13           EXIT
* SETR4    CI,R4    X'15'         CHECK FOR EMBEDDED CR
*          BE      GENTEXT3       YES
*          CI,R13   CSGNTEXT+1    CHECK IF CALLED FROM CS-TEXT
*          BNE     GENTEXT4       NO-QUOTES NOT DOUBLED
*          CI,R4    X'7D'         YES-CHECK FOR QUOTE
*          BNE     GENTEXT4       NO
*          LI,R8    X'7D'         YES-DOUBLE IT
*          BAL,R11  STASHIM       STASH IT
* GENTEXT4 LW,R8    R4            SET R8 FOR STASH
*          LI,R11   GENTEXT2      SET RETURN

```

	B	STASHIM	SET BYTE
GENTEXT3	LI,R12	GENTEXT2	SET RETURN
	AI,R3	0	CHECK IF CR AT START OF LINE
	BGZ	DUMPLING	NO
	LI,R4	X'40'	YES
	STB,R4	IMAGE	GEN BLANK
	LI,R3	1	
	B	DUMPLING	
	PAGE		

```

*
* CINTIM -CONVERT INTEGER TO EBCDIC AND PUT IN IMAGE
*
*   ON ENTRY:R3 IS OFFSET POINTER TO IMAGE
*             R6 POINTS TO NEXT SOURCE BYTE-DO NOT CHANGE
*             R10 IS EXTERNAL LOOP COUNTER-DO NOT CHANGE
*             R13 IS LINK
*             R2,R3,R7,R8,R9,R11,R12 USED AND CHANGED
*
*   ON EXIT, R3 POINT TO NEW OFFSET IN IMAGE
*
*   'CHEKWID' IS USED TO CHECK FOR WIDTH OVERFLOW-IF SO,
*   IMAGE IS DUMPED BEFORE MOVE IF WIDTH EXCEEDED.
*
* CINTIM IS USED FOR INTEGER SCALARS AND VECTORS-ARRAYS ARE
* TREATED SEPARATELY
*

```

	LOCAL	CHEKSIZ, SIZFOUND, POSINT	
CINTIM	LAW,R9	R8	GET ABS VALUE IN R9
	LI,R2	9	NO-CHECK SIZE
CHEKSIZ	CW,R9	TENSTBL,R2	
	BGE	SIZFOUND	
	BDR,R2	CHEKSIZ	
SIZFOUND	AI,R2	1	ADJUST COUNT
	LW,R7	R2	USE SIZE TO CHECK FOR IMAGE FIT
	AI,R8	0	
	BGEZ	POSINT	NON-NEG NUMB
	AI,R7	1	NEG NUMB-KICK SIZE FOR SIGN BIT
POSINT	BAL,R11	CHEKWID	
	AI,R8	0	CHECK SIGN AGAIN
	BGEZ	DIVLOOP	POSITIVE
	LI,R8	X'72'	NEGATIVE-SET SIGN
	BAL,R11	SETR8	
DIVLOOP	LI,R8	0	
	DW,R8	TENSTBL-1,R2	DIVIDE VALUE BY POWER OF 10
	XW,R8	R9	R8=DIGIT, R9=REMAINING VALUE
ZERINT	AI,R8	X'F0'	CONVERT TO EBCDIC
	BAL,R11	SETR8	STASH VALUE IN IMAGE
	BDR,R2	DIVLOOP	LOOP
	B	*R13	EXIT*

```

*
* PAGE
*

```

```

* CINTARR-CONVERT INTEGER FOR ARRAY OUTPUT
* FIXED FIELD WIDTH=DMAS+2
* NUMBER IS RIGHT JUSTIFIED IN FIELD
*
* R13=LINK
* R10 IS EXTERNAL LOOP COUNTER-DO NOT CHANGE
* R3 IS IMAGE OFFSET AND IS CHANGED
* R2,R7,R8,R9,R11,R12 USED AND CHANGED
*

```

	LOCAL	CHEKSIZ, SIZFOUND	
CINTARR	LW,R7	DMAX	GET
	AI,R7	2	FIELD WIDTH
	BAL,R11	CHEKWID	AND CHECK IMAGE OVERFLOW
	LAW,R9	R8	GET +VALUE
	LW,R7	R8	SAVE VALUE FOR SIEN & 0 TESTS
	LW,R2	DMAX	SET LOOP SIZE
CHEKSIZ	CW,R9	TENSTBL-1,R2	CHECK SIZE OF VALUE
	BGE	SIZFOUND	
	BAL,R11	SETR8BL	SET BLANK
	BDR,R2	CHEKSIZ	LOOP
SIZFOUND	BAL,R11	SETR8BL	SET 1 MORE BLANK
	LI,R11	DIVLOOP	
	LW,R8	R7	CHECK VALUE
	BGZ	SETR8BL	+,SET 1 MORE BLANK
	BEZ	ZERINT	0, SHORTCUT
	LI,R8	X'72'	APL -

```

      B      SETR8
SETR8BL  LI,R8  X'40'      SET BLANK
      B      SETR8
      PAGE

*
*   CREALBIN-CONVERTS FL-LONG VALUE IN R8-R9  TO INTEGER IN R8-R9
*   WITH-DECIMAL POSITION IN R12
*   R12=0 INDICATES .XXX
*   R12>0 INDICATES NO.DIGITS LEFT OF DECIMAL
*   R12<0 INDICATES NO.ZEROS BETWEEN DECIMAL AND FIRST DIGIT
*
*   R11 IS LINK
*   R7=X'72' (APL-SIGN) IF  R8-R9 NEGATIVE,BLANK IF NOT
*   R2 USED INTERNALLY
*   R5 USED INTERNALLY
*
CREALBIN  LOCAL  SCALER,ROUNDER,FIXIT,ZEROREAL,DownSCL,UPSCALE
          LW,R5  DIGITS      PRECISION SETTING
          LI,R7  BLANK      SIGN FLAG-OFF
          AI,R8  0          CHECK SIGN AND VALUE
          BEZ    ZEROREAL    0
          BGZ    PLUSREAL    +
          LI,R7  X'72'      -
          LCD,R8  R8        USE POSITIVE VALUE
PLUSREAL  LW,R12  R5        GET SIGNIFICANCE
          LI,R2  3          SET SCALER LOOP
SCALER    CD,R8  PRANGE,R5  CHECK HIGH END
          BGE    DownSCL    TOO HIGH
          CD,R8  PRANGE-2,R5 CHECK LOW END
          BL     UPSCALE    TOO LOW
ROUNDER   FAL,R8  FLHALF    ROUND OFF
          CD,R8  PRANGE,R5  CHECK OVERFLOW OF RANGE
          BL     FIXIT      NO
          AI,R12 1          YES-
          LD,R8  PRANGE-2,R5 ADJUST
FIXIT      FAL,R8  X4E1      CONVERT TO BINARY INTEGER
          SW,R8  X4E1
          B      *R11      EXIT
ZEROREAL  LI,R9  0          ASSURE TRUE 0
          LI,R12 1          SET DECIMAL POS
          B      *R11      EXIT
DownSCL   FDL,R8  FLTENS-2,R2 DIVIDE BY POWER OF 10
          AW,R12  POWERS-1,R2 ADJUST DECIMAL POSITION
          CD,R8  PRANGE,R5  CHECK
          BGE    DownSCL    -MORE-
          BDR,R2 SCALER     CHECK AGAIN
          B      ROUNDER    FALL-THROUGH-SHOULDN'T BE REACHED
UPSCALE   FML,R8  FLTENS-2,R2 MULTIPLY BY POWER OF 10
          SW,R12  POWERS-1,R2 ADJUST DECIMAL POSITION
          CD,R8  PRANGE-2,R5 CHECK
          BL     UPSCALE    NEEDS MORE
          BDR,R2 SCALER     (SHOULD ALWAYS BRANCH)
          B      ROUNDER    FALL-THROUGH-SHOULDN'T BE REACHED
          PAGE

*
*   GENDIGS-ROUTINE TO GENERATE EBCDIC DIGITS
*   FROM BINARY VALUE IN R8-R9
*   DECIMAL INDICATOR IN R12
*
*   GENDIGSE-ENTRY POINT FOR E FORMAT
*   R4=LINK
*   R2,R5,R7,R11 USED INTERNALLY
*
*   NO LEADING OR TRAILING BLANKS GENERATED
*   NO TRAILING ZEROS RIGHT OF DECIMAL POINT
*
          LOCAL  ZLOOP,CDIGITS,DIVID,DIVIDS,NONZERO,NODOT
          LOCAL  ZEROPUT,ZLOOP2
GENDIGSE  AI,R12  -1
          STW,R12  EVALUE    SAVE 'E' VALUE
          LI,R12  1          SET TO GEN FORM  X.XX---
GENDIGS   AI,R12  0          CHECK DECIMAL INDICATOR
          BGZ    CDIGITS
          BAL,R11 SETZER      0
          LI,R7  X'4B'      .
          BAL,R11 SETCHAR
          AI,R12  0          CHECK AGAIN

```

	BEZ	CDIGITS	NO LEADING ZEROS
	LI, R11	ZLOOP	
	AI, R12	-1	ADJUST COUNT
ZLOOP	BIR, R12	SETZER	OUTPUT ZEROS AS REQUIRED
CDIGITS	LI, R7	0	SET SHORT FORM (LOW HALF=0)
	LW, R2	DIGITS	CHECK DIGITS
	CI, R2	10	
	BLE	DIVIDS	OK DIVIDE R8-R9 BY PWR OF 10
	DW, R8	TENSTBL+9	DAMN: EXTENDED FORM
	LW, R7	R8	SAVE LOW HALF OF VALUE
	AI, R2	-9	ADJUST 'DIGITS'
DIVID	LI, R8	0	CLEAR R8
DIVIDS	DW, R8	TENSTBL-1, R2	DIVIDE BY POWER OF 10
	XW, R8	R9	SHIFT DIGIT TO R8-REST TO R9
	AI, R8	X'F0'	FORM EBCDIC VALUE
	STB, R8	0, R5	STASH IT
	AI, R5	1	
	AI, R9	0	CHECK FOR 0 REMAINDER
	BNEZ	NONZERO	NO
	AI, R7	0	MAYBE-CHECK LOW HALK
	BEZ	ZEROPUT	YES
NONZERO	AI, R12	-1	NO-CHECK FOR DECIMAL POSITION
	BNEZ	NODOT	NO
	LI, R8	X'4B'	YES
	STB, R8	0, R5	
	AI, R5	1	
NODOT	BDR, R2	DIVID	LOOP ON 'DIGITS'
	XW, R9	R7	DONE-CHECK FOR EXTENDED FORM
	BEZ	0, R4	NO
	LI, R2	9	DIGITS GR THAN 10-TWO PHASES OF
	B	DIVID	DIVISION
ZEROPUT	LI, R11	ZLOOP2	
ZLOOP2	BDR, R12	SETZER	
	B	0, R4	
SETZER	LI, R7	X'F0'	
SETCHAR	STB, R7	0, R5	
	AI, R5	1	
	B	*R11	
	PAGE		
*			
* GENEXP-GENERATE EX EXX E-X OR E-XX			
* R4=LINK R7, R8, R9, R11, R5 USED			
*			
GENEXP	LOCAL	POSEXP, ONECHAR	
	LI, R7	'E'	GENERATE 'E'
	BAL, R11	SETCHAR	
	LW, R9	EVALUE	GET EXP. VALUE
	BGEZ	POSEXP	
	LI, R7	X'72'	SET - IF NEGATIVE
	BAL, R11	SETCHAR	
	LAW, R9	EVALUE	
POSEXP	LI, R8	0	
	DW, R8	TENSTBL+1	SEPARATE EVALUE INTO 10'S AND 1'S
	BEZ	ONECHAR	NO 10'S
	LW, R7	R9	
	AI, R7	X'F0'	SET 10'S DIGIT
	BAL, R11	SETCHAR	
ONECHAR	LW, R7	R8	
	LW, R11	R4	
	AI, R7	X'F0'	SET 1'S DIGIT
	B	SETCHAR	AND EXIT
	PAGE		
*			
* CREALIM-CONVERT REAL VALUE IN R8-R9 TO PRINT FORMAT AND			
* PLACE IN IMAGE BUFFER			
*			
* THIS ROUTINE IS FOR CONVERSION OF SCALARS OR ELEMENTS			
* OF VECTORS-LEADING OR TRAILING BLANKS ARE **NOT** GENERATED			
* (BECAUSE OF USE IN CODESTRING OUTPUT)			
*			
* R13=LINK			
*			
* REGISTERS R1, R2, R4, R5, R7, R11, R12 USED INTERNALLY			
* REGISTER 3 USED AND UPDATED(IMAGE POINTER)			
*			
* ROUTINES USED: CREALBIN-CONVERT TO INTERMEDIATE FORM			
* GENDIGS -GENERATE DIGITS AND DEC. PT.			
* MTEMPIM -MOVE CHARACTERS FROM TEMP TO IMAGE			

```

*
*      THE RESULT IS FORMED IN A TEMP AREA BECAUSE LENGTH IS NOT
*      DETERMINATE PRIOR TO CONVERSION-WIDTH CHECK IS MADE BY MTEMPIM
*      PRIOR TO MOVE TO IMAGE
*

```

```

LOCAL      NOTNEG,EFORM,MOVEIM
CREALIM    BAL,R11  CREALBIN      CONVERT TO INTEGER & DEC POS
           LI,R5    BA(NAMETEMP)
           CI,R7    X'72'         NEG SIGN CHECK
           BNE      NOTNEG
           BAL,R11  SETCHAR      SET -
NOTNEG     CI,R12   -4           CHECK OUTPUT FORM
           BLE      EFORM        E FORM-TOO LOW
           CW,R12   DIGITS
           BG       EFORM        E FORM-TOO HIGH
           BAL,R4   GENDIGS      ORDINARY FORM-GENERATE DIGITS
MOVEIM     AI,R5    -BA(NAMETEMP) GET LENGTH
           LW,R1    R5
MTEMPIM    LW,R7    R1
           BAL,R11  CHEKWID      CHECK IF NAME FITS ON IMAGE
MTEMPIM2   LI,R8    BA(NAMETEMP) SET SOURCE ADDRESS FOR MBS
           LI,R9    BA(IMAGE)
           AW,R9    R3           SET TARGET ADDRESS FOR MBS
           STB,R1   R9           SET COUNT FOR MBS
           MBS,R8   0
           AW,R3    R1
           B        *R13
EFORM      BAL,R4   GENDIGSE      E FORM GENERATE  X.XXX---
           BAL,R4   GENEXP        GENERATE  EXX OR E-XX
           B        MOVEIM
TEXTC2I    LB,R1    *R8          LINK=R13  GET BYTE COUNT OF TEXTC ITEM.
           SLS,R8   2
           AI,R8    1            = BA(TEXT STRING)
           B        MTEMPIM2+1    MOVE THE STRING INTO IMAGE BUFFER.
PAGE

```

```

*
*      CREALARR-CONVERT REAL ARRAY DATUM TO PRINT FORM IN IMAGE BUFFER
*

```

```

*      R13=LINK
*      R2,R4,R5,R7,R11,R12 USED INTERNALLY
*      R3 USED AND UPDATED (IMAGE BUFFER POINTER)
*
*      ROUTINES USED:CHEKWID      CHECK IF FIELD FITS IN IMAGE
*                      CREALBIN    CONVERT TO INTEGER-DEC. EXP. FORM
*                      SETCHAR     SET CHARACTER
*                      GENDIGS,GENDIGSE  GENERATE DIGITS
*                      GENEXP      GENERATE  E-FIELD
*
*      FIELD WIDTH IS DIGITS+7    BLANKS AND ZEROS INSERTED AS
*                                  FORM INDICATES
*

```

```

LOCAL      MAXBLNK,BLOOP,ENDBLNKS,BLOOP2,EFORMARR,ZLOOPE
CREALARR   LOCAL    NODOT
           LW,R7    DIGITS        FIELD WIDTH IS FIXED
           AI,R7    7             AT DIGITS+7
           BAL,R11  CHEKWID      CHECK VS IMAGE POSITION
           BAL,R11  CREALBIN     CONVERT TO INTEGER-DEC. EXP. FORM
           LI,R5    BA(IMAGE)    GET ADDRESS
           AW,R5    R3           TO STASH BYTES
           LW,R4    R7           SAVE SIGN OR BLANK
           MTW,0    EFLAG        CHECK IF E-FORMAT
           BNEZ     EFORMARR     YES
           LW,R2    DMINP5       NO-CHECK FOR BLANK
           CI,R12   1            INSERTION REQUIREMENT
           BLE      MAXBLNK      MAX BLANK INSERTION
           AI,R2    1
           SW,R2    R12          VARIED BLANK INSERTION
MAXBLNK    LI,R11  BLOOP
           LI,R7    BLANK
BLOOP      BDR,R2  SETCHAR
           LW,R7    R4           - SIGN OR BLANK
           BAL,R11  SETCHAR
           BAL,R4   GENDIGS      GENERATE DIGITS AND DEC. PT.
ENDBLNKS   LW,R2   DIGITS
           SW,R2    R5           CALCULATE BLANK PAD TO RIGHT
           AW,R2    R3           FOR FIELD WIDTH DIGITS+7
           AI,R2    BA(IMAGE)+8
           LI,R7    BLANK       GEN. BLANKS

```

BLOOP2	LI, R11	BLOOP2	
	BDR, R2	SETCHAR	
	AW, R3	DIGITS	SET NEW IMAGE POS FOR R3
	AI, R3	7	
	B	*R13	EXIT
EFORMARR	LI, R7	BLANK	LEADING BLANK
	BAL, R11	SETCHAR	
	LW, R7	R4	SIGN OR BLANK
	BAL, R11	SETCHAR	
	BAL, R4	GENDIGSE	DIGITS E FORMAT
	AI, R12	0	CHECK IF DECIMAL NEEDED
	BNEZ	NODOT	NO
	LI, R8	X'4B'	YES-INSERT IT
	STB, R8	0, R5	
	AI, R5	1	
NODOT	LW, R2	DIGITS	
	AI, R2	BA(IMAGE)+4	CALCULATE 0'S PAD TO RIGHT
	AW, R2	R3	BEFORE GENERATING E-FIELD
	SW, R2	R5	
	LI, R7	X'F0'	GEN. 0'S
	LI, R11	ZLOOPE	
ZLOOPE	BDR, R2	SETCHAR	
	LI, R4	ENDBLNKS	SET EXIT
	B	GENEXP	GENERATE E-FIELD
	PAGE		

*

* SINGOUT AND MIXEDOUT--EXECUTION OUTPUT ROUTINES

*

* MIXEDOUT IS CALLED AT THE END OF EXECUTION OF A STATEMENT
* IF ONE OR MORE DATA BLOCKS ARE TO BE PRINTED

*

* SINGOUT IS CALLED FOR QUAD OUTPUT OF A SINGLE DATA BLOCK

*

* ENTRY CONDITIONS:

* MIXEDOUT-R13=# OF BLOCKS TO BE OUTPUT
* R1 = POINTER TO LIST OF DATA BLOCK POINTERS(-1)
* SINGOUT-R4 CONTAINS DATA BLOCK POINTER

*

* REGISTER USEAGE: R1-R14 SAVED AND RESTORED

*

* R14=LINK NORMAL RETURN TO CALL+2
* ERROR RETURN TO CALL+1 (NOT USED)

*

*

*

	LOCAL	NXTBLKOT, BLOCKOUT, GETSIZE, BYTDBASE, NOTDWB, NEXTROW
	LOCAL	ROWLOOP, DECRSIZE, PANELER, DUMPLAST
	LOCAL	QTEXT, LOGICSET, VECTOR, REALVECT, INTVECT1, INTVECT
	LOCAL	REALVSP, INTVSP
	LOCAL	LOGTEXT
MIXEDOUT	STW, R13	BLOCKNUM
	STW, R1	PTRPTR
	LCI	14
	STM, R1	SAVE114
		R1 TO R14
NXTBLKOT	MTW, 1	PTRPTR
	LW, R4	*PTRPTR
	B	UPDATE POINTER TO DBPTR
		GET NEXT DBPTR
		BLOCKOUT
SINGOUT	LCI	14
		SAVE
	STM, R1	SAVE114
		R1 TO R14
	LI, R3	0
		SET IMAGE OFFSET 0
	STW, R3	BLOCKNUM
		FLAG SINGLE BLOCK ENTRY
BLOCKOUT	LI, R15	%
		FOR ERROR INFO IF SYSTERR
	LB, R2	*R4
		GET DATA TYPE IN R2
	BEZ	SYSTERR
		INVALID DATA TYPE
	CI, R2	5
	BG	EMPTY
		INVALID DATA TYPE-TREAT AS EMPTY
	BE	INDEX-SEQUENCE SPECIAL CASE
	LI, R7	1
	LB, R6	*R4, R7
		GET RANK IN R6
	BEZ	SCALAR
	CI, R6	1
	BE	VECTOR
	STW, R6	RANKARR
		ARRAY-SAVE RANK
	AI, R4	1
	STW, R4	LBASEM1
		SAVE BASE-1 OF LENGTH VALUES
	LW, R9	*LBASEM1, R6
	STW, R9	ROWSIZE
		SAVE LAST DIMENSION-ROW SIZE
	LI, R9	1

GETSIZE	MW, R9	*LBASEM1, R6	SIZE=PRODUCT OF LENGTHS
	BEZ	EMPTY	
	BDR, R6	GETSIZE	
	STW, R9	DSIZE	SET NUMBER OF UNITS IN BLOCK
	AW, R4	RANKARR	R4 POINTS TO LAST LENGTH WORD
	AI, R4	1	NOW TO FIRST DATA WORD
QTEXT	CI, R2	2	CHECK TYPE FOR TEXT
	BL	LOGICSET	
	BE	BYTDBASE	YES
	CI, R2	4	CHECK IF SHIFT TO DW BOUND NEEDED
	BNE	NOTDWB	NO
	AI, R4	1	YES-INSURE DW BOUND
	AND, R4	FFFFFFFE	
	B	NOTDWB	
LOGICSET	LI, R11	32	SET INITIAL COUNT
	STW, R11	BITCOUNT	FOR LOGIC VALUE OUTPUT
	B	NOTDWB	
BYTDBASE	SLS, R4	2	SHIFT R4 TO BYTE ADDRESSING
NOTDWB	STW, R4	DBASE	SAVE ADDRESS OF DATA
	AI, R6	0	CHECK FOR VECTOR (TEXT OR LOGIC)
	BNEZ	CVARRTBL-1, R2	YES-SHORTCUT
	CI, R2	3	NO-CHECK FOR INTEGER OR REAL ARRAY
	BGE	GETRANGE-3, R2	YES-GET RANGE OF VALUES(ABSOLUTE)
SETCONV	LW, R9	CVARRTBL-1, R2	NO
	STW, R9	CONVRTR	TO FETCH & CONVERSION ROUTINES
	LW, R9	BLOCKNUM	CHECK IF NOT 1ST DATA BLOCK
	BLEZ	NEXTROW	FIRST-NO PROBLEM
	BAL, R12	DUMPLING	LATER-GENERATE RESIDUAL IMAGE
NEXTROW	LW, R10	ROWSIZE	SET LOOP
	LI, R13	DECRSIZE	SET EXIT FROM CONVERT SUBROUTINES
ROWLOOP	EXU	CONVRTR	PROCESS DKTUM
DECRSIZE	MTW, -1	DSIZE	DECREMENT TOTAL COUNT
	BEZ	LASTROW	QUIT IF DONE
	BDR, R10	ROWLOOP	LOOP ON ROWSIZE
LINEOUT	BAL, R12	DUMPLING	DUMP LINE WITH CR, RESET R3=0
	LW, R6	RANKARR	SET RANK FOR LOOP
	AI, R6	-1	REDUCE TO RANK-1
	LI, R4	0	
	LW, R5	DSIZE	GET REMAINING DATA COUNT
	DW, R4	ROWSIZE	DIVIDE BY ROW SIZE
PANELER	DW, R4	*LBASEM1, R6	USE REMAINDER TEST TO DETERMINE
	AI, R4	0	PANEL SEPARATION
	BNEZ	NEXTROW	REMAINDER NOT=0, NO MORE BLANKS
	LI, R8	BLANKS	SET TO
	LI, R9	1	OUTPUT
	BAL, R7	WROUTWB	A SINGLE BLANK WITH CR
	BDR, R6	PANELER	LOOP (SHOULD NOT FALL THROUGH)
LASTROW	MTW, -1	BLOCKNUM	LAST ROW-CHECK IF LAST(OR ONLY) BLK
	BGZ	NXTBLKOT	NO
DUMPLAST	BAL, R12	DUMPLING	YES-OUTPUT WITH CR
OUTEXIT	LCI	14	RESTORE
	LM, R1	SAVE114	R1 TO R14
	AI, R14	1	INCREMENT FOR CALL+2 RETURN
	B	*R14	RETURN
EMPTY	MTW, -1	BLOCKNUM	CHECK IF MORE DATA BLOCKS
	BGZ	NXTBLKOT	YES-CONTINUE PROCESSING
	AI, R3	0	NO-CHECK IF IMAGE IS EMPTY
	BEZ	OUTEXIT	YES-EXIT
	B	DUMPLAST	NO
* VECTOR-SET UP TO PROCESS VECTOR FOR OUTPUT			
* VECTOR			
VECTOR	LI, R6	0	SET INDEX
	AI, R4	2	POINT TO LENGTH OF VECTOR
	LW, R10	*R4	GET LENGTH IN R10 FOR LOOP
	BEZ	EMPTY	
	AI, R4	1	POINT TO FIRST DATA WORD (UNLESS DW)
	CI, R2	3	CHECK TYPE
	BL	LOGTEXT	LOGIC OR TEXT
	BE	INTVECT1	INTEGER VECTOR-READY TO PROCESS
	AI, R4	1	REAL-SKIP TO DW BOUND
	STW, R4	DBASE	SAVE DATA BASE ADDRESS
REALVECT	LW, R8	BREAKFLG	CHECK FOR BREAK
	BNEZ	OUTEXIT	YES
	LD, R8	*DBASE, R6	NO-GET DATUM
	AI, R6	1	
	BAL, R13	CREALIM	CONVERT IT
	BDR, R10	REALVSP	GO SET SPACES (UNLESS DONE)

REALVSP	B	LASTROW	DONE!
	BAL, R11	STASHBL	
	BAL, R11	STASHBL	
	B	REALVECT	LOOP
INTVECT1	STW, R4	DBASE	SAVE BASE ADDRESS OF DATA
INTVECT	LW, R8	BREAKFLG	CHECK FOR BREAK
	BNEZ	OUTEXIT	YES
	LW, R8	*DBASE, R6	NO-GET DATUM
	AI, R6	1	
	BAL, R13	CINTIM	CONVERT IT
	BDR, R10	INTVSP	GO SET SPACES (UNLESS DONE)
	B	LASTROW	DONE!
INTVSP	BAL, R11	STASHBL	
	BAL, R11	STASHBL	
	B	INTVECT	LOOP
LOGTEXT	STW, R10	DSIZE	SAVE DATA SIZE
	STW, R10	ROWSIZE	ALSO ROW LENGTH
	STW, R6	RANKARR	FAKE RANK 0 ON VECTOR
	LI, R6	1	SET TO CATCH VECTOR
	B	QTEXT	REENTER ARRAY ROUTINE
*			
*	GETRANGE-GET MAGNITUDE RANGE OF INTEGER OR REAL ARRAY		
*			
	LOCAL	RNGLOOP, UPINDEX, EFORM, CHEKHI, INTRANGE, MAGLOOP	
GETRANGE	LOCAL	KIKINDX, DMXLOOP, SETDMAX	
	B	INTRANGE	INTEGER
	LD, R8	ZEROZERO	
	STD, R8	HIREAL	PRESET HIREAL
	LD, R8	MAXREAL	
	STD, R8	LOREAL	AND LOREAL
	LW, R10	DSIZE	
RNGLOOP	LAD, R8	*DBASE, R6	GET DATUM
	BNEZ	%+2	SKIP IF NOT ZERO
	LD, R8	FLONE	REPLACE ZERO BY ONE
	CD, R8	LOREAL	CHECKLO
	BGE	CHEKHI	
	STD, R8	LOREAL	UPDATE LOW
UPINDEX	AI, R6	1	
	BDR, R10	RNGLOOP	LOOP
	LD, R8	LOREAL	GET LOW VALUE
	BAL, R11	CREALBIN	FIND DECIMAL POSITION
	CI, R12	-3	CHECK IF LOW
	BL	EFORM	YES-E FORMAT INDICATED
	AI, R12	5	
	STW, R12	DMINP5	SET DMINP5-USED BY CREALARR ROUTINE
	LD, R8	HIREAL	GET HIGH VALUE
	BAL, R11	CREALBIN	CHECK DECIMAL POSITION
	CW, R12	DIGITS	VS DIGITS
	BG	EFORM	TOO-HIGH, EFORM
	CW, R12	DMINP5	VS DMINP5
	BGE	EFORM	RANGE TOO WIDE-EFORM
	LI, R11	0	RESET EFLAG
EFORM	STW, R11	EFLAG	
	LI, R2	4	RESTORE TO 'REAL' TYPE
	B	SETCONV	PROCEED-AT LAST-TO CONVERSION
CHEKHI	CD, R8	HIREAL	CHECK VS HIREAL
	BLE	UPINDEX	
	STD, R8	HIREAL	UPDATE
	B	UPINDEX	
*			
INTRANGE	STW, R6	DMAX	INTEGER ARRAY RANGE CHECK
	LW, R10	DSIZE	SET LOOP
MAGLOOP	LAW, R8	*DBASE, R6	GET DATUM
	CW, R8	DMAX	CHECK
	BLE	KIKINDX	
	STW, R8	DMAX	UPDATE DMAX
KIKINDX	AI, R6	1	
	BDR, R10	MAGLOOP	LOOP
	LI, R6	10	
	LW, R8	DMAX	GET HIGHEST VALUE
DMXLOOP	CW, R8	TENSTBL-1, R6	CHECK # OF DIGITS
	BGE	SETDMAX	
	BDR, R6	DMXLOOP	LOOP
SETDMAX	STW, R6	DMAX	SET # OF DIGITS
	B	SETCONV	PROCEED WITH OUTPUT CONVERSION
*			
*	INDSEQ-INDEX SEQUENCE OUTPUT CONVERTER		
*			


```

LOCAL      INDLOOP, FIRSTOUT
INDSEQ     AI, R4      2
           LW, R10    *R4      GET COUNT
           BEZ        EMPTY    MIGHTY SHORT INDEX SEQUENCE!
           AI, R4      1
           LW, R8      *R4      GET BASE INTEGER
           STW, R8     DBASE
           AI, R4      1
           LW, R9      *R4      GET ADDEND INTEGER
           STW, R9     DSIZE
           B          FIRSTOUT  OUTPUT FIRST ELEMENT AND LOOP
INDLOOP    BAL, R11    STASHBL   GENERATE
           BAL, R11    STASHBL   BLANK SPACERS
           LW, R8      BREAKFLG  BREAK TEST
           BNEZ       OUTEXIT    SCRAM!
FIRSTOUT   LW, R8      DBASE
           AW, R8      DSIZE      UPDATE ELEMENT
           STW, R8     DBASE
           BAL, R13    CINTIM     OUTPUT ELEMENT
           BDR, R10    INDLOOP    LOOP
           B          LASTROW    EXIT
*
* SCALAR-PROCESSOR FOR SCALAR OUTPUT
*
LOCAL      SCALTYPE, SCALTEXT, SCALOGIC, SETLOGIC, ONBIT
SCALAR     LI, R13     LASTROW    SET EXIT
           AI, R4      2          POINT TO DATUM
           LD, R8      *R4      GET DW (MAY ONLY NEED PART OF IT)
SCALTYPE    B          SCALTYPE, R2 VECTOR ON TYPE 1-4      @@@@@@@@
           B          SCALOGIC    1-LOGIC                  @
           B          SCALTEXT    2-TEXT                   @
           B          CINTIM      3-INTEGER                 @
           B          CREALIM     4-REAL                   @@@@@@@@
SCALTEXT    LB, R4     *R4      GET CHARACTER
           B          GENTEXT1
SCALOGIC    BLZ        ONBIT      CHECK IF 1-BIT
           LI, R8      X'F0'      NO
SETLOGIC    LI, R11    LASTROW
           B          STASHIM     STASH 1 OR 0 BYTE
ONBIT       LI, R8      X'F1'
           B          SETLOGIC
*
* LOGICOUT-OUTPUT FOR LOGIC VECTOR OR ROW OF LOGIC ARRAY
*
LOCAL      NEWORD, TESTBIT, ABIT, BITSHIFT, SBLNKS, LOGARRAY, ONEBIT
LOGICOUT    LW, R7     BITCOUNT  GET BIT POS IN WORD
NEWORD      LW, R9     *DBASE      GET WORD
TESTBIT     CW, R9     BITPOS-32, R7 CHECK BIT
           BANZ       ONEBIT      1
           LI, R8      X'F0'      0
ABIT        LW, R11    BREAKFLG   CHECK FOR BREAK
           BNEZ       OUTEXIT    YES-QUIT
           BAL, R11    STASHIM     NO-SET BIT
           BDR, R10    SBLNKS     LOOP ON ROWSIZE
           LW, R10     DSIZE       END OF ROW
           SW, R10     ROWSIZE     ADJUST DSIZE
           BEZ        LASTROW     DONE!
           STW, R10    DSIZE
           BDR, R7     BITSHIFT    UPDATE BIT POINTER
           MTW, 1      DBASE       (AND WORD IF NECESSARY)
           LI, R7      32
BITSHIFT    STW, R7     BITCOUNT
           B          LINEOUT
SBLNKS      BAL, R11    STASHBL    SET BLANK
           LW, R11    RANKARR     CHECK VECTOR VS ARRAY
           BNEZ       LOGARRAY   ARRAY
           BAL, R11    STASHBL    VECTOR-2 BLANKS BETWEEN VALUES
LOGARRAY    BDR, R7     TESTBIT
           MTW, 1      DBASE
           LI, R7      32
ONEBIT      B          NEWORD
           LI, R8      X'F1'
           B          ABIT        SET '1' BYTE
*
* TEXTOUT-PROCESS TEXT VECTOR OR ROW OF TEXT ARRAY
*
TEXTOUT     LW, R6      DBASE      GET DATA ADDRESS
           AWM, R10    DBASE      UPDATE FOR NEXT PASS

```

```

        BAL,R13  GENTEXTM          GENERATE ROW OF TEXT
        LW,R10   DSIZE
        SW,R10   ROWSIZE          UPDATE REMAINING DATA SIZE
        BEZ      LASTROW          QUIT IF LAST ROW
        LW,R13   BREAKFLG        CHECK FOR BREAK
        BNEZ     OUTEXIT          YES-EXIT
        STW,R10  DSIZE
        B        LINEOUT
*
* INTARRAY-PROCESS ELEMENT IN INTEGER ARRAY
*
INTARRAY LW,R8    BREAKFLG        CHECK BREAK
        BNEZ     OUTEXIT          YES-QUIT
        LW,R8    *DBASE          GET DATUM
        MTW,1    DBASE           KICK POSITION
        B        CINTARR         CONVERT AND OUTPUT
*
* REALARRY-PROCESS ELEMENT IN REAL ARRAY
*
REALARRY LW,R8    BREAKFLG        CHECK BREAK
        BNEZ     OUTEXIT          YES-QUIT
        LD,R8    *DBASE          GET DATUM
        MTW,2    DBASE           KICK POSITION
        B        CREALARR        CONVERT AND OUTPUT
*
* CVARRTBL-PROGRAM SWITCHES-SET IN CONVRTR
*
CVARRTBL B        LOGICOUT
        B        TEXTOUT
        B        INTARRAY
        B        REALARRY
        PAGE
*
* GENCHAR-GENERATE A CHARACTER OR MNEMONIC IN IMAGE
*
* R13=LINK
* ON ENTRY:R7 CONTAINS INTERNAL FORM OF CHARACTER
* R3 IS IMAGE OFFSET TO STASH CHARACTER
*
* INTERNALLY, R2,R7, AND R14 USED AND CHANGED
*
* ON EXIT: R3 IS IMAGE OFFSET FOR NEXT CHARACTER
*
* THIS ROUTINE DOES NOT PERFORM WIDTH CHECK
*
GENCHAR  LOCAL    GENLOOP,ONECHAR
        LB,R2     OUTMNEMT,R7    CHECK CHARACTER CLASS
        BEZ      ONECHAR        1 FOR 1
        CI,R2     4
        BGE      ONECHAR        OVSTK-LEAVE AS IS
        LB,R7     OUTRANST,R7    MNEMONIC-EXPAND IT (GET OFFSET)
        AW,R7     MNEMTN-1,R2    BA OF MNEMONIC
        AI,R2     1              LOOP SIZE
        LI,R14    X'43'          APL %
GENLOOP  STB,R14   IMAGE,R3
        AI,R3     1
        LB,R14    0,R7           MNEMONIC CHARACTER
        AI,R7     1
        BDR,R2    GENLOOP        LOOP
        B        *R13
ONECHAR  STB,R7     IMAGE,R3      SINGLE CHARACTER
        AI,R3     1
        B        *R13            EXIT
        PAGE
*
* CNAMEIM-CONVERT A NAME-MOVE TO IMAGE BUFFER
*
* ON ENTRY:R3 IS OFFSET PTR FOR IMAGE
* R6 POINTS AT 2ND BYTE OF NAME POINTER
* R5 IS NAME ENTRY OFFSET FROM SYMT
* R4 IS FLAG -1=STOPNAME 0=TRACENAME 1=NAME
* R13 IS LINK
*
* REGISTERS R1 TO R12-EXCEPT R6-ARE USED AND CHANGED
*
* GENNAME IS AN ALTERNATE ENTRY WHICH DOES NOT CHECK FOR STOP OR TRACE
* FOR THIS ENTRY, R8 IS THE NAME OFFSET TO SYMT
*

```

```

*
*   ON EXIT, R6 IS UNCHANGED. R3 POINTS TO NEW OFFSET IN IMAGE
*
*   MNEMONICS ARE EXPANDED IF ENCOUNTERED, AND THE NAME IS FORMED
*   IN 'NAMETEMP'
*   'CHEKWID' IS USED TO CHECK FOR WIDTH OVERRUN-IF SO, IMAGE IS
*   DUMPED PRIOR TO MOVING THE NAME FROM NAMETEMP TO IMAGE
*
      LOCAL      CNAM1, CNAM2, CNAM3, CNAM4, LONGNAME
      LOCAL      STUFNAM1, STUFNAM2, STUFNAM3, STUFNAM4
      LOCAL      CNAM5, CNAM6
GENNAME0  LI, R1      1
          STW, R1     SAVE212          SET FLAG-NO INDENT ON WIDTH OVERFLOW
          LI, R1      0
          B           CNAM2
GENNAME   LI, R1      0          SET NAMETEMP POINTER
          STW, R1     SAVE212          SET FLAG FOR INDENT ON WIDTH OVERFLO
          B           CNAM2          BYPASS STOP-TRACE CHECKS
CNAMEIM   LI, R1      0          SET NAMETEMP POINTER
          STW, R1     SAVE212          SET FLAG FOR INDENT ON WIDTH OVRFLOW
          LI, R2      'S'          PRESET STOP-NAME
          LW, R8       R5          SAVE SYMT OFFSET IN R8
          AI, R4       0          CHECK NAME-TYPE FLAG
          BGZ         CNAM2          PLAIN NAME
          BLZ         CNAM1          STOP
          LI, R2      'T'          TRACE
CNAM1     BAL, R11    STUFNAM4          SET S OR T
          LI, R2      X'48'
CNAM2     BAL, R11    STUFNAM1          SET DELTA OR %DLT
          AW, R8       SYMT          GET SYMBOL ADDRESS
          AI, R8       1
          LB, R10     *R8          CHECK NAME TYPE
          CI, R10     BLANK
          BL          LONGNAME
          LI, R10     4          SHORT NAME-SET LOOP TO 4
CNAM3     LI, R4      0          SET POINTER FOR NAMETEMP
CNAM4     LB, R2      *R8, R4          GET BYTE
          AI, R4       1
          BAL, R11    STUFNAM1          SET BYTE-OR MNEMONIC
          BDR, R10    CNAM4          LOOP
CNAM5     LW, R2      SAVE212          GET FLAG FOR WIDTH OVERFLOW ACTION
          B           CNAM6, R2
CNAM6     B           MTEMPIM          INDENT-USE MTEMPIM
          LW, R7       R1          NO INDENT
          AW, R7       R3          CHECK WIDTH
          CW, R7       WIDTH
          BL          MTEMPIM2          OK-MBS
          BAL, R12    DUMPLING          OVERFLOW-DUMP LINE FIRST
          B           MTEMPIM2          NOW-MBS
LONGNAME  SLS, R10    2
          LW, R8      *R8
          AND, R8     X1FFFF          ADDRESS MASK
          B           CNAM3
STUFNAM1  LB, R5      OUTMNEMT, R2          GET MNEMONIC FLAG
          BEZ         STUFNAM3          NOT MNEMONIC-CHECK FOR BLANK
          CI, R5       4          MAYBE
          BGE         STUFNAM4          NOT MNEMONIC OR BLANK
          CI, R1       251          CHECK FOR TOO DAMN BIG
          BG          CNAM5          YES-TRUNCATE
          LB, R7       OUTRANST, R2          MNEMONIC-GET TABLE OFFSET
          AW, R7       MNEMTN-1, R5          ADD TABLE ADDRESS
          LI, R2      X'43'          SET '%'
          AI, R5       1          SET MNEMONIC SIZE LOOP
STUFNAM2  STB, R2     NAMETEMP, R1          STASH % AND MNEMONIC NAME
          AI, R1       1
          LB, R2      0, R7
          AI, R7       1
          BDR, R5     STUFNAM2          LOOP
          B           *R11          EXIT
STUFNAM3  CI, R2      BLANK          BLANK CHECK
          BE          CNAM5          YES-END OF SCAN
STUFNAM4  STB, R2     NAMETEMP, R1          STORE NON-BLANK CHAR
          AI, R1       1
          CI, R1       255          CHECK FOR GROSS OVERFLOW
          BLE         *R11          NO-RETURN
          B           CNAM5          YES-TRUNCATE
          PAGE

```

*

```

* FUNLDISP-DISPLAY FUNCTION NAME AND LINE NO.
* (FUNLDIS% - ALT.ENTRY, R6 HAS CURRLINO & NAME PTR AND
* R8 HAS NAME PTR FOR FUNCTION.)
*
* R14=LINK
*
* ON ENTRY, R1 POINTS TO F STACK ENTRY
* NAME AND LINE # SET IN IMAGE STARTING AT POSITION 0
* ON EXIT R3 IS NEXT IMAGE BYTE OFFSET
* R6 CONTAINS CURRLINO AND FDEFPTR
* NO REGISTERS ARE SAVED-MOST ARE USED
*
FUNLDISP LW,R6      1,R1      GET CURRLINO AND FDEFPTR IN R6
          LW,R8      FNOFF,R6  GET NAME POINTER IN R8
FUNLDIS% LI,R3      0          START AT BYTE 0 IN IMAGE
          BAL,R13     GENNAME   CONVERT NAME & MOVE TO IMAGE
          LI,R4      X'B4'
          BAL,R13     GENTEXT1  LEFT BRACKET-MAY BE MNEMONIC
          LH,R8      R6
          SLS,R8      -1        GET FUNCTION LINE #
          BAL,R13     CINTIM    CONVERT
          LI,R4      X'B5'
          BAL,R13     GENTEXT1  RIGHT BRACKET(MAY BE MNEMONIC)
          BAL,R11     STASHBL   TWO BLANKS
          BAL,R11     STASHBL   FOR SPACING
          B          *R14      AND EXIT
          PAGE
*
* SHOWSTOP-GENERATE STOP DISPLAY-EXIT TO INPDIR
*
SHOWSTOP BAL,R14     FUNLDISP   DISPLAY FUNCTION NAME-LINE NUMBER
          BAL,R12     DUMPLING
          B          INPDIR      GO TO DIRECT INPUT
          PAGE
*
* SIDR-(HOW ABOUT THAT!)-ROUTINE TO WARN USER THAT SI DAMAGE WILL
* RESULT IF HE CLOSES THE CURRENTLY OPEN FUNCTION
*
* RESPONSE IS REQUESTED: GO MEAN3 PROCEED TO CLOSE
*
* R14=LINK RETURN IS VIA R14 WITH CONDITION CODES AS INDICATOR
* 'EQUAL' MEANS GO
*
SIDR      LW,R7      ON%OFF
          BEZ        *R14      OFF-LINE, ASSUME 'GO'
          AW,R7      BREAKFLG  (ON%OFF=1)
          BLEZ       *R14      BREAKFLG NEG.=HANGUP='GO'
          LI,R8      SIDAMSG
          BAL,R7      WRTEXTC   GENERATE MESSAGE
          LI,R8      0
          BAL,R11     RDAPL+1    READ RESPONSE
          LH,R7      INBUF
          CH,R7      GOMSG      CHECK VS 'GO'
          B          *R14
          SIDAMSG    TEXTC      'SI DAMAGE WILL RESULT:TYPE ''GO'' TO CLOSE'
          GOMSG      TEXT      'GO '
          PAGE
*
* SHOWFL-ROUTINE TO DISPLAY FUNCTION LINE OR HEADER
*
* ENTRY CONDITIONS:1. R14=LINK, MERGECOL IS SET NEGATIVE
* DISPLAY ONLY, RETURN VIA R14
* 2. MERGECOL=0 GENERATE LINE AS PROMPT-EXIT TO
* FIHANDLER
* 3. MERGECOL POSITIVE-GENERATE LINE AND BLANK PROMPT
* TO MERGECOL-1, EXIT TO EDITCTRL
*
* A. CURRCS=0 NO DECODESTRINGING REQUIRED
*
* B CURRCS=PTR TO CS OLOCK-DECODESTRINGING REQUIRED
*
SHOWFL    LOCAL      SAVNBLNK
          STW,R3      IMAGEPOS  SET NO.OF BYTES ALREADY IN IMAGE
          LW,R12     CURRCS     CHECK LINE VS HEADER
          BNEZ       DECODOPS   LINE-REQUIRES DECODESTRING
          LW,R12     MERGECOL   CHECK DISPLAY-APPEND-EDIT
          BEZ        INPF       APPEND
          BAL,R12     DUMPLING   DISPLAY OR EDIT

```


	BNEZ	SYSTERR	*NOT CODESTRING-SYSTEM ERROR
	LW, R6	CURRCS	
	LH, R8	*R6	OFFSET
	SLS, R6	2	BA(OFFSET)
	AI, R6	1	
	LI, R10	CSEND	DECODET OFFSET FOR EOS FLAG
	STB, R10	0, R6	SET LEFT-END STOP
	AW, R6	R8	END OF CS+1
	LB, R9	0, R6	GET BYTE TO RIGHT OF CODESTRING
	STB, R10	0, R6	SET RIGHT-END STOP
	STB, R9	R8	R8-BYTE 3=LEFT END-1, BYTE 0=RT END+1
	STW, R8	SAVENDS	
	LI, R11	-1	SET RIGHT-LEFT SCAN
	B	STEP1	MOVE LEFT 1 BYTE AND START
*			
*			
*			
*			
*			
*			
	LOCAL	PICKBYTE, SHIFTN, SHIFTV, STEP1, STEP3, STEP5, STEP9, STEP2	
	LOCAL	LFACT, BITS, LONGTEXT, LONGT1, LONGT2, ENDFLIP	
PICKBYTE	LB, R5	0, R6	GET BYTE FROM CODESTRING
	CI, R5	CSEND	CHECK RANGE
	BG	SYSTERR	!IMPOSSIBLE!
	LB, R4	DECODET, R5	DECODE BYTE
	CI, R4	X'11'	CHECK IF MULTI-BYTE ENTRY
	BG	STEP1	NO-SKIP IT
	BE	ENDFLIP	EOS-STOP FLIP-FLOP SCAN
	AI, R4	-2	CHECK TYPE OF SPECIAL CODESTRING
	BLEZ	STEP1	SINGLE CHARACTER-SKIP IT
	LW, R3	R6	SET POINTER
	AW, R3	R11	TO NEXT BYTE (MAY BE LENGTH)
	LB, R9	0, R3	R9=2ND BYTE
	AI, R4	-2	
	BLEZ	STEP2	2 CHARACTER SET-FLIP THEM
	AI, R4	-4	
	BLZ	STEP3	3 CHARACTERS
	BEZ	STEP5	5 CHARACTERS
	AI, R4	-2	
	BLZ	STEP9	9 CHARACTERS
	AW, R3	R11	SET TO 3RD BYTE
	CI, R4	6	
	BE	LONGTEXT	LENGTH IS 2 BYTES
	LW, R10	R9	SET LOOP COUNTER TO LENGTH
	BEZ	STEP2	EMPTY TEXT VECTOR OR NULL COMMENT
	LI, R12	STEP2	SET EXIT FROM BYTE-SHIFT LOOP
	EXU	LFACT, R4	GET BYTE COUNT FROM LENGTH
	B	SHIFTV	ENTER BYTE-SHIFT LOOP
SHIFTN	LI, R12	STEPN	
SHIFTV	LB, R7	0, R3	BYTE SHIFT
	STB, R7	0, R6	LOOP
	AW, R6	R11	
	AW, R3	R11	
	BDR, R10	SHIFTV	
	B	*R12	EXIT
STEP2	STB, R9	0, R6	STORE 2ND BYTE
	AW, R6	R11	
STEPN	STB, R5	0, R6	STORE KEY BYTE
STEP1	AW, R6	R11	MOVE TO NEXT BYTE
	B	PICKBYTE	
STEP3	LI, R10	2	NAME ITEM HW+KEY BYTE
	B	SHIFTN	
STEP5	LI, R10	4	INTEGER WORD SCALAR
	B	SHIFTN	
STEP9	LI, R10	8	REAL DW SCALAR
	B	SHIFTN	
*	LFACT-EXU	VECTOR	
LFACT	B	BITS	LOGIC VECTOR @@@@@@@@
	B	SHIFTV	BYTE-FIELD @
	B	SHIFTV	BYTE-FIELD @
	SLS, R10	2	WORD FIELD-COUNT=4*L @
	SLS, R10	3	DW FIELD-COUNT=8*L @
	B	SHIFTV	BYTE FIELD @@@@@@@@
*			
BITS	AI, R10	7	
	SLS, R10	-3	GET BYTE COUNT FROM BIT COUNT
	B	SHIFTV	

```

*
LONGTEXT LB,R4      0,R3      GET 3RD BYTE-PART OF L
          AW,R3      R11      MOVE TO 4TH BYTE
          CI,R11     -1       CHECK DIRECTION
          BE         LONGT1    RIGHT-LEFT
          XW,R9      R4       LEFT-RIGHT, SWITCH BYTES 2 & 3
LONGT1    LW,R10     R4       R4=L1
          SLS,R10    8        R9=L2
          AW,R10     R9       R10=LENGTH
          CI,R11     -1
          BNE        LONGT2    LEFT-RIGHT
          XW,R9      R4       RIGHT-LEFT, SWITCH BYTES 2 & 3
LONGT2    BAL,R12    SHIFTV    SHIFT L BITS
          STB,R4     0,R6     SET L1 OR L2
          AW,R6      R11
          B          STEP2    FINISH LAST 2 BYTES
*
ENDFLIP   CI,R11     -1       CHECK PHASE
          BNE        DECODEND  PHASE 3-
          LW,R11     SAVENDS
          STB,R11    0,R6     RESTORE 2ND BYTE OF OFFSET
          LI,R11     0
          STD,R11    IMAGES AND IMMERBYT PRESET FOR PHASE 2
          STW,R11    LASTSCAN
          LW,R3      CURRCS    GET ADDRESS OF CS
          SLS,R3     2        FORM BYTE ADDRESS
          MTW,0      MODE      CHECK MODE
          BNEZ       ERRDISPL  ERROR DISPLAY
          AI,R3      10000     FUNCTION EDIT-SET LARGE VALUE
          B          DECODOUT
ERRDISPL  AW,R3      OFFSET
*
* PHASE 2-SCAN LEFT-RIGHT AND GENERATE LINE DISPLAY
*
          LOCAL      PASSERR,SETBYTE,CHEKMNEM,RESLAST,HARDWAY
          LOCAL      TEXT,TEXT1,COMMENT,NOTEXT,REALV1,REALV,REAL1
          LOCAL      INTV1,INTEGERV,INTEGER1,INTEGERB,LOGICV,LOGICV1
          LOCAL      LOGICV2,YES,NXTBIT,NAME
DECODOUT  STW,R3     CSERRBYT  SET CS ERROR BA (OR HI VALUE)
          LW,R3      IMAGEPOS
SKIP      AI,R6      1
COMPBYTE  CW,R6      CSERRBYT  CHECK IF PAST ERROR POSITION
          BL         PASSERR   NO-OR NO ERROR INDICATED
          MTW,0      IMERRBYT  CHECK IF ERROR CARET ALREADY SET
          BNEZ       PASSERR   YES
          STW,R3     IMERRBYT  NO-SET ERROR CARET POSITION
PASSERR   LB,R5      0,R6      R5=CS VALUE
          LB,R4      DECODET,R5 R4=APL INTERNAL OR SPECIAL INDIC.
          CI,R4      X'11'     CHECK FOR SPECIAL
          BL         HARDWAY   SPECIAL
          BE         ENDCONV   EOS-QUIT DISPLAY
          CI,R5      9         SINGLE CHAR-CHECK DIGIT
          BG         CHEKMNEM  NO
          BAL,R11    QSTASHBL  YES-SET BLANK IF LASTSCAN INDICATES
SETBYTE   LW,R8      R4       R8=INTERNAL APL CHAR.
          LI,R11    SKIP      SET EXIT
          B          STASHIM   SET BYTE
*
CHEKMNEM  LB,R7      OUTMNEMT,R4 CHECK IF BYTE IS MNEMONIC FOR
          BEZ       RESLAST    OUTPUT DEVICE
          CI,R7      4
          BGE       RESLAST    NO
          BAL,R11    QSTASHBL  YES-SET BLANK IF LASTSCAN INDICATES
          BAL,R5     GENMNEM   GENERATE MNEMONIC
          B          SKIP
*
* RESLAST-RESET LASTSCAN=0 AND SET BYTE
*
RESLAST   LI,R8      0
          STW,R8     LASTSCAN
          B          SETBYTE
*
* HARDWAY-START OF PROCESSING FOR SPECIAL CS ITEMS
*
HARDWAY   CI,R4      3
          BL         SKIP      DUMMY,BOS
          BE         TEXT1
          AI,R6      1

```

	LB, R10	0, R6	GET 2ND BYTE
	CI, R4	X'B'	CHECK FOR TEXT VECTOR
	BL	NOTEXT	
	BE	TEXT	
	CI, R4	X'C'	CHECK FOR COMMENT
	BE	COMMENT	
	CI, R4	X'F'	CHECK FOR LINE-SCAN ERROR
	BE	LSCERR	YES
	BL	NOTEXT	
	AI, R6	1	GET L2
	LB, R11	0, R6	
	SLS, R10	8	
TEXT	AW, R10	R11	SET TEXT BYTE COUNT IN R10
	LI, R8	X'7D'	SET LEADING QUOTE
CSGNTEXT	BAL, R11	STASHIM	
	BAL, R13	GENTEXT	
	LI, R4	X'7D'	
	B	RESLAST	SET CLOSING QUOTE AND RESET LASTSCAN
*			
TEXT1	LI, R10	1	
	B	TEXT	
*			
COMMENT	AI, R10	1	KICK COUNT FOR COMMENT CHARACTER
	LI, R4	X'59'	COMMENT CHARACTER
	BAL, R13	GENTEXTC	
	B	SKIP	
*			
* LSCERR			LINE-SCAN ERROR-TEXT+ERROR POINTER
*			
LSCERR	AW, R6	R10	MOVE TO ERROR POINTER
	LB, R5	0, R6	GET IT
	SW, R6	R10	MOVE BACK
	LW, R4	R3	GET IMAGE POINTER
	AW, R4	R10	ADD LENGTH
	SW, R4	R5	SUBTRACT ERROR POINTER
	AI, R4	-1	1 MORE
	MTW, 0	MODE	CHECK IF DISPLAY
	BEZ	%+2	YES-DONT INDICATE ERROR
	STW, R4	IMERRBYT	SAVE IMAGE ERROR POSITION
	AI, R10	-1	DONT COUNT THE ERROR POINTER AS TEXT
	BAL, R13	GENTEXT	GENERATE TEXT
	AI, R6	1	SKIP THE ERROR POINTER
	B	SKIP	PROCEED (SHOULD HIT END NEXT)
*			
*			
NOTEXT	BAL, R11	QSTASHBL	INSERT BLANK IF NEEDED -SET LASTSCAN
	CI, R4	4	
	BE	INTEGERB	BYTE INTEGER
	CI, R4	8	
	BL	NAME	NAME, STOP-NAME, OR TRACE-NAME
	BE	INTEGER1	SCALAR INTEGER
	CI, R4	9	
	BE	REAL1	SCALAR REAL
	AI, R6	1	MOVE TO 3RD BYTE
	CI, R4	X'D'	
	BE	INTERGERV	INTEGER VECTOR
	BL	LOGICV	LOGIC VECTOR
	B	REALV	REAL VECTOR-LENGTH IN R10
REALV1	BAL, R11	STASHBL	SET INTERVENING BLANK
REALV	LW, R7	SET89	SET TO MOVE FROM ADDRESS IN R6
	MBS, R6	0	TO R8-R9
	BAL, R13	CREALIM	CONVERT AND STORE IN IMAGE
	BDR, R10	REALV1	
	B	COMPBYTE	
REAL1	LI, R10	1	SET COUNT TO 1
	B	REALV	
SET89	DATA	X'08000020'	MBS 8 BYTES TO R8-R9
*			
INTV1	BAL, R11	STASHBL	SET INTERVENING BLANK
INTERGERV	LW, R7	SET8	SET TO MOVE FROM CS TO R8
	MBS, R6	0	LOAD R8
	BAL, R13	CINTIM	CONVERT AND STASH
	BDR, R10	INTV1	LOOP
	B	COMPBYTE	EXIT
INTEGER1	LI, R10	1	INTEGER SCALAR
	B	INTERGERV	
INTEGERB	LW, R8	R10	1 BYTE INTEGER
	LI, R13	SKIP	SET EXIT


```

SET8      B      CINTIM      CONVERT AND STASH
*      DATA      X'04000020'      MBS 4 BYTES TO R8

LOGICV      LI,R5      8      SET BIT-IN-BYTE COUNTER
      LB,R9      0,R6      GET BYTE
LOGICV1      CW,R9      BITPOS-32,R5      CHECK BIT
      BANZ      YES      1
      LI,R8      X'F0'      0
LOGICV2      BAL,R11      STASHIM      STASH CHAR.
      BDR,R10      NXTBIT      LOOP ON TOTAL BIT COUNT
      B      SKIP      EXIT
YES      LI,R8      X'F1'      1
      B      LOGICV2
NXTBIT      BAL,R11      STASHBL      SET INTERVENING BLANK
      BDR,R5      LOGICV1      LOOP ON BIT-IN-BYTE
      AI,R6      1      WORK NEXT BYTE
      B      LOGICV
*
NAME      AI,R4      -6      SET R4 -1 FOR STOP,0 FOR TRACE
      AI,R6      1      +1 FOR ORDINARY NAME
      LB,R5      0,R6
      SLS,R10      8
      AW,R5      R10      R5=NAME POINTER ,R4=FLAG FOR S OR T
      LI,R13      SKIP
      B      CNAMEIM
*
*
* ENDCONV-END OF DECODESTRING CONVERSION FOR OUTPUT-PHASE2
*
ENDCONV      LI,R12      ENDCONV1      END OF CODESTRING
      LW,R6      MODE      CHECK MODE
      BGZ      DUMPLINE      NOT EDIT-MUST BE DISPLAY
      LW,R6      MERGECOL      CHECK DISPLAY-APPEND-EDIT
      BNEZ      DUMPLING      DISPLAY OR EDIT-USE CR
      BAL,R11      PRMTIMAG      APPEND-NO CR
ENDCONV1      LW,R6      CURRCS
      SLS,R6      2      SET R6 FOR LEFT-RIGHT SCAN OF CS
      AI,R6      1      (PHASE 3)
      LI,R11      1      SET DIRECTION LEFT-RIGHT
      B      STEP1      START PHASE 3 SCAN
*
* DECODEND-END OF PHASE 3
*
DECODEND      LB,R11      SAVENDS
      STB,R11      0,R6      RESTORE BYTE PAST END OF CS
      LW,R3      MODE      CHECK MODE
      BGZ      UNREF      NOT EDIT-DEREF CS BLOCK
      LW,R3      ENDIMAGE      FN EDIT
      AI,R3      1      GET CHAR COUNT
      B      DECODRTN
*
*
* DECODET-TABLE TO CONVERT CODESTRING TO CHARACTER IMAGE FOR OUTPUT
* THIS TABLE IS USED IN THE FOLLOWING PHASES:
* 1. RIGHT-LEFT SCAN TO FLIP KEY AND LENGTH BYTES TO
* THE LEFT ON MULTTI-BYTE ITEMS.
*
* 2. LEFT-RIGHT SCAN TO GENERATE IMAGE OF INTERNAL
* CHARACTERS-NOTE-THIS SCAN ALSO USES OUTTRANS TABLE
* TO SET MNEMONICS AS REQUIRED FOR TTY,PRINTER,ETC.
*
* 3. LEFT-RIGHT SCAN TO RESTORE CODESTRING TO EXECUTION
* FORM.
*
* ENTRIES ABOVE BLANK (X'40') ARE SINGLE CHARACTER CODESTRING
* ITEMS WHICH MAP TO SINGLE INTERNAL APL CHARACTERS
*
* ENTRIES BELOW BLANK INDICATE SPECIAL ACTION TO GET IMAGE
* CHARACTERS-NUMERIC TRANSLATION,NAME LOOK UP,ETC.
* CODESTRING BYTE IS KEY FOLLOWED BY INFORMATION BYTES
* ACTION VECTOR:      BYTES IN CODESTRING
* 0      DUMMY-SKIP ITEM      1
* 1      ERROR      1
* 2      BOS-SKIP ITEM      1
* 3      SCALAR-TEXT      2
* 4      SCALAR-INTEGER-BYTE      2
* 5      STOPNAME -SYMTAB PTR      3
* 6      TRACENAME-SYMTAB PTR      3

```

```

*          7 NAME- SYMTAB PTR          3
*          8 SCALAR-INTEGER            5
*          9 SCALAR-REAL                9
*          A VECTOR-BITS                (L+7)/8+2
*          B VECTOR-TEXT-SHORT          L+2
*          C COMMENT                    L+2
*          D VECTOR-INTEGERS             4L+2
*          E VECTOR-REALS                8L+2
*          F LINE-SCAN ERROR            L+2
*          10 VECTOR-TEXT-LONG           L+3 (L=2 BYTES)
*          11 END OF STATEMENT (TEMPORARY VALUE)
*
DECODET DATA X'F0F1F2F3' 0,1,2,3
DATA X'F4F5F6F7' 4,5,6,7
DATA X'F8F90408' 8,9,INTEGER-BYTE,INTEGER
DATA X'09030A0D' REAL,CHAR,LOGICS,INTEGERS
DATA X'0E0B100F' REALS,CHARS,MANY CHARS,LINESCAN ERR
DATA X'0C050607' COMMENT,STOPNAME,TRACENAME,NAME
DATA X'01B6B7B8' ERROR,QUAD0,QUAD1,QUAD2
DATA X'B9BABBBBC' QUAD3-QUAD6
DATA X'BDBEBF53' QUAD7,QUAD8,QUAD9,QUAD
DATA X'52025EB4' QQUAD,BOS,SEMI,LEFT BRACKET
DATA X'B54D5DFC' RIGHT BRACKET,LPAREN,RPAREN,BRANCH
DATA X'FD4B5A01' ASSIGNMENT,.,SMALL CIRCLE,ERROR
DATA X'0101FE6F' ERROR,ERROR,DOMINO,QUESTION MARK
DATA X'54636149' T-BAR,REDUCTION1,REDUCTION,IOTA
DATA X'6C6B6858' RHO,COMMA,ROTATE1,ROTATE
DATA X'B24E60FA' TRANSPOSE,+, -, MULTIPLY
DATA X'FB5C5756' DIVIDE,*,LOG,CIRCLE
DATA X'62444F41' MAX,MIN,ABS,BANG
DATA X'01015146' ERROR,ERROR,I-BEAM,GRADE UP
DATA X'735F0101' GRADE DOWN,NOT,ERROR,ERROR
DATA X'FE6F5463' DOMINO,QUESTION MARK,T-BAR,REDUCT1
DATA X'61496C6B' REDUCTION,IOTA,RHO,COMMA
DATA X'6858B24E' ROTATE1,ROTATE,TRANSPOSE,+
DATA X'60FAFB5C' -,MULTIPLY,DIVIDE,*
DATA X'57566244' LOG,CIRCLE,MAX,MIN
DATA X'4F414C74' ABS,BANG,LT,LE
DATA X'6E767B7E' GT,GE,NE,=
DATA X'70797778' AND,OR,NAND,NOR
DATA X'01014255' ERROR,ERROR,DECODE,TEE
DATA X'6A64B0B1' TAKE,DROP,EXPAND1,EXPAND
DATA X'45010171' EPSILON,ERROR,ERROR,DIERESIS
DATA X'726D437C' NEGSIGN,UNDERSCORE,%,ALPHA
DATA X'667F754A' OMEGA,DEL,LOCK,CPL
DATA X'67505B7A' CPR,CAP,CUP,:
DATA X'01010011' ERROR,ERROR,DUMMY,END-OF-STMT
*
END

```

TITLE	'APLUTSI-B00,10/16/73,DWG702985'	
SYSTEM	SIG7F	
SYSTEM	BPM	
* * REF'S:		
REF	@CLEAR	
REF	@@CLEAR	
REF	@@CONT	
REF	@@SAVE	
REF	APLSINON	APL'S SIGN-ON MESSAGE.
REF	BITPOS	TABLE OF WORDS WITH SINGLE 1-BIT
REF	CALOPNXT	CAL'S
REF	CALOPTF	IN
REF	CALOPTRM	OPEN)TERM FILE
REF	CALOPWS1	MODULE
REF	CALOPWS2	APLUTSC
REF	CALOPWS3	
REF	CALOPWS4	
REF	CALOP1ST	
REF	CALRDTF	
REF	CALRDTRM	
REF	CALRDWS	
REF	CALRDWSI	
REF	CALWRTF	
REF	CALWRWS	
REF	CLOSTHIS	CLOSE FILE
REF	CMDB	COMMAND BRANCH VECTOR
REF	CMDERR	COMMAND ERROR
REF	CMDEXIT	COMMAND EXIT
REF	CURRCS	CURRENT CODESTRING PTR.
REF	DIVZERO	EXEC MOD CHECK FOR SPEC. VALID DIV
REF	DREF	DEREFERENCER.
REF	DUMPLING	LINE OUTPUT ROUTINE
REF	ERBADCMD	BAD COMMAND EXIT
REF	ERDOMAIN	DOMAIN ERROR
REF	ERRFTFIO	FIO SUBSYSTEM ERROR HANDLER
REF	ERRFWS6	ERROR PROCESSOR
REF	FIOERR	FIO MONITOR ERROR
REF	FPTOPTRM	FPT'S
REF	FPTRDTRM	IN MODULE
REF	FPTRDWS	APLUTSC
REF	FPTWR2	FPT TO WRITE CONTROL TO 4013
REF	FREETBL	FREE SPACE TABLE
REF	FREETOTL	TOTAL FREE SPACE
REF	GARBCOLL	GARBAGE COLLECT ROUTINE
REF	GENTEXT	TEXT GENERATOR ROUTINE
REF	IDBUF	ID BUFFER
REF	IDFILBSY	ERROR I.D. FOR: FILE IN USE
REF	IDFILREF	BAD FILE REF
REF	IDWSNOTF	WS NOT FOUND
REF	IDIOERR	I/O ERR
REF	IDFILSPC	FILE SPACE TOO LOW
REF	J:ACCN	ACCOUNT #
REF	J:DELTAT	C
REF	J:IDELTAT	P
REF	J:PTIME	U
REF	J:UTIME	TIME
REF	J:JIT	
REF	LOADCONT	LOAD CONTINUE WS ON START-UP.
REF	M:UC	USERS CONSOLE DCB-USED FOR TABS
REF	NCMDS	# OF SYSTEM COMMANDS
REF	NEWBOUND	NEW DYNAMIC BOUND
REF	OPBREAK	BREAK IN OPERATOR
REF	QUIETFLG	QUIET FLAG
REF	RANKARR	ARRAY RANK
REF	RDWRLOOP	READ-WRITE LOOP
REF	RESACCT	RESET ACCT TO USERS ACCT.
REF	SAVEOPEN	OPEN FOR SAVE
REF	SAVE312	SAVE TEMP
REF	TABPNTR	TAB POINTER
REF	TENSTBL	POWERS OF 10 (FIXED POINT)
REF	UTSIOFF	FULL EXIT.
REF	UTSIOFFH	OFF-HOLD EXIT
REF	XSEGBRK	XSEG BREAK FLAG,SET=OPBREAK
REF	XWLOCGLB	EXCHANGES LOCALS & GLOBALS.
REF	X1FFFF	
* * REF'S TO MODULE ORIGINS		

*

REF	ACQCONS@	APLC1
REF	CONSTS@	APLC2
REF	CONTEXT@	APLC3
REF	CS@	APLC4
REF	INTRINS@	APLC5
REF	WMAQ@	APLC6
REF	LIB@	APLC7
REF	OPR@	APLC8
REF	MIX@	APLC9
REF	INDX@	APLC10
REF	EVAL@	APLC11
REF	APLINP@	APLC12
REF	APLOUT@	APLC13
REF	APLUTSI@	APLC14
REF	FUNDEF@	APLC15
REF	WSCOPY@	APLC16
REF	CMD@	APLC17
REF	APLUTSC@	APLC18
REF	CODEXEQ@	APLC19
REF	APLFIO@	APLC20
REF	APLFMT@	APLC21
REF	MAT@	APLC22
REF	RCMD@	APLC23
REF	SCMD@	APLC24
REF	ERROR@	APLC25
REF	GRAF@	APLC26
REF	UTSG@	APLC27

*

DEF'S:

DEF	APLSIZ	TERMINAL INPUT RECORD MAX SIZE
DEF	APLUTSI@	START OF PROCEDURE
DEF	APLUTSI@	START OF CONTEXT
DEF	APLVERSN	VERSION NAME
DEF	BCDTIME	TIME AND DATE
DEF	BLINBUF	BLIND INPUT BUFFER.
DEF	BLINDOUT	BLIND OUTPUT ROUTINE
DEF	BREAKFLG	USER BREAK FLAG
DEF	BSPFLAG	BACKSPACE VALIDITY FLAG
DEF	CALTABS	CAL TO SET TABS
DEF	CALTIME	GET TIME
DEF	CALWROUT	APL WRITE OUTPUT RECORD
DEF	CHKTERM	CHECK IF 4013 TERMINAL
DEF	CLEARCOM	RELEASE COMMON TO CLEAR STATE
DEF	CLEARDYN	RELEASE DYNAMIC TO CLEAR STATE
DEF	CLEARMEM	CLEAR WS MEMORY TO MINIMUM
DEF	CLEARsiz	SIZE OF 'CLEAR' TABLE
DEF	CLOSV	CLOSE AND SAVE
DEF	CLOSR	CLOSE AND RELEASE
DEF	CMNDTYPE	COMMAND TYPE
DEF	COMPRMST	TABLE OF COMMON PARAMETERS
DEF	CONSTBUF	BLK TO HOLD NOS. ACQUIRED FOR A CONSTANT & CERTAIN CMND PARAMS.
DEF	CONSTDT	DBLWD TEMP FOR CONSTANT ACQ.
DEF	CONSTDTX	EXTRA DBLWD TEMP FOR CONSTANT ACQ.
DEF	CORAVAIL	AVAIL WS CORE (TOTAL)
DEF	CORLEFT	VARIABLE CORE REMAINING
DEF	CPUTIME	GET CPU TIME
DEF	CURRKEYT	CURRENT KEY IN FILE
DEF	CXDTEMP	DBLWD TEMP FOR CODESTRING XEQ.
DEF	DATE	GET DATE
DEF	DELAYER	DELAY SPECIFIED TIME
DEF	DIGITS	
DEF	DIGRANGE	DIGITS RANGE
DEF	DWIDTH	DEFAULT PRINT WIDTH
DEF	DYNBOUND	DYNAMIC BOUND
DEF	ERBADWS	BAD WS ERROR EXIT
DEF	ERLIBREF	BAD LIB REF
DEF	ERRFTF	ERROR ROUTINE, F:TF
DEF	ERRFWS	ERROR ROUTINE, F:WS
DEF	ERRORCHR	ERROR SIGNAL CHARACTER
DEF	EVALTMPs	TEMPS FOR EVAL(APLC11)
DEF	EWROUTWB	ERROR CALL TO: WROUTWB
DEF	EWRTXTC	WRTEXTC
DEF	FDBLOCK	FUNCTION DEFN MODE STORAGE BLOCK.
DEF	FDTEMPs	FUNCTION DEFN MODE TEMP AREA.
DEF	FFFFFE00	PAGE BOUND MASK
DEF	FIOABNT	FIO INABN MODE FLAG
DEF	FIOACCC	FIO ACCOUNT CONTROL

U14-0004

DEF	FIOACCT	FIO ACCOUNT	
DEF	FIOBUF	FIO BUFFER ADDRESS	
DEF	FIODCB	FIO DCB ADDRESS	
DEF	FIODCBNO	FIO DCB NO. IN USE	
DEF	FIODCBT	TABLE OF FIO DCB ADDRESSES	
DEF	FIOKEY	KEY IN USE FOR FIO	
DEF	FIOMODE	FIO I/O MODE	
DEF	FIONAME	FIO NAME	
DEF	FIOPASC	FIO PASSWORD CONTROL	
DEF	FIOPASS	FIO PASSWORD	
DEF	FIOREADC	FIO READ CONTROL	
DEF	FIOSIZ	FIO REC SIZE	
DEF	FIOSNC	SERIAL NUMBER CONTROL WORD	U14-0008
DEF	FIOSN	SERIAL NO. ENTRY FOR PRIVATE PACKS	U14-0009
DEF	FIOTIE	TABLE OF FIO TIE VALUES	U14-0010
DEF	FIOWRITC	FIO WRITE CONTROL	
DEF	FMTMPS	TEMPS FOR DELTAFMT	
DEF	FPARAMS	FILE PARAMETERS	
DEF	FPTOP1ST	OPEN FIRST FILE OF AN ACCOUNT	
DEF	FPTOPFIO	FPT FOR FIO OPEN	
DEF	FPTOPNXT	OPEN NEXT FILE OF AN ACCOUNT	
DEF	FPTOPTF	FPT OPEN TEMP FILE	
DEF	FPTOPWS	FPT TO OPEN A FILE	
DEF	FPTRD1ST	READ 1ST RECORD OF A FILE (FOR ID)	
DEF	FPTXCOFF	RESET EXIT CONTROL	
DEF	FQTABL	TABLE OF DCB ADDRESSES	
DEF	FRSTKEYT	FIRST KEY IN FILE	
DEF	FUZZBIT	FUZZ PARAMETERS	
DEF	FUZZCNT		
DEF	FUZZLIMS		
DEF	FUZZMASK		
DEF	GDIGITS	DIGITS SETTING FOR GRAPHICS	
DEF	GETCOM	GET PAGE OF COMMON	
DEF	GETDYN	GET PAGES OF DYNAMIC	
DEF	GETTIME	GET CURRENT TIME AND DATE	
DEF	GRAFBUF	GRAPHICS BUFFER	U14-0012
DEF	HICOMMON	HIGHEST VARIABLE CORE ADDRESS	
DEF	HOLDFLG	OFF-HOLD FLAG	
DEF	INBUF	INPUT-OUTPUT BUFFER	
DEF	INDXTMPS	TEMPS FOR INDX(APLC10)	U14-0014
DEF	INPBLIND		
DEF	INTRANS		
DEF	IOERCODE	MONITOR I/O ERROR CODE-SUBCODE	
DEF	KEY1	KEY VALUE 1	
DEF	LASTKEYT	LAST KEY IN FILE	
DEF	LIBTEMPS	TEMPS FOR LIB(APLC07)	U14-0016
DEF	LOADCASE	FLAG FOR LOAD CATEGORY	
DEF	LOCOMMON	LOWEST COMMON ADDRESS	
DEF	LODACCT	LOAD ACCOUNT #	
DEF	LODYN	LOWEST VARIABLE CORE ADDRESS	
DEF	LOGONTIM	LOGON TIME (TO APL)	
DEF	MIXTEMPS	TEMPS FOR MIX(APLC09)	U14-0018
DEF	MODE	EXECUTION MODE	
DEF	NAMEGRN1	GRANULE #-1ST NAME RECORD	
DEF	NBIO	# OF BLIND I/O CHANNELS	
DEF	NCOMPRMS	SIZE OF COMMON PARAMETER TABLE	
DEF	NUMFILES	# OF DCBS FOR FIO SUBSYSTEM	
DEF	NUMUSERS	GET # OF USERS	
DEF	ON%OFF	ONLIN-OFFLINE MODE FLAG	
DEF	OPRTEMPS	TEMPS FOR OPR(APLC08)	U14-0020
DEF	OP1STACC	ACCOUNT IN FPTOP1ST	
DEF	OP1STACT	ACCT CONTROL IN FPTOP1ST	
DEF	OPENFN	NAME POINTER TO FORCED CLOSE FN	
DEF	OPER	DBLWD TEMP FOR OPERATOR INFO AND SAVING OTHER INFO.	
*			
*	DEF	OPTEMPS	TEMP AREA FOR OPERATOR WORK AND ALSO FOR SAVING OTHER INFO.
DEF	OPWSACT	WS ACCOUNT	
DEF	OPWSACTC	OPEN FPT-ACT CONTROL	
DEF	OPWSMODE	OPEN FPT MODE (I/O)	
DEF	OPWSNAME	WS NAME FOR 'OPEN'	
DEF	OPWSPAS	OPEN FPT PASSWORD	
DEF	OPWSPASC	OPEN FPT PASSWORD CONTROL	
DEF	OPWSWRTC	WRITE ACCOUNT CONTROL	
DEF	ORGADJ	ORIGIN PARAMETERS	
DEF	ORIGIN		
DEF	OUTMNEMT	OUTPUT MNEMONIC-OVST FLAG TABLE	
DEF	OUTRANST	OUTPUT TRANSLATION TABLE	

DEF	OUTSIZ	SIZE OF OUTPUT RECORD(5)
DEF	OUTMAXSZ	TERMINAL OUTPUT BLOCKING SIZE
DEF	OVERTIME	GET OVERHEAD TIME.
DEF	PCOPYFLG	PROTECTED COPY FLAG
DEF	PRINTFNM	PRINT FILE NAME
DEF	QTABVLID	CHECK TAB VALIDITY
DEF	QUITFLAG	FLAG TO AVOID REPEAT 'CONTINUE'
*		ATTEMPTS IN OFF-LINE MODE.
DEF	RANDOM	RANDOM SEED
DEF	RDAPL	READ APL INPUT RECORD
DEF	RELCOM	RELEASE COMMON TO TOPOSTAK
DEF	RELEASER	RELEASE MEMORY AS INDICATED
DEF	RELDYN	RELEASE PAGES OF DYNAMIC
DEF	RETURN14	SHARED TEMP
DEF	SAVE14	SAVE REGISTER-OR TEMP
DEF	SETDIGIT	SET DIGITS-GET OLD VALUE
DEF	SETWIDTH	SET WIDTH-GET OLD VALUE
DEF	SICTRL	STATE-INDICATOR CONTROL SETTING.
DEF	SNEWTABS	SET NEW TABS
DEF	STATEPTR	SI STATE POINTER
DEF	STKLIMIT	EXECUTION STACK LIMIT
DEF	SYSTERR	SYSTEM ERROR
DEF	TABSET	SET NEW TABS-GET OLD VALUES
DEF	TABVALS	TAB SETTING BUFFER IN FPT
DEF	TELEXIT	RETURN TO TEL
DEF	TERMKEY	TERMINAL RECORD KEY
DEF	TERMSIZ	TERMINAL RECORD SIZE
DEF	TERMTYPE	TERMINAL TYPE NO.
DEF	TIMODAY	GET TIME OF DAY
DEF	TOPOSTAK	TOP OF EXECUTION STACK
DEF	TRAP	
DEF	USERACCT	USER ACCOUNT
DEF	WIDRANGE	
DEF	WIDTH	
DEF	WINDOW	A ONE-PAGE SCRATCH AREA.
DEF	WROUT	APL WRITE OUTPUT RECORD
DEF	WROUTWB	
DEF	WRTEXTC	
DEF	WSIDNAME	WORKSPACE NAME
DEF	WSIDPASS	WSID PASSWORD
DEF	WSOFFSET	WS LOAD OFFSET
DEF	XFF	MASK
DEF	XSEGBASE	BASE FOR EXECUTION SEGMENTS.
*	STANDARD EQU'S:	
*	REGISTERS	
R0	EQU	0
R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6
R7	EQU	7
R8	EQU	8
R9	EQU	9
R10	EQU	10
R11	EQU	11
R12	EQU	12
R13	EQU	13
R14	EQU	14
R15	EQU	15
*	MEMORY CONTROL PARAMETERS	
MINALLOW	EQU	2 MINIMUM 'DYNAMIC' PAGES
MININCR	EQU	1 MINIMUM INCREMENT
MINANNEX	EQU	1 MINIMUM ANNEX
*		
*	BLIND AND FILE I/O CHANNEL ALLOWANCES	
*		
NBIO	EQU	3]0. OF BLIND I/O CHANNELS+1
NUMFILES	EQU	8
*		
*		
*	USERACCT-EQUATED TO J:ACCN (MONITOR NAME IN JIT TABLE)	
*		
USERACCT	EQU	J:ACCN
*		
*	MODULE DESCRIPTION:	
*		

```

*      THIS MODULE CONSISTS OF TWO CSECTS AND A NUMBER OF DSECTS
*      THE MODULE CONTAINS ALL DIRECT MONITOR INTERFACES AND
*      HOPEFULLY ALL MONITOR DEPENDANT CODE.
*
*      THE DSECTS ARE USED TO DEFINE THE DCBS USED BY APL.
*
*      A CSECT 0 INCLUDES THE FIXED CONTEXT FOR APL, PRIMARILY
*      NON-EXECUTABLE TABLES, FLAGS BUFFERS, ETC. THIS CSECT ALSO
*      INCLUDES THE EXECUTABLE INITIALIZATION ROUTINE, WHICH IS
*      VOLATILE, BEING UNNEEDED FOR ANY LATER OPERATIONS.
*      THE CSECT ALSO INCLUDES FPTS WHICH MUST BE VARIED DURING
*      OPERATIONS.
*
*      A CSECT 1 INCLUDES DIRECT MONITOR INTERFACES-CAL'S AND FIXED
*      FPT'S. CALS AND FPTS ARE CLEARLY LABELLED FOR LOCATION VIA
*      CONCORDANCE. THIS CSECT ALSO INCLUDES THOSE PORTIONS OF I/O
*      AND MEMORY MANAGEMENT ROUTINES WHICH ARE MONITOR SENSITIVE
*
*      PAGE
*
*      DATA CONTROL BLOCK GENERATION
*
*      THIS SECTION INCLUDES THE PROCEDURES USED TO GENERATE
*      DCBS FOR APL. NO EXECUTABLE CODE IS GENERATED. THE DCBS
*      ARE LOADED BY UTS IN A SPECIAL AREA. ACCESS IS READ ONLY
*      TO USER, READ-WRITE TO MONITOR.
*
F:APL      DSECT      1              APL INPUT DCB
F:APL      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (ERR, ERRFAPL), (ABN, ABNFAPL), (DEVICE, 'CR')
              ORG, 1      BA(F:APL)+3
              DATA, 1    3              SET DCB TO DEVICE MODE
              ORG      F:APL+26
              DATA      X'02000002'      SET ACCT CONTROL OFF
ABNFAPL    EQU      ERRFAPL      USE SAME ROUTINE FOR ABN & ERR
ERRFAPL    EQU      ERRRDAPL      ** MAY BE CHANGED
*
F:OUT      DSECT      1              APL OUTPUT DCB
F:OUT      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (OUTSN), (ERR, ERRFOUT), (ABN, ABNFOUT), (DEVICE, 'LP')
              ORG, 1      BA(F:OUT)+3
              DATA, 1    3              SET DCB TO DEVICE MODE
              ORG      F:OUT+26
              DATA      X'02000002'      SET ACCT CONTROL OFF
*
F:WS       DSECT      1              APL WORKSPACE I/O DCB
F:WS       M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFWS), (ABN, ABNFWS), ;
              (WRITE, 'NONE', 'APLAPLAP')
ABNFWS     EQU      ERRFWS
*
F:TF       DSECT      1              APL TEMP-FILE DCB
F:TF       M:DCB      (FILE), (PASS), (SYNON), (ERR, ERRFTF), (ABN, ABNFTF)
ABNFTF     EQU      ERRFTF
*
*      BLIND I/O DCBS FOLLOW:
*      NBIO DETERMINES THE NUMBER (0 TO 9) OF DCBS
*      TO BE CREATED FOR BLIND I/O
*      DEFAULT IS NBIO=2
*      NBIO ALSO CONTROLS PROCEDURE CODE FOR BLIND I/O
*
*      GOTO, NBIO  NOBIO, BI01, BI02, BI03, BI04, BI05, BI06, BI07, BI08, BI09
*
BI09       SPACE      1
ERRFQ9     EQU      ERRRDQ
ABNFQ9     EQU      ERRRDQ
F:Q9       DSECT      1
F:Q9       M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ9), (ABN, ABNFQ9)
BI08       SPACE      1
ERRFQ8     EQU      ERRRDQ
ABNFQ8     EQU      ERRRDQ
F:Q8       DSECT      1
F:Q8       M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ8), (ABN, ABNFQ8)
BI07       SPACE      1
ERRFQ7     EQU      ERRRDQ
ABNFQ7     EQU      ERRRDQ

```

```

F:Q7      DSECT      1
F:Q7      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ7), (ABN, ABNFQ7)

BI06      SPACE      1
ERRFQ6    EQU        ERRRDQ
ABNFQ6    EQU        ERRRDQ
F:Q6      DSECT      1
F:Q6      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ6), (ABN, ABNFQ6)

BI05      SPACE      1
ERRFQ5    EQU        ERRRDQ
ABNFQ5    EQU        ERRRDQ
F:Q5      DSECT      1
F:Q5      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ5), (ABN, ABNFQ5)

BI04      SPACE      1
ERRFQ4    EQU        ERRRDQ
ABNFQ4    EQU        ERRRDQ
F:Q4      DSECT      1
F:Q4      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ4), (ABN, ABNFQ4)

BI03      SPACE      1
ERRFQ3    EQU        ERRRDQ
ABNFQ3    EQU        ERRRDQ
F:Q3      DSECT      1
F:Q3      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ3), (ABN, ABNFQ3)

BI02      SPACE      1
ERRFQ2    EQU        ERRRDQ
ABNFQ2    EQU        ERRRDQ
F:Q2      DSECT      1
F:Q2      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ2), (ABN, ABNFQ2)

BI01      SPACE      1
ERRFQ1    EQU        ERRRDQ
ABNFQ1    EQU        ERRRDQ
F:Q1      DSECT      1
F:Q1      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (ERR, ERRFQ1), (ABN, ABNFQ1)

NOBIO     RES        0
*
*      FILE I/O DCBS FOLLOW:
*      NUMFILES DETERMINES THE NUMBER (0 TO 8) OF DCBS
*      TO BE CREATED FOR FILE I/O SYSTEM
*      DEFAULT IS NUMFILES=3
*      NUMFILES ALSO CONTROLS PROCEDURE CODE FOR FILE I/O
*
      GOTO, NUMFILES+1  NOFIO, FI01, FI02, FI03, FI04, FI05, FI06, FI07, FI08
*
FI08      SPACE      1
F:F8      DSECT      1
F:F8      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;
              (ERR, ERRFF), (ABN, ABNFF)

FI07      SPACE      1
F:F7      DSECT      1
F:F7      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;
              (ERR, ERRFF), (ABN, ABNFF)

FI06      SPACE      1
F:F6      DSECT      1
F:F6      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;
              (ERR, ERRFF), (ABN, ABNFF)

FI05      SPACE      1
F:F5      DSECT      1
F:F5      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;
              (ERR, ERRFF), (ABN, ABNFF)

FI04      SPACE      1
F:F4      DSECT      1
F:F4      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;
              (ERR, ERRFF), (ABN, ABNFF)

FI03      SPACE      1
F:F3      DSECT      1
F:F3      M:DCB      (FILE, 'NAME11CHARS', 'ACCT8CHR'), (PASS), (KEYM, 3), ;
              (INSN), (OUTSN), (READ, 'NONE'), (WRITE, 'NONE'), ;

```



```

        (ERR,ERRFF),(ABN,ABNFF)
FI02    SPACE    1
F:F2    DSECT    1
F:F2    M:DCB    (FILE,'NAME11CHARS','ACCT8CHR'),(PASS),(KEYM,3),;
        (INSN),(OUTSN),(READ,'NONE'),(WRITE,'NONE'),;
        (ERR,ERRFF),(ABN,ABNFF)

FI01    SPACE    1
F:F1    DSECT    1
F:F1    M:DCB    (FILE,'NAME11CHARS','ACCT8CHR'),(PASS),(KEYM,3),;
        (INSN),(OUTSN),(READ,'NONE'),(WRITE,'NONE'),;
        (ERR,ERRFF),(ABN,ABNFF)

NOFIO    RES      0
*
        PAGE
APLUTSI0 CSECT    0
*
*   START-INITIALIZATION ROUTINE-VOLATILE
*   THIS ROUTINE IS ORG'D IN A BUFFER AREA AND IS AVAILABLE
*   ONLY AT INITIALIZATION TIME.
*
*   THE FUNCTIONS PERFORMED ARE THOSE ASSOCIATED WITH INITIAL
*   SETUP FOR APL. IN SUMMARY:
*       ESTABLISH ON-LINE VS OFFLINE MODE-SET UP DCB'S
*       AND TERMINAL TYPE ACCORDINGLY
*
*       SET BREAK AND TRAP CONTROL AND MODES
*
*       ESTABLISH RANGE OF VARIABLE MEMORY AVAILABLE AND
*       GET INITIAL PAGES OF MEMORY
*
*       SET TFILE NAME-UNIQUE TO USER
*
*       GET 'CONTINUE' OR 'CLEAR' WORKSPACE
*
START    LOCAL    OFFLINE,TEMPNAME,TERMKEY4
          STW,R0    LSPD          SAVE ADDRESS OF STACK POINTER DW
          LD,R2     *LSPD
          STD,R2    ISPD          SAVE INITIAL VALUE OF SPD
          BAL,R14   RESACCT      *BREF* RESET ACCT CONTROL-SET USERS ACCT.
          LFI       0           RESET FLOATING TRAPS
CALINT    CAL1,8    FPTINT       SET BREAK ROUTINE ADDRESS
CALXCON   CAL1,8    FPTXCON      SET EXIT CONTROL ADDRESS
CALTRAP   CAL1,8    FPTTRAP      SET TRAP ROUTINE ADDRESS
CALMCAC   CAL1,8    FPTMCAC      CHANGE ACTIVATION CHARACTER SET
          LW,R2     J:JIT        GET ON-OFF MODE BIT AND UNIQUE
          BGEZ      OFFLINE     JOB ID
          MTW,1     ON%OFF       SET MODE ONLINE
CALAPLUC   CAL1,1   FPTAPLUC     SET F:APL TO USERS CONSOLE
CALOUTUC   CAL1,1   FPTOUTUC     SET F:OUT TO USERS CONSOLE
          LI,R1     100
          STW,R1    APLSIZ       SET INPUT RECORD SIZE-ONLINE MODE
          LI,R8     64           (MUST BE MULTIPLE OF 4)
          STW,R8    OUTMAXSZ     SET MAX SIZE OF F:OUT RECORD
CALQTERM   CAL1,8    FPTQTERM    GET TERMINAL TYPE AND COC MODES
          LI,R2     8
          LI,R1     3
QTERM1    LB,R3     R8,R1        SAVE COC MODES FOR
          STH,R3    FPTRSTRM,R2  RESTORING ON EXIT
          AI,R2     -2
          BDR,R1    QTERM1       LOOP
CALSTRMB   CAL1,8    FPTSTRMB    SET BS EDIT OFF
          LB,R8     R8           CHECK TERMINAL TYPE ONLY
          CI,R8     6
          BE        CALSTERM     APL-2741-EBCD
          CI,R8     10
          BE        CALSTERM     APL-2741-SELECTRIC
          LI,R5     2
          CI,R8     4
          BE        CALSTRM2     NONAPL-2741-EBCD
          CI,R8     8
          BE        CALSTRM2     NONAPL-2741-SELECTRIC
          LI,R5     3
          LI,R6     72
          STW,R6    DWIDTH       DEFAULT WIDTH=72 FOR TTY
          B         SETTERM
CALSTERM   CAL1,8    FPTSETRM    RESET TAB SIM,TAB REL, AND
*                                     SPACE INSERTION
          B         TEMPNAME

```

CALSTRM2	CAL1, 8	FPTSETRM		
	B	SETTERM	SET TERMINAL TYPE	
OFFLINE	LI, R1	400		
	STW, R1	APLSIZ	SET INPUT RECORD SIZE-OFFLINE MODE	
	LI, R1	256	OFF-LINE, USE FULL IMAGE RECORDS	
	STW, R1	OUTMAXSZ	SET MAX SIZE OF F:OUT RECORD	
	LI, R1	X'15'	SET END-OF.-RECORD VALUE TO CR FOR	
	STW, R1	EORVAL	OFF-LINE MODE(NO 'LOOK-AHEAD')	
	LI, R1	15	MASK	
	AND, R1	F:APL	CHECK F:APL	
	CI, R1	3	IF DEVICE, ASSUME CARD READER	U14-0022
	BNE	CALPAGE	NOT DEVICE	U14-0023
	LI, R1	X'FFFF'	CHECK IF CR	U14-0024
	AND, R1	F:APL+1		U14-0025
	CI, R1	X'C3D9'	'CR'	
	BNE	CALPAGE	NO	
	LI, R8	THROWAPL	BUFFER TO THROW OUT !APL CARD	
	LI, R9	0	WHICH IS LEFT FOR FIRST READ	
*			FOR SHARED PROCESSOR WITH SOURCE	
*			FROM THE M:C DEVICE (CR)	
CALAPLCR	CAL1, 1	FPTRDAPL	READ THE !APL CARD AND FORGET IT	
CALPAGE	CAL1, 1	FPTPAGE	PAGE EJECT	
	LI, R5	4		
SETTERM	STW, R5	TERMTYPE	SET TERMINAL TYPE	
	AWM, R5	TERMKEY		
CALOPTRI	CAL1, 1	FPTOPTRM	OPEN TO READ TERMINAL TRANS. REC.	
CALRDTRI	CAL1, 1	FPTRDTRM	READ TRANSLATE TABLES	
	LI, R5	F:TF		
	BAL, R6	CLOSV	CLOSE AND SAVE F:TF	
TEMPNAME	LI, R1	3		
	STH, R2	TFNAME, R1	SET UNIQUE JOB ID IN TEMP FILE NAME	
CALGDDL	CAL1, 8	FPTGDDL	GET MEMORY LIMITS	
	STW, R8	LODYN	LOWEST ADDRESS OF DYNAMIC STORAGE	
	CI, R10	128		
	BLE	WSSIZE	ALLOW UP TO 128 PAGES	
	LI, R10	128		
WSSIZE	SLS, R10	9		
	STW, R10	CORAVAIL	NO. OF WORDS OF CORE AVAILABLE	
	AI, R8	-1		
	AW, R8	CORAVAIL		
	STW, R8	HICOMMON	HIGHEST ADDRESS OF COMMON	
	AI, R8	-511		
	STW, R8	LOCOMMON	PRESET LOCOMMON-1 PAGE	
	LW, R13	R8		
CALGCPI	CAL1, 8	FPTGVP	GET COMMON PAGE AS VIRTUAL PAGE	
	LI, R1	MINALLOW	SET MINIMUM DYNAMIC ALLOWANCE	
	LW, R13	LODYN		
CALGDPI	CAL1, 8	FPTGVP	GET INITIAL DYNAMIC PAGES AS	
	AI, R13	512	VIRTUAL PAGES	
	BDR, R1	CALGDPI		
	STW, R13	HIDYN	SAVE IT	
	LW, R13	LOCOMMON	COMPUTE	
	SW, R13	HIDYN	CORE	
	STW, R13	CORLEFT	LEFT	
	BAL, R14	TIMODAY	GET TIME ADL CALLED	
	STW, R11	LOGONTIM	SAVE IT	
	LI, R8	APLSINON		
	BAL, R7	WRTEXTC	SIGN-ON MESSAGE	
	LW, R11	J:DELTAT		
	SW, R11	J:IDELTAT		
	AW, R11	J:PTIME		
	AW, R11	J:UTIME		
	STW, R11	INITCPU	SAVE INITIAL CPU USAGE.	
	LW, R11	J:PTIME+1		
	AW, R11	J:UTIME+1		
	STW, R11	INITOVH	SAVE INITIAL OVERHEAD USAGE.	
	LI, R1	4	INOUT MODE	
	LI, R5	F:WS	(FOR CLOSV)	
	LI, R6	BCLEAR	'CLEAR' IF NO VALID AUTO-CONTINUE	
	STW, R1	OPWSMODE		
CALOPWS5	CAL1, 1	FPTOPWS	TRY TO OPEN 'CONTINUE'	
	LW, R10	KEY1		
	STW, R10	FWSKEY		
	LI, R10	2048	512 WORD RECORD	
	LW, R11	LODYN	BUFFER	
CALRDAC1	CAL1, 1	FPTRDWS	READ 1ST RECORD	
	LW, R4	LODYN		
	LW, R9	6, R4		

```

        CW,R9      APLVERSN      CHECK IF WS
        BNE        CLOSV        NO-)CLEAR
        LW,R9      12,R4        CHECK IF SAVED BY 'CONTINUE'
        AI,R9      NCMD5+CMDB-@@CONT
        BNE        CLOSV        NO-)CLEAR
        LI,R9      @@SAVE-NCMD5-CMDB YES,DAMN IT
        STW,R9     12,R4        SWITCH TO 'SAVED'
CALWRSC1 CAL1,1    FPTWRWS1     WRITE REC 1-INDICATE NOT AUTOSAVE
        BAL,R6     CLOSV
        LI,R6      0           SET SO SAVED MSG WILL BE DISPLAYED.
        STW,R6     QUIETFLG
        B          LOADCONT     DO A LOAD OF THE CONTINUE WS.
BCLEAR  LI,R6      @@CLEAR-NCMD5-CMDB
        STW,R6     CMNDTYPE     SET 'CLEAR' COMMAND TYPE
        B          @CLEAR       GO TO CLEAR ROUTINE
ERFWSCS0 CI,R8      CALOPWS5+1   CHECK IF OPEN FOR AUTO-CONTINUE
        BE         BCLEAR       YES-)(CLEAR
        CI,R8      CALRDAC1+1    CHECK IF ID-READ FOR AUTO CONTINUE
        BE         CLOSV        YES
        B          IOERR        NO-MYSTERY ERROR
        LOCAL

*
*   FPT'S USED AT INITIALIZATION ONLY
*
*   OPEN F:APL TO 'UC'
FPTAPLUC GEN,8,7,17 X'14',0,F:APL DCB ADDRESS
        DATA      X'00040000'    DEVICE FLAG
        DATA      X'0000E4C3'    'UC'
*
*   OPEN F:OUT TO 'UC'
FPTOUTUC GEN,8,7,17 X'14',0,F:OUT DCB ADDRESS
        DATA      X'00040000'    DEVICE
        DATA      X'0000E4C3'    'UC'
*
*   PAGE EJECT ON F:OUT (OFF-LINE MODE)
FPTPAGE GEN,8,7,17 X'04',0,F:OUT
*
*   GET RANGE OF MEMORY AVAILABLE
FPTGDDL DATA      X'1B000000'
*
*   FREE ALL PAGES-INITIALIZATION ONLY
FPTFP GEN,8,7,17 X'09',0,128
*
*   INTERRUPT CONTROL FPT
FPTINT GEN,8,7,17 X'0E',0,BREAK ADDRESS OF BREAK ROUTINE
*
*   TRAP CONTROL FPT-IGNORE TRAPS FOR FIXED POINT AND DECIMAL ARITH
FPTTRAP GEN,8,7,17 X'14',0,TRAP ADDRESS OF TRAP ROUTINE
        DATA      X'003C8003'    TRAP CONTROL BITS
*
*   FPTMCAC-CHANGE ACTIVATION CHARACTER SET
FPTMCAC DATA      X'06820000'    CHANGE ACTIVATION TO EOT ONLY
*
*   FPTWRWS1-REWRITE RE"ORD 1 OF 'CONTINUE' TO INDICATE 'SAVED'
FPTWRWS1 GEN,8,7,17 X'11',0,F:WS DCB ADDRESS
        DATA      X'FC000050'    P'S, WAIT, AND ONEMKEY
        DATA      ERRFWS         ERR
        DATA      ERRFWS         ABN
        GEN,1,31 1,LODYN         BUF
        DATA      2048          SIZ
        DATA      FWSKEY        KEY
        DATA      0             BTD
FWSKEY EQU RETURN14            TEMP FOR KEY ADDR
*
*   FPTSETRM-SETS COC MODES AS INDICATED
FPTSETRM DATA      X'06200000'
        DATA      X'E0000000'    P1,2,3

```

```

DATA      8      RESET TAB SIMULATION
DATA      X'20'   RESET SPACE INSERTION
DATA      X'A0'   RESET BS EDIT AND TAB REL.
FPTSTRMB  DATA   X'06200000'
DATA      X'20000000'
DATA      X'20'   TURN BS EDIT OFF
*
*  FPTQTERM-CHECK TERMINAL TYPE
*
FPTQTERM DATA   X'06400000'
*
*  FPTXCON-SET EXIT CONTROL ADDRESS
*
FPTXCON DATA    X'19000000'+HANGUP   EXIT CONTROL ADDRESS
THROWAPL RES     0      THROW-AWAY BUFFER FOR !APL CARD
PAGE
ORG           START
*
*  OVERLAY THE START AREA WITH INBUF, THE APL INPUT-OUTPUT BUFFER
*  NOTE THAT INITIALIZATION PROCESSES MUST NOT USE INBUF
*
WINDOW     BOUND   8
SPACE 3    RES     512      A ONE-PAGE SCRATCH AREA.
TLOC       SET     0      *****
TEMP       CNAME   1      *
DTEMP      CNAME   2      *  PROC TO ALLOCATE N WORDS OR *
PROC                               *  DOUBLEWORDS IN THE WINDOW AREA. *
DO          NAME=2      *  EXAMPLE FORMS: *
TLOC       SET     ((TLOC+1)/2)*2    *  A TEMP 1 ONE WORD. *
FIN                               *  B DTEMP 3 THREE DBLWDS. *
LF          EQU     WINDOW+TLOC      *
TLOC       SET     TLOC+AF(1)*NAME   *
PEND                               *****
SPACE 3
INBUF      DTEMP    65      INPUT(F:APL)-OUTPUT(F:OUT) BUFFER.
BLINBUF    EQU      INBUF    BLIND INPUT BUFFER OF 128 WORDS.
CONSTBUF   DTEMP    66      BLK FOR CONST ACQ & CMND PARAMS.
XSEGBASE   EQU      CONSTBUF  BASE FOR EXECUTION SEGMENTS.
FMTMPS     EQU      CONSTBUF  TEMPS FOR DELTAFMT
FPARAMS    EQU      CONSTBUF+40 TEMP FOR FILE PARAMETERS
CONSTDTX   DTEMP    1      FOR CONST ACQ.
CONSTDTX   DTEMP    1      FOR CONST ACQ.
CXDTMP     DTEMP    1      FOR CODESTRING EXECUTION.
OPTMPS     TEMP     50      TEMP BLK FOR OPERATOR & INDEXING
*                               EXECUTION. CONTAINS, FOR EXAMPLE,
*                               RSTYPE, RSRANK, AND RSSIZE.
*                               SO, MAY BE USED BY INTRINSIC FUNCS.
*
OPER       DTEMP    1      TEMP & COMM DBLWD FOR CODESTRING &
*                               OPERATOR EXECUTION.
*
FDTEMPS    DTEMP    19      FUNCTION DEFN MODE TEMP AREA.
GRAFBUF    EQU      FDTEMPS  SHARE FDTEMPS WITH GRAPHICS BUFFER U14-0029
FDBLOCK    TEMP     47      FUNCTION DEFN MODE STORAGE BLOCK.
*                               U14-0031
*  NOTE--THE FOLLOWING TEMPS MUST BE DW ALIGNED: LIBTEMPS U14-0032
*                               OPRTEMPS U14-0033
*                               MIXTEMPS U14-0034
*                               INDXTMPS U14-0035
*                               EVALTMPS U14-0036
*                               U14-0037
LIBTEMPS   EQU      FDBLOCK   SHARE FDBLOCK U14-0038
OPRTEMPS   EQU      FDBLOCK+10 WITH LIB AND OPR TEMPS U14-0039
MIXTEMPS   DTEMP    14      TEMPS FOR MIX U14-0040
INDXTMPS   DTEMP    20      INDX
EVALTMPS   DTEMP    5      EVAL U14-0042
SPACE 3
WINDOWSZ   SET      TLOC      DISPLAY AMOUNT OF WINDOW USED U14-0044
ERROR,X'F',TLOC>512 'TOO MUCH FOR WINDOW TO HOLD'
PAGE
*
*  TERMINAL TRANSLATION TABLES-OVERLAID BY )TERM COMMAND
*
PAGE
*
*  INPUT TRANSLATION TABLE
*  THIS VERSION IS FOR ADL STANDARD TERMINAL (2741)
*  ALTERNATE VERSIONS ARE LOADED FOR OTHER TERMINAL TYPES
*

```



```

*      DATA      0,0,0,X'00000400'      F0-FF
*
OUTRANST DATA X'00010203',X'04050607',X'08090A0B',X'0C0D0E0F' 00-0F
DATA X'10111213',X'14151617',X'18191A1B',X'1C1D1E1F' 10-1F
DATA X'20212223',X'24252627',X'28292A2B',X'2C2D2E2F' 20-2F
DATA X'30313233',X'34353637',X'38393A3B',X'3C3D3E3F' 30-3F
DATA,1 X'40',17,X'42',22 40-43
DATA,1 X'44',X'45',11,12 44-47
DATA X'48494A4B',X'4C4D4E4F' 48-4F
DATA,1 X'50',3,28,X'53' 50-53
DATA,1 33+2*NBI0,X'55',X'56',36+2*NBI0 54-57
DATA,1 18,24,X'5A',X'5B' 58-5B
DATA X'5C5D5E5F' 5C-5F
DATA,1 X'60',X'61',X'62',51+2*NBI0 60-63
DATA X'64656667' 64-67
DATA,1 37+2*NBI0,X'69',X'6A',X'6B' 68-6B
DATA X'6C6D6E6F' 6C-6F
DATA,1 X'70',X'71',X'72',21 70-73
DATA,1 X'74',50+2*NBI0,X'76',45+2*NBI0 74-77
DATA,1 49+2*NBI0,X'79',X'7A',X'7B' 78-7B
DATA X'7C7D7E7F' 7C-7F
DATA X'80C1C2C3',X'C4C5C6C7',X'C8C98A8B',X'8C8D8E8F' 80-8F
DATA X'90D1D2D3',X'D4D5D6D7',X'D8D99A9B',X'9C9D9E9F' 90-9F
DATA X'A0A1E2E3',X'E4E5E6E7',X'E8E9AAAB',X'ACADAEAF' A0-AF
DATA,1 55+2*NBI0,X'B1',39+2*NBI0,X'B3' B0-B3
DATA,1 X'B4',X'B5' B4-B5
OL1 WHILE OL1<NBI0
DATA,1 28+NBI0+OL1 B6-XX
FIN
OL2 WHILE OL2<10-OL1
DATA,1 X'B5'+OL1+OL2 XX-BF
FIN
DATA X'C0C1C2C3',X'C4C5C6C7',X'C8C9CACB',X'CCCDCECF' C0-CF
DATA X'D0D1D2D3',X'D4D5D6D7',X'D8D9DADB',X'DCDDDEDF' D0-DF
DATA X'E0E1E2E3',X'E4E5E6E7',X'E8E9EAEB',X'ECDEEEEF' E0-EF
DATA X'F0F1F2F3',X'F4F5F6F7',X'F8F9FAFB' F0-FB
DATA,1 X'FC',X'FD',29+2*NBI0,X'FF' FC-FF
PAGE
*
*   BSPFLAG-BACKSPACE VALIDITY FLAG-TERMINAL ATTRIBUTE
*
BSPFLAG DATA 1 YES-FOR 2741
*
*   ERRORCHR      TERMINAL ATTRIBUTE
*
*   BYTE 0 IS ERROR INDICATOR
*   BYTES 1 AND 2 ARE EVALUATED INPUT PROMPT
*   BYTE 3 IS FLAG(IF NON-ZERO) FOR QQUAD PROMPT
ERRORCHR DATA X'70537A00' CARET-QUAD-COLON-0
*
ENDTERM RES 0 END OF TERMINAL TRANSLATION AREA
TERMSIZ EQU BA(ENDTERM)-BA(INTRANS) TERMINAL TRANS. TBL. SIZE
PAGE
*
*   THE FOLLOWING AREA INCLUDES FPT'S WITH VARIABLE ELEMENTS
*
*
*   FPTTABS-FPT TO SET TABS
*
FPTTABS GEN,8,7,17 X'28',0,F:OUT DCB ADDRESS
GEN,1,31 1,0
TABVALS DATA 0,0,0,0,0 BYTES FOR TAB VALUES
*
*   FPTOPTF-OPEN TEMP FILE IN SCRATCH MODE
*
FPTOPTF GEN,8,7,17 X'14',0,F:TF DCB ADDRESS
DATA X'07400001' P6-P8,P10,F12
DATA 1 ORG-CONSEL (P6)
DATA 1 ACC-SEQUENTIAL (P7)
DATA 8 MODE (OUTIN)
DATA 1 REL
DATA X'01000202' NAME CONTROL 14-00001
TFNAME TEXTC 'APLTF00'
DATA X'02000000',X'03010000' RESET ACCT AND PASSWORD 14-00003
NAMEGRN1 DATA 0 GRANULE # FOR 1ST NAME RECORD
BOUND 8 @@@@@@@@
OPWSFLR1 DATA 0 ** SPACE FILLER @

```

```

*
* FPTOPWS-FPT TO OPEN F:WS FOR )SAVE,)LOAD,)COPY, ETC.
*
* THIS FPT IS IN CSECT 0 BECAUSE OF VARIABLE PARAMETERS USED
*
FPTOPWS GEN,8,7,17 X'14',0,F:WS DCB ADDRESS @
DATA X'07400001' P6,P7,P8,P10,F12 @
OPWSORG DATA 2 ORG (P6) @
OPWSACC DATA 2 ACC (P7) @
OPWSMODE DATA 0 MODE (P8) @
DATA 2 SAVE (P10) @
DATA X'01000303' NAME-3 WORDS RESERVED-3 WORDS USED @
OPWSNAME TEXTC 'CONTINUE' WS NAME-3WORDS @
OPWSPASC DATA X'03000003' PASSWORD CONTROL-DEFAULT OFF @
OPWSPAS DATA 0,0 ** DW BOUND **PASSWORD- @
OPWSFLR2 DATA 0 ** SPACE FILLER ** @
OPWSACTC DATA X'02000002' LACOUNT CONTROL-DEFAULT OFF
OPWSACT DATA 0,0 ** DW BOUND
OPWSWRTC DATA X'06010404' WRITE ACCOUNT CONTROL
TEXT 'NONE APLAPLAP'

*
* FPTOP1ST-USED TO INITIALIZE SEARCH OF AN ACCOUNT
*
BOUND 8 @@@
FPTOP1ST GEN,8,7,17 X'14',4,F:TF DCB ADDRESS AND 'TEST' OPEN
DATA X'07600401' P6,7,8,10,11 F2,12
DATA 1 ORG (P6)
DATA 1 ACC (P7)
DATA 1 MODE(P8)
DATA 2 SAVE(P10)
DATA FPARAMS (P11) FILE PARAMETER ADDRESS
DATA X'01000101' NAME CONTROL
DATA 0 EMPTY NAME
OP1STACC DATA X'02010002' ACCT CONTROL
OP1STACT DATA 0,0 ** DW BOUND EMPTY ACCOUNT

*
* FPTOPFIO-OPEN F:FN IN INDICATED MODE TO INDICATED FILE
*
* MUST START AT ODD LOCATION FOR DW BOUND CONTROL @@@@@@@@
BOUND 8 @
FIODCB DATA F:F1 ** DEFAULT TO 1ST FIODCB U14-0046
FPTOPFIO GEN,1,7,7,17 1,X'14',0,FIODCB DCB ADDRESS @@@@@@@@
DATA X'07400001' P6,P7,P8,P10,F12
DATA 2 ORG (P6)
DATA 2 ACC (P7)
FIOMODE DATA 0 MODE (P8)
DATA 2 SAVE (P10)
DATA X'01000303' NAME CONTROL
FIONAME DATA 0,0,0 NAME-3 WORDS
FIOACCC DATA X'02000003' ACCOUNT CONTROL
FIOACCT DATA 0,0 ACCOUNT @@@@@@@@
FIOKEY DATA 0 ** FILLER FOR DW BOUND CONTROL @
FIOSNC DATA X'07000003' PRIVATE PACK CONTROL WORD U14-0048
FIOSN DATA 0,0,0 SERIAL NUMBER ENTRIES U14-0049
FIOPASC DATA X'03000002' PASSWORD CONTROL @@@@@@@@
FIOPASS DATA 0,0 PASSWORD
DATA X'05000101' READ CONTROL
FIOREADC TEXT 'NONE' READ OPTION-NONE OR ALL
DATA X'06010101' WRITE CONTROL
FIOWRTC TEXT 'NONE' WRITE OPTION-NONE OR ALL

*
* FPTWAIT-SUSPEND OPERATIONS FOR INDICATED TIME
*
FPTWAIT DATA X'0F000000'

*
* FPTRSTRM-RESET TERMINAL MODES ON LEAVING APL
*
FPTRSTRM DATA X'06200000' CHANGE TERMINAL MODE
DATA X'E0000000' PRESENCE BITS
DATA X'8C' MASK-MODE
DATA X'6F' MODE2
DATA X'A0' MODE3
PAGE

*
* THE FOLLOWING AREA INCLUDES VALUES WHICH ARE SET AND MODIFIED
* DURING EXECUTION BUT CANNOT BE CONSIDERED TEMPS
*

```

```

*      DATA TO RESET TSTACK AFTER TRAP
*      ISPD-(INITIAL STACK-POINTER-DOUBLEWORD) AND LSPD-(ADDRESS OF SPD)
*
      BOUND      8
ISPD      RES      2      INITIAL SPD VALUE
LSPD      EQU      OPWSFLR1      LOCATION OF STACK PTR DW
*
*      WSIDNAME-DEFAULTED TO 'CLEAR WS'
*
      BOUND      8
WSIDPASS  DATA     0,0      WSID PASSWORD
WSOFFSET  DATA     0      WS LOAD OFFSET
LOADCASE  DATA     0      LOAD CATEGORY
WSIDNAME  TEXTC     'CLEAR WS'      WORKSPACE ID NAME-UP TO 11 CHAR'S.
*
*      ONLINE-OFFLINE MODE FLAG
*
ON%OFF    EQU      OPWSFLR2      ON%OFF-1=ON-LINE    0=OFF-LINE
*
*      MAXIMUM INPUT RECORD SIZE (DIFFERS FOR ON-LINE VS OFFLINE)
APLSIZ    RES      1
EORVAL    DATA     0      ON-LINE DEFAULT(ALLOWS 'LOOK-AHEAD')
DWIDTH    DATA     120      DEFAULT PRINT WIDTH
*      OUTPUT RECORD(F:OUT) BLOCKING SIZE (DIFFERS FOR ON-LINE VS OFF)
OUTMAXSZ  RES      1      MAX F:OUT BLOCK SIZE-GLOBAL
*
*      TERMINAL TYPE FLAG
TERMTYPE  DATA     1      DEFAULT TO 2741-APL
TERMKEY   DATA     X'03000000'      KEY OF TERMINAL RECORD
*
*      LOGON TIME IN 60THS OF SECONDS
*
LOGONTIM  RES      1
*
INITCPU   RES      1      INITIAL CPU USAGE.
INITOVH   RES      1      INITIAL OVERHEAD USAGE.
*
*      MEMORY BOUNDS
*
      BOUND      8
HIDYN     RES      2      HIGH END OF DYNAMIC (+1)
LOCOMMON  EQU      HIDYN+1      LOW END OF COMMON
DYNBOUND  EQU      HIDYN      DYNAMIC BOUND
STKLIMIT  EQU      LOCOMMON
LODYN     RES      1      LOWEST ADDRESS IN 'DYNAMIC'-FIXED
HICOMMON  RES      1      HIGHEST ADDRESS IN 'COMMON'-FIXED
CORAVAIL  RES      1      NUMBER OF WORDS AVAIL-TOTAL-FIXED
CORLEFT   RES      1      WORDS OF MEMORY YET AVAILABLE
*
*      BREAK FLAG
*
BREAKFLG  DATA     0      BREAK FLAG
*
*      QUITFLAG-USED TO PREVENT RUNAWAY OF ERROR MESSAGES ON
*      ATTEMPTS TO 'CONTINUE' IN OFF-LINE MODE
*
QUITFLAG  DATA     1
*
OUTSIZ    RES      1      OUTPUT RECORD SIZE
*
*      THIS REGION INCLUDES PARAMETERS WHICH MUST BE TRANSFERRED TO THE
*      COMMON AREA OF A WORKSPACE BY )SAVE OR )CONTINUE
*      AND WHICH MUST BE SET FROM THE COMMON AREA OF A WORKSPACE
*      BY )LOAD.
*
      BOUND      8
COMPRMST  RES      0      START OF COMMON PARAMETERS TABLE
FUZZBIT   DATA     0,X'200'      1 BIT IN LEAST SIG. POS.
FUZZLIMS  DATA     -X'35200000',X'35200000'      FUZZ LIMITS
FUZZMASK  DATA     -X'400'      RIGHT HAND SIGNIFICANCE MASK
FUZZCNT   DATA     10      COUNT OF BITS TO BE IGNORED
ORGADJ    DATA     0      1-ORIGIN
ORIGIN    DATA     1      1 OR 0
DIGITS    DATA     10
WIDTH     DATA     120
RANDOM     DATA     123456789      RANDOM SEED (MUST BE + AND ODD)
ECLEARST  RES      0
CLEARST   EQU      ECLEARST-COMPRMST SIZE OF 'CLEAR' TABLE

```


STATEPTR	DATA	0	SI STATE POINTER	
TOPOSTAK	DATA	0	TOP OF EXECUTION STACK	
MODE	DATA	1	-1=FORCED CLOSE 0=FN OPEN 1=DIR INP 2=EVAL INP	
OPENFN	DATA	0	POINTER TO FN OF FORCED CLOSE	
ECOMPRMS	RES	0		
NCOMPRMS	EQU		ECOMPRMS-COMPRMST SIZE OF COMMON PARAMETERS TABLE	
*				
PCOPYFLG	RES	1	PROTECTED COPY FLAG	
	BOUND	8		
LODACCT	TEXT	'	LOAD ACCOUNT-INITIALLY BLANK	
CMNDTYPE	RES	1	COMMAND TYPE-NEG. POS. IN VECTOR	
HOLDFLG	DATA	0	HOLD-OFF FLAG FOR CONTINUE	
SICTRL	DATA	0 = ON & NZ=OFF=NO	FN.SUSP ON ERR. STATE-INDIC.CTRL.	
GDIGITS	DATA	3	GRAPHIC DIGITS SETTING	
IOERCODE	DATA	0	MONITOR I/O ERROR CODE-SUBCODE	
	PAGE			
*				
* TEMPS USED BY APLUTSI				
*				
RETURN14	RES	1	RETURN FROM MEMORY ACCESS ROUTINES	
BCDTIME	RES	4	BCD TIME VALUE	
SAVE14	RES	1	SAVE R14	
	BOUND	8		
ERMSAV	RES	2	SAVE R4-5 FOR ERMS	
ERMSBUF	RES	10	BUFFER FOR ERROR MESSAGES	
	PAGE			
*				
* CONTEXT USED BY FIO SUBSYSTEM				
*				
FIOBUF	RES	1	POINTER TO FIO BUFFER (DATA BLOCK)	
FIOSIZ	RES	1	SIZE OF FIOBUF	
FIOABNT	RES	1	FIO INABN FLAG	
FIODCBNO	DATA	1	FIO DCB NO,DEFAULT=1	U14-0053
*				
* KEY TABLES FOR FIO SUBSYSTEM				
FRSTKEYT	D01	NUMFILES	# OF FIO DCB'S	
	DATA	0	PRESET TO 0	
CURRKEYT	D01	NUMFILES	# OF FIO DCB'S	
	DATA	0	PRESET TO 0	
LASTKEYT	D01	NUMFILES	# OF FIO DCB'S	
	DATA	0	PRESET TO 0	
*				
* TABLE OF FIO TIE VALUES				
*				
FIO TIE	D01	NUMFILES	NO. OF FIO DCBS	U14-0055
	DATA	0	PRESET TO ZERO	U14-0056
	PAGE			U14-0057
*				
* PATCH REGION-TO BE REMOVED BEFORE RELEASE				
*				
IVIE	DEF	IVIE		
	B	IVIE		
	STW,R14	IVIE+7		
	BAL,R14	WSCHEK		
	B	IVIE+5		
	DATA	0		
	LW,R14	IVIE+7		
	B	0		
	DATA	0		
	REF	WSCHEK		
	PAGE			
APLUTSI@	CSECT	1		
*				
* CONSTANT REGION				
*				
APLVERSN	BOUND	8		
DIGRANGE	TEXT	'APLWSB00'	MUST BE APLWS---, ---=VERSION	
WIDRANGE	DATA	1,16	DIGITS RANGE	
KEY1	DATA	30,254	WIDTH RANGE	
F0	DATA	X'03000001'	KEY VALUE 1	
XFF	DATA	X'F0'	MASK	
FFFFFFE00	DATA	X'FF'	MASK	
SIX	DATA	X'FFFFFFE00'	MASK FOR PAGE BOUNDING	
	DATA	6		
*				
* FPT'S WHICH CONTAIN NO VARIABLE PARAMETERS EXCEPT VIA INDIRECT ADDRESSING				
*				
*				

```

*
* FPTOPNXT-USED TO GET 'NEXT' FILE IN AN ACCOUNT
*
FPTOPNXT GEN,8,7,17 X'14',4,F:TF DCB NAME AND 'TEST' OPEN
      DATA X'00200401' P11 F2,F12
      DATA FPARAMS (P11) FILE PARAMETER BUFFER
*
* FPTRDAPL READ F:APL -TERMINAL OR CARD READER-
*
FPTRDAPL GEN,8,7,17 X'10',0,F:APL DCB ADDRESS
      DATA X'F4000010' P1,2,3,4,6 AND WAIT FLAG
      DATA ERRRDAPL ERR (P1)
      DATA ERRRDAPL ABN (P2)
      GEN,1,31 1,R8 BUF (P3)
      GEN,1,31 1,APLSIZ SIZ (P4)
      GEN,1,31 1,R9 BTD (P6)
*
* FPTRD1ST-READ FIRST RECORD OF A FILE-FOR FILE ID PURPOSES
*
FPTRD1ST GEN,8,7,17 X'10',0,F:TF DCB ADDRESS
      DATA X'F4000010' P1-P4,P6,F3
      DATA ERRFTF ERR (P1)
      DATA ERRFTF ABN (P2)
      DATA IDBUF BUF (P3)
      DATA 32 SIZ (P4)
      DATA 0 BTD (P6)
*
* FPTWROUT-FPT FOR OUPUT ON F:OUT
*
FPTWROUT GEN,8,7,17 X'11',0,F:OUT DCB ADDRESS
      DATA X'F4000010' OPTIONS AND WAIT FLAG
      DATA ERRWROUT ERROR EXIT P1
      DATA ABNWROUT ABN EXIT P2
      GEN,1,31 1,R8 P3
      GEN,1,31 1,R9 P4
      GEN,1,31 1,R10 P6
ERRWROUT EQU ERRFOUT F:OUT WRITE ERROR-QUIT
ABNWROUT EQU ABNFOUT F:OUT WRITE ABN -QUIT
*
* FPTRDQ READ 'BLIND INPUT' VIA M:QN (N=1,2...NBIO)
*
FPTRDQ GEN,1,7,7,17 1,X'10',0,R11 DCB ADDRESS IN R11
      DATA X'F4000010' P1,P2,P3,P4,P6 AND WAIT
      DATA ERRRDQ ERR (P1)
      DATA ABNRDQ ABN (P2)
      DATA QINBUF BUF (P3)
      DATA QINSIZ SIZ (P4)
      DATA 0
QINBUF EQU BLINBUF BLIND INPUT BUFFER
QINSIZ EQU 512 BLIND INPUT BUFFER SIZE
ABNRDQ EQU ERRRDQ
*
* FPTWRQ -WRITE RECORD VIA F:QN (BLIND OUTPUT)
*
FPTWRQ GEN,1,7,7,17 1,X'11',0,R6 DCB ADDRESS IN R6
      DATA X'F4000010' P1,P2,P3,P4,P6 AND WAIT
      DATA ERRWRQ ERR (P1)
      DATA ABNWRQ ABN (P2)
      GEN,1,31 1,RQWRBUF BUF (P3)
      GEN,1,31 1,RQWRSIZ SIZ (P4)
      DATA 0 BTD (P6)
ABNWRQ EQU ERRWRQ
RQWRBUF EQU R4
RQWRSIZ EQU R1
*
* FPTCLOSV-CLOSE AND SAVE A DCB
*
FPTCLOSV GEN,1,7,7,17 1,X'15',0,R5 DCB ADDRESS IN R5
      DATA X'80000000' P1
      DATA 2 SAVE
*
* FPTCLOSR-CLOSE AND RELEASE A DCB
*
FPTCLOSR GEN,1,7,7,17 1,X'15',0,R5 DCB ADDRESS IN R5
      DATA X'80000000' P1
      DATA 1 REL
*
* GET VIRTUAL PAGE

```

```

*
FPTGVP  GEN,1,7,7,17  1,X'04',0,R13 ADDRESS IN R13
*
*   FREE VIRTUAL PAGE
*
FPTFVP  GEN,1,7,7,17  1,X'05',0,R13 ADDRESS IN R13
*
*   FPTTIME-GET TIME,EBCDIC IN 4-WORD BLOCK AT *R6,BINARY IN R8-R10
*
FPTTIME DATA      X'90800006'      'TIME',TUN, ADDRESS *R6
*
*   FPTSNAPT-SNAPSHOT-TRAP
*
FPTSNAPT DATA      0
          DATA      MODTBLS          DISPLAY ADDRESSES
          DATA      MODTBLE          OF ASSEMBLY MODULES
          TEXT      '***TRAP**'
          B          CALSNAPT+1        EXIT TO CALSNAPT+1
*
*   FPTSNAAPS-SNAPSHOT-SYSTERR
*
FPTSNAAPS DATA      0
          DATA      MODTBLS          DISPLAY ADDRESSES
          DATA      MODTBLE          OF ASSEMBLY MODULES
          TEXT      'SYS ERR '
          B          CALSNAAPS+1
*
*   FPTSNAPI-SNAPSHOT-I/O ERROR
*
FPTSNAPI DATA      0
          DATA      MODTBLS          DISPLAY ADDRESSES
          DATA      MODTBLE          OF ASSEMBLY MODULES
          TEXT      'I/O ERR '
          B          CALSNAPI+1        GO TO ERR EXIT
*
*   FPTSNA PW-FPT SNAPSHOT-WS DAMAGE DETECTED
*
FPTSNA PW DATA      0
          DATA      MODTBLS          DISPLAY ADDRESSES
          DATA      MODTBLE          OF ASSEMBLY MODULES
          TEXT      '*BAD WS*'
          B          CALSNA PW+1
*
*   FPTSNA PA-FPT SNAPSHOT-RUN ABORTED BY OPERATOR
*
FPTSNA PA DATA      0
          DATA      MODTBLS
          DATA      MODTBLE
          TEXT      'ABORTED '
          B          CALSNA PA+1
          PAGE
*
*   BREAK-RESPONSE TO USERS INTERRUPT
*
*   IF XSEGBRK IS SET,RESETS XSEGBRK,RESETS STACK,EXITS TO OPBREAK
*   OTHERWISE,SETS BREAK FLAG,RETURNS TO INTERRUPTED OPERATION
*   FOR DELAYED BREAK RESPONSE
*
BREAK    MTW,1        BREAKFLG          SET BREAK FLAG
          LW,R2        XSEGBRK          CHECK FOR BREAK IN EXECUTION SEG.
          AI,R2        -OPBREAK
          BNEZ         DLYBREAK          NO
          STW,R2       XSEGBRK          YES-RESET FLAG
          LD,R8        ISPD             RESTORE
          STD,R8       *LSPD            STACK
          B            OPBREAK          *BREF* EXIT TO OPBREAK
DLYBREAK LI,R2        CALWR0UT          CHECK FOR BREAK ON TERMINAL OUTPUT
          LW,R3        X1FFFF
          CS,R2        *R1
          BNE         CALTRTN          NO
          MTW,1        *R1             YES-SKIP THE CAL
CALTRTN  CAL1,9       5                TRAP-BREAK RETURN
*
*   TRAP-ROUTINE TO HANDLE TRAPS
*
*   IF NOT FL PT TRAP: SYSTEM ERROR
*   IF FL PT OVERFLOW: DOMAIN ERROR
*   IF DIV BY 0 BUT NOT R6:DOMAIN ERROR
*   IF DIV BY 0 AND R6:CALL DIVZERO

```

```

*                                     RETURN TO CALL+1:DOMAIN ERROR
*                                     RETURN TO CALL+2:RESTORE R6-7 AND TRTN
*
*      TRAP (TSTACK) IS RESET  BY RESTRAP IF EXIT IS TO ERROR ROUTINE
*
*      LOCAL      FLTRAP,DOMERR
*      TRAP      LW,R0      18,R1      GET TYPE OF TRAP
*                STW,R0     XSEGBRK    RESET IMMEDIATE BRK RSP FLAG      14-00007
*                CI,R0      X'44'      CHECK IT
*                BE         FLTRAP      FLOATING
*                LW,R0      0,R1        R0= 1ST HALF OF PSD
*                LW,R1      3,R1        R1= PRE-TRAP VALUE OF R1
*                BAL,R15    CHKTERM     SWITCH IF 4013
*      CALSNAPT   CAL1,3    FPTSNAPT    GENERATE SNAPSHOT
*                B         ERREXIT     CRASH
*      FLTRAP     LW,R2      0,R1        CHECK WHICH FLOATING TRAP
*                LC         R2
*                BCS,3      DOMERR      OVERFLOW
*                LW,R4      *R1         DIV-BY-ZERO,GET PSD WORD 1
*                LH,R4      *R4         GET TRAPPED INSTRUCTION
*                AND,R4     F0          MASK 'R' FIELD
*                CI,R4      X'60'       CHECK IF R6
*                BNE        DOMERR      DOMAIN ERROR
*                LW,R6      8,R1        GET OLD R6 AND R7
*                LW,R7      9,R1
*                BAL,R5     DIVZERO     LET OPERATOR EXECUTION ROUTINE TEST
*                B         DOMERR      NOT VALID-DOMAIN ERROR
*                STW,R6     8,R1        VALID-PUT R6 AND R7
*                STW,R7     9,R1        BACK IN TSTACK
*                MTW,1      0,R1        KICK INSTRUCTION ADDRESS
*                B         CALTRTN      TRAP RETURN
*      DOMERR     LD,R8      ISPD        RESET
*                STD,R8     *LSPD       TRAP AND
*                B         ERDOMAIN    *BREF*   EXIT
*                PAGE
*
*      *
*      * HANGUP-EXIT CONTROL ROUTINE-REACHED BY
*      * HANGUP OR OTHER EXTERNAL CAUSE
*      *
*      HANGUP     RES      0
*      CALXCOPH   CAL1,8    FPTXCOFF    RESET EXIT CONTROL
*                CI,R8      8+4        CHK 4 LINE DISCONNECT OR LIMIT
*                BAZ        HANGUP1    NO
*                LI,R7      -5         YES-SET FLAG FOR HANGUP
*                STW,R7     BREAKFLG
*      CALXRTN    CAL1,9    256+5      RETURN TO WIND UP PROCESSING
*      HANGUP1    CI,R8      X'F0'      CHECK FOR OPERATOR ABORT
*                BAZ        HANGUP2    YES
*                CI,R8      1          NO-CHECK FOR TRAP
*                BAZ        CALSNAPT    YES
*                B         IOERR       NO-ASSUME IO ERROR
*      HANGUP2    RES      0
*      CALSNAPA   CAL1,3    FPTSNAPA    SEND ABORT MESSAGE
*                B         CALRSTRM    AND EXIT
*
*      *
*      * FPTXCOFF-TURN OFF EXIT CONTROL
*      *
*      FPTXCOFF   DATA     X'19000000'
*                PAGE
*
*      *
*      * DELAYER-ROUTINE  TO DELAY SPECIFIED INTERVAL
*      *
*      * R5=LINK R7=NO. OF SECONDS TO DELAY
*      *
*      *
*      DELAYER    AI,R7      1
*                MI,R7      5
*                DW,R7      SIX        (THE CAL USES  1.2 SECOND UNITS)
*                STW,R7     FPTWAIT
*                LI,R7      X'0F'
*                STB,R7     FPTWAIT
*      CALWAIT    CAL1,8    FPTWAIT    GENERATE WAIT CALL TO UTS
*                B         0,R5
*                PAGE
*
*      *
*      * RELEASER-ROUTINE TO RELEASE PAGES AS INDICKTED BY
*      * TOPOSTAK AND FREETOTL
*      *

```

```

*      R11=LINK
*      R8,12,13,AND 14 USED INTERNALLY
*
*      ROUTINES CALLED:RELCOM   LINK R14
*                      GARBCOLL LINK R8 -NO OTHER R'S CHANGED
*                      RELDYN   LINK R14-USES R12
*
*      LOC'S UPDATED:DYNBOUND,FREETBL+1,FREETOTL,STKLIMIT
*
*      LOCAL      OVLAP
* RELEASER MTW,0   CORLEFT      CHECK IF ALL PAGES IN USE
*          BLEZ    OVLAP        YES
*          BAL,R14 RELCOM        NO-RELEASE COMMON TO TOPOSTAK
*          LW,R14  FREETOTL      CHECK DYNAMIC STATUS
*          AI,R14  -((MINANNEX+1)*512)
*          BLEZ    *R11          NO DYNAMIC PAGES TO RELEASE
*          BAL,R8  GARBCOLL      GARBAGE COLLECT BEFORE RELEASE
* SRELDYN  LW,R12  FREETBL
*          AI,R12  MINANNEX*512  GET NEW DYNBOUND
*          BAL,R14 RELDYN
*          LW,R14  DYNBOUND
*          SW,R14  FREETBL      SET SINGLE FREETABLE ENTRY TO
*          STW,R14 FREETBL+1    THE WHOLE SCHMEER
*          STW,R14 FREETOTL     WHICH EQUALS FREETOTL
*          B       *R11          EXIT
* OVLAP    BAL,R8  GARBCOLL      GARBAGE COLLECT IF OVERLAP
*          LW,R13  TOPOSTAK
*          AND,R13 FFFFE00      GET PAGE BOUND FOR STAKLIM
*          LW,R14  R13          SAVE IT
*          SW,R14  FREETBL      CHECK IF RELEASE POSSIBLE
*          AI,R14  -((MINANNEX+1)*512)
*          BLEZ    *R11          NO-EXIT
*          STD,R13 DYNBOUND      YES-SET DYNBOUND AND STKLIMIT
*          B       SRELDYN      THEN RELEASE DYNAMIC
*          PAGE
*
*      CLEARMEM-CLEAR COMMON AND DYNAMIC      )CLEAR OR )LOAD
*
*      R11=LINK, USES CLEARCOM AND CLEARLYN ROUTINES
*      R7,R12,R14 VOLATILE
*      NOTE:RESOLVES OVERLAPPED COMMON-DYNAMIC USE OF PAGE BY SHIFTING LIMIT
*      BOUNDS TO COMMON SIDE OF PAGE
*
*      LOCAL      NOVLAP
* CLEARMEM MTW,0   CORLEFT      CHECK IF ALL CORE USED
*          BGZ     NOVLAP        NO
*          LW,R7   DYNBOUND      YES-FORCE DYNBOUND AND
*          AI,R7   511           STKLIMIT TO START
*          AND,R7  FFFFE00      OF PAGE
*          CW,R7   HICOMMON      CHECK IF TOO MUCH
*          BL      NOVLAP-1      NO
*          AI,R7   -512          YES-MOVE BACK 1 PAGE
*          STD,R7  DYNBOUND
* NOVLAP   BAL,R14 CLEARLYN      CLEAR DYNAMIC
*          BAL,R14 CLEARCOM      CLEAR COMMON
*          B       *R11          RETURN
*          PAGE
*
*      RELDYN-ROUTINE TO RELEASE ONE OR MORE 'DYNAMIC' PAGES
*      TWO ENTRIES:
*      1. RELDYN-RELEASE PAGES TOWARD ADDRESS IN R12
*          THIS IS PARTIAL CLEAR
*      2. CLEARLYN-RELEASE PAGES TOWARD LODYN,SUBJECT TO MINALLOW.
*          THIS ENTRY FOR TOTAL CLEAR TO MINIMUM.
*      REGISTERS:R14 IS LINK. R13 IS USED BUT RESTORED.
*          R12 USED.
*      IF PAGES ARE RELEASED,HIDYN AND CORLEFT ARE CHANGED
*      LOCAL      FREEDYN,CHECK
* RELDYN  STW,R14 RETURN14      SAVE LINK
* FREEDYN XW,R13  HIDYN        GET HIDYN-SAVE R13 TEMPORARILY
*          AI,R13 -512          MOVE BACK TO START OF HIGHEST PAGE
* CHECK   CW,R13  R12          CHECK IF PAGE TO BE RELEASED
*          BGE    CALFVPDY
*          AI,R13 512          NO-SET FORWARD
*          B      SETCORHY      EXIT
* CALFVPDY CAL1,8 FPTFVP      YES-RELEASE PAGE-ADDRESS IN R13
*          AI,R13 -512          BACK UP 1 PAGE
*          B      CHECK

```

```

CLEAR DYN LI,R12 MINALLOW*512 GET INITIAL ALLOWANCE
          AW,R12 LODYN SET UP COMPARISON ADDRESS
          B RELDYN
          LOCAL

*
* RELCOM-ROUTINE TO RELEASE ONE OR MORE COMMON PAGES
* TWO ENTRIES:
* 1. RELCOM-RELEASE TOWARD TOPOSTAK
* 2. CLEARCOM-RELEASE TOWARD HICOM
* REGISTERS:R14 IS LINK. R13 IS USED BUT RESTORED.
* IF PAGES ARE RELEASED,LOCOMMON AND CORLEFT ARE CHANGED
* IF PAGES ARE RELEASED,LOCOM IS CHANGED.
*
*
          LOCAL FREECOM,CHECK
RELCOM STW,R14 RETURN14 SAVE LINK
        LW,R14 TOPOSTAK GET LOWEST COMMON ADDRESS NEEDED
FREECOM AND,R14 FFFFFFFE00 REDUCE TO PAGE ADDRESS
        XW,R13 LOCOMMON GET LOCOMMON, SAVE R13
CHECK CW,R13 R14 CHECK IF PAGE TO BE RELEASED
        BGE SETCORLO IF NOT,RESTORE LOCOM AND R13
CALFVPCO CAL1,8 FPTFVP YES-RELEASE PAGE-ADDRESS IN R13
        AI,R13 512 MOVE UP 1 PAGE
        B CHECK
CLEARCOM STW,R14 RETURN14 SAVE LINK
        LW,R14 HICOMMON SET UP TO RELEASE
        B FREECOM ALL BUT 1 PAGE OF COMMON
        LOCAL

*
* GETDYN-ROUTINE TO GET DYNAMIC PAGES AS INDICATED BY ADDRESS
* NEWBOUND AND MININCR(MINIMUM INCREMENT)
* (VIRTUAL ACCESS IS USED)
* IF PAGES ARE ACCESSED,HIDYN IS CHANGED
*
* ROUTINE ALSO HAS COMMON EXIT FOR RELDYN,RELCOM,AND GETCOM
* ROUTINES,WHICH UPDATES CORLEFT AFTER ANY CHANGE.
* REGISTERS:R14 IS LINK. R13 IS USED BUT RESTORED.
*
          LOCAL CHECK,SETCOR,SETCORZ
GETDYN STW,R14 RETURN14 SET RETURN
        LW,R14 NEWBOUND GET TARGET ADDRESS
        AI,R14 MININCR*512 -AND INCREMENT-
        XW,R13 HIDYN GET LAST IN-CORE ADDRESS+1
CHECK CW,R13 R14 CHECK IF DONE
        BG SETCORHY YES
        CW,R13 LOCOMMON NO-CHECK FOR ENCROACHMENT ON
        BGE SETCORHY COMMON
CALGVDPY CAL1,8 FPTGVP NO-GET A PAGE
        AI,R13 512 AND LOOP
        B CHECK
SETCORHY XW,R13 HIDYN SET HIDYN-RESTORE R13
SETCORZ LW,R14 LOCOMMON
        SW,R14 HIDYN CALCULATE CORE LEFT
        STW,R14 CORLEFT STASH IT
        LW,R14 RETURN14 RESTORE R14
        B *R14 AND EXIT
SETCORLO XW,R13 LOCOMMON SET LOCOMMON-RESTORE R13
        B SETCORZ

*
* GETCOM-ROUTINE TO GET A PAGE OF COMMON (VIRTUAL ACCESS IS USED)
*
* IF PAGE IS ACCESSED,LOCOMMON IS CHANGED
*
* REGISTERS:R14 IS LINK. R13 IS USED BUT RESTORED.
*
          LOCAL
GETCOM STW,R14 RETURN14 SAVE RETURN
        XW,R13 LOCOMMON GET LOWEST COMMON ADDRESS-SAVE R13
        AI,R13 -512 BACK 1 PAGE
CALGVPCO CAL1,8 FPTGVP GET IT
        B SETCORLO EXIT (UPDATE LOCOM,RESTORE R13)
        PAGE

*
* CALTIME-ROUTINE TO GET TIME
*
* R7=LINK R6 HAS ADDRESS OF 4-WORD BLOCK TO HOLD EBCDIC VERSION
* ON EXIT, R8,R9,R10 HAVE BINARY VERSION OF TIME-DATE
*

```

```

GETTIME  LI,R6      BCDTIME          SET ADDRESS FOR BLD VERSION
CALTIME  CAL1,8    FPTTIME
          B          0,R7
          PAGE
*
* NUMUSERS-DELIVERS NUMBER OF USERS IN R11
*
*          R14=LINK  R5,R6,R7 VOLATIILE
*
NUMUSERS  RES          0
CALDSPL   CAL1,8      FPTDSPL          CALL UTS 'DISPLAY'
          LW,R11      R7              NO. USERS INTO R11
          B          *R14            RETURN
          DATA      X'13000000'
          PAGE
*
* CPUTIME-DELIVERS CPU TIME (60THS OF A SECOND) IN R11
*
*          R14=LINK  R10 VOLATILE
*
CPUTIME   LW,R11      J:DELTAT
          SW,R11      J:IDELTAT
          AW,R11      J:PTIME
          AW,R11      J:UTIME
          SW,R11      INITCPU
CV60THS   MI,R11      12
          DW,R11      TENSTBL+2      CONVERT TO 1/60THS' OF SECONDS
          B          *R14
*
* OVHTIME-DELIVERS OVERHEAD TIME (60THS OF A SECOND) IN R11
*
*          R14=LINK  R10 VOLATILE
*
OVERTIME  LW,R11      J:PTIME+1
          AW,R11      J:UTIME+1
          SW,R11      INITOVH
          B          CV60THS
          PAGE
*
* TIMODAY-DELIVERS TIME OF DAY IN 60THS OF SECONDS IN R11
*
*          R14=LINK, R6-R13 VOLATILE
*
TIMODAY   LOCAL      TIMLOOP
          BAL,R7      GETTIME          GET TIME
          LI,R7       3
          LB,R11      R9,R7          'TMS'
          MI,R11      12
          DW,R11      TENSTBL+2      CONVERT TO 60'THS OF SECONDS
TIMLOOP   AI,R7       -1
          BLZ        *R14
          LB,R13      R9,R7          HOURS,MINUTES,SECONDS
          MW,R13      T60S,R7
          AW,R11      R13
          B          TIMLOOP
T60S      DATA      216000,3600,60
          PAGE
*
* DATE-DELIVERS DATE-IN WEIRD(THANKS TO IBM) UNITS-IN R17
*
*          R14=LINK, R6-R10 VOLATILE
*
DATE      LOCAL      MONLOOP,ADDMONS,MONTBL
          BAL,R7      GETTIME
          LH,R11      R8              YEAR
          LB,R7       BCDTIME+2      3RD CHAR OF MON
          LW,R6       BCDTIME+1      D-BLANK-FIRST 2 CHARS OF MON
          STB,R7      R6              3RD CHAR-BLANK-1ST CHAR-2ND CHAR
          LI,R7       12
MONLOOP   CW,R6       MONTBL-1,R7
          BE
          BDR,R7      MONLOOP
ADDMONS   MI,R7       10000
          AW,R11      R7              YEARS+10,000*MONTHS
          LI,R6       10              TENS OF DAYS PTR
          LB,R7       BCDTIME,R6
          AI,R7       -X'F0'          CONVERT TO BINARY
          MI,R7       1000

```

```

        AW,R11    R7
        LI,R6     11                DAYS-UNITS PTR
        LB,R7     BCDTIME,R6
        AI,R7     -X'F0'            CONVERT TO BINARY
        MI,R7     100                10;000*MONTHS+100*DAYS+YEARS
        AW,R11    R7
        B         *R14
MONTBL  TEXT     'N JAB FER MAR APY MAN JUL JUG AUP SET OCV NOC DE'
        PAGE

*
* CLOSV-CLOSE AND SAVE
*
*         DCB ADDRESS IN R5
*         R6=LINK  R7 USED
*
CLOSV   LW,R7     *R5                GET WORD 0 OF DCB
        CW,R7     BITPOS-10          CHECK IF DCB OPEN
        BAZ       0,R6                NO-RETURN
CALCLOSV CAL1,1   FPTCLOSV           YES-CLOSE IT
        B         0,R6                RETURN
*
* CLOSР-CLOSE AND RELEASE
*
*         DCB ADDRESS IN R5
*         R6=LINK  R7 USED
*
CLOSР   LW,R7     *R5                GET WORD 0 OF DCB
        CW,R7     BITPOS-10          CHECK IF DCB OPEN
        BAZ       0,R6                NO
CALCLOSР CAL1,1   FPTCLOSР           YES-CLOSE IT
        B         0,R6                EXIT
*
*         RDADL-READ AN INPUT RECORD FROM F:ADL
*
*         R11 IS LINK
*         R7,R8,R9 ARE USED INTERNALLY
*         ON ENTRY:OUTSIZ CONTAINS # OF BYTES OFFSET FROM INBUF
*         TO START INPUT
*
*         RECORD IS READ INTO INBUF AS OFFSET BY OUTSIZ
*
*         IF INPUT IS VIA CARD READER,CR IS INSERTED IN COLUMN 73
*         IF INPUT IS VIA OTHER SOURCE, '0' IS SET AFTER LAST BYTE AS
*         END-OF-RECORD FLAG
*
*         OUTSIZ IS RESET TO 0
*         ON EXIT, R7=-1
*
RDAPL   LOCAL     CARDIN,SETEOR
        LW,R8     OUTSIZ              GET PROMPF OFFSET-IF ANY
        CI,R8     520                 CHECK IF TOO BIG
        BL       RDAPL1              NO
        BAL,R15   SYSTERR             YES-SYSTEM ERROR
RDAPL1  LI,R9     0
        SLD,R8    -2                  SHIFT TO WORD OFFSET
        SCS,R9    2                   SET BYTE DISPLACEMENT
        AI,R8     INBUF               BASE OF INPUT BUFFER
CALRDAPL CAL1,1   FPTRDAPL           * READ A RECORD *
        LI,R7     0
        LW,R8     BREAKFLG           CHECK BREAK
        BLEZ     %+2                 OK-OR HANG-UP
        STW,R7    BREAKFLG           BREAK-RESET IT
        XW,R7     OUTSIZ              RESET PROMPT OFFSET-SAVE IN R7
        LH,R8     F:APL+4            SET END-OF-RECORD FLAG
        SLS,R8    -1                  AT BYTE AFTER
        AW,R7     R8                  LAST INPUT BYTE
        LW,R8     EORVAL              GET ON-LINE OR OFF-LINE EOR VALUE
SETEOR  STB,R8    INBUF,R7           SET END-OF-RECORD
        LI,R7     -1                 PRESET INCOL POSITION
        B         *R11               RETURN
        PAGE
*
* CALWROUT-OUTPUTS RECORD VIA F:OUT
*
*         R7=LINK
*         R8=BUFFER ADDRESS
*         R9=SIZE IN BYTES
*         R10=BYTE DISPLACEMENT

```



```

*
* IF ENTRY IS VIA WROUT-
* OUTPUT IS ABORTED IF BREAKFLG IS SET
*
EWROUTWB EQU WROUTWB ERROR CALL TO WROUTWB.
WROUTWB LI,R10 0 WRITE FROM WORD BOUND
WROUT MTW,0 BREAKFLG CHECK FOR BREAK
BNEZ 0,R7 YES-EXIT
CALWROUT CAL1,1 FPTWROUT OUTPUT A RECORD ON F:OUT
B 0,R7 THEN EXIT
*
* WRTEXTC-WRITE TEXTC ERROR MESSAGE
*
* R7=LINK R8,R9,R10 USED
*
* ON ENTRY, R8=ADDR OF MESSAGE
*
EWRTEXTC EQU WRTEXTC ERROR CALL TO WRTEXTC.
WRTEXTC STD,R4 ERMSAV SAVE R4-R5
LB,R4 *R8 SET COUNT
WRTEXTC1 LB,R5 *R8,R4 GET BYTE
LB,R5 OUTRANST,R5 TRANSLATE IT
STB,R5 ERMSBUF,R4 SAVE IT
BDR,R4 WRTEXTC1 LOOP
LD,R4 ERMSAV RESTORE R4-R5
LB,R9 *R8 SET COUNT
LI,R10 1 OFFSET
LI,R8 ERMSBUF AND BUFFER ADDRESS
B WROUT USE WROUT
PAGE
*
* QTABULID-CHECK VALIDITY OF TABS
*
* ON ENTRY-R4=ADDRESS OF 1ST TAB R6= # OF TABS
* R4,R8 VOLATILE
* R6 SAVED IN SAVE312, THEN VOLATILE
* R5=LINK
*
* RETURNS TO CALL+1 ON ERROR, CALL+2 IF OK
*
QTABVLID LOCAL GETNUTAB,ONETABV
STW,R6 SAVE312 SAVE # OF TABS
CI,R6 1
BL 1,R5 NO VALUE-OK RETURN
BE ONETABV ONE VALUE
CI,R6 16 CHECK IF TOO MANY
BG 0,R5 YES
AI,R6 -1
GETNUTAB LW,R8 0,R4
CI,R8 2
BLE 0,R5 TOO LOW-ERROR RETURN
CI,R8 128
BGE 0,R5 TOO HIGH-ERROR RETURN
CW,R8 1,R4
BGE 0,R5 NOT ASCENDING-ERROR RETURN
AI,R4 1
BDR,R6 GETNUTAB LOOP
B 1,R5 NORMAL EXIT
ONETABV LW,R8 0,R4 SINGLE VALUE
BLZ 0,R5 NEGATIVE-BAD
BEZ 1,R5 0-OK
AI,R8 -2
BLEZ 0,R5 1 OR 2-BAD
AI,R8 -126
BGEZ 0,R5 128 OR MORE-BAD
B 1,R5 OK
*
* SNEWTABS-SET NEW TABS(IF INDICATED)
*
* R5=LINK,R6 AND R8 VOLATILE
*
* ON ENTRY, SAVE312=# OF TABS TO BE SET
* IF ZERO,RETURN WITHOUT CHANGING TAB SETTING
* R4=ADDRESS OF FIRST NEW TAB
*
* VALUES HAVE BEEN VALIDATED BY 'QTABVLID'
*
LOCAL TABLOOP1,SEVTABS,LASTAB,TABLOOP2

```

SNEWTABS	LW, R6	SAVE312	GET # OF TABS
	CI, R6	1	
	BL	0, R5	NONE
	BG	SEVTABS	SEVERAL
	LW, R8	*R4	ONE-TAB INTERVAL
	STW, R8	TABPNTR	SET TAB POINTER WITH INTERVAL
TABLOOP1	STB, R8	TABVALS, R6	SET A TAB
	AI, R6	1	
	CI, R6	16	CHECK HOW MANY
	BG	LASTAB	ENOUGH
	AW, R8	*R4	INCREMENT BY INTERVAL
	CI, R8	128	CHECK SIZE
	BL	TABLOOP1	OK
LASTAB	AI, R6	-1	DONE
	STB, R6	TABVALS	SET COUNT
	B	CALTABS	DO CAL
SEVTABS	STB, R6	TABVALS	SEVERAL TABS-SET COUNT
	AI, R6	1	
	LI, R8	0	
	STB, R8	TABVALS, R6	SET END FLAG
	AI, R6	-1	
	AI, R4	-1	
TABLOOP2	LW, R8	*R4, R6	SET VALUES
	STB, R8	TABVALS, R6	
	BDR, R6	TABLOOP2	
	LI, R8	128	SET HI VALUE
	STW, R8	TABPNTR	
CALTABS	CAL1, 1	FPTTABS	
	B	0, R5	RETURN

*
 * TABSET-CHECK VALIDITY OF NEW TABS-SET NEW-GET OLD VALUE(S)
 * IN CONSTBUF
 * R2=LINK QTABVLID AND SNEWTABS ARE USED
 *
 * ON ENTRY- R6=# OF TABS R4=ADDRESS OF 1ST TAB
 *
 * ERROR-EXIT TO CALL+1 NORMAL EXIT TO CALL+2
 *
 * R4, R5, R6, R8 VOLATILE
 *

TABSET	LOCAL	SOLDTABS, TABSET1, TABSET2	
	STW, R4	SAVE312+1	SAVE ADDRESS OF 1ST TAB
	BAL, R5	QTABVLID	CHECK VALIDITY
	B	0, R2	NO
	LW, R6	TABPNTR	YES
	CI, R6	128	CHECK VARIETY
	BL	TABSET2	1 VALUE
	LB, R6	TABVALS	VECTOR
	STW, R6	CONSTBUF	SET SIZE
SOLDTABS	LB, R4	TABVALS, R6	
	STW, R4	CONSTBUF, R6	SET OLD VALUES
	BDR, R6	SOLDTABS	
TABSET1	LW, R4	SAVE312+1	GET ADDRESS OF 1ST NEW TAB
	BAL, R5	SNEWTABS	SET NEW TABS
	B	1, R2	RETURN
TABSET2	STW, R6	CONSTBUF+1	SET 1 VALUE
	LI, R5	1	
	STW, R5	CONSTBUF	INDICATE 1 VALUE
	B	TABSET1	
	PAGE		

*
 * SETWIDTH-SET NEW WIDTH, GET OLD
 *

* R5=LINK
 *
 * ON ENTRY: R7=NEW WIDTH(INTEGER)
 * ERROR EXIT-CALL+1 (OUT OF RANGE)
 * NORMAL EXIT-CALL+2 R7=OLD VALUE
 *

SETWIDTH	CLM, R7	WIDRANGE	CHECK RANGE
	BCS, 9	0, R5	NO
	XW, R7	WIDTH	OK-SET NEW, GET OLD
	B	1, R5	

*
 * SETDIGIT-SET NEW DIGIT VALUE, GET OLD
 *

* R5=LINK R7 IS NEW VALUE ON ENTRY, OLD ON EXIT
 *

```

*      ERROR RETURN TO CALL+1  NORMAL TO CALL+2
*
SETDIGIT CLM,R7      DIGRANGE      CHECK RANGE
          BCS,9      0,R5          NO
          XW,R7      DIGITS        OK
          B          1,R5
          PAGE

*
* INPBLIND-BLIND INPUT ROUTINE  READ UP TO 512 BYTES FROM F:QN
*
*      R14=LINK
*      R6 =QUAD NUMBER-CHANGED IN ROUTINE
*      ON EXIT,R11=# OF CHARACTERS IN BLINBUF
*      NO OTHER REGISTERS USED
INPBLIND  LW,R11      FQTABL-1,R6      GET DCB ADDRESS
CALRDQ    CAL1,1      FPTRDQ          READ RECORD
          LI,R6      8
          LH,R11     *R11,R6
          SLS,R11     -1
RDQEX     AI,R14      1              GET ACTUAL RECORD SIZE
          B          *R14             NORMAL RETURN IS TO CALL+2
ERRRDQ    RES        0
ERRWRQ    RES        0
          STW,R10     IOERCODE        SAVE MONITOR CODE-SUBCODE
          LB,R10      IOERCODE
          CI,R10      6              CHECK FOR EOF
          BNE        *R14             NO-ERROR EXIT
          LI,R11      0              YES-SET LENGTH TO 0
          B          RDQEX

*
* FQTABL-USED BY INPBLIND AND BLINDOUT
*
VDATA     DATA      F:OUT          (FQTABL-1)-USED BY CLOSFILS
          CNAME
          PROC
N          DO         AF(1)
LF(N)     DATA      AF(N+1)
          FIN
          PEND
FQTABL    VDATA      NBIO-1,F:Q1,F:Q2,F:Q3,F:Q4,F:Q5,F:Q6,F:Q7,F:Q8,F:Q9
*
* FIODCBT-TABLE OF DCB ADDRESSES
*      USED BY FIO SUBSYSTEM
*
*      NOTE:ASSEMBLY WILL INDICATE ERRORS UNLESS NUMFILES=8
*      THE ORG AT THE BOTTOM OF THE TABLE NULLIFIES THE APPARENT
*      ERROR
*
FIODCBT   VDATA      NUMFILES,F:F1,F:F2,F:F3,F:F4,F:F5,F:F6,F:F7,F:F8
*
* BLINDOUT-BLIND OUTPUT ROUTINE GENERATE UNTRANSLATED TEXT OUTPUT
*
BLINDOUT  LOCAL      GETSIZE,BLIND1
          LW,R6      FQTABL-1,R6      GET DCB ADDRESS IN R6
          LI,R1      1
          LB,R5      *R4,R1          GET RANK
          BEZ        BLIND1          SCALAR
          STW,R5      RANKARR
          AI,R4      1              R4=LOC OF 61-1
          LI,R1      1
GETSIZE   MW,R1      *R4,R5          GET SIZE IN R1
          BDR,R5      GETSIZE
          AW,R4      RANKARR
          AI,R4      1
          CALWRQ     CAL1,1          GET DATA WORD ADDRESS IN R4
          CAL1,1      FPTWRQ        WRITE RECORD
          AI,R14      1              NORMAL RETURN TO CALL+2
          B          *R14
BLIND1    LI,R1      1              SET LENGTH TO 1
          AI,R4      2              AND POINTER TO SCALAR
          B          CALWRQ
          PAGE

*
* ERROR ROUTINES ASSOCIATED WITH MONITOR INTERFACES
*
*
ERRFOUT   EQU        IOERR
ABNFOUT   EQU        ERRFOUT          F:OUT OPEN  ABN  -QUIT
*

```

* ERRFTF-ERROR ON F:TF OPERATION

*

ERRFTF	AND,R8	X1FFFF	ADDRESS MASK
	STW,R10	IOERCODE	SAVE MONITOR ERROR CODE-SUBCODE
	CI,R8	CALOPTRI+1	CHECK IF OPEN FOR INITIAL TERM
	BE	IOERR	YES-CRASH
	CI,R8	CALRDTRI+1	CHECK IF READ FOR INITIAL TERM
	BE	IOERR	YES-CRASH
	CI,R8	CALOPTRM+1	CHECK IF OPEN FOR)TERM
	BE	IOERRCM	YES-TERM FILE LOST
	CI,R8	CALRDTRM+1	CHECK IF READ FOR)TERM
	BNE	ERRFTF1	NO
	LI,R5	F:TF	YES-INDICATE DCB
	BAL,R6	CLOSV	CLOSE AND SAVE
	B	ERBADCMD	
ERRFTF1	CI,R8	CALOP1ST+1	CHECK IF FIRST)LIB FILE
	BE	ERROP1ST	YES
	CI,R8	CALOPNXT+1	NO-CHECK IF LATER)LIB FILE
	BNE	ERRFTF2	
ERROP1ST	LI,R9	2	
	CB,R9	R10	CHECK IF LAST FILE
	BNE	PRINTFNE	NO-CONTINUE
BCMDEXIT	B	CMDEXIT	*BREF* COMMAND EXIT
PRINTFNM	LI,R9	0	FLAG-NOT PROTECTED FILE
PRINTFNE	LI,R7	1	
	LI,R6	F:TF+23	ADDRESS OF NAME
	LB,R10	*R6	BYTE COUNT
	SLS,R6	2	BYTE ADDRESS (OF COUNT)
	LI,R3	0	IMAGE OFFSET
	BAL,R13	GENTEXT	GENERATE NAME
	CI,R9	2	CHECK IF PROTECTED
	BNE	BCLOSTHS-1	NO
	LB,R10	PROTECTD	
	LI,R6	BA(PROTECTD)	
	BAL,R13	GENTEXT	ISSUE RESTRICTED ACCESS MESSAGE
	BAL,R12	DUMPLING	SHOVE LINE OUT
BCLOSTHS	B	CLOSTHS	*BREF* RESUME FILE SEARCH
PROTECTD	TEXTC	'*RESTRICTED ACCESS*'	
ERRFTF2	RES	0	
	CI,R8	CALOPTF+1	CHECK IF OPEN FOR COPY TFILE
	BE	ERRCOPY	YES-QUIT
	CI,R8	CALRDTF+1	CHECK IF COPY TFILE READ
	BE	IOERR	YES-QUIT
	CI,R8	CALWRTF+1	CHECK IF COPY TFILE WRITE
	BE	IOERR	YES-QUIT
	LB,R10	R10	SET CODE-ONLY IN R10
	B	ERRFTF10	FIO SUBSYSTEM ERROR HANDLER

*

* ERRFWS-ERROR OR ABN RETURN ON CAL TO F:WS

*

* CHECKS WHICH CALL ERRORED AND-AS APPROPRIATE-THE ERROR CONDITION

*

	LOCAL	LBRFMSG,ERFWS1,FBSYMSG,ERFWS2	
ERRFWS	AND,R8	X1FFFF	ADDRESS MASK
	STW,R10	IOERCODE	SAVE MONITOR ERROR CODE-SUBCODE
	CI,R8	CALOPWS1+1	CHECK IF THIS IS OPEN FOR TESTOLDF
	BNE	ERFWS1	NO
	LI,R9	3	YES-CHECK IF 'FILE NOT FOUND'
	CB,R9	R10	
	BE	SAVEOPEN	GOOD-NORMAL IN THIS CASE
ERLIBREF	LI,R8	IDFILREF	= ERR I.D. FOR 'BAD FILE REF'.
	B	CMDERR	CMD ERROR EXIT.
ERFWS1	CI,R8	CALOPWS2+1	CHECK IF THIS IS OPEN FOR SAVE
	BNE	ERFWS2	NO
	LI,R9	X'1402'	'CHECK IF BUSY'
	CH,R9	R10	
	BNE	IOERRCM	I/O ERR IN CMD.
	LI,R8	IDFILBSY	= ERR I.D. FOR 'FILE IN USE'.
	B	CMDERR	CMD ERROR EXIT.
ERFWS2	CI,R8	CALOPWS3+1	CHECK IF OPEN FOR LOAD
	BNE	ERFWS3	NO
ERXWSNF	LI,R6	BCMDEERR	YES-
	LI,R8	IDWSNOTF	= ERR I.D. FOR 'WS NOT FOUND'.
	B	CLOSV	

*

* ERFFS2-REACHED BY 'TRYLOAD' IF OPEN OK AND RECORD LONGER THAN

*

IDRECSIZ-(WHICH IT SHOULD BE)

*

ERFWS3	CI, R8	CALRDWSI+1	CHK IF READ OF ID REC	
	BNE	ERFWS4	NO	
	LI, R9	7	CHECK IF RECORD TOO BIG	
	CB, R9	R10		
	BNE	ERXWSNF	NO	
	LI, R4	IDBUF	YES-OK-GET ID RECORD ADDRESS	
	LW, R9	6, R4	'APLVERSN' FOR FILE	
	CW, R9	APLVERSN	CHECK IF APL FILE	
	BNE	CLOSV	NO-CLOSE AND ERROR EXIT	
	LW, R9	LODYN	COMPUTE AND	
	SW, R9	4, R4	SAVE	
	STW, R9	WSOFFSET	WS DYNAMIC OFFSET	
	LI, R6	BCMDERR	SET NEW ERROR EXIT	
	B	*R14	RETURN	
ERFWS4	CI, R8	CALOPWS4+1	CHECK IF OPEN FOR 'DROP'	
	BNE	ERFWS5	NO	
	LI, R8	IDWSNOTF	= ERR I.D. FOR 'WS NOT FOUND'.	
BCMDERR	B	CMDERR	*BREF* NO SCORE	
ERFWS5	RES	0		
	CI, R8	CALWRWS+1	CHECK IF WS WRITE	
	BE	ERRSAVE	YES-ABORT THE SAVE.	
	CI, R8	CALRDWS+1	CHECK IF WS READ	
	BNE	ERFWS0	NO-ASSUME INITIALIZATION PROBLEM	
	LI, R9	7	CHECK IF RECORD TOO BIG	
	CB, R9	R10		
	BNE	IOERR	NO-QUIT	
	CI, R14	RDWRLOOP	CHECK IF READ FOR COPY	
	BE	ERRFWS6	YES-OK	
	B	IOERR	NO-QUIT	
* ERBADWS-ERROR RETURN FROM WS CHECKER ON LOAD,SAVE,OR COPY				
* ERBADWS RES 0				
CALSNAPW	CAL1, 3	FPTSNAPW	GENERATE SNAPSHOT	
	B	BCLEAR	GO TO CLEAR.	
* ERRSAVE-ERROR DURING)SAVE AFTER SUCCESSFUL OPEN.				
* ERRCOPY-ERROR DURING FILE COPY ON OPEN OF TFILE				
* ERRCOPY LI, R5 F:TF DCB				
	B	ERRSAVE1		
ERRSAVE	LI, R4	0	MAY BE SAVE FOR AUTOSTART	
	XW, R4	CURRCS		
	BEZ	ERRSAVE0	NO.	
	AI, R4	-2	YES, DEREf. CODESTRING.	
	BAL, R7	DREF		
ERRSAVE0	LI, R5	F:WS	DCB	
ERRSAVE1	BAL, R14	XWLOCGLB	RE-EXCHANGE LOCALS & GLOBALS.	
	BAL, R6	CLOSR	CLOSE AND RELEASE.	
	LI, R8	IDFILSPC	= ERR I.D. FOR 'FILE SPACE TOO LOW'.	
	LB, R1	IOERCODE		
	CI, R1	X'57'	CK FOR ERROR CODE 57...	
	BE	CMDERR	YES, FILE SPACE TOO LOW.	
IOERRCM	LI, R8	IDIOERR	= ERR I.D. FOR 'I/O ERR'.	
	B	CMDERR		
* SYSTERR-SYSTEM ERROR ROUTINE				
* SYSTERR RES 0				
	STW, R15	XSEGBRK	RESET IMMEDIATE BRK RSP FLAG	14-00009
	LW, R0	R15	SAVE R15 IN R0	
	BAL, R15	CHKTERM	SWITCH IF 4013	
CALSNAPS	CAL1, 3	FPTSNAPS	GENERATE SNAPSHOT	
	B	ERREXIT	CRASH	
* IOERR-UNRESOLVABLE I/O ERROR-NOT ON F:OUT				
* IOERR RES 0				
	STW, R8	XSEGBRK	RESET IMMEDIATE BRK RSP FLAG	14-00011
	BAL, R15	CHKTERM	SWITCH IF 4013	
CALSNAPI	CAL1, 3	FPTSNAPI	GENERATE SNAPSHOT	
	B	ERREXIT	CRASH	
* ERRRDAPL-ERROR ON APL READ (M:APL)				
* ERRRDAPL LB, R9 R10 ERROR CODE				
	CI, R9	7	CHECK FOR PARITY ERROR	
	BE	CALRDAPL+1	YES-USE THE DATA	

```

        CI,R9      6                CHECK FOR EOF
        BNE        IOERR            NO-IO ERROR
        MTW,0      ON%OFF           YES-CHECK IF ON-LINE
        BGZ        CALRDAPL+1      ON-LINE,CONTINUE
BOFFH   B          UTSIOFFH * BREF * RETURN TO MONITOR
*
* ERREXIT-CRASH EXIT
*
ERREXIT LI,R15     -1                SET FLAG TO ABORT
        STW,R15    HOLDFLG
        B          CALXCOFF
TELEXIT RES        0
        BAL,R15    CHKTERM          SWITCH IF 4013
CALXCOFF CAL1,8    FPTXCOFF         RESET EXIT CONTROL
CALRSTRM CAL1,8    FPTRSTRM         RESET TERMINAL MODE
        MTW,0      HOLDFLG         TEST FOR ABORT
        BGEZ       CALEXIT          NO
        MTW,0      ON%OFF           YES-TEST IF OFF-LINE
        BGZ        CALEXIT          NO-CONTINUE
CALERR  CAL1,9     2                ERROR RETURN TO MONITOR
CALEXIT CAL1,9     1                RETURN TO TEL
*
*
CHKTERM XW,R15     TERMTYPE
        AI,R15     -13              CHECK IF TERMINAL TYPE 13
        BNEZ       *TERMTYPE        NO
        LI,R15     X'1B0F'          YES
        STH,R15    GRAFBUF
CAL13TTY CAL1,1    FPTWR2           SWITCH 4013 TO TTY MODE
        B          *TERMTYPE        EXIT
        PAGE
*
* ERRFF-FILE IO MONITOR ERROR PROCESSOR
*
* ERROR PROCESSOR RESIDES IN APLFIO MODULE
*
ERRFF   B          FIOERR           IN MODULE APLFIO
ABNFF   EQU        ERRFF           ABN-OPEN
*
* MODTBL-TABLE OF ASSEMBLY MODULE ORIGINS
* USED IN CRASH ANALYSIS
*
MODTBLS RES        0
        GEN,8,24   X'01',ACQCONS@
        GEN,8,24   X'02',CONSTS@
        GEN,8,24   X'03',CONTEXT@
        GEN,8,24   X'04',CS@
        GEN,8,24   X'05',INTRINS@
        GEN,8,24   X'06',WMAQ@
        GEN,8,24   X'07',LIB@
        GEN,8,24   X'08',OPR@
        GEN,8,24   X'09',MIX@
        GEN,8,24   X'10',INDX@
        GEN,8,24   X'11',EVAL@
        GEN,8,24   X'12',APLINP@
        GEN,8,24   X'13',APLOUT@
        GEN,8,24   X'14',APLUTSI@
        GEN,8,24   X'15',FUNDEF@
        GEN,8,24   X'16',WSCOPY@
        GEN,8,24   X'17',CMD@
        GEN,8,24   X'18',APLUTSC@
        GEN,8,24   X'19',CODEXEQ@
        GEN,8,24   X'20',APLFIO@
        GEN,8,24   X'21',APLFMT@
        GEN,8,24   X'22',MAT@
        GEN,8,24   X'23',RCMD@
        GEN,8,24   X'24',SCMD@
        GEN,8,24   X'25',ERROR@
        GEN,8,24   X'26',GRAF@
        GEN,8,24   X'27',UTSG@
MODTBLE EQU        %-1
        END        START

```

	TITLE	'FUNDEF-B00,08/22/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	
FUNDEF@	RES	0	CONTROL CARDS NOT PRINTED.
			ORIGIN OF FUNCTION DEFINITION MODULE

*	REF'S	AND	DEF'S	
*				
	DEF		FUNDEF@	= START OF FUNDEF MODULE.
*	DEF		DELCK	ENTERED FROM CODESTRINGER WHEN IT
				ACQUIRES A DEL OR LOCKED-DEL.
	DEF		FIHANDLR	FUNCTION-INPUT HANDLER.
	DEF		REXPROMPT	RE-PROMPTS WITH LATEST LINE NUMBER.
*	DEF		MERGECOL	WHEN CALLING 'SHOWFL', MERGECOL IS
*				SET TO AN EDIT-DIRECTIVE COLUMN-
*				VALUE OR NEGATIVE IF NOT AN EDIT.
*				WHEN CALLING 'INP4MERG', MERGECOL
*				IS 1 LESS THAN THE 1ST COLUMN AT
				WHICH MERGING BEGINS.
*	DEF		VISIMAGE	BUFFER TO HOLD VISUAL IMAGE OF A
				LINE TO BE EDITED.
	DEF		FERASECK	CKS FOR AN ERASE OF CURRENT OPEN FN.
	DEF		FORCLOSE	FORCES A CLOSE OF THE OPEN FUNCTION.
*	DEF		USEBADFL	USE FUNC.LINE CODESTRING EVEN THOUGH
				IT ENDS ON A LINE-SCAN ERROR.
	DEF		RFSIDAM	RESUME AFTER SI DAMAGE DISPLAY.
	DEF		RFUNDEF	RESUME AFTER ERRFDWS,ERRFDSYM,SHOWFL
SPACE 3				
*				REFS TO PROCEDURE:
	REF		ACQNXCC	ACQ NEXT CHAR & ITS CODE.
	REF		ACQXNB	ACQ NEXT NON-BLANK & IS CODE.
	REF		ACQNB	ACQ NON-BLANK & ITS CODE.
	REF		ACQNAME	ACQ A NAME.
	REF		CSN	START CODESTRINGING AT N-TH CHAR.
	REF		CSZ	FINISH CURRENT CODESTRING.
	REF		ERLSCAN	ENTER CODESTRINGER FOR LINE-SCAN ERR
	REF		DREF	DE-REFERENCE A DATA BLOCK.
	REF		MAYDREF	DE-REF. DATA BLK IF DB PTR IS NZ.
*	REF		ALOCNONX	ALLOC. DATA BLK FOR NEW FUNCTION
*				DESCRIPTOR & FOR ANY LABEL DATA
				BLKS THAT WILL BE NEEDED.
	REF		CTEST	ATTEMPTS TO GET MORE EXEC. STACK.
	REF		GENCHAR	PUTS CHAR OR ITS EQUIV. IN IMAGE BUF
	REF		GENNAME	PUTS (VISUAL) NAME IN IMAGE BUFFER.
*	REF		SHOWFL	DISPLAYS IMAGE AFTER DE-CODESTRING,
				IF NECESSARY.
	REF		INPF	PROMPTS & REQUESTS FUNCTION INPUT.
	REF		INP4MERG	DISPLAYS EDITED LINE FOR MERGING.
	REF		CMDEXIT	EXITS ACCORDING TO MODE & ERR-COND. U15-0004
*	REF		SIDR	ASKS FOR 'GO' RESPONSE WHEN
				SI DAMAGE WILL OCCUR.
	REF		SIDAMF	DISPLAYS -- SI DAMAGE -- IN FUNDEF.
	REF		ERRFDWS	DISPLAYS -- WS FULL -- DUE TO FUNDEF
	REF		ERRFDSYM	DISPLAYS -- SYMBOL TABLE FULL.
*	REF		ERRDEFN	DISPLAYS -- DEFN ERROR.
				REFS TO CONTEXT:
	REF		FDBLOCK	FUNCTION DEFINITION FIXED CONTEXT
	REF		FDTEMPS	FUNCTION DEFINITION TEMP AREA.
	REF		ERRORID	SET NZ IF ERROR DETECTED.
*	REF		OLDMODE	MODE WHEN THE FUNC OPEN OCCURRED: U15-0007
*				1 = DIRECT INPUT MODE. U15-0008
*				2 = EVALUATED INPUT MODE. U15-0009
*				2 + E-FLAG = 'EXECUTE' MODE. U15-0010
*				U15-0011
	REF		MODE	MODE OF EXECUTION:
*				0 = FUNCTION DEFINITION MODE.
*				1 = DIRECT INPUT MODE.
*				2 = EVALUATED INPUT MODE.
*				NEGATIVE = FORCED CLOSE MODE.
	REF		IMAGE	BUFFER FOR INPUT & PROMPTS.
	REF		IMAGES	= # LINES USED TO PERFORM A DISPLAY.
	REF		INBUF	USED AS BUILD-BUFFER DURING EDIT.
	REF		EDITCK	= NZ IF 'DELETE SEQUENCE' INDICATED.
	REF		CURRCS	PTS AT OFFSET WD OF CURRENT CODESTRG
*	REF		FDEFPTR	PTS AT 'OLD' FUNCTION DESCRIPTOR IF
*				A RE-OPEN OCCURS & IS NOT WIPED
				OUT BY CHANGING THE NEW HEADER.
	REF		RESULT	PTS AT NEW FUNCTION DESCRIPTOR.

	REF	TOPOSTAK	CURRENT TOP OF EXECUTION STACK.
	REF	STATEPTR	TOP STATE-ENTRY IN EXECUTION STACK.
	REF	STKLIMIT	CURRENT LIMIT FOR EXECUTION STACK.
	REF	LOCNEED	NEW STACK LOC NEEDED WHEN OVER LIMIT
	REF	SYMT	PTS AT 1ST WD OF SYMBOL TABLE
	REF	CONSTBUF	BUFFER USED TO HOLD NAME PTRS GIVEN
*			VIA AN)ERASE COMMAND.
	REF	OPENFN	FUNC. NAME PTR AT FORCED CLOSE.
	REF	BREAKFLG	= NEG. IF AUTOMATIC CONTINUE CAUSES
*			A FORCED CLOSE.
*			REFS TO CONSTANTS:
	REF	XFFFF	= X'FFFF'
	REF	X1FFFF	= X'1FFFF'
	REF	BLANKS	WD OF BLANKS.
	REF	ZEROZERO	DBLWD OF ZERO.
	REF	FUNTPES	RANGE OF FUNCTION TYPES.
	REF	BITPOS	32-WD TBL OF BITS (BITPOS-K CONTAINS
			A WD HAVING A 1 ONLY IN BIT POS K)
BLOC	SET	0	*****
BLOCK	CNAME		*
	PROC		* PROC TO 'ALLOCATE' IN FDBLOCK. *
LF	EQU	FDBLOCK+BLOC	*
BLOC	SET	BLOC+AF(1)	*
	PEND		*****
SAVEDEL	BLOCK	1	SAVE OPENING DEL.
SAVECDEL	BLOCK	1	SAVE CLOSING DEL.
FUNSAVE	BLOCK	3	SAVE REGS.
GLOBLOC	BLOCK	1	LOC.OF GLOBAL REF TO RE-OPENED FUN.
LINKFD	BLOCK	1	LINK FOR USE IN FUNCTION DEFN MODE.
HILINE	BLOCK	1	PTS AT HI BOUNDARY LINE-CHAIN ENTRY.
LINVAL	BLOCK	1	CURRENT LINE VALUE.
FBLK	BLOCK	5	FUNCTION NAME PTRS BLOCK + 1.
NFL	EQU	FBLK+4	HOLDS NO.OF FUNC.LINES FOR NEW FUNC.
*			NOTE-- THE NLCLS & NLBLS WD MUST BE THE
*			NEXT WD AFTER NFL.
NLCLS	BLOCK	1	HI HALF HOLDS # LOCALS FOR NEW FUNC.
NLBLS	EQU	NLCLS	LO HALF HOLDS # LABELS FOR NEW FUNC.
LOCALS	BLOCK	32	MAX.POSS. # LOCAL NAME PTRS @ 2/WD.
SPACE 2			
TLOC	SET	0	*****
TEMP	CNAME		*
	PROC		* PROC TO 'ALLOCATE' IN FDTEMPS. *
LF	EQU	FDTEMPS+TLOC	* (FDTEMPS IS ON A DBLWD BOUND) *
TLOC	SET	TLOC+AF(1)	*
	PEND		*****
*			
*	TEMPS	WHILE OPENING:	
*			
NAMECNT	TEMP	1	NAME COUNTER.
TFBLK	TEMP	4	TEMP FUNCTION NAME PTRS BLOCK.
TNLCLS	TEMP	1	TEMP NO.OF LOCALS FOR NEW HDR.
TLOCALS	TEMP	32	HOLDS TEMP LOCAL NAME PTRS (HALFWDS)
*			
*	TEMPS	FOR DIRECTIVE HANDLING:	
*			
TLOC	SET	0	(OVERLAY OPENING TEMPS)
LINKHIRE	TEMP	2	DBLWD TO HOLD LINKAGE VALUES.
MERGECOL	TEMP	1	FLAG & COLUMN INDICATOR FOR EDIT.
STARTLV	TEMP	1	HOLDS 1ST LINE VALUE ON A DISPLAY-N.
VISIMAGE	TEMP	33	VISUAL IMAGE FOR EDITING.
*			
*	TEMPS	FOR CLOSING:	
*			
TLOC	SET	0	(OVERLAY DIRECTIVE HANDLING TEMPS)
NEWFTYPE	TEMP	1	HOLDS NEW FUNCTION'S TYPE.
XREFLOC	TEMP	1	LOC FOR REF-EXCHANGE PROCEDURE.
PAGE			
*			
*	STANDARD EQU'S		
*		REGISTERS	
R1	EQU	1	
R2	EQU	2	
R3	EQU	3	
R4	EQU	4	
R5	EQU	5	
R6	EQU	6	
R7	EQU	7	
R8	EQU	8	


```

R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
*   EBCDIC CHARACTERS
*
NEWLINE EQU      X'15'      END-OF-STMT.
QUAD     EQU      X'53'      QUAD.
DEL      EQU      X'7F'      DEL.
LBRACKET EQU      X'B4'      LEFT BRACKET.
RBRACKET EQU      X'B5'      RIGHT BRACKET.
LFARROW  EQU      X'FD'      LEFT ARROW.
*
*   CODESTRING VALUES:
*
NAMECODE EQU      23          ORDINARY NAME.
BOSCODE  EQU      37          BEGINNING-OF-STMT.
LASTCSV  EQU      138         LAST CODESTRING VALUE.
*
*   OFFSETS IN FUNCTION DESCRIPTOR:
*
XSIZEOFF EQU      2          EXECUTION STACK SIZE.
RESOFF   EQU      XSIZEOFF+1 RESULT NAME PTR.
FBLKSIZE EQU      4          (4 NAME PTRS IN A FUNCTION BLOCK)
FNOFF    EQU      RESOFF+2    FUNCTION NAME PTR.
NFLOFF   EQU      RESOFF+FBLKSIZE NO.OF FUNCTION LINES.
*
*   OTHER EQU'S
*
CATLC    EQU      11          LINE-CHAIN CATEGORY OF EXEC.STACK.
EFLAG    EQU      X'10000'    INDICATES 'EXECUTE' MODE.          U15-0014
LOCKFLAG EQU      X'10000'    LOCKED-FUNCTION FLAG.
PENDFLAG EQU      X'8000'     PENDENT STATE-ENTRY FLAG.
NFLAG    EQU      X'20000'    NON-ASSIGNABLE FLAG FOR LABEL
                                REFERENT INDICATORS.
STDFDSIZ EQU FBLKSIZE+3      STANDARD FUNCTION DESCRIPTOR SIZE (MINIMUM),
                                COVERS 4-WD FUNCTION BLK + XSIZE WD + NFL WD +
                                LOCALS & LABELS COUNT WD.
*
*   DOUBLEWORD CONSTANTS:
*
DELDEL   DATA    129,130     CODESTRING VALUES FOR DEL & LOCK DEL
*
*   CONSTANTS:
*
LCENTRY  DATA    CATLC**24    'DEAD' (SO FAR) LINE-CHAIN CATEGORY.
MAXLINV  DATA    X'9999999'   MAX LINE VALUE = 9999.999.
LBLDBHDR DATA    X'03000004'  LABEL'S DATA BLK HDR --
                                INTEGER, SCALAR OF SIZE 4-WDS.
*
PAGE
*****
*
*   HNAMEX -- ENTRY FOR HNAME THAT ACQUIRES THE NEXT NON-BLANK.
*   HNAME  -- ACQUIRES NAMES FOR A FUNCTION HEADER.
*   REGS:  R7 -- LINK.  EXIT DEPENDS ON NON-BLANK AFTER THE NAME:
*           0,R7-- NEWLINE, DEL, OR LOCKED DEL.
*           1,R7-- OTHER (E.G. ANOTHER NAME, ETC.).
*           2,R7-- SEMICOLON.
*   R1 -- (ENTRY) PTS AT NON-BLANK CHAR, SHOULD START NAME
*   (EXIT) PTS AT NON-BLANK CHAR AFTER THE NAME.
*   R2 -- (ENTRY) THE CHAR. HOPEFULLY STARTING A NAME.
*   (EXIT) THE CHAR AFTER THE NAME (NON-BLANK).
*   R3 -- (ENTRY) CODE FOR START CHAR.
*   (EXIT) CODE FOR TERMINATING CHAR.
*   R4,R12,R13 ARE VOLATILE, BUT SEE ALSO 'ACQNAME'.
*
HNAMEX   BAL,R4    ACQNXNB      ACQ. NEXT NON-BLANK.
HNAME    STW,R7    LINKFD       SAVE LINK.
          CI,R3     LASTCSV      CK FOR NAME-START CODE...
          BLE       ERDEFN       NO -- DEFN ERROR.
          BAL,R12   ACQNAME      YES, ACQUIRE THE NAME.
          B         SYMFULL      (SYM TBL FULL RETURN)
          B         WSFULL       (WS FULL RETURN, DUE TO LONG NAME)
          AI,R13    -NAMECODE    CK FOR ORDINARY NAME...

```

```

        BNEZ      ERDEFN      STOP OR TRACE NAME -- DEFN ERROR.
        LW,R7     LINKFD      RESTORE LINK.
        CI,R2     ';'
        BE        2,R7        SEMICOLON EXIT.
        CI,R2     NEWLINE
        BE        0,R7        NEW LINE EXIT.
        CLM,R3    DELDEL
        BCS,9     1,R7        OTHER EXIT.
        B         0,R7        DEL OR LOCKED DEL EXIT.

PAGE
*****
*
* RAISE3 -- RAISE EXECUTION STACK BY 3 WORDS.
* REGS:  R10 -- LINK, EXIT VIA *R10.
*        R1  -- (EXIT) PTS AT NEW TOP OF STACK.
*        R8 IS VOLATILE.
*
RAISE3  LW,R1     TOPOSTAK      PT AT CURRENT TOP OF STACK.
        AI,R1     -3            RAISE IT 3 WDS.
        CW,R1     STKLIMIT      DOES THIS HIT STACK LIMIT...
        BG        RAISED       NO, OK.
        STW,R1    LOCNEED       YES, SAVE LOCATION NEEDED.
        AI,R1     3            RESTORE STACK PTR IN CASE WS FULL   U15-0016
        BAL,R8    CTEST         GET MORE COMMON, IF POSSIBLE.
        B         WSFULL       NO LUCK -- WS FULL.
RAISED  STW,R1    TOPOSTAK      PT AT NEW TOP OF STACK.
        B         *R10         EXIT.

PAGE
*****
*
* INITLC -- INITIALIZES THE LINE-CHAIN, SETTING 'HILINE'.
* REGS:  R14 -- LINK, EXIT VIA *R14.
*        R10,R11,R12,R13 ARE VOLATILE.
*
INITLC  BAL,R10   RAISE3        RAISE STACK 3 WDS.
        LW,R11    LCENTRY       = EMPTY LINE-CHAIN CATEGORY.
        LI,R12    X'FFFD'       = HALFWO OF MINUS 3 (TO 0 ENTRY).
        LW,R13    BITPOS-3      LINE-VALUE BOUND ( > MAXLINV).
        LCI       3
        STM,R11   0,R1          SET HI BOUNDARY OF LINE-CHAIN.
        STW,R1    HILINE        SET PTR TO HI LINE-CHAIN ENTRY.
        BAL,R10   RAISE3
        LD,R12    ZEROZERO      0 OFFSET, 0 LINE-VALUE.
        LCI       3
        STM,R11   0,R1          SET LO BOUNDARY OF LINE-CHAIN.
        B         *R14         EXIT

PAGE
*****
*
* GFNREF -- FINDS REFERENT PTR (GLOBAL) FOR FUNCTION NAME.
* REGS:  R14 -- LINK, EXIT VIA *R14
*        R6 -- (ENTRY) NAME PTR FOR FUNCTION NAME.
*        R4 -- (EXIT) REFERENT PTR (POSSIBLY ZERO), (COND.CODE)
*        R5 -- (EXIT) ADDRESS MASK X'1FFFF'.
*        R7 -- (EXIT) LOC OF GLOBAL REFERENCE FOR NAME GIVEN.
*        R13 IS VOLATILE.
*
GFNREF  LI,R5     X'1FFFF'
        LI,R4     0
        LW,R7     SYMT          PT AT SYM TBL ENTRY FOR THIS NAME.
        AW,R7     R6
        LW,R13    0,R7          TEST FOR GLOBAL REF IN SYM TBL...
        BGEZ      GFNSHAD      NO, IT IS SHADOWED.
GFNREFOK LS,R4     0,R7          YES, EXTRACT REFERENT PTR.
        B         *R14         EXIT WITH THAT CONDITION CODE.
GFNSHAD  LW,R7     TOPOSTAK      PT AT TOP OF EXECUTION STACK.
GFNSHADT AI,R7     1            PT AT NEXT ENTRY.
        LW,R13    0,R7          LOOK FOR G-BIT THAT IS ON...
        BGEZ      GFNSHADT     NO.
        CW,R6     -1,R7        YES, LOOK BACK AT ITS NAME PTR...
        BNE       GFNSHADT     WRONG NAME, TRY AGAIN.
        B         GFNREFOK     BINGO.

PAGE
*****
*
* LCSEARCH -- SEARCHES THE LINE-CHAIN UNTIL IT FINDS THE GIVEN LINE
* VALUE OR THE NEXT LOWEST LINE VALUE IN THE CHAIN.
* REGS:  R5 -- LINK; EXIT VIA 0,R5 IF NO MATCH

```

```

*                                     1,R5 IF MATCH.                                     *
* R12-- (ENTRY) LINE VALUE TO SEARCH FOR.                                         *
* R7 -- (EXIT) LOC OF CHAIN ENTRY REACHED.                                         *
* R6 -- (EXIT) WD OFFSET FROM PRIOR (HIGHER LINE VALUE) CHAIN ENTRY.               *
* R2 -- (EXIT) = 3, I.E. THE HALFWORD ADD-ON NEEDED TO FIND THE CHAIN VALUE WITHIN A *
* LINE-CHAIN ENTRY.                                                                *
* NO REGS ARE VOLATILE.                                                            *
*
LCSEARCH LI,R2      3                                     PT AT MAX+ LINE-CHAIN ENTRY.
        LW,R7      HILINE                                GET OFFSET TO NEXT LOWER LC ENTRY.
LOW      LH,R6      *R7,R2                               PT AT THAT ENTRY.
        AW,R7      R6                                   CK FOR MATCHING LINE VALUE...
        CW,R12     2,R7                                GIVEN LINE VALUE LOWER THAN THIS.
        BL        LOW                                  MATCH.
        BE        1,R5                                NO MATCH, BUT STOP SEARCH; WE'VE
        BG        0,R5                                FOUND A LINE VALUE LOWER THAN
*                                                THE GIVEN ONE.
*
PAGE
*****
*
* NXTFLINE -- CALCULATES THE NEXT FUNCTION LINE VALUE (FOR PROMPTING).
* REGS:  R12-- LINK, EXIT VIA *R12.
*        R6 -- (EXIT) CONTAINS THE NEW LINE VALUE.
*        R3,R4,&R7 ARE VOLATILE.
*
NXTFLINE LW,R6      LINVAL      GET OLD LINE VALUE.
        LI,R4      4            = # HEX DIGITS FROM RT. TO 1'S POS.
        LI,R3      0            R3 IS A SHIFT COUNTER.
CARRY    LI,R7      0            CLEAR R7 FOR NEXT SHIFT.
DTAIL    AI,R3      4            ACCUM SHIFT COUNT.
        SLD,R6     -4           SHIFT TRAILING DIGIT INTO TOP OF R7.
        AI,R7      0            TEST FOR ZERO...
        BNEZ      INCR          NO, INCREMENT AT THAT POSITION.
        BDR,R4     DTAIL        YES, SKIP TAILING ZEROS.
INCR     AW,R7      BITPOS-3    BUMP THE HEX DIGIT.
        LB,R4      R7           TEST FOR BUMPED TO 10, AND IF SO
        AI,R4      X'FFF60'     CLEAR R4 TO AVOID ANY MORE SKIPS.
        BEZ        CARRY        YES, CLEAR AND INCREMENT NEXT POS.
        SLD,R6     0,R3         NO, DE-SHIFT.
        CW,R6      MAXLINV      PERCHANCE CARRIED INTO LFMOST HEXPOS
        BLE        LVRDY        NO, LINE VALUE IS OK.
        LW,R6      MAXLINV      YES, SUBSTITUTE MAX ALLOWED VALUE.
LVRDY    STW,R6     LINVAL      SAVE THE LINE VALUE.
        B         *R12
*
PAGE
*****
*
* GENFPRMT -- GENERATES A FUNCTION LINE NO. PROMPT IN IMAGE BUFFER.
* REGS:  R12-- LINK, EXIT VIA *R12.
*        R3 -- (EXIT) CONTAINS NO.OF CHARS GENERATED IN IMAGE.
*        R4,R6,R7,&R13 ARE VOLATILE, BUT SEE ALSO 'GENCHAR'.
*
GENFPRMT LI,R3      0            = # CHARS IN IMAGE.
        LI,R7      LBRACKET     GEN. LEFT BRACKET.
        BAL,R13    GENCHAR
        LW,R6      LINVAL      GET THE LINE VALUE.
        LI,R4      1+4         = # HEX DIGITS IN INTEGER PART.
        LI,R7      0            CLEAR R7 FOR LEADING ZERO TESTS.
DLEAD    SCD,R6     4            SHIFT HEX DIGIT INTO LO END OF R7.
        AI,R7      0            TEST FOR ZERO...
        BNEZ      GENDIGIT      NO, START DIGIT GENERATION.
        BDR,R4     DLEAD        YES, SKIP TILL END OF INTG PART.
        LI,R4      1            OK, USE INTEGER ANYWAY--I.E. GEN. 0.
GENDIGIT AI,R7      '0'         CONVERT TO EBCDIC INTEGER.
GENPT    BAL,R13    GENCHAR     GENERATE THE CHARACTER.
        LI,R7      0            CLEAR R7 AGAIN.
        AI,R6      0            TEST FOR FURTHER NON-ZERO DIGITS...
        BNEZ      GENMORE       YES, MORE TO GENERATE.
        CI,R4      1            NO, IS INTEGER PART GENERATED...
        BLE        GENRB        YES -- GENERATE THE RT. BRACKET.
GENMORE  SCD,R6     4            SHIFT NEXT DIGIT INTO LO END OF R7.
        AI,R4      -1           DO WE NEED A DEC.PT. FIRST...
        BNEZ      GENDIGIT      NO.
        SCD,R6     -4           YES, SHIFT BACK TILL DEC.PT HAS
        LI,R7      ' '          BEEN GENERATED.

```

GENRB	B	GENPT	GEN. THE RIGHT BRACKET.
	LI, R7	RBRACKET	
GENBLANK	BAL, R13	GENCHAR	
	LI, R7	' '	GEN. A BLANK.
	BAL, R13	GENCHAR	
	CI, R3	6	BLANK OUT TO COLUMN 6, IF NECESSARY.
	BL	GENBLANK	
	B	*R12	
PAGE			
DELCK	LW, R5	MODE	IS THIS FUNC.DEFN. MODE ALREADY...
	BEZ	CLOSEQ	YES -- PROBABLY A CLOSING DEL.
	STW, R2	SAVEDEL	NO, SAVE OPENING DEL.
	STW, R5	OLDMODE	(SAVE CURRENT MODE). U15-0018
	CI, R7	2	IS CODESTR.OFFSET AT 'BEGIN-O-STMT'.
	BNE	ERLSCAN	NO -- LINE-SCAN ERROR.
	LI, R4	0	YES, THROW CODESTRING AWAY, I.E.
	STW, R4	ERRORID	(CLEAR ERROR I.D.)
	STW, R4	GLOBLOC	(CLEAR GLOBAL REF LOC.)
	XW, R4	CURRCS	
	AI, R4	-2	
	BAL, R7	DREF	DE-REFERENCE IT.
	AI, R5	-1	IS THIS DIRECT INPUT MODE...
	BEZ	CLROPEN	YES. U15-0022
	LI, R5	EFLAG	NO, QUAD-STATE MODE. EXTRACT U15-0023
	AND, R5	*STATEPTR	E-FLAG ('EXECUTE' VS EVAL-INPUT) U15-0024
	STS, R5	OLDMODE	AND MARK OLD MODE SIMILARLY. U15-0025
CLROPEN	BAL, R4	ACQXNB	ACQ NEXT NON-BLANK AFTER DEL.
ACQHDR	LI, R5	-1	(THE CLEAR NAME PTR INDICATOR)
	LI, R4	-FBLKSIZE	= - MAX # STANDARD NAMES IN HEADER.
CLRTFBLK	STW, R5	TFBLK+FBLKSIZE, R4	CLEAR TEMP STANDARD NAME BLOCK.
	BIR, R4	CLRTFBLK	
	LI, R5	0	
	STW, R5	TNLCLS	CLEAR TEMP # LOCALS.
	STW, R5	NAMECNT	CLEAR NAME COUNTER.
	BAL, R7	HNAME	@ ACQ HDR NAME & TERMINATION:
	B	ONLY1	@ NEWLINE OR DEL -- ONLY 1 NAME.
	B	LBQ	@ OTHER (LEFT BRACKET, ARROW, NAME...)
	B	SETFN	@ SEMICOLON -- SET FUNCTION NAME.
LBQ	CI, R2	LBRACKET	LEFT BRACKET...
	BNE	ARROWQ	NO (WE SHOULD BE ON A NEW HDR).
	LW, R7	MODE	YES, THIS IS A REOPEN, NO DOUBT...
	BNEZ	TESTFN	WELL, MODE OK -- TEST FUN NAME.
	B	ERDEFN	OOPS -- DEFN ERROR.
ARROWQ	CI, R2	LFARROW	LEFT ARROW...
	BNE	SETANAME	NO (PROBABLY ANOTHER NAME).
	STW, R6	TFBLK	YES, SET TEMP RESULT NAME PTR.
	BAL, R7	HNAMEX	@ ACQ NEXT HDR NAME & TERMINATION:
	B	SETFN	@ NEWLINE OR DEL -- SET FUNC. NAME.
	B	SETANAME	@ OTHER (BETTER BE ANOTHER NAME).
	B	SETFN	@ SEMICOLON -- SET FUNCTION NAME.
SETANAME	MTW, 1	NAMECNT	INCR NAME COUNTER.
	LW, R7	NAMECNT	
	STW, R6	TFBLK, R7	STORE THIS TEMP NAME PTR.
	AI, R7	-2	WILL NEXT NAME BE ONE TOO MANY...
	BGZ	ERDEFN	YES -- DEFN ERROR.
	BAL, R7	HNAME	@ NO, ACQ HDR NAME & TERMINATION:
	B	DETFNAME	@ NEWLINE OR DEL.
	B	SETANAME	@ OTHER (BETTER BE ANOTHER NAME).
*			@ SEMICOLON.
*			@ DETERMINE WHICH IS THE FUNCTION NAME
DETFNAME	STW, R6	TFBLK+3	@ STORE TEMP LAST STD. NAME PTR, IT
*			HAS TO BE THE RIGHT DUMMY.
	LW, R7	NAMECNT	WAS THERE 2 PRIOR NAMES BESIDES
	AI, R7	-2	ANY RESULT NAME...
	BEZ	NEWFQ	YES, DYADIC FUNCTION.
	LI, R6	-1	NO, MONADIC FUNCTION.
SETFN	XW, R6	TFBLK+1	MOVE THE 1ST NAME (IT'S FUNC. NAME).
NEWFQ	STW, R6	TFBLK+2	PUT TEMP FUN NAME PTR IN ITS PLACE.
	LW, R6	TFBLK+2	GET THE NEW FUNCTION NAME PTR.
	BAL, R14	GFNREF	IS ITS GLOBAL REFERENT IN USE NOW...
	BEZ	LCLQ	NO.
	CW, R4	FDEFPTR	YES, IS THIS A NEW HDR FOR REOPEN
*			FUNCTION...
	BE	LCLQ	YEP, CK FOR LOCALS IN IT.
*	CW, R7	GLOBLOC	DOES NEW HDR NAME ORIGINALLY OPENED
			FUNCTION...
	BNE	ERDEFN	UH-UH -- DEFN ERROR.
	STW, R4	FDEFPTR	UH-HUH, SAVE PTR TO FUNC.DESCR.

LCLQ	MTW,1 CI,R2 BNE BAL,R7 B B	1,R4 '.' SETFBLK HNAMEX LCL ERDEFN	AGAIN & INCR ITS REF-COUNT AGAIN. LOCAL COMING UP... NO -- SET THE NEW HDR BLOCK. @ YES, ACQ NEXT LOCAL & TERMINATION: @ NEWLINE OR DEL. @ OTHER -- DEFN ERROR. @ SEMICOLON.
*			
LCL	LH,R7 STH,R6 MTH,1 B	TNLCLS TLOCALS,R7 TNLCLS LCLQ	@ GET TEMP LOCALS COUNT. SAVE TEMP LOCAL NAME PTR. INCR TEMP LOCALS COUNT.
ONLY1	LW,R7 BEZ	MODE SETFN	ARE WE ALREADY IN FUN DEFN MODE... YES -- MUST BE A NEW HDR.
TESTFN	BAL,R14 BNEZ CI,R2 BE STW,R6	GFNREF REOPEN LBRACKET ERDEFN TFBLK+2	DOES FUN NAME HAVE A GLOBAL REF... YES -- MUST BE A REOPEN. NO, NEW HDR UNLESS LEFT BRACKET... OOPS -- DEFN ERROR. OK, SET NEW FUNCTION NAME.
SETFBLK	LI,R4	-FBLKSIZE	
MOVEFBLK	LW,R5 STW,R5 BIR,R4 LI,R4	TFBLK+FBLKSIZE,R4 FBLK+FBLKSIZE,R4 MOVEFBLK -1	MOVE TEMP NAME PTRS & ZEROS INTO THE CURRENT OPEN HDR BLOCK.
MOVECLCS	AI,R4 CH,R4 BE LH,R5 STH,R5 B	1 TNLCLS SETNLCLS TLOCALS,R4 LOCALS,R4 MOVECLCS	MOVE TEMP LOCAL NAME PTRS INTO THE CURRENT OPEN HDR LOCALS BLOCK.
SETNLCLS	STH,R4 STW,R1 LW,R4 BNEZ LW,R5 BEZ LW,R5 CW,R5 BE XW,R4 LI,R7 B	NLCLS FUNSAVE MODE SETFMODE FDEFPTR ZTEST FNOFF,R5 FBLK+2 ZTEST FDEFPTR ZTEST DREF	SET NO.OF LOCALS IN CURRENT OPEN BLK SAVE BYTE PTR. ARE WE ALREADY IN FUN DEFN MODE... NO -- A NEW OPEN. YES -- A RE-HEADER. (NO OLD FUNCTION CURRENTLY OPEN) GET OLD FUNCTION'S NAME PTR. DOES IT MATCH NEW HDR'S FUN NAME PTR YES, SOME OTHER HDR CHANGE. NO, FORGET OLD FUNCTION. RETURN TO ZTEST AFTER DE-REFERENCING OLD FUNC.DESCRPTOR INITIALIZE THE LINE CHAIN.
SETFMODE	BAL,R14 LI,R4 STW,R4 STW,R4	INITLC 0 MODE LINVAL	SET FUN DEFN MODE. SET LINE VALUE = 0.
ZTEST	CI,R2 BE LW,R1 B	NEWLINE NXTPROMT FUNSAVE CLOSEDEL	DID HDR END ON NEWLINE OR DEL... NEWLINE -- PROMPT FOR LINE 1. (RESTORE BYTE PTR). DEL -- CLOSE THE FUNCTION.
REHDR	CLM,R3 BCS,9 B	DELDEL ACQHDR CLOSEDEL	IS NEW LINE 0 JUST A CLOSING DEL... NO -- ACQ A NEW HDR FOR OPEN FUNC. YES -- CLOSE THE FUNCTION.
REOPEN	STW,R1 STW,R7	FUNSAVE GLOBLOC	SAVE PTR TO LF.BRACKET,NEWLINE,DEL. SAVE LOC.OF GLOBAL REFERENCE TO THIS NAME. THIS LOC WILL NOT CHANGE DURING FUNCTION DEFN MODE.
*			GET TYPE OF REFERENT FOR FUN NAME.
*			IT SHOULD BE A FUNCTION DESCRIPTOR.
	LB,R14 CLM,R14 BCS,9 LI,R12 CW,R12 BANZ LI,R12 LW,R7 LW,R14 BDR,R14 B	*R4 FUNTPYES ERDEFN LOCKFLAG 0,R4 ERDEFN PENDFLAG STATEPTR 1,R4 RESTATEQ REOPENOK X'7FFF'	OOPS -- DEFN ERROR. OK, IS IT A LOCKED FUNCTION... YES -- DEFN ERROR. NO, SET FOR PENDENT TEST. PT AT TOP STATE-ENTRY IN EXEC.STACK. = NO.OF REFERENCES TO FUNC.DESCR. TEST STATE-ENTRY IF MORE THAN 1 REF. OK, ONLY 1 (FOR GLOBAL NAME). EXTRACT OFFSET TO NEXT STATE-ENTRY.
RESTATE	LI,R11 AND,R11 BEZ	0,R7 ERDEFN	OOPS, FINAL ENTRY -- DEFN ERROR (THIS DEFN ERROR RESULTS DUE TO PENDENT STATUS; A D-ENTRY, DYADIC FUNCTION, EXISTS FOR THE NAMED FUNCTION. THE D-ENTRY IS NOT A STATE-ENTRY, BUT IT WILL BECOME ONE AS SOON AS ITS LEFT ARGUMENT IS RESOLVED.)
*			
*			
*			
*			
*			
*			
	AW,R7	R11	PT AT IT.

RESTATEQ	CS, R4	1, R7	DOES IT REFER TO THE FUNC.DESCR...
	BNE	RESTATE	NO.
	CW, R12	0, R7	YES, IS IT A PENDENT STATE...
	BANZ	ERDEFN	YEP -- DEFN ERROR.
	BDR, R14	RESTATE	NOPE, LOOP TILL REFS ALL TESTED.
REOPENOK	MTW, 1	1, R4	INCR FUNC.DESCRPTOR'S REF-COUNT AND
	STW, R4	FDEFPTR	SAVE PTR TO FUNC.DESCRPTOR.
	LCI	4	GET RESULT, LF.DUMMY, FUNNAME, AND
	LM, R11	RESOFF, R4	RT. DUMMY NAME PTRS.
	STM, R11	FBLK	SET CURRENT OPEN FUNCTION BLOCK.
	AW, R4	NFLOFF, R4	PT AT THE FUNC.DESCRPTOR'S
	AI, R4	NFLOFF+1	LOCAL & LABEL COUNTS.
	LH, R7	*R4	GET NO.OF LOCALS.
	STH, R7	NLCLS	
	INT, R11	0, R4	GET NO.OF LABELS.
	SLS, R11	1	(2 WDS PER LABEL).
	AW, R4	R11	PT AT 1ST WD CONTAINING LOCAL PTR.
	AI, R4	1	
	SLS, R4	1	USE HALFWD RESOLUTION.
	LI, R7	-1	SET FOR LOCAL COUNT TEST.
RELCLS	AI, R7	1	BUMP LOCAL COUNT.
	CH, R7	NLCLS	HAVE WE GOTTEN ALL LOCALS YET...
	BGE	RECHAIN	YES -- DEVELOP THE LINE-CHAIN.
	LH, R12	0, R4	NO, GET LOCAL NAME PTR.
	STH, R12	LOCALS, R7	PUT IT IN CURRENT OPEN LOCAL BLOCK.
	AI, R4	1	PT AT NEXT HALFWD.
	B	RELCLS	
RECHAIN	BAL, R14	INITLC	INITIALIZE THE LINE-CHAIN.
	LI, R12	3	= OFFSET FROM NEXT LC ENTRY TO LAST.
	LI, R13	0	PRESET LINE-VALUE HOLDER.
	LI, R7	0	PRESET LINE COUNTER.
RELINE	LW, R4	FDEFPTR	PT AT FUNC.DESCRPTOR AGAIN.
	CW, R7	NFLOFF, R4	HAVE ALL OF ITS LINE PTRS BEEN USED.
	BL	REPTR	NOT YET.
	STW, R13	LINVAL	YES, SAVE LAST LINE-VALUE.
	SW, R1	HILINE	= OFFSET FROM TOP LC ENTRY TO BIG-1.
	AND, R1	XFFFF	(OFFSETS ARE HALFWD).
	LW, R4	HILINE	PT AT THE HI BOUNDARY LINE-CHAIN
	STW, R1	1, R4	ENTRY AND UPDATE ITS OFFSET.
	LW, R1	FUNSAVE	RESTORE PTR TO LF.BRACKET OR WHATEVR
	LI, R9	0	PRE-SET DIR.CNT SO LF BRACK OVERRIDE
	STW, R9	MODE	SET FUNCTION DEFINITION MODE. U15-0031
	LB, R2	0, R1	GET THE TERMINATION CHAR AGAIN.
	AI, R1	-1	BACK UP FOR DIRECTIVE HUNT.
	AI, R2	-LBRACKET	TERMINATE ON LEFT BRACKET...
	BNEZ	DIRHUNT	NO, WE'LL USUALLY REPROMPT (N+1)
	B	DIRHUNTR	YES, IT MAY BE AN OVERRIDE.
REPTR	BAL, R10	RAISE3	RAISE EXECUTION STACK BY 3 WDS.
	LI, R3	16	SET SHIFT COUNT.
	SCD, R12	-16	SHIFT LO DIGIT OF LINE NO. TO TOP 12
RELINEA	AW, R12	BITPOS-3	INCR.
	LB, R2	R12	LOOK AT IT.
	AI, R2	X'FFF60'	CK FOR INCR TO 10.
	BNEZ	RELINEOK	NO, OK.
	LI, R12	0	YES, CLEAR & CARRY.
	AI, R3	4	BUMP SHIFT COUNT.
	SCD, R12	-4	SHIFT NEXT DIGIT TO TOP OF R12.
	B	RELINEA	
RELINEOK	SCD, R12	0, R3	SHIFT INCR LINE VALUE BACK TO R13 &
*			(AUTOMATICALLY) RESTORE R12 = 3.
	AI, R7	1	INCR LINE COUNTER.
	AW, R4	R7	GET THAT LINE'S PTR WD.
	LW, R11	NFLOFF, R4	
	AW, R11	LCENTRY	MAKE IT LINE-CHAIN CATEGORY.
	LCI	3	
	STM, R11	0, R1	STORE THE LINE-CHAIN ENTRY.
	AI, R11	-1	BUMP THE REF COUNT OF THE CODESTRING
	MTW, 1	*R11	DATA BLOCK.
	B	RELINE	
FIHANDLR	LI, R1	BA(IMAGE) -1	SET TO BEGIN SCANNING IMAGE.
DIRHUNT	LI, R9	-1	= DIRECTIVE COUNTER
DIRHUNTR	LI, R11	-1	SET NON-NEG. IF EDIT OR DISPLAY-N
	LI, R10	-1	SET NON-NEG. IF EDIT OR DISPLAY-1.
	STW, R10	SAVECDEL	NO CLOSING DEL, SO FAR.
DIRQ	AI, R9	1	BUMP DIRECTIVE COUNT.
	BAL, R4	ACQNXNB	GET NEXT NON-BLANK CHAR & CODE.
	CI, R2	LBRACKET	TEST FOR LEFT BRACKET...
	BNE	PASTDIR	NO -- WE ARE PAST DIRECTIVES.

	LI, R10	-1	YES, NEW DIRECTIVE; IGNORE ANY
	LI, R11	-1	PRIOR DIRECTIVE SETTINGS.
	BAL, R4	ACQNXNB	
	CI, R2	QUAD	CK FOR LEADING QUAD...
	BE	DIRSHOWN	YES -- MUST BE A DISPLAY-N DIR.
	BAL, R14	DIRLV	NO, EXPECT A LINE VALUE.
	STW, R12	LINVAL	IT BECOMES THE CURRENT LINE VALUE.
	CI, R2	QUAD	WAS IT FOLLOWED BY A QUAD...
	BNE	DIRZ	NO -- COULD BE OVERRIDE, DELETE,
*			INSERT, REPLACE, OR SKIP.
	LW, R10	LINVAL	YES, COULD BE EDIT OR DISPLAY-1.
	BAL, R4	ACQNXNB	
	CI, R3	9	CK FOR DIGIT CODE NEXT...
	BLE	DIREDIR	YES -- AN EDIT-DIRECTIVE.
DIRZ	AI, R2	-RBRACKET	END DIRECTIVE ON A RT. BRACKET...
	BEZ	DIRQ	OK, CK FOR ANOTHER DIRECTIVE.
	B	ERDIR	OOPS -- DIRECTIVE ERROR.
DIRSHOWN	LI, R11	0	DISPLAY ALL (0) OR SOMETHING.
	BAL, R4	ACQNXNB	
	CI, R2	RBRACKET	
	BE	DIRQ	CK FOR RT. BRACKET NEXT...
*			YES, DISPLAY-ALL; CK FOR ANOTHER
			DIRECTIVE.
	BAL, R14	DIRLV	NO, EXPECT LINE VALUE.
	LW, R11	R12	PUT IT IN DISPLAY-N INDICATOR.
	B	DIRZ	
DIREDIR	LI, R11	0	CLEAR COLUMN ACCUMULATOR.
DIRE	MI, R11	10	
	AW, R11	R3	ACCUMULATE COLUMN NO. TO START EDIT.
	BAL, R4	ACQNXCC	ACQ. NEXT CHAR & CODE.
	CI, R3	9	CK FOR DIGIT...
	BLE	DIRE	YES.
	LI, R4	DIRZ	NO, SET EXIT POINT 'DIRZ' AFTER
	B	ACQNB	INSURING NON-BLANK REACHED.
DIRAN	BAL, R4	ACQNXCC	ACQ. NEXT CHAR & CODE.
DIRAQM	CI, R3	9	CK FOR DIGIT...
	BG	0, R6	NO -- RETURN.
	SLS, R12	4	YES, ACCUMULATE A HEX DIGIT.
	AW, R12	R3	
	AI, R5	-4	DECR SHIFT COUNT.
	BGEZ	DIRAN	OK, KEEP GOING, NOT TOO MANY DIGS.
	B	ERDIR	OOPS -- DIRECTIVE ERROR.
DIRLV	LI, R12	0	CLEAR LINE VALUE ACCUMULATOR.
	CI, R3	9	ARE WE ON A DIGIT...
	BLE	DIRINT	YES, HANDLE INTEGER PART.
	LI, R5	12	NO, SET SHIFT COUNT FOR MAX. 3
	AI, R2	-'. '	DIGITS, AND VERIFY DEC.PT.
	BEZ	DIRPT	OK, HANDLE FRACTION PART.
	B	ERDIR	OOPS -- DIRECTIVE ERROR.
DIRINT	LI, R5	16	NO MORE THAN 4 SIG. DIGITS ACCEPTED.
	LI, R4	DIRLZ	IGNORE LEADING ZEROES.
DIRLZ	AI, R3	0	
	BEZ	ACQNXCC	GOT ONE, ACQ. NEXT CHAR & CODE.
	BAL, R6	DIRAQM	OK, ACCUMULATE INTEGER PART.
	LI, R5	12	SET SHIFT COUNT FOR MAX. 3 MORE DIGS
	CI, R2	-. '	CK FOR DEC.PT...
	BNE	DIRSHIFT	NO, SHIFT TO INTEGER LINE VALUE.
DIRPT	BAL, R6	DIRAN	ACCUM FRACTION PART, IF ANY.
DIRSHIFT	SLS, R12	0, R5	SCALE THE LINE VALUE APPROPRIATELY.
	LW, R4	R14	SET TO RETURN FROM 'DIRLV' AFTER
	B	ACQNB	INSURING NON-BLANK REACHED.
PASTDIR	LCI	3	SAVE SCAN REGS.
	STM, R1	FUNSAVE	
	AI, R11	0	CK FOR SHOW-N OR EDIT DIRECTIVE...
	BGEZ	VERTERM	YES, VERIFY PROPER TERMINATION.
	AI, R10	0	NO, CK FOR SHOW-1 DIRECTIVE...
	BLZ	CSNQ	NO.
	LI, R1	-17	YES.
SAVIMAGE	LD, R2	IMAGE+34, R1	SAVE CURRENT IMAGE.
	STD, R2	VISIMAGE+34, R1	
	BIR, R1	SAVIMAGE	
	STW, R9	CONSTBUF	SAVE DIRECTIVE COUNT.
	BAL, R4	SHOW1	DISPLAY THE LINE, CLOBBERING IMAGE.
	LW, R9	CONSTBUF	RESTORE DIRECTIVE COUNT.
	LI, R1	-17	
GETIMAGE	LD, R2	VISIMAGE+34, R1	RESTORE THE IMAGE, IT MIGHT BE A
	STD, R2	IMAGE+34, R1	REPLACEMENT FOR DISPLAYED LINE.
	BIR, R1	GETIMAGE	
CSQ	LCI	3	RESTORE SCAN REGS.

CSNQ	LM, R1	FUNSAVE	WAS DIRECTIVE FOLLOWED BY NEW LINE.	
	CI, R2	NEWLINE	YES -- NO CODESTRINGING.	
	BE	NOCS	WAS IT FOLLOWED BY A DEL...	
	CLM, R3	DEDEL	YES -- CLOSE THE FUNCTION.	
	BCR, 9	CLOSEDEL	NO, IS LINE VALUE ZERO...	
	LW, R12	LINVAL	NOPE, CODESTRING THE FUNCTION LINE	
	BNEZ	CSFUNLIN	YEP, BUT IS IT A CMND...	
	CI, R2	')'	JA -- REWRITE THE HEADER.	
	BNE	REHDR	DO CODESTRING OR RECOGNIZE COMMAND.	
CSFUNLIN	BAL, R12	CSN	SEARCH THE LINE CHAIN FOR THE	U15-0033
CSLINECK	LW, R12	LINVAL	LINE VALUE...	
	BAL, R5	LCSEARCH	NO MATCH -- AN INSERT.	
	B	INSERT	MATCH -- A REPLACEMENT.	
	LI, R4	0	CLEAR & GET CODESTRING PTR.	
	XW, R4	CURRCS	ADD LINE-CHAIN CATEGORY VALUE.	
	AW, R4	LCENTRY	PUT & TAKE.	
REPLACE	XW, R4	0, R7	DID WE TAKE A LIVE ENTRY...	
	AND, R4	X1FFFF	NO.	
	BEZ	CDELQ	YES, PT AT ITS DATA BLK HEADER.	
	AI, R4	-2	CK FOR CLOSE ON THIS LINE OF INPUT	
	LI, R7	CDELQ	AFTER DEREFERENCING THIS BLOCK.	
	B	DREF	ENTERED DUE TO LINE-SCAN ERROR.	U15-0035
USEBADFL	LI, R12	EFLAG	IS THIS AN 'EXECUTED' FUN DEFN...	U15-0036
	CW, R12	OLDMODE	NO, ACCEPT THE BAD LINE.	U15-0037
	BAZ	CSLINECK	YES, DON'T CHANGE THE FUNCTION.	U15-0038
	B	ERRORCK	WAS A DELETE INDICATED...	
NOCS	LW, R12	EDITCK	YES.	
	BNEZ	DELETE	NO, WAS A LINE NO. OVERRIDE GIVEN.	
	AI, R9	-1	NO -- MOVE UP A LINE VALUE.	
	BLEZ	NXTPROMT	YES -- ISSUE THAT LINE NO. AS	
	B	REXPROMT	THE FUNCTION LINE PROMPT.	
*			FIND THE LINE VALUE IN THE LINE-CHN;	
DELETE	LW, R12	LINVAL	HOWEVER -- CAN'T DELETE HEADER.	
	BEZ	ERDEFN		
	BAL, R5	LCSEARCH		
	B	CDELQ	NO MATCH.	
	LW, R4	LCENTRY	MATCH -- REPLACE BY A DEAD ENTRY,	
	B	REPLACE	I.E. CATEGORY VALUE BUT NO CS PTR.	
INSERT	BAL, R10	RAISE3	RAISE EXECUTION STACK BY 3 WDS.	
	STW, R12	2, R1	PUT NEW LINE VALUE IN THE LC ENTRY.	
	LI, R4	0	CLEAR & GET CODESTRING PTR.	
	XW, R4	CURRCS		
	AW, R4	LCENTRY	ADD LINE-CHAIN CATEGORY VALUE.	
	STW, R4	0, R1	SET TOP WD OF LINE-CHAIN ENTRY.	
	SW, R7	TOPOSTAK	= OFFSET TO NEXT LOWER LINE-CHAIN	
	STW, R7	1, R1	ENTRY; SET THIS ONE'S CHAIN WD.	
	SW, R6	R7	= OFFSET FROM NEXT HIGHER LC ENTRY.	
	SW, R1	R6	PT AT NEXT HIGHER LINE-CHAIN ENTRY.	
	STH, R6	*R1, R2	CORRECT THAT CHAIN WD.	
CDELQ	LW, R12	SAVECDEL	DID THIS INPUT END ON CLOSING DEL...	
	BGZ	CLOSE	YES -- CLOSE THE FUNCTION.	
NXTPROMT	BAL, R12	NXTFLINE	CALC. NEXT LINE VALUE.	
REXPROMT	LI, R12	EFLAG		U15-0040
	CW, R12	OLDMODE	IF THIS IS AN 'EXECUTED' FUNC.DEFN,	U15-0041
	BANZ	ACLOSE	CLOSE AUTOMATICALLY.	U15-0042
DOPROMPT	LI, R12	INPF	REQUEST FUNCTION INPUT AFTER	U15-0043
	B	GENFPRMT	GENERATING THE LINE NO. TO PROMPT.	
SHOW1	STW, R11	MERGECDL	SET NEG. TO INDICATE NON-EDIT.	
	STW, R4	LINKFD	SAVE LINK.	
	LW, R12	LINVAL	GET THE LINE VALUE.	
	BEZ	SHOW1HDR	ZERO -- SHOW THE HEADER ONLY.	
	BAL, R5	LCSEARCH	FIND ITS LINE-CHAIN ENTRY, IF ANY.	
	B	*LINKFD	NONE.	
GENLINE	LI, R4	X'1FFFF'	EXTRACT THE CODESTRING PTR, IF ANY.	
	AND, R4	0, R7		
	BEZ	*LINKFD	NONE (DEAD LINE-CHAIN ENTRY).	
	STW, R4	CURRCS	OK, SET PTR FOR 'SHOWFL'.	
	LW, R6	2, R7	DISPLAYED LINE BECOMES THE	
	STW, R6	LINVAL	CURRENT LINE VALUE.	
	BAL, R12	GENFPRMT	GENERATE THE LINE NO.	
SHOW	LW, R14	LINKFD	RETURN VIA SAVED LINK AFTER	
	B	SHOWFL	DISPLAYING THE LINE.	
SHOW1HDR	BAL, R12	GENFPRMT	GENERATE LINE NO. ZERO.	
GENHDR	LI, R12	0	ZERO THE IMAGE COUNTER.	
	STW, R12	IMAGES		
	LW, R8	FBLK	GET RESULT NAME PTR, IF ANY.	
	BLZ	GENLDMY	NO RESULT.	
	BAL, R13	GENNAME	GEN. THE NAME.	

	LI,R7	LFARROW	GEN. THE LEFT ARROW.	
GENLDMY	BAL,R13	GENCHAR		
	LW,R8	FBLK+1	GET LEFT DUMMY NAME PTR, IF ANY.	
	BLZ	GENFN	NO LEFT DUMMY.	
	BAL,R13	GENNAME	GEN. THE NAME.	
	LI,R7	' '	GEN. A BLANK.	
GENFN	BAL,R13	GENCHAR		
	LW,R8	FBLK+2	GET FUNCTION NAME PTR.	
	BAL,R13	GENNAME	GEN. THE NAME.	
	LI,R6	-1	PRESET LOCAL COUNTER.	
	LW,R8	FBLK+3	GET RIGHT DUMMY NAME PTR, IF ANY.	
	BLZ	GENLCLS	NO RIGHT DUMMY.	
	LI,R7	' '	GET A BLANK CHAR.	
GENCN	BAL,R13	GENCHAR	GEN. THE CHAR.	
	BAL,R13	GENNAME	GEN. THE NAME.	
GENLCLS	AI,R6	1	INCR. THE LOCAL COUNTER...	
	CH,R6	NLCLS	VS NO.OF LOCALS...	
	BGE	SHOW	DONE -- SHOW THE HEADER.	
	LI,R7	','	MORE, GET A SEMICOLON CHAR.	
	LH,R8	LOCALS,R6	GET THAT LOCAL NAME PTR.	
	B	GENCN		
EDIT	LI,R4	EFLAG	IF THIS IS AN 'EXECUTED' ATTEMPT	U15-0045
	CW,R4	OLDMODE	TO EDIT...	U15-0046
	BANZ	ERDIR	DIRECTIVE ERROR.	U15-0047
EDITOK	BAL,R4	SHOW1	@ DISPLAY FOR EDIT (R11 IS POSITIVE)	U15-0048
	CI,R14	EDITOK+1	@ DID WE FIND LINE (SEE ALSO 'SHOW')	U15-0049
	BNE	REXPROMPT	NO (DIDN'T EXIST OR WAS DELETED).	
	LI,R1	0	INIT: NO.OF CHARS IN -- VISUAL IMAGE (OLD)	
	LI,R2	0	EDIT CONTROL LINE	
	LI,R3	0	VISUAL IMAGE (NEW)	
	LI,R11	9999	INIT. NO.OF CHARS BEFORE MERGING BEGINS.	
	B	ECTRL	START LOOKING AT EDIT CONTROL LINE.	
EGET1	LB,R12	VISIMAGE,R1	GET NEXT CHAR FROM OLD VISUAL IMAGE.	
	AI,R1	1	COUNT THAT CHAR & OFFSET TO NEXT.	
	CI,R12	NEWLINE	IS IT END-OF-STMT...	
	BNE	1,R7	NO -- NORMAL EXIT.	
	BDR,R1	0,R7	YES -- BACK UP & TAKE END EXIT.	
EDELETE	BAL,R7	EGET1	@ GET (& FORGET) 1 CHAR OF OLD IMAGE.	
	B	ENXTCTRL	@ (END-OF-STMT) LOOK AT NEXT CONTROL	
	B	ENXTCTRL	@ (NORMAL CHAR) LOOK AT NEXT CONTROL	
EDIG09	CW,R3	R11	HAVE WE FOUND MERGE PT. ALREADY...	
	BGE	EBLANKER	YES.	
	LW,R11	R3	NO, SET MERGE PT. NOW.	
EBLANKER	LI,R12	' '	SET TO INSERT A BLANK IN NEW IMAGE.	
	LI,R7	EBLANQ	RETURN FROM 'EMOVER' TO 'EBLANQ'.	
EBLANQ	AI,R9	-1	DECR THE BLANK COUNT...	
	BGEZ	EMOVER	PUT 1 MORE IN NEW IMAGE.	
EMOVE1	BAL,R7	EGET1	@ GET NEXT CHAR OF OLD IMAGE...	
	B	ENXTCTRL	@ (END-OF-STMT) LOOK AT NEXT CONTROL	
	BAL,R7	EMOVER	@ (NORMAL CHAR) PUT IT IN NEW IMAGE.	
ENXTCTRL	AI,R2	1	OFFSET TO NEXT CONTROL CHAR.	
ECTRL	LB,R9	IMAGE,R2	GET EDIT-CONTROL CHAR.	
	CI,R9	' '		
	BE	EMOVE1	BLANK -- MOVE 1 OLD CHAR TO NEW.	
	CI,R9	','		
	BE	EDELETE	SLASH -- SKIP 1 OLD CHAR.	
	CI,R9	NEWLINE		
	BE	EZCTRL	END-OF-EDIT-CONTROL-LINE.	
	AI,R9	- '0'		
	BLZ	ELETTER		
	CI,R9	9		
	BLE	EDIG09	DIGIT -- INSERT 0 TO 9 BLANKS.	
ELETTER	AI,R9	'0'-'A'+1	(NON-CONTROL-CHAR)	
	BLZ	REXPROMPT		
	CI,R9	9		
	BLE	ELETAI	A - I -- INSERT 5 TO 45 BLANKS.	
	AI,R9	'A'-'J'	(NON-CONTROL-CHAR)	
	BLZ	REXPROMPT		
	CI,R9	9		
	BLE	ELETJR	J - R -- INSERT 50 TO 90 BLANKS.	
	AI,R9	'J'-'S'	(NON-CONTROL-CHAR)	
	BLZ	REXPROMPT		
	CI,R9	8		
	BG	REXPROMPT	(NON-CONTROL-CHAR)	
	AI,R9	9	S - Z -- INSERT 95 TO 130 BLANKS.	
ELETJR	AI,R9	9		
ELETAI	MI,R9	5	SET BLANK COUNT.	
	B	EDIG09		

EZCTRL	BAL, R7	EGET1	@ MOVE OLD CHARS INTO NEW VISUAL IMAGE
	B	EDITOUT	@ UNTIL END-OF-OLD-STMT. OCCURS.
	LI, R7	EZCTRL	@ AFTER MOVE, GET 1 MORE CHAR.
EMOVER	CI, R3	130	@ REFUSE TO MOVE MORE THAN 130 TO NEW.
	BE	0, R7	EXIT WITHOUT MOVING.
	STB, R12	INBUF, R3	PUT CHAR IN NEW VISUAL IMAGE.
	AI, R3	1	COUNT THAT CHAR & OFFSET TO NEXT.
	B	0, R7	EXIT.
EDITOUT	CW, R3	R11	WERE ANY BLANK INSERTIONS INDICATED.
	BGE	EMERGCOL	YES.
	LW, R11	R3	NO, SET MERGE-PT. NOW.
EMERGCOL	STW, R11	MERGECOL	SET MERGE COLUMN (-1).
	LI, R4	-17	MOVE NEW VISUAL IMAGE (+) TO IMAGE.
ESWITCH	LD, R8	INBUF+34, R4	
	STD, R8	IMAGE+34, R4	
	BIR, R4	ESWITCH	
	B	INP4MERG	
*			INPUT FOR MERGE INTO IMAGE (R3 IS
VERTERM	CI, R2	NEWLINE	THE NO.OF CHARS PRESENTLY IN IMAGE).
	BE	SHEDQ	DIRECTIVE FOLLOWED BY:
	CLM, R3	DELDEL	END-OF-STMT -- OK.
	BCS, 9	ERDIR	DEL OR LOCKED DEL...
	BAL, R4	ACQNXNB	NO -- DIRECTIVE ERROR.
	AI, R2	-NEWLINE	YES, VERIFY END-OF-STMT NEXT.
	BNE	ERDIR	
SHEDQ	AI, R10	0	OOPS -- DIRECTIVE ERROR.
	BGEZ	EDIT	IS THIS A SHOW-N OR EDIT...
	STW, R10	MERGECOL	EDIT.
	LI, R4	SHOWUP	(SET NEG. TO INDICATE NON-EDIT)
	STW, R4	LINKFD	SHOW-N.
	LW, R12	R11	SET LINKAGE FOR DISPLAY LOOP.
	STW, R12	STARTLV	GET STARTING LINE VALUE.
SHOWSRCH	BAL, R5	LCSEARCH	SAVE IT.
	B	SHOWHIRE	SEARCH FOR LINE VALUE IN LINE-CHAIN.
	STD, R6	LINKHIRE	MISS, USE NEXT HIGHER ENTRY.
	AI, R12	0	OK, SAVE PTR AND BACK-LINKAGE.
	BNEZ	GENLINE	IS THIS THE HEADER LINE...
	LI, R6	GENHDR	NO -- GEN. THE LINE NORMALLY.
*			YES -- GEN. THE HEADER AFTER
GENDEL	LW, R12	BLANKS	DISPLAYING OPENING DEL.
	STW, R12	IMAGE	GEN. 4 BLANKS.
	LI, R3	4	
	LW, R7	SAVEDEL	R3 = CHAR COUNT.
	BAL, R13	GENCHAR	GET THE OPENING DEL.
	LI, R7	' '	GEN. IT.
	LW, R13	R6	GEN. ANOTHER BLANK AND RETURN.
	B	GENCHAR	
SHOWUP	LD, R6	LINKHIRE	RESTORE PTR AND BACK-LINKAGE.
SHOWHIRE	SW, R7	R6	BACK UP TO NEXT HIGHER ENTRY.
	LW, R12	2, R7	GET ITS LINE VALUE.
	CW, R12	MAXLINV	IS IT THE UPPER BOUND...
	BLE	SHOWSRCH	NO -- HUNT FOR IT AGAIN (OH HELL).
	LW, R12	STARTLV	YES, CK STARTING LINE VALUE...
	BNEZ	SHOWN	DIDN'T START ON HEADER.
	BAL, R6	GENDEL	HEADER, GEN. CLOSING DEL.
	BAL, R14	SHOWFL	DISPLAY IT.
SHOWN	LW, R2	FUNSAVE+1	RESTORE NEWLINE OR CLOSING DEL...
	CI, R2	NEWLINE	
	BE	NXTPROMT	
	B	CLOSED	NEWLINE -- ISSUE NEXT LINE PROMPT.
CLOSEQ	LCI	3	CLOSING DEL -- CLOSE THE FUNCTION.
	STM, R1	FUNSAVE	SAVE SCAN REGS.
	BAL, R4	ACQNXNB	
	CI, R2	NEWLINE	ACQ. NEXT NON-BLANK.
	BE	CLOSECS	IS IT A NEW LINE CHAR...
	LCI	3	YES, CLOSE THE CODESTRING.
	LM, R1	FUNSAVE	NO, RESET SCAN REGS.
	B	ERLSCAN	
CLOSECS	LW, R5	FUNSAVE+1	ISSUE -- LINE-SCAN ERROR.
	STW, R5	SAVECDEL	GET THE CLOSING DEL,
	B	CSZ	AND SAVE IT FOR 'CDELQ'.
CLOSEDEL	STW, R2	SAVECDEL	FINISH CODESTRING.
	BAL, R4	ACQNXNB	SAVE CLOSING DEL.
	AI, R2	-NEWLINE	ACQ. NEXT NON-BLANK.
	BEZ	CLOSE	IS IT A NEW LINE CHAR...
	B	ERDEFN	YES, CLOSE THE FUNCTION.
ACLOSE	LW, R2	ERRORID	NO -- DEFN ERROR.
	BNEZ	ERRORCK	DID AN ERROR OCCUR...
			YES, ABANDON FUN DEFN ATTEMPT.

	LW, R2	SAVEDEL	NO, AUTOMATIC CLOSE ON OPENING DEL	
	B	CLOSED	FOR THE CLOSING DEL.	U15-0052
FORCLOSE	MTW, -1	MODE	FORCED-CLOSE; MODE = -1.	
	STW, R14	LINKFD	SAVE LINK TO FORCLOSE.	
	LW, R14	FBLK+2	PUT OPEN FUNCTION NAME AWAY.	
	STW, R14	OPENFN		
	LW, R2	OLDMODE	SAVE OLD MODE FOR RECOVERY WHEN THE	U15-0054
	STB, R2	OPENFN	FORCED OPEN OCCURS.	U15-0055
	LI, R2	DEL	ASSUME CLOSE ON ORDINARY DEL.	
	STW, R2	SAVEDEL		
CLOSED	STW, R2	SAVECDEL		
CLOSE	LW, R14	FUNTPES	INIT. FOR TYPE DETERMINATION:	
	LW, R8	FBLK		
	BLZ	TRYDY	NO RESULT.	
	AI, R14	1	RESULT.	
TRYDY	LW, R8	FBLK+1		
	BLZ	TRYMON		
	AI, R14	4	DYADIC	
	B	SAVETYPE		
TRYMON	LW, R8	FBLK+3		
	BLZ	SAVETYPE	NILADIC.	
	AI, R14	2	MONADIC.	
SAVETYPE	SLS, R14	8	MOVE UP 1 BYTE--LAST BIT= LOCK (=0).	
	STW, R14	NEWFTYPE	SAVE NEW FUNCTION TYPE.	
	LW, R4	FDEFPTR	IS THIS AN EXCHANGE OF FUNCTIONS...	
	BEZ	FWDLC	NO, CONVERT TO FORWARD LINE-CHAIN.	
	LI, R8	NODAMAGE	YES, ASSUME NO SI DAMAGE, SAVING	
	STW, R8	XREFLOC	EXCHANGE-REFERENT-LOCATION.	
	SLS, R14	-8		
	CB, R14	*FDEFPTR	COMPARE NEW TYPE VS OLD TYPE.	
	BE	FWDLC	SAME.	
	LW, R14	1, R4	DIFFERENT, GET REF-COUNT OF OLDIE.	
	AI, R14	-2	2 REFS = GLOBAL & FDEFPTR...	
	BEZ	FWDLC	OK TO EXCHANGE THEM.	
	LI, R14	EFLAG	OH-OH, SI DAMAGE WILL OCCUR.	U15-0057
	CW, R14	OLDMODE	IS THIS AN 'EXECUTED' FUNC.DEFN...	U15-0058
	BANZ	SIDAMGO	YES, GO AHEAD & DO DAMAGE.	U15-0059
	BAL, R14	SIDR	OLDIE SUSPENDED -- WISH TO GO ON..	
	BE	SIDAMGO	YES, SI DAMAGE IS ACCEPTED.	
	LI, R14	0	NO, STAY IN FUN DEFN MODE SO	
	STW, R14	MODE	USER CAN CORRECT NEW HEADER.	
	STW, R14	LINVAL	PROMPT WITH LINE ZERO.	
	B	REPROMPT		
SIDAMGO	LI, R14	DAMMSG	SET FOR SI DAMAGE MESSAGE WHEN THE	
	STW, R14	XREFLOC	ACTUAL EXCHANGES OCCUR.	
FWDLC	LI, R2	2		
	LI, R3	3		
	LI, R6	0		
	STW, R6	NFL	CLEAR FUNCTION LINE COUNT.	
	STH, R6	NFL, R3	CLEAR NO.OF LABELS.	
	LI, R7	X'1FFFF'		
	LI, R8	0		
	LI, R10	BOSCODE		
	LW, R1	HILINE	PT AT HI BOUNDARY LINE-CHAIN ENTRY.	
	LI, R5	0		
FWDLCE	LH, R9	*R1, R3	= OFFSET TO NEXT LOWER ENTRY.	
	STW, R5	1, R1	THIS ENTRY NOW LINKS TO NEXT HIGHER.	
	STH, R8	*R1, R2	CLEAR UPPER HALF OF LINK WORD.	
	LS, R6	0, R1	DOES THIS ENTRY PT TO CODESTRING...	
	BEZ	FWDLC0	NO.	
	MTW, 1	NFL	YES, BUMP LINE COUNT.	
	CB, R10	*R6, R2	DOES CODESTRING START ON BEGIN-STMT.	
	BE	FWDLC0	YES.	
	MTW, 1	NLBLS	NO, BUMP LABEL COUNT; LABELED LINE	
	LW, R11	0, R6	LABEL NAME PTR IS IN LO HALF; PUT IT	
	AI, R11	1	(AN ODD-VALUED VERSION)	
	STH, R11	*R1, R2	IN UPPER HALF OF THE LINK WORD.	
FWDLC0	AW, R1	R9	PT AT NEXT LOWER ENTRY.	
	LCW, R5	R9	= OFFSET BACK TO THE ONE JUST HIT...	
	BNEZ	FWDLCE	BUT ZERO MEANS LO BOUNDARY HIT.	
	LI, R11	STDFDSIZ	= STANDARD FUNC.DESRIPTOR SIZE.	
	AW, R11	NFL	+ 1 WD PER LINE PTR.	
	INT, R5	NLBLS		
	MI, R5	6	+ 2 WDS/LABEL & ALSO 4-WD DATA BLK.	
	AW, R11	R5		
	LH, R5	NLCLS		
	AI, R5	1		
	SLS, R5	-1	+ 1 WD/LOCALPAIR & MAYBE AN ODD ONE.	

	AW, R11	R5	= TOTAL BLK NEEDED EXC. 2-WD HDR.
	BAL, R14	ALOCNONX	ALLOC. SPACE, IF POSSIBLE.
	B	NOSPACE	YUCK -- WS FULL.
	STW, R4	RESULT	FINE, SAVE PTR TO NEW FUNC.DESCR.
	LW, R8	SAVEDEL	
	AW, R8	SAVECDEL	CK FOR LOCKED DEL ON OPEN OR CLOSE.
	AI, R8	-DEL-DEL	
	BEZ	SETLTYPE	NO.
	LI, R8	1	YES, SET LOCK-BIT.
SETLTYPE	AW, R8	NEWFTYPE	
	STH, R8	*R4	SET NEW FUN TYPE & LOCK FIELDS.
	SLD, R8	-9	SHIFT RESULT-BIT INTO SIGN OF R9,
	AI, R8	-4	STRIP OUT FUN-TYPE-BIT, R8 NOW =
	AI, R9	0	0 IF NILADIC, 1 IF MONADIC, 2 IF DY.
	BGEZ	SETXSIZE	R9 = + IF NO RESULT & - IF RESULTBIT
	AI, R8	1	FINALLY, R8 = # DUMMIES -- GASP.
SETXSIZE	LI, R5	1	
	AH, R8	NLBLS, R5	+ NO.OF LABELS.
	AH, R8	NLCLS	+ NO.OF LOCALS.
	SLS, R8	1	= 2 WDS PER SHADOWED NAME.
	AI, R8	4	+ MIN.OF 4 WDS STACKED ON FUN CALL.
	LW, R5	GLOBLOC	DID WE 'COPY' AN OLD FUNCTION...
	BEZ	SETXSIZW	NO.
	LW, R5	0, R5	
	LW, R9	XSIZOFF, R5	YES, GET OLDIE'S XSIZE WORD.
	CI, R9	X'E0000'	DOES IT PT TO AN ERR-CTRL TABLE...
	BAZ	SETXSIZW	NO.
	SLS, R9	15	YES. MOVE PTR TO ERR-CTRL TBL UP
	SLD, R8	17	& SHIFT IT IN, MOVING XSIZE.
	LW, R5	R8	PT AT ERR-CTRL DATA BLK.
	MTW, 1	1, R5	BUMP ITS REF-COUNT.
SETXSIZW	STW, R8	XSIZOFF, R4	SET 'XEQ STAK SIZE' FOR CALL &
*			(MAYBE) PTR TO ERR-CTRL TBL.
	LCI	FBLKSIZE+1	
	LM, R8	FBLK	SET RESULT THRU 'NO.OF FUN. LINES'.
	STM, R8	RESOFF, R4	
	LW, R8	LBLDBHDR	= HEADER WD FOR A LABEL'S DATA BLK.
	LI, R9	1	= REF COUNT OF NEW DATA BLK.
	LI, R10	0	= THE LABEL'S LINE NO. VALUE (LATER)
	AI, R4	NFLOFF	R4 PTS AT WD BEFORE 1ST LINE PTR.
	LW, R5	R4	
	AW, R5	NFL	R5 WILL PT AT THE NEXT
	AI, R5	2	LABEL-ENTRY PAIR, IF ANY.
	INT, R11	*RESULT	GET SIZE OF TOTAL RESULT BLOCK.
	AW, R11	RESULT	R11 PTS AT WD AFTER TOTAL BLOCK.
	AI, R11	NFLAG	STICK ON THE 'NON-ASSIGNABLE' FLAG.
*			
*	NOTE --	AT THIS POINT, IT IS NECESSARY THAT R2 = 2 AND R3 = 3 STILL.	
*		(SEE ALSO 'FWDLC').	
*			
	LI, R13	X'7FFFF'	MASK FOR STOP & TRACE BITS + PTR.
	LW, R1	HILINE	
	AI, R1	-3	PT AT LO BOUNDARY OF LINE-CHAIN.
	B	LCENDQ	START TRACKING THE LINE-CHAIN.
LLOOK	AW, R1	R6	PT AT NEXT HIGHER ENTRY.
	LS, R12	0, R1	IS IT LIVE...
	BEZ	LCENDQ	NO, A DELETED LINE PROBABLY.
	AI, R4	1	YES, PT AT NEXT FUN-LINE PTR WD.
	AI, R10	1	INCR LINE NUMBER.
	STW, R12	0, R4	SET THE FUNCTION-LINE PTR WD.
	LH, R12	*R1, R2	CK FOR LABEL NAME PTR ON THAT LINE.
	BEZ	LCENDQ	NO.
	AI, R11	-4	YES, PT AT DATA BLK FOR THE LABEL.
	AI, R12	-1	(MAKE EVEN NAMEPTR AGAIN)
	LCI	3	
	STM, R8	*R11	ESTABLISH DATA BLK USING LINE NUMBER
	STW, R12	0, R5	PUT LABEL'S NAME PTR AND ITS
	STW, R11	1, R5	N-FLAGGED DATA BLK PTR IN
	AI, R5	2	FUNC.DESCR. & PT AT NEXT WD.
LCENDQ	LH, R6	*R1, R3	= OFFSET TO NEXT HIGHER CHAIN ENTRY,
	BNEZ	LLOOK	ZERO MEANS HI BOUNDARY ENTRY.
	LW, R6	STATEPTR	DELETE THE ENTIRE LINE CHAIN.
	STW, R6	TOPOSTAK	
	LW, R14	NLCLS	
	STW, R14	1, R4	SET 'LOCALS & LABELS' COUNT WORD.
	SW, R11	RESULT	= ACTUAL SIZE OF FUNCTION DESCRIPTOR
	LI, R4	1	(IT INCLUDED LABEL DATA BLKS
	STH, R11	*RESULT, R4	ORIGINALLY, SO REVISE ACCORDING).

	SLS, R5	1	PT AT HALFWD AFTER LAST LABEL-ENTRY.	
	LI, R4	-1		
SETLCLS	AI, R4	1	= NO.OF LOCAL NAME PTRS SET, SO FAR.	
	CH, R4	NLCLS	FINISHED SETTING THEM YET...	
	BE	GLOBALIT	YES -- MAKE FUNC.DESCR. GLOBAL.	
	LH, R8	LOCALS, R4	NO, SET NEXT LOCAL.	
	STH, R8	0, R5		
	AI, R5	1		
	B	SETLCLS		
GLOBALIT	LI, R2	0	CLEAR RESULT PTR.	
	XW, R2	RESULT	R2 CONTAINS LOC OF NEW FUNC.DESCR.	
	AW, R2	BITPOS-0	SET 'GLOBAL' BIT.	
	LW, R6	FBLK+2	GET NAME PTR FOR NEW FUNCTION.	
	LW, R4	*SYMT, R6	IS GLOBAL REF IN SYM TBL...	
	BGEZ	SHADOWED	NO, ITS SHADOWED.	
	STW, R2	*SYMT, R6	YES, MAKE NEW GLOBAL REF-IND.	
	B	DOWNOLD		
SHADOWED	LW, R1	TOPOSTAK	PT AT TOP OF EXEC. STACK & FORGET IT	
SHADY	AI, R1	1	PT AT NEXT WD IN STACK.	
	LW, R4	1, R1	LOOK AT ITS SUCCESSOR, PROFESSOR...	
	BGEZ	SHADY	NOT A SHADOWED GLOBAL.	
	CW, R6	0, R1	GLOBAL -- SAME NAME...	
	BNE	SHADY	NO, BUT ITS THERE SOMEWHERE.	
	STW, R2	1, R1	YES, REVISE THAT GLOBAL REF-IND.	
DOWNOLD	BAL, R7	MAYDREF	DEREFERENCE ANY OLD FUN DESCR. PTR.	
	LW, R4	FDEFPTR	ARE WE POINTING AT OLD ONE...	
	BEZ	CLOSEOUT	NO.	
	LI, R3	X'1FFFF'	YES, SET TO EXCHANGE REFERENCES	
	LI, R5	X'1FFFF'	TO OLDIE WITH NEW OR SI DAMAGE.	
	LW, R1	STATEPTR	PT AT TOP STATE-ENTRY.	
	MTW, -1	1, R4	TEMPORARILY DECR OLDIE'S REF-COUNT.	
	BGZ	XREFQ	LOOK FOR STACKED REFS. TO IT.	
	B	XREFDONE		
XREFN	LI, R11	X'7FFF'		
	AND, R11	0, R1	PT AT NEXT STATE-ENTRY.	
	AW, R1	R11		
XREFQ	CS, R4	1, R1	IS IT A REFERENCE TO OLDIE.	
	BNE	XREFN	NO.	
	B	*XREFLOC	YES -- EXCHANGE OR DAMAGE.	
DAMMSG	B	SIDAMF	DISPLAY -- SI DAMAGE, AFTER RESUMING	
RFSIDAM	LI, R11	DAMAGE	SET TO DO DAMAGE.	
	STW, R11	XREFLOC		
	LI, R2	0		
DAMAGE	STW, R2	1, R1	ZILCH THE WHOLE FDEFPTR WD.	
	B	DECROLD		
NODAMAGE	STS, R2	1, R1	EXCHANGE FDEFPTR'S.	
	MTW, 1	1, R2	INCR NEW FUNC.DESCRPTOR'S REF-COUNT	
DECROLD	MTW, -1	1, R4	DECR OLDIE'S REF-COUNT.	
	BGZ	XREFN	KEEP GOING IF REFS REMAIN IN STACK.	
XREFDONE	MTW, 1	1, R4	DONE, FDEFPTR STILL PTS AT OLDIE.	
	LI, R3	0		
	STW, R3	FDEFPTR	OH NO, IT DOESN'T.	
	BAL, R7	DREF	RELEASE OLDIE & ALL IT REFS EXCLUS.	
CLOSEOUT	LW, R3	MODE	IS THIS A FORCED-CLOSE...	
	BLZ	*LINKFD	YES, RETURN.	U15-0061
RECOMODE	LI, R3	X'FFFF'	NO, RECOVER OLD MODE.	U15-0062
	AND, R3	OLDMODE		U15-0063
	STW, R3	MODE		U15-0064
	B	CMDEXIT	EXIT FUNCTION DEFN MODE.	U15-0065
NOSPACE	LI, R14	EFLAG		U15-0067
	CW, R14	OLDMODE	WAS THIS AN EXECUTED FUN DEFN...	U15-0068
	BANZ	WSFULL	YES, ABORT IT ON WS FULL.	U15-0069
	LW, R14	MODE	IS THIS A FORCED CLOSE...	U15-0070
	BEZ	BAKLC	NO, FIX LINE-CHAIN & RE-PROMPT.	
	LW, R14	BREAKFLG	YES, WAS IT A LINE DISCONNECT...	
	BGEZ	BAKLC	NO, FORGET FORCED-CLOSE.	
FORGETIT	LI, R14	X'FFFF'	YES, RESTORE OLD MODE & ABANDON	U15-0072
	AND, R14	OLDMODE	THE FUN DEFN ATTEMPT.	U15-0073
	STW, R14	MODE		U15-0074
ZAP	LI, R4	0		U15-0075
	XW, R4	FDEFPTR	CLEAR ANY REFERENCE TO OLDIE.	
	LI, R5	X'1FFFF'		
	LI, R14	CATLC	LINE-CHAIN CATEGORY OF STACK ENTRY.	
	BAL, R7	MAYDREF		
ZAPLC	CB, R14	*TOPOSTAK	IS LINE-CHAIN ENTRY STILL ON TOP.	
	BNE	*LINKFD	NO -- EXIT.	
	LS, R4	*TOPOSTAK	YES, GET ITS LINE PTR, IF LIVE.	
	MTW, 3	TOPOSTAK	DROP THAT LINE-CHAIN ENTRY.	

	AI, R4	-2	IF LIVE, PT AT CODESTR. DATA BLK HDR	
	BGZ	DREF	LIVE -- DE-REF IT & GO TO ZAPLC.	
	B	ZAPLC	DEAD.	
FERASECK	STW, R14	LINKFD	SAVE LINK FROM)ERASE CMND DRIVER.	
	LW, R14	FBLK+2	GET NAME PTR OF CURRENT OPEN FUNCT.	
	LW, R5	CONSTBUF	= NO.OF NAME PTRS FROM)ERASE CMND.	
ERASENQ	CW, R14	CONSTBUF, R5	CK FOR MATCHING NAME PTR IN ERASE...	
	BE	FORGETIT	YES, FORGET IT.	U15-0077
	BDR, R5	ERASENQ		
	B	*LINKFD	NO -- EXIT TO)ERASE CMND DRIVER.	
BAKLC	LI, R3	3	RESET LINE-CHAIN TO PT FROM HIGH	
	LW, R1	HILINE	ENTRY TO NEXT LOWER ENTRY...	
	AI, R1	-3		
	LI, R5	0		
BAKLCE	LH, R9	*R1, R3		
	STH, R5	*R1, R3		
	AW, R1	R9		
	LCW, R5	R9		
	BNEZ	BAKLCE		
	STW, R5	MODE	INSURE FUN DEFN MODE.	
	LI, R14	DOPROMPT	RE-PROMPT AFTER ISSUING -- WS FULL	U15-0079
	B	ERRFDWS	WHILE IN FUN DEFN MODE.	
RFUNDEF	B	*R14 (RETURN FROM	ERRFDWS, ERRFDSYM, OR SHOWFL--SEE	
*		EDITOK)		
WSFULL	BAL, R14	ERRFDWS	ISSUE -- WS FULL.	
ERRORCK	LI, R4	0		U15-0081
	XW, R4	CURRCS	DE-REFERENCE ANY CURRENT CODESTRING.	U15-0082
	BEZ	ERRORCKM		U15-0083
	AI, R4	-2		U15-0084
	BAL, R7	DREF		U15-0085
ERRORCKM	LI, R14	RECOMODE	PREPARE FOR EXIT.	U15-0086
	STW, R14	LINKFD	USE EXIT AS LINK IN CASE OF A ZAP.	U15-0088
	LW, R14	MODE	DID WE MAKE IT TO FUN DEFN MODE...	U15-0089
	BNEZ	ZAP	NO, ABANDON THE ATTEMPT.	U15-0090
	LI, R14	EFLAG	YES.	U15-0091
	CW, R14	OLDMODE	WAS THIS AN EXECUTED FUN DEFN...	U15-0092
	BANZ	ZAP	YES, ABORT THE ATTEMPT.	U15-0093
	B	DOPROMPT	NO, RE-PROMPT.	U15-0094
SYMFULL	LI, R14	ERRORCK	RETURN TO ERROR CHECK AFTER	
	B	ERRFDSYM	ISSUING -- SYMBOL TABLE FULL.	
ERDEFN	EQU	ERRDEFN	DEFINITION ERROR.	
ERDIR	EQU	ERDEFN	DEFN ERROR IN DIRECTIVE ANALYSIS.	U15-0099
	PAGE			

	SPACE 2			
Z	SET	%-FUNDEF@	SIZE OF FUNDEF MODULE IN HEX.	
	SPACE			
Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536	SIZE IN DECIMAL.	
	SPACE 3			
	END			

	TITLE	'WSCOPY-B00,08/22/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	CONTROL CARDS NOT PRINTED.
WSCOPY@	RES	0	ORIGIN OF WORKSPACE COPY MODULE.

*	REF'S	AND	DEF'S	
*				
	DEF		WSCOPY@	=START OF WSCOPY MODULE.
	DEF		COPY	ENTRY PT WHEN COMMITTED TO TRY COPY.
	DEF		RNOTFND	RESUME AFTER ERNOTFND.
	DEF		RNOTCPY	RESUME AFTER ERNOTCPY.
	DEF		RCSIDAM	RESUME AFTER SIDAMC.

SPACE 3			
*			

	REF	ACQCC	ACQUIRE CHAR & ITS CODE.
	REF	ACQNB	ACQUIRE NON-BLANK CHAR & ITS CODE.
	REF	ALOCNONX	ALLOC. N+2 WDS IF POSS.
	REF	CLEARMEM	CLEARs MEMORY, RELEASING UNNEC.PAGES
	REF	CLOSR	CLOSES & RELEASES A FILE.
	REF	CMDERR	COMMAND ERROR EXIT.
	REF	DREF	DE-REFERENCES A DATA BLK.
	REF	ERNOTCPY	ERROR -- NAME NOT COPIED (PENDENT).
	REF	ERNOTFND	ERROR -- NAME NOT FOUND.
	REF	SIDAMC	ERROR -- SI DAMAGE -- DUE TO COPYING
	REF	FINDNAME	FINDS NAME (OR HOLE) IN ACTIVE SYMT.
	REF	GARBCOLL	GARBAGE COLLECTOR.
	REF	RDACTIV	READS ACTIVE WS, ON TEMP FILE.
	REF	RDCOPY	READS COPY'S DYNAMIC.
	REF	RDCOPYDR	READS COPY DATA REGION, CLOSING FILE
	REF	RDNAMER	READS A NAME RECORD INTO WINDOW.
	REF	COPYDMES	REPORTS SAVED INFO & TAKES CMD EXIT.
	REF	SETFREET	INITs. FREE TABLE & OTHER CELLS.
	REF	SYSTERR	ERROR -- SYSTEM ERROR (HOPELESS).
	REF	WRACTIV	WRITES ACTIVE WS, ON TEMP FILE.
	REF	WRCOPYDR	WRITES COPY DATA REGION.
	REF	WRNAMER	WRITES A NAME RECORD FROM WINDOW.
	REF	XWLOCGLB	EXCHANGES LOCALS & GLOBALS.

*			
	REF	BACKUP	= 0 IF EXTENT OF COPY NOT YET
*			DETERMINED DUE TO A GROUP COPY.
	REF	BLANKCNT	NO.OF UNUSED ENTRIES IN ACTIVE
*			SYMBOL TABLE, SO FAR.
	REF	CONSTBUF	130 WORD TEMP BLK.
	REF	COPYBASE	LOC OF COPY DATA REGION IN COPY WS.
	REF	COPYHOME	LOC OF COPY DATA REGION IN ACTIVE WS
	REF	COPYSAVE	16 WD SAVE AREA ON DBLWD BOUND.
	REF	COPYSIZE	# WDS COPY DYNAMIC OR DATA REGION.
	REF	CORAVAIL	# WDS AVAIL. FOR THIS USER'S WS.
	REF	CORLEFT	# WDS NOT YET USED (PAGES NOT USED).
	REF	CORVSFIL	= 0 IF CORE-COPY OR = -1 IF FILE-CPY
	REF	CSYMT	LOC OF COPIED SYMBOL TABLE.
	REF	DISPLAC	DISPLACEMENT VALUE FOR RE-LOCATING.
	REF	DYNBOUND	BOUNDARY LOC FOR CURRENT DYNAMIC.
	REF	FREETBL	LOC OF FREE BLKS & THEIR SIZES.
	REF	FREETOTL	TOTAL AMT OF FREE DYNAMIC.
	REF	F:TF	TEMP FILE DCB.
	REF	GARBSAVE	REGISTER SAVE AREA, USED BY GARBCOLL
	REF	HICOMMON	LOC OF HIGHEST PT IN COMMON AREA.
	REF	IDBUF	TRYLOAD PUT COPY'S HDR INFO HERE.
	REF	IMAGE	CHARACTER BUFFER.
	REF	IMAGEPOS	BYTE ADDR.OF COPY-LIST IN IMAGE.
	REF	LINKCY	LINKAGE HOLDER FOR COPY ROUTINES.
	REF	LINKWY	LINKAGE HOLDER FOR WINDOWING.
	REF	LODYN	LOC OF LOWEST PT IN DYNAMIC AREA.
	REF	NAMEBUF	20-WD BLK TO HOLD NAMES.
	REF	NAMERKEY	KEY FOR ACCESSING NAME RECORDS ON
*			THE TEMP FILE (NOT THE REAL KEY).
	REF	NAMEWDS	TEMP CELL.
	REF	NILCK	STRAY DATA BLK PTR HOLDER.
	REF	NSYMTWDS	NO.OF WDS IN ACTIVE SYMBOL TABLE.
	REF	OFFSETK	TEMP CELL.
	REF	PCOPYFLG	= 0 IF COPY, SMALL NEG.NO. IF PCOPY.
	REF	STATEPTR	PTS AT TOP STATE-ENTRY IN XEQ STACK.
	REF	SYMT	PTS AT ACTIVE SYMBOL TABLE.
	REF	SYMTSIZE	# OF (2-WD) ENTRIES, ACTIVE SYM TBL.
	REF	TOPOSTAK	PTS AT TOP OF EXECUTION STACK.
	REF	WINDOW	512 WD TEMP BLK FOR NAME RECORD

```

*
*      REF      ERRORID      BLOCKING.
*                               ERROR I.D. CELL.
*                               REFS TO CONSTANTS & PARAMETERS.
*      REF      BITPOS      32-WD TBL OF BITS (BITPOS-K CONTAINS
*                               A WD HAVING A 1 ONLY IN BIT POS K)
*      REF      FUNTPES      RANGE OF FUNCTION TYPES.
*      REF      HASHINC      INCREMENT ON HASH CRASH.
*      REF      IN2CODE      BYTE TBL OF CODES FOR EACH CHAR.
*      REF      STACKOFF      OFFSET OF STACK FROM HI COMMON.
*      REF      XFFFF        = X'FFFF'
*      REF      X1FFFF        = X'1FFFF'
*      REF      IDT00BIG      ERROR I.D. FOR 'TOO BIG'.
*      REF      ID2NAMEY      ERROR I.D. FOR 'TOO MANY SYMBOLS'.
*
* EQU'S RELATED TO THE CONTEXT REF'S:
*
HASHAQM EQU COPYHOME      HASH ACCUMULATOR.
COPYBND EQU COPYSIZE      BOUNDS COPY KRUNCH REGION.
*
* STANDARD EQU'S
*
*      REGISTERS
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
*
NEWLINE EQU X'15'      END-OF-STMT CHAR.
NAMECODE EQU 23      NAME CODESTRING DESIGNATION.
PENDFLAG EQU X'8000'      MARKS PENDENT FUNCTION STATE-ENTRY.
TYPELIST EQU 6      DATA BLK TYPE = LIST.
TYPEGRP EQU 17      DATA BLK TYPE = GROUP.
LISTLOFF EQU 2      OFFSET TO LENGTH WD OF LIST DATA BLK
XSIZOFF EQU 2      OFFSET TO XSIZE WD OF FUNC.DESCR.
RESOFF EQU 3      OFFSET TO RESULT WD OF FUN DESCR.
NFLOFF EQU 7      OFFSET TO NO.OF FUN LINES WD IN A
*                               FUNCTION DESCRIPTOR.
*
*
*      CONSTANTS
*
THREE DATA 3
E0000000 DATA X'E0000000'      BITS 0,1,2
B1T011 DATA X'7FF00000'      BITS 1 TO 11.
NB1T013 DATA X'8003FFFF'      NOT BITS 1 TO 13.
NOTNBIT DATA X'FFFFFFF'      NOT N-BIT (BIT POS 7).
PAGE
*****
*
*      HASH -- HASHES A NEW NAME INTO THE ACTIVE SYMBOL TABLE.
*      REGS:  R14 -- LINK, EXIT VIA *R14.
*              R2 -- (ENTRY) PTS AT 1ST WD OF NAME.
*              R3 -- (ENTRY) NO.OF WORDS FOR THAT NAME.
*              R6 -- (EXIT) ACTIVE NAME PTR.
*              R2 AND R3 ARE VOLATILE.
*
*
HASH STW,R7 COPYSAVE+7      SAVE R7.
SLD,R2 2      USE BYTE ADDRESSING & BYTE COUNT.
LI,R6 0      SET R6=0 FOR LATER DIVISION.
STW,R6 HASHAQM      CLEAR HASH ACCUMULATOR.
LB,R7 0,R2 GET CHAR.
CI,R7 ' '      CK FOR BLANK...
BNE HASHC      NO, HASH IT.
HASHD AI,R2 -1      YES, BACK UP 1 CHAR.
LB,R2 0,R2 GET THAT (EBCDIC) CHAR.
LW,R7 HASHAQM      GET PARTIAL HASH VALUE.
SLS,R7 1      DOUBLE IT.
AW,R7 R2      ADD EBCDIC CHAR VALUE.
DW,R6 SYMTSIZE      DIVIDE BY SYM TBL SIZE.

```



```

SLS,R6      1      = TENTATIVE NAME PTR.
AI,R6       1      OFFSET FOR NAME-INDICATOR QUERYING.
HASHQ      LW,R7   *SYMT,R6  IS NAME-INDIC WD OCCUPIED...
           BNEZ    HASHI     YES, LOOK FOR A HOLE ELSEWHERE.
           LW,R7   COPYSAVE+7 NO, RESTORE R7.
           AI,R6   -1      = ACTIVE NAME PTR.
           B       *R14     EXIT.
HASHI      AI,R6   HASHINC+HASHINC INCREMENT THE HASH PTR.
           CW,R6   NSYMTWDS  ARE WE STILL WITHIN SYM TBL...
           BL      HASHQ     YES.
           SW,R6   NSYMTWDS  NO, CYCLE BACK IN.
           B       HASHQ
HASHC      LB,R7   IN2CODE,R7  GET CODE FOR THAT CHAR.
           AWM,R7  HASHAQM    ACCUMULATE PARTIAL HASH VALUE.
           AI,R2   1      PT AT NEXT CHAR, IF ANY...
           BDR,R3  HASHN     OK.
           B       HASHD     DONE.

PAGE
*****
*
* UCNP -- UPDATES A COPIED NAME PTR, REPLACING IT BY ITS ACTIVE ONE. *
* REGS:  R14 -- LINK, EXIT VIA *R14. *
*        R6  -- (ENTRY) CONTAINS THE COPY NAME PTR. *
*        (EXIT) CONTAINS THE ACTIVE NAME PTR. *
*        R7  -- VOLATILE. *
*
UCNP      AI,R6     2      CONVERT TO ODD COPY NAME PTR.
           SLS,R6   19
           OR,R6    SYMT    PT AT ACTIVE SYM TBL.
           LW,R7    B1T011  MASK FOR ODD COPY NAME PTR ONLY.
           B        UCNPQ   SEARCH ACTIVE SYM TBL FOR THE COPY
*                               NAME PTR, IT'S THERE SOMEWHERE.
UCNPU     AI,R6     2      PT AT NEXT ACTIVE SYM TBL ENTRY.
UCNPQ     CS,R6     0,R6   LOOK FOR COPY NAME PTR IN IT.
           BNE      UCNPU   NOPE, TRY AGAIN.
           AND,R6   X1FFFF  YEP.
           SW,R6    SYMT    GET ACTIVE NAME PTR (TO THIS ENTRY).
           B        *R14    EXIT.

PAGE
*****
*
* GNP -- GLOBAL NAME PROCESSOR.  R4 PTS AT COPY'S REFERENT DATA BLK. *
* R1 PTS AT COPY SYM TBL REF-INDICATOR WD. *
* GNEW -- ENTERED VIA LNP TO ACCOUNT FOR A NEW NAME IN ACTIVE SYM TBL. *
* REGS:  R14 -- LINK (DUAL RETURN USED BY LNP). *
*        R6  -- (ENTRY) PTS AT COPY'S REF-INDICATOR WD. *
*        (EXIT, IF NEW NAME & CORE-COPY) ACTIVE NAME PTR *
*        R10 -- (ENTRY) A-BIT ONLY. *
*        R14 IS VOLATILE. *
*
GNP      MTW,1      1,R4   BUMP REFERENT'S REF-COUNT.
           CW,R10    0,R1   TEST A-BIT OF COPY ENTRY...
           BANZ     *R14    SET -- RETURN.
           LW,R6     R1
GNEW     AWM,R10    0,R6   NOT SET -- SET NOW & ACCT NEW NAME
           MTW,-1    BLANKCNT DECR. NO.OF HOLES IN ACTIVE SYM TBL.
           BGEZ     G4      OK, THERE IS ROOM FOR THIS NAME.
           B        2MANYSYM OOPS -- TOO MANY SYMBOLS
G4       STW,R14    COPYSAVE+14 SAVE LINK.
           STD,R2    COPYSAVE+2 SAVE R2 & R3.
           LW,R2     1,R6   GET COPY'S NAME-INDICATOR WD.
           LB,R3     R2     GET BYTE 0 OF IT.
           LW,R14    CORVSFIL IS THIS A CORE-COPY...
           BEZ       G5      YES.
           AI,R3     -20     NO, FILE COPY. LONG NAME...
           BGZ       G9      SHORT NAME.
           MTW,1     -1,R2   LONG NAME, BUMP REF-COUNT OF THE
           B        G9      LONG NAME DATA BLK.
G5       AI,R6      1      PT AT COPY'S NAME-INDICATOR WD.
           STW,R6    COPYSAVE+6 SAVE THAT PTR.
           CI,R3     20     LONG OR SHORT NAME...
           BLE      G5L     LONG, R2 PTS AT IT.
           LI,R3     1      SHORT, 1 WD NAME.
           LW,R2     COPYSAVE+6 PT AT THAT NAME.
           B        G5S
G5L      MTW,1      -1,R2   BUMP REF-COUNT OF LONG-NAME DATA BLK
G5S      BAL,R14    HASH    HASH THE NEW NAME INTO ACTIVE SYMTBL
           LW,R2     *COPYSAVE+6 GET COPY'S NAME-INDICATOR WD.

```

```

*          STW,R6 *COPYSAVE+6          PUT ACTIVE NAME PTR IN THE RT HALF
*                                     OF THE COPY NAME-INDIC ENTRY.
*
*          LI,R14  ' '
*          STB,R14 *COPYSAVE+6          PUT A BLANK IN BYTE 0 OF IT.          U16-0006
*          LI,R3   X'40000'            SET W-BIT OF THE
*          STS,R3  *SYMT,R6             ACTIVE SYM TBL ENTRY.
*          AI,R6   1                    PT TO ACTIVE NAME-INDICATOR WD.
*          STW,R2  *SYMT,R6             PUT COPY'S NAME-INDICATOR THERE.
*          AI,R6   -1                   = ACTIVE NAME PTR AGAIN.
G9         LD,R2   COPYSAVE+2           RESTORE R2 & R3.
*          B       *COPYSAVE+14        EXIT.
*
PAGE
*****
*
* LNP -- LOCAL NAME PROCESSOR.
* LNP6 -- ENTERED WHEN R6 PTS AT COPY SYM TBL ENTRY.
* REGS:  R14 -- LINK DUAL RETRUN PREPARED:
*          RETURN-0 (CORE-COPY)
*          RETURN-1 (FILE-COPY)
*          R6 -- (ENTRY) COPY NAME PTR.
*          (EXIT) ACTIVE NAME PTR (IF CORE-COPY).
*          R10 -- (ENTRY) A-BIT ONLY.
*          R13 -- (ENTRY) L-BIT ONLY.
*          R14 IS VOLATILE AND R6 IS VOLATILE IF FILE COPY.
*          (SEE ALSO GNEW)
*
LNP        AW,R6   CSYMT                PT AT COPY SYM TBL ENTRY.
LNP6       SW,R14  CORVSFIL             SET RETURN 1 OR 0.
*          STS,R13 0,R6                 SET L-BIT OF COPY ENTRY.
*          CW,R10   0,R6                 TEST A-BIT OF COPY ENTRY...
*          BAZ      GNEW                 NOT SET -- ACCOUNT FOR NEW NAME.
*          LW,R6    1,R6                 SET, EXTRACT ACTIVE NAME PTR (IF
*          AND,R6   XFFFF                CORE-COPY) OR GARBAGE.
*          B        *R14                RETURN APPROPRIATELY.
*
PAGE
*****
*
* HUP -- HEADER NAME UPDATE ENTRY.  SEE ALSO UCNP.
* HNP -- HEADER NAME PROCEDURE ENTRY.  SEE ALSO LNP.
* THE HEADER OF A FUNCTION DESCRIPTOR INCLUDES:
*          RESULT
*          LF DUMMY
*          FUNCTION NAME
*          RT DUMMY
* IF ONE OF THESE IS NOT USED, ITS NAME PTR WORD IS MINUS ONE.
*
* REGS:  R12 -- LINK, EXIT VIA *R12.
*          R1 -- (ENTRY) PTS AT FUNCTION DESCRIPTOR DATA BLK.
*          (EXIT) PTS AT NO.OF FUN LINES WD IN THAT BLK.
*          R3, R4, R6, R14 ARE VOLATILE; SEE ALSO AS APPROPRIATE
*          UCNP OR LNP.
*
HUP        LI,R4   UCNP                ENTER UCNP FOR NAME PTR UPDATE.
*          B        HUPA
HNP        LI,R4   LNP                ENTER LNP FOR NAME PTR PROCEDURE.
HUPA       AI,R1   RESOFF              PT AT RESULT WD.
*          LI,R3    4                    4 NAME PTRS ARE POSSIBLE.
HUPU       LW,R6   0,R1                GET WD...
*          BLZ      HUPK                 NOT A NAME PTR.
*          BAL,R14  0,R4                 @ NAME PTR -- UPDATE OR PROCEDURE.
*          STW,R6   0,R1                 @ REPLACE NAME PTR BY ACTIVE ONE.
HUPK       AI,R1   1                    @ PT AT NEXT WD IN FUN DESCRIPTOR.
*          BDR,R3   HUPU                 LOOP TILL HEADER NAMES CHECKED.
*          B        *R12                EXIT.
*
PAGE
*****
*
* CSUP -- CODESTRING'S NAMES UPDATE ENTRY.  SEE ALSO UCNP.
* CSNP -- CODESTRING'S NAMES PROCEDURE ENTRY.  SEE ALSO LNP.
* REGS:  R12 -- LINK, EXIT VIA *R12.
*          R1 -- (ENTRY) PTS AT THE OFFSET WORD OF THE CODESTR.
*          DATA BLK.
*          R1,R3,R6,R14 ARE VOLATILE; SEE ALSO AS APPROPRIATE
*          UCNP OR LNP.
*
CSUP       LI,R3   UCNP                ENTER UCNP FOR NAME PTR UPDATE.
*          B        CSU1
CSNP       LI,R3   LNP                ENTER LNP FOR NAME PTR PROCEDURE.
CSU1       STW,R1  OFFSETK             SAVE PTR TO CODESTRING.

```

CSU2	LH, R1	*OFFSETK		GET OFFSET TO RTMOST CODE DESIGNATOR
	LB, R6	*OFFSETK, R1		GET A CODESTRING DESIGNATOR...
	CI, R6	NAMECODE		
	BG	CSU5		ABOVE NAME CODE -- SKIP IT.
	BE	CSUN		NAME CODE -- WORK ON NAME PTR.
	AI, R6	-10		
	BLZ	CSU5		0 THRU 9 -- SKIP IT.
	B	CSU4, R6		10 THRU 22 -- VECTOR ON THAT CODE.
CSU4	BDR, R1	CSU5	@10	SCALAR INTG (BYTE)
	B	CSUK4	@11	SCALAR INTG (WORD)
	B	CSUK8	@12	SCALAR REAL
	BDR, R1	CSU5	@13	SCALAR TEXT
	BDR, R1	CSUKL	@14	VECTOR LOGL
	BDR, R1	CSUKI	@15	VECTOR INTG
	BDR, R1	CSUKR	@16	VECTOR REAL
	BDR, R1	CSUKT	@17	VECTOR TEXT
	BDR, R1	CSUKTL	@18	VECTOR TEXT (LONG)
	BDR, R1	CSUKT	@19	ERROR STRING
	BDR, R1	CSUKT	@20	COMMENT
	B	CSUN	@21	STOPNAME
*	B	CSUN	@22	TRACENAME
CSUN	AI, R1	-1	@	OFFSET TO LO BYTE OF NAME PTR.
	LB, R14	*OFFSETK, R1		
	AI, R1	-1		OFFSET TO HI BYTE OF NAME PTR.
	LB, R6	*OFFSETK, R1		
	SLS, R6	8		
	OR, R6	R14		FORM COPY'S NAME PTR.
	BAL, R14	0, R3	@	DO NAME PTR UPDATE OR PROCEDURE.
	B	CSUNU	@	REPLACE COPY NAME PTR BY ACTIVE.
CSU5	AI, R1	-1	@	PT AT NEXT CODESTR.DESIGNATOR,
	CI, R1	1		IF ANY...
	BG	CSU2		OK.
	B	*R12		DONE -- EXIT.
CSUNU	AI, R1	+1		
	STB, R6	*OFFSETK, R1		REPLACE LO BYTE.
	AI, R1	-1		
	SLS, R6	-8		
	STB, R6	*OFFSETK, R1		REPLACE HI BYTE.
	B	CSU5		
CSUK8	AI, R1	-4		SKIP 4 BYTES.
CSUK4	AI, R1	-4		SKIP 4 BYTES.
	B	CSU5		
CSUKT	LB, R6	*OFFSETK, R1		GET LENGTH OF STRING.
	B	CSUKB		SKIP THAT MANY BYTES.
CSUKTL	LB, R14	*OFFSETK, R1		GET LO HALF OF LENGTH OF STRING.
	AI, R1	-1		
	LB, R6	*OFFSETK, R1		GET HI HALF.
	SLS, R6	8		
	OR, R6	R14		MERGE THEM.
	B	CSUKB		SKIP THAT MANY BYTES.
CSUKL	LB, R6	*OFFSETK, R1		GET LENGTH OF LOGL VECTOR.
	AI, R6	7		
	SLS, R6	-3		CALC. NO.OF BYTES FOR LOGL VALUE.
	B	CSUKB		SKIP THAT MANY BYTES.
CSUKR	LB, R6	*OFFSETK, R1		GET NO.OF DBLWDS FOR REAL VECTOR.
	SLS, R6	3		GET NO.OF BYTES.
	B	CSUKB		
CSUKI	LB, R6	*OFFSETK, R1		GET NO.OF WDS FOR INTG VECTOR.
	SLS, R6	2		GET NO.OF BYTES.
CSUKB	SW, R1	R6		SKIP TO LEFTMOST BYTE OF ITEM.
	B	CSU5		

PAGE

*			*
*	WINDI	-- READS A WORD FROM THE WINDOW.	*
*	WINDRD	-- ENTERED WHEN READING THE 1ST WORD OF WINDOW RECORDS.	*
*	REGS:	R14 -- LINK, EXIT VIA *R14 WITH CONDITION CODE SET	*
*		ACCORDING TO THE WORD JUST READ.	*
*		R3 -- (EXIT) CONTAINS THE WORD.	*
*		R2 -- (ENTRY) LOC OF EARLIER WORD IN WINDOW.	*
*		R2 IS UPDATED, NO OTHER REGS ARE AFFECTED.	*
*			*

WINDI	AI, R2	1		PT AT NEXT WINDOW WD.
WINDIA	LW, R3	0, R2		GET THAT WORD.
	BNEZ	*R14		EXIT UNLESS TRUNCATED WINDOW.
WINDRD	LCI	14		SAVE REGS.
	STM, R1	GARBSAVE+1		
	BAL, R14	RDNAMER		READ NAME RECORD INTO WINDOW.

```

        LCI      14
        LM,R1    GARBSAVE+1      RESTORE REGS.
        LI,R2    WINDOW          PT AT 1ST WINDOW WD.
        B        WINDIA

PAGE
*****
*
* WINDO -- PUTS A WD IN THE WINDOW; IF NEC. TRUNCATES & WRITES RECORD.
* REGS:      R14 -- LINK, EXIT VIA *R14.
*            R3  -- CONTAINS THE WD.
*            R2  -- OFFSET INTO THE WINDOW (FROM -511 TO 0).
*            R2 IS UPDATED, NO OTHER REGS. ARE AFFECTED.
*
WINDO      STW,R3    WINDOW+511,R2    PUT WD IN WINDOW.
           AI,R2     1                UPDATE OFFSET.
           BLZ      *R14             EXIT IF OFFSET DOESN'T REACH ZERO...
           STW,R2    WINDOW+511      TRUNCATE ON ZERO WD.
           LI,R2     -511            SET FOR FRESH WINDOW.
           LCI      14
           STM,R1    GARBSAVE+1      SAVE REGS.
           BAL,R14   WRNAMER         WRITE THE CURRENT WINDOW RECORD.
           LCI      14              RESTORE REGS.
           LM,R1     GARBSAVE+1
           B        *R14            EXIT.

PAGE
*****
*
* WINDON -- PUTS A SHORT OR LONG NAME IN THE WINDOW.
* REGS:      R14 -- LINK (MAY EXIT VIA 'WINDO').
*            R1  -- PTS AT SYM TBL ENTRY.
*            R3, R4, R12, R14 ARE VOLATILE.
* WINDON12 -- ENTRY TO PUT A LONG NAME IN THE WINDOW.
* REGS:      R14 -- LINK.
*            R1  -- PTS AT SYM TBL ENTRY.
*            R12 -- CONTAINS WORD COUNT OF LONG NAME.
*            R3, R4, R12, R14 ARE VOLATILE.
*
WINDON     LW,R3     1,R1            GET SYM TBL ENTRY'S NAME-INDIC. WD.
           LB,R12    R3              GET WORD COUNT IF LONG NAME...
           CI,R12    20
           BG        WINDO          SHORT, EXIT VIA 'WINDO'.
WINDON12   LW,R4     1,R1            LONG, PT AT LONG NAME'S 1ST WD.
           STW,R14   LINKWY         SAVE LINK.
WINDONL    LW,R3     0,R4            GET NAME WD.
           AI,R4     1              PT AT NEXT WD.
           BAL,R14   WINDO          PUT NAME WD IN WINDOW.
           BDR,R12   WINDONL        LOOP TILL DONE.
           B        *LINKWY        EXIT VIA SAVED LINK.

PAGE
*****
*
* SEEKN -- SEEKS NAME (POINTED TO BY R4) IN COPY SYM TBL.
* SEEKNI -- ENTRY PT WHEN R4 CONTAINS A NAME-INDICATOR.
* REGS:      R14 -- LINK EXITS: RETURN-0 IF NOT FOUND.
*            RETURN-1 IF FOUND.
*            R3  -- (ENTRY) NO.OF WORDS FOR NAME.
*            R4  -- (ENTRY) PTS AT 1ST WORD OF NAME (OR ON A SHORT
*            NAME USING SEEKNI-ENTRY, THAT NAME).
*            R6  -- (EXIT) PTS AT COPY SYM TBL ENTRY IF FOUND.
*            R12 & R14 ARE VOLATILE, R4 IS VOLATILE ON SHORT NAME.
*
SEEKN      CI,R3     1              LONG OR SHORT NAME...
           BE        SEEKNS         SHORT.
           B        SEEKNL         LONG.
SEEKNI     CI,R3     1              LONG OR SHORT NAME...
           BE        SEEKNS4        SHORT, IT IS IN R4.
SEEKNL     STD,R2    COPYSAVE+2     LONG.
           STD,R4    COPYSAVE+4     SAVE R2,R3,R4,R5.
           LW,R6     CSYMT
           AI,R6     1              PT AT 1ST COPY SYM TBL NAME-INDIC.
           LW,R12    IDBUF+1        GET NO.OF COPY SYM TBL ENTRIES.
           B        SEEKNLQ
SEEKNLU    AI,R6     2              PT AT NEXT NAME-INDICATOR.
SEEKNLQ    CB,R3     *R6            DO WORD-COUNTS MATCH...
           BNE      SEEKNLK         NO.
           LW,R2     0,R6           YES, PT AT 1ST WD OF COPY NAME.
           LW,R4     COPYSAVE+4     PT AT 1ST WD OF NAME SOUGHT.
SEEKNLW    LW,R5     0,R4          GET NAME WORD SOUGHT.

```

	CW, R5	0, R2	MATCH COPY NAME WORD...
	BNE	SEEKNL3	NO.
	AI, R4	1	YES, PT AT NEXT WORDS.
	AI, R2	1	
	BDR, R3	SEEKNLW	COMPARE NEXT WORDS, IF NOT DONE...
	AI, R14	1	BINGO, SET FOR RETURN-1, AND SET
SEEKNL3	BDR, R6	SEEKNLX	R6 TO POINT AT THE REF-INDIC.
	LW, R3	COPYSAVE+3	RESTORE WORD-COUNT.
SEEKNLK	BDR, R12	SEEKNLU	TRY AGAIN, IF NOT DONE...
SEEKNLX	LD, R2	COPYSAVE+2	DONE.
	LD, R4	COPYSAVE+4	RESTORE R2, R3, R4, R5.
	B	*R14	TAKE APPROPRIATE RETURN.
SEEKNS	LW, R4	0, R4	GET NAME SOUGHT.
SEEKNS4	LW, R6	CSYMT	PT AT 1ST COPY SYM TBL ENTRY.
	LW, R12	IDBUF+1	GET NO.OF COPY SYM TBL ENTRIES.
	B	SEEKNSQ	
SEEKNSU	AI, R6	2	PT AT NEXT COPY SYM TBL ENTRY.
SEEKNSQ	CW, R4	1, R6	MATCH...
	BNE	SEEKNSK	NO.
	AI, R14	1	BINGO.
	B	*R14	RETURN-1.
SEEKNSK	BDR, R12	SEEKNSU	TRY AGAIN, IF NOT DONE...
	B	*R14	RETURN-0.
PAGE			

*			
*	NAMEL -- MARKS (N-BIT) EVERY COPY SYM TBL ENTRY THAT MAY BE OF		
*	INTEREST BASED ON THE COPY CMND'S NAME-LIST.		
*	REGS:	R14 -- LINK.	
*	ALL REGS MAY BE CONSIDERED VOLATILE.		
*			
NAMEL	LW, R11	BITPOS-7	R11 = N-BIT.
	STW, R14	LINKCY	SAVE LINK.
	LW, R2	IMAGEPOS	GET BA(1ST NON-BLANK IN COPY-LIST).
	LB, R3	0, R2	
	AI, R3	-NEWLINE	IS THERE A LIST OF NAMES TO COPY...
	BNEZ	NB	YES.
	LW, R14	IDBUF+1	NO, COPY ALL.
	LW, R6	CSYMT	
COPYALL	STS, R11	0, R6	SET N-BIT FOR ALL COPY SYM TBL ENTS.
	AI, R6	2	
	BDR, R14	COPYALL	
	B	*LINKCY	EXIT.
NOTFND	LCI	11	SAVE MOST REGS.
	STM, R1	GARBSAVE+1	
	B	ERNOTFND	DISPLAY -- NAME NOT FOUND.
RNOTFND	LCI	11	
	LM, R1	GARBSAVE+1	RESTORE REGS.
	B	NCX	
NC1M	AI, R3	1	PUT CHAR IN IMAGE.
	STB, R2	0, R3	
	AI, R1	1	PT AT NEXT CHAR OF COPY-LIST.
	B	NC1	
NB	LI, R3	BA(CONSTBUF)	
	OR, R3	BITPOS-0	(= CHAR COUNT OF 128)
	MBS, R2	0	MOVE COPY-LIST FROM IMAGE TO
	LI, R1	BA(CONSTBUF)	CONSTBUF & PT AT 1ST CHAR.
NC	LI, R3	BA(IMAGE)-1	PRESET TO PUT NAME-CHARS IN IMAGE.
NC1	LB, R2	0, R1	GET CHAR FROM COPY-LIST.
	CI, R2	' '	CK FOR BLANK OR NEWLINE...
	BG	NC1M	NO, A NAME-CHAR.
	LI, R4	' '	YES.
NC2	AI, R3	1	
	STB, R4	0, R3	APPEND AT LEAST ONE BLANK.
	CI, R3	3	
	BANZ	NC2	BLANK TILL ON A WD BOUNDARY.
	AI, R3	-BA(IMAGE)	
	SLS, R3	-2	CALC. NO.OF WORDS FOR THE NAME.
	LI, R4	IMAGE	PT AT 1ST WORD OF NAME.
	BAL, R14	SEEKN	SEEK NAME IN COPY SYM TBL...
	B	NOTFND	NOT FOUND.
	LI, R4	X'1FFFF'	FOUND.
	AND, R4	0, R6	DOES IT REF. A GLOBAL...
	BEZ	NOTFND	NO.
	STS, R11	0, R6	YES, SET COPY ENTRY'S N-BIT.
NCX	BAL, R4	ACQNB	ACQ NON-BLANK FROM COPY-LIST.
	CI, R2	NEWLINE	IS THAT ALL FOR COPY-LIST...
	BNE	NC	NOT YET.

```

      B      *LINKCY      EXIT.
PAGE
*****
*
* XDACT -- DISPLACES COPIED DATA BLK PTRS IN THE ACTIVE SYM TBL
*          (COPIED REF-PTRS ARE INDICATED BY R-BIT AND LONG-NAME
*          PTRS BY W-BIT) AND REMOVES EXTRANEIOUS BITS FROM THE
*          ACTIVE SYM TBL.
* XACT -- ENTRY TO ONLY REMOVE EXTRANEIOUS BITS FROM ACTIVE SYM TBL.
* REGS:   R14 -- LINK, EXIT VIA *R14.
*          R1, R4, R5, R7, R8, R9, R12, R13 ARE VOLATILE.
*
XDACT   LI,R7   X2          SET LOOP CONTROL FOR DISPLACING.
        LW,R9   DISPLAC     GET THE DISPLACEMENT.
        LW,R12  BITPOS-12   R-BIT.
        LI,R13  X'40000'    W-BIT.
        B      X1
XACT    LI,R7   X3          SET LOOP CONTROL FOR REMOVAL ONLY.
X1      LI,R4   0
        LW,R5   NB1T013     MASK TO REMOVE BITS 1 - 13.
        LW,R1   SYMT        PT AT ACTIVE SYM TBL.
        LW,R8   SYMTSIZE    GET ITS SIZE.
        B      0,R7        START VIA LOOP CONTROL.
X2      CW,R12  0,R1        IS THERE A COPIED REF-PTR...
        BAZ     X21         NO.
        AWM,R9  0,R1        YES, DISPLACE IT.
X21     CW,R13  0,R1        IS THERE A COPIED LONG-NAME PTR...
        BAZ     X3         NO.
        AWM,R9  1,R1        YES, DISPLACE IT.
X3      LS,R4   0,R1        EXTRACT NON-EXTRANEIOUS BITS AND
        STW,R4  0,R1        REPLACE THE ENTRY.
        AI,R1   2          PT AT NEXT SYM TBL ENTRY.
        BDR,R8  0,R7        LOOP VIA LOOP CONTROL TILL DONE.
        B      *R14        EXIT.
PAGE
*****
*
* UNREF -- ELIMINATES A GLOBAL REFERENT, IF ANY, FROM THE ACTIVE
*          SYMBOL TABLE. IN THE CASE OF A SUSPENDED FUNCTION, UNREF
*          DAMAGES THE SUSPENSIONS & REPORTS SI DAMAGE.
* REGS:   R7 -- LINK, MAY EXIT VIA 'DREF'.
*          R2 -- (ENTRY) = 0.
*          R3 -- (ENTRY) = X'1FFFF'.
*          R4 -- (ENTRY & EXIT) NO BITS OTHER THAN ADDR. FIELD.
*          R5 -- (ENTRY) = X'1FFFF'.
*          R6 -- (ENTRY) IS THE ACTIVE NAME PTR.
*          R4 AND R14 ARE VOLATILE; SEE ALSO 'DREF'.
*
UNREF   LS,R4   *SYMT,R6     GET ACTIVE REF-PTR OR NIL.
        BEZ     0,R7        NIL -- EXIT QUICKLY.
        STS,R2  *SYMT,R6     REMOVE IT FROM ACTIVE SYM TBL.
        LB,R14  *R4         GET TYPE OF DATA BLK REFERENCED.
        CLM,R14 FUNTPYES    IS IT A DEFINED FUNCTION...
        BCS,9   DREF        NO, DE-REFERENCE IT & EXIT.
        LW,R14  1,R4        YES, GET ITS REF-COUNT.
        AI,R14  -1         DISCOUNT THE SYM TBL REF...
        BEZ     DREF        THAT'S ALL, DE-REFERENCE & EXIT.
        LCI     13         IT IS A SUSPENDED FUNCTION.
        STM,R1  GARBSAVE+1   SAVE REGS.
        LW,R6   STATEPTR    PT AT TOP STATE-ENTRY IN XEQ. STACK.
        B      UNREFQ       HUNT FOR THOSE SUSPENSIONS.
UNREFS  LI,R13  X'7FFF'
        AND,R13 0,R6        MOVE TO NEXT STATE-ENTRY STACKED.
        AW,R6   R13
UNREFQ  CS,R4   1,R6        DOES IT REFER TO OUR FUNCTION...
        BNE     UNREFS      NO.
        STW,R2  1,R6        YES, DAMAGE IT.
        BDR,R14 UNREFS      LOOP TILL LAST SUSPENSION DAMAGED.
        B      SIDAMC       ERROR -- SI DAMAGE -- RESUME RCSIDAM
RCSIDAM LCI     13
        LM,R1   GARBSAVE+1   RESTORE REGS.
        STW,R2  1,R4        ZERO THE REF-COUNT OF THE FUNCTION.
        B      DREF        DEREFERENCE IT (BYE-BYE) & EXIT.
PAGE
*****
*
* KRUNCH -- COALESCEs DATA BLKS TO BE COPIED.
* REGS:   R8 -- LINK, EXIT VIA GARBCOLL

```

```

*          R1  -- (ENTRY) PTS AT 1ST COPY BLK (DATA OR FREE).      *
*          R1, R3, R4, R7  ARE VOLATILE; SEE ALSO 'DREF'.          *
*                                                                    *
KRUNCH    LW,R4    DYNBOUND      SET BOUND TO KRUNCH REGION.      U16-0008
          STW,R4    COPYBND
          B          KRUNCHT
KRUNCHS   MTW,1    1,R1          BUMP REF-COUNT OF NEXT BLK &
          STW,R1    NILCK        SAVE ITS LOC IN CASE IT MOVES.
          STW,R1    COPYBASE     SAVE ALSO FOR MOVE CK.      U16-0012
          BAL,R7    DREF        PUT LATEST FREE BLK IN FREE TBL.
          LI,R1     0           CLEAR & GET PTR
          XW,R1     NILCK        TO NEXT BLK AGAIN,
          MTW,-1    1,R1        CORRECTING ITS REF-COUNT.
          CW,R1     COPYBASE     DID KRUNCH REGION MOVE...      U16-0014
          BE        KRUNCHT     NO.      U16-0015
          LW,R3     FREETBL     YES, SET NEW BOUND TO
          STW,R3    COPYBND      KRUNCH REGION.      U16-0016
          LW,R3     1,R1        IS THIS A DATA BLK OR FREE BLK... U16-0017
          BNEZ      KRUNCHR     DATA BLK.
          LW,R4     R1          FREE BLK, AIM AT IT.
          INT,R3    0,R1        GET ITS SIZE.
          AW,R1     R3          PT AT NEXT BLK, IF ANY...      U16-0020
          CW,R1     COPYBND
          BL        KRUNCHS     ANOTHER COMING UP.
          LI,R7     GARBCOLL     NO MORE, EXIT VIA GARBCOLL AFTER
          B          DREF        LAST FREE BLK GOES IN FREE TBL.
KRUNCHR   INT,R3   0,R1        GET SIZE OF DATA BLK.
          AW,R1     R3          PT AT NEXT BLK, IF ANY.
          CW,R1     COPYBND
          BL        KRUNCHT     ANOTHER COMING UP.      U16-0022
          B          GARBCOLL     NO MORE, EXIT AFTER GARBAGE
                                   COLLECTION.      U16-0023
*
PAGE
*****
*
*  SA -- SCANS ACTIVE SYM TBL:
*          SETS 'BLANKCNT' = NO.OF UNUSED ENTRIES.
*          IN CORE-COPY, LOOKS FOR ACTIVE NAMES IN THE COPY SYM TBL,
*          AND IF FOUND MODIFIES THE COPY ENTRY APPROPRIATELY.
*          IN FILE-COPY, WRITES ACTIVE NAME RECORDS, MARKING THEM
*          APPROPRIATELY.
*  REGS:   R14 -- LINK (VOLATILE)
*          R1  -- (EXIT) PTS AT 1ST WD AFTER ACTIVE SYM TBL.
*          R8  -- (EXIT) = 0.
*
*          ALL OTHER REGS MAY BE REGARDED AS VOLATILE.
*
SA         STW,R14  LINKCY      SAVE LINK.
          LI,R10    0           R10 = CLEAR BLANK COUNTER.
          LW,R7     CORVSFIL    R7  = 0 IF CORE-COPY, -1 IF FILE-COPY.
          LI,R5     X'1FFFF'    R5  = ADDRESS MASK.
          LW,R9     BITPOS-1    R9  = GR-BIT (GLOBAL REFERENT).
          LW,R11    BITPOS-2    R11 = PF-BIT (PENDENT FUNCTION).
          LW,R13    BITPOS-3    R13 = A-BIT (ACCOUNTED FOR IN ACTIVE TBL).
          LW,R1     SYMT        R1  = PTR TO ACTIVE ENTRY.
          LW,R8     SYMTSIZE    R8  = ACTIVE ENTRY COUNT TO GO.
SA31       LW,R4    1,R1        GET NAME-INDICATOR WORD...
          BEZ      SA38        UNUSED.
          LB,R3     R4          GET WORD-COUNT IF LONG NAME...
          CI,R3     20
          BLE      SA33,R7      LONG.
          LI,R3     1          SHORT, SET WC = 1.
          B        SA33,R7
          B        SA34
SA33       BAL,R14  SEEKNI      @-FILE.
          B        SA39        @-CORE, SEEK NAME IN COPY SYM TBL...
          EXU      SA34A,R7    @  NOT FOUND.
SA34       LI,R4    0           @  FOUND, SET A-BIT.
          LS,R4     0,R1        GET ACTIVE REF-PTR...
          BEZ      SA37,R7      NIL.
          EXU      SA34GR,R7    OK, SET GR-BIT.
          LB,R14    *R4        GET TYPE OF GLOBAL REFERENT...
          CLM,R14  FUNTPES
          BCS,9     SA37,R7
          LW,R14    1,R4        NOT A DEFINED FUNCTION.
          AI,R14    -1          DEFINE FUNCTION, GET REF-COUNT...
          BEZ      SA37,R7      ONLY 1 REF (IN SYM TBL NOW).
          LW,R7     STATEPTR    IT'S REFERENCED IN XEQ STACK.

```

	B	SA34PQ	SEE IF IT IS PENDENT.
	OR, R3	BITPOS-3	@-FILE.
SA34A	STS, R13	0, R6	@-CORE, MODIFY COPY SYM TBL ENTRY.
	OR, R3	BITPOS-1	@-FILE.
SA34GR	STS, R9	0, R6	@-CORE, MODIFY COPY SYM TBL ENTRY.
	OR, R3	BITPOS-2	@-FILE.
SA34PF	STS, R11	0, R6	@-CORE, MODIFY COPY SYM TBL ENTRY.
SA34PS	LI, R12	X'7FFF'	
	AND, R12	0, R7	GET OFFSET TO NEXT STATE ENTRY...
	BEZ	SA34PEND	NONE, FUNCTION MUST BE PENDENT.
	AW, R7	R12	OK, PT AT IT.
SA34PQ	CS, R4	1, R7	DOES THIS STATE ENTRY REF THE FUN...
	BNE	SA34PS	NO, TRY NEXT.
	LI, R12	PENDFLAG	YES, IS IT PENDENT...
	CW, R12	0, R7	
	BANZ	SA34PEND	YEP.
	BDR, R14	SA34PS	NOPE, LOOP UNLESS LAST SUSP.
	LW, R7	CORVSFIL	OK, SUSPENDED ONLY, RESTORE R7.
	B	SA37, R7	
SA34PEND	LW, R7	CORVSFIL	RESTORE R7.
	EXU	SA34PF, R7	SET PF-BIT.
	B	SA37, R7	
	B	SA37FILE	@-FILE.
SA37	LW, R2	R1	@-CORE.
	SW, R2	SYMT	GET ACTIVE NAME PTR (IN RT HALFWD).
	OR, R2	BITPOS-1	PUT 'BLANK' IN BYTE 0.
	STW, R2	1, R6	REPLACE COPY'S NAME-INDICATOR WD.
	B	SA39	
SA37FILE	BAL, R14	WINDO	PUT WORD-1 (I.E. R3) IN WINDOW.
	LI, R14	SA39	GO TO SA39 AFTER PUTTING
	B	WINDON	NAME WORD(S) IN THE WINDOW.
SA38	AI, R10	1	BUMP BLANK COUNT.
SA39	AI, R1	2	PT AT NEXT ENTRY AND
	BDR, R8	SA31	LOOP IF ANY.
	STW, R10	BLANKCNT	SET BLANK COUNT (HOLES IN ACTIVE
	B	*LINKCY	SYM TBL). EXIT VIA SAVED LINK.

PAGE

*			*
*	ACDYN -- ACQUIRE COPY DYNAMIC:		*
*	ALLOCATE SPACE.		*
*	DECLARE THAT SPACE TO BEGIN A SINGLE LARGE FREE BLK		*
*	AT THE END OF CURRENT DYNAMIC.		*
*	READ COPY'S DYNAMIC INTO THE ALLOTED SPACE.		*
*	SET CSYMT TO PT AT THE COPIED SYM TBL.		*
*	SET DISPLAC TO THE DISPLACEMENT BETWEEN COPIED DATA		*
*	BLK PTRS AND THEIR NEW VALUES.		*
*	ENTER RLOC TO ZERO THE REF-COUNTS OF ALL COPIED DATA		*
*	BLKS USING		*
*	RB-ENTRY IF DISPLAC = 0		*
*	RC-ENTRY IF DISPLACEMENTS ARE NECESSARY.		*
*	REGS: R12 -- LINK, EXIT IS IN THE RLOC ROUTINE.		*
*	R1 -- (EXIT) PTS AT 1ST DATA BLK IN COPIED DYNAMIC.		*
*	R2 -- (EXIT) LOOPING LOCATION FOR RLOC.		*
*	SEE ALSO THE RLOC ROUTINE, IN GENERAL ALL REGS		*
*	SHOULD BE CONSIDERED VOLATILE WHEN CALLING		*
*	ACDYN.		*
*			*
ACDYN	STW, R12	LINKCY	SAVE LINK.
	LW, R11	COPYSIZE	GET SIZE OF COPY DYNAMIC.
	AI, R11	-2	DATA BLK HDR IS UNNEC.
	BAL, R14	ALOCNONX	@ ALLOC SPACE FOR COPY DYNAMIC.
	BAL, R15	SYSTERR	@ WS FULL SHOULD BE IMPOSSIBLE.
	LW, R11	FREETOTL	@ GET SIZE OF FREE BLK AFTER SPACE...
	BEZ	AD	NONE.
	STW, R11	*FREETBL	MAKE THAT FREE BLK BE A LARGE FREE
*			BLK (TYPE = 0 & SIZE = FREE TOTAL)
*			ITS REF-COUNT WILL BE ZEROED LATER
*			IN RLOC.
AD	STW, R4	FREETBL	FREE EVERYTHING FROM THE ALLOTED
	LW, R6	DYBOUND	SPACE ONWARD AS A SINGLE FREE BLK,
	SW, R6	FREETBL	THIS WILL PREVENT COPY DYNAMIC
	STW, R6	FREETBL+1	FROM MOVING TILL DESIRED.
	STW, R6	FREETOTL	
	BAL, R14	RDCOPY	READ COPY'S DYNAMIC INTO THAT BLK.
	LW, R1	FREETBL	
	AW, R1	IDBUF+0	SET CSYMT TO PT AT 1ST WD OF
	STW, R1	CSYMT	COPY SYM TBL.


```

        LW,R9    FREETBL
        SW,R9    IDBUF+4
        STW,R9   DISPLAC
        BNEZ     AG
        AW,R1    IDBUF+1
        AW,R1    IDBUF+1
        LI,R2    RB
        B        RB
*
AG      LI,R5    X'1FFFF'
        LW,R14   IDBUF+1
AGQ     LW,R4    1,R1
        BEZ     AGU
        AND,R4   E0000000
        BNEZ    AGR
        AWM,R9   1,R1
AGR     CW,R5    0,R1
        BAZ     AGU
        AWM,R9   0,R1
AGU     AI,R1    2
        BDR,R14  AGQ
        LI,R2    RC
        B        RC
*
PAGE
*****
*
*  RLOC -- THIS ROUTINE, WHICH IS SHARED BY ACDYN, DRIVES ONE OF 3
*  PROCESSES:
*
*      1 - TO ZERO THE REF-COUNTS OF ALL COPIED DATA BLKS.
*
*      2 - TO ZERO THE REF-COUNTS OF ALL COPIED DATA BLKS AND
*          DISPLACE EACH COPIED DATA BLK PTR WITHIN THE COPIED
*          LISTS OR FUNCTIONS.
*
*      3 - TO DISPLACE EACH COPIED DATA BLK PTR WITHIN THE
*          COPIED LISTS OR FUNCTIONS AND TO REPLACE COPY NAME
*          PTRS BY ACTIVE NAME PTRS IN GROUPS, FUNCTIONS, AND
*          CODESTRING FOR THOSE FUNCTIONS.
*
*  RLOC IS THE ENTRY FOR PROCESS 3; R12 IS THE LINK, R9 = DISPLAC.
*  RB IS THE ENTRY PT FOR PROCESS 1; R9=0 & R2=RB UPON ENTRY.
*  RC IS THE ENTRY PT FOR PROCESS 2; R14=0, R9=DISPLAC, & R2=RC (ENTRY)
*  REGISTER 1 PTS AT THE 1ST COPIED DATA BLK UPON ALL ENTRIES.
*
*  ALL REGS SHOULD BE PRESUMED VOLATILE.
*
RLOC    STW,R12  LINKCY      SAVE LINK.
        LI,R2    RD          R2 = LOOP CONTROL.
        B        RD
RB      STW,R9   1,R1        ZERO THE REF-COUNT.
        B        RE          LOOP FOR NEXT DATA BLK.
RC      STW,R14  1,R1        @ ZERO THE REF-COUNT.
RD      LB,R4    *R1         @ GET TYPE OF DATA BLK.
        B        RDV,R4     VECTOR ON THE TYPE:
RDV     B        RE          @ 0 -- FREE.
        B        RE          @ 1 -- LOGL.
        B        RE          @ 2 -- TEXT.
        B        RE          @ 3 -- INTG.
        B        RE          @ 4 -- REAL.
        B        RE          @ 5 -- INDEX SEQUENCE.
        B        RL          @ 6 -- LIST.
        B        RCS-RD,R2   @ 7 -- CODESTRING.
        B        RF          @ 8 -- FUNCTION.
        B        RF          @ 9 -- FUNCTION.
        B        RF          @ 10 -- FUNCTION.
        B        RF          @ 11 -- FUNCTION.
        B        RF          @ 12 -- FUNCTION.
        B        RF          @ 13 -- FUNCTION.
        B        RE          @ 14 -- INTRINSIC FUNCTION.
        B        RE          @ 15 -- INTRINSIC FUNCTION.
        B        RE          @ 16 -- INTRINSIC FUNCTION.
        B        RG-RD,R2    @ 17 -- GROUP.
        B        RE          @ 18 -- LONG NAME.
*
*  NOTE -- RCS MUST FOLLOW A
*          B        RE          @ RC-PROCESS.
*          IN ORDER TO DRIVE RC-
*          PROCESSING RATHER THAN
*          RD-PROCESSING.
RCS     STW,R1   COPYSAVE    SAVE DATA BLK PTR.
        AI,R1    2           PT AT OFFSET WD OF CODESTRING BLK.

```

	BAL, R12	CSUP	UPDATE CODESTRING'S NAME PTRS.
	LW, R1	COPYSAVE	RESTORE DATA BLK PTR.
RE	INT, R3	0, R1	GET SIZE OF CURRENT BLK.
	AW, R1	R3	PT AT NEXT BLK, IF ANY...
	CW, R1	DYNBOUND	
	BL	0, R2	YES, GOTO RB, RC, OR RD.
	B	*LINKCY	NO -- EXIT VIA SAVED LINK.
RL	LW, R3	LISTOFF, R1	GET LENGTH OF LIST...
	BEZ	RE	ZERO, SKIP IT.
	AI, R1	LISTOFF	PT AT THE WD BEFORE 1ST PTR IN IT.
RLQ	LW, R4	*R1, R3	TEST FOR DATA BLK PTR...
	BEZ	RLK	NIL.
	AWM, R9	*R1, R3	YES, DISPLACE IT.
RLK	BDR, R3	RLQ	LOOP TILL DONE.
	AI, R1	-LISTOFF	PT AT DATA BLK HDR AGAIN.
	B	RE	
RF	STW, R1	COPYSAVE	SAVE PTR TO FUNCTION DATA BLK HDR.
	LW, R3	XSIZEOFF, R1	GET FUNDEF'S XSIZE WD.
	CI, R3	X'E0000'	DOES IT PT AT ERR-CTRL TBL...
	BAZ	RFNFL	NO.
	AWM, R9	XSIZEOFF, R1	YES, DISPLACE PTR TO ERR-CTRL TBL.
RFNFL	LW, R3	NFLOFF, R1	GET NO.OF FUNCTION LINE PTRS...
	BEZ	RFN-RD, R2	NONE, PROCEED VIA LOOP CONTROL.
	AI, R1	NFLOFF	PT AT WD BEFORE 1ST PTR.
RFDP	AWM, R9	*R1, R3	DISPLACE EACH LINE PTR.
	BDR, R3	RFDP	
	AW, R1	0, R1	PT AT LAST FUNCTION LINE PTR.
	LI, R3	X'FFFF'	
	AND, R3	1, R1	GET NO.OF LABEL ENTRIES...
	BEZ	RFN-RD, R2	NONE, PROCEED VIA LOOP CONTROL.
RFDLBL	AI, R1	2	PT AT LABEL-ENTRY.
	AWM, R9	1, R1	DISPLACE IT.
	BDR, R3	RFDLBL	LOOP TILL LABEL DATA BLK PTRS DONE.
	B	RFN-RD, R2	PROCEED VIA LOOP CONTROL.
	B	RFX	@-RC.
RFN	LW, R1	COPYSAVE	@-RD, PT AT FUNCTION DATA BLK HDR.
	BAL, R12	HUP	UPDATE HEADER NAME PTRS.
	AW, R1	0, R1	PT AT LAST FUNCTION LINE PTR.
	LW, R3	1, R1	GET NO.OF LOCALS & LABELS...
	BEZ	RFX	NONE.
	STW, R3	NAMEWDS	SAVE IT.
	AND, R3	XFFFF	GET NO.OF LABELS...
	BEZ	RFLCLS	NONE.
RFLBL	AI, R1	2	PT AT LABEL ENTRY.
	LW, R6	0, R1	GET LABEL'S NAME PTR.
	BAL, R14	UCNP	GET ACTIVE VERSION OF THAT NAME PTR.
	STW, R6	0, R1	
	BDR, R3	RFLBL	
RFLCLS	LH, R3	NAMEWDS	GET NO.OF LOCALS...
	BEZ	RFX	NONE.
	AI, R1	2	PT AT 1ST WD OF LOCAL NAME PTRS.
	SLS, R1	1	USE HALFWD RESOLUTION.
RFLCL	LH, R6	0, R1	GET LOCAL'S NAME PTR.
	BAL, R14	UCNP	GET ACTIVE VERSION OF THAT NAME PTR.
	STH, R6	0, R1	
	AI, R1	1	
	BDR, R3	RFLCL	
RFX	LW, R1	COPYSAVE	PT AT DATA BLK HDR AGAIN.
	B	RE	@-RC.
RG	STW, R1	COPYSAVE	@-RD, SAVE PTR TO GROUP DATA BLK HDR.
	AI, R1	2	PT AT LENGTH OF GROUP.
	SLS, R1	1	(HALFWD)
	LH, R3	0, R1	GET LENGTH OF GROUP...
	BEZ	RGX	ZERO.
RGU	AI, R1	1	PT AT GROUP'S NAME PTR.
	LH, R6	0, R1	GET IT.
	BAL, R14	UCNP	GOT IT.
	STH, R6	0, R1	GOOD.
	BDR, R3	RGU	
RGX	LW, R1	COPYSAVE	PT AT DATA BLK HDR AGAIN.
	B	RE	
PAGE			
COPY	BAL, R8	GARBCOLL	GARBAGE COLLECT.
	BAL, R14	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.
	LW, R14	TOPOSTAK	CALC. SIZE AVAILABLE IN ACTIVE.
	SW, R14	FREETBL	
	AI, R14	-4	(EXTRA 4 WDS DEMANDED BY ALOCBLK).
	SW, R14	COPYSIZE	SUBTRACT SIZE OF COPY DYNAMIC.

	SAS, R14	-31	EXTEND SIGN OF RESULT SO THAT
	STW, R14	CORVSFIL	CORVSFIL = 0 IF CORE-COPY.
	BLZ	FA	CORVSFIL = -1 IF FILE-COPY.
	BAL, R12	ACDYN	ACQ COPY DYNAMIC, ZERO ITS REF-CNTS.
	BAL, R14	NAMEL	PROCESS COPY CMND'S NAME-LIST.
	LI, R14	DCOPY	GO TO DCOPY AFTER
	B	SA	SCANNING ACTIVE SYM TBL; RETURN
			IS TO CD, EVENTUALLY.
*			
*			
*	UPON ENTRY TO CD:	R1	= PTR TO COPY SYM TBL'S 1ST WD.
*		R2	= 0
*		R4	= 0
*		R5	= X'1FFFF'
*		R8	= NO.OF COPY SYM TBL ENTRIES.
*		R11	= THE N-BIT.
*			
CD	LI, R3	X'1FFFF'	
CD2	CW, R11	0, R1	IS THIS ITEM TO BE COPIED...
	BAZ	CD5	NO.
	LW, R6	1, R1	YES, GET ITS ACTIVE NAME PTR.
	BAL, R7	UNREF	ELIM ANY ACTIVE GLOBAL REFERENT.
	LI, R7	X'1FFFF'	LOC OF COPY'S GLOBAL REFERENT
	AND, R7	0, R1	TOGETHER WITH THE R-BIT (MARKING
	OR, R7	BITPOS-12	AN ALIEN REF-PTR FOR ACTIVE)
	STS, R7	*SYMT, R6	ARE PUT IN ACTIVE ENTRY.
CD5	AI, R1	2	PT AT NEXT COPY SYM TBL ENTRY,
	BDR, R8	CD2	LOOPING IF ANY.
	STW, R1	COPYBASE	SAVE LOC OF 1ST BLK IN COPY DATA REG
	BAL, R8	GARBCOLL	GARBAGE COLLECT ACTIVE WS, COPY WILL
*			NOT YET MOVE.
*	BAL, R14	XACT	REMOVE EXTRANEIOUS BITS (INCLUDING
			R AND W BITS) FROM ACTIVE SYM TBL.
	LW, R1	COPYBASE	
	SW, R1	DYNBOUND	= - SIZE OF COPY DATA BLK REGION.
	AWM, R1	FREETBL+1	DECLARE THAT REGION TO BE ALLOCATED.
	AWM, R1	FREETOTL	I.E. NO LONGER FREE.
	LW, R1	COPYBASE	PT AT 1ST BLK IN THAT REGION.
	BAL, R8	KRUNCH	COALESCE THOSE BLKS THAT ARE COPIED,
*			FREING UNCOPIED BLKS SEQUENTIALLY
CI	LI, R1	IDBUF	SET UP FOR COPYDMES.
	LI, R14	COPYDMES	ISSUE THE SAVED MESSAGE & EXIT AFTER
	B	XWLOCGLB	RE-EXCHANGING LOCALS & GLOBALS.
FA	BAL, R14	WRACTIV	OPEN TEMP FILE & WRITE ACTIVE WS.
	LI, R14	0	
	STW, R14	NAMERKEY	
	LI, R2	-511	SET TO WINDOW ACTIVE NAME RECORDS.
	BAL, R14	SA	SCAN ACTIVE SYM TBL, WINDOWING.
	LI, R3	-1	END THE ACTIVE NAME RECORDS.
	STW, R3	WINDOW+511, R2	
	BAL, R14	WRNAMER	WRITE THE LAST ACTIVE NAME RECORD.
	BAL, R11	CLEARMEM	CLEAR OUT ACTIVE WS.
	LW, R11	HICOMMON	
	AI, R11	-STACKOFF	
	STW, R11	TOPOSTAK	
	STW, R11	STATEPTR	
	LW, R3	LODYN	FREE ALL OF DYNAMIC.
	BAL, R5	SETFREET	
	BAL, R12	ACDYN	ACQ COPY DYNAMIC, ZERO ITS REF-CNTS.
	LW, R12	CSYMT	
	STW, R12	SYMT	MAKE CURRENT SYM TBL MATCH COPY'S.
	LW, R12	IDBUF+1	
	STW, R12	SYMFSIZE	
	AW, R12	SYMFSIZE	
	STW, R12	NSYMTWDS	
	LI, R8	0	PREPARE TO READ ACTIVE NAME RECORDS.
	STW, R8	NAMERKEY	
	LI, R14	FG1	GO TO FG1 AFTER
	B	WINDRD	READING 1ST ACTIVE NAME WD.
FG2	STB, R3	NAMEWDS	SAVE WORD COUNT.
	LI, R9	X'FFF00'	EXTRACT GR, PF, AND A-BITS.
	AND, R9	R3	
	LB, R8	NAMEWDS	
	LI, R4	NAMEBUF	
	BAL, R14	WINDI	MOVE NAME WORDS
	STW, R3	0, R4	INTO NAMEBUF.
	AI, R4	1	
	BDR, R8	WINDI	
	LI, R4	NAMEBUF	

	LB, R3	NAMEWDS		
	BAL, R14	SEEKN	@	SEEK NAME IN COPY SYM TBL...
	B	FG	@	NOT FOUND.
	STS, R9	0, R6	@	FOUND, SET GR, PF, A-BITS THERE.
	MTB, -1	NAMEWDS		LONG OR SHORT NAME...
	BEZ	FG		SHORT.
	LW, R4	1, R6		LONG, BUMP REF-COUNT OF COPY'S
	MTW, 1	-1, R4		LONG NAME DATA BLK.
FG	BAL, R14	WINDI		GET NEXT WINDOW WD.
FG1	BGZ	FG2		LOOP UNLESS END-WORD (WHICH = -1).
	BAL, R14	NAMEL		PROCESS COPY CMND'S NAME-LIST.
DCOPY	LW, R9	BITPOS-6	R9 =	X-BIT (EXAMINED).
	LW, R13	BITPOS-4	R13 =	L-BIT (LOCAL NAME).
	LW, R10	BITPOS-3	R10 =	A-BIT (ACCOUNTED FOR IN ACTIVE).
	LW, R11	BITPOS-7	R11 =	N-BIT (NAME OF INTEREST).
DB	LW, R8	IDBUF+1		GET COPY SYM TBL SIZE.
	STW, R8	BACKUP		PRESET BACKUP FLAG NON-ZERO.
	LW, R1	CSYMT		PT AT 1ST COPY SYM TBL ENTRY.
DC	LB, R2	*R1		
	AND, R2	THREE		GET X & N BITS:
	B	DCV, R2		- -
DCV	B	DD	@	0 0
	B	DE	@	0 1 -- NEEDS EXAMINATION.
	B	DD	@	1 0
DD	AI, R1	2	@	1 1
	BDR, R8	DC		LOOP TILL COPY SYM TBL SCANNED.
	LW, R2	BACKUP		IS ANOTHER SCAN NEEDED...
	BEZ	DB		YES.
	B	ZAP		NO, GO TO ZAP WITH R1 POINTING AT
*				THE FIRST DATA BLK IN COPY.
DE	STS, R9	0, R1		SET THE X-BIT NOW.
	LW, R4	0, R1		GET THE REF-INDICATOR WD.
	CW, R4	BITPOS-1		DOES THIS NAME HAVE ACTIVE GLOBALREF
	BAZ	DF		NO.
	CW, R9	PCOPYFLG		IS THIS A PCOPY...
	BANZ	DE6		YES, DON'T CLOBBER THE ACTIVE REF.
	CW, R4	BITPOS-2		IS ACTIVE GLOBAL REF A PENDENT FUN.
	BAZ	DF		NO.
	AND, R4	X1FFFF		YES, IS THERE A COPY REFERENT...
	BEZ	DF1		NO SWEAT.
	LCI	13		YES, SAVE A BUNCH OF REGS,
	STM, R1	GARBSAVE+1		
	LW, R2	1, R1		LOOK AT COPY'S NAME-INDIC WD.
	LB, R3	R2		
	AI, R3	-X'40'		IS IT AN ACTIVE NAME PTR TYPE...
	BNEZ	NOTCOPYD		NO, R1 PTS 1 WD BEFORE NAME-INDIC.
	INT, R1	1, R1		YES, GET ACTIVE NAME PTR.
	B	ERNOTCPY		DISPLAY -- NAME NOT COPIED.
NOTCOPYD	SW, R1	SYMT		GET NAME PTR (SYMT = CSYMT).
	B	ERNOTCPY		DISPLAY -- NAME NOT COPIED.
RNOTCPY	LCI	13		
	LM, R1	GARBSAVE+1		AND RESTORE THE REGS.
DE6	BAL, R14	NBITOFF		TURN OFF N-BIT SO WON'T COPY REF.
DF	AND, R4	X1FFFF		IS THERE A COPY REFERENT...
	BNEZ	DG		YES.
DF1	LI, R14	DD		RESUME SCAN AFTER TURNING N-BIT OFF.
NBITOFF	LW, R2	NOTNBIT		GET ALL BUT N-BIT.
	AND, R2	0, R1		EXTRACT FROM COPY REF-INDIC WD.
	STW, R2	0, R1		RESTORE ALL BUT N-BIT WHICH = ZERO.
	B	*R14		EXIT.
DG	LB, R2	*R4		GET TYPE OF REFERENT:
	CI, R2	TYPELIST		
	BL	DG1		(LOGL, TEXT, INTG, REAL, XSEQ)
	BE	DG2		(LIST)
	CLM, R2	FUNTYPES		
	BCR, 9	DG4		(DEFINED FUNCTION)
	CI, R2	TYPEGRP		
	BE	DG3		(GROUP)
DG1	CW, R11	0, R1		(OTHER) IS ITEM TO BE COPIED...
	BAZ	DD		NO (PCOPY OR PENDENT CONFLICT).
	LI, R14	DD		RESUME SCAN AFTER
	B	GNP		DOING GLOBAL NAME PROCEDURE.
DG2	CW, R11	0, R1		WILL WE COPY THIS LIST...
	BAZ	DD		NO, MOVE OUT QUICKLY.
	BAL, R14	GNP		YES, DO GLOBAL NAME PROCEDURE.
DG2A	AI, R4	LISTLOFF		PT AT LENGTH WD OF LIST.
	LW, R7	0, R4		GET LENGTH...
	BEZ	DG2D		EMPTY.

DG2B	AI, R4	1	PT AT PTR WD IN LIST.	
	LW, R3	0, R4	GET THAT PTR...	
	BEZ	DG2C	NIL.	
	MTW, 1	1, R3	OK, BUMP THAT DATA BLK'S REF-COUNT	
	CB, R2	*R3	IS IT TOO A LIST...	
	BNE	DG2C	NO.	
	LW, R5	LISTLOFF, R3	YES, GET ITS LENGTH.	
	AW, R3	R5		
	STW, R4	LISTLOFF+1, R3	SET CHAIN = LOC OF PTR TO SUB-LIST.	
	SLS, R7	17		
	STS, R7	LISTLOFF+1, R3	PUT LENGTH VALUE OF HOST IN CHAIN WD	
	LW, R4	0, R4	PT AT SUB-LIST DATA BLK HDR.	
	B	DG2A		
DG2C	BDR, R7	DG2B	LOOP TILL DONE WITH LIST'S PTR WDS.	
DG2D	AI, R4	1	PT AT THIS LIST'S CHAIN WD.	
	LI, R7	0		
	XW, R7	0, R4	CLEAR & GET IT...	
	BEZ	DD	THAT'S ALL FOLKS.	
	LI, R4	X'1FFFF'	RESTORE HOST'S PTR.	
	AND, R4	R7		
	SLS, R7	-17	RESTORE HOST'S LENGTH VALUE.	
	B	DG2C		
DG3	CW, R11	0, R1	IS GROUP DEFN TO BE COPIED...	
	BAZ	DG3L	NO, BUT GROUP'S REFS. MAY BE.	
	BAL, R14	GNP	YES, DO GLOBAL NAME PROCEDURE.	
DG3L	AI, R4	2	PT AT GROUP'S LENGTH WD.	
	LH, R7	*R4	GET ITS LENGTH...	
	BEZ	DD	EMPTY.	
	SLS, R4	1	USE HALFWD RESOLUTION.	
DG3A	AI, R4	1		
	LH, R6	0, R4	GET COPY NAME PTR.	
	AW, R6	CSYMT	PT AT COPY SYM TBL REF-INDIC. WD.	
	CW, R9	0, R6	HAS THAT NAME BEEN EXAMINED YET...	
	BANZ	DG3B	YES.	
	STS, R11	0, R6	NO, SET ITS N-BIT.	
	CW, R6	R1	WILL IT BE NECESSARY TO BACK UP...	
	BGE	DG3B	NO, IT WILL APPEAR ON THIS SCAN.	
	LI, R5	0	YES, WE WILL NEED ANOTHER SCAN.	
	STW, R5	BACKUP		
DG3B	BAL, R14	LNP6	@ DO LOCAL NAME PROCEDURE.	
	STH, R6	0, R4	@ REPLACE COPY NAME PTR BY ACTIVE.	
	BDR, R7	DG3A	@ LOOP TILL DONE.	
	B	DD		
DG4	CW, R11	0, R1	WILL WE COPY THIS FUNCTION...	
	BAZ	DD	NO.	
	BAL, R14	GNP	(DO GLOBAL NAME PROCEDURE)	
	STW, R1	COPYSAVE	YES, SAVE MASTER PTR.	
	LW, R7	XSIZEOFF, R4	GET FUNDEF'S XSIZE WD.	
	CI, R7	X'E0000'	DOES IT PT AT ERR-CTRL TBL...	
	BAZ	DG4HDR	NO.	
	MTW, 1	1, R7	YES, BUMP REF-CNT OF ERR-CTRL DB.	
DG4HDR	LW, R1	R4	MAKE R1 PT AT FUNDEF DATA BLK.	
	BAL, R12	HNP	DO HEADER NAME PROCEDURE.	
	STW, R1	COPYSAVE+1	SAVE PTR TO FUNDEF'S LENGTH WD, AND	
	LW, R7	0, R1	GET NO.OF FUNCTION LINES...	
	BEZ	DG4B	NONE.	
DG4A	LW, R1	*COPYSAVE+1, R7	GET A CODESTRING PTR.	
	MTW, 1	-1, R1	BUMP CODESTR.DATA BLKS REF-COUNT.	
	LW, R12	-1, R1	TEST THAT REF-COUNT...	U16-0025
	AI, R12	-1	IF NOT 1, THIS CS HAS ALREADY HAD	U16-0026
	BNEZ	DG4A2	ITS NAME PTRS ADJUSTED.	U16-0027
	BAL, R12	CSNP	DO CODESTRING NAME PROCEDURE.	U16-0028
DG4A2	BDR, R7	DG4A	LOOP TILL DONE WITH CS PTRS.	U16-0029
DG4B	LW, R1	COPYSAVE	RESTORE MASTER PTR.	
	LW, R4	COPYSAVE+1	PT AT FUNDEF'S LENGTH WD AGAIN.	
	AW, R4	0, R4	PT AT LAST CODESTRING PTR WD.	
	LW, R7	1, R4	GET NO.OF LOCALS & LABELS...	
	BEZ	DD	NONE.	
	STW, R7	NAMEWDS	SAVE FOR LOCALS.	
	AND, R7	XFFFF	EXTRACT NO.OF LABELS...	
	BEZ	DG4LCLS	NONE.	
DG4LBL	AI, R4	2	PT AT LABEL ENTRY.	
	LW, R5	1, R4	GET PTR TO LABEL'S DATA BLK, AND	
	MTW, 1	1, R5	BUMP ITS REF-COUNT.	
	LW, R6	0, R4	GET COPY NAME PTR FOR THE LABEL.	
	BAL, R14	LNP	@ DO LOCAL NAME PROCEDURE.	
	STW, R6	0, R4	@ REPLACE COPY NAME PTR BY ACTIVE.	
	BDR, R7	DG4LBL	@	

DG4LCLS	LH, R7	NAMEWDS	GET NO.OF LOCALS.
	BEZ	DD	NONE.
	AI, R4	2	
	SLS, R4	1	PT AT 1ST LOCAL NAME PTR (HALFWD).
DG4LCL	LH, R6	0, R4	GET COPY NAME PTR FOR THE LOCAL.
	BAL, R14	LNP	@ DO LOCAL NAME PROCEDURE.
	STH, R6	0, R4	@ REPLACE COPY NAME PTR BY ACTIVE.
	AI, R4	1	@ PT AT NEXT HALFWD.
	BDR, R7	DG4LCL	LOOP TILL DONE.
	B	DD	
ZAP	LW, R2	1, R1	CK REF-COUNT OF DATA BLK...
	BNEZ	ZAPUP	NON-ZERO, DATA BLK TO BE COPIED.
	STB, R2	*R1	0 (DON'T COPY), CHANGE ITS TYPE TO
*			BE A FREE BLK SO ANY INTERNAL
*			PTRS WILL BE DISREGARDED. THIS
*			PREVENTS CONFUSION DURING ANY
*			GARBAGE COLLECTION PROCESS.
ZAPUP	INT, R7	0, R1	GET SIZE OF THIS BLK.
	AW, R1	R7	PT TO NEXT BLK,
	CW, R1	DYNBOUND	IF ANY...
	BL	ZAP	YES.
	LI, R2	0	NO, DO SOME REGISTER SET-UPS.
	LI, R4	0	
	LI, R5	X'1FFFF'	
	LW, R1	CSYMT	(PT AT COPY SYM TBL)
	LW, R8	IDBUF+1	(GET NO.OF ENTRIES IN IT)
	LW, R14	CORVSFIL	IS THIS A CORE-COPY OR FILE-COPY...
	BEZ	CD	CORE-COPY.
*			FILE-COPY.
	AI, R1	1	PT AT 1ST NAME-INDIC WD.
FK1	CW, R11	-1, R1	IS N-BIT SET...
	BAZ	FK3	NO, DON'T COPY A REFERENT.
	LS, R4	-1, R1	YES, REMOVE EXTRANEIOUS BITS,
	STW, R4	-1, R1	KEEPING THE REF-PTR.
	B	FK4	
FK3	CW, R13	-1, R1	IS L-BIT SET...
	STW, R2	-1, R1	(CLEAR REF-INDICATOR WD NOW)
	BANZ	FK4	YES, KEEP THE NAME-INDICATOR.
	LB, R14	*R1	NO, GET BYTE 0 OF NAME-INDIC.
	BEZ	FK4	UNUSED ENTRY.
	AI, R14	-20	LONG OR SHORT NAME...
	BGZ	FK3NIX	SHORT.
	LW, R7	0, R1	LONG, PT AT IT.
	STW, R2	-1, R7	ZERO THE REF-COUNT OF LONG NAME DATA
*			BLK, SO IT WON'T BE COPIED.
FK3NIX	STW, R2	0, R1	CLEAR THE NAME-INDICATOR.
FK4	AI, R1	2	PT AT NEXT NAME-INDIC WD,
	BDR, R8	FK1	IF ANY, LOOPING BACK.
	AI, R1	-1	PT AT 1ST DATA OR FREE BLK.
	STW, R8	NAMERKEY	ZERO THE KEY FOR WRITING COPY RECDS.
	LW, R3	DYNBOUND	
	STW, R3	FREETBL	CLAIM ALL OF
	STW, R8	FREETBL+1	COPY DYNAMIC IS
	STW, R8	FREETOTL	ALLOCATED, AT PRESENT.
	BAL, R8	KRUNCH	FREE UNWANTED BLKS, COALESCING THE
*			BLKS TO BE COPIED INTO A SINGLE
*			COPY DATA REGION
*			(FOLLOWS THE COPY SYM TBL).
	LW, R1	CSYMT	PT AT COPY SYM TBL & GET ITS SIZE;
	LW, R8	IDBUF+1	WE WILL WRITE COPY NAME RECORDS.
	LI, R2	-511	SET INIT. OFFSET INTO WINDOW.
FN2	LW, R3	1, R1	GET NAME-INDICATOR WD.
	BEZ	FN3	NAME OF NO INTEREST.
	BAL, R14	WINDO	PUT IT IN WINDOW.
	LI, R13	0	CLEAR W-FLAG.
	LB, R12	R3	IS THIS A LONG OR SHORT NAME...
	CI, R12	20	
	BG	FN2E	SHORT, NAME WAS IN NAME-INDIC WD.
	LI, R13	X'40000'	LONG, SET W-FLAG = W-BIT.
	BAL, R14	WINDON12	MOVE THE NAME WORDS INTO WINDOW.
FN2E	LW, R3	R1	
	SW, R3	CSYMT	CALCULATE THE ODD COPY NAME PTR.
	AI, R3	2	
	SLS, R3	19	
	OR, R3	R13	MERGE IN W-FLAG.
	OR, R3	0, R1	MERGE REF-PTR, IF ANY.
	LW, R14	0, R1	TEST FOR REF-PTR...
	BEZ	FN2F	NO.

FN2F	OR, R3	BITPOS-12	YES, MERGE R-BIT.
FN3	BAL, R14	WINDO	PUT REF-INDIC ETC IN THE WINDOW.
	AI, R1	2	PT AT NEXT ENTRY.
	BDR, R8	FN2	LOOP TILL DONE.
	STW, R1	COPYBASE	SAVE LOC OF 1ST WD OF COPY DATA REG.
	LI, R3	-1	END THE COPY NAME RECORDS.
	STW, R3	WINDOW+511, R2	
	BAL, R14	WRNAMER	WRITE THE LAST PHYSICAL RECORD.
	LW, R14	FREETBL	
	SW, R14	COPYBASE	CALC. SIZE OF COPY DATA REGION.
	STW, R14	COPYSIZE	
	BAL, R14	WRCOPYDR	WRITE THE COPY DATA REGION.
	BAL, R14	RDACTIV	READ IN THE ACTIVE WS AGAIN.
	LI, R14	0	SET TO READ 1ST COPY NAME RECORD.
	STW, R14	NAMERKEY	
	LI, R14	FP1	GO TO FP1 AFTER
	B	WINDRD	READING THE 1ST WORD.
FP2	STW, R3	COPYSAVE	SAVE COPY NAME-INDIC.
	LB, R7	COPYSAVE	GET BYTE 0 OF IT.
	CI, R7	20	LONG OR SHORT NAME...
	BG	FP2S	SHORT (NAME IS IN R3).
	LI, R1	IMAGE	LONG, SET TO MOVE IT INTO IMAGE.
	BAL, R14	WINDI	GET 1ST NAME WORD.
	STW, R3	0, R1	PUT NAME WORD IN IMAGE.
	AI, R1	1	PT AT NEXT WD IN IMAGE.
	BDR, R7	WINDI	LOOP TILL NAME HAS BEEN MOVED.
	B	FP3	
FP2S	STW, R3	IMAGE	PUT THE NAME IN IMAGE.
	LI, R1	IMAGE+1	PT AT NEXT WD IN IMAGE.
FP3	LI, R3	' '	TERMINATE THE NAME WITH A COMMA.
	STB, R3	*R1	
	BAL, R14	WINDI	GET COPY REF-INDIC, ETC.
	STD, R2	COPYSAVE+2	SAVE IT & WINDOW OFFSET.
	LI, R1	BA(IMAGE)	PT AT 1ST CHAR OF NAME.
	BAL, R4	ACQCC	ACQ CHAR & CODE.
	BAL, R8	FINDNAME	@ FIND NAME IN ACTIVE SYM TBL...
	B	FP8	@ FRESH NAME.
	LW, R2	BITPOS-12	@ ACTIVE NAME.
	CW, R2	COPYSAVE+3	DOES COPY NAME HAVE A REFERENT...
	BAZ	FP5	NO, LEAVE ACTIVE REF ALONE.
	LI, R2	0	YES.
	LI, R3	X'1FFFF'	
	LI, R4	0	DO REG. SET-UPS.
	LI, R5	X'1FFFF'	
	BAL, R7	UNREF	REMOVE ACTIVE REFERENT, IF ANY.
FP5	LB, R7	COPYSAVE	
	AI, R7	-20	LONG OR SHORT NAME...
	BGZ	FP9	SHORT.
	AI, R6	1	PT TO NAME-INDICATOR OF SYM TBL ENT.
	LI, R4	0	CLEAR AND GET
	XW, R4	*SYMT, R6	LONG NAME PTR (ACTIVE ONE).
	AI, R4	-2	PT AT ITS DATA BLK HDR. GET RID OF
	BAL, R7	DREF	THAT ONE, WE'LL USE COPIED ONE.
FP8	AI, R6	-1	PT TO REF-INDIC WD OF SYM TBL ENTRY.
FP9	AW, R6	SYMT	PT AT THAT WD DIRECTLY.
	LD, R2	COPYSAVE+2	RESTORE OFFSET & COPY REF-INDIC.
	STS, R3	0, R6	MERGE INTO REF-INDIC OF SYM TBL:
*			THE ODD COPY NAME PTR (BITS 1-11),
*			THE R-BIT VALUE (BIT 12),
*			THE W-BIT VALUE (BIT 13), AND
*			THE COPY REF-PTR (BITS 15-31).
	LW, R8	COPYSAVE	REPLACE ACTIVE NAME-INDICATOR WD BY
	STW, R8	1, R6	COPY'S NAME-INDIC.
	BAL, R14	WINDI	READ NEXT WORD IN COPY NAME RECORD.
FP1	CI, R3	-1	IS THAT THE END-WORD...
	BNE	FP2	NO -- A COPY NAME-INDICATOR.
	BAL, R8	GARBCOLL	YES, GARBAGE COLLECT ANY FREED
	LW, R11	COPYSIZE	BLOCKS.
	BEZ	NOCOPY	(OOPS, NOTHING TO COPY)
	AI, R11	-2	HDR IS UNNEC.
	BAL, R14	ALOCNONX	@ ALLOCATE SPACE FOR COPY DATA REGION.
	B	TOOBIG	@ WS FULL -- TOO BIG.
	STW, R4	COPYHOME	@ OK, SAVE LOC TO GET 1ST COPIED WD.
	BAL, R14	RDCOPYDR	READ COPY DATA REGION & CLOSE TEMP
	LW, R9	COPYHOME	FILE. CALC. DISPLACEMENT OF COPY
	SW, R9	COPYBASE	LOCATIONS INTO ACTIVE LOCS.
	STW, R9	DISPLAC	
	LW, R12	FREETOTL	GET SIZE OF FREE BLK AFTER COPY D.R.

	BEZ	FQ	NONE.
FQ	STW, R12	*FREETBL	SET FREE BLK HDR, TYPE=0, SIZE=TOTAL
	LW, R1	COPYHOME	PT AT 1ST COPIED DATA BLK.
*	BAL, R12	RLOC	CORRECT COPIED DATA BLK PTRS &
			NAME PTRS IN THE COPY DATA REGION.
	LI, R14	CI	GO TO CI TO FINISH UP AFTER
*	B	XDACT	REMOVING EXTRANEIOUS BITS AND
*			DISPLACING COPY REF-PTRS &
			LONG-NAME PTRS IN THE SYM TBL.
NOCOPY	LI, R5	F:TF	NO COPY, CLOSE TEMP FILE AND
	LI, R6	CI	GO TO CI FOR WRAP UP.
	B	CLOSR	
TOOBIG	LI, R8	IDTOOBIG	= ERROR I.D. FOR 'TOO BIG'.
	STW, R8	ERRORID	
	B	ABORTFIL	ABORT THE FILE-COPY ATTEMPT.
2MANYSYM	LI, R8	ID2NAMEY	= ERROR I.D. FOR 'TOO MANY SYMBOLS'.
	STW, R8	ERRORID	
ABORTCPY	LW, R14	CORVSFIL	ABORT CORE-COPY OR FILE-COPY...
	BNEZ	ABORTFIL	FILE-COPY.
	LI, R10	X'40000'	CORE-COPY, GET W-BIT,
	LI, R9	X'C0000'	AND ITS NEGATIVE.
	LW, R1	SYMT	WE HAVE TO DELETE ANY NEW NAMES IN
	LW, R8	SYMFSIZE	THE ACTIVE SYM TBL + ANY W-BITS.
ABORT1	CW, R10	0, R1	IS THIS A NEW ENTRY...
	BAZ	ABORT2	NO.
	AWM, R9	0, R1	YES, SUBTRACT OUT THE W-BIT.
	STW, R14	1, R1	ZERO THE NAME-INDICATOR WD.
ABORT2	AI, R1	2	PT AT NEXT SYM TBL ENTRY, LOOP IF
	BDR, R8	ABORT1	ANY. NOTE THAT THE COPIED DYNAMIC
*			AREA WILL DISAPPEAR AUTOMATICALLY
*			SINCE IT WAS DECLARED FREE.
ABORTOUT	BAL, R14	XWLOCGLB	RE-EXCHANGE LOCALS & GLOBALS.
	LW, R8	ERRORID	GET ERROR I.D.
	B	CMDERR	COMMAND ERROR EXIT.
ABORTFIL	BAL, R14	RDACTIV	READ IN THE ACTIVE WS AGAIN.
	LI, R5	F:TF	
	LI, R6	ABORTOUT	GO TO ABORTOUT AFTER
	B	CLOSR	CLOSE & RELEASE OF TEMP FILE.

PAGE

SPACE 2			
Z	SET	%-WSCOPY@	SIZE OF WSCOPY IN HEX.
SPACE 1			
Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536	SIZE IN DECIMAL.
SPACE 3			
	END		


```

TITLE      'CMD-B00,09/20/73,DWG702985'
SYSTEM     SIG7F
CSECT      1
PCC         0
CMD@       RES      0  CONTROL CARDS NOT PRINTED.
          *          ORIGIN OF CMD MODULE W/HOOKS TO WSCOPY & UTSC MODS.
          *

```

```

*  REF'S  AND  DEF'S
*

```

DEF	CMD@	= START OF 'CMD' MODULE.	
DEF	@OFF	OFF COMMAND PROCESSOR.	
DEF	KCLEAR	CLEAR COMMAND PROCESSOR.	
DEF	@CLEAR	CLEAR CMD ASSUMED, CLEARS WORKSPACE.	
DEF	SETFREET	SET-UP FREE TABLE	
DEF	@SAVE	SAVE COMMAND PROCESSOR.	
DEF	SAVEOPEN	OPEN FOR SAVE	
DEF	SAVEREL	RESUME AFTER CODESTR'G AUTOSTART.	
DEF	UNSYMS	GETS NO.OF UNUSED SYMBOLS.	
DEF	WRITEWS	ROUTINE TO WRITE WS ONTO FILE	
DEF	CMDRETRN	RETURN TO CMD MODULE.	
DEF	@LOAD	LOAD COMMAND PROCESSOR	
DEF	LOADCONT	LOAD CONTINUE WS AT START-UP.	
DEF	READWS	ROUTINE TO READ WS FROM FILE	
DEF	ACCTCHK	ACCOUNT CHECK ROUTINE	17-00001
DEF	@WSID	WSID COMMAND PROCESSOR.	
DEF	@DROP	DROP COMMAND PROCESSOR.	
DEF	@COPY	COPY COMMAND PROCESSOR.	
DEF	@PCOPY	PCOPY COMMAND PROCESSOR.	
DEF	@LIB	LIB COMMAND PROCESSOR.	
DEF	@SYMBOLS	SYMBOLS COMMAND PROCESSOR.	
DEF	@OPRN	OPRN COMMAND PROCESSOR.	
DEF	@OPR	OPR COMMAND PROCESSOR.	
DEF	@TERM	TERMINAL COMMAND PROCESSOR.	
DEF	@TABS	TABS COMMAND PROCESSOR.	
DEF	@CONTINU	CONTINUE COMMAND PROCESSOR.	
DEF	CONTOFF	'CONTINE'-OFF	

```

SPACE 2

```

```

*
REFS TO PROCEDURE:
REF  ACQIT      ACQUIRES NAME OR NUMERIC ITEM.
REF  ACQNB      ACQUIRES NON-BLANK, MAY BE ON IT NOW
REF  ACQNXCC    ACQUIRES NEXT CHAR & ITS CODE.
REF  ACQNXNB    ACQUIRES NEXT NON-BLANK CHAR.
REF  BAQONLY    ACQ. CONSTANT ONLY & RESUME IN CMD.
REF  BCSN       CODESTRING & RESUME IN CMD.
REF  B4CLOSE    FORCE CLOSE OF FN & RESUME IN CMD.
REF  CALTIME    GET CURRENT TIME-DATE
REF  CINTIM     CONVERT INTEGER TO IMAGE
REF  CLEARMEM   CLEAR WS MEMORY TO MINIMUM
REF  CLOSV      ROUTINE TO CLOSE DCB
REF  CMDERR     COMMAND ERROR.
REF  CMDEXIT    COMMAND EXIT.
REF  COPY       PERFORMS COPY COMMAND.
REF  DELAY6     DELAY 6 SECONDS
REF  DREF       DEREFERENCE DATA BLK R4 PTS AT.
REF  DROPFILE   ROUTINE TO DELETE FILE
REF  DUMPLING   LINE OUTPUT ROUTINE
REF  ERBADCMD   BAD COMMAND.
REF  ERBDCMDA   BAD COMMAND WITH F:WS OPEN.
REF  FNEQWSID   SET FID=WSID
REF  GARBCOLL   GARBAGE COLLECT
REF  GDIGITS    DIGITS FOR GRAPHICS SCALE
REF  GENIS      GENERATES 'IS ' & VALUE IN R7.
REF  GENOPRM
REF  GETCOM
REF  GETDYN     GET DYNAMIC CORE
REF  GETTERM    GET TERMINAL TRANSLATE TABLES
REF  INPDIR     REQUESTS & HANDLES DIRECT INPUT.
REF  LIBUTS     UTS INT. FOR )LIB PROCESSING
REF  LOADREAD   READ A RECORD FOR LOAD
REF  OPENSARE   OPEN FILE FOR SAVE
REF  OUTORANG   EXIT (DUAL) FOR COMMON CMDS.
REF  QTABVLID   CHECK TAB VALIDITY
REF  QUIETFLG   ZERO IF SAVED MSG IS TO BE DISPLAYED
REF  RELEASER   RELEASE MEMORY AS POSSIBLE
REF  RESACCT    RESET ACCOUNT
REF  RESPASS    RESET PASSWORD
REF  SAVEDMES   ISSUE SAVED MESSAGE AND EXIT
REF  SAVWRITE   WRITE RECORD FOR SAVE
REF  SETACCT    SET ACCOUNT

```

REF	SETFNAME	SET FILE NAME
REF	SETPASS	SET PASSWORD
REF	SNEWTABS	SET NEW TAB VALUES
REF	STASHBL	
REF	TESTACCT	TEST ACCOUNT VS USERS'
REF	TESTOLDF	TEST FOR OLD FILE EXISTENCE
REF	TEXTC2I	MOVES TEXTC MSG TO IMAGE.
REF	TRYLOAD	TRY TO START 'LOAD'
REF	UTSIOFF	UTS INTERFACE ROUTINE FOR 'OFF'.
REF	UTSIOFFH	UTS INTERFACE ROUTINE FOR 'OFF HOLD'
REF	VALOUT	PUTS VALUE IN IMAGE, DISPLAYS, EXITS
REF	WRTEXTC	WRITE TEXTC MESSAGE
REF	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.
*		REFS TO CONTEXT:
REF	BREAKFLG	BREAK OR 'HANGUP' FLAG
REF	CLEARsiz	SIZE OF 'CLEAR' PARAMETER TABLE
REF	CATCHTBL	CATCH TBL.
REF	CMNDTYPE	COMMAND TYPE
REF	COMPRMST	START OF COMMON PARAMETERS TABLE
REF	CONSTAD	ZERO-WD USED BY ACQCONST.
REF	CONSTBUF	CONSTANT BUFFER
REF	COPYSIZE	SET TO SIZE OF COPY WS DYNAMIC AREA.
REF	CORAVAIL	AVAILABLE CORE (TOTAL)
REF	CURRCS	CURRENT CODESTRING POINTER
REF	DWIDTH	DEFAULT WIDTH
REF	DYNBOUND	HIGHEST ADDR. IN DYNAMIC REGION.
REF	ERRLOC	ERROR LOCATION.
REF	ERRNUM	ERROR NUMBER.
REF	ERRORID	INTERNAL ERROR I.D.
REF	F:WS	F:WS DCB ADDRESS (MONITOR INTERFACE)
REF	FREETBL	FREE TABLE (2-WD ITEMS-- LOC & AMT).
REF	FREETOTL	TOTAL AMT OF FREE DYNAMIC SPACE.
REF	GRAFPNTR	PTR TO SPECIAL D.B. FOR GRAPHICS.
REF	HICOL	CHARACTER COUNT
REF	HICOMMON	HIGHEST ADDR. IN COMMON REGION.
REF	HOLDFLG	HOLD-OFF FLAG;FOR CONTINUE
REF	IDBUF	TRYLOAD PUTS WS HEADER IN THIS BLK.
REF	IMAGE	IMAGE BUFFER
REF	IMAGEPOS	
REF	LOADCASE	LOAD CASE FLAG
REF	LOCOMMON	LOW COMMON ADDRESS
REF	LODACCT	LOAD-ACCOUNT
REF	LODYN	LOWEST ADDR. IN DYNAMIC REGION.
REF	MODE	EXECUTION MODE
REF	NAMEBUF	HOLDS ITEM ACQUIRED BY ACQIT.
REF	NEWBOUND	NEW DYNAMIC BOUND
REF	NSYMTWDS	NO.OF WORDS IN SYM TBL (2X SYMTSIZE)
REF	OBSFLAG	OBSERVE FLAG.
REF	OBSERVE	OBSERVATION SETTING
REF	ON%OFF	ON-LINE,OFF-LINE MODE FLAG
REF	OPWSACT	WS ACCOUNT
REF	OPWSNAME	WS NAME FOR 'OPEN'
REF	OPWSPAS	WS PASSWORD
REF	PCOPYFLG	PROTECTED COPY FLAG
REF	QUITFLAG	FLAG TO CHECK REPEAT 'CONTINUE'
*		IN OFF-LINE MODE
REF	SAVE312	SAVE AREA FOR APLINPUT
REF	SICTRL	STATE-INDICATOR CONTROL FLAG.
REF	STATEPTR	SI STATE POINTER
REF	STRAYS	DATA BLK PTRS OUTSIDE THE WORKSPACE.
REF	SYMT	LOC OF 1ST WD IN SYMBOL TABLE.
REF	SYMTSIZE	NO.OF ENTRIES IN SYMBOL TABLE.
REF	TABPNTR	TABSET INDICATOR
REF	TABVALS	TAB VALUES (BYTES)
REF	TERMTYPE	TERMINAL TYPE
REF	TOPOSTAK	CURRENT ADDRESS-TOP OF EXEC. STACK
REF	USERACCT	USER ACCOUNT NO.
REF	WIDTH	PRINT WIDTH
REF	WINDOW	BUFFER FOR)LOAD
REF	WSIDNAME	WORKSPACE NAME (TEXTC)
REF	WSIDPASS	AND PASSWORD (TEXT)
REF	WSOFFSET	WORKSPACE OFFSET
*		REFS TO CONSTANTS:
REF	APLVERSN	VERSION ID
REF	BITPOS	32-WD TBL OF BITS (ONLY BIT K IS = 1 IN THE WORD 'BITPOS-K').
*		
REF	BLANKS	
REF	CLEARWS	NAME-CLEAR WS


```

B          ERBADCMD          OOPS -- INCORRECT COMMAND.
PAGE

*
* @CLEAR-SET CLEAR WORKSPACE-EXIT TO INPDIR
*
KCLEAR    AI,R2    -NEWLINE          DID CLEAR I.D. END ON END-OF-INPUT.
          BNEZ     ERBADCMD          NO -- INCORRECT COMMAND.
@CLEAR    RES      0                YES, CLEAR USER'S WORKSPACE.
          BAL,R11  CLEARMEM          CLEAR COMMON AND DYN-SET LIMITS
          LI,R1    NOMSYMS**1       = NO.OF WDS IN NOMINAL SYMBOL TABLE.
          STW,R1   NSYMTWDS
          LI,R1    NOMSYMS          = NO.OF ENTRIES IN NOMINAL SYM TBL.
SETSYM    STW,R1   SYMTSIZE
          LI,R3    SYMTOFF          OFFSET FROM LODYN TO 1ST SYM TBL WD.
          STW,R3   *LODYN           THAT'S THE 1ST WD IN DYNAMIC.
          AW,R3    LODYN            PT AT 1ST WD OF SYMBOL TABLE.
          STW,R3   SYMT
          LI,R9    3
          STW,R9   GDIGITS          SET DIGITS FOR GRAPHICS
          LI,R9    -1
          STW,R9   CATCHTBL         CLEAR THE CATCH TABLE.
          STW,R9   CATCHTBL+2
          LW,R8    BITPOS-0         SET 'GLOBAL' BIT (ONLY).
          LI,R9    0                NOTE--R9 REMAINS ZERO THRUOUT CLEAR DRIVER
          STW,R9   ERRLOC           INIT ERROR LOCATION.
          STW,R9   ERRNUM           ERROR NUMBER.
          STW,R9   ERRORID          INTERNAL ERROR I.D.
          STW,R9   OBSFLAG          OBSERVE FLAG.
          STW,R9   SICTRL           STATE-INDICATOR CTRL = ON.
CLRSYM    STW,R8   0,R3            INIT A REFERENT-INDICATOR WD &
          STW,R9   1,R3            A NAME-INDICATOR WD.
          AI,R3    2                PT AT WD AFTER CLEARED NAME-INDIC.
          BDR,R1   CLRSYM           SET UP A CLEAR SYMBOL TABLE.
          LI,R5    SETSTOFF         SET EXIT
                                     @@@@@@@@
*                                     @
* SETFREET-SET UP FREETABL AND FREETOTL,AND CLEAR STRAYS          @
* R5=LINK R3=1ST FREE ADDRESS                                     @
* R3 IS CHANGED                                                  @
* R1,R8,R9 VOLATILE                                             @
*                                                                 @
SETFREET  STW,R3   FREETBL          ROUTINE TO INITIALIZE FREETBL@@@@@@@@
          LD,R8    ZEROZERO
          LI,R1    -MAXFRENS+1      (EXC 1ST ENTRY, OTHERS MUST BE 0).
CLRFR     STD,R8   FREETBL+MAXFRENS+MAXFRENS,R1  CLEAR FREE TABLE, ALMOST
          BIR,R1   CLRFR
          LW,R3    DYNBOUND         GET AMT OF FREE DYNAMIC.
          SW,R3    FREETBL
          STW,R3   FREETBL+1       SET SIZE-WD OF 1ST FREE TABLE ENTRY.
          STW,R3   FREETOTL        THAT'S THE TOTAL FREE DYNAMIC, TOO.
*
          STW,R9   CONSTAD          THIS CELL SHOULD ALWAYS BE ZERO.
*
          LI,R1    -STRAYBLK        = - TOTAL NO.OF STRAY DATA BLK PTRS.
CLRSTRAY  STW,R9   STRAYS+STRAYBLK,R1  THESE DATA BLK PTRS HAVE TO BE 0,
          BIR,R1   CLRSTRAY         UNTIL THEY PT TO A DATA BLOCK.
          B        0,R5
*
SETSTOFF  LI,R3    -STACKOFF        OFFSET FROM HICOMMON TO EXEC. STACK
          STW,R3   *HICOMMON        THAT'S THE LAST WD IN COMMON.
          AW,R3    HICOMMON        PT AT BEGINNING OF EXECUTION STACK.
          LW,R1    FINAL            SET 1ST STACK ENTRY TO BE THE
          STW,R1   0,R3             'FINAL' F-ENTRY.
          STW,R3   TOPOSTAK         SET STACK POINTER.
          STW,R3   STATEPTR         SET STATE POINTER.
          LI,R3    CLEARSTZ
SETCLEAR  LW,R4    CLEARSTZ-1,R3     INITIALIZE FUZZ,ORIGIN,DIGITS
          STW,R4   COMPRMST-1,R3    WIDTH,AND RANDOM SEED
          BDR,R3   SETCLEAR
          STD,R3   WSIDPASS         RESET WSID PASSWORD
          LW,R4    DWIDTH           SET WIDTH TO
          STW,R4   WIDTH            DEFAULT FOR TERMINAL
          LCI      3
          LM,R4    CLEARWS
          STM,R4   WSIDNAME
          LW,R4    USERACCT
          LW,R5    USERACCT+1      (NOT ON DW BOUND)
          STD,R4   LODACCT          RESET LOAD ACCOUNT ID
          LW,R8    CMNDTYPE        CHECK IF

```

```

        AI,R8      NCMD5-@@CLEAR+CMDB      REACHED BY CLEAR COMMAND
        BNEZ      OLDSYMS                  NO-)SYMBOLS
CLEARWSM LI,R8      CLEARWS
        BAL,R7     WRTEXTC                  ISSUE CLEAR WS MESSAGE
        LW,R7      BREAKFLG                CHECK FOR HANG-UP
        BLZ        UTSIOFFH                YES-QUIT
        B          INPDIR                  GO TO DIRECT INPUT
        PAGE

*
* @SAVE-PROCESSOR FOR SAVE COMMAND
*
*      ENTERED ON RECOGNITION OF 'SAVE',WITH SCAN READY FOR NEXT ITEM
*
*      FINAL EXIT TO COMMON COMAND EXIT AFTER EXECUTION OF
*      COMMAND OR ERROR MESSAGE GENERATION
*
*      MOST REGISTERS USED-NONE PRESERVED
*
*      LAYOUT OF WORKSPACE ID RECORD-1ST 13 WORDS
*
*      0-SYMTOFF
*      1-SYMTSIZE
*      2-FREETBL
*      3-TOPOSTAK
*      4-LODYN
*      5-HICOMMON
*      6-APLVERSN
*      7-APLVERSN+1
*      8-TIME
*      9- AND
*      10- DATE
*      11- IN EBCDIC
*      12-CMNDTYPE OR CURRCS(IF NON-ZERO)-FLAG USED TO INDICATE
*      AUTOCONTINUE OR AUTOSTART
*
@SAVE   LI,R13      0
        STW,R13     ERRORID                  CLEAR INTERNAL ERROR I.D.
        CI,R2       X'15'                    CHECK FOR CR
        BE          NOFID                    YES-NO FID
        BAL,R13     ACQFID                    NO-ESTABLISH FID(OR QUIT)
SAVETACT BAL,R14     TESTACCT                  CHECK IF USERS ACCOUNT
SAVECR   BAL,R14     TESTOLDF                  TEST IF OLD FILE-IF SO,CHECK WSID
SAVEOPEN LW,R14      MODE                     CHECK IF FUNCTION OPEN
        BNEZ        OPENOUT                  NO
        CI,R2       X'5E'                    CHECK IF SAVE FOR AUTOLOAD
        BE          ERBDCMD                  YES-NOT PERMITTED
        BAL,R14     B4CLOSE                  YES-FORCE CLOSE
OPENOUT  BAL,R14     QCLEAR                    CHECK IF CLEAR WS
        BE          CLEARWSM                 YES-DON'T SAVE
        BAL,R14     OPENSARE
        BAL,R8      GARBCOLL                  GARBAGE COLLECT
        CI,R2       X'5E'                    CHECK IF SAVE FOR AUTO-START
        BNE        SAVEREL                  NO
        LI,R2       1                        YES
        CW,R2       MODE                     CHECK IF DIRECT INPUT MODE
        BNE        ERBDCMD                  NO-ERROR
        BAL,R4      ACQNXNB                  GET CHARACTER AFTER ';'
        CI,R2       X'7F'
        BE          ERBDCMD                  QUIT ON FUNCTION OPEN
        CI,R2       X'75'                    OR
        BE          ERBDCMD                  LOCKED FUNCTION OPEN
        CI,R2       X'5D'                    OR
        BE          ERBDCMD                  COMMAND
        B          BCSN                      CODESTRING THE 'AUTOSTART' LINE
SAVEREL  BAL,R11     RELEASER                  RELEASE UNUSED MEMORY
        BAL,R14     XWLOCGLB                  SURFACE GLOBALS TO SYMBOL TABLE
        LI,R14      SAVEDONE
        LW,R12      KEY1                      SET 1ST KEY FOR KEYED WRITES
        LI,R2       0                        SET TO INDICATE WRITE FOR )SAVE
WRITEWS  LI,R7       0
        LI,R6       SYMTOFF-1
        STW,R7      *LODYN,R6
        BDR,R6      %-1
        LD,R6       LODYN
        LD,R8       APLVERSN
        LW,R3       SYMTSIZE
        LW,R4       FREETBL
        LODYN AND HICOMMON
        SET WORKSPACE DESRIPTONS IN LODYN

```

SAVTOPST	LW, R5	TOPOSTAK	
	LCI	7	SET SYMTSIZE, FREETBL, TOPOSTAK,
	STM, R3	1, R6	LODYN, HICOMMON, APLVERSN(DW)
	AI, R6	8	
	BAL, R7	CALTIME	SET TIME AND DATE - 4 WORDS
	LW, R7	CURRCS	GET CURRENT CODESTRING POINTER
	BNEZ	SAVECS	SET AUTOSTART FLAG
	LW, R7	CMNDTYPE	SET AUTOCONTINUE IF INDICATED
SAVECS	STW, R7	4, R6	SAVE FLAG
	LI, R7	0	-----
	LI, R6	1-STACKOFF	- ZERO FIXED COMMON REGION, -
	STW, R7	*HICOMMON, R6	- EXCEPT -STACKOFF (LAST WD). -
	BIR, R6	%-1	-----
	LW, R6	HICOMMON	
	AI, R6	-1	
	LI, R7	NCOMPRMS	NO. OF COMMON PARAMETERS
	STW, R7	0, R6	SAVE AT HICOMMON-1
SCOMPRMS	LW, R8	COMPRMST-1, R7	SAVE COMMON PARAMETERS
	AI, R6	-1	
	STW, R8	0, R6	
	BDR, R7	SCOMPRMS	
	LW, R8	SICTRL	SAVE SI CTRL IN COMMON REGION.
	STW, R8	-1, R6	NOTE -- SICTRL SAVED AT HICOMMON - 17.
	LW, R8	GRAFPNTR	
	STW, R8	-2, R6	NOTE -- GRAFPNTR SAVED AT HICOMMON - 18.
	LW, R8	GDIGITS	
	STW, R8	-3, R6	NOTE -- GDIGITS SAVED AT HICOMMON -19.
*			
	LW, R11	LODYN	GET ADDRESS FOR 1ST PAGE OUT
	LI, R10	2048	SET RECORD SIZE (USED BY COPY)
	LW, R13	TOPOSTAK	SET TEST
	AI, R13	-512	FOR START OF COMMON
SAVLOOP1	CW, R11	FREETBL	CHECK IF PAGE IS IN DYNAMIC USE
	BGE	SAVLOOP4	NO
SAVLOOP2	BAL, R7	SAVWRITE, R2	YES-WRITE IT
	AI, R12	1	UPDATE KEY
SAVLOOP3	AI, R11	512	UPDATE ADDRESS-1 PAGE
	B	SAVLOOP1	CONTINUE
SAVLOOP4	CW, R11	R13	CHECK IF COMMON REACHED
	BLE	SAVLOOP3	NO
	CW, R11	HICOMMON	YES-CHECK IF DONE
	BL	SAVLOOP2	NO
CMDRETRN	B	*R14	RESUME VIA R14.
SAVEDONE	LCI	3	
	LM, R6	OPWSNAME	
	STM, R6	WSIDNAME	SET NEW WSID
	LD, R6	OPWSPAS	
	STD, R6	WSIDPASS	AND PASSWORD
	LW, R6	USERACCT	RESET USERS ACCOUNT
	LW, R7	USERACCT+1	(NOT DW BOUND)
	STD, R6	LODACCT	
	LI, R4	0	RESET CURRCS AND TEST VALUE
	STW, R4	QUIETFLG	(CLEAR THE QUIET-FLAG, BE NOISY)
	XW, R4	CURRCS	
	BEZ	SAVXW	NO CODESTRING
	AI, R4	-2	AUTOSTART CODESTRING EXISTS
	BAL, R7	DREF	DEREFERENCE IT
SAVXW	BAL, R14	XWLOCGLB	RE-EXCHANGE LOCALS AND GLOBALS
	LI, R2	0	SET WS OFFSET TO ZERO
LODEXIT	LI, R5	F:WS	DCB ADDRESS
	BAL, R6	CLOSV	CLOSE THE SAVED FILE
	MTW, 0	BREAKFLG	CHECK FOR 'HANGUP'
	BLZ	UTSIOFFH	YES-QUIT
	B	SAVEDMES	ISSUE 'SAVED' MESSAGE AND EXIT
NOFID	BAL, R14	RESACCT	RESET ACCOUNT
	BAL, R14	FNEQWSID	SET FID TO WSID
	B	SAVECR	
		PAGE	
*			
*	@LOAD-DRIVER TO	LOAD A WORKSPACE (OR ERROR EXIT)	
*			
@LOAD	SW, R6	CMNDTYPE	SET QUIET-FLAG: -1 -- IF QUIET &
	STW, R6	QUIETFLG	0 -- IF NOISY.
	BAL, R13	ACQFID	GET FILE IDENTIFICATION.
LOADCONT	BAL, R14	TRYLOAD	TRY TO OPEN AND READ ID RECORD.
	LI, R2	0	SET TO INDICATE READ FOR LOAD
	STW, R2	ERRORID	(CLEAR INTERNAL ERROR I.D.)
	LI, R14	NEWSID	SET EXIT FROM READWS

READWS	LW, R3	KEY1	START WITH KEY1
	LI, R8	ID2BIGLD	SET FOR 'TOO BIG TO LOAD' ERROR.
	STW, R14	HOLDFLG	SAVE R14 FOR RETURN
	LW, R10	2, R4	
	SW, R10	4, R4	'FREETBL'-'LODYN' FOR FILED WS
	AW, R10	5, R4	
	SW, R10	3, R4	+'HICOMMON'-'TOPOSTAK'=TOTAL SPACE
	CW, R10	CORAVAIL	CHECK FOR FIT IN CORE
	BGE	CLOSV	TOO BIG
	LW, R10	2, R4	
	AI, R10	511	
	AND, R10	FFFFFFE00	ROUND 'FREETBL' UP TO PAGE BOUND
	LW, R9	3, R4	
	AND, R9	FFFFFFE00	ROUND 'TOPOSTAK' DOWN
	LI, R8	0	INITIALIZE LOADCASE FLAG VALUE
	CW, R9	R10	CHECK IF FILED WS SHARES PAGE
	BGE	NODUAL	OF DYNAMIC AND COMMON
	AI, R8	1	YES-SET FLAG
NODUAL	STW, R8	LOADCASE	SAVE FLAG
	BAL, R11	CLEARMEM	CLEAR MEMORY
	AW, R9	HICOMMON	IN-CORE HICOMMON
	SW, R9	5, R4	'HICOMMON' FOR FILE
	LI, R14	FTCHCOMN	SET RETURN
FTCHCOMN	CW, R9	LOCOMMON	CHECK IF COMMON PAGE(S) NEEDED
	BL	GETCOM	YES
	LW, R8	2, R4	'FREETBL'
	AW, R8	WSOFFSET	ADJUST FOR WS OFFSET
	STW, R8	NEWBOUND	
	BAL, R14	GETDYN	GET DYNAMIC PAGES FOR LOAD
	CW, R8	DYNBOUND	CHECK AGAIN FOR OVERLAP
	BLE	SLODSIZ	NO
	STW, R8	DYNBOUND	YES-SET OVERLAP CONDITION
	STW, R8	LOCOMMON	
	MTW, 2	LOADCASE	SET FLAG FOR IN-CORE OVERLAP COND.
SLODSIZ	LI, R10	2048	SET RECORD SIZE (512 WORDS)
	LW, R11	LODYN	SET TO START LOAD
	LW, R12	R3	SET 1ST KEY OR GRANULE #
LODLOOP1	CW, R11	NEWBOUND	ADJUSTED 'FREETBL'
	BGE	LODLOOP2	DYNAMIC PAGES ARE LOADED
	BAL, R14	LOADREAD, R2	READ A DYNAMIC RECORD
	AI, R12	1	KICK KEY
	AI, R11	512	UPDATE BUFFER ONE PAGE
	B	LODLOOP1	LOOP
LODLOOP2	LW, R7	LOADCASE	CHECK WHICH LOAD CASE
	B	LODLOOP3, R7	
LODLOOP3	B	LODOPEN	NO OVERLAPS
	B	LODXPAND	FILE OVERLAP, EXPAND IN CORE
	B	LODPACK	CORE OVERLAP, COMPRESS IN CORE
	B	LODCOMON	BOTH OVERLAP, LOAD COMMON PAGES
LODXPAND	AI, R11	-512	BACK UP 1 PAGE
	LW, R5	LOCOMMON	GET START OF COMMON
	LI, R14	256	LOOP COUNTER
LODLOOP4	LD, R6	*R11	MOVE LAST DYNAMIC PAGE
	STD, R6	*R5	TO FIRST COMMON PAGE
	AI, R11	2	
	AI, R5	2	
	BDR, R14	LODLOOP4	
	LW, R11	R5	SET R11 FOR NEXT COMMON PAGE
	B	LODCOMON	AND LOAD REMAINING COMMON
LODPACK	STW, R11	LOADCASE	SAVE RECORD READ POINTER
	LI, R11	WINDOW	
	BAL, R14	LOADREAD, R2	READ FIRST COMMON PAGE INTO
	AI, R12	1	'WINDOW' AND UPDATE KEY
	LI, R14	X'1FF'	
	AND, R14	3, R4	GET PAGE OFFSET OF 'TOPOSTAK'
	LW, R5	LOADCASE	
	AI, R5	-512	GET START OF LAST DYNAMIC PAGE
	AW, R5	R14	ADD TOPOSTAK OFFSET
	AW, R11	R14	OFFSET 'WINDOW'
LODLOOP5	LW, R6	*R11	MOVE THE USED PART OF COMMON
	STW, R6	*R5	FROM WINDOW TO TARGET ADDRESSES
	AI, R5	1	
	AI, R11	1	
	CI, R11	WINDOW+511	
	BLE	LODLOOP5	
	LW, R11	LOADCASE	SET R11 FOR NEXT COMMON PAGE
	B	LODCOMON	
LODOPEN	CW, R11	R9	CHECK START OF COMMON

	BGE	LODCOMON	YUP
	AI, R11	512	NO
	B	LODOPEN	LOOP
LODCOMON	CW, R11	HICOMMON	CHECK IF LOAD DONE
	BGE	LODFINI	YUP
	BAL, R14	LOADREAD, R2	NOPE-READ NEXT RECORD
	AI, R12	1	KICK KEY
	AI, R11	512	AND BUFFER ADDRESS
	B	LODCOMON	
LODFINI	LI, R5	NCOMPRMS	LOAD DONE, SET COMMON PARAM. BOUND
	LW, R6	HICOMMON	AND INDEX IN HICOMMON
LCOMPRMS	LW, R8	-2, R6	
	STW, R8	COMPRMST-1, R5	MOVE COMMON PARAMETERS TO TABLE
	AI, R6	-1	IN DATA REGION
	BDR, R5	LCOMPRMS	
	LW, R5	HICOMMON	
	SW, R5	5, R4	'HICOMMON' ON FILE
	AWM, R5	STATEPTR	ADJUST STACK
	AWM, R5	TOPOSTAK	POINTERS (IF NECESSARY)
	LW, R5	4, R4	'LODYN' FOR LOADED WS
	AW, R5	0, R4	SYMT OFFSET FOR LOADED WS
	AW, R5	WSOFFSET	
	STW, R5	SYMT	SET SYMT ADDRESS
	LW, R5	1, R4	SET UP
	STW, R5	SYMTSIZE	SYMBOL TABLE
	SLS, R5	1	SIZE
	STW, R5	NSYMTWDS	
	LW, R3	NEWBOUND	ADJUSTED 'FREETBL'
	BAL, R5	SETFREET	SET UP FREETBL
	B	*HOLDFLG	(RETURN VIA SAVED R14)
NEWSID	LCI	3	
	LM, R6	OPWSNAME	
	STM, R6	WSIDNAME	SET WSID
	LD, R6	OPWSACT	
	STD, R6	LODACCT	SET LOAD ACCOUNT
	LD, R6	OPWSPAS	SET WSID
	STD, R6	WSIDPASS	PASSWORD IF PRESENT
	LI, R14	-1	
	STW, R14	CATCHTBL	CLEAR THE CATCH TABLE.
	STW, R14	CATCHTBL+2	
	STW, R9	ERRLOC NOTE: R9=0	CLEAR ERROR LOCATION &
	STW, R9	ERRNUM ----	ERROR NUMBER.
	STW, R9	OBSFLAG	OBSERVE FLAG TOO.
	STW, R9	OBSERVE	
	LW, R14	IDBUF+7	GET APLVERSN+1 FOR THE SAVED WS.
	CW, R14	SA02	CK AGAINST THE 'A02' VERSION...
	BGE	NEWSIDRX	A02 OR LATER.
*			
*			
*			ANCIENT WS ...
	LW, R6	LODYN	-----
	AI, R6	13-1	-
	LI, R7	SYMTOFF-13	- ZERO THOSE WDS OF FIXED DYNAMIC -
	STW, R9	*R6, R7	- & FIXED COMMON THAT WERE UNUSED -
	BDR, R7	%-1	- IN WORKSPACES PRIOR TO A02. -
	LW, R6	HICOMMON	-
	AI, R6	1-17	- NOTE--THIS CODE IS OK ONLY AS -
	LI, R7	17-STACKOFF	- LONG AS SYMTOFF = 20 & -
	STW, R9	*R6, R7	- STACKOFF = 32. -
	BIR, R7	%-1	-----
NEWSIDRX	BAL, R14	XWLOCGLB	RE-EXCHANGE LOCALS & GLOBALS.
	LW, R6	HICOMMON	
	LW, R2	-17, R6 NOTE --	GET SI CTRL FROM SAVED WS,
	STW, R2	SICTRL	HICOMMON - 17.
	LW, R2	-18, R6 NOTE --	GET GRAPHICS SPECIAL BLOCK PTR FROM
	STW, R2	GRAFPNTR	SAVED WS, HICOMMON - 18:
	LW, R2	-19, R6 NOTE --	GET GDIGITS FROM SAVED WS,
	BLEZ	%+2	DON'T USE IF OLD WS.
	STW, R2	GDIGITS	
	LW, R2	IDBUF+12	CHECK FOR AUTOSTART
	BLEZ	SOFFSET	NO
	STW, R2	CURRCS	YES-SET CURRCS
SOFFSET	LW, R2	WSOFFSET	WS OFFSET
	B	LODEXIT	
*	TEXT	'APLW'	THIS IS THE APLVERSN AT WHICH UNUSED FIXED
SA02	TEXT	'SA02'	DYNAMIC & COMMON WAS ZEROED FOR A SAVED WS.
*	PAGE		


```

* ACQFID-ACQUIRE FILE ID FOR COMMAND PROCESSING
*
* SETS  FILENAME,ACCT,PASSWORD, AS INDICATED IN FPT FOR I/O  OPERATION
* SETS  ACCT AND PASSWORD CONTROL WORDS(OR RESETS) AS INDICATED
*
* CHECKS FOR END OF COMMAND AS FORMATS INDICATE
*      (NO ADDITIONAL COMMAND SCAN RELEVANT ON EXIT)
*
* R13=LINK  USES:ACQNXNB FOR CHARACTER ACQUISITION
*           :ACQIT FOR NAME-NUMBER ACQUISITION
* EXITS TO ERBADCMD ON BAD FORMAT
*
* ON EXIT,R1,R2, AND R3 ARE RELEVANT TO THE 1ST NON-BLANK AFTER FID
*
ACQFID  BAL,R14  NAMENUM          CHECK IF NEXT ITEM IS NAME OR NUMBER
        B        SEMICK          NEITHER-NO FID
        B        AFP            NUMBER- ACCT NAME:PASS FORMAT
        BAL,R14  SETFNAME        NAME-SET FILENAME
        CI,R2    X'4B'          .
        BE      ONEDOT          NAME. FORMAT
        CI,R2    X'7A'          :
        BE      PASSNOAC        NAME:PASS FORMAT
        BAL,R14  RESPASS        YES NAME ONLY
        BAL,R14  RESACCT        RESET PASSWORD AND ACCOUNT
        B
ONEDOT  BAL,R4   ACQNXNB
        CI,R2    X'4B'          .
        BE      PASSNOAC        NAME..PASS FORMAT
        BAL,R14  ACCTCHK        GET THE ACCOUNT          17-00004
NUMACCT BAL,R14  SETACCT        SET ACCOUNT
        CI,R2    X'4B'          . CHECK FOR PASSWORD
        BE      PASSWORD        YES
        BAL,R14  RESPASS        RESET PASSWORD
CRCHECK CI,R2    X'15'          CHECK END
        BE      *R13            YES
        CI,R13   COPYFID+1      CHECK IF ACQFID FOR COPY OR PCOPY
        BE      *R13            YES-MORE TEXT MAY FOLLOW
        CI,R13   SAVETACT        CHECK IF ACQFID FOR SAVE
        BE      SAVETACT        YES-MAY BE FOR AUTOLOAD
        B        ERBADCMD        NO-GARBAGE ERROR
PASSNOAC BAL,R14  RESACCT
PASSWORD BAL,R4   ACQNXNB
        BAL,R14  ACCTCHK        GET THE PASSWORD-USE ACCTCHK  17-00006
        BAL,R14  SETPASS        SET PASSWORD
        B        CRCHECK
AFP      BAL,R14  SETACCT        SET ACCOUNT NUMBER
        BAL,R14  NAMENUM        GET FILE NAME
        B        ERBADCMD        GARBAGE
        B        ERBADCMD        NUMBER-NOT VALID
        BAL,R14  SETFNAME        SET FILE NAME
        CI,R2    X'7A'          :
        BNE     CRCHECK
        B        PASSWORD
SEMICK   CI,R2    X'5E'          ;
        BNE     ERBADCMD
        CI,R13   SAVETACT        CHECK IF SAVE COMMAND
        BNE     ERBADCMD        NO-ERROR
        BAL,R14  RESACCT        YES-RESET ACCOUNT
        BAL,R14  FNEQWSID       AND USE WSID FOR SAVE
        B        SAVETACT
*
* NAMENUM-USED BY ACQFID-CHECKS IF NEXT ACQIT ITEM
*      IS :GARBAGE          EXIT TO CALL+1
*      :NUMBER(INTEGER)     CALL+2
*      :NAME                 CALL+3
*
*      USES ACQIT
* R14=LINK
* R12 AND REGISTERS RELEVANT TO ACQIT USED
*
NAMENUM CLM,R3   NONAME          CHECK NAMECHAR-DIGIT
        BCR,9    *R14            NO
        AI,R14   2              YES
        CLM,R2   F0F9           CHECK DIGIT
        BCS,9    ACQIT          NO-GET AND USE NAME
        AI,R14   -1            YES-MESS!
        LW,R12   R14            SAVE RETURN
        BAL,R14  ACQIT          GET MESS

```

GETNCHR	AI, R5	-1	CHECK IF ALL DIGITS	
	LB, R8	NAMEBUF, R5		
	CLM, R8	F0F9		
	BCS, 9	ERBADCMD	NON-DIGIT-ERROR	
	BDR, R5	GETNCHR		
	B	*R12	OK	
*				17-00008
*	ACCTCHK-GETS	ACCOUNT-TERMINATES ON FOLLOWING CHARACTERS:		17-00009
*		COMMA, PERIOD, BLANK, CARRIAGE RETURN		17-00010
*	USES ONLY FIRST 8 CHARACTERS-PRESETS 8 CHARACTERS TO BLANKS			17-00011
*	R14=LINK. R1 TO R5 AND R8 USED.			17-00012
*	ROUTINE ACQNXCC USED			17-00013
*				17-00014
ACCTCHK	AI, R1	-1	BACK UP MOMENTARILY	17-00015
	LW, R5	BLANKS		17-00016
	STD, R5	NAMEBUF	PRESET 8 BLANKS	17-00017
	LI, R5	-1		17-00018
	LI, R8	9	INITIALIZE LOOP	17-00019
ACCTCHK1	AI, R5	1		17-00020
	BAL, R4	ACQNXCC	GET NEXT CHARACTER	17-00021
	CI, R2	X'40'	BLANK	17-00022
	BE	ACCTCHK2	YES-SKIP BLANKS AND EXIT	
	CI, R2	X'6B'	COMMA	17-00024
	BE	*R14	YES-EXIT	17-00025
	CI, R2	X'4B'	PERIOD	17-00026
	BE	*R14	YES-EXIT	17-00027
	CI, R2	X'15'	CARRIAGE RETURN	17-00028
	BE	*R14	YES-EXIT	17-00029
	CI, R2	X'5E'	SEMI-COLON	
	BE	*R14	YES-EXIT	
	STB, R2	NAMEBUF, R5	SET BYTE	17-00030
	BDR, R8	ACCTCHK1	LOOP-SET UP TO 8 BYTES	17-00031
	B	ACQNXCC	OVERRUN LOOP	17-00032
ACCTCHK2	LW, R4	R14	SET RETURN	
	B	ACQNXNB	GET NON-BLANK	
	PAGE			
*				
*	@WSID-DISPLAY	OLD WSID, SET NEW NAME(RESET ACCT-PASS)IF INDICATED		
*				
@WSID	STW, R2	SAVE312	SAVE CR-NOCR FLAG	
	CI, R2	X'15'	CHECK IF NEW NAME GIVEN	
	BNE	WSIDNEW	YES-OR ERROR.	
	LW, R7	ISMSG	NO, GET 'IS '.	
	B	WSIDMSG		
WSIDNEW	BAL, R13	ACQFID	ACQ FID.	
	BAL, R14	TESTACCT	ERROR IF ACCT SPECIFIED.	
	LD, R8	OPWSACT		
	STD, R8	LODACCT		
	LW, R7	WASMSG	GET 'WAS '.	
WSIDMSG	LD, R8	LODACCT	CHECK LOAD ACCT.	
	CW, R8	USERACCT	VS USERS ACCT	
	BNE	GOWSID	DIFFER	
	CW, R9	USERACCT+1		
	BNE	GOWSID	DIFFER	
	LW, R8	BLANKS	SAME-BLANK FIELD	
	LW, R9	BLANKS		
GOWSID	LCI	3		
	LM, R10	WSIDNAME	GET OLD WSID NAME (TEXTC)	
	LCI	6		
	STM, R7	IMAGE		
	LI, R8	X'40'		
	STB, R8	IMAGE+3	BLANK THE C-COUNT	
	LI, R3	24		
	BAL, R12	DUMPLING	ISSUE MESSAGE	
	LW, R2	SAVE312	CHECK IF NEW TO BE SET	
	CI, R2	X'15'		
	BE	CMDEXIT	NO-EXIT	
	LCI	3	YES	
	LM, R8	OPWSNAME		
	STM, R8	WSIDNAME	SET NEW NAME	
	LD, R8	OPWSPAS	SET (OR RESET)	U17-0005
	STD, R8	WSIDPASS	NEW PASSWORD	U17-0006
	LW, R8	USERACCT		
	LW, R9	USERACCT+1		
	STD, R8	LODACCT	RESET LODACCT TO USERS ACCT	
	B	CMDEXIT	EXIT	
	PAGE			
*				

```

* @DROP-RELEASE AN INDICATED WS OR FILE FROM THE USERS ACCOUNT
*
@DROP      BAL,R13  ACQFID      GET FID
           BAL,R14  TESTACCT    QUIT IF ACCT(NOT USERS)SPECIFIED
           BAL,R14  DROPFILE    DROP FILE
           B        CMDEXIT
           PAGE

*
* @COPY AND @PCOPY-DRIVER FOR COPY AND PROTECTED COPY FROM WORKSPACE
*
@COPY      LI,R13   0
           STW,R13  PCOPYFLG    SET PCOPYFLG = 0 (I.E. NOT).
           B        SETQUIET
@PCOPY     STW,R6   PCOPYFLG    SET PCOPYFLG = SMALL NEG.INTEGER.
SETQUIET   SW,R6    CMNDTYPE    SET QUIET-FLAG: -1 -- IF QUIET &
           STW,R6   QUIETFLG    0 -- IF NOISY.
COPYFID    BAL,R13  ACQFID      GET FILE ID FOR WS TO COPY
           BAL,R4   ACQNB       ACQUIRE NON-BLANK CHARACTER      17-00034
           STW,R1   IMAGEPOS    SAVE POINTER TO CHARACTER        17-00035
           AI,R4    2           SET RETURN FROM ACQNXNB(BELOW)  17-00036
           CLM,R3   NONAME      VERIFY NAME-LIST HAS NAME-CHARS ONLY
           BCS,9    ACQNXNB     OK, LOOP PAST ANY BLANKS.
           AI,R2    -NEWLINE    NO, VERIFY END-OF-STMT...
           BNEZ     ERBADCMD    OOPS -- INCORR. CMND.
           LW,R14   MODE        ARE WE IN FUN DEFN MODE...
           BNEZ     A5         NO.
           BAL,R14  B4CLOSE     YES, DO FORCED CLOSE.
A5          BAL,R14  TRYLOAD    FIND COPY WS, PUT HDR IN IDBUF:
*
* IDBUF -- WORDS OF INTEREST WHEN COPYING:
*
* IDBUF + 0      SYMTOFF = OFFSET OF COPY SYM TBL IN COPY DYNAMIC.
* IDBUF + 1      SYMTSIZE = SIZE (# OF ENTRIES) OF COPY SYM TBL.
* IDBUF + 2      FREETBL = LOC OF 1ST WD BEYOND COPY DYNAMIC.
* IDBUF + 4      LODYN   = LOC OF 1ST WD OF COPY DYNAMIC.
* IDBUF + 8 THRU + 11 = TIME & DATE WHEN COPY WS WAS SAVED.
*
           LW,R13   IDBUF+2
           SW,R13   IDBUF+4      CALC. SIZE OF COPY DYNAMIC.
           STW,R13  COPYSIZE
           LW,R13   HICOMMON
           SW,R13   LODYN       CALC. SIZE OF USER'S ALLOCATION AND
           AI,R13   -STACKOFF-4 VERIFY COPY DYNAMIC CAN AT LEAST
           SW,R13   COPYSIZE    BE LOADED...
           BGEZ     COPY        OK.
           LI,R8    ID2BIGLD    BARF -- TOO BIG TO LAAD.
           LI,R6    CMDERR      CLOSE COPY WS FILE, REPORT ERROR,
           B        CLOSV       AND EXIT FROM COMMAND.
           PAGE

*
* @LIB-DRIVER TO LIST WORKSPACES IN INDICATED ACCOUNT
*
*
* WORK IS DONE IN 'APLUTSI'-APL UTS INTERFACE MODULE
*
@LIB      RES      0
           B        LIBUTS
           PAGE

*
* @SYMBOLS-SET SYMBOL TABLE SIZE-(IF THIS IS CLEAR WS)
*
@SYMBOLS  BAL,R14  BAQCONLY    GET CONSTANT, STORE IN CONSTBUF.
           BL       SYMBOLD     NO NEW VALUE
           BG       ERBADCMD    MORE THAN 1 VALUE
           CI,R7    2001        RANGE TEST
           BG       ERBADCMD    TOO HIGH
           LI,R8    21          OK-COMPUTE 20I+41
ADD20     AI,R8    20
           CW,R8    CONSTBUF
           BL       ADD20
           BAL,R14  QCLEAR      CHECK IF CLEAR WS
           BNE      ERBADCMD    NO
           SLS,R8   1
           STW,R8   NSYMTWDS    NO. OF WORDS IN SYMTABLE
           SLS,R8   -1
           XW,R8    SYMTSIZE    SYMBOL TABLE SIZE,OLD VALUE NOW IN 7
           STW,R8   CONSTBUF    SAVE OLD VALUE
           LW,R8    SYMT

```

```

        AW,R8      NSYMTWDS      END OF SYMBOL TABLE
        STW,R8     NEWBOUND
        BAL,R14    GETDYN        GET MEMORY IF REQUIRED
        LW,R1      SYMTSIZE      GET SYMT SIZE
        B          SETSYMT      USE 'CLEAR' TO FINISH
OLDSYMS  LW,R7     CONSTBUF      GET OLD VALUE
        B          OUTORANG+1    ISSUE MESSAGE AND EXIT
SYMUSAGE TEXTC     ' UNUSED OF '
SYMBOLD  BAL,R14  UNSYMS        GET NO.OF UNUSED SYMBOLS.
        LW,R8     R11           PUT THAT NO. IN R8.
        LI,R3     0            INIT. POSITION IN IMAGE.
        BAL,R13   CINTIM       CV NO. TO EBCDIC & PUT IN IMAGE.
        LI,R8     SYMUSAGE     PT AT TEXTC MSG.
        BAL,R13   TEXTC2I      MOVE IT TO IMAGE.
        LW,R8     SYMTSIZE     GET SIZE OF SYM TBL.
        B          VALOUT      APPEND THAT VALUE, DISPLAY, EXIT.
*
* UNSYMS -- GETS NO.OF UNUSED SYMBOLS IN R11.
*
* R14 = LINK.  R3,R8,R13 ARE VOLATILE.
*
UNSYMS  LI,R11    0            CLEAR UNUSED SYMBOL COUNT.
        LW,R13    SYMTSIZE     NO.OF ENTRIES FOR SYM TBL.
        LW,R3     SYMT
UNSYMSQ LW,R8     1,R3        LOOK AT NAME-INDICATOR WORD...
        BNEZ      UNSYMSN     NAME IN USE.
        AI,R11    1           UNUSED.
UNSYMSN AI,R3     2           PT AT NEXT SYM TBL ENTRY.
        BDR,R13  UNSYMSQ      LOOP TILL DONE.
        B          *R14       EXIT
*
* QCLEAR-CHECK IF CLEAR WS
*
* R14=LINK, R13 USED
*
* EXIT WITH CC'S SET FOR = IF CLEAR, NOT= OTHERWISE
*
QCLEAR  LW,R13    DYNBOUND     HIGHEST AVAIL DYNA
        SW,R13    SYMT
        SW,R13    NSYMTWDS     TOTAL AVAIL DYNAMIC
        CW,R13    FREETBL+1
        B          *R14
        PAGE
*
* @OPR-GENERATE OPERATOR MESSAGE-WAIT FOR REPLY
* @OPRN-NO WAIT FOR REPLY
*
*
@OPRN   LI,R6     0
@OPR    STW,R6    PCOPYFLG     SAVE WAIT/NO-WAIT FLAG
        LW,R6     HICOL
        AI,R6     -BA(IMAGE)   MESSAGE LENGTH-1
        STB,R6    IMAGE        CONVERT TO TEXTC WITH LENGTH L-1
        BAL,R14   GENOPRM      GENERATE OPERATOR MESSAGE
        LI,R8     SENTMSG
        BAL,R7    WRTEXTC      GENERATE 'SENT' MESSAGE
        LW,R7     PCOPYFLG     CHECK IF 'WAIT'
        BEZ       CMDEXIT      NO
OPRWAIT BAL,R14   DELAY6       YES-
        MTW,0     BREAKFLG     DELAY UNTIL BREAK
        BEZ       OPRWAIT
        B          CMDEXIT
SENTMSG  TEXTC    'SENT'
        PAGE
*
* @TERM-TERMINAL COMMAND-READ A NEW SET OF TRANSLATION TABLES
*
@TERM   BAL,R14   BAQCONLY     GET N
        BG        ERBADCMD     TOO MANY
        BE        NEWTERM      1
        LW,R7     TERMTYPE     NONE
        B          GENIS       GENERATE THE 'IS ' VERSION.
NEWTERM CW,R7     TERMTYPE     CHECK NEW VS OLD
        BE        OUTORANG+1   SAME-MESSAGE ONLY
        B          GETTERM     DIFFER
        PAGE
*
* @TABS-CHANGE TAB SETTING(S) AND(OR) REPORT OLD VALUE(S)

```

```

*
@TABS    BAL,R14  BAQONLY      GET CONSTANT(S)
          BL      TABSOLD      NONE.
          LI,R4   CONSTBUF     ADDRESS OF TABS
          BAL,R5  QTABVLID     CHECK VALIDITY OF NEW TABS
          B       ERBADCMD     BAD
          LW,R3   WASMSG       OK, GENERATE THE 'WAS ' MSG.
TABSMSG  STW,R3   IMAGE
          LI,R3   4
          LI,R6   1            SET INDEX
          LI,R10  1            AND COUNT
          LW,R8   TABPNTR      CHECK TYPE OF TABS
          CI,R8   128
          BL      TABEVEN      EVEN-ONE VALUE
          LB,R10  TABVALS      ODD- N VALUES
TABODD   LB,R8    TABVALS,R6
          AI,R6   1
TABEVEN  BAL,R13  CINTIM       GENERATE VALUE
          BAL,R11  STASHBL      BLANK
          BDR,R10  TABODD       LOOP
          BAL,R12  DUMPLING     DUMP LINE
          LI,R4   CONSTBUF
          BAL,R5   SNEWTABS     SET NEW TABS
          B       CMDEXIT      EXIT
TABSOLD  LW,R3    ISMSG        GENERATE THE 'IS ' MSG.
          B       TABSMMSG
          PAGE

```

```

*
* CONTINUE;CONTINUE HOLD;AUTOMATIC CONTINUE
*

```

```

@CONTINU RES      0
          AI,R2    -NEWLINE     CHECK CR
          BEZ      CONTOFF      YES
          BAL,R14  ACQIT        NO
          LW,R8    HOLD         CHECK 'HOLD'
          CW,R8    NAMEBUF
          BNE      ERBADCMD     NO-ERROR
CONTOFF  STW,R2    HOLDFLG      SAVE HOLD-OFF FLAG
          LI,R8    @@CONT-CMDB-NCMDS 'CONTINUE' COMMAND TYPE
          STW,R8    CMNDTYPE     (SET FOR 'HANG-UP' CASE)
          MTW,0    ON%OFF       CHECK IF OFF-LINE
          BGZ      CONT1        NO
          MTW,-1   QUITFLAG      YES-AVOID REPEATED ATTEMPTS
          BLZ      UTSIOFFH     TO SAVE 'CONTINUE'
CONT1    LCI      3
          LM,R6    KONTINUE      SET WSID TO 'CONTINUE'
          STM,R6   WSIDNAME
          B        NOFID        ENTER 'SAVE' LOGIC

```

PAGE

```

SPACE 2
Z      SET      %-CMD@          SIZE OF CMD MODULE IN HEX.
SPACE
Z      SET      Z+Z/10*6+Z/100*96+Z/1000*1536  SIZE IN DECIMAL.
SPACE 3
END

```


TITLE 'APLUTSC-B00,09/20/73,DWG702985'
 SYSTEM SIG7F
 SYSTEM BPM

*
 *

REF'S:

REF	@@CONT		
REF	@@LOAD		
REF	ACCTCHK	ACCOUNT CHECK ROUTINE	18-00001
REF	BCX	BRANCH THROUGH ROOT TO CX	
REF	BREAKFLG		
REF	CHKTERM	CHECK IF 4013,IF SO SWITCH TO TTY	
REF	CLOSR	CLOSE AND RELEASE	
REF	CLOSV	CLOSE AND SAVE	
REF	CMDB	COMMAND BRANCH VECTOR	
REF	CMDEXIT	COMMAND EXIT	
REF	CMNDTYPE	COMMZND TYPE	
REF	COPYBASE	BASE FOR COPY DB	
REF	COPYHOME	HOME FOR COPY DB	
REF	COPYSIZE	SIZE OF COPY BLOCK	
REF	CURRCS	CURRENT CODESTRING POINTER	
REF	DONTSAVE	ERROR -- NOT SAVED, THIS WS IS ...	
REF	DUMPLINE	OUTPUT ROUTINE	
REF	ERBADCMD	BAD COMMAND EXIT	
REF	ERBADWS	BAD WS EXIT	
REF	ERLIBREF	BAD LIB REF	
REF	ERRFTF	ERROR ROUTINE	
REF	ERRFWS	ERROR ROUTINE	
REF	F:TF	DCB NAME	
REF	F:WS	DCB NAME	
REF	FPARAMS	FILE PARAMETER BUFFER ADDRESS	18-00010
REF	FPTOPNXT		
REF	FPTOPTF	OPEN TEMP-FILE FPT	
REF	FPTOPWS	OPEN FPT	
REF	FPTOP1ST		
REF	FPTWR2	FPT TO WRITE CONTROL TO 4013	
REF	FPTXCOFF	RESET EXIT CONTROL	
REF	FQTABL	TABLE OF DCB ADDRESSES	
REF	FREETBL	FREE SPACE TABLE	
REF	GRAFBUF	GRAPHICS BUFFER	
REF	HICOMMON		
REF	HOLDFLG	OFF VS HOLD FLAG	
REF	IDBUF	ID BUFFER	
REF	INBUF	INPUT BUFFER	
REF	IMAGE	IMAGE BUFFER	
REF	INTRANS	INPUT TRANSLATE TABLE	
REF	J:ACCN	ACCOUNT #	
REF	KEY1	KEY VALUE 1	
REF	LODYN	LOW DYNAMIC ADDRESS	
REF	NAMEBUF	NAME BUFFER	
REF	NAMEGRN1	GRANULE #,1ST NAME RECORD	
REF	NAMERKEY	NAME RECORD COUNT	
REF	NBIO	NO. OF BLIND IO CHANNELS	
REF	NCMDS	# OF SYSTEM COMMANDS	
REF	NUMFILES	NO. OF FIO CHANNELS	18-00004
REF	OPWSACTC	ACCOUNT CONTROL	
REF	OPWSACT	ACCOUNT	
REF	OPWSMODE	I/O MODE	
REF	OPWSNAME	NAME	
REF	OPWSPASC	PASSWORD CONTROL	
REF	OPWSPAS	PASSWORD	
REF	OPWSWRTC	WRITE CONTROL PARAMETER	18-00012
REF	OP1STACC	OPEN 1ST ACCOUNT CONTROL	
REF	OP1STACT	OPEN 1ST ACCT	
REF	OUTORANG		
REF	PRINTFNM	PRINT FILE NAME	18-00014
REF	QUIETFLG	=0 IF SAVED MSG TO BE DISPLAYED OR =-1 IF NOT (BUT WS MUST BE OK).	
		ROUTINE TO READ WS FROM FILE	
REF	READWS	RELEASE COMMON	
REF	RELCOM		
REF	RELDYN		
REF	RELEASER	RELEASES UNUSED CORE	
REF	RETURN14	TEMP	
REF	SAVESIX	TEMP SAVE	
REF	SAVE14	TEMP	
REF	TELEXIT	EXIT TO TEL	
REF	TERMKEY	TERMINAL RECORD KEY	
REF	TERMSIZ	TERMINAL RECORD SIZE	
REF	TERMTYPE	TERMINAL TYPE	

*

REF	TOPOSTAK	
REF	WINDOW	TEMP AREA
REF	WRITEWS	ROUTINE TO WRITE WS ON FILE
REF	WSCKDSPL	WORKSPACE CHECKER
REF	WSIDNAME	
REF	WSIDPASS	WSID PASSWORD
REF	WSOFFSET	WS OFFSET
REF	XFF	X'FF'

* DEF'S:

DEF	APLUTSC@	MODULE NAME
DEF	CALOPNXT	CAL'S
DEF	CALOPTF	DEF'D
DEF	CALOPTRM	OPEN)TERM FILE
DEF	CALOPWS1	TO
DEF	CALOPWS2	ALLOW
DEF	CALOPWS3	REF
DEF	CALOPWS4	BY
DEF	CALOP1ST	ERROR
DEF	CALRDTF	ROUTINES
DEF	CALRDTRM	IN
DEF	CALRDWS	MODULE
DEF	CALRDWSI	APLUTSI
DEF	CALWRTF	
DEF	CALWRWS	
DEF	CLEARWS	MESSAGE
DEF	CLOSTHIS	CLOSE FILE
DEF	COPYDMES	ROUTINE TO WRITE SAVED MSG FOR COPY
DEF	DELAY6	DELAY 6 SECONDS
DEF	DROPFIL	ROUTINE TO DELETE FILE
DEF	ERRFWS6	ERROR PROCESSOR
DEF	FNEQWSID	SET FNAME TO WSID
DEF	FPTOPTRM	FPT'S
DEF	FPTRDTRM	
DEF	FPTRDWS	
DEF	GENOPRM	GENERATE OPR MESSAGE
DEF	GETTERM	GET TERMINAL TRANSLATE TABLES
DEF	KONTINUE	NAME,'CONTINUE'
DEF	LIBUTS	UTS INT. FOR)LIB COMMAND
DEF	LOADREAD	READ A RECORD FOR LOAD
DEF	OPNSAVE	OPEN F:WS FOR)SAVE
DEF	RDACTIV	READ ACTIVE WS FOR)COPY
DEF	RDCOPY	READ FROM F:WS FOR COPY
DEF	RDCOPYDR	READ COPY DATA RECORD
DEF	RDNAMER	READ NAME RECORD
DEF	RDWRLOOP	READ-WRITE LOOP
DEF	RESACCT	RESET ACCT
DEF	RESPASS	RESET PASSWORD
DEF	SAVEDMES	ISSUE SAVED MESSAGE AND EXIT
DEF	SAVWRITE	WRITE RECORD FOR SAVE
DEF	SETACCT	SET ACCOUNT
DEF	SETFNAME	SET FILE NAME
DEF	SETPASS	SET PASSWORD
DEF	TESTACCT	TEST ACCT # VS USERS
DEF	TESTOLDF	TEST(ON SAVE) IF OLD FILE EXISTS
DEF	TESTPASS	
DEF	TRYLOAD	
DEF	UTSIOFF	OFF
DEF	UTSIOFFH	OFF HOLD
DEF	WRACTIV	WRITE ACTIVE WS FOR)COPY
DEF	WRCOPYDR	WRITE COPY DATA RECORD
DEF	WRNAMER	WRITE NAME RECORD

* STANDARD EQU'S:

* REGISTERS

R0	EQU	0
R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6
R7	EQU	7
R8	EQU	8
R9	EQU	9
R10	EQU	10
R11	EQU	11
R12	EQU	12
R13	EQU	13
R14	EQU	14


```

R15      EQU      15
*
*  MODULE DESCRIPTION:
*
*  THIS IS THE SECOND OF TWO MONITOR INTERFACE MODULES
*  APLUTSI-ROOT UTS INTERFACE MODULE
*  APLUTSC-SPLIT FROM APLUTSI FOR OVERLAY PURPOSES
*  PART OF PATH 1 OVERLAY-HANDLES COMMAND RELATED
*  MONITOR INTERFACES
*
*  PAGE
APLUTSC@ CSECT      1
        BOUND      8
APLAPLAP TEXT      'APLAPLAP'          MAGIC WORDS FOR )LIB          18-00019
KONTINUE TEXTC     'CONTINUE'
        BOUND      8
SAYVE    TEXT      '  SAVED  '
        BOUND      8
CLEARWS  TEXTC     'CLEAR WS'
*
*  FPT'S WHICH CONTAIN NO VARIABLE PARAMETERS EXCEPT VIA INDIRECT
*  ADDRESSING
*
*  OPEN F:TF TO READ TERMINAL TRANSATION RECORD
*
FPTOPTRM GEN,8,7,17 X'14',0,F:TF      DCB ADDRESS
        DATA      X'C7400001'        P1,2,6,7,8 F12
        DATA      ERRFTF              ERR      (P1)
        DATA      ERRFTF              ABN      (P2)
        DATA      2                   KEYED ORG (P6)
        DATA      2                   RANDOM ACC (P7)
        DATA      1                   INPUT MODE (P8)
        DATA      2                   SAVE      (P10)
        DATA      X'01000303'
        TEXTC      'APLTRMSB'          NAME (CHANGED BETWEEN A01 AND B00)
        DATA      X'02000202'
        TEXT      ':SYS'               ACCOUNT= :SYS
        DATA      X'03010202'
        TEXT      'STRUDEL '          PASSWORD
*
*  READ F:TF FOR KEYED RECORD WITH TERMINAL TYPE TRANSLATION TABLE
*
FPTRDTRM GEN,8,7,17 X'10',0,F:TF      DCB ADDRESS
        DATA      X'3C000010'        P3,4,5,6 AND WAIT FLAG
        DATA      TERMBUF             BUFFER WORD ADDRESS
        DATA      TERMSIZ             BUFFER SIZE IN BYTES
        DATA      TERMKEY             ADDRESS OF KEY
        DATA      0                   BYTE DISP.
TERMBUF  EQU      INTRANS              TERMINAL DEPENDANT TABLE START AD.
*
*  FPTRDWS-READ A RECORD VIA F:WS FOR )LOAD OR )COPY
*
FPTRDWS  GEN,8,7,17 X'10',0,F:WS      DCB ADDRESS
        DATA      X'FC000010'        P1 TO P6 AND WAIT
        DATA      ERRFWS              ERR      (P1)
        DATA      ERRFWS              ABN      (P2)
        GEN,1,31  1,R11                BUF      (P3)
        GEN,1,31  1,R10                SIZ      (P4)
        DATA      FWSKEY              KEY      (P5)
        DATA      0                   BTD      (P6)
*
*  FPTRDTF-READ SEQUENTIAL RECORD VIA F:TF FOR )COPY
*
FPTRDTF  GEN,8,7,17 X'10',0,F:TF      DCB ADDRESS
        DATA      X'F4000010'        P1 TO P4,P6 AND WAIT
        DATA      ERRFTF              ERR      (P1)
        DATA      ERRFTF              ABN      (P2)
        GEN,1,31  1,R11                BUF      (P3)
        GEN,1,31  1,R10                SIZ      (P4)
        DATA      0                   BTD      (P6)
*
*  FPTWRWS-WRITE RECORD VIA F:WS, FOR )SAVE OR )CONTINUE
*
FPTWRWS  GEN,8,7,17 X'11',0,F:WS      DCB ADDRESS
        DATA      X'FC000030'        P1 TO P6, NEWKEY, AND WAIT
        DATA      ERRFWS              ERR      (P1)
        DATA      ERRFWS              ABN      (P2)

```

```

GEN,1,31 1,R11          BUF (P3)
DATA      2048          SIZ (P4)
DATA      FWSKEY        KEY (P5)
DATA      0             BTD (P6)
FWSKEY    EQU          RETURN14    TEMP FOR KEY ADDR
*
* FPTWRTF-WRITE RECORD VIA F:TF FOR )COPY
*
FPTWRTF   GEN,8,7,17 X'11',0,F:TF  DCB ADDRESS
DATA      X'F4000010' P1 TO P4,P6 AND WAIT
DATA      ERRFTF      ERR (P1)
DATA      ERRFTF      ABN (P2)
GEN,1,31 1,R11        BUF (P3)
GEN,1,31 1,R10        SIZ (P4)
DATA      0           BTD (P6)
*
* FPTREWTF- REWIND(THAT IS, GO TO BOF) TFILE OPEN ON F:TF
*
FPTREWTF  GEN,8,7,17 X'01',0,F:TF
*
* FPTPREC-BKCK UP ON TFILE FOR )COPY
*
FPTPREC   GEN,8,7,17 X'1D',0,F:TF
DATA      X'80000010' P1 AND REVERSE SKIP
GEN,1,31 1,NAMEGRN1   # OF RECORD TO SKIP IN NAMEGRN1
*
* FPTOFF-EXIT VIA M:LDTRC-TO 'LOGON',WHICH LOGS OFF
*
FPTOFF    GEN,8,22,2 X'03',0,2    ACCOUNT SPECIFIED
TEXTC     'LOGON'
TEXT      ':SYS '                ACCOUNT
*
* FPTMSG-OPERATOR MESSAGE
*
FPTMSG    DATA      0             IDENTIFIES 'MESSAGE' CAL
DATA      X'80000000'            PARAMETER '
DATA      IMAGE          ADDRESS OF MESSAGE
PAGE
*
* GETTERM-ROUTINE TO LOAD NEW SET OF I/O TRANSLATE TABLES
*
* ON ENTRY,R7=TERMINAL NO.
*
GETTERM   RES        0
LI,R5     X'1B0F'     ESC-SI
LI,R6     13          CHECK IF LEAVING
CW,R6     TERMTYPE    4013
BNE       GETT2       NO
GETT1     STH,R5      GRAFBUF      YES-SWITCH IT TO TTY MODE
CAL13SW   CAL1,1     FPTWR2
B         GETT3
GETT2     LI,R5       X'1B0E'     ESC-SO
CI,R7     13          CHECK IF ENTERING 4013
BNE       GETT3       NO-PROCEED
CALSET37  CAL1,8     FPTSET37     YES,SET TO TTY37
B         GETT1       AND SWITCH TO APL MODE
FPTSET37  DATA      X'06000002'  DECLARE TTY 37 TERMINAL
GETT3     LI,R6       3
STW,R7    TERMKEY    SET KEY
STB,R6    TERMKEY    AND KEY SIZE
CALOPTRM  CAL1,1     FPTOPTRM     OPEN TERMINAL FILE
CALRDTRM  CAL1,1     FPTRDTRM     READ RECORD
XW,R7     TERMTYPE    SET TERMTYPE-GET OLD
STW,R7    TERMKEY    SAVE OLD
LI,R5     F:TF
BAL,R6    CLOSV       CLOSE FILE
LW,R7     TERMKEY    GET OLD
B         OUTORANG+1  ISSUE 'WAS' MESSAGE
PAGE
*
* DELAY6-DELAY 6 SECONDD3-USED BY @OPR
*
DELAY6    RES        0
CALWAIT6  CAL1,8     FPTWAIT6
B         *R14
FPTWAIT6  DATA      X'0F000005'  5*1.2 SECONDS
PAGE
*

```

```

* GENOPRM-GENERATE OPERATOR MESSAGE INDICATED IN 'IMAGE'
*
GENOPRM RES 0
CALMSG CAL1,2 FPMESG UTS CALL
B *R14
PAGE
PAGE
*
* ROUTINES TO SET-UP AND EXECUTE OPS ON F:WS FOR
* )SAVE
* )LOAD
* )COPY OR )PCOPY
* )CONTINUE OR AUTOMATIC CONTINUE
*
* RESACCT- RESET ACCOUNT
*
* R14=LINK R4-R5 USED
*
RESACCT LI,R4 2 RESET ACCOUNT CONTROL
LI,R5 0
STB,R5 OPWSACTC,R4
LW,R4 J:ACCN
LW,R5 J:ACCN+1
STD,R4 OPWSACT SET ACCOUNT TO USERS ACCOUNT
B *R14 EXIT
*
* RESPASS- RESET PASSWORD
*
* R14=LINK, R4-R5 USED
*
RESPASS LI,R4 2 RESET PASSWORD CONTROL
LI,R5 0
STB,R5 OPWSPASC,R4
STD,R5 OPWSPAS ZERO THE VALUE
B *R14
*
* SETACCT-SET ACCOUNT CONTROL AND VALUE
*
* R14= LINK, R4 AND R5 USED
*
SETACCT LD,R4 NAMEBUF SET ACCOUNT- 8 CHARS-TRAILING BLANKS
STD,R4 OPWSACT
LI,R4 2 AND ACCOUNT CONTROL
STB,R4 OPWSACTC,R4
B *R14 EXIT
*
* SETFNAME-MOVE FILENAME FROM NAMEBUF TO OPWSNAME & CONVERT TO TEXTC
*
*
* R14=LINK, R4 AND R5 USED
*
* ON ENTRY, R5=BYTE COUNT
*
SETFNAME LOCAL SETEXTC
CI,R5 11 SET FILE NAME IN OPWSNAME
BLE SETEXTC
LI,R5 11 MAX 11 CHARS USED
SETEXTC STB,R5 OPWSNAME SET BYTE COUNT (TEXTC FORM)
LW,R5 MBSNAME SET UP TO MOVE 11 BYTES
LI,R4 BA(NAMEBUF) FROM BA(NAMEBUF) TO
MBS,R4 0 BA(OPWSNAME)+1
B *R14 EXIT
MBSNAME DATA X'0B000001'+BA(OPWSNAME)
*
* SETPASS-SET PASSWORD CONTROL AND VALUE
*
* R14=LINK R4,R5 USED
*
SETPASS LD,R4 NAMEBUF GET PASS WORD
STD,R4 OPWSPAS
LI,R4 2 SET PASSWORD CONTROL
STB,R4 OPWSPASC,R4
B *R14 RETURN
*
* TESTACCT-TEST IF ACCOUNT NOT USERS-QUIT IF NOT
*
*
* R14=LINK R12,R13 USED

```

```

*
TESTACCT LW,R13    OPWSACTC
          CI,R13    X'FF00'
          BAZ      *R14          ACCT NOT SPECIFIED-NO SWEAT
          LW,R12    J:ACCN
          LW,R13    J:ACCN+1      CHECK USERS ACCT
          CD,R12    OPWSACT
          BE      *R14          OK
          B        ERLIBREF      NO DICE
*
* TEST IF PASSWORD SET-QUIT IF YES
*
* R14=LINK R13 USED
*
*
TESTPASS LW,R13    OPWSPASC      CHECK IF PASSWORD SET
          CI,R13    X'FF00'
          BAZ      *R14          NO
          B        ERBADCMD      YES-ERROR
*
* TESTOLDP-CHECKS IF FILE EXISTS WITH INDICATED ID
*
* OK IF NOT
*
* IF S0,CHECKS FURTHER FOR VALIDITY OF SAVE COMMAND
* R14=LINK R5,R6,R7 USED. IF ERROR EXIT, R8,R9 ALSO USED
*
*
TESTOLDP LI,R6     1
          STW,R6    OPWSMODE      SET INPUT MODE
CALOPWS1 CAL1,1    FPTOPWS      AND OPEN
          LI,R5     F:WS          OLD FILE EXISTS-CHECK COMMAND
          BAL,R6    CLOSV          VALIDITY
          LD,R6     OPWSNAME
          CD,R6     WSIDNAME
          BNE       QCONTINU      NOT WSID-CHECK CONTINUE
          LW,R6     OPWSNAME+2
          CW,R6     WSIDNAME+2
          BE      *R14          NAME MATCHES WSID- SAVE OVER-OK
          B        DONTSAVE      NO MATCH, DON'T SAVE.
SAVEXIT  LW,R6     CMNDTYPE      CHECK COMMAND
          AI,R6     NCMSD+CMDB-@@CONT VS CONTINUE(HOLD)
          BNE       QLOAD        NO
          MTW,0     HOLDFLG      YES-CHECK HOLD VS OFF
          BEZ      UTSIOFF
          B        UTSIOFFH
QCONTINU CD,R6     KONTINUE      CHECK '8CONTINU'
          BNE       DONTSAVE      NOPE-NO SAVE
          B        *R14          YES-CHANCE IT
QLOAD    AI,R6     @@CONT-@@LOAD CHECK COMMAND VS LOAD
          BNE       CMDEXIT      NO-CAN'T BE AUTOSTART
          LW,R6     CURRCS       CHECK FOR AUTOSTART
          BEZ      CMDEXIT      NO
          BAL,R15   BCX          YES-- AND AWAY WE GO ---
*
* OPENSARE OPEN F:WS IN OUTPUT MODE FOR SAVE
*
* R14=LINK R13 USED
*
*
OPENSARE LI,R13    2            SET OUT MODE
          STW,R13   OPWSMODE
CALOPWS2 CAL1,1    FPTOPWS      OPEN
          B        *R14
*
* SAVWRITE-WRITE A 'SAVE' RECORD VIA F:WS
*
* R7=LINK R11=BUFFER ADDRESS(WA) R12=KEY
* SIZE IS FIXED-512 WORDS (2048 BYTES)
*
SAVWRITE STW,R12    FWSKEY      SAVE KEY (CAN'T BE IN REGISTER)
CALWRWS  CAL1,1    FPTWRWS      WRITE RECORD
          B        0,R7        RETURN
*
* CPYWRITE-WRITE A WS RECORD VIA F:TF FOR )COPY
*
* R7=LINK R11=BUFFER ADDRESS(WA)
* SIZE IN R10 (BYTES)
*
CPYWRITE RES      0
CALWRWF  CAL1,1    FPTWRWF      WRITE RECORD

```

```

      B      0,R7      RETURN
*
*   SAVDMES-ISSUE 'SAVED' MESSAGE AND TAKE COMMAND EXIT
*
COPYDMES  LI,R2      0
          B          COPYEXIT      SET OFFSET TO ZERO
SAVDMES   LW,R1      LODYN          SET R1 FOR SAVE OR LOAD
COPYEXIT  BAL,R14    WCKDSPL       CHECK OUT THE WORKSPACE
          B          LCI3          OK-CONTINUE
LCI16     LCI        0            ERROR RETURN
          STM,R0     INBUF+40      SAVE REGISTERS
LCI3       LCI        3
          LM,R3      OPWSNAME      GET NAME (+BLANKS)
          LD,R6      SAYVE         AND ' SAVED '
          LCI        4            GET TIME-DATE
          LM,R8      8,R1
          LCI        9
          STM,R3     IMAGE         SET UP IMAGE
          LI,R9      X'40'
          STB,R9     IMAGE         SET INITIAL BLANK
          CI,R14     LCI16         IS WS OK...
          BE         SAVEMESS      NO, DIS PLAY SAVED MSG.
          LW,R3      QUIETFLG      YES, CK QUIET-FLAG...
          BNEZ       SAVEXIT       = -1 -- EXIT NOW.
SAVEMESS  LI,R3      36           DISPLAY SAVED MSG.
          BAL,R12    DUMPLINE
          CI,R14     LCI16         WAS WS OK...
          BNE       SAVEXIT       YES -- NORMAL EXIT.
          LCI        0            NO -- ERROR EXIT AFTER REGS RESET.
          LM,R0     INBUF+40
          B          ERBADWS
*
*   FNEQWSID-SET FUNCT NAME WITH WSID
*
*   R14=LINK, R4 TO R10 MAY BE USED
*
*   IF WS IS CLEAR,(IN NAME), TAKES DONTSAVE EXIT
*   IF WSID HAS NO PASSWORD, EXITS VIA RESPASS
*   IF WSID HAS PASSWORD, EXITS VIA SETPASS
*
FNEQWSID  LCI        3
          LM,R6      WSIDNAME      GET WSID
          STM,R6     OPWSNAME      PUT IN FPT
          CD,R6      CLEARWS       CHECK IF 'CLEAR'
          BE         DONTSAVE      YES-NO SAVE
          LD,R4      WSIDPASS      GET WSID PASSWORD
          BEZ        RESPASS
          B          SETPASS+1
          PAGE
*
*   TRYLOAD-TRY TO OPEN FILE FOR LOAD (OR COPY)
*
*   R14=LINK
*
*   ROUTINE OPENS FILE AND READS ID RECORD.
*   IF NO SUCCESSFUL OPEN OR ID RECORD DOES NOT
*   INDICATE A VALID WS, ERROR EXIT IS TAKEN
*
TRYLOAD   LI,R6      1
          STW,R6     OPWSMODE      SET INPUT MODE AND OPEN
          LI,R5      F:WS          (USED BY CLOSV)
          LI,R6      ERLIBREF      EXIT FROM CLOSV
CALOPWS3  CAL1,1     FPTOPWS      TRY TO OPEN
          LW,R10     KEY1
          STW,R10    FWSKEY
          LI,R10     IDRECSIZ      SIZE-(DELIBERATELY SML)
          LI,R11     IDBUF         BUFFER ADDRESS
CALRDWSI  CAL1,1     FPTRDWS      READ ID RECORD
          B          CLOSV         NOT ABNORMAL-WRONG!
IDRECSIZ  EQU        64          ID RECORD SIZE-16 WORDS
*
*   LOADREAD-READ A RECORD FOR )LOAD OR )COPY
*
*   R14=LINK,R11=BUFFER ADDRESS,R12=KEY
*   R10=SIZE IN BYTES
*
LOADREAD  STW,R12    FWSKEY
CALRDWS   CAL1,1     FPTRDWS      READ RECORD

```

```

      B      *R14
*
* COPYREAD-READ A SEQUENTIAL RECORD FOR )COPY VIA F:TF
*
* R14=LINK,R11=BUFFER ADDRESS (WA)
* R10=SIZE IN BYTES
COPYREAD RES 0
CALRDTF CAL1,1 FPTRDTF
      B      *R14
      PAGE
*
* RDCOPY-ROUTINE TO READ RECORDS FROM F:WS FOR )COPY
*
* R14=LINK FREETBL=BUFFER ADDRESS COPSIZE=# OF WORDS TO BE READ
*
* R2 AND R5 THROUGH R12 USED-NOT SAVED
*
* IF,AS IS PROBABLY THE CASE, A 'SHORT' RECORD IS READ, ERRFWS6 WILL
* BE REACHED FROM ERRFWS
*
* MOST OF WORK IS DONE BY RDWRCOPY-COMMON TO RDCOPYDR,WRCOPYDR,RDCOPY
*
RDCOPY LW,R11 FREETBL BUFFER ADDRESS
      LW,R12 KEY1 FIRST KEY
      LI,R2 LOADREAD READ VIA F:WS
      B RDWRCOPY
*
* WRACTIV-WRITE ACTIVE WS ON TEMP FILE FOR )COPY
*
* R14=LINK
*
* REGISTERS R2 THROUGH R14 USED-NOT SAVED
*
* WRITES SEQUENTIAL RECORDS:
* STARTS WITH RECORD # 1
*
* SAVES FIRST 6 PARAMETERS FOR ACTIVE WS
* AND SETS NAMEGRN1=0 TO INDICATE NO NAME RECORDS IN FILE
*
WRACTIV STW,R14 SAVE14 SAVE R14
CALOPTF CAL1,1 FPTOPTF OPEN TFILE-OUTIN MODE-SEQUENTIAL
      BAL,R11 RELEASER RELEASE UNUSED CORE
      LI,R2 CPYWRITE-SAVWRITE SET TO CALL CPYWRITE
      BAL,R14 WRITEWS WRITE ACTIVE WS
      LCI 6
      LM,R6 *LODYN
      STM,R6 SAVESIX SAVE FIRST 6 WS PARAMETERS
RESNAMRC LI,R12 0
      STW,R12 NAMEGRN1 INDICATE NO NAME RECORDS PRESENT
      B *SAVE14 RETURN
*
* RDACTIV-READ ACTIVE WS FROM TFILE FOR )COPY
*
* R14=LINK
*
* REGISTERS R1 TO R14 USED-NOT SAVED
*
RDACTIV STW,R14 SAVE14 SAVE R14
CALREWTF CAL1,1 FPTREWTF POSITION TO BEGINNING OF FILE
      LI,R2 COPYREAD-LOADREAD SET UP FOR COPY-READ
      LI,R4 0
      STW,R4 WSOFFSET RESET WS OFFSET
      LI,R4 SAVESIX ADDRESS OF WS PARAMETERS
      BAL,R14 READWS READ WS
      B RESNAMRC SET FLAG NOT TO BACK-FILE ON NAMERS
*
* RDNAMER-
* WRNAMER-
*
* SHARED ROUTINE
* FOR RDNAMER, READS NAME RECORD-512 WORDS-VIA WINDOW
* FOR WRNAMER,WRITES NAME RECORD-512 WORDS-VIA WINDOW
*
* R14=LINK R2,R7,R10,R11,R12 USED-NOT SAVED
* NAMERKEY=NAME RECORD # 0-K INCREMENTED BY RDNAMER-WRNAMER
* NAMEGRAN1=GRANULE # FOR 1ST NAME RECORD
*
WRNAMER LW,R7 R14 SET EXIT FROM CPYWRITE
      LI,R2 CPYWRITE SET I/O FUNCTION

```

```

RDNAMER B NAMER1
LI,R2 COPYREAD SET I/O FUNCTION
NAMER1 LW,R11 NAMERKEY CHECK IF FIRST NAME RECORD OF SET
BNEZ NAMER2 NO
LW,R10 NAMEGRN1 YES-CHECK IF PRIOR NAME RECORDS
BEZ NAMER2 NO-FILE POSITIONING NOT NEEDED
CALPREC CAL1,1 FPTPREC BACK UP 'NAMEGRN1' RECORDS
STW,R11 NAMEGRN1 RESET 'NAMEGRN1' TO ZERO
NAMER2 LI,R11 WINDOW SET BUF
LI,R10 2048 SET SIZ
MTW,1 NAMEGRN1 INCREMENT UTSC NAME RECORD COUNT
MTW,1 NAMERKEY INCREMENT NAME RECORD COUNT
B 0,R2 DO I/O AND RETURN
*
* RDCOPYDR-READ COPY DATA 'RECORD' (ACTUALLY MAY BE SEVERAL RECORDS)
*
* R14=LINK COPYSIZE=# OF WORDS COPYHOME=1ST ADDRESS
*
* CLOSE F:TF AFTER READ
*
* MOST REGISTERS USED, NONE SAVED
*
* ROUTINE RDWRCOPY DOES MOST OF WORK
*
RDCOPYDR LW,R11 COPYHOME ADDRESS
LI,R2 COPYREAD TYPE OF OPERATION
B RDWRCOPY
*
* WRCOPYDR-WRITE COPY 'DATA RECORD' (ACTUALLY MAY BE SEVERAL RECORDS)
*
* R14=LINK COPYSIZE=# OF WORDS COPYBASE=1ST ADDRESS
*
* MOST REGISTERS USED,NONE SAVED
*
*
WRCOPYDR LW,R11 COPYBASE ADDRESS
LI,R2 CPYWRITE TYPE OF OPERATION
LI,R7 RDWRL00P EXIT FROM CPYWRITE
RDWRCOPY STW,R14 SAVE14 SAVE R14
LI,R14 RDWRL00P EXIT FROM LOADREAD OR COPYREAD
LW,R9 COPYSIZE SIZE (WORDS)
LI,R10 2048 PHYSICAL RECORD SIZE
RDWRCHSZ CI,R9 512 CHECK SIZE
BGE 0,R2 FULL RECORD-READ OR WRITE
AI,R9 0
BLEZ RDWREXIT DONE-EXIT
LW,R10 R9
SLS,R10 2 SHORT RECORD
B 0,R2 READ OR WRITE
RDWRL00P AI,R11 512 KICK ADDRESS
AI,R9 -512 REDUCE SIZE
AI,R12 1 INCREMENT GRANULE # OR KEY
B RDWRCHSZ CONTINUE
RDWREXIT CI,R2 CPYWRITE CHECK IF COPY WRITE
BE *SAVE14 YES-EXIT
ERRFWS6 LI,R5 F:WS (REACHED BY RDWREXIT OR FROM ERRFWS)
LW,R6 SAVE14
CI,R2 LOADREAD CHECK IF READ OF F:WS FOR COPY
BE CLOSV YES-CLOSE AND SAVE AND EXIT
LI,R5 F:TF NO-READ OF F:TF FOR COPY
B CLOSR CLOSE AND RELEASE AND EXIT
PAGE
*
* DROPFILE-DELETE INDICATED FILE
*
* R14=LINK R5,R6,R7 VOLATILE
*
* IF FILE NOT FOUND-ERROR EXIT
*
DROPFILE LI,R6 1
STW,R6 OPWSMODE
CALOPWS4 CAL1,1 FPTOPWS
LI,R5 F:WS
BAL,R6 CLOSR
B *R14
PAGE
*
* LIBUTS-UTS INTERFACE FOR )LIB COMMAND-WHICH IS ESSENTALLY ALL A

```

```

*          UTS INTERFACE OPERATION
*  MUCH OF WORK IS DONE BY ERRFTF1 AND PRINTFMN, IN APLUTSI
*  ERROR HANDLERS, SINCE )LIB OUTPUT IS GENERATED BY
*  DELIBERATE 'ABNORMAL' OPENS AND READS
*
*  EXIT IS FROM ERROP1ST TO CMDEXIT
*
LIBUTS  RES      0
        LI,R5    0          PRESET 'NO' ACCOUNT
        CI,R2    X'15'      CHECK CR
        BE       S1STACC    YES
        BAL,R14  ACCTCHK    GET ACCOUNT          18-00006
        LD,R4    NAMEBUF    YES
        STD,R4   OP1STACT   SET ACCOUNT
        LI,R5    2          SET ACCOUNT CONTROL
S1STACC LI,R4     2
        STB,R5   OP1STACC,R4
CALOP1ST CAL1,1  FPTOP1ST   OPEN 1ST FILE IN ACCOUNT.
        B        CHKWRACT
CLOSTHIS LI,R5    F:TF      CLOSE DCB (IF OPEN)          18-00021
        BAL,R6   CLOSV
        LW,R6    BREAKFLG  CHECK BREAK
        BNEZ     CMDEXIT   YES-EXIT
CALOPNXT CAL1,1  FPTOPNXT   OPEN- 'NEXT' FILE
CHKWRACT LI,R5    FPARAMS   SET TO SEARCH FPARAMS          18-00023
CHKWR1   LW,R6    0,R5      GET KEYWORD          18-00024
        CW,R6    WRTCHKNE   CHECK FOR WRITE ACCT-4 WORDS
        BE       CHKWRAPL  (NOT LAST FILE PARAM)
        CW,R6    OPWSWRTC   CHECK FOR WRITE ACCT-4 WORDS          18-00025
        BNE      CHKWR2    NO          18-00026
CHKWRAPL LW,R7    4,R5      YES-GET 3RD AND 4TH WORDS
        LW,R6    3,R5
        CD,R6    APLAPLAP   CHECK IF APLAPLAP          18-00028
        BNE      CLOSTHIS  NO-NOT WS          18-00029
        B        PRINTFMN  YES-ASSUME WS          18-00030
        B        PRINTFMN  YES-ASSUME WS          18-00031
WRTCHKNE DATA   X'06000404' 4-WORD WRITE ACCTS-NOT LAST PARAM
CHKWR2   CI,R6    X'10000'   CHECK IF LAST KEYWORD          18-00032
        BANZ     CLOSTHIS  YES-GO TO NEXT FILE          18-00033
        AND,R6   XFF        NO-GET WORD COUNT          18-00034
        AW,R5    R6         MOVE POINTER          18-00035
        AI,R5    1          18-00036
        B        CHKWR1    AND LOOP          18-00037
        PAGE
*
*  UTSIOFF AND UTSIOFFH  EXITS
*
UTSIOFFH RES      0          OFF-HOLD
        BAL,R14  CLOSFILS   CLOSE BLIND I/O FILES
        B        TELEXIT   RETURN TO TEL
UTSIOFF  RES      0
        BAL,R14  CLOSFILS   CLOSE BLIND I/O FILES
        LW,R12   LODYN
        BAL,R14  RELDYN     RELEASE ALL 'DYNAMIC'
        LW,R14   HICOMMON
        AI,R14   1
        STW,R14  TOPOSTAK   RELEASE ALL 'COMMON'
        BAL,R14  RELCOM
CALXCRES CAL1,8  FPTXCOFF   RESET EXIT CONTROL
        BAL,R15  CHKTERM    CHECK IF 4013, IF SO, SWITCH
CALOFF   CAL1,8  FPTOFF     EXIT-ALL THE WA-A-A-Y!
*
*  CLOSFILS-CLOSE BLIND I/O DCB'S WITH SAVE, PRIOR TO EXIT FROM APL
*
CLOSFILS LI,R4    NBIO+NUMFILES SET LOOP FOR TABLE SEARCH
        LW,R5    FQTABL-2,R4 GET DCB ADDRESS
        BAL,R6   CLOSV      CLOSE AND SAVE (IF OPEN)
        BDR,R4   CLOSFILS+1 LOOP
        B        *R14
        END

```


TITLE	'CODEXEQ-B00,08/22/73,DWG702985'		
SYSTEM	SIG7F		
CSECT	1		
PCC	0		
CODEXEQ@ RES	0	CONTROL CARDS NOT PRINTED. ORIGIN OF CODESTRING EXEC. MODULE.	

*	REF'S	AND	DEF'S	
*				
	DEF		CODEXEQ@	= START OF 'CODEXEQ' MODULE.
	DEF		CX	CODESTRING EXECUTION ENTRY PT.
SPACE 2				
	DEF		XEQEMPTY	RETURN PT WHEN AN 'EXECUTE' IS OK, BUT PRODUCED NO CODESTRING, E.G. A SUCCESSFUL COMMAND EXECUTION.
*				
*				
	DEF		LOBRNCK	ENTERED FOR ERR-CTRL BRANCH.
	DEF		OBSERVED	RESUME AFTER AN OBSERVATION.
	DEF		RESQUAD	RESUME AFTER QUOTE-QUAD INPUT.
	DEF		QDBDONE	RESUME AFTER QUAD-ZERO INPUT.
	DEF		QOUTRET	RESUME AFTER QUAD-ZERO OUTPUT.
	DEF		RETURNMX	RETURN FROM MONADIC-OP EXECUTION.
	DEF		RETURNDX	DYADIC-OP EXECUTION.
	DEF		RETURNNI	NILADIC INTRINSIC FUNC.
	DEF		RETURNSX	SUBSCRIPT EXECUTION.
	DEF		RETURNAX	ASSIGNED-INDEXING EXEC.
SPACE 3				
*				

REF	F2I	REFS TO PROCEDURE:	FLOATING-TO-INTEGER CONVERSION.
REF	CTEST		TRIES TO GET MORE COMMON
REF	ALOCBLK		ALLOCATE DATA BLOCK.
REF	ALOCHNW		ALLOCATE HEADER PLUS N WDS.
REF	ALOCRS		ALLOCATE RESULT DATA BLOCK.
REF	ALOCTRES		ALLOCATE TEXT TYPE RESULT DATA BLK.
REF	DREF		DEREFERENCE A DATA BLOCK.
REF	MAYDREF		DEREFERENCE A DATA BLOCK, IF ANY.
REF	ERDOMAIN		DOMAIN ERROR.
REF	ERRANK		RANK ERROR.
REF	ERLENGTH		LENGTH ERROR.
REF	SYSTERR		SYSTEM ERROR HANDLER (TOTALLY LOST).
REF	OPBREAK		STARTS TBL OF ERROR ENTRY PTS.
REF	SICLR%		CLEAR STATE INDICATOR TO GO-STATE.
REF	BREAKFLG		BREAK FLAG (NZ IF BRK OR HANG-UP).
REF	MXDRIVER		MONADIC OPERATOR EXECUTION DRIVER.
REF	DXDRIVER		DYADIC OPERATOR EXECUTION DRIVER.
REF	SXDRIVER		SUBSCRIPTING EXECUTION DRIVER.
REF	AXDRIVER		INDEXED ASSIGNMENT EXECUTION DRIVER.
REF	INNER		INNER PRODUCT EXECUTION DRIVER.
REF	OUTER		OUTER PRODUCT EXECUTION DRIVER.
REF	REDUCE		REDUCTION OPERATOR EXECUTION DRIVER.
REF	SCAN		SCAN OPERATOR EXECUTION DRIVER.
REF	NINTRIN		NILADIC OPERATOR EXECUTION DRIVER.
REF	MINTRIN		MONADIC OPERATOR EXECUTION DRIVER.
REF	DINTRIN		DYADIC OPERATOR EXECUTION DRIVER.
REF	INPDIR		REQUESTS DIRECT INPUT.
REF	INPEVAL		REQUESTS EVALUATED INPUT.
REF	INPQUAD		REQUESTS QUOTE-QUAD INPUT.
REF	INPXEQ		DOES INPUT-TRANS. FOR EXECUTE OPER.
REF	INPBLIND		REQUESTS BLIND INPUT.
REF	QZIN		QUAD-0 (GRAPHICS) INPUT START PT.
REF	QZOUT		QUAD-0 (GRAPHICS) OUTPUT START PT.
REF	MIXEDOUT		STMT DISPLAYER (MIXED OUTPUT).
REF	SINGOUT		DISPLAY SINGLE DATA BLOCK VALUE.
REF	BLINDOUT		OUTPUTS BLINDLY
REF	SHOWSTOP		DISPLAYS STOPPED-FUNC.NAME & LINE #.
REF	FUNLDISP		DISPLAYS FUNC.NAME & LINE NUMBER.
REF	OBSERVER		DISPLAYS AN OBSERVATION.

*			
	REF	OPER	REFS TO CONTEXT:
	REF	CXDTEMP	DBLWD TEMP FOR OPERATOR INFO.
	REF	FDEFPTR	DBLWD TEMP USED BY CODESTRING EXEC.
	REF	XTEMP	(DB PTR) FUNCTION DESCRIPTOR.
	REF	NILCK	(DB PTR) INDEX OR COORD.
	REF	LFARG	(DB PTR)
	REF	RTARG	(DB PTR) LEFT ARG.
	REF	RESULT	(DB PTR) RIGHT ARG.
	REF	CURRCS	(DB PTR) RESULT.
	REF	RSSIZE	PTS AT CURRENT CODESTR. DATA BLK +2.
	REF	RSTYPE	HOLDS NO.OF ELEMENTS FOR RESULT D.B.
	REF	RSRANK	USED TO HOLD TYPE OF RESULT DATA BLK
			USED TO HOLD RANK OF RESULT DATA BLK

U19-0004

	REF	BRNVAL	BRANCH VALUE.
	REF	GOSTATE	GO-STATE FOR STATE INDICATOR CLEARS.
	REF	CATCHTBL	CATCH TABLE.
	REF	OBSFLAG	OBSERVE-CMD FLAG (0=NO & -1=YES).
	REF	OBSERVE	OBSERVATION SETTING (0=NO & -1=YES).
	REF	STATEPTR	PTS AT TOP STATE-ENTRY IN STACK.
	REF	TOPOSTAK	OFTEN PTS AT TOP OF EXECUTION STACK.
	REF	ORGADJ	VALUE USED IN ADJUSTING TO ORIGIN 1.
	REF	OFFSET	OCCAS. USED TO SAVE CURR. CS OFFSET.
	REF	CXSCRTCH	SCRATCH AREA FOR CODESTRING EXEC.
	REF	INBUF	INPUT BUFFER.
	REF	IMAGE	INTERNAL IMAGE OF INPUT LINE.
	REF	SYMT	PTS AT 1ST WD OF SYMBOL TABLE.
	REF	NAMEPTR	PTS TO REF-INDIC WD FOR A VAR. NAME.
	REF	BLINBUF	BLIND INPUT BUFFER
	REF	STKLIMIT	CURRENT LIMIT TO EXEC.STACK.
	REF	LOCNEED	LOC (+1) NEEDED FOR COMMON INCREASE.
*			REFS TO CONSTANTS:
	REF	X1FFFF	X'1FFFF'
	REF	XFF0000	X'FF0000'
	REF	BITPOS	32-WD TBL OF BITS (BITPOS-K CONTAINS A WD HAVING A 1 ONLY IN BIT POS K)
*			
	REF	FUNTYPES	RANGE OF FUNCTION TYPES.
*			
*	EQU'S RELATED TO THE CONTEXT REF'S.		
*			
OPERHOLD	EQU	CXDTEMP	HOLDS OPER INFO TILL OTHER OP EXECD.
XPARAMS	EQU	CXDTEMP	INDEX SEQUENCE PARAMETERS--
XADD	EQU	XPARAMS	ADDITIVE FACTOR.
XMULT	EQU	XPARAMS+1	MULTIPLICATIVE FACTOR.
DENTRY	EQU	BITPOS-4	DY. FUN CAT.OF EXEC.STACK ENTRY.
BENTRY	EQU	BITPOS-5	BRACKET CAT.OF EXEC.STACK ENTRY.
OENTRY	EQU	BITPOS-6	OPERATOR CAT.OF EXEC.STACK ENTRY.
COORD1	EQU	BITPOS-7	COORDINATE-1 (FOR OPS LIKE %RV1).
PTRITEM	EQU	LFARG	FOR PTR SCANITEM.
LINKX	EQU	CXSCRTCH+0	LINK TO 'SUBEX'.
LINKMX	EQU	CXSCRTCH+0	LINK TO 'MONEX'.
ACFLAG	EQU	CXSCRTCH+0	ASSIGNMENT-COMPLETED FLAG.
XSIZE	EQU	CXSCRTCH+0	EXECUTION STACK SIZE FOR FUN.CALL.
STSET	EQU	CXSCRTCH+0	STOP OR TRACE BIT SETTING.
SCANTEMP	EQU	CXSCRTCH+1	SCAN ITEM HOLDER.
NFL4ST	EQU	CXSCRTCH+1	NO.OF FUN LINES FOR STOP OR TRACE.
OFFSETR	EQU	CXSCRTCH+2	OFFSET FOR RE-SCAN OF DESIGNATOR.
LINKT	EQU	CXSCRTCH+2	LINK HOLDER FURING TRACED SHOW SET.
FDPTR	EQU	CXSCRTCH+2	FUN DESCRIPTOR LINE-PTR PTR.
OPCURRB	EQU	CXSCRTCH+3	CURR.BYTE HOLDER FOR OPWRAP.
*			
*	STANDARD EQU'S		
*		REGISTERS	
R1	EQU	1	
R2	EQU	2	
R3	EQU	3	
R4	EQU	4	
R5	EQU	5	
R6	EQU	6	
R7	EQU	7	
R8	EQU	8	
R9	EQU	9	
R10	EQU	10	
R11	EQU	11	
R12	EQU	12	
R13	EQU	13	
R14	EQU	14	
R15	EQU	15	
*			
*	EXECUTION STACK CATEGORIES		
*			
CATV	EQU	0 (NOT USED MUCH)	VALUE PTR.
CATAPRYM	EQU	1	ASSIGNMENT (COMPLETED).
CATO	EQU	2	OPERATOR.
CATX	EQU	3	INDEX (LIST PTR).
CATB	EQU	4	BRACKETED.
CATP	EQU	5	PARENTHESESIZED.
CATS	EQU	6	STMT.
CATQ	EQU	7	QUAD-STATE (FOR EVAL-INPUT).
CATD	EQU	8	DYADIC FUNC. (WITHOUT LEFT ARG).
CATA	EQU	9	ASSIGNMENT (NOT YET COMPLETED).
CATE	EQU	10	FUNCTION-STATE.

```

*   SCAN ITEM INDICATORS:
*
SCANPTR EQU      0 (NOT USED MUCH) PTR
SCANVN EQU      1 VN
SCANBRN EQU     SCANVN+1 BRN
SCANBOS EQU     SCANBRN+1 BOS
SCANLP EQU     SCANBOS+1 LP
SCANLB EQU     SCANLP+1 LB
SCANSEMI EQU    SCANLB+1 SEMI
SCANOPM EQU    SCANSEMI+1 OPM
SCANOP EQU     SCANOPM+1 OP
*
*   DATA BLOCK TYPES:
*
TYPELOGL EQU      1 LOGICAL
TYPETEXT EQU      2 TEXT
TYPEINTG EQU      3 INTEGER
TYPEREAL EQU      4 REAL
TYPELIST EQU      6 LIST
TYPENFN EQU      8 FUNC.DESCR, NILADIC & NO RESULT.
TYPEMFN EQU     X'A' FUNC.DESCR, MONADIC & NO RESULT.
TYPEDI EQU     X'E' DYADIC INTRINSIC
TYPEMI EQU     X'F' MONADIC INTRINSIC
TYPENI EQU     X'10' NILADIC INTRINSIC
*
*   OTHER EQU'S
*
*   OFFSETS INTO A FUNCTION DESCRIPTOR DATABLK
*
XSIZEOFF EQU      2 EXEC. STACK SIZE FOR THE FUNCTION.
RESOFF EQU      XSIZOFF+1 RESULT NAME PTR.
LFOFF EQU      RESOFF+1 LEFT ARGUMENT NAME PTR.
FNOFF EQU      LFOFF+1 FUNCTION-NAME NAME PTR.
RTOFF EQU      FNOFF+1 RIGHT ARGUMENT NAME PTR.
NFOFF EQU      RTOFF+1 NO.OF FUNCTION LINES.
LINSTART EQU     NFOFF+1 LINE PTRS START HERE.
*
CODESTRNG DESIGNATIONS.
QZCODE EQU      25 QUAD-0
Q1CODE EQU      26 QUAD-1
QUADCODE EQU     35 QUAD
STOPNMCD EQU     21 STOPNAME
COMMACOD EQU     87 COMMA OPERATOR (FOR LAMINATION).
BOSCODE EQU     37 BEGINNING-OF-STMT.
SEMICODE EQU     38 SEMICOLON
SMOCODE EQU     46 SMALL CIRCLE
SCAN1COD EQU     118 SCAN OPERATOR (ON 1ST COORD) U19-0006
XEQCODE EQU     120 EXECUTE OPERATOR (EPSILON).
DUMMYCD EQU     138 CODE FOR DUMMY-USED FOR SPACER
*
MISC.
STOPBIT EQU     X'40000' POS.OF STOP FLAG BIT.
TRACEBIT EQU    X'20000' POS.OF TRACE FLAG BIT.
FLINFLG EQU    X'10000' FUNCTION-LINE FLAG BIT (IN S-ENTRY).
DYINFLAG EQU    X'200' DYADIC INTRINSIC OPERATOR FLAG (IN
* 0-ENTRY -- OPTYPE'S HI BYTE).
PENDFLAG EQU    X'8000' PENDENT FLAG BIT (IN F OR Q ENTRY).
EFLAG EQU      X'10000' MARKS 'EXECUTE' TYPE OF Q-ENTRY.
NEWLINE EQU     X'15' END-OF-INPUT CHARACTER.
LBLFLAG EQU     X'20000' LABEL FLAG BIT (IN SYM TBL REFERENT
* INDICATOR WD).
TOPRANK EQU     63 MAXIMUM POSSIBLE RANK.
DYADIC EQU      80 1ST DYADIC OP CODESTRNG DESIGNATION
MONADIC EQU     50 1ST MON. OPER CODESTRNG DESIGNATION
*
*   DOUBLEWORD CONSTANTS
*
BOUND 8
DYSCLRS DATA 91,111 RANGE OF DYADIC SCALAR CODE DESIGS.
MDYSCLRS DATA 91,101 MONADIC DYSCLRS.
MONRANGE DATA 50,101 POSS. MONADIC OPERATOR CODE DESIGS.
DYRANGE DATA 80,120 DYADIC OPERATOR CODESTR. DESIGNAT'S.
REDRANGE DATA X'100'+91,X'100'+111 RANGE OF REDUCTION OP-TYPES.
S2XRANGE DATA SCAN1COD,XEQCODE SCAN-THRU-EXECUTE OPERATOR RANGE. U19-0008
CATO2D DATA CATO,CATD RANGE OF CATEGORIES -- 0 THRU D.
STMASKS DATA STOPBIT,-STOPBIT-1 FLAG & MASK FOR -- STOP VECTOR.
DATA TRACEBIT,-TRACEBIT-1 -- TRACE VECTOR.
*
*   CONSTANTS
*

```

```

PENTRY  DATA  X'5000000'  PAREN-CATEGORY OF STACK ENTRY.
SENTRY  DATA  X'6000000'  STMT-CATEGORY OF STACK ENTRY.
FSENTRY DATA  X'6010000'  FUN-STMT-CATEGORY OF STACK ENTRY.
QPENTRY DATA  X'7008000'  QUAD-STATE CAT. OF STACK ENTRY
*                               (PENDENT).
AENTRY  DATA  X'9000000'  ASSIGNMENT-CAT. OF STACK ENTRY (NOT
*                               YET COMPLETED ASSIGNMENT).
FPENTRY DATA  X'A008000'  FUNC-STATE CAT. OF STACK ENTRY
*                               (PENDENT).

PAGE
*****
*
*  STACKUP -- PUSHES THE STACK UP 1 WD & VERIFIES THAT THIS IS BELOW
*  THE STACK LIMIT (I.E. ROOM FOR 2 WDS IF NEEDED).
*  REGS:   R1 -- (ENTRY) CURRENT TOP OF STACK.
*          (EXIT) NEW TOP OF STACK (UNFILLED).
*          R4 -- LINK, EXIT VIA 0,R4.
*          R8 IS VOLATILE.
*
STACKUP  AI,R1    -1          ADD 1 WD TO EXECUTION STACK.
         CW,R1    STKLIMIT   IS THIS STILL BELOW THE LIMIT.
         BG       0,R4      YES, EXIT.
         STW,R1   LOCNEED   NO, SAVE THE LOCATION NEEDED.
         AI,R1    1         RESTORE STACK PTR IN CASE WS FULL.
         BAL,R8   CTEST     TRY TO GET MORE COMMON...
         B        CXERWS    NO LUCK -- WS FULL.
         B        0,R4     OK, EXIT.

PAGE
*****
*
*  GENLIST -- GENERATES A LIST TYPE OF DATA BLOCK USING NILCK AND THE
*  N V-ENTRIES CONTAINED IN THE SERIES (FOR EXAMPLE, B-ENTRY)
*  AT THE TOP OF THE EXECUTION STACK.
*  REGS:   R11 (ENTRY) CONTAINS N.
*          R13 LINK, EXIT VIA *R13.
*          R4,R5,R11,R12,R14 ARE VOLATILE; SEE ALSO ALOCRS.
*          R1 (EXIT) PTS AT EXECUTION STACK ENTRY AFTER THE
*          N-TH V-ENTRY.
*
SPACE
LOCAL GENLIST1,GENLIST2
SPACE
GENLIST  LI,R12    1          NOTE: ASSUME RANK OF A LIST IS 1;
         STW,R12   RSRANK    (SEE ALSO DELTA-FMT).
         LI,R12   TYPELIST  FOR LIST TYPE DATA BLOCK.
         STW,R12   RSTYPE
         AI,R11    2          ACCOUNT FOR NILCK AND TRACKING WD.
         STW,R11   RSSIZE
         STW,R1    TOPOSTAK  SAVE STACK PTR. IN CASE WS FULL.
         STW,R3    OPER+1    SAVE OFFSET IN CASE WS FULL OCCURS.
         BAL,R14   ALOCRS    ALLOCATE 'RESULT' DATA BLOCK.
         STW,R4    PTRITEM   SAVE LOC OF DATA BLOCK AS PTR ITEM.
         LW,R5     RSSIZE    = NO.OF DATA PTRS IN THE LIST
         AI,R5     -1        BUT NOT COUNTING THE TRACKING WD
         STW,R5    2,R4      (I.E. LENGTH OF LIST).
         LW,R12    NILCK     GET 1ST DATA PTR (OR NIL).
         B         GENLIST2
GENLIST1 LI,R12    0
         XW,R12    0,R1      GET NEXT DATA PTR (OR NIL).
GENLIST2 AI,R4     1          INCR PTR INTO THE DATA BLOCK.
         STW,R12   2,R4      FILL IN A DATA PTR.
         AI,R1     1          PT AT NEXT ENTRY IN EXECUTION STACK.
         BDR,R5    GENLIST1
         STW,R5    3,R4      NOTE--LIST LENGTH MUST NOT START OUT ZERO;
*                               ZERO THE TRACKING WD.
*
B        *R13

PAGE
*****
*
*  NUMSV -- CHECK NUMERIC SCALAR OR VECTOR DATA BLOCK.
*  REGS:   R7 LINK, EXIT VIA 0,R7 FOR SCALAR OR
*          1,R7 FOR VECTOR.
*          R4 (ENTRY) PTS TO DATA BLOCK HEADER.
*          (EXIT) PTS AT DATA BLOCK HEADER + 3.
*          R6 (EXIT) CONTAINS NUMERIC TYPE INDICATOR:
*          1 FOR LOGL
*          2 FOR INTG
*          3 FOR REAL

```

```

*                                     4  FOR XSEQ (INDEX SEQUENCE)                                     *
*                                     R13 (EXIT) = 0.                                           *
*                                     R12 (VECTOR EXIT) CONTAINS LENGTH OF VECTOR                 *
*                                     (SCALAR EXIT) CONTAINS 'LENGTH' = 1.                     *
*                                                                                             *
SPACE
LOCAL  NUMTCK,RNKCK
SPACE
NUMSV   LB,R6  *R4      GET TYPE OF DATA BLOCK
NUMTCK  B      NUMTCK,R6 @ VECTOR ON THE TYPE -- 0 IMPOSSIBLE.
        B      RNKCK    @ 1 -- LOGL (NUMERIC TYPE 1)
        B      NUMCKNIL @ 2 -- TEXT, MAYBE EMPTY VECTOR.
        BDR,R6  RNKCK    @ 3 -- INTG (NUMERIC TYPE 2)
        BDR,R6  RNKCK    @ 4 -- REAL (NUMERIC TYPE 3)
        BDR,R6  RNKCK    @ 5 -- XSEQ (NUMERIC TYPE 4)
        B      CXERDOM   @ 6 -- LIST -- DOMAIN ERROR.
RNKCK   LW,R13  0,R4     DATA BLOCK'S HEADER WORD.
        AI,R4   3        PT AT DATA BLK HDR + 3.
        LI,R12  1        IF SCALAR, PREPARE LENGTH = 1.
        AND,R13 XFF0000  CK THE RANK...
        BEZ     0,R7     0, TAKE SCALAR EXIT.
        LW,R12  -1,R4    NZ, GET LENGTH WORD.
        AI,R13  -X'10000' VERIFY THAT RANK = 1...
        BEZ     1,R7     OK, TAKE VECTOR EXIT.
        B      CXERRANK  NO -- RANK ERROR.
NUMCKNIL LI,R13  -X'0201' MINUS(TYPE 2 RANK 1).
        AH,R13  *R4      = 0 IF TEXT VECTOR...
        BNEZ    CXERDOM  NOT VECTOR -- DOMAIN ERROR.
        AI,R4   3        PT AT DATA BLK HDR + 3.
        LW,R12  -1,R4    GET LENGTH...
        BEZ     1,R7     EMPTY VECTOR (ASSUME INTG) -- EXIT
        B      CXERDOM   OOPS -- DOMAIN ERROR.

PAGE
*****
*                                     *
* SHADOW -- PLACES A NEW REFERENT INDICATOR IN A GIVEN SYMBOL TABLE *
* ENTRY, PUSHES THE OLD REFERENT INDICATOR INTO THE EXECUTION *
* STACK, AND PUSHES THE SYMBOL TABLE ENTRY OFFSET (I.E. THE *
* NAME POINTER) INTO THE STACK. *
* REGS:  R6 -- LINK, EXIT VIA 0,R6. *
*        R4 -- (ENTRY) CONTAINS NEW REFERENT INDICATOR. *
*        R7 -- (ENTRY) CONTAINS THE NAME POINTER. *
*        R1 -- (ENTRY) PTS AT EXECUTION STACK LOC. *
*        (EXIT) PTS AT EXECUTION STACK LOC OF NAME PTR. *
*        R4 IS VOLATILE. *
*                                     *
SHADOW  XW,R4   *SYMT,R7  GET OLD, PLACE NEW REFERENT-INDIC.
        STW,R4  -1,R1     PUSH OLD REF-IND.
        STW,R7  -2,R1     PUSH NAME PTR.
        AI,R1   -2        UPDATE STACK PTR.
        B      0,R6      EXIT.

PAGE
*****
*                                     *
* SUSPCLR -- ENTERED FOR SUSPENSION-CLEAR AND DIRECT INPUT REQUEST. *
*                                     *
SUSPCLR LI,R3   PENDFLAG  = PENDENT FLAG.
        LW,R4   STATEPTR  PT AT TOP STATE ENTRY.
SUSPCLR LI,R5   X'7FFF'   EXTRACT 'NEXT' FIELD...
        AND,R5  0,R4      ZERO = FINAL STATE ENTRY.
        BEZ     SUSPCLRF  PT AT NEXT STATE ENTRY.
        AW,R4   R5        IS IT PENDENT...
        CW,R3   0,R4      YES, KEEP HUNTING.
        BANZ    SUSPCLRP  PT AT STATE TO CLEAR DOWN TO AND
SUSPCLRF STW,R4  GOSTATE   REQUEST DIRECT INPUT AFTER
        LI,R14  INPDIR    CLEARING THE STATE INDICATOR.
        B      SICLR%

PAGE
CXBREAK BAL,R2  INTERR    BRK DETECTED BY CODEXEQ.
CXLSEERR BAL,R2  INTERR
CXERSYN  BAL,R2  INTERR
CXERUND  BAL,R2  INTERR
CXERNOR  BAL,R2  INTERR
CXERIO   BAL,R2  IOERR
CXERDOM  BAL,R2  INTERR
CXERRANK BAL,R2  INTERR
CXERLEN  BAL,R2  INTERR
CXERWS   BAL,R2  INTERR

```

INTERR	STW, R1	TOPOSTAK	SAVE CURRENT TOP OF STACK.
IOERR	STW, R3	OPER+1	SAVE CURRENT BYTE OFFSET IN CODESTR.
	B	OPBREAK-CXBREAK-1, R2	GOTO APPR. ERROR-ENTRY PT.
PAGE			
A38	STW, R1	TOPOSTAK	SAVE STACK PTR.
	LW, R4	1, R1	GET DATA BLK PTR (I.E. V-ENTRY).
	AI, R6	-QZCODE	CHECK FOR QUAD-ZERO...
	BE	QZOUTSET	YES -- SET FOR GRAFIX OUTPUT.
	LI, R14	QOUTRET-1	NORMAL RETURN FROM OUTPUT IS QOUTRET
	AI, R6	QZCODE-QUADCODE	NOT QUAD-ZERO, WHICH IS IT...
	BGEZ	SINGOUT	QUAD OR QUOTE-QUAD (SINGLE D.B.).
	AI, R6	QUADCODE-Q1CODE+1	BLIND, GET QUAD NUMBER.
	LI, R5	TYPETEXT	
	CB, R5	*R4	VERIFY DATA BLK IS TEXT TYPE.
	BE	BLINDOUT	OK.
	B	CXERDOM	OOPS -- DOMAIN ERROR.
QZOUTSET	STW, R3	OPER+1	SAVE CS OFFSET TO THE QUAD CODE.
	STW, R4	RTARG	SET PTR TO ARG.
	MTW, 1	1, R4	BUMP ITS REF-COUNT.
	LW, R5	BREAKFLG	TEST FOR BREAK...
	BEZ	QZOUT	NO -- DO GRAFIX OUTPUT.
	B	CXBREAK	YES.
	B	CXERIO	@ ERROR RETURN FOR QUAD-OUT -- I/O ERR
QOUTRET	LW, R1	TOPOSTAK	@ RESTORE STACK PTR.
	LI, R2	CATAPRYM	SET TOP CAT = A-PRIME.
	STB, R2	*R1	
	BDR, R3	NXTSCAN	
A25	AI, R6	-QUADCODE	WHICH TYPE OF QUAD...
	BEZ	A25EVAL	EVALUATED-INPUT.
	STW, R1	TOPOSTAK	(SAVE STACK PTR).
	BLZ	A25Q	BLIND OR QUAD-ZERO.
	B	INPQUAD	QUOTE-QUAD INPUT;
RESQUAD	LW, R1	TOPOSTAK	RESUME AFTER QUOTE-QUAD INPUT TRY.
	AI, R13	0	AFTER PTING AT TOP OF STACK, CK FOR:
	BGZ	DCSSUSP	0-U-T, DROP CS & CLR SUSPENSION.
	BLZ	CXBREAK	HANGUP, BREAK CODESTR.EXECUTION.
	AI, R11	-1	NORMAL, DROP END-OF-INPUT CHAR.
	LI, R2	BA(IMAGE)	PT AT IMAGE BUFFER
QDB	STW, R11	RSSIZE	SAVE LENGTH OF INPUT.
	LI, R4	TYPETEXT	
	STW, R4	RSTYPE	SET FOR TEXT TYPE OF DATA BLK.
	LI, R4	1	ASSUME VECTOR RANK.
	CW, R4	RSSIZE	BUT IF LENGTH IS 1, USE SCALAR RANK.
	BNE	QDBRANK	
	LI, R4	0	
QDBRANK	STW, R4	RSRANK	
	STW, R1	TOPOSTAK	SAVE STACK PTR. IN CASE WS FULL.
	STW, R3	OPER+1	SAVE OFFSET IN CASE WS FULL OCCURS.
	BAL, R14	ALOCTRES	ALLOC TEXT RESULT DATA BLK.
	LW, R11	RSSIZE	
	STW, R11	2, R4	SET LENGTH WORD IN CASE OF VECTOR.
	BEZ	QDBDONE	0, EMPTY VECTOR.
	AI, R4	3	PT AT 1ST VALUE WD FOR A VECTOR.
	SLS, R4	2	USE BYTE LOC.
	CI, R11	1	IS IT A VECTOR...
	BNE	QDBFILL	YES.
	AI, R4	-4	NO, BACK UP 1 WD.
QDBFILL	LB, R5	0, R2	
	STB, R5	0, R4	
	AI, R2	1	MOVE TEXT CHARS INTO THE DATA BLK.
	AI, R4	1	
	BDR, R11	QDBFILL	
QDBDONE	LI, R4	0	MOVE RESULT INTO PTRITEM.
	XW, R4	RESULT	
	STW, R4	PTRITEM	
	B	QRESCAN	
A25Q	AI, R6	QUADCODE-QZCODE	CHECK FOR QUAD-ZERO...
	BNEZ	A25BLIND	BLIND INPUT.
	STW, R3	OPER+1	QUAD-ZERO, SAVE CS OFFSET TO IT.
	LW, R5	BREAKFLG	TEST FOR BREAK...
	BEZ	QZIN	NO -- DO GRAFIX INPUT.
	B	CXBREAK	YES.
A25BLIND	RES	0	R6 HAS THE QUAD NUMBER.
	BAL, R14	INPBLIND	@ DO BLIND INPUT.
	B	CXERIO	@ ERROR RETURN -- I/O ERROR.
	LW, R1	TOPOSTAK	@ NORMAL RETURN, RESTORE STACK PTR.
	LI, R2	BA(BLINBUF)	PT AT BLIND INPUT BUFFER.
	B	QDB	

A25EVAL	LI, R13	0	(NO E-FLAG)	
	LI, R14	INPEVAL	SET TO REQUEST EVALUATED INPUT.	
STACKQE	BAL, R4	STACKUP	ADD A WD TO EXECUTION STACK.	
	LI, R6	0		
	XW, R6	CURRCS	MOVE CURRENT CODESTRING DATA BLK PTR	
	STW, R6	0, R1		
	SLS, R3	17	AND CURRENT BYTE POSITION	
	STS, R3	0, R1	INTO THE STACK.	
	AI, R1	-1	PT AT WD TO RECEIVE Q-ENTRY.	
	STW, R1	TOPOSTAK	SAVE STACK PTR.	
	XW, R1	STATEPTR	UPDATE STATE PTR.	
	SW, R1	TOPOSTAK	CALC OFFSET TO NEXT STATE ENTRY.	
	OR, R1	QENTRY	SET Q CATEGORY, PENDENT STATE.	
	STW, R1	*TOPOSTAK		
	STS, R13	*TOPOSTAK	SET E-FLAG VALUE.	
	B	*R14	DO EVAL-INPUT OR EXECUTE-TRANSLAT'N.	
SORXV	B	CKSCANOP	SCAN OPERATOR (ON 1ST COORD).	U19-0010
	B	CKSCANOP	SCAN OPERATOR.	U19-0011
*	B	XEQCK	EXECUTE OPERATOR.	U19-0012
XEQCK	LW, R4	RTARG		U19-0013
	LI, R5	1	SET FOR 1 ELEM (& FOR RANK TEST).	
	LI, R6	NEWLINE	GET END-OF-INPUT CHAR.	
	LB, R13	*R4, R5	GET RANK...	
	BEZ	XEQCKS	SCALAR	
	AI, R13	-1	ARRAY, VERIFY VECTOR...	
	BNEZ	ERRANK	OOPS -- RANK ERROR.	
	LW, R5	2, R4	GET NO.OF ELEMS...	
	BEZ	XEQCKL	NONE, JUST USE END-OF-INPUT CHAR.	
	CI, R5	512	VERIFY NOT TOO MANY...	
	BG	ERLENGTH	OOPS -- LENGTH ERROR.	
	AI, R4	3	PT AT 1ST VALUE WD.	
XEQCKTYP	LB, R13	*RTARG		
	AI, R13	-TYPETEXT	VERIFY TEXT VECTOR OR SCALAR...	
	BEZ	XEQCKBUF	OK, START MOVING CHARS (RT TO LF).	
	B	ERDOMAIN	OOPS -- DOMAIN ERROR.	
XEQCKS	AI, R4	2	PT AT VALUE WD.	
	B	XEQCKTYP		
XEQCKC	LB, R6	*R4, R5	GET CHAR.	
XEQCKBUF	STB, R6	INBUF, R5	PUT CHAR IN INPUT BUFFER.	
	BDR, R5	XEQCKC	LOOP TILL REACHING LEFTMOST CHAR.	
	LB, R6	*R4	PUT IT IN.	
XEQCKL	STB, R6	INBUF		
	LI, R4	0		
	XW, R4	RTARG	DEREFERENCE RT ARG.	
	BAL, R7	DREF		
	LI, R4	0		
	XW, R4	XTEMP	TEST FOR COORD TO LEFT OF XEQ-OPER.	
	BEZ	XEQCKR	NO, LEAVE NIL ENTRY IN STACK.	
	STW, R4	0, R1	YES, PUT X-ENTRY IN STACK.	
XEQCKR	LW, R3	OFFSETR	PT TO DESIGNATOR TO BE RE-SCANNED	
*			WHEN EXECUTE IS COMPLETED.	
	LI, R13	EFLAG	SET E-FLAG.	
	LI, R14	INPXEQ	SET TO ENTER 'INPUT' TRANSLATOR FOR	
	B	STACKQE	EXECUTE & STACK Q-ENTRY.	
REQUIN	STW, R1	TOPOSTAK	SAVE STACK PTR.	
	B	INPEVAL	DO EVALUATED INPUT AGAIN.	
A6	LB, R6	*CURRCS, R3	LOOK AT ITEM BEFORE BRANCH ARROW.	
	AI, R6	-BOSCODE	SHOULD BE BEGINNING-OF-STMT CODE.	
	BNEZ	CXERSYN	OOPS -- SYNTAX ERROR.	
	AI, R3	1	OK, RESTORE PTR IN CASE OF ERROR.	
	LW, R6	NILCK	CK FOR BRANCH VALUE...	
	BEZ	DCSSUSP	NONE, DROP CS & SUSPENSION-CLEAR	
	B	CXERSYN	OOPS -- SYNTAX ERROR	
*			YOU CAN'T BRANCH FROM EVAL-INPUT	
A9	LI, R4	0	ELIMINATE THE REFERENCE TO THE	
	XW, R4	CURRCS	CODESTRING DATA BLK THAT BROUGHT	
	AI, R4	-2	THE EVALUATED INPUT.	
	BAL, R7	DREF		
	LI, R4	0		
	XW, R4	NILCK	MOVE VALUE ACQUIRED, IF ANY...	
	BNEZ	A9VAL	VALUE IS PRESENT.	
QETEST	LI, R4	EFLAG	NO VALUE, CK Q-ENTRY...	
	CW, R4	0, R1		
	BAZ	REQUIN	EVAL-INPUT, RE-REQUEST IT.	
	STW, R1	TOPOSTAK	SAVE STACK PTR IN CASE WS FULL.	
XEQEMPTY	LW, R1	TOPOSTAK		
	LI, R11	4		
	BAL, R7	ALOCBLK	ALLOCATE FOR EMPTY VECTOR.	

	LI, R11	(TYPEINTG**8)+1	TYPE=INTG & RANK=1.
	STH, R11	*R4	
	LI, R11	0	LENGTH=0.
A9VAL	STW, R11	2, R4	
	STW, R4	PTRITEM	SET PTR TO VALUE DATA BLK.
	LI, R4	X'7FFF'	SET STATE PTR TO AIM AT NEXT STATE-ENTRY IN EXECUTION STACK.
	AND, R4	0, R1	
	AWM, R4	STATEPTR	
	LI, R3	X'1FFFF'	RESTORE CURRENT CODESTRING DATA BLK
	AND, R3	1, R1	PTR THAT HAD THE EVAL-INPUT QUAD.
	STW, R3	CURRCS	
	LW, R3	1, R1	RESTORE THAT BYTE OFFSET.
	SLS, R3	-17	
	AI, R1	2	DROP THE Q-ENTRY FROM EXEC. STACK.
	LI, R11	EFLAG	WAS IT EVAL-INPUT OR EXECUTE...
	CW, R11	-2, R1	
	BAZ	QRESCAN	EVAL-INPUT.
	BAL, R14	QOBS	(QUERY OBSERVE SETTING)
	LI, R4	0	EXECUTE.
	XW, R4	PTRITEM	MOVE VALUE PTR OUT OF PTRITEM, AND
	XW, R4	0, R1	REPLACE X OR NIL ENTRY...
	BEZ	A9XEQEND	NIL.
	AI, R1	-1	X-ENTRY (COORD FOR OPER UP NEXT)
	STW, R4	0, R1	PUSH IT BACK INTO STACK.
A9XEQEND	LB, R2	*R1	GET TOP CATEGORY (X OR V).
	B	NXTSCAN	RESCAN WHERE EXECUTE ORIGINATED.
QRESCAN	LI, R4	0	SET 'PTR' TYPE OF SCAN ITEM.
	LB, R2	*R1	GET NEW TOP CAT SURFACED.
	B	CXPTR, R2	VECTOR ON THAT CATEGORY.
A16	STD, R6	OPER	SAVE OPERATOR INFORMATION.
	LI, R14	NXTSCAN	RETURN TO RESUME SCANNING AFTER
	BDR, R3	MONEX	MONADIC OPERATOR EXECUTION.
A37	LD, R10	STMASKS-STOPNMCD-STOPNMCD, R6	SET BIT & MASK FOR
*			STOP OR TRACE.
	LI, R2	CATAPRYM	CONVERT TOP CAT TO A-PRIME.
	STB, R2	*R1	
	LW, R4	1, R1	GET V-ENTRY, I.E. DATA BLK PTR.
	LI, R7	1	
	LB, R7	*R4, R7	IS ITS RANK = 2...
	AI, R7	-2	
	BNEZ	A37ST	NO, CK STOP OR TRACE VECTOR.
	AI, R10	-STOPBIT	YES, CK SIDETRACK TBL.
	BNEZ	CXERRANK	OOPS -- RANK ERR (TRACE TBL).
	LW, R11	3, R4	GET 2ND LENGTH WD...
	BEZ	A37ECLR	ZERO, CLEAR OUT ERR-CTRL TABLE.
	AI, R11	-2	
	BNEZ	A37EQ1Z	NOT 0 OR 2, 1ST LENGTH WD MUST = 0
	LW, R11	2, R4	TWO, GET 1ST LENGTH WD...
	BEZ	A37ECLR	ZERO, CLEAR OUT ERR-CTRL TABLE.
	LB, R6	*R4	NZ, GET TYPE OF DATA BLK.
	CI, R6	TYPEINTG	
	BE	A37EINTG	INTEGER TYPE.
	STW, R5	FDEFPTR	NON-INTG TYPE, SAVE PTR TO &
	MTW, 1	1, R5	BUMP REF-CNT OF FUNCT.DEScriptor.
	SLS, R11	1	= # WDS FOR INTG VERSION OF TABLE.
	AI, R11	4	+ 2-WD HDR 0 2 LENGTH WDS.
	BAL, R7	ALOCBLK	ALLOC DATA BLK FOR INTG TYPE TABLE.
	STW, R4	RESULT	SAVE PTR TO IT.
	LI, R7	(TYPEINTG**8)+2	
	STH, R7	*RESULT	SET TYPE = INTEGER & RANK = 2.
	LW, R10	1, R1	GET PTR TO NON-INTG TABLE AGAIN.
	LI, R11	2	
	STW, R11	3, R4	SET 2ND LENGTH WD = 2.
	AI, R10	2	PT AT NON-INTG TABLE'S 1ST LENGTH WD
	LW, R11	*R10	AND GET THAT LENGTH IN ORDER TO
	STW, R11	2, R4	SET 1ST LENGTH WD OF INTG TBL.
	SLS, R11	1	GET # ELEMENTS FOR ERR-CTRL TBL.
	AI, R4	4	PT AT 1ST INTG ELEMENT POSITION.
	AI, R6	-TYPEREAL	TEST TYPE OF NON-INTG TBL...
	BEZ	A37EREAL	REAL.
	AI, R6	TYPEREAL-TYPEOGL	
	BNEZ	CXERDOM	ROTTEN -- DOMAIN ERR.
A37ELWD	AI, R10	1	LOGL, PT AT LAST LENGTH WD.
	LI, R13	32	32 LOGL BITS PER WD.
	AI, R10	1	
	LW, R7	*R10	GET NEXT WD OF LOGL DATA.
A37ELBIT	LI, R6	0	
	SLD, R6	1	GET NEXT LOGL VALUE -- 1 OR 0.

	STW, R6	0, R4	SET INTEGER VALUE.
	AI, R4	1	PT AT NEXT WD OF INTG DATA BLK.
	AI, R11	-1	IS INTG ERR-CTRL TBL READY YET...
	BEZ	A37ERDY	YES.
	BDR, R13	A37ELBIT	NO, LOOP TILL LOGL WD EXHAUSTED
	B	A37ELWD	THEN GET ANOTHER. BULL-POOP.
A37EREAL	AI, R10	2	PT AT NEXT REAL VALUE.
	LD, R6	*R10	GEDDIT.
	BAL, R5	F2I	@ CONVERT FL.PT VALUE TO INTEGER...
	B	CXERDOM	@ WON'T -- DOMAIN ERR.
	STW, R7	0, R4	@ OK, SET INTEGER VALUE.
	AI, R4	1	PT AT NEXT WD OF INTG DATA BLK.
A37ERDY	BDR, R11	A37EREAL	LOOP TILL INTG ERR-CTRL TBL READY.
	LI, R5	0	REMOVE REF TO FUNCTION DESCRIPTOR &
	XW, R5	FDEFPTR	SET R5 TO PT AT IT AGAIN.
	MTW, -1	1, R5	
	LI, R4	0	MOVE PTR TO INTG ERR-CTRL TABLE
	XW, R4	RESULT	INTO R4.
	B	A37ESET	
A37EINTG	MTW, 1	1, R4	BUMP REF-CNT OF ERR-CTRL TBL.
A37ESET	XW, R4	XSIZEOFF, R5	INSTALL PTR TO ERR-CTRL IN XSIZE WD.
	LI, R7	X'E0000'	DID THE ERR-CTRL FUNCTION HAVE AN
	AND, R7	R4	OLD ERR-CTRL TABLE...
	BEZ	A37ENEW	NO.
A37EOLD	STS, R7	XSIZEOFF, R5	YES, SET XSIZE VALUE BACK IN.
	LI, R7	NXTSCANU	RESUME CODESTRING SCAN AFTER
	B	DREF	DEREFERENCING OLD ERR-CTRL TBL.
A37ENEW	LW, R7	R4	GET XSIZE VALUE.
	SLS, R7	17	MOVE IT UP ABOVE ADDR.FIELD.
	STS, R7	XSIZEOFF, R5	SET XSIZE VALUE BACK IN THERE.
	BDR, R3	NXTSCAN	RESUME CODESTRING SCAN.
A37EQ1Z	LW, R11	2, R4	IS 1ST LENGTH WD ZERO...
	BNEZ	CXERLEN	NO, 2ND LENGTH WD BAD -- LENGTHERR
A37ECLR	LW, R4	XSIZEOFF, R5	GET OLD XSIZE WD FOR FUN.DESRIPTOR.
	CI, R4	X'E0000'	CK FOR PTR TO ERR-CTRL TBL...
	BAZ	NXTSCANU	NO (ALREADY CLEARED) RESUME SCAN.
	SLS, R4	-17	YES, GET XSIZE VALUE ONLY.
	XW, R4	XSIZEOFF, R5	REPLACE OLD XSIZE WD &
	B	A37EOLD	DEREF OLD ERR-CTRL TBL.
A37ST	BAL, R7	NUMSV	@ CK NUMERIC SCALAR OR VECTOR...
	AI, R4	-1	@ SCALAR, PT AT HEADER + 2.
*			@ VECTOR.
A37CLR	STW, R10	STSET	SAVE STOP OR TRACE FLAG SETTING.
	LI, R10	X'10000'	IS THIS A LOCKED FUNCTION...
	CW, R10	0, R5	
	BANZ	NXTSCANU	YES, RESUME SCAN.
	LI, R10	0	CLEAR FOR SELECTIVE LOADS, BELOW.
	LW, R13	NFLOFF, R5	NO.OF FUN LINE PTRS; R5 = FUNDEF PTR
	BEZ	NXTSCANU	NO LINES, RESUME SCAN.
	STW, R13	NFL4ST	SAVE IT FOR STOP OR TRACE BOUND CK.
	AI, R5	LINSTART-1	PT JUST BEFORE LINE 1 AND
	STW, R5	FDPTR	SAVE THAT LOC.
	AI, R5	1	PT AT LINE 1.
A37CLREM	LS, R10	0, R5	CLEAR OLD STOP OR ELSE TRACE
	STW, R10	0, R5	FLAGS.
	AI, R5	1	
	BDR, R13	A37CLREM	
	AI, R12	0	TEST FOR EMPTY VECTOR.
	BEZ	NXTSCANU	YES, RESUME SCAN.
	LW, R10	STSET	NO, GET STOP OR ELSE TRACE FLAG.
A37VT	B	A37VT, R6	@ VECTOR ON NUMERIC TYPE:
	B	A37LOGL	@ 1 -- LOGL
	B	A37INTG	@ 2 -- INTG
	B	A37REAL	@ 3 -- REAL
	B	A37XSEQ	@ 4 -- INDEX SEQUENCE
A37XSEQ	LI, R7	A37X	@ SET RETURN TO A37X, SEE ALSO A37DECR
	STW, R7	RSTYPE	
	LW, R6	0, R4	GET INDEX PARAMETERS
	LW, R7	1, R4	AND
	STD, R6	XPARAMS	SAVE THEM.
	LI, R4	1	START INDEX SEQUENCE
A37X	LW, R7	R4	
	MW, R7	XMULT	CALCULATE CORRESPONDING VALUE.
	AW, R7	XADD	
	B	A37Z	CK VALUE AS A LINE NO.
A37REAL	LI, R7	A37R	SET RETURN TO A37R, SEE ALSO A37DECR
	STW, R7	RSTYPE	
	AI, R4	1	PT AT 1ST DBLWD (ODD OR EVEN IS OK).

A37R	SLS, R4	-1	= DBLWD LOC OF 1ST VALUE.
	LD, R6	0, R4	GET REAL VALUE FOR CONVERSION.
	BAL, R5	F2I	CONVERT FLOATING TO INTEGER...
	B	CXERDOM	NO -- DOMAIN ERROR.
	B	A37Z	YES, CHECK LINE NO.
A37INTG	LI, R7	A37I	SET RETURN TO A37I, SEE ALSO A37DECR
	STW, R7	RSTYPE	
A37I	LW, R7	0, R4	GET INTEGER VALUE.
A37Z	BLEZ	A37DECR	NEG OR ZERO IS IGNORED, AND SO ARE
	CW, R7	NFL4ST	
	BG	A37DECR	VALUES ABOVE LAST LINE NO.
	AWM, R10	*FDLPTR, R7	OK, SET THAT FLAG BIT.
A37DECR	AI, R4	1	PT AT NEXT VALUE.
	BDR, R12	*RSTYPE	LOOP BACK ACCORDING TO TYPE OF DATA.
	BDR, R3	NXTSCAN	RESUME SCAN.
A37LOGL	AWM, R6	FDLPTR	PTS AT LINE 1'S ENTRY NOW.
A37WD	LI, R5	32	32 BITS PER WORD.
	LW, R13	0, R4	GET LOGICAL VALUE(S).
	BGEZ	A37LDECR	1ST BIT NOT A 1.
A37L1	AWM, R10	*FDLPTR	SET STOP OR TRACE FLAG FOR LINE 1.
	BDR, R3	NXTSCAN	RESUME SCAN.
A37SHIFT	SLS, R13	1	TEST NEXT BIT...
	BOV	A37L1	1, LINE 1 IS TO BE FLAGGED.
A37LDECR	AI, R12	-1	DECR ELEMENT COUNT...
	BEZ	NXTSCANU	0, RESUME SCAN; NO TRACE OR STOP.
	BDR, R5	A37SHIFT	DECR BIT COUNT...
	AI, R4	1	0, USE NEXT DATA WORD.
	B	A37WD	
A8	LI, R7	A8SHOW	RETURN FROM 'STMTV' TO 'A8SHOW'
STMTV	LI, R5	0	MOVE NILCK INTO R5.
	XW, R5	NILCK	
	BEZ	0, R7	EXIT IF NIL.
	LW, R12	ACFLAG	NON-NIL (V-ENTRY), IF AN ASSIGNMENT
	BEZ	STMTADD	WASN'T JUST COMPLETED PUT V IN
*			THE S-ENTRY.
STMTTRCK	LW, R12	OBSERVE	DID WE OBSERVE...
	BNEZ	STMTDREF	YES, SKIP TRACE CHECK.
	LI, R12	TRACEBIT	IS THIS STMT BEING TRACED...
	CW, R12	0, R1	
	BANZ	STMTADD	YES, PUT V-ENTRY IN THE S-ENTRY.
STMTDREF	LW, R4	R5	NO, PT AT V-ENTRY'S DATA BLK.
	B	DREF	DEREFERENCE IT (DREF EXITS 'STMTV')
*			SINCE DREF ALSO USES R7 AS LINK).
STMTADD	XW, R5	0, R1	POP S-ENTRY AND PUSH V-ENTRY.
	AI, R5	1	INCR S-ENTRY'S COUNT.
	BAL, R4	STACKUP	ALLOC 1 MORE WD FOR EXEC. STACK.
	STW, R5	0, R1	PUSH NEW S-ENTRY CATEGORY WD.
	B	0, R7	EXIT.
A8SHOW	LI, R6	A8WRAPCS	RETURN FROM 'STMTSHOW' TO 'A8WRAPCS'
STMTSHOW	LI, R13	X'FFFF'	TEST S-ENTRY'S COUNT.
	AND, R13	0, R1	
	BEZ	NOSHOW	ZERO, NO STMT DISPLAY, NO DEREF-ING.
	LW, R4	OBSERVE	DID WE OBSERVE THIS LINE...
	BEZ	STMTSHOM	NO.
	CI, R13	1	YES, SINGLE VALUE...
	BE	DONTSHOW	YEP, ITS BEEN SHOWN ALREADY.
STMTSHOM	STW, R1	TOPOSTAK	SAVE STACK PTR.
	LI, R3	0	PRESET COLUMN INDICATOR.
	LI, R12	TRACEBIT	IS THIS A TRACED LINE...
	CW, R12	0, R1	
	BAZ	SHOWEM	NO.
	AI, R4	0	YES, BUT WAS IT OBSERVED...
	BNEZ	SHOWEM	YEP, JUST SHOW MIXED OUTPUT.
	STW, R6	LINKT	SAVE LINK.
	LW, R1	STATEPTR	YES, PT AT ITS F-ENTRY.
	BAL, R14	FUNLDISP	DISPLAY FUNC.NAME & LINE NUMBER.
	LW, R6	LINKT	RESTORE LINK.
	LW, R1	TOPOSTAK	PT AT THE S-ENTRY AND
	INT, R13	0, R1	GET ITS COUNT AGAIN.
SHOWEM	BAL, R14	MIXEDOUT	@ STMT DISPLAY (MIXED OUTPUT).
	B	CXERIO	@ ERR RETURN -- I/O ERROR.
	LW, R1	TOPOSTAK	@ OK RETURN, RESTORE STACK PTR.
DONTSHOW	LW, R12	0, R1	SAVE S-ENTRY CATEGORY WORD.
	AI, R1	1	PT AT 1ST V-ENTRY UNDER S-ENTRY.
SENTDREF	LI, R4	0	MOVE V-ENTRY'S
	XW, R4	0, R1	DATA BLK PTR INTO R4.
	BAL, R7	DREF	DEREFERENCE.
	AI, R1	1	PT AT NEXT WD IN EXECUTION STACK.

	BDR, R13	SENTDREF	
	B	0, R6	EXIT, R1 PTS AT F-ENTRY.
NOSHOW	LW, R12	0, R1	SAVE S-ENTRY.
	AI, R1	1	PT AT F-ENTRY.
	B	0, R6	EXIT.
A8WRAPCS	LI, R6	SCANBOS	SCAN ITEM WAS BEGINNING-OF-STMT.
WRAPCS	LI, R4	0	MOVE CODESTRING DATA BLK PTR
	XW, R4	CURRCS	INTO R4.
	AI, R4	-2	PT AT ITS HEADER.
	BAL, R7	DREF	DEREFERENCE.
	AI, R6	-SCANBOS	WHICH SCAN ITEM...
	BEZ	NONBRN	BEGINNING OF STMT.
	LI, R7	X'FFFF'	BRANCH ARROW.
	AND, R7	0, R1	IS F-ENTRY THE FINAL ENTRY...
	BNEZ	LOBRNCK	NO, FUNCTION-STATE.
CXTHRU	STW, R1	TOPOSTAK	SAVE STACK PTR.
	B	INPDIR	GO TO DIRECT INPUT HANDLER.
LOBRNCK	LW, R7	BRNVAL	GET BRANCH VALUE--
	BGZ	GFL	ABOVE ZERO, TRY TO GET FUNC LINE.
	B	FEXIT	TOO LOW, EXIT THE FUNCTION.
NONBRN	CI, R12	FLINFLG	WAS STMT A FUNC LINE...
	BANZ	NXTFLIN	YES, TRY TO GET NEXT FUNC LINE.
	B	CXTHRU	NO.
NXTFLIN	LW, R7	1, R1	GET CURRENT LINE NO.
	SLS, R7	-17	
	AI, R7	1	INCR IT.
GFL	LI, R6	0	MOVE CURRLINO AND FDEFPTR INTO R6...
	XW, R6	1, R1	
	BNEZ	GF	NZ, WE'VE GOT A GOOD F-ENTRY.
	B	SUSPCLR	0, NO GOOD (I.E. SI DAMAGE).
GF	CW, R7	NFLOFF, R6	DOES LINE NO. EXCEED HIGHEST ONE...
	BLE	FLINE	NO, WE'VE GOT A FUNC. LINE.
	B	FXRES	YES, START EXIT FROM FUNCTION.
FEXIT	LI, R6	0	MOVE CURRLINO AND FDEFPTR INTO R6...
	XW, R6	1, R1	
	BEZ	SUSPCLR	0 IF F-ENTRY HAS HAD SI DAMAGE.
FXRES	LW, R5	RESOFF, R6	GET NAME PTR FOR RESULT.
	BLZ	FXCALL	(NONE, LEAVE PTRITEM = ZERO.)
	LI, R3	X'1FFFF'	
	AND, R3	*SYMT, R5	EXTRACT REFERENT (DBPTR OR NIL).
	STW, R3	PTRITEM	
	BEZ	FXCALL	(NIL)
	MTW, 1	1, R3	IF DBPTR, INCR DATA BLK REF COUNT.
FXCALL	LI, R3	X'1FFFF'	EXTRACT CALLPTR FROM F-ENTRY AND
	AND, R3	2, R1	RESTORE CURRCS TO PT AT THE OLD
	STW, R3	CURRCS	CODESTRING DATA BLK (+2).
	LI, R3	0	CLEARING EXTRA REF, GET CALLBYTE--
	XW, R3	2, R1	RESTORING R3 TO THE BYTE OFFSET
	SLS, R3	-17	EXISTING WHEN THE CALL OCCURRED.
	LI, R14	X'7FFF'	EXTR OFFSET TO NEXT STATE-ENTRY
	AND, R14	0, R1	AND SET STATE POINTER TO PT AT
	AWM, R14	STATEPTR	THAT ENTRY AGAIN.
	LW, R14	OBSFLAG	IS OBSERVE-CMD IN EFFECT...
	BEZ	FXDEREFF	NO.
	LW, R4	STATEPTR	YES, PT AT CALLING STATE ENTRY.
	LI, R2	CATF	
FXFINDF	CB, R2	*R4	LOOK FOR AN F-STATE...
	BE	FXF	OK.
	LI, R5	X'7FFF'	NOPE, TRY NEXT.
	AND, R5	0, R4	
	AW, R4	R5	
	B	FXFINDF	
FXF	LI, R5	PENDFLAG	PENDENT OR SUSPENDED...
	CW, R5	0, R4	
	BAZ	FXOBSERV	SUSPENDED -- OBSERVE DIRECT LINE.
	LI, R5	TRACEBIT	PENDENT.
	CW, R5	-1, R4	WAS THIS A TRACED LINE...
	BANZ	FXOBSERV	YES -- OBSERVE TRACED FUN.LINE.
	LI, R14	0	NO, DON'T OBSERVE THIS LINE.
FXOBSERV	STW, R14	OBSERVE	OBSERVATION SETTING.
FXDEREFF	MTW, -1	1, R6	DECR REF-COUNT OF FUNDEF DATA BLK.
	AI, R1	4	POP DOWN TO WD FOLLOWING THE COUNT--
	LW, R14	-1, R1	NO.OF SHADOW PAIRS.
	BEZ	FPOPPED	NONE, F-ENTRY IS POPPED.
	LI, R7	FUND	RETURN FROM 'DREF' TO 'FUND'.
FUNSHAD	LW, R5	0, R1	GET SHADOWED NAME PTR.
	LI, R4	0	
	XW, R4	1, R1	GET SHADOWED REFERENT INDICATOR.

	AI, R1	2	POP THAT SHADOW PAIR.
	XW, R4	*SYMT, R5	UNSHADOW.
	AND, R4	X1FFFF	EXTR. DBPTR (IF ANY) FOR LOCAL.
	BNEZ	DREF	DEREFERENCE DBPTR (REMOVES LOCAL).
FUND	BDR, R14	FUNSHAD	
FPOPPED	LI, R4	0	CLAIM CURRENT SCAN ITEM = PTR
	LB, R2	*R1	SET TOP CAT FOR ENTRY SURFACED.
	LW, R14	PTRITEM	TEST FOR FUNCTION RESULT...
	BNEZ	CXPTR, R2	YES, DO CONTEXT ANALYSIS FOR IT.
	B	NORESULT, R2	NO, VECTOR ON THE TOP CAT.
LOOKY	LB, R6	*CURRCS, R3	CK ITEM TO LEFT OF FUNCTION CALL...
	CI, R6	BOSCODE	
	BE	CXBOS	BEGINNING OF STMT, OK.
	CI, R6	SEMICODE	
	BE	CXSEMI	SEMICOLON, OK (EMPTY SUBSTMT VALUE).
	CI, R6	DUMMYCD	CHECK FOR DUMMY CODE
	BNE	CXERNOR	NO RESULT-ERROR
	BDR, R3	LOOKY	SKIP DUMMY AND KEEP LOOKING
NORESULT	B	CXERNOR	@ V
	B	NXTSCANU	@ A-PRIME (JUST RETURNED FROM A CATCH)
	B	CXERNOR	@ 0
	B	CXERNOR	@ X
	B	CXERNOR	@ B
	B	CXERNOR	@ P
	BDR, R3	LOOKY	@ S -- LOOK AHEAD FOR BOS OR SEMI.
	BDR, R3	LOOKY	@ Q -- LOOK AHEAD FOR BOS (SEMI FAILS)
	B	CXERNOR	@ D
	B	CXERNOR	@ A (CAN'T GET HERE FROM THERE)
*			
*			
*	NOTE--	R12 CONTAINS A FLAG-BIT (FLINFLG) = 1 IF FLINE IS REACHED VIA	
*		A STMT WITHIN THE FUNCTION OR VIA INVOKING THE FUNCTION.	
*		IT IS 0 IF FLINE IS REACHED VIA A DIRECT BRANCH STMT.	
*			
FLINE	SLS, R6	-1	(FDEFPTR IS EVEN, SO THIS IS OK).
	STH, R7	R6	PUT NEW LINE NO. IN PLACE OF CURRENT
	SLS, R6	1	LINE NO. AND PUT THE NEW
	STW, R6	1, R1	'CURRLINO, FDEFPTR' WD IN F-ENTRY.
FLINE1	AW, R7	R6	GET THAT LINE PTR ENTRY FROM THE
	LW, R7	LINSTART-1, R7	FUNCTION DESCRIPTOR.
	CI, R7	STOPBIT	IS ITS STOP FLAG SET...
	BAZ	FGO	NO.
	CI, R12	FLINFLG	YES, OVERRIDE THE STOP...
	BAZ	FGO	YEP, DIRECT BRANCH DOESN'T STOP.
	LI, R5	-PENDFLAG-1	NOPE, GET 1'S COMPLEMENT OF THE
	AND, R5	0, R1	PENDENT FLAG-- I.E. TO MAKE THIS
	STW, R5	0, R1	F-ENTRY A STOPPED FUNCTION.
	STW, R1	TOPOSTAK	SAVE STACK PTR.
	B	SHOWSTOP	DISPLAY THE STOP, R1 PTS AT F-ENTRY.
FGO	LI, R13	PENDFLAG	SET F-ENTRY TO BE PENDENT.
	STS, R13	0, R1	
	LI, R2	0	SET FOR NO OBSERVATION.
	LI, R12	TRACEBIT	EXTRACT TRACE FLAG FROM LINE PTR
	AND, R12	R7	ENTRY...
	BEZ	FLSETOBS	0--NO OBSERVATION OF THIS LINE.
	LW, R2	OBSFLAG	NZ--USE OBSERVE-CMD SETTING.
FLSETOBS	STW, R2	OBSERVE	=0 IF NO OBSERVATION & =-1 IF YES.
	OR, R12	FSENTRY	COMPOSE S-ENTRY FOR FUN.LINE (CNT=0)
	STW, R12	-1, R1	PUT S-ENTRY ABOVE F-ENTRY IN STACK.
	AND, R7	X1FFFF	EXTR. PTR TO CODESTRING DATA BLK +2.
	STW, R7	CURRCS	THAT IS LOC OF CURRENT CODESTR. NOW.
	MTW, 1	-1, R7	INCR REF-COUNT OF CODESTRING DATA BK
	LH, R3	*CURRCS	GET OFFSET TO RTMOST CODE DESIGNATOR
	LI, R2	CATS	TOP CAT IS AN S-ENTRY.
	AI, R1	-1	SET STACK PTR TO AIM AT NEW S-ENTRY
	LW, R6	BREAKFLG	TEST FOR BREAK...
	BEZ	NXTSCAN	NO -- START SCANNING.
	B	CXBREAK	YES -- BREAK OR HANG-UP.
A5	LB, R4	*CURRCS, R3	GET ITEM TO LEFT OF BRANCH ARROW...
	AI, R4	-BOSCODE	
	BEZ	A5NILCK	BEGINNING OF STMT, OK.
	AI, R4	BOSCODE-SEMICODE	
	BEZ	A5NILCK	SEMICOLON, OK.
	B	CXERSYN	ELSE -- SYNTAX ERROR.
A5NILCK	AI, R3	1	BACK TO BRANCH ARROW IN CASE ERROR.
	LW, R4	NILCK	TEST NILCK...
	BNEZ	A5C23	NON-NIL, A5 CASE 2 OR 3.
	BAL, R6	STMTSHOW	IF NEC. DISPLAY STMT VALUES, POP THE
*			S-ENTRY, AND DEREFERENCE VALUES.

DCSSUSP	LI, R4	0	DEREFERENCE CURRENT CODESTRING D.B.
	XW, R4	CURRCS	
	AI, R4	-2	
	LI, R7	SUSPCLR	AND THEN DO SUSPENSION-CLEAR.
	B	DREF	
A5C23	BAL, R7	NUMSV	@ CK NUMERIC SCALAR OR VECTOR...
	BDR, R4	A5C3	@ SCALAR -- DEFINITELY NON-EMPTY.
	AI, R12	0	@ VECTOR -- EMPTY...
	BEZ	A5C2	YES.
A5C3	LI, R5	A5BSET-1	NO, SET LINK TO IVAL SO THAT
*			NON-INTEG EXIT WILL GIVE ERROR
			AND INTEGER EXIT WILL BE TO A5BSET
IVAL	B	IVAL, R6	@ VECTOR ON NUMERIC TYPE:
	B	ILOGL	@ 1 -- LOGL
	B	IINTG	@ 2 -- INTG
	B	IREAL	@ 3 -- REAL
*	B	IXSEQ	@ 4 -- XSEQ (INDEX SEQUENCE)
IXSEQ	LW, R7	0, R4	@ JUST ADD THE 2 PARAMETERS.
	AW, R7	1, R4	
	B	1, R5	
IREAL	AI, R4	1	INTEGER EXIT.
	SLS, R4	-1	PT AT VALUE WD (EVEN OR ODD WD OK).
	LD, R6	0, R4	PT AT VALUE DBLWD.
	B	F2I	GET VALUE FOR FLOAT-TO-INTG CONV.,
*			IF NON-INTEG, F2I EXITS 0, R5
			IF INTEGER (WITHIN FUZZ) EXIT 1, R5
ILOGL	LI, R7	0	ASSUME VALUE IS 0.
	LW, R6	0, R4	GET LOGICAL.
	BGEZ	1, R5	IT IS 0, INTEGER EXIT.
	LI, R7	1	
	B	1, R5	NO, USE 1, INTEGER EXIT.
IINTG	LW, R7	0, R4	GET INTEGER
	B	1, R5	INTEGER EXIT.
	B	CXERDOM	@ NON-INTEG BRANCH -- DOMAIN ERROR.
A5BSET	STW, R7	BRNVAL	@ SAVE BRANCH VALUE.
	LI, R5	0	
	XW, R5	NILCK	MOVE NILCK INTO R5.
	BAL, R7	STMTTRCK	HANDLE TRACE CHECK.
	LI, R6	WRAPCS	RETURN TO CODESTRING WRAP-UP FROM
	B	STMTSHOW	STMT DISPLAY AND DEREFER--
*			NOTE THAT R6 WILL NOT = SCANBOS.
A5C2	LI, R4	0	
	XW, R4	NILCK	MOVE NILCK INTO R4.
	LI, R7	NXTSCAN	RESUME SCAN AFTER
	BDR, R3	DREF	DEREFING THE EMPTY VECTOR.
A31	B	A31VORP, R4	VECTOR ON SCANITEM VN OR PTR:
A31VORP	B	A31NILCK	@ 0 -- PTR.
*	B	A31VN	@ 1 -- VN.
A31VN	AND, R5	X1FFFF	@ EXTR. DATA BLK PTR FROM REF-IND. WD.
	STW, R5	PTRITEM	
	MTW, 1	1, R5	INCR REF COUNT OF THAT DATA BLOCK.
A31NILCK	BAL, R14	QNILCK	QUERY SUBSCRIPTING.
DYFUN	LW, R5	0, R1	PULL D-ENTRY. USE A30 TO DROP IT
*			FROM THE EXECUTION STACK.
	MTW, -1	1, R5	DEREFERENCE IT TEMPORARILY.
A30	AI, R1	1	DROP A-PRIME (OR D) ENTRY.
A29	LW, R7	0, R1	PULL V-ENTRY.
	STW, R7	RTARG	SAVE AS RIGHT ARGUMENT.
	AI, R1	1	DROP V-ENTRY FROM EXECUTION STACK.
A26	AND, R5	X1FFFF	EXTR. FUNDEF PTR FROM REF-IND. WD
*			OR FROM PULLED D-ENTRY.
	MTW, 1	1, R5	INCR REF COUNT OF FUNDEF DATA BLK.
	STW, R5	FDEFPTR	
	LW, R8	XSIZEOFF, R5	GET XSIZE WD.
	CI, R8	X'E0000'	DOES IT CONTAIN AN ERR-CTRL TBL PTR
	BAZ	FXSIZE	NO.
	SLS, R8	-17	YES, SHIFT XSIZE DOWN.
FXSIZE	STW, R8	XSIZE	SAVE EXEC.STACK SIZE FOR FUN.CALL.
	SW, R1	XSIZE	ALLOC EXEC.STACK SPACE FOR F-ENTRY.
	CW, R1	STKLIMIT	DOES IT FIT...
	BG	FBUILD	YES, BUILD F-ENTRY.
	STW, R1	LOCNEED	SAVE LOC NEEDED.
	AW, R1	XSIZE	RESTORE STACK PTR IN CASE WS FULL.
	BAL, R8	CTEST	GET ENUF MORE COMMON, IF POSSIBLE.
	B	CXERWS	WS FULL.
FBUILD	LW, R5	FDEFPTR	PT AT FUNCTION DESCRIPTOR DATA BLK.
	AW, R1	XSIZE	RESTORE STACK POINTER (BEFORE THE
	STW, R1	TOPOSTAK	F-ENTRY) AND SAVE THAT LOC.
	LW, R7	RESOFF, R5	GET RESULT NAME POINTER...

	BLZ	FLF	NO RESULT.
	LI, R4	0	RESULT, USE NO REFERENT.
FLF	BAL, R6	SHADOW	
	LW, R7	LFOFF, R5	GET LF DUMMY NAME POINTER...
	BLZ	FRT	NO LF DUMMY.
	LI, R4	0	LF DUMMY, USE AS REFERENT
	XW, R4	LFARG	THE LEFT ARGUMENT.
	BAL, R6	SHADOW	
FRT	LW, R7	RTOFF, R5	GET RT DUMMY NAME POINTER...
	BLZ	FNFL	NO RT DUMMY.
	LI, R4	0	RT DUMMY, USE AS REFERENT
	XW, R4	RTARG	THE RIGHT ARGUMENT.
	BAL, R6	SHADOW	
FNFL	AW, R5	NFLOFF, R5	PT AT THE FUNDEF WORD THAT FOLLOWS
	AI, R5	LINSTART+1	THE 'LOCALS AND LABELS' WORD.
	LW, R8	-1, R5	GET LOCALS AND LABELS COUNTS.
	BEZ	FNSHAD	NO LOCALS OR LABELS.
	LI, R9	X'FFFF'	
	AND, R9	-1, R5	GET NO.OF LABELS.
	BEZ	FLCLS	NONE, TRY LOCALS.
	LI, R6	FLBL	RETURN FROM 'SHADOW' TO 'FLBL'.
FLBL	AI, R5	2	PT AT WD AFTER SHADOW PAIR.
	LW, R7	-2, R5	GET NAME POINTER
	LW, R4	-1, R5	GET REFERENT INDICATOR FOR LABEL.
	MTW, 1	1, R4	INCR REF COUNT OF LABEL'S DATA BLK.
	BDR, R9	SHADOW	
	BAL, R6	SHADOW	SHADOW LAST LABEL
FLCLS	LH, R9	R8	GET NO.OF LOCALS.
	BEZ	FNSHAD	NO LOCALS.
	LI, R6	FLCL	RETURN FROM 'SHADOW' TO 'FLCL'.
	AI, R9	1	FAKE AN EXTRA LOCAL FOR LOOPING.
	SLS, R5	1	USE HALFWORD RESOLUTION.
FLCL	LI, R4	0	NO REFERENT FOR LOCAL SHADOWING.
	LH, R7	0, R5	GET LOCAL NAME PTR (EXCEPT LAST TRY)
	AI, R5	1	PT AT NEXT HALFWORD.
	BDR, R9	SHADOW	
FNSHAD	LW, R13	TOPOSTAK	CALC. NO.OF SHADOW WORDS IN STACK.
	SW, R13	R1	
	SLS, R13	-1	USE NO.OF SHADOW PAIRS.
	STW, R13	-1, R1	PUT IT IN STACK.
	SLS, R3	17	MOVE CALLBYTE TO PROPER FIELD.
	STS, R3	CURRCS	OR IT INTO CALLPTR WORD.
	LI, R13	0	
	XW, R13	CURRCS	MOVE CALLBYTE--CALLPTR
	STW, R13	-2, R1	INTO THE STACK.
	LI, R6	0	
	XW, R6	FDEFPTR	MOVE FDEFPTR
	AI, R6	X'20000'	AND CURRLINO (=1)
	STW, R6	-3, R1	INTO THE STACK.
	AI, R1	-4	PT AT TOP OF F-ENTRY.
	STW, R1	TOPOSTAK	SAVE IT.
	LW, R13	STATEPTR	GET OLD STATE PTR.
	SW, R13	TOPOSTAK	CALC. OFFSET FROM F-ENTRY TO NEXT
*			STATE-ENTRY.
	OR, R13	FENTRY	FUNCTION-STATE CATEGORY, PENDING.
	STW, R13	0, R1	SET F-ENTRY.
	STW, R1	STATEPTR	UPDATE STATE PTR TO NEW F-ENTRY.
	LI, R12	FLINFLG	FAKE S-ENTRY SO STOP ON 1 WILL WORK.
	LI, R7	1	SET LINE NO. = 1.
	LW, R13	NFLOFF, R6	GET NO.OF FUNC LINES...
	BGZ	FLINE1	OK, GET LINE 1.
	STW, R13	PTRITEM	ZERO, SET PTRITEM NIL.
	B	FXCALL	EXIT THIS FUNCTION CALL.
A14	INT, R11	0, R1	EXTR COUNT FROM THE B-ENTRY.
	BAL, R13	GENLIST	GENERATE THE LIST DATA BLOCK.
	LI, R2	CATX	SET X CATEGORY.
	STB, R2	PTRITEM	
	LI, R12	0	
	STW, R12	NILCK	ELIMINATE EXTRA REFS
	STW, R12	RESULT	
	AI, R3	-1	SET TO RESUME CODESTRING SCAN.
CXPUSHDP	CW, R1	STKLIMIT	IS THERE ROOM FOR 1 MORE WD...
	BG	CXPUSHIT	YES.
	STW, R1	LOCNEED	NO, SAVE LOC NEEDED (ABOVE LIMIT).
	BAL, R8	CTEST	GET MORE COMMON, IF POSSIBLE.
	B	CXERWS	WS FULL.
CXPUSHIT	XW, R12	PTRITEM	CLEAR & GET DB PTR & ITS CATEGORY.
	STW, R12	-1, R1	STACK IT.

A15	BDR, R1	NXTSCAN	FIX STACK PTR & GO SCAN.
	LW, R12	NILCK	GET V-ENTRY...
	BNEZ	A15OK	OK.
	B	CXERSYN	NIL -- SYNTAX ERROR.
A15OK	LI, R11	X'FFFF'	EXTRACT THE COUNT FIELD OF
	AND, R11	0, R1	THE P-ENTRY.
	BEZ	A15ONE	IF 0, USE ONE VALUE PTR (IN NILCK).
	LI, R13	A15LIST	NZ, USE LIST DATA BLOCK PTR.
	B	GENLIST	GENERATE THE LIST DATA BLOCK.
A15ONE	STW, R12	PTRITEM	SET PTR ITEM.
	AI, R1	1	PT AT ENTRY IN EXEC. STACK AFTER P.
A15LIST	LB, R2	*R1	GET NEW TOP-OF-STACK CATEGORY.
A15PAREN	BAL, R6	CXPVSET	SET UP AND RESCAN FOR PTR ITEM.
CXDOT	CI, R2	CATO	TOP CAT AN 0...
	BNE	CXERSYN	NO -- SYNTAX ERROR.
	INT, R13	0, R1	YES, EXTRACT OPTYPE FROM 0 ENTRY.
	CLM, R13	DYSLRS	IS IT ONE DYADIC SCALAR...
	BCS, 9	CXERSYN	NO -- SYNTAX ERROR.
	AI, R3	-1	YES -- BACK UP TO NEXT SCAN ITEM.
	LB, R6	*CURRCS, R3	GET IT.
	CLM, R6	DYSLRS	IS IT A DYADIC SCALAR...
	BCR, 9	A21	YES, ENTER ACTION ROUTINE.
	CI, R6	SMOCODE	NO, IS IT A SMALL CIRCLE...
	BNE	CXERSYN	NO -- SYNTAX ERROR.
A21	SLS, R6	8	OK, PUT IT IN UPPER BYTE OF
	AWM, R6	0, R1	OPTYPE FIELD OF THE 0 ENTRY.
	BDR, R3	NXTSCAN	RESUME CODESTRING SCAN.
CXAP	B	A4	BRN -- A-PRIME ENTRY @
	B	A10	BOS -- A-PRIME ENTRY @@
	B	A4	LP -- A-PRIME ENTRY @
	B	A4	LB -- A-PRIME ENTRY @
	B	A10	SEMI-- A-PRIME ENTRY @
A10	STW, R4	ACFLAG	SET ACFLAG NON-NIL. @
A4	AI, R1	1	DROP A-PRIME ENTRY.
A3	LI, R12	0	
	XW, R12	0, R1	PICK OUT THE V-ENTRY AND
	STW, R12	NILCK	PUT IT IN NILCK.
	AI, R1	1	DROP IT FROM THE EXECUTION STACK.
	LB, R2	*R1	GET SURFACED CATEGORY, I.E. TOP CAT.
	B	ZRESCAN, R2	RESCAN FOR TERMINATOR CLASS ITEM.
CXBE	B	CXERSYN	BRN -- B ENTRY @
	B	CXERSYN	BOS -- B ENTRY @@
	B	CXERSYN	LP -- B ENTRY @
	B	A14	LB -- B ENTRY @
	B	A11	SEMI-- B ENTRY @
A11	LW, R12	NILCK	GET V ENTRY OR NIL @
CXPLENT	XW, R12	0, R1	EXCHANGE (I.E. POP, SAVE, AND PUSH)
	LI, R13	0	
	STW, R13	NILCK	ELIMINATE ANY EXTRA REF TO NEW V.
	AI, R12	1	INCR COUNT.
	AI, R3	-1	SET TO RESUME CODESTRING SCAN.
CXPUSHC	CW, R1	STKLIMIT	IS THERE ROOM FOR 1 MORE WD...
	BG	CXPUSHOK	YES.
	STW, R1	LOCNEED	NO, SAVE LOC NEEDED (ABOVE LIMIT).
	BAL, R8	CTEST	GET MORE COMMON, IF POSSIBLE.
	B	CXERWS	WS FULL.
CXPUSHOK	STW, R12	-1, R1	STACK THE WD.
	BDR, R1	NXTSCAN	FIX STACK PTR & GO SCAN.
CXPE	B	CXERSYN	BRN -- P ENTRY @
	B	CXERSYN	BOS -- P ENTRY @@
	B	A15	LP -- P ENTRY @
	B	CXERSYN	LB -- P ENTRY @
	B	A12	SEMI-- P ENTRY @
A12	LW, R12	NILCK	GET V ENTRY... @
	BNEZ	CXPLENT	YES, PUSH LIST ENTRY.
	B	CXERSYN	NO -- SYNTAX ERROR.
CXSE	BDR, R3	A5	BRN -- S ENTRY (LOOK AHEAD NEXT) @
	B	A8	BOS -- S ENTRY @@
	B	CXERSYN	LP -- S ENTRY @
	B	CXERSYN	LB -- S ENTRY @
	B	A13	SEMI-- S ENTRY @
A13	LI, R7	NXTSCAN	RESUME SCANNING AFTER @
	BDR, R3	STMTV	HANDLING THE SUBSTMT VALUE.
CXQE	BDR, R3	A6	BRN -- Q ENTRY (LOOK AHEAD NEXT) @
	B	A9	BOS -- Q ENTRY @@
	B	CXERSYN	LP -- Q ENTRY @
	B	CXERSYN	LB -- Q ENTRY @
	B	CXERSYN	SEMI-- Q ENTRY @

A17	EQU	CXOPAP	
A20	EQU	CXOPAP	
CXOPAP	AI, R1	1	DROP A-PRIME ENTRY.
	B	CXOPVE-SCANOPM, R4	VECTOR ON SCAN ITEM.
CXOPVE	B	A16	ENTER ACTION ROUTINE. @
A19	BAL, R4	STACKUP	ADD 1 WD TO EXEC. STACK. @
	STW, R7	0, R1	PUT 2ND OPER WD.
	OR, R6	OENTRY	SET 0-ENTRY CATEGORY.
	STW, R6	-1, R1	PUT 1ST (CATEGORY) OPER WD.
	LI, R2	CATO	SET TOP CAT.
	AI, R1	-1	PT AT NEW TOP-0-STACK.
	BDR, R3	NXTSCAN	RESUME SCAN.
CXOPV	B	CXOPVE-SCANOPM, R4	V VECTOR ON SCAN ITEM. @
	B	CXOPAP	A-PRIME @
	B	A7	0 @
	B	A18, R5	X VECTOR ON COORD ALLOWANCE. @
	B	CXERSYN	B @
	B	CXERSYN	P @
	B	CXERSYN	S @
	B	CXERSYN	Q @
	B	CXERSYN	D @
	B	CXERSYN	A @
* NOTE -- A18 MUST BE LOCATED 1 WD AFTER A 'B CXERSYN', COORD DISALLOWD			
A18	STD, R6	OPERHOLD	HOLD CURRENT OPERATOR INFO. @
	LB, R6	OPERHOLD+1	DOES IT ALREADY HAVE A COORDINATE...
	BNEZ	CXERSYN	YES -- SYNTAX ERROR.
	STW, R4	SCANTEMP	OK, SAVE SCANOPM OR SCANOP.
	LI, R4	0	
	XW, R4	0, R1	SAVE THE X-ENTRY.
	STW, R4	XTEMP	
	AI, R1	1	DROP THE X-ENTRY.
	LB, R2	*R1	NEW TOP CAT A V-ENTRY...
	BEZ	A18X	YES.
	CI, R2	CATO	NO, AN 0-ENTRY...
	BNE	CXERSYN	NAY -- SYNTAX ERROR.
	BAL, R14	CXOPMON	AYE, POP AND EXECUTE IT MONADIC.
A18X	LW, R4	XTEMP	PT AT THE LIST DATA BLK OF X-ENTRY.
	LW, R14	2, R4	CK THAT IT IS A LIST OF LENGTH = 1.
	AI, R14	-1	
	BNEZ	CXERRANK	NO -- RANK ERROR.
	LW, R4	3, R4	YES, PT AT THE INDICATED DATA BLK.
	BEZ	CXERSYN	NO COORD GIVEN -- SYNTAX ERROR.
	BAL, R7	NUMSV	@ CK NUMERIC SCALAR OR VECTOR...
	BDR, R4	A18VAL	@ SCALAR, PT AT DATA BLK HDR + 2.
	AI, R12	-1	@ VECTOR, VERIFY LENGTH = 1.
	BNEZ	CXERRANK	OOPS -- RANK ERROR.
A18VAL	BAL, R5	IVAL	@ GET INTEGER VALUE...
	B	A18LAMCK	@ NON-INTEGER, TRY LAMINATE ON FLOOR
A18ORG	AW, R7	ORGADJ	@ ADJUST INTEGER TO INT. ORIGIN OF 1.
	BLEZ	CXERRANK	OOPS -- RANK ERROR.
	CI, R7	TOPRANK	IS RANK RIDICULOUSLY HIGH...
A18HICK	BG	CXERRANK	OOPS -- RANK ERROR.
	STB, R7	OPERHOLD+1	OK, SET COORD (MAYBE LAMINATE BIT).
	LI, R4	0	
	XW, R4	XTEMP	DEREFERENCE THE LIST DATA BLK.
	BAL, R7	DREF	
	LD, R6	OPERHOLD	GET THE OPERATOR INFORMATION.
	LW, R4	SCANTEMP	GET SCANOPM OR SCANOP.
	B	A18EXIT-SCANOPM, R4	EXIT ACCORDINGLY:
A18EXIT	B	A16	OPM @
	B	A19	OP @
A18LAMCK	AW, R7	ORGADJ	ADJUST TO INTERNAL ORIGIN OF 1.
	BLZ	CXERRANK	OOPS -- RANK ERROR.
	AI, R7	X'40'	SET LAMINATE BIT.
	LI, R5	COMMACOD	VERIFY THAT THIS IS A COMMA OPERATOR
	CB, R5	*CURRCS, R3	
	BNE	CXERRANK	OOPS -- RANK ERROR.
	CI, R7	TOPRANK+X'40'	OK, VERIFY RANK ISN'T TOO HIGH.
	B	A18HICK	
CXOP1	LI, R4	SCANOP	SET 'OP' SCAN ITEM.
CXCOORD1	LW, R7	COORD1	COORDINATE-1 OPERATION.
	OR, R7	R3	= 2ND OPER WORD.
	LI, R5	-1	TO DISALLOW FURTHER COORD SPECIFCTN.
	STW, R3	OFFSETR	SAVE OFFSET TO OPERATOR DESIGNATOR.
	B	CXOPV, R2	CONTEXT ANALYSIS START FOR OP OR OPM
CXOPM1	LI, R4	SCANOPM	SET 'OPM' SCAN ITEM.
	B	CXCOORD1	
CXOP	LI, R4	SCANOP	SET 'OP' SCAN ITEM.

CXNC00R3	STW, R3	OFFSETR	SAVE OFFSET TO OPERATOR DESIGNATOR.	
CXNC00RD	LI, R5	-1	TO DISALLOW COORDINATE SPECIFICATION	
	LW, R7	R3	= 2ND OPER WORD.	
	B	CXOPV, R2	CONTEXT ANALYSIS START FOR OP OR OPM	
CXOPM	LI, R4	SCANOPM	SET 'OPM' SCAN ITEM.	
	B	CXNC00R3		
CXCOP	LI, R4	SCANOP	SET 'OP' SCAN ITEM.	
CXCOK	LI, R5	0	TO ALLOW COORDINATE SPECIFICATION.	
	LW, R7	R3	= 2ND OPER WORD, SO FAR.	
	STW, R3	OFFSETR	SAVE OFFSET TO OPERATOR DESIGNATOR.	
	B	CXOPV, R2	CONTEXT ANALYSIS START FOR OP OR OPM	
CXCOPM	LI, R4	SCANOPM	SET 'OPM' SCAN ITEM.	
	B	CXCOK		
CXREDSET	OR, R7	R3	= 2ND OPER WD, PTS AT DYADIC SCALAR.	
	AI, R6	X'100'	SET THE 'REDUCTION' FLAG.	
	B	CXOPV, R2	CONTEXT ANALYSIS FOR THIS OPM.	
CXCRED	LI, R5	0	TO ALLOW COORDINATE SPECIFICATION.	
	LI, R7	0	START WITH NO COORDINATE.	
CXRED	LI, R4	SCANOPM	SET 'OPM' SCAN ITEM.	
	STW, R3	OFFSETR	SAVE OFFSET TO REDUCE-OP DESIGNATOR.	
	AI, R3	-1	GET THE CODESTRING DESIGNATOR TO THE	
	LB, R6	*CURRCS, R3	LEFT OF THE REDUCTION SLASH.	
	CLM, R6	DYSLCLRS	IS IT A DYADIC SCALAR...	
	BCR, 9	CXREDSET	YES, SET FOR REDUCTION.	
	AI, R6	DYADIC-MONADIC	NO, MAYBE IT WAS CODESTRUNG AS AN	
	CLM, R6	MDYSLCLRS	OPM. TEST BIASED VERSION...	
	BCR, 9	CXREDSET	NOW ITS A DYADIC SCALAR.	
	B	CXERSYN	00PS -- SYNTAX ERROR.	
CXRED1	LI, R5	-1	TO DISALLOW FURTHER COORD SPECIFCTN.	
	LW, R7	C00RD1	COORDINATE-1 REDUCTION.	
	B	CXRED		
CKSCANOP	LI, R4	1	PUT OFFSET TO LEFT-MOST OPERATOR	U19-0015
	STH, R3	OPER+1, R4	IN 'CURRBYTE' IN CASE ERROR.	U19-0016
	MTW, -1	OFFSETR	UPDATE SAVED OFFSET FOR A7 RETURN.	U19-0017
	LB, R6	*CURRCS, R3	GET CURRENT CODESTRING DESIGNATOR.	U19-0018
	CLM, R6	DYSLCLRS	IS IT A DYADIC SCALAR...	U19-0019
	BCR, 9	CKSCANOK	YES, SET FOR SCAN-OPERATOR EXEC.	U19-0020
	AI, R6	DYADIC-MONADIC	NO, BUT MIGHT BE MONADIC VERSION.	U19-0021
	CLM, R6	MDYSLCLRS	NOW IS IT A DYADIC SCALAR...	U19-0022
	BCS, 9	CXERSYN	NOPE -- SYNTAX ERROR.	U19-0023
CKSCANOK	STW, R6	OPER	USE THAT OP FOR OPER TEMPS.	U19-0024
	B	SCAN		U19-0025
A7R	STW, R3	OFFSETR	SAVE OFFSET TO SCAN ITEM DESIGNATOR.	
A7	BAL, R14	CXPOMON	HANDLE STACKED OP MONADICALLY.	
	LW, R3	OFFSETR	RESTORE SAVED OFFSET.	
	B	NXTSCAN		
CXSEMI	LI, R4	SCANSEMI	SET 'SEMI' SCAN ITEM.	
CXACFLAG	LI, R13	0	CLEAR ACFLAG, I.E. SET NIL.	
	STW, R13	ACFLAG		
CXZCLASS	LI, R13	0	(TERMINATOR CLASS OF ITEM)	
	STW, R13	NILCK	CLEAR NILCK, I.E. SET NIL.	
	B	ZRESCAN, R2	CONTEXT ANALYSIS START FOR Z-CLASS.	
ZRESCAN	B	A3	TOP-CAT = V	@
	B	CXAP-SCANBRN, R4	A-PRIME VECTOR ON SCAN ITEM.	@
	B	A7R	0	@
	B	CXERSYN	X	@
	B	CXBE-SCANBRN, R4	B VECTOR ON SCAN ITEM.	@
	B	CXPE-SCANBRN, R4	P VECTOR ON SCAN ITEM.	@
	B	CXSE-SCANBRN, R4	S VECTOR ON SCAN ITEM.	@
	B	CXQE-SCANBRN, R4	Q VECTOR ON SCAN ITEM.	@
	B	CXERSYN	D	@
	B	CXERSYN	A	@
CXBRN	LI, R4	SCANBRN	SET 'BRN' SCAN ITEM.	
	B	CXZCLASS		
CXBOS	LI, R4	SCANBOS	SET 'BOS' SCAN ITEM.	
	B	CXACFLAG		
CXLP	LI, R4	SCANLP	SET 'LP' SCAN ITEM.	
	B	CXZCLASS		
CXLB	LI, R4	SCANLB	SET 'LB' SCAN ITEM.	
	B	CXZCLASS		
CXRB	CI, R2	CATAPRYM	TOP CAT AN A-PRIME...	
	BNE	A1	NO.	
A2	AI, R1	1	YES, DROP A-PRIME ENTRY.	
A1	LW, R12	BENTRY	GET B-ENTRY WITH COUNT = 0.	
CXPCNXT	LB, R2	R12	SET TOP CAT FOR NEXT	
	BDR, R3	CXPUSHC	CATEGORY AND SCAN AFTER PUSHING.	
A34	LW, R12	AENTRY	GET A-ENTRY.	
	B	CXPCNXT		

A24	LW, R12	PENTRY	GET P-ENTRY.	
	B	CXPCNXT		
A28	AI, R1	1	DROP A-PRIME ENTRY.	
A27	LI, R12	X'1FFFF'	OBTAIN LOC OF FUNCTION DESCRIPTOR	
	AND, R12	*SYMT, R4	DATA BLOCK.	
	OR, R12	DENTRY	D ENTRY CATEGORY (DYADIC FUNC).	
	LI, R2	CATD	SET NEW TOP CAT.	
	XW, R12	PTRITEM	SAVE D-ENTRY & CLEAR R12.	
	MTW, 1	1, R5	INCR REF COUNT OF FUNDEF DATA BLK.	
	BDR, R3	CXPUSHDP	PUSH D-ENTRY & RESUME SCAN.	
CXQUAD	CLM, R2	CATO2D	TOP CAT FOR QUAD...	
	BCR, 9	A25	OK, INPUT.	
	BCS, 8	A38	OK, OUTPUT.	
	B	CXERSYN	BAD -- SYNTAX ERROR.	
CXRP	CLM, R2	CATO2D	TOP CAT OK FOR RIGHT PAREN...	
	BCR, 9	A24	OK.	
	B	CXERSYN	NO -- SYNTAX ERROR.	
CXASS	CI, R2	CATO	TOP CAT OK FOR ASSIGN ARROW...	
	BL	CXASSV, R2	OK, HIT A34 OR A35.	
	BE	A7R	MAYBE, HIT A7.	
	B	CXERSYN	BAD, SYNTAX ERROR.	
CXASSV	B	A34		@
A35	LI, R2	CATA	CONVERT A-PRIME TO A ENTRY	@
	STB, R2	*R1	AT TOP OF STACK.	
	BDR, R3	NXTSCAN		
A36	CI, R5	LBLFLAG	TRYING TO ASSIGN TO A LABEL...	
	BAZ	AOK	NO.	
	B	CXERSYN	YES -- SYNTAX ERROR.	
AOK	LW, R7	NILCK	IS THIS AN INDEXED ASSIGNMENT...	
	BEZ	ASIMPLE	NO, SIMPLE ASSIGNMENT.	
	STW, R1	TOPOSTAK	YES, SAVE STACK PTR.	
	STW, R3	OPER+1	SAVE CURRENT CODESTRING OFFSET.	
	LW, R6	1, R1	GET V-ENTRY.	
	STW, R6	RTARG	IT IS THE RT. ARGUMENT FOR ASS.	
	MTW, 1	1, R6	INCR REF COUNT OF ITS DATA BLK.	
	LW, R3	NAMEPTR	PT TO REF-INDIC. WD FOR VN.	
	LI, R4	X'1FFFF'		
	AND, R4	*SYMT, R3	GET REFERENT DATA BLK PTR.	
	MTW, 1	1, R4	BUMP ITS REF-COUNT	
	STW, R4	LFARG	MOVE IT INTO LF. ARG. PTR.	
	B	AXDRIVER	ENTER INDEXED-ASSIGNMENT EXECUTION	
*			DRIVER, IT WILL RETURN TO RETURNAX	
RETURNAX	LW, R1	TOPOSTAK	RESTORE STACK PTR; (R3 SET).	
	LI, R4	0		
	XW, R4	RTARG	DE-REF. (THE EXTRA REF) TO RTARG.	U19-0027
	BAL, R7	DREF		
	LI, R4	0		
	XW, R4	LFARG	DE-REF. OLD DATA BLK FOR VN.	U19-0029
	BAL, R7	DREF		
	LW, R5	NAMEPTR	PT TO REF-INDICATOR FOR VN.	
	LW, R4	*SYMT, R5	GET OLD DATA BLK REFERENCE.	
	LI, R7	X'1FFFF'		
	LI, R6	0		
	XW, R6	RESULT	MOVE NEW DATA BLK PTR INTO SYM TBL.	
	STS, R6	*SYMT, R5		
	LI, R7	A36DONE	GO TO 'A36DONE' AFTER	
	B	DREF	DE-REFING OLD DATA BLK.	
ASIMPLE	LW, R6	NAMEPTR	PT TO REF-INDIC. WD FOR VN.	
	LI, R4	X'E0000'		
	AND, R4	*SYMT, R6	COMBINE FLAG-BITS AND NEW DATA BLK	
	OR, R4	1, R1	PTR (V-ENTRY).	
	MTW, 1	1, R4	INCR REF-COUNT OF THAT DATA BLK.	
	XW, R4	*SYMT, R6	INSERT NEW REFERENT-INDICATOR WD.	
	BAL, R7	MAYDREF	DEREFERENCE THE OLD (UNLESS NIL).	
A36DONE	LI, R2	CATAPRYM	SET TOP CAT TO BE A-PRIME (I.E.	
	STB, R2	*R1	ASSIGNMENT-COMPLETED).	
	LW, R4	CATCHTBL	IS A CATCH IN EFFECT...	
	BLZ	NXTSCANU	NO, RESUME SCAN.	
	CW, R4	NAMEPTR	YES, FOR THIS VN...	
	BNE	A36CATC2	NO, TRY OTHER CATCH.	
	LW, R6	CATCHTBL+1	YES, GET FUN.NAME PTR.	
A36CATCH	LI, R5	X'1FFFF'	EXTRACT REF-PTR FOR FUN.NAME...	
	AND, R5	*SYMT, R6		
	BEZ	NXTSCANU	NONE, RESUME SCAN.	
	LB, R6	*R5	GET DATA BLK TYPE.	
	AI, R6	-TYPENFN	VERIFY NILADIC FUNC., NO RESULT...	
	BNEZ	NXTSCANU	NOPE, RESUME SCAN.	
	B	A26	OK, EXECUTE CATCH FUNCTION.	

A36CATC2	LW, R4	CATCHTBL+2	IS 2ND CATCH IN EFFECT...
	BLZ	NXTSCANU	NO, RESUME SCAN.
	CW, R4	NAMEPTR	YES, FOR THIS VN...
	BNE	NXTSCANU	NO, RESUME SCAN.
	LW, R6	CATCHTBL+3	YES, GET FUN.NAME PTR.
	B	A36CATCH	
A32	B	A32VORP, R4	VECTOR ON SCANITEM VN OR PTR:
A32VORP	B	A32NILCK	@ 0 -- PTR.
*	B	A32VN	@ 1 -- VN.
A32VN	AND, R5	X1FFFF	@ EXTR. DATA BLK PTR FROM REF-IND. WD.
	STW, R5	PTRITEM	
	MTW, 1	1, R5	INCR REF COUNT OF THAT DATA BLOCK.
A32NILCK	LI, R14	CXPUSHV	SET TO EXIT TO CXPUSHV.
QNILCK	LW, R7	NILCK	IS THERE A SUBSCRIPT...
	BNEZ	SUBEX	YES.
QOBS	LW, R7	OBSERVE	NO, OBSERVE SET...
	BEZ	*R14	NOPE -- EXIT.
	AI, R6	-A15PAREN-1	YEP.
	BEZ	*R14	(EXIT IF VIA PAREN -- REDUNDANT).
	STW, R3	OFFSET	SAVE CS POS. FOR OBSERVE'S MARKER.
	LW, R6	PTRITEM	PT AT OPERAND'S DATA BLK.
	B	OBSERVER	OBSERVE THAT VALUE,
OBSERVED	B	*R14	RESUME HERE.
SUBEX	STW, R3	OPER+1	SAVE CURRENT CODESTRING OFFSET.
	STW, R1	TOPOSTAK	SAVE EXECUTION STACK PTR.
	STW, R14	LINKX	SAVE LINK TO SUBEX.
	LW, R6	BREAKFLG	TEST FOR BREAK...
	BEZ	SXDRIVER	NO -- ENTER SUBSCRIPTING DRIVER,
	B	CXBREAK	
*			IT WILL RETURN TO RETURNSX.
RETURNSX	LW, R1	TOPOSTAK	RESTORE STACK PTR; (R3 SET).
	LI, R4	0	
	XW, R4	RTARG	DE-REF. RTARG (WAS LFARG WHEN
*			ENTERING 'SXDRIVER').
	BAL, R7	DREF	
	LI, R5	0	
	XW, R5	RESULT	MOVE RESULT INTO PTRITEM.
	STW, R5	PTRITEM	
	LW, R14	LINKX	SET TO EXIT VIA SAVED LINK (SUBEX)
	B	QOBS	AFTER QUERY OBSERVE SETTING.
CXPUSHV	BAL, R4	STACKUP	ADD 1 LOC TO EXECUTION STACK.
	LI, R5	0	GET PTR OUT OF PTRITEM
	XW, R5	PTRITEM	= V ENTRY FOR NEW STACK LOC.
	STW, R5	0, R1	FILL IT WITH V-ENTRY.
	LI, R2	CATV	TOP CAT IS FOR V-ENTRY.
	BDR, R3	NXTSCAN	RESUME SCAN.
A23	B	A23VORP, R4	VECTOR ON SCANITEM VN OR PTR:
A23VORP	B	A23NILCK	@ 0 -- PTR.
*	B	A23VN	@ 1 -- VN.
A23VN	AND, R5	X1FFFF	@ EXTR. DATA BLK PTR FROM REF-IND. WD.
	STW, R5	PTRITEM	
	MTW, 1	1, R5	INCR REF COUNT OF THAT DATA BLOCK.
A23NILCK	BAL, R14	QNILCK	QUERY SUBSCRIPTING.
DYEX	STW, R3	OFFSET	SAVE CODESTRING OFFSET
	LI, R6	X'FFFF'	GET OPTYPE
	AND, R6	0, R1	AND
	LW, R7	1, R1	2ND OPER. INFO WD.
	STD, R6	OPER	SAVE BOTH.
	STW, R7	OPCURRBY	(SAVE CURR.BYTE FOR OPWRAP).
	AI, R1	2	DROP 0-ENTRY FROM EXECUTION STACK.
	LI, R14	0	PULL THE
	XW, R14	0, R1	V-ENTRY AND
	STW, R14	RTARG	MAKE IT THE RT ARGUMENT.
*			LFARG (LEFT ARG) IS IN 'PTRITEM'
*			LFARG AND PTRITEM ARE SAME CELL.
	STW, R1	TOPOSTAK	HOLD TOP-0-STACK, RESULT WILL FILL.
	LW, R5	BREAKFLG	TEST FOR BREAK...
	BEZ	DRANGECK	NO -- RANGE CHECK THE OPER.
	B	CXBREAK	YES.
DRANGECK	CLM, R6	DYRANGE	CK FOR ORDINARY DYADIC OP.
	BCR, 9	DXDRIVER	YES, ENTER DYADIC EXECUTION DRIVER,
*			IT WILL RETURN TO RETURNDX.
	LI, R5	2	GET MOST SIGNIFICANT BYTE OF THE
	LB, R5	OPER, R5	OPTYPE FIELD.
	CI, R5	SMOCODE	CK FOR SMALL CIRCLE...
	BG	INNER	HI -- INNER PRODUCT...
	BE	OUTER	ON -- OUTER PRODUCT...
	AI, R6	-DYINFLAG	(REMOVE DYADIC-INTRINSIC FLAG)

	B	DINTRIN	LO -- DYADIC INTRINSIC...
*			NOTE--THEY ALL RETURN TO RETURNNDX.
RETURNNDX	LW, R1	TOPOSTAK	RESTORE STACK PTR; (R2=0, I.E. CATV)
	LW, R3	OFFSET	RESTORE CODESTRING OFFSET.
	LI, R4	0	
	XW, R4	LFARG	DEREFERENCE LEFT ARG.
	BAL, R7	DREF	
	LI, R14	NXTSCANU	SET TO EXIT TO NXTSCANU.
OPWRAP	LI, R4	0	WRAP UP OPERATOR.
	XW, R4	RTARG	
	BAL, R7	DREF	DEREFERENCE RIGHT ARG.
	LI, R6	0	
	XW, R6	RESULT	MOVE RESULT DB PTR INTO TOP-0-STACK.
	STW, R6	0, R1	
	LW, R7	OBSERVE	IS OBSERVE SET...
	BEZ	*R14	NO -- EXIT.
	INT, R7	OPCURRBY	YES, SAVE CURR.BYT TO OPERATOR.
	STW, R7	OFFSET	RESUME FOR EXIT AT OBSERVED AFTER
	B	OBSERVER	DISPLAYING THE OBSERVATION.
A33	LW, R7	NILCK	IS ITEM ALREADY TO BE INDEXED...
	BEZ	A33N	NO.
	B	A33VORP, R4	YES, VECTOR ON SCANITEM VN OR PTR:
A33VORP	B	A33X	@ 0 -- PTR.
*	B	A33VN	@ 1 -- VN.
A33VN	AND, R5	X1FFFF	@ EXTR. DATA BLK PTR FROM REF-IND. WD.
	STW, R5	PTRITEM	
	MTW, 1	1, R5	INCR REF COUNT OF THAT DATA BLOCK.
A33X	BAL, R14	SUBEX	EXECUTE SUBSCRIPTING (ONCE).
	LI, R4	0	SET SCANITEM FOR PTR (NOT VN).
A33N	LI, R7	0	
	XW, R7	0, R1	PULL X-ENTRY OUT OF TOP-0-STACK
	STW, R7	NILCK	AND SAVE IT IN NILCK (NON-NIL).
	AI, R1	1	POP AND GET
	LB, R2	*R1	NEW TOP CAT.
	B	CXPTR, R2	CONTEXT ANALYSIS FOR VN OR PTR.
CXNMV	LI, R4	0	
	STW, R4	NILCK	CLEAR NILCK FLAG, I.E. SET NIL.
	LI, R4	SCANVN	SET 'VN' SCAN ITEM.
	B	CXPTR, R2	CONTEXT ANALYSIS FOR VN.
CXNMNI	CLM, R2	CAT02D	TOP CAT OK FOR NILADIC INTRINSIC...
	BCS, 9	CXERSYN	NO -- SYNTAX ERROR.
	LH, R2	*R5	YES, EXTRACT INTRINSIC'S NO.
	AI, R2	-(TYPEINI**8)	
	STD, R2	OPER	SAVE I AND SAVE OFFSET TO CODESTRING
	STW, R1	TOPOSTAK	SAVE STACK PTR.
	LW, R4	BREAKFLG	TEST FOR BREAK...
	BEZ	NINTRIN	NO -- EXEQ NILADIC INTRINSIC & GO
	B	CXBREAK	
*			TO RETURNNI AFTER 'RESULT' SET.
RETURNNI	LW, R1	TOPOSTAK	RESTORE STACK PTR (NOTE--R4 = 0).
	XW, R4	RESULT	MOVE RESULT INTO PTRITEM.
	STW, R4	PTRITEM	
	B	FPOPPED	ASSUME WE'VE POPPED AN F-ENTRY.
CXNMNI	CI, R2	CATO	IS AN OPERATOR AT TOP OF STACK...
	BE	A22	YES -- IT IS MONADIC.
	LH, R6	*R5	NO, EXTRACT MONADIC-INTRINSIC'S
	AI, R6	-(TYPEMI**8)	'OPERATOR' NO.
	LI, R4	SCANOPM	SET 'OPM' SCAN ITEM.
	B	CXNCOORD	
CXNMNI	CI, R2	CATO	IS AN OPERATOR AT TOP OF STACK...
	BE	A22	YES -- IT IS MONADIC.
	LH, R6	*R5	NO, EXTR. DYADIC-INTRIN'S 'OPER.'
	AI, R6	-(TYPEDI**8)+DYINFLAG	NO. AND FLAG IT.
	LI, R4	SCANOP	SET 'OP' SCAN ITEM.
	B	CXNCOORD	
CXNMNI	CLM, R2	CAT02D	TOP CAT OK FOR NILADIC FUNCTION...
	BCR, 9	A26	YES, ENTER ACTION ROUTINE.
	B	CXERSYN	NO -- SYNTAX ERROR.
CXNMNI	CI, R2	CATO	TOP CAT OK FOR DYADIC FUNCTION...
	BLE	CXNMDFV, R2	YES, HIT PROPER ACTION ROUTINE.
	B	CXERSYN	NO -- SYNTAX ERROR.
CXNMDFV	B	A27	@
	B	A28	@
	B	A22	@
CXNMNI	CI, R2	CATO	TOP CAT OK FOR MONADIC FUNCTION...
	BLE	CXNMDFV, R2	YES, HIT PROPER ACTION ROUTINE.
	B	CXERSYN	NO -- SYNTAX ERROR.
CXNMDFV	B	A29	@

	B	A30		@
A22	BAL, R14	CXPOPMON	HANDLE STACKED OP MONADICALLY.	
	LW, R4	NAMEPTR	RESTORE PTR TO FUNCTION NAME.	
	B	CXNRESC	RESCAN FOR THAT NAME.	
CXPOPMON	LI, R6	X'FFFF'	GET OPTYPE	
	AND, R6	0, R1	AND	
	LW, R7	1, R1	2ND OPER. INFO WD.	
	STD, R6	OPER	SAVE BOTH.	
	AI, R1	2	DROP 0-ENTRY FROM EXECUTION STACK.	
MONEX	STW, R14	LINKMX	SAVE LINK.	
	STW, R3	OFFSET	SAVE CODESTRING OFFSET.	
	STW, R7	OPCURRBY	(SAVE CURR.BYTE FOR OPWRAP).	
	LI, R14	0	PULL THE	
	XW, R14	0, R1	V-ENTRY AND	
	STW, R14	RTARG	MAKE IT THE RT ARGUMENT.	
	STW, R1	TOPOSTAK	HOLD TOP-0-STACK, RESULT WILL FILL.	
	LW, R5	BREAKFLG	TEST FOR BREAK...	
	BEZ	MRANGECK	NO -- RANGE CHECK THE OPER.	
	B	CXBREAK	YES.	
MRANGECK	CLM, R6	MONRANGE	CK FOR ORDINARY MONADIC OP.	
	BCR, 9	MXDRIVER	YES, ENTER MONADIC EXECUTION DRIVER,	
*			IT WILL RETURN TO RETURNMX.	
*	BCS, 1	MINTRIN	NO, (LO) TRY INTRINSIC MONADIC, IT	
			TOO WILL RETURN TO RETURNMX.	
	CLM, R6	REDRANGE	NO, (HI) CK FOR REDUCTION RANGE.	
	BCR, 9	REDUCE	OK, REDUCE WILL RETURN TO RETURNMX	
	CLM, R6	S2XRANGE	NO, CK FOR SCAN OR EXECUTE CASE... U19-0031	
	BCR, 9	SORXV-SCAN1COD, R6	YES, VECTOR ACCORDING TO WHICH. U19-0032	
	B	CXERSYN	OOPS -- SYNTAX ERROR	
RETURNMX	LW, R1	TOPOSTAK	RESTORE STACK PTR; (R2=0, I.E. CATV)	
	LW, R3	OFFSET	RESTORE CODESTRING OFFSET.	
	LW, R14	LINKMX	SET TO EXIT VIA SAVED LINK (MONEX)	
	B	OPWRAP	AFTER WRAPPING UP THE OPERATION.	
CXSLOGL1	LI, R6	-1	PUT A 1 IN BIT POSITION ZERO.	
CXSLOGL0	LI, R13	TYPEOGL	LOGICAL TYPE DATA BLOCK.	
	B	CXGENWD	GEN LOGICAL SCALAR DATA BLOCK.	
CXSIBYTE	LB, R6	*CURRCS, R3	GET BYTE-FORM OF INTEGER.	
CXSI09	LI, R13	TYPEINTG	FOR INTEGER TYPE OF DATA BLOCK.	
CXGENWD	LI, R11	2	ASK FOR 2 WDS PLUS HEADER.	
	STW, R1	TOPOSTAK	SAVE STACK PTR. IN CASE WS FULL.	
	STW, R3	OPER+1	SAVE OFFSET IN CASE WS FULL OCCURS.	
	BAL, R7	ALOCHNW	ALLOCATE DATA BLOCK.	
CXSTWD	STW, R6	2, R4	FILL IN VALUE WORD.	
	STW, R4	PTRITEM	SAVE LOC OF DATA BLOCK FOR PTR ITEM.	
	STB, R13	*PTRITEM	FILL IN DATA BLOCK TYPE.	
	LI, R4	0	SET 'PTR' SCAN ITEM.	
	STW, R4	NILCK	CLEAR NILCK FLAG, I.E. SET NIL.	
	B	CXPTR, R2	CONTEXT ANALYSIS FOR PTR.	
CXPTR	B	CXERSYN	V -- SYNTAX ERROR.	@
	B	CXERSYN	A-PRIME -- SYNTAX ERROR.	@
	B	A23	0	@
	B	A33	X	@
	B	A32	B	@
	B	A32	P	@
	B	A32	S	@
	B	A32	Q	@
	B	A31	D	@
	B	CXVORP, R4	A -- IS SCAN ITEM PTR OR VN...	@
CXVORP	B	CXERSYN	@ PTR -- SYNTAX ERROR.	@
	B	A36	@ VN, ENTER ACTION ROUTINE.	
CXSIWORD	SLS, R3	-2	WORD OFFSET TO INTEGER VALUE.	
	LW, R6	*CURRCS, R3	GET INTEGER.	
	SLS, R3	2	OFFSET TO LEFTMOST BYTE OF INTEGER.	
	B	CXSI09		
CXSREAL	SLS, R3	-2	WORD OFFSET TO 2ND REAL WD.	
	LW, R13	*CURRCS, R3	GET 2ND REAL WD.	
	AI, R3	-1	OFFSET TO 1ST REAL WD.	
	LW, R6	*CURRCS, R3	GET 1ST REAL WD.	
	SLS, R3	2	OFFSET TO LEFTMOST BYTE OF REAL PAIR	
	LI, R11	2	ASK FOR 2 WDS PLUS HEADER.	
	STW, R1	TOPOSTAK	SAVE STACK PTR. IN CASE WS FULL.	
	STW, R3	OPER+1	SAVE OFFSET IN CASE WS FULL OCCURS.	
	BAL, R7	ALOCHNW	ALLOCATE DATA BLOCK.	
	STW, R13	3, R4	FILL IN 2ND REAL WD.	
	LI, R13	TYPEREAL	FOR REAL TYPE OF DATA BLOCK.	
	B	CXSTWD	FILL IN 1ST REAL WD AND SET PTR.	
CXSTEXT	LB, R6	*CURRCS, R3	GET TEXT SCALAR BYTE.	
	SCS, R6	-8	PUT IT IN BYTE ZERO.	

	LI, R13	TYPETEXT	FOR TEXT TYPE OF DATA BLOCK.
	B	CXGENWD	GEN THAT DATA BLOCK.
CXVINTG	LI, R6	TYPEINTG	FOR INTEGER TYPE OF DATA BLOCK.
CXVLEN	LB, R11	*CURRCS, R3	GET LENGTH BYTE
CXVALOC	STW, R11	RSSIZE	= NO.OF ELEMENTS OF TYPE.
	STW, R6	RSTYPE	TYPE OF DATA BLOCK.
	LI, R14	1	RANK IS 1 FOR VECTOR.
	STW, R14	RSRANK	
	STW, R1	TOPOSTAK	SAVE STACK PTR. IN CASE WS FULL.
	STW, R3	OPER+1	SAVE OFFSET IN CASE WS FULL OCCURS.
	BAL, R14	ALOCRS	ALLOCATE 'RESULT' DATA BLOCK.
	STW, R4	PRITEM	SAVE LOC OF DATA BLK FOR PTR ITEM.
	AI, R4	2	PT AT LENGTH WD OF DATA BLK.
	LW, R5	RSSIZE	GET LENGTH AGAIN.
	STW, R5	0, R4	PUT IT IN DATA BLK.
	BEZ	CXPVSET	EMPTY VECTOR.
CXFILLV	B	CXFILLV, R6	TYPE... (0 IS IMPOSSIBLE)
	B	CXFLOGL	@ 1 FILL IN LOGICAL VALUES.
	B	CXFTEXT	@ 2 FILL IN TEXT VALUES.
	B	CXFINTG	@ 3 FILL IN INTEGER VALUES.
*	B	CXFREAL	@ 4 FILL IN REAL VALUES.
CXFREAL	AI, R4	1	@ WD AFTER LENGTH IS GARBAGE, WE NEED
	SLS, R5	1	EVEN BNDRY. 2 WDS PER ELEMENT.
	STW, R5	RSSIZE	SET SIZE IN WORDS, NOT DBLWDS.
CXFINTG	AW, R4	RSSIZE	PT AT LAST VALUE WD OF DATA BLK.
	SLS, R3	-2	WD OFFSET TO LENGTH BYTE IN CODESTR.
CXFWORD	AI, R3	-1	FILL IN WORD VALUES,
	LW, R10	*CURRCS, R3	FROM LAST
	STW, R10	0, R4	TO FIRST.
	AI, R4	-1	
	BDR, R5	CXFWORD	
	SLS, R3	2	OFFSET TO LEFTMOST BYTE OF 1ST WD.
CXPVSET	LI, R4	0	SET UP FOR PTR ITEMS OF VECTOR CLASS
	STW, R4	RESULT	ELIMINATE THE EXTRA REF TO DATA BLK.
	STW, R4	NILCK	CLEAR NILCK FLAG, I.E. NIL.
	B	CXPTR, R2	CONTEXT ANALYSIS FOR PTR.
CXFLOGL	AI, R5	7	CALC. NO.OF BYTES
	SLS, R5	-3	FOR LOGICAL VALUES
	STW, R5	RSSIZE	
CXFTEXT	SLS, R4	2	PT AT BYTE IN DATA BLOCK THAT
	AI, R4	3	PRECEDES 1ST VALUE BYTE SPOT.
	AW, R4	RSSIZE	PT AT LAST VALUE BYTE TO FILL.
CXFBYTE	AI, R3	-1	FILL IN BYTE VALUES,
	LB, R10	*CURRCS, R3	FROM LAST
	STB, R10	0, R4	TO FIRST.
	AI, R4	-1	
	BDR, R5	CXFBYTE	
	B	CXPVSET	FINISH POINTING AT LEFTMOST VAL BYTE
CXVLOGL	LI, R6	TYPELOGL	FOR LOGICAL TYPE DATA BLOCK.
	B	CXVLEN	
CXVREAL	LI, R6	TYPEREAL	FOR REAL TYPE DATA BLOCK.
	B	CXVLEN	
CXVTEXT	LI, R6	TYPETEXT	FOR TEXT TYPE DATA BLOCK.
	B	CXVLEN	
CXVTEXTL	LB, R11	*CURRCS, R3	GET LO BYTE OF LONG-TEXT LENGTH.
	SCD, R10	-8	HOLD IT.
	AI, R3	-1	OFFSET TO HI BYTE.
	LB, R11	*CURRCS, R3	GET IT.
	SCD, R10	8	FORM LENGTH HALFWORD, IN R11.
	LI, R6	TYPETEXT	FOR TEXT TYPE DATA BLOCK.
	B	CXVALOC	PREPARE TO ALLOCATE IT.
CXCOMMNT	LB, R6	*CURRCS, R3	GET LENGTH OF COMMENT.
	SW, R3	R6	OFFSET TO 1ST COMMENT CHAR, IF ANY.
	BDR, R3	NXTSCAN	FORGET IT.
CXSTORTR	LB, R4	*CURRCS, R3	GET RIGHT BYTE OF NAME POINTER.
	SLD, R4	-8	SAVE IT.
	AI, R3	-1	GET LEFT BYTE.
	LB, R4	*CURRCS, R3	
	SLD, R4	8	= THE NAME POINTER.
	LI, R5	X'1FFFF'	EXTRACT REFERENT, I.E.
	AND, R5	*SYMT, R4	DATA BLOCK PTR.
	BEZ	CXERUND	OOPS, NONE -- UNDEFINED.
	LB, R10	*R5	OK, GET ITS TYPE.
	CLM, R10	FUNTYPES	VERIFY THAT ITS A FUNCTION NAME.
	BCS, 9	CXERSYN	GOOF -- SYNTAX ERROR.
	CI, R2	CATA	IS TOP-0-STACK CATEGORY AN A...
	BE	A37	YES, ENTER ACTION ROUTINE.
	B	CXERSYN	NO, SYNTAX ERROR

CXNAME	STW, R3	OFFSETR		SAVE OFFSET TO NAME DESIGNATOR.
	AI, R3	-1		
	LB, R4	*CURRCS, R3		GET RIGHT BYTE OF NAME POINTER.
	SLD, R4	-8		HOLD IT.
	AI, R3	-1		GET LEFT BYTE.
	LB, R4	*CURRCS, R3		
	SLD, R4	8		= THE NAME POINTER.
	STW, R4	NAMEPTR		
CXNRESC	LW, R5	*SYMT, R4		GET ITS REFERENT INDICATOR WORD.
	CI, R5	X'1FFFF'		IS THERE A REFERENT...
	BANZ	CXNAMFND		YES.
	CI, R2	CATA		NO, IS THIS AN ASSIGNMENT...
	BE	CXNMV		YEP, GO AHEAD.
	B	CXERUND		NOPE -- UNDEFINED.
CXNAMFND	LB, R6	*R5		GET TYPE OF DATA BLK REFERENCED.
	B	CXNAMTYP, R6		
CXNAMTYP	BAL, R15	SYSTERR	@ 0	-- SHOULD BE IMPOSSIBLE.
	B	CXNMV	@ 1	-- LOGL VARIABLE
	B	CXNMV	@ 2	-- TEXT VARIABLE
	B	CXNMV	@ 3	-- INTG VARIABLE
	B	CXNMV	@ 4	-- REAL VARIABLE
	B	CXNMV	@ 5	-- INDEX SEQUENCE VARIABLE
	B	CXNMV	@ 6	-- LIST VARIABLE
	BAL, R15	SYSTERR	@ 7	-- CODESTRING -- MAYBURY'S ERROR
	B	CXNMNF	@ 8	-- NILADIC FUNC, NO RESULT.
	B	CXNMNF	@ 9	-- NILADIC FUNC, RESULT.
	B	CXNMNF	@ 10	-- MONADIC FUNC, NO RESULT.
	B	CXNMNF	@ 11	-- MONADIC FUNC, RESULT.
	B	CXNMDF	@ 12	-- DYADIC FUNC, NO RESULT.
	B	CXNMDF	@ 13	-- DYADIC FUNC, RESULT.
	B	CXNMDI	@ 14	-- DYADIC INTRINSIC.
	B	CXNMMI	@ 15	-- MONADIC INTRINSIC.
	B	CXNMNI	@ 16	-- NILADIC INTRINSIC.
	B	CXERSYN	17	-- GROUP -- SYNTAX ERROR.
CXBADCD	EQU	SYSTERR -- CRASH IF BAD CODE; SOMETHING IS HAYWIRE.		
CXV	B	CXSLOGL0	0@ 0	LOGICAL SCALAR CONSTANT
	B	CXSLOGL1	1@ 1	LOGICAL SCALAR CONSTANT
	B	CXSI09	2@ 2	SCALAR CONSTANT
	B	CXSI09	3@ 3	SCALAR CONSTANT
	B	CXSI09	4@ 4	SCALAR CONSTANT
	B	CXSI09	5@ 5	SCALAR CONSTANT
	B	CXSI09	6@ 6	SCALAR CONSTANT
	B	CXSI09	7@ 7	SCALAR CONSTANT
	B	CXSI09	8@ 8	SCALAR CONSTANT
	B	CXSI09	9@ 9	SCALAR CONSTANT
	BDR, R3	CXSI BYTE	10@	INTEGER BYTE SCALAR CONSTANT
	BDR, R3	CXSI WORD	11@	INTEGER WORD SCALAR CONSTANT
	BDR, R3	CXSREAL	12@	REAL SCALAR CONSTANT
	BDR, R3	CXSTEXT	13@	TEXT SCALAR CONSTANT
	BDR, R3	CXVLOGL	14@	LOGICAL VECTOR CONSTANT
	BDR, R3	CXVINTG	15@	INTEGER VECTOR CONSTANT
	BDR, R3	CXVREAL	16@	REAL VECTOR CONSTANT
	BDR, R3	CXVTEXT	17@	TEXT VECTOR CONSTANT
	BDR, R3	CXVTEXTL	18@	LONG TEXT VECTOR CONSTANT
	B	CXLSERR	19@	ERROR (DURING LINE SCAN)
	BDR, R3	CXCOMMNT	20@	COMMENT
	BDR, R3	CXSTORTR	21@	STOPNAME
	BDR, R3	CXSTORTR	22@	TRACENAME
	B	CXNAME	23@	NAME
	BAL, R15	CXBADCD	24@	*
	B	CXQUAD	25@	QUAD-0
	B	CXQUAD	26@	QUAD-1
	B	CXQUAD	27@	QUAD-2
	B	CXQUAD	28@	QUAD-3
	B	CXQUAD	29@	QUAD-4
	B	CXQUAD	30@	QUAD-5
	B	CXQUAD	31@	QUAD-6
	B	CXQUAD	32@	QUAD-7
	B	CXQUAD	33@	QUAD-8
	B	CXQUAD	34@	QUAD-9
	B	CXQUAD	35@	QUAD
	B	CXQUAD	36@	QUOTE-QUAD
	B	CXBOS	37@	BEGINNING-OF-STMT
	B	CXSEMI	38@	SEMICOLON
	B	CXLB	39@	LEFT BRACKET
	B	CXRB	40@	RIGHT BRACKET
	B	CXLP	41@	LEFT PAREN
	B	CXRP	42@	RIGHT PAREN

B	CXBRN	43@	BRANCH ARROW	
B	CXASS	44@	ASSIGN ARROW	
B	CXDOT	45@	DOT	
BAL, R15	CXBADCD	46@	SMALL CIRCLE	
BAL, R15	CXBADCD	47@	*	
BAL, R15	CXBADCD	48@	*	
BAL, R15	CXBADCD	49@	*	
B	CXOPM	50@	DOMINO	MONADIC USE
B	CXOPM	51@	ROLL	MONADIC USE
B	CXOPM	52@	T-BAR	MONADIC USE
B	CXRED1	53@	REDUCE ON 1ST COORD	MONADIC USE
B	CXCRED	54@	REDUCE	MONADIC USE
B	CXOPM	55@	IOTA	MONADIC USE
B	CXOPM	56@	RHO	MONADIC USE
B	CXOPM	57@	RAVEL	MONADIC USE
B	CXOPM1	58@	REVERSE ON 1ST COORD	MONADIC USE
B	CXCOPM	59@	REVERSE	MONADIC USE
B	CXOPM	60@	TRANSPOSE	MONADIC USE
B	CXOPM	61@	IDENTITY	MONADIC USE
B	CXOPM	62@	MINUS	MONADIC USE
B	CXOPM	63@	SIGNUM	MONADIC USE
B	CXOPM	64@	RECIPROCAL	MONADIC USE
B	CXOPM	65@	EXPONENTIAL	MONADIC USE
B	CXOPM	66@	NATURAL LOG	MONADIC USE
B	CXOPM	67@	PI TIMES	MONADIC USE
B	CXOPM	68@	CEILING	MONADIC USE
B	CXOPM	69@	FLOOR	MONADIC USE
B	CXOPM	70@	ABSOLUTE	MONADIC USE
B	CXOPM	71@	FACTORIAL (GAMMA)	MONADIC USE
BAL, R15	CXBADCD	72@	*	
BAL, R15	CXBADCD	73@	*	
B	CXOPM	74@	I-BEAM	
B	CXOPM	75@	GRADE UP	
B	CXOPM	76@	GRADE DOWN	
B	CXOPM	77@	NOT	
BAL, R15	CXBADCD	78@	*	
BAL, R15	CXBADCD	79@	*	
B	CXOP	80@	DOMINO	
B	CXOP	81@	ROLL OR DEAL	
B	CXOP	82@	T-BAR	
B	CXOP1	83@	COMPRESS ON 1ST COORD	
B	CXCOP	84@	COMPRESS	
B	CXOP	85@	IOTA	
B	CXOP	86@	RHO	
B	CXCOP	87@	COMMA	
B	CXOP1	88@	REVERSE OR ROTATE ON 1ST COORD	
B	CXCOP	89@	REVERSE OR ROTATE	
B	CXOP	90@	TRANSPOSE	
B	CXOP	91@	PLUS	
B	CXOP	92@	MINUS	
B	CXOP	93@	TIMES	
B	CXOP	94@	DIVIDE	
B	CXOP	95@	EXPONENT	
B	CXOP	96@	LOG	
B	CXOP	97@	CIRCLE	
B	CXOP	98@	MAX OR CEILING	
B	CXOP	99@	MIN OR FLOOR	
B	CXOP	100@	ABSOLUTE OR RESIDUE	
B	CXOP	101@	EXCLAMATION PT	
B	CXOP	102@	LESS THAN	
B	CXOP	103@	LESS THAN OR EQUAL	
B	CXOP	104@	GREATER THAN	
B	CXOP	105@	GREATER THAN OR EQUAL	
B	CXOP	106@	NOT EQUAL	
B	CXOP	107@	EQUAL	
B	CXOP	108@	AND	
B	CXOP	109@	OR	
B	CXOP	110@	NAND	
B	CXOP	111@	NOR	
BAL, R15	CXBADCD	112@	*	
BAL, R15	CXBADCD	113@	*	
B	CXOP	114@	DECODE	
B	CXOP	115@	ENCODE	
B	CXOP	116@	TAKE	
B	CXOP	117@	DROP	
B	CXOP1	118@	EXPAND OR SCAN ON 1ST COORD	U19-0034
B	CXCOP	119@	EXPAND OR SCAN	U19-0035
B	CXOP	120@	EPSILON	

	BAL, R15	CXBADCD	121@ *
	BAL, R15	CXBADCD	122@ *
	BAL, R15	CXBADCD	123@ DIERESIS
	BAL, R15	CXBADCD	124@ NEGATIVE SIGN
	BAL, R15	CXBADCD	125@ UNDERSCORE
	BAL, R15	CXBADCD	126@ DOLLAR SIGN
	BAL, R15	CXBADCD	127@ ALPHA
	BAL, R15	CXBADCD	128@ OMEGA
	BAL, R15	CXBADCD	129@ DEL
	BAL, R15	CXBADCD	130@ LOCKED DEL
	BAL, R15	CXBADCD	131@ LEFT CUP
	BAL, R15	CXBADCD	132@ RIGHT CUP
	BAL, R15	CXBADCD	133@ CAP
	BAL, R15	CXBADCD	134@ CUP
	BAL, R15	CXBADCD	135@ COLON
	BAL, R15	CXBADCD	136@ *
	BAL, R15	CXBADCD	137@ *
NXTSCANU	AI, R3	-1	138@ DUMMY -- PT AT NEXT CODESTR.DESIG.
NXTSCAN	LB, R6	*CURRCS, R3	GET CODESTRING DESIGNATOR.
	B	CXV, R6	VECTOR TO THAT CD'S PRELIM PROCESSOR
CX	LH, R3	*CURRCS	OFFSET TO RIGHTMOST ITEM OF CODESTR.
	LW, R1	TOPOSTAK	PT AT TOP OF EXECUTION STACK.
	LB, R2	*TOPOSTAK	GET TOP CATEGORY.
	CI, R2	CATQ	CK FOR Q-ENTRY (EVALUATED-INPUT REQ)
	BE	NXTSCAN	YES, START SCANNING.
	LW, R12	SENTRY	NO, PUSH AN S-ENTRY (COUNT = 0)
	LI, R2	CATS	SET TOP CAT TO S-ENTRY.
	LW, R8	OBSFLAG	SET OBSERVATION SETTING FOR THIS
	STW, R8	OBSERVE	DIRECT LINE: 0=NO & -1=YES.
	B	CXPUSHC	PUSH CATEGORY WD AND START SCAN.
PAGE			

SPACE 2			
Z	SET	%-CODEXEQ@	
SPACE			
Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536	SIZE IN DECIMAL.
SPACE 2			
END			

TITLE 'APLFIO-B00,10/16/73,DWG702985'
 SYSTEM SIG7F
 SYSTEM BPM
 PAGE

*
 * REF'S
 *

REF	ACQCC	ACQUIRE CHARACTER AND CODE	
REF	ACQIT	ACQUIRE ALPHANUMERIC ITEM	U20-0004
REF	ACQNB	ACQUIRE NON-BLANK CHAR AND CODE	U20-0005
REF	ACQNXCC	ACQ. NEXT CHAR. AND CODE	
REF	ACQNXNB	ACQUIRE NEXT NON-BLANK CHAR.	U20-0007
REF	ALOCBLK	ALLOCATE DATA BLOCK	
REF	ALOCHNW	DATA BLOCK ALLOCATION ROUTINE	
REF	BITPOS	BIT MASK TABLE	
REF	BLANKS	WORD OF BLANKS	
REF	BREAKFLG	BREAK FLAG	
REF	CKVDB	CHECK DATA BLOCK VALIDITY	U20-0009
REF	CLOSР	CLOSE AND RELEASE DCB	
REF	CLOSV	CLOSE AND SAVE DCB	
REF	CONSTBUF	CONSTANT BUFFER	
REF	CURRKEYT	CURRENT KEY IN FILE	
REF	DREF	DEREFERENCE DATA BLOCK	
REF	DXRETURN	DYADIC EXECUTION RETURN	
REF	ERDOMAIN	DOMAIN ERROR	
REF	ERLENGTH	LENGTH ERROR	
REF	ERRANK	RANK ERROR	
REF	F:TF	DCB	
REF	FFFFFFFE	MASK	
REF	FIOABNT	FIO INABN MODE FLAG	
REF	FIOACCC	ACCOUNT CONTROL WORD IN FPT	
REF	FIOACCT	ACCOUNT IN FIO FPT	
REF	FIOBUF	ADDRESS OF FIO BUFFER	
REF	FIODCB	FIO DCB ADDRESS	
REF	FIODCBNO	DCB #	
REF	FIODCBT	TABLE OF FIO DCB ADDRESSES	
REF	FIOТIE	TABLE OF FILE TIE NUMBERS	U20-0011
REF	FIOKEY	KEY IN USE	
REF	FIOMODE	MODE CONTROL	
REF	FIONAME		
REF	FIOASC	FIO PASSWORD CONTROL	
REF	FIOPASS	FIO PASSWORD	
REF	FIOREADC	READ CONTROL	
REF	FIOSIZ	FIO REC SIZE	
REF	FIOWRITC	WRITE CONTROL	
REF	FPTOPFIO	FPT TO OPEN FIO FILE	
REF	FPTOPNXT	OPEN TO NEXT FILE IN ACCT	
REF	FPTOP1ST	OPEN TO FIRST FILE IN ACCT	
REF	FPTRD1ST	READ 1ST RECORD OF FILE-FOR ID	
REF	FRSTKEYT	FIRST KEY IN FILE	
REF	GARBCOLL	GARBAGE COLLECT	
REF	GETTIME	CAL TO GET TIME-DATE	
REF	GIVEBACK	GIVE BACK UNUSED MEMORY IN DB	U20-0013
REF	IDBUF	FIO ID REC. BUFFER	
REF	IV1	REDUCE RTARG TO SCALAR INT. OR QUIT	
REF	J:ACCN	USER ACCOUNT	
REF	LASTKEYT	LAST KEY IN FILE	
REF	LFARG	LEFT ARG ADDRESS	
REF	NAMEBUF	NAME BUFFER	U20-0015
REF	NONAME	TEST FOR LETTER-DIGIT	
REF	NUMFILES	NO OF FIO CHANNELS	
REF	OP1STACC	ACCOUNT IN FPTOP1ST	
REF	OP1STACT	ACCT CONTROL IN FPTOP1ST	
REF	RESULT	RESULT ADDRESS	
REF	RTARG	RIGHT ARG ADDRESS	
REF	XFFFF	HW MASK	
REF	X1FFFF	ADDRESS MASK	
REF	ZEROZERO	0,0 DW BOUND	
REF	FIOSN	SERIAL NO. ENTRIES IN FPT	U20-0017
REF	FIOSNC	SERIAL NO. CONTROL WORD IN FPT	U20-0018
REF	WHATERR	HOLDS INTR. I.D. 0=14T1 & NZ=14T2.	
REF	IOERCODE	I/O ERR OR ABN WD.	
REF	ERFILEIO	FILE I/O ERROR PROCESSOR.	
REF	IDFILSPC, IDFIOERR, IDFILDAM, IDFILNAM, IDNOTAPL, IDFTFULL		
REF	IDFILACC, IDFILTIE, IDNOPACK, IDFILIDX, IDFILBSY		
	PAGE		

*
 * DEF'S

```

*
      DEF      APLFIO@      START OF PROCEDURE
      DEF      ERRFTFIO     FIO ERR ON F:TF DCB
      DEF      FILEOPS      ENTRY POINT
      DEF      FIOERR       FIO MONITOR ERROR
      PAGE

*
*  STANDARD EQU'S
*  REGISTERS
*
R0      EQU      0
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
NFIOPS  EQU      27          NO. OF FILE I/O OPERATIONS
SAVER4   EQU      CONSTBUF+12  TEMP FOR R4
TYPEINTG EQU      3          INTEGER DATA BLOCK TYPE
      PAGE
*
*
*  MODULE DESCRIPTION:
*  THIS MODULE CONTAINS THE PROCEDURE FOR THE APL FIO SUBSYSTEM
*  THE ENTRY POINT IS FILEOPS, WHICH IS REACHED BY THE
*  DYADIC INTRINSIC OPERATOR FOR FILE I/O
*
*  THIS MODULE IS PRIMARILY A UTS INTERFACE MODULE, BUT HAS BEEN
*  SEPARATED FROM APLUTSI TO ALLOW OVERLAY OF APLFIO
*
*  THE CONTEXT USED BY APLFIO IS PRIMARILY IN THE CONTEXT
*  REGION OF APLUTSI
*
*  EXITS FROM APLFIO ARE TO VARIOUS EXECUTION ERROR ROUTINES,
*  SUCH AS ERDOMAIN, ERLLENGTH, ETC OR TO DXRETURN
*
*  FILEOPS-PRECHECKS THE ARGUMENTS AND, IF NO ERROR IS INDICATED
*  BRANCHES TO ONE OF 27 FILE I/O PRIMITIVE ROUTINES
*
*  FI01- SET A DCB # RANGE 1 TO NUMFILES
*  FI02- SET FILE NAME
*  FI03- SET OR RESET ACCOUNT
*  FI04- SET OR RESET PASSWORD
*  FI05- OPEN A DCB (SET MODE-READ CONTROL-WRITE CONTROL-INABN FLAG)
*  FI06- CLOSE AND SAVE INDICATED DCG
*  FI07- CLOSE AND RELEASE INDICATED DCB
*  FI08- GET FIRST, CURRENT, OR LAST KEY VALUE FOR DCB
*  FI09- SET CURRENT KEY (AND POSSIBLY LAST KEY) FOR DCB
*  FI010-WRITE A DATA RECORD
*  FI011-WRITE AN ID RECORD AND A DATA RECORD
*  FI012-READ A DATA RECORD USING CURRENTKEY
*  FI013-READ AN ID RECORD AND DATA RECORD-KEYED OR SEQUENTIALLY
*  FI014-READ AN ID RECORD ONLY-KEYED OR SEQUENTIALLY
*  FI015-DELETE RECORD WITH GIVEN KEY
*  FI016-DELETE ID AND DATA RECORDS-ID REC MATCHES GIVEN KEY
*  FI017-SKIP RECORD OR READ RECORD SEQUENTIALLY
*  FI018-GENERATE TEXT ARRAY OF FIO FILES IN INDICATED ACCT.
*  FI019-GENERATE TEXT ARRAY OF OPEN FILES OR NUMERIC VECTOR OF OPEN #'S
*  FI020-SET OR RESET SERIAL NO. FOR PRIVATE PACK
*  FI021-SET FILE ID-SINGLE PRIMITIVE FOR NAME, ACCT, PASSWORD
*  FI022-WRITE A NON-APL DATA RECORD
*  FI023-READ A NON-APL DATA RECORD USING CURRENT KEY
*  FI024-READ A NON-APL DATA RECORD SEQUENTIALLY
*  FI025-CONVERT TEXT VECTOR TO LOGIC VECTOR
*  FI026-CONVERT TEXT VECTOR TO INTEGER VECTOR
*  FI027-CONVERT TEXT VECTOR TO REAL VECTOR

```

U20-0021

U20-0025

U20-0026

```

*
APLFIO@ PAGE
CSECT 1
BOUND 8
*
* CONSTANTS
*
YEARANGE DATA 4718765,5046272 6172-1/77 (CHANGE LATER)
NONEALL TEXT 'NONEALL '
FIOMBSN DATA X'0C000000'+BA(FIONAME)
FIOMBSN1 DATA X'0B000001'+BA(FIONAME)
FIOIDHDR DATA X'03010008' INTEGER VECTOR-8 WORD DATA BLOCK U20-0028
FIOTXMAT DATA X'02020004'
X1F DATA X'1F' MASK
X3F DATA X'3F' MASK
XFFFFFFF DATA X'00FFFFFF' MASK TO DELETE BYTE 0
KEYBYTE DATA X'03000000'
MAXKEY DATA 9999998 MAXIMUM KEY VALUE
PAGE
*
* FILEOPS-CHECKS ARGUMENTS-ERROR EXITS OR BRANCHES TO 1 OF 19 ROUTINES
*
FILEOPS STW,R6 WHATERR SAVE INTRINSIC IDENTIFIER.
LH,R5 *LFARG CHECK LEFT ARGUMENT.
LW,R4 LFARG
LW,R4 2,R4 ASSUME SCALAR-CHECK VALUE
BLZ FILEOP1 NEGATIVE-CHK FOR LOGIC 1 SCALAR
BEZ ERDOMAIN ZERO-NOT VALID IN ANY CASE
CI,R5 X'300' POSITIVE-VERIFY INTEGER SCALAR
BNE ERDOMAIN NO-ERROR
CI,R4 NFIOPS RANGE TEST
BLE FILEOP2 OK
B ERDOMAIN NO-ERROR
FILEOP1 CI,R5 X'100' CHECK-TYPE, LOGIC-RANK, 0
BNE ERDOMAIN NO
LI,R4 1 YES-VALUE IS 1
FILEOP2 LB,R5 *RTARG CHECK THE RIGHT ARGUMENT
CI,R5 5 CHECK IF RTARG IS DATA TYPE
BG ERDOMAIN NO-ERROR
LW,R15 FIOPSTBL,R4 GET VALIDITY MASK AND BRANCH ADDRESS
LH,R7 R15
BGEZ FILEOPS1 NO TEST NEEDED FOR FILE-OPEN
LW,R8 *FIODCB TEST IF FILE IS OPEN
CW,R8 BITPOS-10
BANZ FILEOPS1 YES-OK
FIOCLSD LW,R12 R4 NO-ERROR EXIT
B ERRAPL
FILEOPS1 CI,R7 X'4000' CHECK IF WRITE OPERATION
BANZ *R15 YES-ANY RTARG IS VALID
CI,R5 2 NO-CHECK DATA TYPE
BE FILEOPS2 TEXT
CI,R7 X'2000' NUMERIC-CHECK IF PERMITTED
BAZ ERDOMAIN NO
STW,R4 SAVER4 YES-SAVE R4
BAL,R14 IV1 GET INTEGER OR ERROR EXIT
LW,R4 SAVER4
AI,R7 0 CHECK VALUE
BLZ ERDOMAIN ERROR ON NEGATIVE RTARG
B *R15 OK
FILEOPS2 CI,R7 X'1000' CHECK FOR CONVERTER PRIMITIVE
BANZ FIO25 YES-USE FIO25 FOR ALL 3
CI,R7 X'0800' CHECK IF TEXT PERMITTED
BAZ ERDOMAIN NO
LH,R5 *RTARG YES-GET TYPE AND RANK
LW,R6 RTARG
AI,R5 -X'201' CHECK RANK
BEZ FILEOPS3 VECTOR
BGZ ERRANK ARRAY
LI,R8 1 SCALAR-SET LENGTH 1
LW,R9 2,R6
STD,R8 CONSTBUF SET LENGTH AND BYTE
B FILEOPS4
FILEOPS3 LW,R8 2,R6 GET LENGTH OV VECTOR
BLEZ ERLNGTH TOO SMALL
LI,R7 BA(CONSTBUF+1)
CI,R8 40
BG ERLNGTH TOO BIG
STB,R8 R7 SET BYTE COUNT FOR MBS

```

STW,R8	CONSTBUF	SET BYTE COUNT FOR TEXTC
AI,R6	3	WA(TEXT STRING)
SLS,R6	2	BA(TEXT STRING)
MBS,R6	0	FORM TEXTC IN CONSTBUF
LW,R6	FIOPTSTBL,R4	GET MAX TEXT LENGTH
SLS,R6	-20	WHICH IS IN BYTES 5-11
AND,R6	X3F	MASK OFF OTHER BITS
CW,R8	R6	
BG	ERLENGTH	TOO LONG
FILEOPS4	LI,R7	SET TEXT FLAG
FIOPTSTBL	B	GO TO PROPER ROUTINE
	*R15	
GEN,1,1,1,1,1,1,7,20	0,0,1,0,0,0,FI01	TIE NUMBER-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,0,0,1,11,FI02	NAME-TEXT
GEN,1,1,1,1,1,1,7,20	0,0,1,0,1,8,FI03	ACCOUNT-TEXT OR 0
GEN,1,1,1,1,1,1,7,20	0,0,1,0,1,8,FI04	PASSWORD-TEXT OR 0
GEN,1,1,1,1,1,1,7,20	0,0,1,0,0,0,FI05	OPEN-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,1,0,0,0,FI06	CLOSE-SAVE-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,1,0,0,0,FI07	CLOSE-REL-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI08	GET KEY-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI09	SET KEY-INTEG
GEN,1,1,1,1,1,1,7,20	1,1,0,0,0,0,FI010	WRITE-ANY DATA
GEN,1,1,1,1,1,1,7,20	1,1,0,0,0,0,FI011	WRITE COMP-ANY DATA
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI012	READ-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI013	READ COMP-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI014	READ ID-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI015	DELREC-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI016	DELCOMP-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI017	SKIP OR READ-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,1,0,1,8,FI018	LIB-TEXT OR 0
GEN,1,1,1,1,1,1,7,20	0,0,1,0,0,0,FI019	FNAM-FNUM-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,1,0,1,12,FI020	PACK NO-TEXT OR 0
GEN,1,1,1,1,1,1,7,20	0,0,0,0,1,40,FI021	FID-TEXT
GEN,1,1,1,1,1,1,7,20	1,1,0,0,0,0,FI022	WRITE RAW-ANY DATA
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI023	READ RAW-INTEG
GEN,1,1,1,1,1,1,7,20	1,0,1,0,0,0,FI024	SEQ READ RAW-INTEG
GEN,1,1,1,1,1,1,7,20	0,0,0,1,0,0,FI025	CNVRT C TO L-TEXT
GEN,1,1,1,1,1,1,7,20	0,0,0,1,0,0,FI025	CNVRT C TO I-TEXT
GEN,1,1,1,1,1,1,7,20	0,0,0,1,0,0,FI025	CNVRT C TO R-TEXT

*

* THE FIELDS IN FIOPTSTBL, ABOVE, HAVE THE FOLLOWING MEANING

*

*	FIELD 1 (1 BIT)	1=TEST IF FILE OPEN
*	FIELD 2 (1 BIT)	1=WRITE OPERATION, ANY RTARG OK
*	FIELD 3 (1 BIT)	1=NUMERIC INTEGER OK FOR RTARG
*	FIELD 4 (1 BIT)	1='CONVERT' PRIMITIVE-RTARG MUST BE TEXT VECTOR
*	FIELD 5 (1 BIT)	1=RTARG MAY BE TEXT VECTOR OF SPECIFIED MAX LNTH
*	FIELD 6 (7 BITS)	MAX LENGTH IF FIELD 5=1
*	FIELD 7 (20 BITS)	ADDRESS OF PRIMITIVE ROUTINE

*

* FIO1-SET FIODCB AND FIODCBNO (INDICATES I/O STREAM FOR LATER ACTIONS)

*

FIO1	BLEZ	ERDOMAIN	STREAM NO. MUST BE POSITIVE	U20-0071
	LI,R3	NUMFILES	SET LOOP TO LOOK FOR EXISTING TIE	U20-0072
FIO1A	CW,R7	FIO1A	CHECK FOR MATCH	U20-0073
	BE	FIO1D	SCORE	U20-0074
	BDR,R3	FIO1A	NO-LOOP	U20-0075
	LI,R3	NUMFILES	SET LOOP TO LOOK FOR NEW SLOT	U20-0076
FIO1B	LW,R5	FIO1A	CHECK FOR UNUSED SLOT	U20-0077
	BEZ	FIO1C	FOUND--USE IT	U20-0078
	BDR,R3	FIO1B	LOOP	U20-0079
	LI,R12	3	TABLE FULL	U20-0080
	B	ERRAPL		
FIO1C	STW,R7	FIO1A	FILL NEW SLOT	U20-0082
FIO1D	STW,R3	FIODCBNO	SET DCB NO.	U20-0083
	LW,R5	FIODCBT-1,R3		U20-0084
	STW,R5	FIODCB	SET DCB ADDRESS	U20-0085
	B	FIOEX		

*

* FIO2-SET FILE NAME IN FIO OPEN FPT

*

FIO2	BGEZ	ERDOMAIN	NUMERIC-INVALID
	LI,R6	BA(CONSTBUF)+3	SET-UP
	LW,R7	FIOBMSN	MBS
	MBS,R6	0	FORM NAME IN TEXTC
	B	FIOEX	

*

* FIO3-SET(OR RESET) ACCOUNT IN FIO FPT

```

*
FI03      BLZ      FI03A      TEXT-SET ACCT
          BGZ      ERDOMAIN    ERROR
          LI,R4    BA(FIOACCC)+2 RESET ACCOUNT
          B        FIO3OR4R    (R7=0)
FI03A     LI,R3    FIOACCT      ACCT ADDRESS
          LI,R4    BA(FIOACCC)+2 ACCT CONTROL (BA)
          LI,R7    BA(FIOACCT)  ACCT ADDRESS (BA)-FOR MBS
FI03OR4S  LW,R9    CONSTBUF     BYTE COUNT
          STB,R9   R7           SET MBS COUNT
          LW,R5    BLANKS
          STD,R5   *R3          PRE-BLANK ACCT OR PASSWORD
          LI,R6    BA(CONSTBUF+1) SET UPMBS
          MBS,R6   0           SET ACCT OR PASS
          LI,R7    2
FI03OR4R  STB,R7   0,R4        SET OR RESET ACCT OR PASS CONTROL
          B        FIOEX       EXIT
*
* FI04-SET (OR RESET) PASSWORD IN FIO FPT
*
FI04      BLZ      FI04A      TEXT-SET PASS
          BGZ      ERDOMAIN    ERROR
          LI,R4    BA(FIOPASC)+2 0-RESET PASS
          B        FIO3OR4R
FI04A     LI,R3    FIOPASS      PASS ADDRESS
          LI,R4    BA(FIOPASC)+2 PASS CONTROL (BA)
          LI,R7    BA(FIOPASS)   PASS ADDRESS (BA)-FOR MBS
          B        FIO3OR4S     SET PASSWORD
*
* FI05-SET FIO MODE,READ AND WRITE CONTROL AND FIOABNT FLAG-
* OPEN CURRENT I/O CHANNEL IN INDICATED MODE
*
* ERROR IN CAL---- EXITS TO ERRFF FOR PROCESSING
*
* ON SUCCESS,OUT OR OUTIN, SET FIRST,CURRENT, AND LAST KEYS=1000
* IN OR INOUT,GET FIRST KEY FROM DCB,POSITION FILE
* TO END,GET LAST KEY FROM DCB,REPOSITION
* TO BEGINNING,SET CURRENT KEY=FIRST
*
FI05      CI,R7    104          64+32+8
          BG       ERDOMAIN    UNGOOD
          LW,R5    FIODCBNO     CHECK IF TIE NO. EXISTS          U20-0087
          LW,R5    FIOTIE-1,R5
          BLEZ     FIOCLSD      ERROR EXIT IF NOT          U20-0088
          LD,R4    ZEROZERO     PRESET TO 'NONE'          U20-0089
          STW,R4   FIOABNT      RESET FIOABN FLAG
          CI,R7    64           CHECK WRITE CONTROL
          BAZ      FIO5A
          LI,R4    1            'ALL'
FI05A     CI,R7    32           CHECK READ CONTROL
          BAZ      FIO5B
          LI,R5    1            'ALL'
FI05B     LW,R6    NONEALL,R4
          STW,R6   FIOWRITC     SET WRITE CONTROL
          LW,R6    NONEALL,R5
          STW,R6   FIOREADC     SET READ CONTROL
          AND,R7   X1F          MASK OFF READ-WRITE
          BEZ      ERDOMAIN     NO IO MODE
          CI,R7    17           CHECK FOR INABN
          BNE      FIO5C        NO
          LI,R7    1            YES-SET MODE TO IN
          STW,R7   FIOABNT      AND SET INABN FLAG
          B        FIOSMODE
FI05C     CI,R7    8            CHECK I/O MODE
          BE       FIOSMODE     OUTIN
          CI,R7    4            INOUT
          BE       FIOSMODE
          CI,R7    2            IN OR OUT
          BLE      FIOSMODE     INCONSISTENT I/O MODE
          B        ERDOMAIN
FIOSMODE  STW,R7   FIOMODE
CALOPFIO  CAL1,1  FPTOPFIO     OPEN DCB
          LW,R6    FIOABNT      CHECK IF INABN
          BNEZ     FIOINABN     YES-ERROR
          LW,R5    FIODCBNO
          CI,R7    10           CHECK IF OUT OR OUTIN
          BAZ      FIOINOK      NO-MUST BE IN OR INOUT
FIOMPTF   LW,R6    MAXKEY      SET FIRST KEY FOR EMPTY FILE          20-00002

```

	STW,R6	FRSTKEYT-1,R5		
	LI,R6	1	SET CURR.-LAST KEYS FOR EMPTY FILE	20-00004
	STW,R6	LASTKEYT-1,R5		
	STW,R6	CURRKEYT-1,R5		
	B	FIOEX		
FI0INABN	LI,R12	0	ERROR TYPE=0 (OLD FILE)	
	LW,R5	FIODCB	DCB ADDRESS	
	LI,R6	ERRAPL	ERROR EXIT	
	B	CLOSV	CLOSE AND SAVE	
FI0INOK	RES	0		
CALRDFII	CAL1,1	FPTRDFIS	READ 1ST RECORD	
	BAL,R8	FIOSETKY	GET KEY FROM DCB	
	STW,R4	FRSTKEYT-1,R5	SET FIRST KEY	
CALFPFEX	CAL1,1	FPTFPFE	POSITION TO END OF FILE	
	BAL,R8	FIOSETKY	GET KEY FROM DCB	
	STW,R4	LASTKEYT-1,R5	SET LAST KEY	
	LW,R4	FRSTKEYT-1,R5	SET CURRENT	
	STW,R4	CURRKEYT-1,R5	KEY=FIRST KEY	
CALFPFBX	CAL1,1	FPTFPFB	POSITION TO BEGINNING OF FILE	
	B	FIOEX		
*				
* FI06-CLOSE AND SAVE FILE FOR INDICATED DCB #				
*				
FI06	BAL,R6	FIONUMCK	GET DCB NO. (OR ERROR EXIT)	U20-0091
	LI,R6	FIOEX	SET EXIT	
	B	CLOSV	CLOSE AND SAVE	
*				
* FI07-CLOSE AND RELEASE FILE FOR INDICATED DCB #				
*				
FI07	BAL,R6	FIONUMCK	GET DCB NO. (OR ERROR EXIT)	U20-0093
	LI,R6	X'FF00'	SET MASK	U20-0094
	AND,R6	26,R5	CHECK IF ACCT SET IN DCB	U20-0095
	BEZ	FI07A	NO-OK	U20-0096
	LW,R6	J:ACCN	YES-CHECK IF USERS ACCT	U20-0097
	LW,R7	J:ACCN+1		U20-0098
	LW,R8	27,R5	(ACCT IN DCB)	U20-0099
	LW,R9	28,R5		U20-0100
	CD,R6	R8		U20-0101
	BE	FI07A	OK	U20-0102
	BAL,R6	CLOSV	NOT USERS ACCT,CLOSE AND SAVE	U20-0103
	LI,R12	4	SET ERROR VALUE	U20-0104
	B	ERRAPL		
FI07A	LI,R6	FIOEX	SET EXIT	U20-0106
	B	CLOSR	CLOSE AND RELEASE	
*				
* FIONUMCK-SEARCH DCB TABLE FOR TIE NO. IN R7-ERROR EXIT IF NOT FOUND				
* USED IN 'CLOSE' OPERATIONS ONLY-'UNTIES' FILE				
* IF FOUND,SET R5 TO ADDRESS OF DCB				
* R3 IS TABLE INDEX				
* R6 IS LINK				
* TABLE VALUE IS ZEROED				
*				
FIONUMCK	BLEZ	ERDOMAIN	OUT OF RANGE	
	LI,R3	NUMFILES	SET LOOP	U20-0116
FIONUMC1	CW,R7	FIOTIE-1,R3	CHECK FOR TIE NUMBER	U20-0117
	BE	FIONUMC2	FOUND	U20-0118
	BDR,R3	FIONUMC1	LOOP	U20-0119
	B	FI0CLSD	NOT FOUND-NO OPEN FILE	U20-0120
FIONUMC2	LI,R5	0		U20-0121
	STW,R5	FIOTIE-1,R3	ZERO TABLE ENTRY (FOR 'UNTIE')	U20-0122
	LW,R5	FIODCBT-1,R3	GET DCB ADDRESS (FOR CLOSE)	U20-0123
	B	0,R6	RETURN	U20-0124
*				
* FI08-GETS VALUE OF FIRST,CURRENT,OR LAST KEY FOR DCB				
*				
FI08	BLEZ	ERDOMAIN	OUT OF RANGE	
	CI,R7	3		
	BG	ERDOMAIN	OUT OF RANGE	
	LW,R7	WHICHKEY-1,R7	GET KEY TABLE ADDRESS	
	AW,R7	FIODCBNO	OFFSET BY DCB #	
	LW,R12	0,R7	GET KEY VALUE	
	B	GENSCLR	GEN. SCALAR INTG DATA BLK & EXIT.	
WHICHKEY	DATA	FRSTKEYT-1		
	DATA	CURRKEYT-1		
	DATA	LASTKEYT-1		
*				
* FI09-SET CURRENT KEY FOR DCB				


```

*
FI09      BLEZ      ERDOMAIN      OUT OF RANGE
          BAL,R4    FIOKEYSZ      CHECK IF IN KEY RANGE
          LW,R6     FIODCBNO      GET DCB #
          STW,R7    CURRKEYT-1,R6 SET CURRENT KEY
          B         FIOEX

*
* FI010-WRITE A DATA RECORD WITH CURRENT KEY AND DCBNO
*      USES FIORECSU,IN FI011, AND GOES TO CALWRFD IN FI011
*      TO WRITE THE RECORD
*
FI010     BAL,R6    FIORECSU      SET UP TO WRITE RECORD
          B         CALWRFD

*
* FI011-WRITE AN FIO ID RECORD AND A DATA RECORD
*      THIS MAKES UP ONE FIO-'COMPONENT'
*
*      ERROR IN CAL---- EXITS TO ERRFF FOR PROCESSING
*
FI011     BAL,R7    GETTIME       LEAVES TIME IN R8-R9  USES R6-R10
          BAL,R6    FIORECSU      SET UP TO WRITE RECORDS-SIZE IN R7
          LW,R10    J:ACCN        ACCOUNT IN R10-R11
          LW,R11    J:ACCN+1
          LCI       5
          STM,R7    FIDBUF        FORM ID RECORD
CALWRFI    CAL1,1   FPTWRFI       WRITE IT
          BAL,R8    FIOSETKY      FIRST KEY MAY NEED UPDATE      20-00009
          MTW,1     FIOKEY        UPDATE KEYS
CALWRFD    CAL1,1   FPTWRFD       WRITE DATA RECORD
          BAL,R8    FIOSETKY      UPDATE CURRENT (AND MAYBE LAST)KEY
          B         FIOEX

*
* FIORECSU-ROUTINE TO SETUP FOR WRITE OF DATA RECORD
*      GETS ADDRESS AND SIZE OF RTARG-CONTINUE AT-
* FIORCSUA-SECOND ENTRY POINT
*      SETS RECORD SIZE IN BYTES
*      SETS KEY
*      R6=LINK R4,R5,R7 USED
*
FIORECSU  LW,R5     RTARG         SET-UP TO WRITE FIO RECORD
          STW,R5    FIOBUF        ADDRESS
          LW,R7     *RTARG
FIORCSUA  AND,R7    XFFFF
          SLS,R7    2
FIORCSUB  STW,R7    FIOSIZ        SIZE IN BYTES
          LW,R5     FIODCBNO      DCB #
          LW,R4     CURRKEYT-1,R5
          OR,R4     KEYBYTE
          STW,R4    FIOKEY        KEY
          B         0,R6

*
* FI012-READ A DATA RECORD  USING  CURRENT KEY
*
*      ALSO ENTERED AT FI014D FROM FI013-FI014(AFTER READING ID RECORD)
*
*      INCLUDES ROUTINES:
*      FIOSETRD-SETUP FOR READ-(CALLED FROM FI017)
*          *      R8=LINK
*          *      R4,R5,R6,R7,R10,R11 USED
*          *      CALLS ALOCBLK AND FIORCSUA
*      FIOENDRD-SET UP RESULT AFTER READ-(CALLED FROM FI017)
*          *      R8=LINK
*          *      R4,R5,R7 USED
*
*      ERROR IN CALRDFDK EXITS TO ERRFF FOR PROCESSING
*
FI014D    MTW,1     CURRKEYT-1,R5 SET KEY TO READ DATA RECORD
          LW,R7     FIDBUF
FI012     BLEZ      ERDOMAIN      NO DICE
          LI,R8     CALRDFDK      SET EXIT FROM FIOSETRD
FIOSETRD  AI,R7     3             ADD 3 BYTES FOR WORD BOUND      20-00012
          SLS,R7    -2           SET WORD SIZE                  20-00013
          LW,R11    R7           20-00014
          AI,R11    1
          AND,R11   FFFFFFFF     INSURE EVEN VALUE FOR DATA BLOCK
          STW,R11   CONSTBUF     SAVE SIZE IN WORDS
          BAL,R7    ALOCBLK      ALLOCATE DATA BLOCK
          STW,R4    RESULT       SET RESULT

```

	STW, R4	FIOBUF	SET READ ADDRESS	
	LW, R7	CONSTBUF	RESTORE SIZE IN WORDS	
	BAL, R6	FIOCSUA	SET SIZE IN BYTES AND KEY	
	B	*R8	RETURN	
CALRDFDK	CAL1, 1	FPTRDFDK	READ RECORD	
	LI, R8	DXRETURN	SET EXIT FROM FIOENDRD	
FIOENDRD	LH, R6	*RESULT	SAVE TYPE-RANK IN R6	U20-0126
	LW, R4	CONSTBUF	ALLOCATED SIZE, TEMP TYPE-RANK=0	U20-0127
	LI, R5	1	REF COUNT=1	U20-0128
	STD, R4	*RESULT		U20-0129
	LI, R7	13		U20-0130
	LW, R11	*FIODCB, R7	GET ACTUAL RECORD SIZE	U20-0131
	AI, R11	7		U20-0132
	SLS, R11	-2		U20-0133
	AND, R11	FFFFFFFF	NO. OF WORDS, ROUNDED TO EVEN NO.	U20-0134
	XW, R11	CONSTBUF	SWITCH WITH ALLOCATED SIZE	U20-0135
	SW, R11	CONSTBUF	ALLOCATED SIZE-NEEDED SIZE	U20-0136
	LW, R4	RESULT	DB POINTER	U20-0137
	BAL, R7	GIVEBACK	RETURN UNUSED MEMORY	U20-0138
	STH, R6	*RESULT	RESTORE ORIGINAL TYPE-RANK	U20-0139
	SLS, R6	-8	R6=TYPE	U20-0140
	LW, R4	RESULT		U20-0141
	LH, R7	*RESULT	CHECK FOR SCALAR INTEGER	20-00045
	CI, R7	X'0300'		20-00046
	BNE	FIOVCK	NO	20-00047
	STB, R5	*RESULT, R5	YES, SET TO VECTOR	20-00048
	XW, R5	2, R4	SET LENGTH TO ONE	20-00049
	STW, R5	3, R4	AND MOVE VALUE DOWN	20-00050
FIOVCK	BAL, R7	CKVDB	CHECK VALIDITY OF DATA BLOCK	U20-0143
	B	ERRBADDB	NO GOOD	U20-0144
	B	*R8	EXIT	
*				
* FIO13-READ AN ID RECORD AND DATA RECORD-KEYED OR SEQUENTIALLY				
* FIO14-READ AN ID RECORD ONLY-KEYED OR SEQUENTIALLY				
*				
* COMMON CODE USED TO READ THE ID RECORD FOR EITHER ENTRY				
*				
* ERROR IN CAL---- EXITS TO ERRFF FOR PROCESSING				
*				
* IF SEQUENTIAL, FIO14C DOES THE READ				
*				
* FIOSETKY-GETS KEY FROM DCB, SETS AS CURRENT KEY				
* (CALLED FROM FIO5 AND FIO17)				
* R8=LINK				
* R4, R5 USED				
*				
* FIO14B-CHECKS ID RECORD FOR APPARENT VALIDITY				
* -SIZE =20 BYTES				
* -TIME WORD IN REASONABLE RANGE				
* DAY #173 OF YEAR #72 TO DAY #0 OF YEAR #77				
* THIS TEST SHOULD ELIMINATE ESSENTIALLY ALL FALSE FIO ID RECRDS				
*				
* FIO14B THEN CHECKS IF ENTRY WAS FOR ID RECORD ONLY OR 'COMPONENT'				
* IF COMPONENT, GOES TO FIO14D				
* IF ID RECORD ONLY, FORMS RESULT AND EXITS				
*				
FI013	RES	0		
FI014	BEZ	FIO14C	SEQUENTIAL READ	
	BAL, R4	FIOKEYSZ	RANGE CHECK KEY	
	LW, R5	FIODCBNO	GET DCB H	
	STW, R7	CURRKEYT-1, R5	SET CURR KEY	
	OR, R7	KEYBYTE		
	STW, R7	FIOKEY	AND FIOKEY	
CALRDFIK	CAL1, 1	FPTRDFIK	READ ID RECORD	
FI014B	LI, R4	8	CHECK ID RECORD FORMAT	
	LI, R12	1	ERROR CODE=0, SUBCODE=1	
	LH, R14	*FIODCB, R4		
	SLS, R14	-1	CHECK ARS	
	AI, R14	-20	IF NOT 20, ERROR EXIT	
	BNEZ	ERRAPL		
	LW, R14	FIDBUF+1	CHECK TIME WORD	
	CLM, R14	YEARANGE	IF NOT IN REASONABLE	
	BCS, 9	ERRAPL		
	LW, R14	SAVER4	CHECK IF	U20-0146
	CI, R14	13	ID ONLY	
	BEZ	FI014D	NO-PROCEED TO READ DATA RECORD	
	LI, R11	8	YES-CREATE	

	BAL, R7	ALOCBLK	TEXT DATA BLOCK	
	STW, R4	RESULT	FOR ID REC	
	LW, R5	FIOIDHDR	AS RESULT	
	LI, R6	1	MOVE DATA	REF COUNT=1
	LI, R7	5	AND HEADER	LENGTH=5
	LCI	5	TO ALLOCATED	
	LM, R8	FIDBUF	DATA BLOCK	
	LCI	8		
	STM, R5	*R4		
	B	DXRETURN		
*				
FI014C	RES	0		
CALRDFIS	CAL1, 1	FPTRDFIS	READ SEQUENTIAL-LOOK FOR ID REC.	
	LI, R8	FI014B	SET EXIT FROM FIOSETKY	
FIOSETKY	LW, R4	FIODCB		
	LW, R4	10, R4	GET KEY FROM DCB	
	LW, R4	*R4		
	AND, R4	XFFFFFFF	MASK OFF BYTE 0	
	LW, R5	FIODCBNO		
	STW, R4	CURRKEYT-1, R5	SET IN CURRKEYT	
	CW, R4	LASTKEYT-1, R5		
	BLE	FI0SK1		20-00016
	STW, R4	LASTKEYT-1, R5		
FI0SK1	CW, R4	FRSTKEYT-1, R5	CHECK FIRST KEY	20-00018
	BGE	*R8		20-00019
	STW, R4	FRSTKEYT-1, R5	SET NEW FIRST KEY	20-00020
	B	*R8		
*				
* FI015-DELETE A RECORD WITH INDICATED KEY				
*				
* ERROR IN CALDELRL EXITS TO ERRFF FOR PROCESSING				
*				
FI015	BAL, R4	FIOMODCK	CHECK FOR MODE AND KEY	U20-0148
	BAL, R4	FIOKEYSZ	CHECK RANGE	
	OR, R7	KEYBYTE		
	STW, R7	FIOKEY	SET KEY	
CALDELRL	CAL1, 1	FPTDELRL	DELETE RECORD	
CALFPFB	CAL1, 1	FPTFPFB	POSITION TO BEGINNING OF FILE	20-00022
	B	FIOINOK	GET NEW FIRST AND LAST KEYS	20-00023
*				
* FI016-DELETE 'COMPONEMT'-ID REC AND DATA REC WITH INDICATED KEY				
* TRIES TO DELETE DATA RECORD FIRST, THEN ID REC				
*				
* ERROR IN CALDELRL1 EXITS TO ERRFF FOR PEOCESSING				
*				
FI016	BAL, R4	FIOMODCK	CHECK FOR MODE AND KEY	U20-0150
	AI, R7	1	GET KEY OF DATA RECORD	
	BAL, R4	FIOKEYSZ	CHECK RANGE	
	OR, R7	KEYBYTE		
	STW, R7	FIOKEY	SET KEY	
CALDELRL1	CAL1, 1	FPTDELRL	DELETE DATA RECORD	
	MTW, -1	FIOKEY	BACK UP TO ID RECORD	
	B	CALDELRL	DELETE IT	
*				
* FIOMODCK-CHECK FOR NEGATIVE KEY OR NON-UPDATE MODE ON DELETE OPER.				
* R4=LINK, R5 AND R6 USED				
*				
FIOMODCK	BLEZ	ERDOMAIN	BAD KEY VALUE	U20-0152
	LI, R6	5		U20-0153
	LB, R5	*FIODCB, R6	GET MODE FROM DCB	U20-0154
	SLS, R5	-1		U20-0155
	AI, R5	-4		U20-0156
	BEZ	0, R4	UPDATE-RETURN	U20-0157
	LI, R12	20	OTHER MODE-SET ERROR CODE	U20-0158
	B	ERRAPL		U20-0159
*				
* FI017-READ DATA RECORD SEQUENTIALLY OR SKIP RECORD				
*				
* ERROR IN CALRDFDS EXITS TO ERRFF FOR PROCESSING				
*				
FI017	BGZ	FI017A		
CALFPR	CAL1, 1	FPTFPR	SKIP RECORD	
	BAL, R8	FIOSETKY	SET CURRENT KEY FROM DCB	
	B	FIOEX		
FI017A	BAL, R8	FIOSETRD	SET UP FOR SEQ. READ(+UNUSED KEY)	
CALRDFDS	CAL1, 1	FPTRDFDS	READ SEQUENTIAL DATA	
	BAL, R8	FIOENDRD	CLEAN UP ALLOCATED DATA BLOCK	
	BAL, R8	FIOSETKY	SET CURRENT KEY FROM DCB	

```

*           B           DXRETURN           RETURN
*
* FIO18-FORM TEXT ARRAY OF FILES IN INDICATED ACCOUNT
*       FILES LISTED ARE THOSE WHOSE FIRST RECORDS APPEAR
*       TO BE PROPERLY FORMATTED FIO ID RECORDS, AND FILES
*       WHICH ARE PASSWORDED OR READ PROTECTED
*
* ERROR IN CAL---- EXITS TO ERRFF FOR PROCESSING
*
* FIO18 CALLS GARBCOLL AND PERFORMS DATA BLOCK EXTENSION BECAUSE
*       THE SIZE OF THE RESULT CANNOT BE PREDETERMINED
*
FIO18      BGZ          ERDOMAIN           CHECK IF NUMERIC-YES-QUIT
          BEZ          %+2              ZERO-OK-RESET ACCT.
          LI,R7        2                TEXT-SET ACCT
          LI,R6        2
          STB,R7       OP1STACC,R6      SET OR RESET ACCOUNT CONTROL
          LW,R5        BLANKS
          STD,R5       OP1STACT        PRESET ACCT DATA TO BLANKS
          AI,R7        0                CHECK IF ACCT INDICATED
          BEZ          FIO18A          NO-LEAVE BLANKS
          LI,R6        BA(CONSTBUF+1)
          LI,R7        BA(OP1STACT)
          LW,R8        CONSTBUF
          STB,R8       R7
FIO18A     MBS,R6      0                SET ACCT DATA-WITH TRAILING BLANKS
          BAL,R8       GARBCOLL        GARBAGE COLLECT-CONTIGUOUS SPACE
          LI,R11       4                NEEDED FOR TEXT DATA BLOCK
          BAL,R7       ALOCBLK         GET 1ST 4 WORDS-WILL BE EXTENDED
          LW,R7        FIO1XMAT
          STW,R7       *R4              SET UP AS TEXT MATRIX-4 WORDS
          STW,R4       RESULT          SET AS RESULT
          LI,R7        0
          STW,R7       2,R4            SET # OF ROWS=0
          LI,R7        24
          STW,R7       3,R4            SET # OF COLUMNS=24
          AI,R4        2
          STW,R4       CONSTBUF        SAVE POINTER TO # OF ROWS
CALFION1   CAL1,1     FPTOP1ST
          B            CALFIO1R1
FIONXTF    LI,R5      F:TF
          BAL,R6       CLOSV
          MTW,0        BREAKFLG
          BNEZ         DXRETURN        CHECK FOR BREAK
CALFIONX   CAL1,1     FPTOPNXT        YES(OR HANGUP)-CLEAR OUT
CALFIO1R1  CAL1,1     FPTRD1ST        OPEN NEXT FILE
          LH,R14       F:TF+4         READ 1ST RECORD
          SLS,R14      -1             CHECK ARS
          AI,R14       -20
          BNEZ         FIONXTF        NOT AN FIO ID REC.
          LW,R14       FIDBUF+1       CHECK TIME WORD
          CLM,R14      YEARANGE
          BCS,9        FIONXTF        NOT AN FIO ID REC
          LI,R0        X'40'         BLANK
FIO18B     LI,R11     6                GET AN EXTENSION TO DATA BLOCK
          BAL,R7       ALOCBLK
          MTW,6        *RESULT        UPDATE TRU DATA BLOCK SIZE
          MTW,1        *CONSTBUF      AND # OF ROWS
          LD,R6        OP1STACT       STORE
          STD,R6       *R4            ACCOUNT
          LW,R7       BLANKS
          SLS,R4       -1             DW ALIGNMENT
          STD,R7       2,R4           PREBLANK
          STD,R7       4,R4           NAME FIELD
          LI,R6        BA(F:TF+23)+1  SOURCE ADDRESS-NAME
          LW,R7        R4
          SLS,R7       3             BYTE ALIGNMENT
          AI,R7        10            DESTINATION ADDRESS
          LB,R8        F:TF+23       BYTE COUNT
          STB,R8       R7
          MBS,R6       0             MOVE NAME
          SLS,R4       3             BA ALIGNMENT
          AI,R4        22            OFFSET TO END OF NAME
          STB,R0       0,R4          STORE BLANK OR ASTERISK
          B            FIONXTF        GO TO NEXT FILE
*
* FIO19-FORMS TEXT ARRAY OF CURRENTLY OPEN FILE-ACCT-NAMES
*       OR INTEGER VECTOR OF CURRENTLY OPEN DCB NO'S

```

```

*
FI019  BLEZ      ERDOMAIN
      AI,R7      -2
      BGZ      ERDOMAIN
      LI,R6      1
      LI,R5      0
      LW,R8      FIODCBT-1,R6
      LW,R8      *R8
      CW,R8      BITPOS-10
      BAZ      FI019B
      AI,R5      1
      STW,R6     CONSTBUF+1,R5
      AI,R6      1
      CI,R6      NUMFILES
      BLE      FI019A
      STW,R5     CONSTBUF+1
      AI,R7      0
      BLZ      FI019C
      AI,R5      4
      B         FI019D
      MI,R5      5
      AI,R5      5
      AND,R5     FFFFFFFF
      LW,R11     R5
      BAL,R7     ALOCBLK
      STW,R4     RESULT
      MTW,-2     CONSTBUF
      BLZ      FI019E
      LI,R7      X'0301'
      STH,R7     *R4
      LW,R5      CONSTBUF+1
      BNEZ      FI019S
      STW,R5     2,R4
      B         DXRETURN
      LW,R6      CONSTBUF+1,R5
      LW,R6      FIOTIE-1,R6
      STW,R6     CONSTBUF+1,R5
      BDR,R5     FI019S
      LW,R5      CONSTBUF+1
      AI,R5      1
      SCS,R5     -4
      LC        R5
      LM,R5      CONSTBUF+1
      STM,R5     2,R4
      B         DXRETURN
      LI,R7      X'0202'
      STH,R7     *R4
      LI,R6      20
      STW,R6     3,R4
      LW,R7      CONSTBUF+1
      STW,R7     2,R4
      BEZ      DXRETURN
      LI,R5      1
      LW,R6      CONSTBUF+1,R5
      LW,R6      FIODCBT-1,R6
      AI,R6      23
      LW,R9      BLANKS
      STD,R9     R10
      STD,R9     R12
      LI,R7      X'FF00'
      AND,R7     3,R6
      BEZ      FI019G
      LW,R9      4,R6
      LW,R10     5,R6
      LB,R8      *R6
      SLS,R6     2
      AI,R6      1
      LI,R7      45
      STB,R8     R7
      MBS,R6     0
      LCI       5
      STM,R9     4,R4
      AI,R4      5
      AI,R5      1
      CW,R5      CONSTBUF+1
      BLE      FI019F
      B         DXRETURN

      RANGE TEST
      TOO BIG-QUIT

      PRESET ITEM COUNT
      GET DCB ADDRESS
      CHECK IF OPEN

      NO
      YES-KICK ITEM COUNT
      STORE DCB #

      LOOP
      SAVE ITEM COUNT
      CHECK WHICH OPTION
      TEXT
      NUMBERS

      TEXT TAKES 5 WORDS/ACCT-NAME
      AND IS ARRAY (RANK 2)

      GET THE DATA BLOCK

      INTEGER VECTOR
      SET TYPE AND RANK
      GET COUNT
      BRANCH IF NOT EMPTY
      EMPTY-INDICATE IT

      SWITCH FROM
      DCB NUMBERS
      TO TIE NUMBERS
      LOOP
      GET COUNT
      +1

      GET COUNT+VALUES
      STASH IN RESULT
      RETURN
      TEXT-MATRIX
      SET TYPE-RANK
      NO. OF COLUMNS

      # OF ROWS
      QUIT IF EMPTY

      POINT TO DCB TABLE
      THEN TO DCB
      THEN TO NAME

      BLANK R9-R11
      BLANK R12-R13
      MASK
      CHECK IF ACCT SPECIFIED
      NO
      YES-SET UP R9-R10

      BYTE COUNT OF NAME
      BA(NAME)-1
      SOURCE ADDRESS FOR MBS
      BA(R11)+1,DESTINATION ADDRESS
      COUNT
      FORM NAME+TRAILING BLANKS IN R11-R13
      LOAD ACCT-NAME IN DATA BLOCK
      UPDATE R4
      AND POINTER
      LOOP
      DONE

```

*

```

* FIOKEYSZ-CHECK FOR VALID KEY NUMBER
* R4=LINK,R7 USED BUT NOT CHANGED
*
FIOKEYSZ CW,R7    MAXKEY
        BLE      0,R4
        B        ERDOMAIN
*
* FIO20-SET(OR RESET) SERIAL NUMBER FOR PRIVATE PACKS
*
FIO20   BLZ      FIO20A    TEXT-SET SERIAL NO.
        BGZ      ERDOMAIN    ERROR
        LI,R4    BA(FIOSNC)+2  RESET SERIAL NO. CONTROL
        B        FIO30R4R    (R7=0)
FIO20A  LI,R3    FIOSN      SERIAL NO. ADDRESS
        LI,R4    BA(FIOSNC)+2 SN CONTROL WORD COUNT
        LI,R7    BA(FIOSN)    SERIAL NO. ADDRESS
        LW,R9    CONSTBUF    GET COUNT
        STB,R9   R7          SET UP FOR MBS
        LW,R5    BLANKS
        STD,R5   *R3
        STW,R5   2,R3        PRE-BLANK FIOSN
        LI,R6    BA(CONSTBUF+1)
        MBS,R6   0          LOAD FIOSN
        LW,R7    CONSTBUF    GET COUNT AGAIN
        AI,R7    3          ROUND UP TO
        SLS,R7   -2        NO. OF WORDS
        B        FIO30R4R    USE 'RESET' TO SET CONTROL WORD
*
* FIO21-ESTABLISHES FILE ID (NAME-ACCT-PASSWORD) AS SINGLE PRIMITIVE
*
FIO21   BGEZ     ERDOMAIN    RTARG MUST BE TEXT
        LW,R5    BLANKS
        STD,R5   FIOACCT    RESET ACCOUNT AND PASSWORD
        STD,R5   FIOPASS
        LI,R9    0
        LI,R6    BA(FIOACCC)+2
        LI,R7    BA(FIOPASC)+2
        STB,R9   0,R6
        STB,R9   0,R7
        LW,R5    CONSTBUF
        LI,R8    X'15'
        STB,R8   CONSTBUF+1,R5 SET CR AT END OF TEXT
        LI,R1    BA(CONSTBUF+1) SET TO SCAN
        BAL,R4   ACQNB      GET FIRST NON-BLANK IN TEXT
FIO21A  BAL,R14  ACQIT      ACQUIRE ALPHANUMERIC VALUE
        AI,R5    0
        BEZ     FIO21E      QUIT IF EMPTY
        CLM,R3   NONAME     CHECK IF NEXT ITEM IS ALPHANUMERIC
        BCR,9    FIO21B      NO
        AI,R9    0          YES-CHECK IF ACCT ALREADY SET
        BNEZ    FIO21E      YES-ERROR
        CI,R5    8          CHECK SIZE
        BG      FIO21E
        LD,R8    NAMEBUF
        STD,R8   FIOACCT    SET ACCOUNT
        LI,R9    2
        STB,R9   0,R6
FIO21B  B        FIO21A      LOOP FOR NAME
        CI,R5    11         NAME ITEM
        BG      FIO21E      TOO LONG
        STB,R5   FIONAME
        LI,R4    BA(NAMEBUF) MOVE NAME TO
        LW,R5    FIOMBSN1    FIONAME IN TEXTC
        MBS,R4   0
        CI,R2    ':'        CHECK FOR PASSWORD
        BE      FIO21C      YES
        CI,R2    ':'        CHECK FOR ACCT
        BNE     FIO21X      NO
        BAL,R4   ACQNXNB    MAYBE
        CI,R2    ':'
        BE      FIO21C      NO-PASSWORD ONLY
        BAL,R14  FIOAORP    YES-GET ACCOUNT
        STD,R8   FIOACCT
        LI,R9    2
        STB,R9   0,R6
        BAL,R4   ACQNB      GET NON-BLANK
        CI,R2    ':'        CHECK FOR PASSWORD
        BE      FIO21C      YES

```

FI021X	CI, R2	X'15'	CHECK FOR CR	U20-0265
	BE	FIOEX		U20-0266
FI021E	LI, R12	21	SET ERROR CODE	U20-0267
	B	ERRAPL		
FI021C	BAL, R4	ACQNXNB	PROCESS PASSWORD	U20-0269
	BAL, R14	FIOAORP		U20-0270
	STD, R8	FIOPASS		U20-0271
	LI, R9	2		U20-0272
	STB, R9	0, R7		U20-0273
	B	FI021X		U20-0274
*				U20-0275
FIOAORP	AI, R1	-1		U20-0276
	LW, R8	BLANKS		U20-0277
	LW, R9	BLANKS	PRESET R8-R9 TO BLANKS	U20-0278
	LI, R5	-1		U20-0279
FIOAORP1	AI, R5	1		U20-0280
	BAL, R4	ACQNXCC	GET NEXT CHAR.	U20-0281
	CI, R2	X'40'	BLANK	U20-0282
	BE	FIOAORP2	YES-QUIT	U20-0283
	CI, R2	' '	PERIOD	U20-0284
	BE	*R14	YES-QUIT	U20-0285
	CI, R2	' '	COMMA	U20-0286
	BE	*R14	YES-QUIT	U20-0287
	CI, R2	X'15'	CR	U20-0288
	BE	*R14	YES-QUIT	U20-0289
	STB, R2	R8, R5	STASH BYTE	U20-0290
	CI, R5	7	HOW MANY	U20-0291
	BG	FI021E	TOO MANY	U20-0292
	B	FIOAORP1	LOOP	U20-0293
FIOAORP2	BAL, R4	ACQNXNB	SKIP TRAILING BLANKS	U20-0294
	B	*R14		
*				
*	* FI022-WRITES APL DATA WITHOUT HEADER INFORMATION FOR EXTERNAL			
*	FILE CREATION. WRITES KEYED FILE.			
*	SIZE IS ACTUAL SIZE OF DATA EXCEPT FOR LOGIC VALUES, WHICH			
*	ARE ROUNDED UP TO BYTES(MULTIPLES OF 8 BITS)			
*	INDEX SEQUENCES ARE CONVERTED TO INTEGER VECTORS PRIOR TO			
*	OUTPUT.			
*				
FI022	LB, R5	*RTARG	R5=DATA TYPE	
	CI, R5	5		
	BE	FI022F	INDEX SEQ-NEEDS TO BE EXPANDED	
FI022A	LI, R7	1		
	LB, R6	*RTARG, R7	R6=RANK	
	LW, R4	RTARG	START OF DATA BLOCK HEADER	
	AI, R4	2	SKIP 2 HEADER WORDS	
	AW, R4	R6	SKIP LENGTH WORDS	
	CI, R5	4	IS DATA TYPE REAL	
	BNE	FI022B	NO	
	AI, R4	1	YES-ROUND TO DW BOUND	
	AND, R4	FFFFFFFFE		
FI022B	STW, R4	FIOBUF	WORD ADDRESS-OUTPUT BUFFER	
	LI, R4	2		
	LI, R7	1	ASSUME SCALAR	
	AI, R6	0	CHECK IF IT IS	
	BLEZ	FI022D	YES	
FI022C	MW, R7	*RTARG, R4	NO-COMPUTE NO. OF UNITS	
	AI, R4	1		
	BDR, R6	FI022C	LOOP	
FI022D	EXU	FI022I-1, R5	CONVERT FROM NO. OF UNITS TO BYTES	
FI022E	LI, R6	CALWRFD	SET EXIT FROM FIORCSU	
	B	FIORCSUB	COMPLETE PROCESSING	
FI022F	LW, R5	RTARG		
	LW, R11	2, R5	LENGTH OF INDEX SEQUENCE	
	AI, R11	1	+1 FOR LENGTH WORD	
	BAL, R7	ALOCHNW	ALLOCATE DATA BLOCK FOR INTEGER VECT	
	XW, R4	RTARG	R4= ADDR OF ISEQ DB, RTARG =NEW DB	
	LW, R5	RTARG	R5= ADDR OF NEW DB	
	LI, R11	X'0301'		
	STH, R11	*RTARG	SET TYPE AND RANK	
	LW, R11	2, R4	LENGTH	
	STW, R11	2, R5		
	BEZ	FI022H	EMPTY	
	LW, R8	3, R4	BASE VALUE	
FI022G	AW, R8	4, R4	INCREMENT VALUE	
	STW, R8	3, R5	STORE INTEGER VECTOR ELEMENT	
	AI, R5	1		
	BDR, R11	FI022G	LOOP	

FI022H	BAL,R7	DREF	DROP THE INDEX SEQUENCE
	LI,R5	3	SET INTEGER TYPE
	B	FI022A	AND OUTPUT THAT BLOCK
FI022I	B	FI022J	LOGIC VALUE
	NOP		CHARACTER
	SLS,R7	2	INTEGER
	SLS,R7	3	REAL
FI022J	AI,R7	7	ROUND LOGIC UP
	SLS,R7	-3	DIVIDE BY 8
	B	FI022E	
*			
* FI023-READ A DATA RECORD OF NON-APL FORM USING CURRENT KEY			
* RESULT IS A CHARACTER VECTOR WITH LENGTH=ARS			
* PRIMITIVES 25,26, AND 27 USED TO CONVERT CHARACTER			
* VECTOR RESULT TO LOGIC,INTEGER,OR REAL IF REQUIRED			
* FOR PROPER DATA REPRESENTATION.			
*			
FI023	BLEZ	ERDOMAIN	
	AI,R7	12	ADD 3 WORDS FOR HEADER
	BAL,R8	FIOSETRD	USE NORMAL SETUP FOR READ
	LI,R8	-12	
	AWM,R8	FIOSIZ	ADJUST TO READ BEYOND HEADER
	MTW,3	FIOBUF	
CALRDRDK	CAL1,1	FPTRDFDK	READ KEYED RECORD
	LI,R8	DXRETURN	SET EXIT FROM FIOENDRR
FIOENDRR	LW,R4	CONSTBUF	ALLOCATED NUMBER OF WORDS
	LI,R5	X'0201'	CHARACTER VECTOR ID
	STH,R5	R4	
	LI,R5	1	REF COUNT=1
	LI,R6	13	
	LW,R6	*FIODCB,R6	R6=ARS
	LCI	3	
	STM,R4	*RESULT	SET UP HEADER
	AI,R6	19	ARS+HEADER+DW ROUND
	SLS,R6	-2	
	AND,R6	FFFFFFFFE	SIZE OF BLOCK ACTUALLY NEEDED
	LW,R11	CONSTBUF	SIZE ALLOCATED
	SW,R11	R6	SURPLUS
	LW,R4	RESULT	
	BAL,R7	GIVEBACK	RETURN IT
	B	*R8	EXIT
*			
* FI024-READ SEQUENTIALLY, NON-APL DATA RECORD			
* SIMILAR TO FI023 BUT NOT KEYED READ			
*			
FI024	BLEZ	ERDOMAIN	
	AI,R7	12	ADD 3 WORDS FOR HEADER
	BAL,R8	FIOSETRD	USE NORMAL SETUP FOR READ
	LI,R8	-12	
	AWM,R8	FIOSIZ	ADJUST TO READ DATA ONLY
	MTW,3	FIOBUF	
CALRDRDS	CAL1,1	FPTRDFDS	SEQUENTIAL READ
	BAL,R8	FIOENDRR	PROCESS RESULT
	BAL,R8	FIOSETKY	SET KEY (IF ANY)
	B	DXRETURN	EXIT
*			
* FI025-CONVERT CHARACTER VECTOR TO LOGIC VECTOR			
* FI026(EQU FI025) INTEGER VECTOR			
* FI027(EQU FI025) REAL VECTOR			
*			
FI025	LH,R5	*RTARG	
	CI,R5	X'0201'	CHECK IF TEXT VECTOR
	BNE	ERRANK	NO-RANK ERROR
	LW,R3	RTARG	
	LW,R11	2,R3	CHECK LENGTH FOR CONFORMABILITY
	CW,R11	FI025T1-25,R4	(0,3,OR 7)
	BANZ	ERLENGTH	LENGTH NOT CONFORMABLE
	LW,R6	R4	SAVE PRIMITIVE NO. IN R6
	LW,R5	1,R3	REF COUNT OF RTARG
	AI,R5	-1	
	BEZ	FI025D	DATA BLOCK IS REUSEABLE
	INT,R11	0,R3	NOT REUSEABLE,GET NEW BLOCK
	BAL,R7	ALOCBLK	
	STW,R4	RESULT	SET RESULT
FI025A	INT,R11	0,R3	LENGTH OF DATA BLOCK
	AW,R4	R11	POINT 1 PAST END OF RESULT
	AW,R3	R11	POINT 1 PAST END OF RTARG
	CI,R6	27	CHECK IF CONVERT TO REAL


```

        BNE      FIO25B      NO
        AI,R3    -1          YES-SET FOR OFFSET
FIO25B  AI,R11    -3          SUBTRACT HEADER WORDS
FIO25C  AI,R3     -1          MOVE DATA
        AI,R4     -1          FROM RTARG
        LW,R7     0,R3        TO RESULT,
        STW,R7    0,R4        OFFSETTING IF
        BDR,R11   FIO25C      RESULT IS REAL.
        LW,R3     RTARG
        LW,R4     RESULT
        B         FIO25E
FIO25D  MTW,1     1,R3        REUSE RTARG-INCREMENT REF COUNT
        LW,R4     RTARG
        STW,R4    RESULT      SET RESULT=RTARG
        CI,R6     27          CHECK IF REAL
        BE        FIO25A      YES-OFFSET DATA
FIO25E  LW,R7     FIO25T2-25,R6
        STH,R7    *RESULT     SET TYPE-RANK
        LW,R7     2,R3        GET LENGTH
        EXU       FIO25T3-25,R6 ADJUST TO NEW TYPE
        STW,R7    2,R4
        B         DXRETURN    EXIT
FIO25T1 DATA     0,3,7
FIO25T2 DATA     X'0101',X'0301',X'0401'
FIO25T3 SLS,R7     3
        SLS,R7    -2
        SLS,R7    -3

```

*

PAGE

*

* FPT'S FOR FILE I/O SUBSYSTEM FOLLOW:

*

* EXCEPTIONS: FPTOPFIO-IN CSECT 0 OF APLUTSI

*

FPTOP1ST-IN CSECT 0 OF APLUTSI

*

FPTOPNXT-IN CSECT 1 OF APLUTSI

*

FPTRD1ST-IN CSECT 1 OF APLUTSI

*

* ERROR EXIT 'EQUATES'-ERRFF IS COMMON ACTUAL FIO ERROR ADDRESS

```

ABNFPR  EQU      ERRFF      ABN-SKIP RECORD
ERRRDFI EQU      ERRFF      READ ID REC-ERR
ABNRDFI EQU      ERRFF      READ ID REC-ABN
ERRRDFD EQU      ERRFF      READ DATA REC-ERR
ABNRDFD EQU      ERRFF      READ DATA REC-ABN
ERRWRFI EQU      ERRFF      WRITE ID REC-ERR
ABNWRFI EQU      ERRFF      WRITE ID REC-ABN
ERRWRFD EQU      ERRFF      WRITE DATA REC-ERR
ABNWRFD EQU      ERRFF      WRITE DATA REC-ABN

```

* FPTRDFIS-FPT TO READ ID-FIO RECORD-SEQUENTIALLY

*

```

FPTRDFIS GEN,1,7,7,17 1,X'10',0,FIODCB    DCB ADDRESS-INDIRECT
        DATA     X'F4000010'          P1,2,3,4,6 AND WAIT
        DATA     ERRRDFI              ERR (P1)
        DATA     ABNRDFI              ABN (P2)
        DATA     FIDBUF              BUF (P3)
        DATA     FIDSIZ              SIZ (P4)
        DATA     0                   BTD (P6)
FIDSIZ  EQU      20                   BYTES PER FID RECORD
FIDBUF  EQU      IDBUF

```

*

* FPTRDFIK-FPT TO READ KEYED FIO-ID RECORD

*

```

FPTRDFIK GEN,1,7,7,17 1,X'10',0,FIODCB
        DATA     X'FC000010'          P1,2,3,4,5,6 AND WAIT
        DATA     ERRRDFI              ERR (P1)
        DATA     ABNRDFI              ABN (P2)
        DATA     FIDBUF              BUF (P3)
        DATA     FIDSIZ              SIZ (P4)
        DATA     FIOKEY              KEY (P5)
        DATA     0                   BTD (P6)

```

*

* FPTRDFDS-FPT TO READ FIO RECORD-SEQUENTIALLY

*

```

FPTRDFDS GEN,1,7,7,17 1,X'10',0,FIODCB    DCB ADDRESS-INDIRECT
        DATA     X'F4000010'          P1,2,3,4,6 AND WAIT
        DATA     ERRRDFD              ERR (P1)
        DATA     ABNRDFD              ABN (P2)
        GEN,1,31 1,FIOBUF              BUF (P3)

```

```

GEN,1,31 1,FIOSIZ      SIZ (P4)
DATA      0            BTD (P6)
*
*  FPTRDFDK-FPT TO READ FIO DATA RECORD-KEYED
*
FPTRDFDK GEN,1,7,7,17 1,X'10',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      X'FC000010'      P1,2,3,4,5,6 AND WAIT
DATA      ERRRDFD          ERR (P1)
DATA      ABNRDFD          ABN (P2)
GEN,1,31 1,FIOBUF          BUF (P3)
GEN,1,31 1,FIOSIZ          SIZ (P4)
DATA      FIOKEY           KEY (P5)
DATA      0               BTD (P6)
*
*  FPTWRFI-WRITE FIO ID RECORD
*
FPTWRFI  GEN,1,7,7,17 1,X'11',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      X'FC000070'      P1,2,3,4,5,6+WAIT,NEWKEY,ONEWKEY
DATA      ERRWRFI          ERR (P1)
DATA      ABNWRFI          ABN (P2)
DATA      FIDBUF           BUF (P3)
DATA      FIDSIZ           SIZ (P4)
DATA      FIOKEY           KEY (P5)
DATA      0               BTD (P6)
*
*  FPTWRFD-WRITE FIO DATA RECORD
*
FPTWRFD  GEN,1,7,7,17 1,X'11',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      X'FC000070'      P1,2,3,4,5,6+WAIT,NEWKEY,ONEWKEY
DATA      ERRWRFD          ERR (P1)
DATA      ABNWRFD          ABN (P2)
GEN,1,31 1,FIOBUF          BUF (P3)
GEN,1,31 1,FIOSIZ          SIZ (P4)
DATA      FIOKEY           KEY (P5)
DATA      0               BTD (P6)
*
*  FPTDELR-DELETE RECORD-KEYED
*
FPTDELR  GEN,1,7,7,17 1,X'0D',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      X'80000000'      P1
DATA      FIOKEY           KEY
*
*  FPTFPFB-POSITION TO BEGINNING OF FILE
*
FPTFPFB  GEN,1,7,7,17 1,X'1C',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      16
*
*  FPTFPFE-POSITION TO END OF FILE
*
FPTFPFE  GEN,1,7,7,17 1,X'1C',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      0
*
*  FPTFPR-SKIP RECORD -FORWARD
*
FPTFPR   GEN,1,7,7,17 1,X'1D',0,FIODCB      DCB ADDRESS-INDIRECT
DATA      X'C0000000'      P1,2 FORWARD
DATA      1                SKIP 1 RECORD
DATA      ABNFPR           ABN
*
PAGE
*
*  FIOERR-FILE I/O MONITOR ERROR PROCESSOR
*
FIOERR   RES      0
ERRFF    STW,R10  IOERCODE      RECORD LATEST ERR OR ABN CONDITION.
          LH,R12   R10          GET ERROR CODE AND SUBCODE
          SLS,R12  -1           SHIFT OUT EXTRA BIT
          AND,R8   X1FFFF       MASK ERROR ADDRESS
          CI,R12   3*128        CHECK FOR FILE NOT FOUND
          BNE      ERRFF1       NO
          MTW,0    FIOABNT      YES-CHECK FOR INABN MODE
          BNEZ     FIOEX        YES-NORMAL EXIT
ERRFF1   CI,R8    CALRDFII+1    CHECK IF READ FOR 'OPEN'      20-00025
          BE       ERRFF4       YES                          20-00026
          CI,R8    CALRDFDS+1    NO-CHECK IF READ ERROR      20-00027
          BE       ERRFF2       YES
          CI,R8    CALRDFDK+1    MAYBE
          BE       ERRFF2       YES

```

	CI,R8	CALDRDRS+1	MAYBE	
	BE	ERRFF2	YES	
	CI,R8	CALDRDRK+1	MAYBE	
	BE	ERRFF2	YES	
	CI,R8	CALRDFIS+1	CHECK IF SEQUENTIAL ID REC READ	
	BNE	ERRFF3	NO	
	CI,R12	7*128	YES-CHECK IF RECORD TOO BIG	
	BNE	ERRFF3	NO	
	B	CALRDFIS	YES-KEEP ON READING	
ERRBADDB	LI,R12	2	ERROR CODE=0, SUBCODE=2	
	LI,R10	X'40000'	FAKE SAME CODES FOR R10.	
ERRFF2	LW,R6	CONSTBUF	ERROR ON READ, SET PROPER	20-00029
	LI,R7	1	SIZE AND REF COUNT	20-00030
	STD,R6	*RESULT	IN RESULT DATA BLOCK	20-00031
	LI,R4	0	SET TO Deref	20-00032
	XW,R4	RESULT	DEREFERENCE THE RESULT BLOCK	
	BAL,R7	DREF		
ERRFF3	STW,R10	IOERCODE	SAVE CODE, SUBCODE, (MAYBE) DCB LOC.	
	LW,R11	WHATERR	SHOULD APL PROCESS THE ERROR...	
	BNEZ	HANDLERR	YES (14 T-BAR 2 WAS THE INTRINSIC)	
GENSCLR	LI,R11	4	NO (14 T-BAR 1 WAS THE INTRINSIC).	
	BAL,R7	ALOCBLK	ALLOCATE DATA BLOCK	
	LI,R11	TYPEINTG	INTEGER TYPE	
	STB,R11	*R4		
FIOEX1	STW,R12	2,R4		
	STW,R4	RESULT	SET RESULT	
	B	DXRETURN		
ERRFF4	LW,R5	FIODCBN0	SET STREAM NO.	20-00034
	CI,R12	6*128	CHECK IF EOF	20-00035
	BE	FIOMPF	YES-EMPTY FILE	20-00036
	CI,R12	7*128	CHECK IF RECORD TOO BIG	20-00037
	BE	CALRDFII+1	YES-NOT ID RECORD-OK	20-00038
	B	ERRFF3	NO-REAL ERROR	20-00039
* ERRFTFIO-MONITOR CALL ERROR ON F:TF (FIO18)				
* ERRFTFIO				
	CI,R8	CALFION1+1	CHECK IF OPEN OF 1ST FILE	
	BE	ERRFTFN1	YES-TREAT SAME AS NEXT FILE OPEN	
	CI,R8	CALFIONX+1	CHECK IF NXTF OPEN	
	BE	ERRFTFN1	YES-	
	CI,R10	7	NO-MUST BE READ, CHECK IF	
	BE	FIONXTF	RECORD TOO BIG	
	LI,R0	'*'	SET '*' FLAG	
	B	FIO18B	NO-PROBABLY TIED, LIST IT	
ERRFTFN1	CI,R10	2	CHECK IF LAST FILE	
	BE	DXRETURN	YES-QUIT	
	LI,R0	'*'	SET '*' FLAG	
	B	FIO18B	NO-ASSUME PROTECTED AND LIST IT	
* FIOEX-EXIT WITH EMPTY VECTOR RESULT				
* FIOEX				
	LI,R11	4		
	BAL,R7	ALOCBLK		
	LI,R11	X'0301'	INTEGER VECTOR'	
	STH,R11	*R4		
	LI,R12	0		
	B	FIOEX1		
* ERRAPL-FILE I/O ERR DETECTED BY APL, NOT BY THE MONITOR.				
* ERRAPL				
	LW,R10	R12	COPY APL-ERROR (SUBCODE). FAKE THE	
	SLS,R10	17	I/O ERR CODE WD: CODE=0,	
	B	ERRFF3	SUBCODE=APL-ERR, DCB ADDR = 0.	
PAGE				
* THE 'CASECODE' & 'CASEID' TABLES ARE IN 1-TO-1 CORRESPONDENCE.				
* CASECODE CONSISTS OF HALFWDs CONTAINING AN I/O ERR OR ABN VALUE,				
* CODE & SUBCODE. THESE HEX CODES OCCUPY BITS 1-8 & 9-15				
* RESPECTIVELY, WITH BIT 0 OF THE HALFWD ZEROED (NOTE THE GEN'S).				
* CASEID IS A MATCHED SET OF BYTES CONTAINING INTERNAL ERROR I.D.				
* VALUES FOR EACH CODE & SUBCODE OF INTEREST.				
* BOUND 4				
CASECODE	GEN,1,8,7	0,X'FF',X'7F'	0 (NEVER USED)	
	GEN,1,8,7	0,0,0	1	
	GEN,1,8,7	0,0,21	2	
	GEN,1,8,7	0,0,1	3	
	GEN,1,8,7	0,0,2	4	

GEN,1,8,7	0,0,3	5
GEN,1,8,7	0,0,4	6
GEN,1,8,7	0,0,20	7
GEN,1,8,7	0,X'03',0	8
GEN,1,8,7	0,X'14',0	9
GEN,1,8,7	0,X'14',X'01'	10
GEN,1,8,7	0,X'56',0	11
GEN,1,8,7	0,X'06',0	12
GEN,1,8,7	0,X'0D',0	13
GEN,1,8,7	0,X'42',0	14
GEN,1,8,7	0,X'43',0	15
GEN,1,8,7	0,X'57',0	16
GEN,1,8,7	0,X'75',0	17
GEN,1,8,7	0,X'75',X'01'	19
GEN,1,8,7	0,X'75',X'02'	19
GEN,1,8,7	0,X'75',X'03'	20
GEN,1,8,7	0,X'75',X'04'	21
GEN,1,8,7	0,X'75',X'05'	22
GEN,1,8,7	0,X'75',X'06'	23
GEN,1,8,7	0,X'20',X'01'	24
GEN,1,8,7	0,X'20',X'02'	25
GEN,1,8,7	0,X'20',X'03'	26
GEN,1,8,7	0,X'20',X'04'	27
GEN,1,8,7	0,0,5	28
GEN,1,8,7	0,0,6	29
GEN,1,8,7	0,0,7	30
GEN,1,8,7	0,0,8	31
GEN,1,8,7	0,0,9	32
GEN,1,8,7	0,0,10	33
GEN,1,8,7	0,0,11	34
GEN,1,8,7	0,0,12	35
GEN,1,8,7	0,0,13	36
GEN,1,8,7	0,0,14	37
GEN,1,8,7	0,0,15	38
GEN,1,8,7	0,0,16	39
GEN,1,8,7	0,0,17	40
GEN,1,8,7	0,X'2E',0	41
GEN,1,8,7	0,X'44',0	42
GEN,1,8,7	0,X'51',X'00'	43
GEN,1,8,7	0,X'25',0	44

SPACE
NCASES EQU HA(%) -HA(CASECODE) # SPEC.CASES OF INTEREST.
SPACE
* NOTE -- CASEID TABLE MUST MATCH CASECODE TABLE.
SPACE

CASEID	BOUND	4	I.D. FOR:
DATA,1	IDFIOERR	0	FILE I/O ERR
DATA,1	IDFILNAM	1	FILE NAME ERR
DATA,1	IDFILNAM	2	
DATA,1	IDFILDAM	3	FILE DAMAGE
DATA,1	IDNOTAPL	4	NOT APL FILE
DATA,1	IDFTFULL	5	FILE TBL FULL
DATA,1	IDFILACC	6	FILE ACCESS ERR
DATA,1	IDFILACC	7	
DATA,1	IDFILNAM	8	FILE NAME ERR
DATA,1	IDFILACC	9	FILE ACCESS ERR
DATA,1	IDFILBSY	10	FILE IN USE
DATA,1	IDFILSPC	11	FILE SPACE TOO LOW
DATA,1	IDFILIDX	12	FILE INDEX ERR
DATA,1	IDFILIDX	13	
DATA,1	IDFILIDX	14	
DATA,1	IDFILIDX	15	
DATA,1	IDFILSPC	16	FILE SPACE TOO LOW
DATA,1	IDFILDAM	17	FILE DAMAGE
DATA,1	IDFILDAM	18	
DATA,1	IDFILDAM	19	
DATA,1	IDFILDAM	20	
DATA,1	IDFILDAM	21	
DATA,1	IDFILDAM	22	
DATA,1	IDFILDAM	23	
DATA,1	IDNOPACK	24	PRIVATE PACK UNAVAIL, CALL OPR.
DATA,1	IDNOPACK	25	
DATA,1	IDNOPACK	26	
DATA,1	IDNOPACK	27	
DATA,1	IDFILTIE	28	FILE TIE ERR
DATA,1	IDFILTIE	29	
DATA,1	IDFILTIE	30	
DATA,1	IDFILTIE	31	

	DATA,1	IDFILTIE	32
	DATA,1	IDFILTIE	33
	DATA,1	IDFILTIE	34
	DATA,1	IDFILTIE	35
	DATA,1	IDFILTIE	36
	DATA,1	IDFILTIE	37
	DATA,1	IDFILTIE	38
	DATA,1	IDFILTIE	39
	DATA,1	IDFILTIE	40
	DATA,1	IDFILTIE	41
	DATA,1	IDFILTIE	42
	DATA,1	IDFILTIE	43
	DATA,1	IDFILTIE	44
SPACE			
	BOUND	4	
PAGE			
*			
*	HANDLERR-ENTERED ON ERROR IF THE 'ERROR-HANDLING' VERSION OF THE		
*	FILE I/O INTRINSIC (14 T-BAR 2) IS CURRENTLY APPLICABLE.		
*			
*	R10 CONTAINS THE ERROR CODE (BITS 0-7) & SUBCODE (BITS 8-14).		
*			
HANDLERR	SLS,R10	-17	GET ONLY CODE & SUBCODE.
	LI,R1	NCASES	= NO.OF ERROR CASES TO CK.
CASECHK	CH,R10	CASECODE,R1	DOES CODE,SUBCODE MATCH A SPEC.CASE
	BNE	NEXTCASE	NO.
	B	CASEHIT	YES.
NEXTCASE	BDR,R1	CASECHK	LOOP TILL R1=0 (JUST 'FILE I/O ERR')
CASEHIT	LB,R2	CASEID,R1	GET ERROR I.D. FOR THIS CASE.
	B	ERFILEIO	EXIT TO HANDLE ERR PROCESSING.
	END		


```

TITLE      'APLFMT-B00,10/10/73,DWG702985'
SYSTEM     SIG7F
APLFMT@    CSECT      1
*
* DEF'S
DEF         APLFMT@
DEF         DELTAFMT
*
* REF'S
*
REF         ALOCBLK      ROUTINE TO ALLOCATE DATA BLOCK
REF         ALOCHNW      ALLOCATE DATA BLOCK-N WORDS+HEADER
REF         BITMASK      TABLE OF 1 BIT MASKS
REF         BREAKFLG     BREAK FLAG
REF         CURRCS       CURRENT CODESTRING POINTER
REF         DXRETURN     DYADIC EXECUTION RETURN
REF         ERDOMAIN     DOMAIN ERROR
REF         ERFORMAT     FORMAT SYNTAX ERROR
REF         ERRANK       RANK ERROR
REF         F0F9         CONSTANTS 'F0' AND 'F9'
REF         FLHALF       FL. PT. LONG 0.5                      U21-0004
REF         FMTMPS       BASE OF TEMPS FOR DELTAFMT-DW BOUND-
REF         LETTERS      DW 'A','Z'
REF         LFARG        LEFT ARGUMENT POINTER
REF         OFFSET       CODESTRING OFFSET
REF         OPBREAK      TRANSFER POINT FOR BREAK
REF         PLUSREAL     ENTRY TO REAL # CONV ROUTINE
REF         RESULT       RESULT POINTER
REF         RTARG        RIGHT ARGUMENT POINTER
REF         SINGOUT      ROUTINE TO WRITE A DATA BLOCK
REF         STATEPTR     STATE POINTER
REF         TENSTBL      TABLE OF INTEGER POWERS OF 10
REF         X4E1         CONST. USED TO FIX FL. PT. NO.        U21-0006
REF         ZEROZERO     DW ZERO
*
* TEMPS FOR APLFMT-LOCATED IN APLUTSI (CSECT 0)
*
DOUBLE-WORD-GROUPS
NROWS      EQU      FMTMPS      DWB      MAX # OF ROWS      @@@@@@@@
NCOLS      EQU      FMTMPS+1    DWB      TOTAL # OF COLUMNS  @
ARGWORDS   EQU      FMTMPS+2    DWB-DW   ARG ADDRESS AND COUNT  @
ARGADR     EQU      FMTMPS+4    DWB      ARG POINTER ADDRESS  @
ARGCOUNT  EQU      FMTMPS+5    DWB      ARG COUNT          @
ARGROWS    EQU      FMTMPS+6    DWB      # OF ARG ROWS       @
ARGCOLS    EQU      FMTMPS+7    DWB      # OF ARG COLUMNS    @
SAVEREAL   EQU      FMTMPS+8    DW      DW SAVE LOC FOR REAL VALUE @
*
* DECORATOR ADDRESS AND LENGTH TEMPS
*
DECORTEX   EQU      MTEXT-1STDECOR  TABLE OFFSET      @@@@@@@@
MTEXT      EQU      FMTMPS+20      ADDRESSES OF DECORATORS @
NTEXT      EQU      MTEXT+1        @
PTEXT      EQU      MTEXT+2        @
QTEXT      EQU      MTEXT+3        @
RTEXT      EQU      MTEXT+4        @@@@@@@@
*
DECORLEN   EQU      MLENGTH-1STDECOR  TABLE OFFSET      @@@@@@@@
MLENGTH    EQU      FMTMPS+25      LENGTHS OF DECORATORS @
NLENGTH    EQU      MLENGTH+1      @
PLENGTH    EQU      MLENGTH+2      @
QLENGTH    EQU      MLENGTH+3      @
RLENGTH    EQU      MLENGTH+4      @@@@@@@@
*
* SINGLE TEMPS-SPACE LEFT FOR INSERTION OF OTHERS-ALPHABETIC ORDER
*
COMMACNT   EQU      FMTMPS+40      COMMA COUNT
COMMALOC   EQU      FMTMPS+42      COMMA INSERTION LOCATION
DATALOC    EQU      FMTMPS+44      DATUM ADDRESS
DATASPEC   EQU      FMTMPS+46      FLAG TO INDICATE DATA USE
DORS       EQU      FMTMPS+48      'D' OR 'S' SPECIFICATION
EXPSTART   EQU      FMTMPS+52      EXPONENT STARTING CHAR POS
FORMLIM    EQU      FMTMPS+54      FORMAT POINTER LIMIT
GAPSIZE    EQU      FMTMPS+56      SIZE OF FIELD GAP
INTLEFT    EQU      FMTMPS+58      LOCATION LEFT END OF INTEGER PART
INTRIGHT   EQU      FMTMPS+60      LOCATION RIGHT END OF INTG. PART
INTSIZE    EQU      FMTMPS+62      SIZE OF INTEGER PART
LDFLAG     EQU      FMTMPS+63      LONG DIVIDE FLAG
LOGLBITS   EQU      FMTMPS+64      LOGICAL DATA BITS
LOGLCNT    EQU      FMTMPS+65      LOGICAL DATA BIT COUNT

```

LOWEND	EQU	FMTMPS+66	LOW END OF # IN LONG DIVIDE
NSIZE	EQU	FMTMPS+67	# OF DIGITS IN R8-R9 TO BE CNVRTD
RESLTPTR	EQU	FMTMPS+68	INITIAL ADDRESS OF RESULT DATA
RESWIDTH	EQU	FMTMPS+70	# OF CHARACTERS/ROW OF RESULT
RETAIN	EQU	FMTMPS+72	FLAG INDICATING RETAIN OR DISPLAY
ROWINDEX	EQU	FMTMPS+74	INDEX OF ROW BEING PROCESSED
RPTCOUNT	EQU	FMTMPS+76	REPEAT COUNT FOR A FORMAT
TEXTADR	EQU	FMTMPS+78	TEXT STRING ADDRESS
VALFLAG	EQU	FMTMPS+80	VALUE FLAG;NEG,ZERO,OR POS

*
* REGISTER DESIGNATIONS
*

R0	EQU	0
R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6
R7	EQU	7
R8	EQU	8
R9	EQU	9
R10	EQU	10
R11	EQU	11
R12	EQU	12
R13	EQU	13
R14	EQU	14
R15	EQU	15

* CHARACTER CODES

END	EQU	-1	END-OF-FORMAT
MINUSIGN	EQU	X'72'	NEGATIVE CONSTANT SIGN
QUADQUOT	EQU	X'52'	QUOTE-QUAD
UNDRSCOR	EQU	X'6D'	UNDERSCORE
BOSCODE	EQU	37	BEGINNING OF STATEMENT

* FORMAT SPECIFIER VALUES:BITS

IFORMAT	EQU	1	FORMAT CODES	I
FFORMAT	EQU	2		F
EFORMAT	EQU	3		E
AFORMAT	EQU	4		A
XFORMAT	EQU	5		X
TFORMAT	EQU	6		T
* QUALIFIER BIT FLAGS				
BBIT	EQU	1		B
CBIT	EQU	2		C
LBIT	EQU	4		L
ZBIT	EQU	8		Z
* DECORATION BITS				
MBIT	EQU	X'10'		M
NBIT	EQU	X'20'		N
PBIT	EQU	X'40'		P
QBIT	EQU	X'80'		Q
RBIT	EQU	X'100'		R

* DATUM TYPE CODES

LOGL	EQU	1	LOGICAL (BIT)
CHAR	EQU	2	TEXT (BYTE)
INTG	EQU	3	INTEGER (WORD)
FLOT	EQU	4	REAL (DW)
INDSEQ	EQU	5	INDEX SEQ-HANDLED AS INTEGER
LIST	EQU	6	LIST
	BOUND	8	
FESPEC	DATA	FFORMAT,EFORMAT	2,3
* ONE EQU BITMASK+1			
TWO	EQU	FESPEC	2
THREE	EQU	FESPEC+1	3
X1F	DATA	X'1F'	MASK
XFF	DATA	X'FF'	MASK

* TABLES FOR FORMAT SCAN ROUTINES

SPECTBL	TEXTC	'IFEAXBCLZMNPQR'	FORMAT, QUAL, AND DECOR SPEC'S.
* 1STQUAL EQU 6 INDEX OF 1ST QUALIFIER SPEC.			


```

1STDECOR EQU      10                      INDEX OF  1ST DECORATION SPEC.
*
QDBITS   EQU      %-1STQUAL
DATA     BBIT
DATA     CBIT
DATA     LBIT
DATA     ZBIT
DATA     MBIT
DATA     NBIT
DATA     PBIT
DATA     QBIT
DATA     RBIT
*
DEFAULTS EQU      %-1                      DEFAULT DECORATION DATA
DATA     BA(DSTR)      M TEXT
DATA     BA(DSTR)+1    N TEXT
DATA     BA(DSTR)+1    P TEXT
DATA     BA(DSTR)+1    Q TEXT
DATA     BA(DSTR)+2    R TEXT
DATA     1,0,0,0,1
*
DSTR      DATA,1    MINUSIGN,QUADQUOT,' ',QUADQUOT
*
* DELTAFMT-DYADIC SCALAR TO PROVIDE FORMATTED OUTPUT FOR PRINTING
* OR INTERNAL ASSIGNMENT
*
* THE FIRST SECTION OF CODE- DELTAFMT TO FMT9 -CONCERNS ANALYSIS OF
* ARGUMENTS AND GENERATION OF RESULT STRUCTURE
*
* THE LEFT ARGUMENT MUST BE A TEXT VECTOR, THE CONTENTS OF WHICH
* MUST BE VALID SPECIFICATION FORMS FOR OUTPUT CONVERSION AND
* FORMATTING
*
* THE RIGHT ARGUMENT MAY BE ANY VALID 'DATA' TYPE WITH RANK 2 OR
* LESS OR A 'LIST' OF RANK 1,WHOSE ELEMENTS ARE VALID DATA BLKS
*
* THE RESULT WILL BE A TEXT VECTOR OR MATRIX:
* A MATRIX IS POSSIBLE ONLY IF THE RIGHT ARGUMENT INDICATES
* MORE THAN ONE 'ROW' OF RESULT AND THE RESULT IS TO BE
* RETAINED-THAT IS-THE NEXT ITEM IN APL CS EXECUTION IS
* NOT 'BEGINNING OF STATEMENT'
*
CATQ      EQU      7                      EXECUTE OR EVALUATED INPUT
* CATEGORY FOR STATE ENTRY.
DELTAFMT  LOCAL    FMT1,FMT2,FMT3,FMT4,FMT5,FMT6,FMT7,FMT8,FMT9
LI,R1     CATQ      TEST FOR EXEC. OR EVAL INPUT
CB,R1     *STATEPTR
BE        SRETAIN   IF S0,SET TO RETAIN DATA
LW,R1     OFFSET    CHECK IF
AI,R1     -1        RESULT IS FOR
LB,R1     *CURRCS,R1 OUTPUT ONLY-THAT IS-
AI,R1     -BOSCODE  FOR BEGINNING OF STATEMENT
SRETAIN   STW,R1    RETAIN    0=DON'T RETAIN-DISPLAY ONLY
LB,R1     *LFARG    CHECK LEFT ARGUMENT
CI,R1     CHAR      FOR DOMAIN
BNE       ERDOMAIN  AND RANK
LH,R1     *LFARG
AI,R1     -CHAR*X'100'-1
BNEZ      ERRANK
STD,R1    NROWS      AND NCOLS -INITIALIZE MAX ROWS AND COLUMNS
BAL,R14   INITARG    SET UP FOR ARGUMENT SCAN
FMT1      BAL,R14   GETARG    GET NEXT ARGUMENT-INCLUDES RANK CHK
B         FMT3      NO MORE
CW,R8     NROWS
BLE       FMT2
STW,R8    NROWS      UPDATE MAXROWS IF INDICATED
FMT2      AWM,R9    NCOLS    UPDATE MAXCOLS
CI,R5     LIST      CHECK DATA TYPE
BL        FMT1      OK
B         ERDOMAIN  ERROR-LIST OR HIGHER
FMT3      LW,R8     NROWS
BEZ       NULRESLT   NO DATA-NULL RESULT
LI,R1     0
STW,R1    RESWIDTH   PRESET RESULT WIDTH TO 0
LW,R1     NCOLS      SET # OF COLUMNS
BEZ       NULRESLT   IF DATA COLUMN COUNT=0 GIVE NULL RES
BAL,R14   INITFORM   INITIALIZE FORMAT SCAN
FMT4      BAL,R12   FPHRASE  SCAN A FORMAT PHRASE

```

	AWM, R3	RESWIDTH	ADJUST RESULT WIDTH	
	CI, R4	AFORMAT	CHECK IF FORMAT USES DATUM	
	BG	FMT4	NO, KEEP SEARCHING	
	AI, R1	-1	YES-DECREMENT	
	BGZ	FMT4	LOOP IF ALL DATA COLUMNS NOT USED	
FMT5	BAL, R12	FPHRASE	DATA IS USED UP-CHECK FOR	
RESWCHK	CI, R4	AFORMAT	TRAILING NON-DATA FORMAT PHRASES	
	BLE	FMT6	NO-DATA USER ENCOUNTERED	
	AWM, R3	RESWIDTH	YES-KICK FIELD SIZE	
	B	FMT5	AND LOOP	
NOWRAP	RES	0	RETURN FROM FPHRASE IF 'END' REACHED	
FMT6	LW, R11	NROWS	NO. OF ROWS	U21-0008
	LW, R9	RETAIN	MATRIX-CHECK IF TO BE RETAINED	
	BNEZ	FMT7	YES	U21-0010
	LI, R11	1		
FMT7	LW, R8	R11	SAVE EFFECTIVE NROWS IN R8	U21-0012
FMT8	MW, R11	RESWIDTH	# OF BYTES	
	AI, R11	11	'BOUND 4' ROUND UP +8 BYTES	U21-0014
*			FOR LENGTH WORDS	U21-0015
	SAS, R11	-2	# OF WORDS	
	BAL, R7	ALOCHNW	GET DATA BLOCK	
	STW, R4	RESULT	ASSIGN TO RESULT	
	LW, R7	RESULT		
	AI, R7	4	POINT TO DATA	U21-0018
	LI, R2	CHAR*X'100'+2	CHARACTER MATRIX	U21-0019
	STH, R2	*RESULT	SET TYPE AND RANK	
	STW, R8	-2, R7	SET # OF ROWS	
	LW, R9	RESWIDTH	SET NO. OF COLUMNS	U21-0022
	STW, R9	-1, R7	SET # OF COLUMNS	
	SLS, R7	2		
	STW, R7	RESLTPTR	SET R7 TO POINT TO BA-RESULT	
	B	MAINSCAN		
	LOCAL			
NULRESLT	LI, R11	4	EMPTY RESULT-QUIT	
	BAL, R7	ALOCBLK		
	LI, R11	X'0201'	TEXT VECTOR	
	STH, R11	*R4		
	LI, R11	0		
	STW, R11	2, R4		
	STW, R4	RESULT		
	B	DXRETURN		
*				
*	* MAINSCAN-THIS IS THE DRIVER TO ACTUALLY GENERATE A RESULT OR RESULTS			
*	* USES THE INDICATED SUBROUTINES TO SCAN THE FORMAT STATEMENT			
*	* -REPEATEDLY IF NECESSARY-UNTIL DATA IN RIGHT ARGUMENT IS			
*	* EXHAUSTED			
*				
MAINSCAN	BAL, R14	INITARG	INITIALIZE FOR 'GETARG'	
	BAL, R14	INITDATA	INITIALIZE FOR 'GETDATUM'	
NEXTROW	BAL, R14	INITFORM	INITIALIZE FOR 'FPHRASE'	
	MTW, 1	ROWINDEX	INCREMENT ROW INDEX	
	LI, R8	1		
	STW, R8	ARGCOLS	INITIALIZE ARGUMENT COLUMN #	
	LD, R8	ARGWORDS	INITIALIZE ARG ADDRESS AND COUNT	
	STD, R8	ARGADR AND ARGCOUNT		
NXPHRASE	BAL, R12	FPHRASE	SCAN NEXT FORMAT PHRASE	
FTYPETBL	B	FTYPETBL, R4		@@@@@@@@
	B	IFORMGEN	IW	@
	B	FFORMGEN	FW.D	@
	B	EFORMGEN	EW.D	@
	B	AFORMGEN	AW	@
	B	XFORMGEN	XW	@
	B	TFORMGEN	'TEXT'	@@@@@@@@
ENDROW	LW, R13	BREAKFLG	CHECK FOR BREAK	
	BNEZ	OPBREAK	YES-GET OUT	
	LW, R8	ROWINDEX		
	CW, R8	NROWS		
	BGE	DXRETURN		
	BAL, R13	SENDROW	NO-OUTPUT ROW IF INDICATED	
	B	NEXTROW	PROCEED	
*				
*	* SENDROW-IF RETAIN=0 :OUTPUT THE CURRENT ROW			
*	* SINGOUT USED-SAVES & RESTORES REGISTERS			
*	* R7 SET TO START OF RESULT DATA BLOCK, WHICH			
*	* IS REUSED			
*	* IF RETAIN NOT=0, RETURNS WITH NO ACTION			
*				
SENDROW	LW, R1	RETAIN	CHECK IF DATA TO BE RETAINED	

```

        BNEZ      *R13          YES-NO DISPLAY
        LW,R4     RESULT
        BAL,R14   SINGOUT      OUTPUT RESULT
        B         OPBREAK      (ERROR RETURN-VERY UNLIKELY)
        LW,R7     RESLTPTR     RESET POINTER TO RE-USE
        B         *R13
        PAGE

*
*  TFORMGEN-TEXT GENERATION
*
TFORMGEN LW,R1     TEXTADR      ADDRESS
        LW,R2     R3           COUNT
        BAL,R14   PUTSTRNG     PUT'EM
        B         NXPHRASE     RETURN

*
*  XFORMGEN-OUTPUT BLANKS
*
XFORMGEN LW,R1     R3          COUNT
        BAL,R14   PUTBLNKS     PUT'EM
        B         NXPHRASE     RETURN

*
*  AFORMGEN-OUTPUT TEXT AND,AS INDICATED, BLANKS
*
AFORMGEN BAL,R13   GETDATUM     GET DATUM
        CI,R5     CHAR         CHECK FOR CHAR DATA
        BNE      ERDOMAIN
AFORM1   LW,R1     R3          GET W
        AI,R1     -1          CHECK IF 1
        BLEZ     AFORM2       YES-NO BLANKS
        BAL,R14   PUTBLNKS     NO-STUFF W-1 BLANKS
AFORM2   STB,R8    0,R7        STUFF THE CHARACTER
        AI,R7     1
        B         NXPHRASE     CONTINUE

*
*
*  EFORMGEN- E FORMAT PROCESSOR
*
*      GENERATES ONE FIELD ENTRY
*      DORS=NO OF DIGITS PRINTED
*      W=FIELD WIDTH
*
*      MAY EXIT TO OVERFLOW    NORMAL EXIT TO NXPHRASE
*
LOCAL    EFORM1,EFORM2,EFORM3,EFORM4,EFORM5
EFORMGEN BAL,R13   GETDATUM     GET A VALUE
        CI,R5     CHAR         CHECK FOR TEXT
        BE       ERDOMAIN     YES-ERROR
        LCW,R1    DORS        -S
        BIR,R1    EFORM0      1-S
        AI,R1     1           IF S=1, 2-5
EFORM0   AW,R1     R3          W+1-S OR,IF S=1, W+2-S
        LW,R0     VALFLAG
        BGEZ     EFORM1
        AI,R1     -1          SPACE FOR SIGNBIT
EFORM1   AI,R1     -6          R1=W-S-4(FOR 'E+XX')
*                                     -1(FOR DECIMAL PT-UNLESS S=1)
*                                     -1(FOR SIGN BIT IF VALUE NEGATIVE)
        BEZ      EFORM2      JUST RIGHT
        BLZ      ERFORMAT     W AND S NOT COMPATIBLE
        BAL,R14   PUTBLNKS     NEEDS LEADING BLANKS
EFORM2   LW,R0     VALFLAG
        BGEZ     EFORM3      CHECK FOR '-'
        LI,R0     MINUSIGN
        STB,R0    0,R7
        AI,R7     1
EFORM3   LW,R2     NSIZE       # OF DIGITS IN R8-R9 VALUE
        LW,R11    DORS        # OF DIGITS TO BE PRINTED
        LI,R12    1          # OF DIGITS LEFT OF DECIMAL
        BAL,R13   GENDIGSF     GENERATE DIGIT FIELD
        LI,R0     'E'         AND 'E'
        BAL,R14   SETCHAR
        STW,R7     EXPSTART     SAVE POSITION
        LW,R9     INTSIZE
        AI,R9     -1          GET EXPONENT SIZE
        BGEZ     EFORM4      +
        LI,R0     MINUSIGN    -
        BAL,R14   SETCHAR
        LAW,R9     R9

```

EFORM4	LI,R8	0	
	DW,R8	TENSTBL+1	DEVELOP 10'S VALUE
	BEZ	EFORM5	NONE
	AI,R9	X'F0'	GENERATE 10'S VALUE
	STB,R9	0,R7	
	AI,R7	1	
EFORM5	AI,R8	X'F0'	GENERATE UNIT'S VALUE
	STB,R8	0,R7	
	AI,R7	1	
	LW,R1	EXPSTART	GET POSITION OF EXPONENT
	SW,R1	R7	
	AI,R1	3	CHECK IF BLANK FILL NEEDED
	BLEZ	NXPHRASE	NO-EXIT
	LI,R14	NXPHRASE	YES-SET EXIT
	B	PUTBLNKS	GENERATE BLANKS
* IFORMGEN-			
* FFORMGEN-PROCESSORS FOR F AND I FORMATS			
* THIS ONE IS REALLY A GRINDER!			
* IN ADDITION TO FORMING THE EBCDIC DIGITS IN THE FIELD			
* 'DECORATORS' AND 'QUALIFIERS' ARE HANDLED			
* THIS INCLUDES-ACCORDING TO THE OPTIONS:			
* LEFT OR RIGHT JUSTIFICATION			
* INSERTION OF DECORATORS ON LEFT OR RIGHT			
* INSERTION OF COMMAS ALA- 2,345,678.9012345			
* BLANKING ZERO VALUES			
* INSERTING LEADING ZEROS			
*			
	LOCAL	FIF1,FIF2,FIF3,FIF4,FIF5,FIF6,FIF7,FIF8,FIF9,FIF10	
	LOCAL	FIF11,FIF12,FIF13,FIF14,FIF15	
	LOCAL	FIF16,FIF17,FIF18,FIF19,FIF20	
IFORMGEN	BAL,R13	GETDATUM	GET A VALUE
	CI,R5	CHAR	SCRUB ON TEXT
	BE	ERDOMAIN	
	LW,R1	R3	GET FIELD WIDTH
	B	FIFORMGN	
FFORMGEN	BAL,R13	GETDATUM	GET A VALUE
	CI,R5	CHAR	SCRUB ON TEXT
	BE	ERDOMAIN	
	LW,R1	R3	GET FIELD WIDTH
	AI,R1	-1	SPACE FOR '.'
	SW,R1	DORS	# OF FRACTIONAL DIGITS
FIFORMGN	CI,R10	BBIT	CHECK IF 'B' OPTION SET
	BAZ	FIF1	NO
	LW,R11	VALFLAG	YES
	BE	GAPFIELD	(BLANKS OR 'R' DECORATION)
FIF1	LW,R11	INTSIZE	
	CI,R11	1	GET # OF INTEGER DIGITS OR 1 IN R11
	BGE	FIF2	
	LI,R11	1	
FIF2	SW,R1	R11	ADJUST SIZE
	STW,R11	SAVEREAL	SAVE # OF INTEGERS
	LW,R2	VALFLAG	
	BGEZ	FIF3	
	SW,R1	MLENGTH	SPACE FOR M/N TEXT LENGTHS IF -
	SW,R1	NLENGTH	(DEFAULT IS MLENGTH=1,NLENGTH=0)
	B	FIF4	
FIF3	SW,R1	PLENGTH	SPACE FOR P/Q TEXT LENGTHS IF +
	SW,R1	QLENGTH	(DEFAULT IS PLENGTH=0,QLENGTH=0)
FIF4	BLZ	OVERFLOW	TOO DAMN BIG
	STW,R1	GAPSIZE	OK:REMEMBER GAP SIZE
	LI,R2	0	
	STW,R2	COMMACNT	SET COMMA COUNT TO ZERO
	CI,R10	CBIT	CHECK FOR COMMAS
	BAZ	FIF7	NO-THANK GOD!
	LI,R11	LBIT+ZBIT	YECCH!-CHECK FOR C.AND.Z.AND.NOT.L
	CS,R10	BITMASK+4	(ZBIT)AND NOT L
	BNE	FIF5	NO-HOORAY
	AW,R1	SAVEREAL	YES-THE GAP WILL BE LEADING ZEROS
	SAS,R1	-2	INFESTED WITH COMMAS
	B	FIF6	
FIF5	LW,R1	SAVEREAL	IN THIS CASE ONLY 1 COMMA FOR EVERY
	AI,R1	-1	3 INTEGER PART DIGITS
	DW,R1	THREE	(STARTING WITH THE 4TH DIGIT)
FIF6	STW,R1	COMMACNT	

21-00001

	LCW, R1	COMMACNT		
	AWM, R1	GAPSIZE	REDUCE THE GAP SIZE	
	BLZ	OVERFLOW	THE COMMAS BLEW IT !	
FIF7	CI, R10	LBIT+ZBIT		
	BANZ	FIF8	IF Z OR L SET, NO GAP AT LEFT	
	BAL, R14	PUTGAP		
FIF8	LW, R1	VALFLAG	CHECK VALUE	
	BLZ	FIF9	NEGATIVE	
	LW, R2	PLENGTH	POSITIVE	
	BLEZ	FIF11	SET PTEXT IF PRESENT (DEFAULT-NO)	
	LW, R1	PTEXT		
	B	FIF10		
FIF9	LW, R2	MLENGTH	SET MTEXT IF PRESENT (DEFAULT-'-')	
	BLEZ	FIF11		
	LW, R1	MTEXT		
FIF10	BAL, R14	PUTSTRNG		
FIF11	STW, R7	INTLEFT	SAVE POSITION OF 1ST INTEGER	
	AW, R7	COMMACNT	ADD SPACE FOR COMMAS IF PRESENT	
	LI, R11	LBIT+ZBIT	TEST FOR Z AND NOT.L	
	CS, R10	BITMASK+4	(Z, NOT L)	
	BNE	FIF12	NO	
	LW, R1	GAPSIZE	YES	
	BEZ	FIF12	NO GAP	
	BAL, R14	PUTZEROS	GAP IS LEADING ZERO'S	
FIF12	LI, R1	0		
	STW, R1	INTRIGHT	PRESET-NO INTEGER RIGHT POSITION	
	LW, R2	NSIZE	# OF CHARS IN R8-R9 TO CONVERT	
	LW, R12	INTSIZE	# OF CHARS LEFT OF '.'	
	LW, R11	SAVEREAL		
	AW, R11	DORS	R11=# OF DIGITS TO BE GENERATED	
	BAL, R13	GENDIGSF	DO IT	
	LW, R1	INTRIGHT	CHECK IF INTRIGHT SET	
	BNEZ	FIF13	YES	
	STW, R7	INTRIGHT	NO-SET IT	
	CI, R4	FFORMAT	CHECK IF F FORMAT	
	BNE	FIF13	NO-PROCEED	
	LW, R1	DORS	YES-CHECK IF D IS ZERO	
	BGZ	FIF13	NO-PROCEED	
	LI, R0	'.'	YES-REQUIRES DECIMAL POINT	
	BAL, R14	SETCHAR	INSERTION	
FIF13	LW, R1	VALFLAG	CHECK VALUE	
	BLZ	FIF14	NEGATIVE	
	LW, R2	QLENGTH	POSITIVE	
	BLEZ	FIF16	SET QTEXT IF PRESENT(DEFAULT-NO)	
	LW, R1	QTEXT		
	B	FIF15		
FIF14	LW, R2	NLENGTH	SET NTEXT IF PRESENT(DEFAULT-NO)	
	BLEZ	FIF16		
	LW, R1	NTEXT		
FIF15	BAL, R14	PUTSTRNG		
FIF16	CI, R10	LBIT	CHECK 'L'	
	BAZ	FIF17	NO	
	BAL, R14	PUTGAP	YES-PUT GAP HERE	
FIF17	CI, R10	CBIT	CHECK INSANITY	
	BAZ	NXPHRASE	NO-ESCAPE	
	LCW, R9	COMMACNT	YES-DRAT!	
	BEZ	NXPHRASE	NO COMMAS SET-SCRAM	
	AW, R9	INTRIGHT		
	SW, R9	INTLEFT	(INTRIGHT-INTLEFT-COMMACNT)	
	LI, R8	0		
	DW, R8	THREE	MODULO 3	
	AI, R8	0	CHECK FOR 0	
	BGZ	FIF17A	NO	21-00011
	LI, R11	LBIT+ZBIT	CHECK FOR (Z, NOT L)	21-00012
	CS, R10	BITMASK+4		21-00013
	BE	FIF17A	YES, COMMA MAY LEAD	21-00014
	LI, R8	3	NO, ADD 3, COMMA MAY NOT LEAD	21-00015
FIF17A	AW, R8	INTLEFT		21-00016
	STW, R8	COMMALOC	FIRST COMMA LOCATION	
	LW, R1	INTLEFT		
	AW, R1	COMMACNT	FIRST INTEGER CHARACTER	
FIF18	XW, R7	INTLEFT	SWITCH TARGET BYTE ADDRESS AND SAVE	
	CW, R7	COMMALOC	IS IT COMMA TIME	
	BNE	FIF19	NO-MOVE BYTES	
	LI, R0	'.'	YES	
	STB, R0	0, R7	STASH	
	AI, R7	1	A COMMA	
	MTW, 4	COMMALOC	BUMP LOC FOR NEXT COMMA	

```

        MTW, -1    COMMACNT          KNOCK COUNT
        BLEZ      FIF20              QUIT IF DONE
FIF19   LB, R0     0, R1              GET A DIGIT
        AI, R1     1
        STB, R0    0, R7              SET A DIGIT
        AI, R7     1
        B          FIF18              LOOP
FIF20   LW, R7     INTLEFT            RESTORE R7 TO END OF FIELD
        B          NXPHRASE          ESCAPE
        LOCAL
        PAGE

*
* INITFORM-INITIALIZE FORMAT (LEFT ARG) SCAN
* R14=LINK
* R6 IS SET AS POINTER TO FORMAT SCAN
* ---R6 IS RESERVED REGISTER FOR THIS PURPOSE---
* R9 IS USED-0 RETURNED ON EXIT
* RPTCOUNT SET=0
* DATASPEC SET=0
*
INITFORM LW, R6     LFARG              GET DB POINTER
        LW, R9     2, R6              AND LENGTH OF FORMAT STATEMENT
        SLS, R6    2                  FORM BYTE ADDRESS OF FIRST FORMAT
        AI, R6     X'8000C'           CHARACTER, SET HI-ENV NEG. FOR BIR
        AW, R9     R6
        STW, R9    FORMLIM            SET LIMIT OF SCAN (ALSO NEG. VALUE)
        LI, R9     0
        STW, R9    RPTCOUNT          INITIALIZE REPEAT COUNT
        STW, R9    DATASPEC           AND DATA SPECIFIER FLAG
        B          *R14

*
* FPHRASE-SCAN FORMAT PHRASE
*
* THIS ROUTINE SCANS A FORMAT PHRASE AND RETURNS INFORMATION
* FOR PROCESSING SAME:
*
* IF A PHRASE INCLUDES A REPEAT COUNT, SUBSEQUENT CALLS RETURN
* THE SAME INFORMATION UNTIL THE REPEAT COUNT IS EXHAUSTED
*
* R12=LINK          REGISTERS NOT TOUCHED: R1, R2, R5, R7, R8, R11, R15
* ROUTINES CALLED:
* FCHAR-R0, R6, R14
* FSTRING-R0, R3, R6, R13-FCHAR
* FNUM -R0, R6, R9, R13-FCHAR
*
ON EXIT:
* R4=FORMAT TYPE 1, I 2, F 3, E 4, A 5, X 6, TEXT
* R3=FIELD WIDTH
* R10=QUALIFIER AND DECORATOR FLAG BITS (FORMAT I OR F ONLY)
* BITS 23-31 ARE: 23 24 25 26 27 28 29 30 31
*                R Q P N M Z L C B
*
FOR DECORATION BIT, STRING POINTERS AND LENGTHS ARE ALSO SET:
* M ADDRESS IN MTEXT LENGTH IN MLENGTH
* N NTEXT NLENGTH
* P PTEXT PLENGTH
* Q QTEXT QLENGTH
* R RTEXT RLENGTH
*
DEFAULT STRINGS ARE ESTABLISHED FOR NON-SPECIFIED DECORATIONS
* 'DORS' IS SET TO D OR S VALUE FOR F OR E FORMAT TYPE
* IS SET TO ZERO FOR I FORMAT TYPE
        LOCAL      FPH1, FPH2, FPH3, FPH4, FPH5, FPH5A, FPH6, FPH7, FPH8, FPH9
        LOCAL      FPH10, FPH11, FPH12, FPH13
FPHRASE MTW, -1    RPTCOUNT          CHECK FOR A REPEATER
        BGZ      *R12                GOOD- SAVED SOME TIME HERE!
FPH1    BAL, R14  FCHAR              START NEW SCAN
FPH2    CLM, R0   LETTERS             CHECK 1ST CHARACTER
        BCS, 8    FPH4                DIGIT (OR ERROR)
        BCR, 1    FPH5                LETTER
        CI, R0    QUADQUOT           CHECK QUOTE-QUAD
        BE        FPH3                -START OF SUBSTRING
        CI, R0    END                 CHECK END
        BNE      ERFORMAT            NO VALID FORM
        LW, R9    DATASPEC           END-CHECK IF ANY DATA USE SPECIFIED
        BEZ      ERFORMAT            NONE-THATS A NO-NO
        CI, R12   RESWCHK            CHECK IF FPHRASE CALLED FOR
        BE        NOWRAP              RESULT WIDTH CALC. QUIT IF YES
        CI, R12   FTYPETBL           CHECK IF MAINSCAN CALL                21-00003
        BNE      WRAPPER              WRAPAROUND IF NOT                21-00004
        LW, R9    ARGCOLS            CHECK IF COLUMNS OF ARG USED UP    21-00005

```

	BDR, R9	WRAPPER	NO-WRAP AROUND	21-00006
	MTW, 0	ARGCOUNT	CHECK IF ARGUMENTS USED UP	21-00007
	BLEZ	ENDROW	YES	21-00008
WRAPPER	LW, R6	LFARG	NO, ITS WRAP AROUND TIME	21-00009
	SLS, R6	2		
	AI, R6	X'8000C'	SET POINTER TO START FROM BEGINNING	
	B	FPH1	AND GO-	
FPH3	BAL, R13	FSTRING1	PROCESS SUBSTRING	
	LI, R4	TFORMAT	SET FORMAT TYPE -TEXT	
	BAL, R14	FCHAR	GET FIELD TERMINATOR CHARACTER	
	B	FPH13	EXIT	
FPH4	BAL, R13	FNUM1	THIS HAS TO BE THE REPEAT COUNT	
	STW, R9	RPTCOUNT		
	AI, R9	0	CHECK SIZE	
	BGZ	FPH2	OK (NEXT CHAR IS IN R0)	
	B	ERFORMAT	NO DICE!	
FPH5	LI, R10	0	PRESET BIT FLAGS TO 0	
	STW, R10	DORS	SET 'D OR S' TO ZERO	
	LI, R4	RLENGTH-MTEXT+1		
FPH5A	LW, R9	DEFAULTS, R4	PRESET DEFAULTS FOR	
	STW, R9	MTEXT-1, R4	MTEXT THROUGH RLENGTH	
	BDR, R4	FPH5A		
FPH6	LB, R4	SPECTBL	(BYTE COUNT)	
FPH7	CB, R0	SPECTBL, R4	CHECK CONVERSION TYPE	
	BE	FPH8	FOUND IT	
	BDR, R4	FPH7		
	B	ERFORMAT	NONE-ERROR	
FPH8	CI, R4	1STQUAL	CHECK IF TYPE IS QUAL OR DECOR	
	BL	FPH10	NO-CLEAN CONVERSION	
	OR, R10	QDBITS, R4	YES-SET A FLAG BIT	
	CI, R4	1STDECOR	IS IT A DECOR	
	BL	FPH9	NO-	
	BAL, R13	FSTRING	YES-GET THE SUBSTRING	
	LW, R9	TEXTADR		
	STW, R9	DECORTEX, R4		
	STW, R3	DECORLEN, R4		
FPH9	BAL, R14	FCHAR	SCAN FOR NEXT FORMAT ID, QUAL	
	B	FPH6	OR DECOR.	
FPH10	CI, R4	FFORMAT	CHECK FORMAT ID	
	BLE	FPH11	I OR F	
	AI, R10	0	NOT I OR F-NO QUAL OR DECOR ALLOWED	
	BNEZ	ERFORMAT		
FPH11	BAL, R13	FNUM	GET FIELD WIDTH	
	LW, R3	R9		
	BLEZ	ERFORMAT	FIELD WIDTH MUST BE GR THAN ZERO	
	CLM, R4	FESPEC	IS THIS AN F OR E TYPE	
	BCS, 9	FPH12	NO	
	CI, R0	' '	YES-NEED A DOT HERE	
	BNE	ERFORMAT	BAD FORM	
	BAL, R13	FNUM	GET D OR S VALUE	
	STW, R9	DORS	SAVE IT	
	CI, R4	EFORMAT	CHECK IF E FORMAT	
	BNE	FPH12	NO	
	CI, R9	0	YES-CHECK S VALUE	
	BLE	ERFORMAT	MUST BE GR THAN ZERO	
FPH12	CI, R4	XFORMAT	CHECK FOR BLANK INSERTION	
	BE	FPH13	YES-NO DATA SPEC.	
	MTW, 1	DATASPEC	SET DATA SPEC. FLAG	
FPH13	CI, R0	' '	FORMAT PHRASE TERMINATOR	
	BE	*R12	OK	
	CI, R0	END		
	BE	*R12	OK	
	B	ERFORMAT	ERROR	
*				
*	FCHAR-GET NEXT NON-BLANK FORMAT CHARACTER			
*	IF THERE ARE NO MORE, RETURNS 'END'			
*	R14=LINK			
*	CHAR RETURNED IN R0			
*	R6, FORMAT POINTER, IS UPDATED			
*				
FCHAR	CW, R6	FORMLIM	CHECK FOR END	
	BL	FCHAR1	NO	
	LI, R0	END	YES	
	B	*R14		
FCHAR1	LB, R0	0, R6	GET CHAR	
	AI, R6	1	KICK POINTER	
	CI, R0	' '	CHECK FOR BLANK	
	BNE	*R14	NO	

```

*           B           FCHAR           YES-SKIP
*
* FNUM-SCAN FORMAT NUMBER
*           SCANS AND EVALUATES A DEC. NO. IN THE FORMAT STRING
* R13=LINK
*           ON EXIT, R9=VALUE OF NUMBER
*           R0=TERMINATING CHARACTER
*           R6(FORMAT POINTER) UPDATED
*
*           CALLS FCHAR, WHICH USES R0, R6, R14
* FNUM1-ENTRY USED WHEN FIRST CHARACTER ALREADY SCANNED AND IN R0
*
FNUM      LOCAL      FNUMLOOP
FNUM1     BAL, R14    FCHAR           GET 1ST CHAR
          CLM, R0     F0F9           RANGE CHECK
          BCS, 9      ERFORMAT       NO NUMBER WHEN REQUIRED
          LW, R9       R0
          AI, R9       - '0'         SET R9=VALUE
FNUMLOOP  BAL, R14    FCHAR           GET NEXT CHAR
          CLM, R0     F0F9           RANGE TEST
          BCS, 9      *R13          DONE
          AI, R0       - '0'         GET VALUE
          MI, R9       10            FORM RESULT
          BOV         ERFORMAT       TOO BIG
          AW, R9       R0
          BNOV        FNUMLOOP      LOOP FOR NEXT DIGIT
          B           ERFORMAT       TOO BIG
*
* FSTRING-SCAN FORMAT SUBSTRING-BETWEEN QUOTE-QUADS-
* R13=LINK
* RETURNS BA OF 1ST CHAR. IN 'TEXTADR'
* LENGTH OF STRING IN
* R6 IS UPDATED TO END OF SUBSTRING+1
* FSTRING1-ENTRY FOR USE WHEN LEFT QUOTE-QUAD ALREADY SCANNED
*
* CALLS FCHAR, WHICH USES R0, R6, AND R14
*
FSTRING   LOCAL      FSTRING2, FSTRING3
FSTRING   BAL, R14    FCHAR           GET FIRST CHAR
          CI, R0      QUADQUOT
          BNE        ERFORMAT       MUST BE QUOTE-QUAD
FSTRING1  STW, R6     TEXTADR         SAVE BA OF TEXT STRING
          LI, R0      QUADQUOT       SET TO TEST FOR END
FSTRING2  CW, R6      FORMLIM        CHECK OVERRUN
          BGE        ERFORMAT       YES-ERROR
          CB, R0      0, R6          CHECK END
          BE         FSTRING3        YES
          BIR, R6     FSTRING2       NO-LOOP
FSTRING3  LW, R3      R6
          SW, R3      TEXTADR         GET LENGTH
          BIR, R6     *R13
          PAGE
*
* INITIALIZE RIGHT ARGUMENT SCAN-SET-UP FOR GETARG
* R14=LINK
* R1, R2, R5, R8, R9 USED
*
* ON EXIT: R5=DATA TYPE
*
INITARG   LOCAL      INITARG1
INITARG   LI, R1      1
          STW, R1     ARGCOUNT      SET ARG COUNT=1
          LI, R1      RTARG
          STW, R1     ARGADR         SET ARG POINTER
          LB, R5      *RTARG
          CI, R5      LIST           CHECK FOR LIST
          BNE        INITARG1       NO-DONE!
          LH, R2      *RTARG         YES-DAMN!
          AI, R2      -LIST*X'100'-1
          BNEZ       ERRANK          RANK MUST BE 1
          LW, R1      RTARG          SET FOR ARGADR
          LW, R2      2, R1
          STW, R2     ARGCOUNT      ARGCOUNT=LENGTH OF LIST
          AI, R1      3
          STW, R1     ARGADR         ARG POINTER FOR 1ST ARG IN LIST
INITARG1  LD, R8      ARGADR         AND ARGCOUNT
          STD, R8     ARGWORDS       SET-UP FOR 'RESETARG'
          B           *R14          RETURN

```



```

*
* GETARG-GETS NEXT(OR ONLY) RIGHT ARGUMENT
* RETURNS INFORMATION PERTAINING TO ARGUMENT
* IF NO ARG, RETURNS TO CALL+1
* IF ARG, RETURNS TO CALL+2
* ON EXIT: R1=ARG ADDRESS
* R2=RANK
* R5=DATA TYPE 1 LOGL
* 2 CHAR
* 3 INTG
* 4 FLOT
* 5 INDS (INTEGER INDEX SEQUENCE)
* R8=NO. OF ROWS
* R9=NO. OF COLUMNS
*
* R14=LINK, R1, R2, R5, R8, R9 USED
*
GETARG MTW, -1 ARGCOUNT CHECK AND DECREMENT ARGUMENT COUNT
BLZ *R14 NONE-ZAP!
LW, R1 *ARGADR GET ARG POINTER
MTW, 1 ARGADR KICK POINTER-POINTER
LH, R2 *R1
AND, R2 XFF
CI, R2 2
BG ERRANK
EXU ROWSETBL, R2
EXU COLSETBL, R2
LB, R5 *R1
AI, R14 1
B *R14
*
ROWSETBL LI, R8 1 SCALAR
LW, R8 2, R1 VECTOR
LW, R8 2, R1 ARRAY
COLSETBL LI, R9 1 SCALAR
LI, R9 1 VECTOR
LW, R9 3, R1 ARRAY
*
* INITDATA-INITIALIZES PARAMETERS USED BY 'GETDATUM'
*
* R14=LINK R8, R9 USED
*
INITDATA LI, R8 0
STW, R8 ROWINDEX ROW INDEX=0
STW, R8 ARGCOLS SIMULATE 'LAST COLUMN' CONDITION
LD, R8 ARGWORDS INITIALIZE ARG ADR AND COUNT
STD, R8 ARGADR AND ARGCOUNT
B *R14
*
* GETDATUM-GET A SINGLE ELEMENT (OR NULL) FROM AN ARGUMENT
* DATA ARE ACCESSED COLUMN BY COLUMN (FOR MATRICES)
* THEN ADVANCE TO COLUMN 1 OF NEXT ARGUMENT
*
* R13=LINK CALLS:GETARG-WHICH USES R1, R2, R5, R8, R9 AND R14
* PLUSREAL-WHICH USES R2, R5, R8, R9, AND R12
* (R5 IS SAVED AND RESTORED)
*
* ON EXIT, R5=TYPE 1=LOGL 2=CHAR 3=INTG 4=FLOT
*
* IF CHAR, VALUE IS IN R8
* IF LOGL, INTG, OR FLOT, RESULT IS:
* VALUE AS LONG INTEGER IN R8-R9
* TOTAL # OF DIGITS (MAX=16) IN NSIZE
* # OF INTEGER DIGITS IN INTSIZE
*
* IF THE CURRENT ARGUMENT HAS FEWER ROWS THAN CURRENT ROW INDEX
* EXIT TO XFORMGEN-GENERATE BLANK FIELD
*
LOCAL GETD1, GETD2, GETD3, GETD4, GETD5, GETD6, GETD7
LOCAL GETD8, GETD9, GETD10
GETDATUM MTW, -1 ARGCOLS ANY MORE COLUMNS IN THIS ARG
BGZ GETD4 YES-EASY
BAL, R14 GETARG NO-NEXT ARGUMENT
B ENDROW NO MORE ARG'S THIS ROW
STD, R8 ARGROWS AND ARGCOLS
CW, R8 ROWINDEX IF ROWINDEX > ARGROWS
BL GETD5 THIS IS NULL

```

	CI, R5	INDSEQ	CHECK FOR INDEX SEQ	
	BE	GETD2	YES-SPECIAL CASE	
	MW, R9	ROWINDEX	COMPUTE OFFSET (IN ELEMENTS)	
	SW, R9	ARGCOLS	TO DESIRED ELEMENT	
	AW, R1	R2	ADD # OF LENGTH WORDS TO DB POINTER	
	AI, R1	2	OFFSET 2 HDR WORDS-POINT TO DATA	
	EXU	SETDATBL, R5	CONVERT TO 1ST ELEMENT ADDRESS	
GETD1	AW, R1	R9	OFFSET TO DESIRED ELEMENT	
	STW, R1	DATALOC	THAT'S IT!	
	B	GETD7		
GETD2	LW, R9	4, R1	'B'	
	MW, R9	ROWINDEX	'B'*'I'	
	AW, R9	3, R1	+'A'	
	LW, R8	R9	VALUE FOR IND SEQ ELEMENT	
	LI, R5	INTG	FAKE INTEGER DATA TYPE	
SETDATBL	B	CONVINT	CONVERT INTEGER DATA TYPE	@@@@@@@@
	B	GETD3	LOGIC-YIKES!	@
	SLS, R1	2	TEXT- SHIFT	@
	NOP		INTG-LEAVE ALONE	@
	B	%+1	FLOT-	@
	AI, R1	1	INSURE DW-BOUND	@
	SLS, R1	-1	SHIFT TO DW ADDRESS	@
	B	GETD1	CONTINUE	@@@@@@@@
GETD3	LI, R2	X'1F'	LOGICAL DATA:SEPARATE OFFSET INTO	
	AND, R2	R9	WORD OFFSET-R9	
	SLS, R9	-5	BIT # -R2	
	LW, R8	*R9, R1	GET WORD	
	SLS, R8	0, R2	POSITION TO BIT 0	
	STW, R8	LOGLBITS	SAVE VALUE (32 OF 'EM)	
	EOR, R2	X1F		
	AI, R2	1		
	STW, R2	LOGLCNT	LOGLCNT=32-BIT #	
	B	GETD1		
GETD4	LW, R8	ROWINDEX	CHECK IF ARG HAS THIS MANY ROWS	
	CW, R8	ARGROWS		
	BLE	GETD6	YES	
GETD5	B	XFORMGEN	EXIT-GENERATE BLANKS	
GETD6	LW, R1	DATALOC	GET ELEMENT ADDRESS	
GETD7	EXU	GETDATBL, R5	GET ELEMENT	
	MTW, 1	DATALOC	BUMP ELEMENT ADDRESS	
GETDATBL	B	CONVRTR	CONVERT DATUM	@@@@@@@@
	B	GETD8	LOGIC-YECH!	@
	B	GETD11	TEXT	@
	LW, R8	0, R1	INTG	@
	LD, R8	0, R1	FLOT	@@@@@@@@
GETD8	MTW, -1	LOGLCNT	LOGIC VALUE:ANY MORE BITS	
	BGEZ	GETD9	YES	
	LI, R9	31	NOT IN THIS WORD	
	STW, R9	LOGLCNT	RESET COUNTER	
	LW, R8	1, R1	GET NEXT WORD	
	MTW, 1	DATALOC	KICK ELEMENT ADDRESS	
	B	GETD10	SKIP	
GETD9	LW, R8	LOGLBITS	GET CURRENT WORD	
GETD10	SCS, R8	1	CYCLE	
	STW, R8	LOGLBITS	USE A BIT	
	AND, R8	ONE	MASK IT	
CONVLGL	STW, R8	VALFLAG	SAVE SIGN-ZERO FLAG	
	LW, R9	R8	SET VALUE IN R9	
	LI, R8	1		
CONVIORL	STW, R8	INTSIZE	SET INTEGER SIZE	
	STW, R8	NSIZE	SET NUMBER SIZE	
	LI, R8	0		
	B	*R13		
CONVINT	STW, R8	VALFLAG	SAVE SIGN-ZERO FLAG	
	LAW, R9	R8	GET ABS VALUE IN R9	
	LI, R2	-9		
CONVINT1	CW, R9	TENSTBL+10, R2	RANGE TEST	
	BL	CONVINT2	SIZE FOUND	
	BIR, R2	CONVINT1		
CONVINT2	AI, R2	10	INTSIZE	
	LW, R8	R2		
	CI, R13	EFORMGEN+1	CHECK IF E FORMAT	
	BNE	CONVIORL	NO-OK	
	SW, R2	DORS	YES-CHECK IF INT. MUST BE ROUNDED	
	BLEZ	CONVIORL	NO	
	LW, R2	TENSTBL, R2	YES-GET POWER OF 10	
	SLS, R2	-1	DIVIDE BY 2	
	AW, R9	R2	ROUND	

	LW, R2	R8	RESTORE INTSIZE	
	CW, R9	TENSTBL, R2	CHECK FOR OVERFLOW BY ROUND	
	BL	CONVIORL	NO	
	AI, R8	1	YES-KICK INTSIZE	
	B	CONVIORL		
CONVRTR	CI, R5	FLOT	CHECK DATA TYPE (REAL OR INTEGER)	
	BNE	CONVINT	INTEGER	
	STW, R8	VALFLAG	REAL	
	LAD, R8	R8	SET + VALUE	
	BEZ	CONVLGL	QUICK EXIT IF ZERO	
	CI, R13	EFORMGEN+1	CHECK E FORMAT	
	BE	CONVRL1	YES-ROUNDOFF IS FIXED BY DORS	
	STD, R8	SAVEREAL	NO-SAVE VALUE	
	BAL, R11	PLUSREAL	CONVERT TO GET DECIMAL EXPONENT	
	LW, R5	R12	CALCULATE NSIZE	
	AW, R5	DORS		
	BLZ	CONVRL5	ANOTHER ZERO CASE	
	BEZ	CONVRLH	0 TO 1, SPECIAL CASE.	U21-0024
	CI, R5	FLOT	CHECK FOR COINCIDENT CORRECT CONV.	
	BE	CONVRL4	YES	
	LD, R8	SAVEREAL	NO-RESTORE UNCONVERTED VALUE	
	B	CONVRL2	CONTINUE CONVERSION	
CONVRLH	LD, R8	SAVEREAL	RESTORE UNCONVERTED VALUE	U21-0026
	FAL, R8	FLHALF	ROUND IT	U21-0027
	FAL, R8	X4E1	FIX	U21-0028
	SW, R8	X4E1	IT	U21-0029
	LI, R5	1	SET NSIZE TO 1	U21-0030
	LI, R12	1	SET INTSIZE TO 1	U21-0031
	AI, R9	0	CHECK IF ZERO	U21-0032
	BNEZ	CONVRL4	NO-OK	U21-0033
	STW, R9	VALFLAG	YES-SET VALUE FLAG	U21-0034
	B	CONVRL4		U21-0035
CONVRL1	LW, R5	DORS	E-FORMAT, NSIZE=DORS	
CONVRL2	CI, R5	16	CHECK MAXIMUM NSIZE	
	BLE	CONVRL3	NO	
	LI, R5	16	YES-USE 16 MAX	
CONVRL3	BAL, R11	PLUSREAL	CONVERT	
CONVRL4	STW, R5	NSIZE	SAVE NSIZE	
	STW, R12	INTSIZE	AND INTSIZE	
	LI, R5	FLOT	RESTORE R5	
	B	*R13	EXIT	
CONVRL5	LD, R8	ZEROZERO	LOW VALUE-SET ZERO	
	LI, R5	FLOT	RESET R5	
	B	CONVLGL		
GETD11	LB, R8	0, R1	TEXT	
	MTW, 1	DATALOC	UPDATE DATA LOCATION	
	B	*R13	EXIT	
*				
* PUTSTRNG-STUFF A TEXT STRING				
* R2=COUNT R0 USED INTERNALLY				
* R1=BA(TEXT)				
* R14=LINK				
* R7=BA CURRENT RESULT				
*				
PUTSTRNG	LB, R0	0, R1		
	STB, R0	0, R7		
	AI, R7	1		
	AI, R1	1		
	BDR, R2	PUTSTRNG		
	B	*R14		
*				
* PUTBLNKS-STUFF N BLANKS R1=N				
* R14=LINK R0 USED INTERNALLY				
* R7=BA CURRENT RESULT				
PUTBLNKS	LI, R0	' '		
PUTIT	STB, R0	0, R7		
	AI, R7	1		
	BDR, R1	PUTIT		
	B	*R14		
*				
* PUTZEROS-STUFF N ZEROS R1=N				
*				
PUTZEROS	LI, R0	'0'		
	B	PUTIT		
*				
* PUTGAP-FILLS GAP WITH BLANKS, OR, IF RBIT SET, WITH 'RTEXT'				
* IF USED, 'RTEXT' IS REPEATED AS REQUIRED TO FILL THE GAP				
*				

```

*   R14=LINK  R0,R1,R2,R7,R11 USED
*
*   LOCAL      PUTGAP1,PUTGAP2,PUTGAP3
GAPFIELD CI,R10  RBIT      CHECK IF 'R' SET
          BAZ      XFORMGEN  NO-EASY OUT
          STW,R3    GAPSIZE   YES-GAPSIZE=FIELD WIDTH
          LI,R14    NXPHRASE  EXIT TO NXPHRASE
PUTGAP   LW,R1     GAPSIZE   GET GAPSIZE
          BLEZ      *R14     EMPTY GAP
          CI,R10    RBIT      CHECK IF RBIT IS SET
          BAZ      PUTBLNKS  NO
          B         PUTGAP2  ENTER LOOP
PUTGAP1  BDR,R11   PUTGAP3    DECREMENT R-COUNT,SKIP IF NOT DONE
PUTGAP2  LW,R11    RLENGTH   R-COUNT
          LW,R2     RTEXT     R-ADDRESS
PUTGAP3  LB,R0     0,R2      GET 'R' BYTE
          STB,R0    0,R7     STASH IT
          AI,R7     1         UPDATE
          AI,R2     1         POINTERS
          BDR,R1    PUTGAP1   DECREMENT GAP COUNT-LOOP IF NOT DONE
          B         *R14
*
*   OVERFLOW-REACHED FROM E-F-OR I FORMAT GEN IF CONVERSION WON'T FIT
*   R3=FIELD WIDTH
*   EXIT TO NXPHRASE
OVERFLOW LI,R0     '*'      OVERFLOW CHARACTER
          LW,R1     R3       WIDTH
          LI,R14    NXPHRASE EXIT LOC
          B         PUTIT    ENTER LOOP
*
*   GENDIGSF-ROUTINE TO GENERATE DIGITS FOR DELTAFORMAT
*
*   R13=LINK
*
*   ON ENTRY  R2=# OF DIGITS IN R8-R9 VALUE TO BE CONVERTED
*             R11=# OF DIGITS+UNDERSCORES OR BLANKS TO BE GENERATED
*             R12=# OF DIGITS LEFT OF DECIMAL
*             R8-R9=VALUE TO BE CONVERTED
*             R7= BA TO STORE DIGITS
*
*   REGISTERS NOT USED:R3(W) R4(FORMAT TYPE)R5(DATUM TYPE)
*                       R6(FORMAT POINTER)R10(FLAG BITS)
*
*   REGISTERS USED AND VOLATILE:R0,R1,R2,R8,R9,R11,R12,R14
*
*   ON EXIT   R7= BA FOR NEXT FIELD ITEM
*
*   LOCAL      GENDIG0,GENDIG1,GENDIG2,GENDIG3,GENDIG4,GENDIG5
*   LOCAL      GENDIG6,GENDIG7,GENDIG8,GENDIG9,GENDIG10,GENDIG11
*   LOCAL      GENDIG12,GENDIG13,GENDIG14,GENDIG15
GENDIGSF AI,R12    0         TEST 'INTSIZE'
          BGZ      GENDIG4   INTEGER CHARS EXIST
          BAL,R14  SETZER     NO INTEGER CHARS-INSERT 0
          STW,R7   INTRIGHT   SAVE END OF INTEGER POSITION
          LI,R14   GENDIG1
GENDIG0  LI,R0     '.'
          BDR,R11  SETCHAR    F OR E FORMAT BRANCHES
          B         *R13      I FORMAT EXIT FOR 0
GENDIG1  AI,R12    0         CHECK FOR FRACT. WITH LEADING ZEROS
          BEZ      GENDIG4   NO
          LI,R14   GENDIG3   YES
          AI,R12   -1        ADJUST COUNT FOR BIR LOOP
GENDIG2  BIR,R12   SETZER     SET ZEROS
          B         GENDIG4
GENDIG3  BDR,R11   GENDIG2    DECREMENT 'TSIZE' FOR EACH ZERO
          B         *R13      EXIT IF TSIZE REACHED
GENDIG4  LI,R1     0
          STW,R1   LDFLAG     RESET LONG-DIVIDE FLAG
          LI,R1    16         SET SIGNIFICANCE COUNT
          CI,R2    10         CHECK MSIZE
          BLE     GENDIG6     SHORT DIVIDE
          MTW,1    LDFLAG     LONG DIVIDE-SET FLAG
          DW,R8    TENSTBL+9  DIVIDE
          STW,R8   LOWEND     SAVE LOW END
          AI,R2    -9         ADJUST DIVIDE INDEX
GENDIG5  LI,R8     0         SET FOR DIVIDE
GENDIG6  DW,R8     TENSTBL-1,R2
          XW,R8     R9        DIVIDE OUT A DIGIT

```

	AI, R8	X'F0'	FORM EBCDIC CHAR
	STB, R8	0, R7	STASH IT
	AI, R7	1	
	AI, R1	-1	DECREMENT SIGNIFICANCE COUNT
	BDR, R11	GENDIG7	DECREMENT CHAR COUNT
	B	*R13	EXIT IF DONE
GENDIG7	AI, R12	-1	DECREMENT DECIMAL POSITION
	BNEZ	GENDIG8	NOT TIME FOR '.'
	STW, R7	INTRIGHT	TIME FOR '.'-SAVE POSITION
	LI, R0	'.'	
	BAL, R14	SETCHAR	SET '.'
GENDIG8	BDR, R2	GENDIG5	LOOP
	MTW, -1	LDFLAG	CHECK LONG-DIVIDE FLAG AND RESET
	BLZ	GENDIG9	SHORT DIVIDE(OR 2ND LOOP DONE)
	LI, R2	9	SETUP FOR 2ND LOOP
	LW, R9	LOWEND	GET LOW END OF VALUE
	B	GENDIG5	DO 2ND LOOP
GENDIG9	LI, R14	GENDIG10	STUPID LONG FIELD CASE!
GENDIG10	AI, R12	-1	ADJUST AND TEST DECIMAL POSITION
	BGZ	GENDIG13	INTEGERS LEFT
	BLZ	GENDIG15	FRACTION LEFT
	LI, R14	GENDIG12	AT '.' POINT
GENDIG11	AI, R1	-1	DECREMENT SIGNIFICANCE AND TEST
	BGEZ	SETZER	OK-GEN ZERO
	LI, R0	UNDRSCOR	16+ GEN UNDERSCORE
	B	SETCHAR	
GENDIG12	LI, R14	GENDIG10	GENERATE '.'
	B	GENDIG0	OR QUIT IF I FORMAT
GENDIG13	LI, R14	GENDIG14	
	B	GENDIG11	
GENDIG14	BDR, R11	GENDIG10	
	B	*R13	
GENDIG15	XW, R11	R1	SWITCH TSIZE AND SIGNIF. COUNT
	LW, R14	R13	SET EXIT
	AI, R11	0	CHECK IF SIG. CHARS. REMAIN
	BLEZ	PUTBLNKS	NO-PUT ALL BLANKS
	CW, R1	R11	CHECK IF SIG CHARS EXCEED TSIZE
	BLE	PUTZEROS	YES-PUT ALL ZEROS
	XW, R1	R11	NO-MIXED BAG
	SW, R11	R1	SAVE BLANK COUNT IN R11
	BAL, R14	PUTZEROS	PUT SOME ZEROS
	B	GENDIG15	R1=0 HERE-NEXT PASS PUTS THE BLANKS
SETZER	LI, R0	X'F0'	
SETCHAR	STB, R0	0, R7	SET A CHARACTER
	AI, R7	1	
	B	*R14	
	LOCAL		
	END		

TITLE 'MAT-B00,08/22/73,DWG702985'
PAGE

E X T E R N A L C O M M U N I C A T I O N

DEFINITIONS

DEF	DMATDIV	DYADIC MATRIX DIVIDE OP ROUTINE
DEF	MMATINV	MONADIC MATRIX INVERT OP ROUTINE
DEF	MAT@	START OF PROCEDURE

REFERENCES

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS
REF	ALOCRS	ALLOCATE RESULT DATA BLOCK
REF	DREF	DE-REF
REF	DTYPEF	DYADIC TYPE CHECK: FLOT RESULT
REF	DXRETURN	DYADIC EXECUTION DRIVER RETURN
REF	ERLENGTH	LENGTH ERROR
REF	ERRANK	RANK ERROR
REF	ERSING	SINGULAR MATRIX ERROR EXIT
REF	EXECUTE	EXECUTE XSEG
REF	FLOT0	FLOATING POINT 0.0
REF	FLOT1	FLOATING POINT 1.0
REF	FSQRT	F SQUARE ROOT EVAL
REF	GENLOAD	GEN LOAD BY RSTYPE
REF	GIVEBACK	SHRINK DATA BLOCK
REF	GXSEGINI	GEN XSEG INITIALIZATION
REF	LFARG	LEFT ARG PNTR
REF	LOOPLOC	LOOP LOCATION
REF	MTYPEF	MONADIC TYPE CHECK: FLOT RESULT
REF	MXRETURN	MONADIC EXECUTION DRIVER RETURN
REF	OPBREAK	OP BREAK HANDLER
REF	RESULT	RESULT PNTR
REF	RETURN	RETURN ADR CELL
REF	RSADR	RESULT ADDRESS
REF	RSRANK	RESULT RANK
REF	RSSIZE	RESULT SIZE
REF	RTARG	RIGHT ARG PNTR
REF	SETADR	SET UP ARG ADR CELL
REF	XSEGBASE	XSEG BASE
REF	XSEGBRK	XSEG BREAK FLAG
REF	XSETUP	SET ADR FOR INDEXED ACCESS
	PAGE	

A S S E M B L Y P A R A M E T E R S

SYSTEM	SIG5F
PROGSECT CSECT	1

REGISTERS

N	EQU	1	XSEG EXECUTION-TIME INDEX
N1	EQU	4	XSEG EXECUTION REG
X	EQU	1	GENERAL INDEX REG
K	EQU	2	SUBSCRIPT REG
T	EQU	2	TYPE
XL	EQU	3	XSEG LOC
A	EQU	4	ARG ADR/INDEX
LX	EQU	5	INDEX LINK REG
LX7	EQU	7	INDEX LINK REG
GIJ	EQU	1	INDEX TO ACCESS G(I,J)
GIK	EQU	2	INDEX TO ACCESS G(I,K)
GKI	EQU	3	INDEX TO ACCESS G(K,I)
VI	EQU	2	INDEX TO ACCESS V(I)
RI	EQU	2	INDEX TO ACCESS R(I)
FJ	EQU	3	INDEX TO ACCESS F(J)
FI	EQU	3	INDEX TO ACCESS F(I)
X0	EQU	2	EVEN/ODD INDEX PAIR
X1	EQU	3	*
AIJ	EQU	4	INDEX TO ACCESS A(I,K)
AF	EQU	6	ACCUM FOR FLOT VALUES
AF1	EQU	7	*
BF	EQU	8	2ND ACCUM FOR FLOT VALUES

BF1	EQU	9	*
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS
R	EQU	8	GENERAL WORK REG
S	EQU	11	SIZE
NI	EQU	12	I-LOOP COUNT
NJ	EQU	13	J-LOOP COUNT
L3	EQU	12	LINK REG
L2	EQU	13	LINK REG
L1	EQU	14	LINK REG
	PAGE		

*

*

* P R O C S

*

*

*

*

(THIS MODULE RESIDES IN AN OVERLAY, SO THE USUAL
TEMP ALLOCATION PROCEDURE WILL NOT WORK).

TLOC SET 0
MATTEMPS EQU XSEGBASE+50

*

TEMP CNAME 1
DTEMP CNAME 2

PROC
DO1 NAME=2
TLOC SET TLOC+(TLOC&1)
DISP TLOC+50
LF EQU MATTEMPS+TLOC
TLOC SET TLOC+NAME
PEND

*

*

EVEN CNAME 0
ODD CNAME 1
PROC
LF EQU %
ERROR,1,(CF(2)+NAME)&1 'REGISTER HAS WRONG PARITY'
PEND

*

*

EQUAL CNAME
PROC
LF EQU %
ERROR,1,1-(CF(2)=CF(3)) 'REGISTERS MUST BE EQUAL'
PEND

*

*

NB CNAME X'680'
NBGE CNAME X'681'
NBLE CNAME X'682'
NBE CNAME X'683'
NBL CNAME X'691'
NBG CNAME X'692'
NBNE CNAME X'693'
PROC
ERROR,1,(AF>=0)+(NUM(AF)>1) 'AF MUST BE NEG CONST ADR'
LF GEN,12,20 NAME-1,AF
PEND
PAGE

*

*

* XSEG GEN PROCS

*

*

GEN OPEN GEN
CNAME
OPEN M,N,MN,I
PROC
LF EQU %
ERROR,1,1-(NUM(CF)=3) 'WRONG NUMBER OF CF ARGS'
M SET CF(2)
N SET CF(3)
MN SET M+N
ERROR,1,1-(NUM(AF)=(M>0)+(N>0)) 'WRONG NUMBER OF AF ARGS'
DO M>0
I DO N
LW,AF(1)+M+I-1 AF(2)+I-1
FIN


```

I      DO      MN*(MN<3)
      STW,AF(1)+I-1  I-1,XL
      ELSE
      LCI      MN
      STM,AF(1)  0,XL
      FIN
      ELSE
I      DO      MN*(MN<3)
      LW,BUF   AF(1)+I-1
      STW,BUF  I-1,XL
      ELSE
      LCI      N
      LM,BUF   AF(1)
      STM,BUF  0,XL
      FIN
      FIN
      AI,XL    MN
      PEND
      CLOSE   M,N,MN,I
      PAGE

```

```

*
*
*  M A T R I X      I N V E R T / D I V I D E      O P      R O U T I N E S
*

```

```

MAT@      USECT    PROGSECT
RES        0              START OF PROCEDURE
*
*
MMATINV    EQU      %
BAL,L1     MTYPEF
LW,A       RTARG
LW,R       2,A
B          MATOP1
          CONTINUE LIKE MATRIX DIVIDE,
          PRETENDING THERE WAS AN M-BY-M
          LEFT ARG.
*
*
*
DMATDIV    EQU      %
BAL,L1     DTYPEF
LI,X       1
LB,R       *LFARG,X
LW,A       LFARG
CLM,R      10R2
BCS,9      ERRANK
BCR,3      1Z1
LW,R       3,A
BEZ        ERLLENGTH
          IF RANK =1, SET P=1
          RANK =2, P = 2'ND DIMEN
MATOP1     EQU      %
1Z1        STW,R     PVAL
          LW,R       2,A
          STW,R      MVAL
          LW,A       RTARG
          LI,X       1
          LB,X       *RTARG,X
          CI,X       2
          BNE        ERRANK
          STW,X      RSRANK
          CW,R       2,A
          BNE        ERLLENGTH
          LW,S       3,A
          BEZ        ERLLENGTH
          CW,S       MVAL
          BG         ERLLENGTH
          STW,S      NVAL
          AW,S       PVAL
          STW,S      ROWSIZE
          ODD,S
          MW,S       MVAL
          COMPUTE ROW SIZE OF G
          (M BY N+P).
          COMPUTE SIZE OF G
          NOTE: THIS SIZE IS >0 SINCE
          M*(N+P) > M*N >= N*N > 0.
          WE'LL ALOCATE THE G DATA BLOCK
          IN RESULT POSITION. AT THE MOMENT,
          IT'S BIGGER THAN THE FINAL RESULT;
          WE'LL PARE IT DOWN LATER.
*
*
          STW,S      RSSIZE
          BAL,L1     ALOCRS
*
*
          LW,R       NVAL
          STW,R      2,A
          LW,R       PVAL
          STW,R      3,A
          SET RESULT DIMENS:
          N-BY-P.

```

	BDR, R	1Z5	IF P=1,
	LI, X	1	REDUCE RESULT RANK TO 1
	STB, X	*RESULT, X	(2ND DIMEN BECOMES GAP WORD).
1Z5	LW, A	LFARG	
	BEZ	1Z2	IF LFARG EXISTS (IE., MATRIX DIV),
	LI, A	0	SET ITS ADR FOR INDEXED LOAD.
	BAL, LX	XSETUP	
1Z2	LI, A	1	SET RTADR FOR INDEXED LOAD
	BAL, LX	XSETUP	
	LI, X	-1	SET RSADR FOR SEQUENTIAL STORE
	BAL, LX	SETADR	
	BAL, LX	GXSEGINI	INIT XSEG CODE (RESULT ISN'T NULL)
	GEN, 0, 3	CODE1	GEN XSEG TO BUILD G;
	LI, A	1	EITHER G = B, A (CATENATED
	BAL, L1	GENLOAD	OR G = B, I COLUMN-WISE)
	GEN, 0, 6	CODE2	WHERE A = LFARG,
	LW, R	RSADR	B = RTARG,
	AWM, R	-6, XL	I = IDENTITY MATRIX
	STW, XL	LOOPLOC	
	LW, A	LFARG	TEST DIVIDE/INVERT
	BNEZ	1Z3	
	GEN, 0, 8	CODE3	INVERT: GEN CODE TO BUILD IDENTITY
	AWM, XL	-5, XL	MATRIX, ROW BY ROW.
	AWM, XL	-7, XL	
	LW, R	PVAL	
	STW, R	TESTVAL	
	B	1Z4	
1Z3	LI, A	0	DIVIDE: GEN CODE TO APPEND ROWS
	BAL, L1	GENLOAD	OF LFARG.
	GEN, 0, 3	CODE5	
1Z4	LW, R	RSADR	
	AWM, R	-3, XL	
	B	EXECUTE	EXECUTE XSEG
*			
*			
*			
*			
*			
		XSEG CODE:	
	LCW, N	RSSIZE	0 INIT STORE INDEX / RESULT COUNT
CODE1	LI, K	0	1 INIT LEFT INDEX
	STW, K	KTEMP	2 INIT RIGHT INDEX
	LW, N1	NVAL	3 INIT LEFT LOOP COUNT
*	LOAD/CNV	RTARG(K)	4 LEFT LOOP: MOVE A ROW OF 'B'
CODE2	STD, AF	0 RESULT(N)	TO RESULT.
	AI, N	1	BUMP STORE AND
	AI, K	1	LEFT INDECES.
	BDR, N1	XSEGBASE+4	COUNT LEFT LOOP
	XW, K	KTEMP	SAVE LEFT INDEX, GET RIGHT INDEX
	LW, N1	PVAL	INIT RIGHT LOOP COUNT
*			
*			
*			
		FOR INVERT:	
*LOOPLOC	EQU	%	
CODE3	CW, N1	TESTVAL	RIGHT LOOP: COMPUTE A ROW OF 'I'
	NBNE	-4 (%+3)	
	LD, AF	FLOT1	ONE ELEMENT IS 1.0,
	NB	-3 (%+2)	
	LD, AF	FLOT0	ALL OTHERS ARE 0.0.
	STD, AF	0 RESULT(N)	MOVE TO RESULT (G)
	BIR, N	CODE4	BUMP RESULT INDEX
	B	XSEGDONE	EXIT WHEN G FILLED
CODE4	BDR, N1	*LOOPLOC	* COUNT RIGHT LOOP
	MTW, -1	TESTVAL	* END OF ROW, ADVANCE POSITION OF 1.0
	XW, K	KTEMP	* SAVE RIGHT INDEX (UNUSED), GET LEFT
	B	XSEGBASE+3	* GO DO LEFT LOOP
*			
*			
*			
		FOR DIVIDE:	
*LOOPLOC	LOAD/CNV	LFARG(K)	RIGHT LOOP: MOVE A ROW OF 'A'
CODE5	STD, AF	0 RESULT(N)	TO RESULT.
	BIR, N	CODE6	BUMP STORE INDEX
	B	XSEGDONE	EXIT WHEN G FILLED
CODE6	AI, K	1	* BUMP RIGHT INDEX
	BDR, N1	*LOOPLOC	* COUNT RIGHT LOOP
	XW, K	KTEMP	* SAVE RIGHT INDEX, GET LEFT
	B	XSEGBASE+3	* GO DO LEFT LOOP
	PAGE		
*			
*			

```

XSEGDONE EQU      %      WE NOW HAVE OUR INITIAL G MATRIX
LI,A          0      DON'T ALLOW BREAKS DURING ALOC'S
STW,A        XSEGBRK  AND DREF'S. (FLAG WAS SET
*              BY 'EXECUTE').

XW,A          LFARG
BEZ          2Z1      IF LFARG EXITS,
BAL,LX7      DREF      GET RID OF IT.
LI,A          0
2Z1          XW,A      RTARG      GET RID OF RTARG
BAL,LX7      DREF
LW,S          MVAL      ALOC DB FOR ROW-MAX VECTOR
SLS,S        1          R(1),...,R(M).
BAL,LX7      ALOCHNW
STW,A        LFARG
LI,R          X'0100'    SET IT TO LEGIT TYPE
STH,R        *LFARG
LW,S          NVAL      ALOC DB FOR SCALING FACTOR VECTOR,
SLS,S        1          F(1),...,F(N).
BAL,LX7      ALOCHNW
STW,A        RTARG
LI,R          X'0100'    SET IT TO LEGIT TYPE
STH,R        *RTARG
LI,R          OPBREAK    NOW THAT ALL MEMORY FIDDLING IS
STW,R        XSEGBRK    DONE, IT'S SAFE TO ALLOW BREAKS.
AI,A          2          GET PAST RTARG HEADER
SLS,A        -1
STW,A        F1ADR      SET DBLWORD ADR OF F(1)
LW,A          LFARG
AI,A          2          GET PAST LFARG HEADER
STW,A        V1ADR      SET WORD ADR OF V(1) (INTG.VECTOR)
SLS,A        -1
STW,A        R1ADR      SET DBLWORD ADR OF R(1)
LW,A          RESULT
AI,A          4          GET PAST RESULT HEADER AND DIMENS
SLS,A        -1
STW,A        G11ADR     SET DBLWORD ADR OF G(1,1)
PAGE

*
*
* DO SCALING
*
* ROW SCALING: FOR I=1 TO M,
* SET R(I) = MAX OF ABS(G(I,J)) FOR J=1 TO N
*
LW,RI        R1ADR
LW,GIJ       G11ADR
LW,NI        MVAL      I LOOP COUNT = M
LD,AF        FLOT0      INIT BIGGEST ELMT = 0.0
3Z1          STD,AF      BIGGEST    REMEMBER BIGGEST ELEMENT
LD,AF        FLOT0      I LOOP: INIT R(I)=0.0
STD,AF       0,RI
LW,NJ        NVAL      J LOOP COUNT = N
3Z2          LAD,AF      0,GIJ      J LOOP
CD,AF        0,RI      SET R(I)=MAX(R(I),ABS(G(I,J)))
BLE          3Z3
STD,AF       0,RI
3Z3          AI,GIJ      1          MOVE GIJ ALONG ROW
BDR,NJ       3Z2          COUNT J LOOP
LD,AF        BIGGEST    IS THIS STILL THE BIGGEST ?
CD,AF        0,RI      IF NOT,
BGE          3Z31        UPDATE IT.
LD,AF        0,RI
3Z31         AW,GIJ      PVAL      MOVE ACCROSS LAST P COLS, TO NXT ROW
AI,RI        1          MOVE RI TO NEXT ELMT
BDR,NI       3Z1          COUNT I LOOP
FML,AF       EPSILON    SET PIVOT TEST VALUE
STD,AF       PIVOTMIN    = BIGGEST ELEMENT * EPSILON.

*
* COLUMN SCALING: FOR J=1 TO N,
* SET F(J) = MAX OF ABS(G(I,J))/R(I)
* FOR I = 1 TO M.
*
LW,FJ        F1ADR
LW,GIJ       G11ADR
LW,NJ        NVAL      J LOOP COUNT = N
3Z4          STW,GIJ     G1JADR     J LOOP: REMEMBER ADR OF G(1,J)
LD,AF        FLOT0
STD,AF       0,FJ      INIT F(J)=0

```

```

LW,RI    R1ADR    INIT R TO R(1)
LW,NI    MVAL     I LOOP COUNT = M
3Z5      LAD,AF    0,GIJ    I LOOP
          FDL,AF    0,RI     A = ABS(G(I,J))/R(I)
          CD,AF     0,FJ     SET F(J)= MAX(F(J),A)
          BLE       3Z6
          STD,AF    0,FJ
3Z6      AW,GIJ    ROWSIZE    BUMP G DOWN COLUMN
          AI,RI     1         BUMP R
          BDR,NI    3Z5      COUNT I LOOP
          AI,FJ     1         BUMP TO NEXT F(J)
          LW,GIJ    G1JADR    RECALL ADR OF G(1,J)
          AI,GIJ    1         BUMP TO NEXT COL
          BDR,NJ    3Z4      COUNT J LOOP
*
*      APPLY SCALE FACTORS TO MATRIX COLUMNS:
*      FOR I=1 TO M, AND J=1 TO N,
*      SET G(I,J) = G(I,J)/F(J)
*
LW,GIJ    G11ADR
LW,NI    MVAL     I LOOP COUNT = M
3Z7      LW,FJ    F1ADR
LW,NJ    NVAL     J LOOP COUNT = N
3Z8      LD,AF    0,GIJ
          FDL,AF    0,FJ     SET G(I,J) = G(I,J)/F(J)
          STD,AF    0,GIJ
          AI,GIJ    1         BUMP G ALONG ROW
          AI,FJ     1         BUMP TO NEXT F
          BDR,NJ    3Z8      COUNT J LOOP
          AW,GIJ    PVAL     BUMP G PAST LAST P COLS, TO NXT ROW
          BDR,NI    3Z7      COUNT I LOOP
*
*      INITIALIZE COLUMN PERMUTATION VECTOR:
*      SET V(I) = I-1 FOR I = 1 TO N.
*
LW,X      NVAL     I LOOP COUNT = N
3Z9      STW,X    *V1ADR,X  V(I+1)=I
          BDR,X    3Z9      COUNT I LOOP
          STW,X    *V1ADR    V(1)=0
          PAGE
*
*
*      PERFORM HOUSEHOLDER TRANSFORMATIONS,
*      LEADING TO UPPER TRIANGULAR SYSTEM.
*
*      INITIALIZE K LOOP PARAMETERS
*      K RUNS FROM 1 TO N.
*
LW,GIJ    G11ADR    SET INIT (K=1) VALUES FOR:
STW,GIJ    GK1ADR    ADR(G(K,1)),
STW,GIJ    GKKADR    ADR(G(K,K)),
LW,R      MVAL
STW,R      MK1VAL    M-K+1,
LW,R      NVAL
STW,R      NK1VAL    N-K+1,
LI,R      0
STW,R      K1VAL     K-1.
*
*      SELECT PIVOT ELEMENT: FIND S, T SUCH THAT
*      ABS(G(S,T)) = MAX(ABS(G(I,J)))
*      FOR I = K TO M, J = K TO N.
*
KLOOP     LW,GIJ    GK1ADR    START G AT G(K,1)
          LD,AF     FLOT0     INIT ABS(G(S,T)) = 0
          STD,AF    ABSGST
          LW,NI     MK1VAL    I LOOP COUNT = M-K+1
4Z2      AW,GIJ    K1VAL     I LOOP: SET G TO G(I,K)
          LW,NJ     NK1VAL    J LOOP COUNT = N-K+1
4Z3      LAD,AF    0,GIJ     J LOOP:
          CD,AF     ABSGST    IF ABS(G(I,J)) > ABSGST,
          BLE       4Z4
          STD,AF    ABSGST
          STW,GIJ    GSTADR    SET ABSGST = ABS(G(I,J)),
          AI,GIJ    1         AND REMEMBER S AND T.
4Z4      BDR,NJ    4Z3      BUMP G ALONG ROW
          AW,GIJ    PVAL     COUNT J LOOP
          BDR,NI    4Z2      BUMP G PAST LAST P COLS TO NEXT ROW
          LD,AF     ABSGST    COUNT I LOOP

```

```

CD,AF      PIVOTMIN      IF PIVOT ELMT DANGEROUSLY LOW,
BLE        ERSING        PROCLAIM THE MATRIX SINGULAR.

*
*      PERMUTE COLUMNS K/T: FOR I = 1 TO M,
*      EXCHANGE G(I,K) AND G(I,T),
*      EXCHANGE V(K) AND V(T).
*

LW,X1      GSTADR        GET ADR(G(S,T))
LI,X0      0
EVEN,X0
EQUAL,X0+1,X1
SW,X1      G11ADR        SEPARATE S,T INDEX INTO DISTINCT
DW,X0      ROWSIZE      ROW & COL INDECEES:
STW,X0     T1VAL        X0 = T-1,
                        X1 = S-1.
*
CW,X0      K1VAL        IF T=K, NO EXCHANGE NEEDED
BE         4Z6
LW,GIJ     K1VAL        GET K-1
LW,R       *V1ADR,X0     EXCHANGE V(T) AND V(K)
XW,R       *V1ADR,GIJ
STW,R      *V1ADR,X0
AW,X0      G11ADR        COMPUTE ADR(G(1,T)),
AW,GIJ     G11ADR        AND ADR(G(1,K)).
LW,NI      MVAL        I LOOP COUNT = M
4Z5        LD,AF         0,GIJ      I LOOP
LD,BF      0,X0         EXCHANGE G(I,K) AND G(I,T)
STD,AF     0,X0
STD,BF     0,GIJ
AW,GIJ     ROWSIZE      BUMP G DOWN COLUMN
AW,X0      ROWSIZE      *
BDR,NI     4Z5          COUNT I LOOP
4Z6        EQU         %
*
*      PERMUTE ROWS S/K: FOR J = 1 TO N+P,
*      EXCHANGE G(S,J) AND G(K,J).
*

CW,X1      K1VAL        IF S=K, NO EXCHANGE NECESSARY
BE         4Z8
LW,X1      GSTADR
SW,X1      T1VAL        GET ADR(G(S,1)),
LW,GIJ     GK1ADR        AND ADR(G(K,1)).
LW,NJ      ROWSIZE      J LOOP COUNT = N+P
4Z7        LD,AF         0,GIJ      J LOOP
LD,BF      0,X1         EXCHANGE G(S,J) AND G(K,J)
STD,AF     0,X1
STD,BF     0,GIJ
AI,GIJ     1            BUMP G ALONG ROW
AI,X1      1            *
BDR,NJ     4Z7          COUNT J LOOP
4Z8        EQU         %
*
*      APPLY K'TH TRANSFORMATION:
*
*      S = SQRT(G(K,K)**2 + G(K+1,K)**2 + ... + G(M,K)**2)
*      U = G(K,K) + S*SGN(G(K,K))
*      B = 1/(S*S + S*ABS(G(K,K)))
*      FOR J = K+1 TO N+P,
*      Y = B*U*G(K,J)+B*(G(K+1,J)*G(K+1,K) + ...
*      ... + G(M,J)*G(M,K))
*      G(I,J):=G(I,J)-Y*G(I,K) FOR I=K+1 TO M
*      G(K,J):=G(K,J)-U*Y
*      G(K,K):=-S*SGN(G(K,K)) = G(K,K)-U
*

LD,AF      FLOT0        INIT SUM (FOR S**2) TO 0.0
LW,NI      MK1VAL
LW,GIJ     GKKADR        I LOOP COUNT = M-K+1
5Z0        STD,AF       S2VAL      START AT G(K,K)
LD,AF      0,GIJ        I LOOP
FML,AF     0,GIJ        ADD IN G(I,K)**2
FAL,AF     S2VAL
AW,GIJ     ROWSIZE      BUMP G DOWN COLUMN
BDR,NI     5Z0          COUNT I LOOP
BAL,LX     FSQRT        COMPUTE S=SQRT(S**2)
STD,AF     SVAL        SAVE S
FAL,AF     ABGKK        S+ABS(G(K,K))
FML,AF     SVAL        S**2 + S*ABS(G(K,K)) = 1/B
LD,BF      FLOT1

```

	FDL,BF	AF	B
	STD,BF	BVAL	
	LW,GIJ	GKKADR	GET ADR(G(K,K))
	LD,AF	0,GIJ	
	BGEZ	5Z1	IF G(K,K)<0,
	FSL,AF	SVAL	U = G(K,K)-S,
	LD,BF	SVAL	G(K,K):=S.
	B	5Z2	IF G(K,K)>0,
5Z1	FAL,AF	SVAL	U = G(K,K)+S,
	LCD,BF	SVAL	G(K,K):=-S.
5Z2	STD,AF	UVAL	U = G(K,K)+S*SGN(G(K,K))
	STD,BF	0,GIJ	G(K,K) := -S*SGN(G(K,K))
	LW,NJ	NK1VAL	
	AW,NJ	PVAL	J LOOP COUNT = N+P-K
	B	5Z8	
5Z3	STW,GIJ	GKJADR	J LOOP: SET ADR(G(K,J)),
	LW,GIK	GKKADR	AND ADR(G(K,K)).
	LD,AF	0,GIJ	START COMPUTATION OF Y:
	FML,AF	UVAL	AF = U*G(K,J).
	LW,NI	MK1VAL	I LOOP COUNT = M-K (POSSIBLY 0)
	B	5Z5	START I LOOP WITH I=K+1
5Z4	STD,AF	YVAL	I LOOP:
	LD,AF	0,GIK	AF=AF+G(I,K)*G(I,J).
	FML,AF	0,GIJ	
	FAL,AF	YVAL	
5Z5	AW,GIK	ROWSIZE	BUMP G DOWN COLUMN
	AW,GIJ	ROWSIZE	
	BDR,NI	5Z4	COUNT I LOOP
	FML,AF	BVAL	Y = B*AF
	STD,AF	YVAL	
	LW,GIK	GKKADR	RESTORE INIT G ADR'S
	LW,GIJ	GKJADR	
	LCD,AF	UVAL	G(K,J):=G(K,J)-Y*U
	LW,NI	MK1VAL	I LOOP COUNT = M-K (POSSIBLY 0)
	B	5Z7	START I LOOP WITH I=K+1
5Z6	LCD,AF	0,GIK	
5Z7	FML,AF	YVAL	G(I,J):=G(I,J)-Y*G(I,K)
	FAL,AF	0,GIJ	
	STD,AF	0,GIJ	
	AW,GIK	ROWSIZE	BUMP G DOWN COLUMN
	AW,GIJ	ROWSIZE	
	BDR,NI	5Z6	COUNT I LOOP
	LW,GIJ	GKJADR	UPDATE ADR(G(K,J)) BY
5Z8	AI,GIJ	1	BUMPING IT ALONG ROW K.
	BDR,NJ	5Z3	COUNT J LOOP
*			
*			
*			
*			
		UPDATE K LOOP PARAMETERS	
*	MTW,-1	NK1VAL	UPDATE N-K+1 VALUE
			IF N-K+1=0 (K>N),
	BEZ	SOLVE	WE'RE DONE.
	MTW,-1	MK1VAL	K<=N: UPDATE M-K+1,
	MTW,1	K1VAL	K-1,
	LW,S	ROWSIZE	
	AWM,S	GK1ADR	ADR(G(K,1)),
	AI,S	1	
	AWM,S	GKKADR	AND ADR(G(K,K)).
	B	KLOOP	CONTINUE K LOOP
	PAGE		
*			
*			
*			
*			
		SOLVE UPPER TRIANGULAR SYSTEM	
*			
*			
*			
*		FOR K = N TO 1, AND J = N+1 TO N+P, SET	
*		G(K,J) := (G(K,J) - G(K,K+1)*G(K+1,J) - ...	
*		... - G(K,N)*G(N,J))/G(K,K)	
*			
SOLVE	MTW,1	NK1VAL	RESTORE N-K+1 VALUE
	LW,GIJ	GKKADR	START G(K,N) AT G(N,N)
6Z1	STW,GIJ	GKNADR	REMEMBER ADR(G(K,N))
	AI,GIJ	1	START G(K,J) AT G(K,N+1)
	LW,NJ	PVAL	J LOOP COUNT = P
6Z2	STW,GIJ	GKJADR	REMEMBER ADR(G(K,J))
	LW,GKI	GKKADR	START G(K,I) AT G(K,K)
	LD,AF	0,GIJ	START SUM WITH G(K,J)
	LW,NI	NK1VAL	I LOOP COUNT = N-K (POSSIBLY 0)

```

6Z3      B      6Z4      START WITH I=K+1
        STD,AF  SUM
        LCD,AF  0,GKI    SUM := SUM-G(K,I)*G(I,J)
        FML,AF  0,GIJ
        FAL,AF  SUM
6Z4      AI,GKI  1        BUMP G(K,I) ALONG ROW
        AW,GIJ  ROWSIZE  BUMP G(I,J) DOWN COLUMN
        BDR,NI  6Z3      COUNT I LOOP
        LW,GIJ  GKKAADR  COMPUTE NEW G(K,J)
        FDL,AF  0,GIJ    = SUM/G(K,K).
        LW,GIJ  GKJADR
        STD,AF  0,GIJ
        AI,GIJ  1        STORE NEW G(K,J)
        BDR,NJ  6Z2      UPDATE ADR(G(K,J))
        MTW,-1  K1VAL    COUNT J LOOP
        BLZ     NORMALYZ  COUNT K LOOP (UPDATE K-1)
        MTW,1   NK1VAL    IF K=0, WE'RE DONE
        LCW,R   ROWSIZE  K>0: UPDATE N-K+1,
        AI,R    -1
        AWM,R   GKKAADR          ADR(G(K,K)),
        LW,GIJ  GKNAADR          AND ADR(G(K,N)).
        SW,GIJ  ROWSIZE
        B      6Z1
        PAGE

*
*
*   NORMALIZE THE SOLUTION TO CORRECT FOR
*   COLUMN SCALING AND INTERCHANGES.
*
*       FOR I = 1 TO N, AND J = 1 TO P,
*           EXCHANGE G(I,N+J) WITH G(L,N+J),
*           AND COPY V(I) TO V(L), WHERE
*           V(L)=I.
*
NORMALYZ LW,AIJ  G11ADR    START 'A' AT A(1,1)
        LW,NI   NVAL      I LOOP COUNT = N
        LI,VI   0         INIT TO V(1)
7Z1      CW,VI  *V1ADR,VI  IF V(I)=I, NO EXCHANGE NEEDED
        BNE     7Z10
        AW,AIJ  ROWSIZE   (IN WHICH CASE, BUMP TO NEXT ROW)
        B      7Z3
7Z10     LW,GIJ  VI        L:=I
7Z11     LW,GIJ  *V1ADR,GIJ L:=V(L)
        CW,VI   *V1ADR,GIJ SEARCH V FOR I: IF I.NE.V(L),
        BNE     7Z11      SET L:=V(L) AND TRY AGAIN.
        LW,R    *V1ADR,VI  COPY V(I)
        STW,R   *V1ADR,GIJ TO V(L).
        ODD,GIJ
        MW,GIJ  ROWSIZE   COMPUTE ADR(G(L,N+1))
        AW,GIJ  NVAL
        AW,GIJ  G11ADR
        AW,AIJ  NVAL      MOVE 'A' TO A(I,N+1)
        LW,NJ   PVAL      J LOOP COUNT = P
7Z2      LD,AF  0,GIJ     EXCHANGE G(L,N+J)
        LD,BF  0,AIJ     AND A(I,N+J).
        STD,AF  0,AIJ
        STD,BF  0,GIJ
        AI,GIJ  1        BUMP G ALONG ROW
        AI,AIJ  1        BUMP A ALONG ROW
        BDR,NJ  7Z2      COUNT J LOOP
7Z3      AI,VI  1        BUMP TO NEXT V(I)
        BDR,NI  7Z1      COUNT I LOOP

*
*
*       FOR I = 1 TO N, AND J = 1 TO P,
*           SET A(I,J) = G(I,N+J)/F(I).
*           'A' IS THE N-BY-P CORNER OF G IN
*           WHICH THE FINAL RESULT APPEARS.
*
        LW,AIJ  G11ADR    START A AT A(1,1)
        LW,GIJ  G11ADR    START G AT G(1,1)
        LW,FI   F1ADR     START F AT F(1)
        LW,NI   NVAL      I LOOP COUNT = N
7Z4      AW,GIJ  NVAL      START G(I,J) AT G(I,N+1)
        LW,NJ   PVAL      J LOOP COUNT = P
7Z5      LD,AF  0,GIJ     GET G(I,N+J)
        FDL,AF  0,FI      DIVIDE BY F(I)
        STD,AF  0,AIJ     STORE AS A(I,J)

```

```

AI,GIJ      1      BUMP G AND A
AI,AIJ      1      ALONG ROW.
BDR,NJ      7Z5    COUNT J LOOP
AI,FI       1      BUMP TO NEXT F(I)
BDR,NI      7Z4    COUNT I LOOP

*
*
*      NOW GIVE BACK THE M*(N+P)-N*P DOUBLEWORDS OF 'G' THAT
*      ARE NO LONGER NEEDED.
*
LI,A        0      MUSTN'T ALLOW BREAKS DURING MEMORY
STW,A       XSEGBRK MANAGEMENT OPERATIONS TO FOLLOW.
LW,S        NVAL   FINAL RESULT SIZE = N*P
MW,S        PVAL
XW,S        RSSIZE COMPUTE NR OF DBLWORDS TO GIVE BACK
SW,S        RSSIZE = OLD G SIZE - RESULT SIZE.
SLS,S       1      CHANGE TO NR OF WORDS
LW,A        RESULT
BAL,LX7     GIVEBACK GIVE THEM BACK
LI,A        DXRETURN IF DYADIC ENTRY USED, SIMPLY EXIT
CW,A        RETURN  DYADICALLY; BOTH LFARG (V(I))
BE          DXRETURN AND RTARG (F(J)) WILL BE DEREFFED.
LI,A        0      OTHERWISE, WE HAVE TO Deref
XW,A        LFARG   LFARG OURSELVES.
BAL,LX7     DREF
B           MXRETURN
PAGE

*
*
*      DATA / TEMPS
*
*
10R2        BOUND   8
EPSILON     DATA   1,2      RANGE 1 TO 2, FOR ARG RANK TEST
*           DATA   X'34100000',0 16**(-13) FOR SINGULARITY TEST
*
V1ADR       TEMP    WORD ADR OF V(1)
R1ADR       TEMP    DBLWORD ADR OF R(1)
F1ADR       TEMP    DBLWORD ADR OF F(1)
G11ADR      TEMP    DBLWORD ADR OF G(1,1)
GK1ADR      TEMP    DBLWORD ADR OF G(K,1)
GKKADR      TEMP    DBLWORD ADR OF G(K,K)
GKJADR      TEMP    DBLWORD ADR OF G(K,J)
GKNADR      TEMP    DBLWORD ADR OF G(K,N)
G1JADR      TEMP    DBLWORD ADR OF G(1,J)
GSTADR      TEMP    DBLWORD ADR OF G(S,T)
*
MVAL        TEMP    VALUE OF M
NVAL        TEMP    VALUE OF N
PVAL        TEMP    VALUE OF P
K1VAL       TEMP    VALUE OF K-1
T1VAL       TEMP    VALUE OF T-1
MK1VAL      TEMP    VALUE OF M-K+1
NK1VAL      TEMP    VALUE OF N-K+1
ROWSIZE     TEMP    VALUE OF N+P
TESTVAL     TEMP    INDEX OF NEXT DIAGONAL POS. (XSEG)
KTEMP       TEMP    LEFT/RIGHT ARG INDEX (XSEG)
*
ABSGST      DTEMP    VALUE OF ABS(G(S,T))
ABSGKK      EQU      ABSGST    = ABS(G(K,K)) AFTER EXCHANGES
S2VAL       DTEMP    SUM(I=K,M) OF G(I,K)**2
SVAL        EQU      S2VAL     SQRT OF ABOVE
BVAL        DTEMP    1/(S**2 + S*ABS(G(K,K)))
UVAL        DTEMP    G(K,K) + S*SGN(G(K,K))
YVAL        DTEMP    B*U*G(K,J)+B*SUM(G(I,K)*G(I,J))
SUM          DTEMP    G(K,J)-SUM(G(K,I)*G(I,J))
PIVOTMIN    DTEMP    PIVOT TEST VALUE
BIGGEST     DTEMP    BIGGEST ELEMENT OF MATRIX
*
*
7Z          END

```


	TITLE	'RCMD-B00,10/16/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	CONTROL CARDS NOT PRINTED.
RCMD@	RES	0	ORIGIN OF ROOT-CMD MODULE.
*			
*	REF'S	AND	DEF'S
*			
	DEF	RCMD@	= START OF 'RCMD' MODULE.
	DEF	SYMTOFF	SYMBOL TABLE OFFSET FROM LODYN
	DEF	STACKOFF	= OFFSET FROM HI COMMON TO XEQ STACK
	DEF	CMDREC	COMMAND RECOGNIZER.
	DEF	CMDB	COMMAND BRANCH VECTOR
	DEF	NCMDS	# OF SYSTEM COMMANDS
	DEF	TEXTCLEA	'CLEA'
	DEF	TEXTOFF	'OFF '
	DEF	@@CLEAR	
	DEF	@@CONT	
	DEF	@@COPY	
	DEF	@@LOAD	
	DEF	@@OFF	
	DEF	@@PCOPY	
	DEF	@@SAVE	
	DEF	XWLOCGLB	EXCHANGE LOCALS AND GLOBALS
	DEF	ACQCONLY	ACQUIRES CONSTANT ONLY.
	DEF	OUTORANG	
	DEF	VALOUT	PUT VALUE IN IMAGE, DISPLAY, & EXIT.
	DEF	ISVAL	DISPLAY 'IS ' & VALUE.
	DEF	GENIS	DISPLAY 'IS ' USING VALUE IN R7.
	DEF	ISMSG	'IS '
	DEF	WASMSG	'WAS '
	DEF	ERBDCMDA	BAD COMMAND WITH F:WS OPEN, PERHAPS.
	DEF	ERBADCMD	
	DEF	CMDERR	COMMAND ERROR
	DEF	CMDEXIT	
	DEF	CMDEXITO	CMD EXIT FOR ERR & ON-OFF CK.
	DEF	CMDEXITM	CMD EXIT VIA MODE.
	DEF	DONTSAVE	ERROR -- NOT SAVED, THIS WS IS ...
	DEF	ERRERASE	ERROR -- NOT ERASED.
	DEF	ERNOTFND	GENERATE NAME-NOT FOUND MESSAGE
	DEF	ERNOTCPY	GENERATE NAME-NOT COPIED MESSAGE
	DEF	SIDAME	ERROR -- SI DAMAGE (VIA ERASE).
	DEF	SIDAMC	ERROR -- SI DAMAGE (VIA COPY).
	DEF	SIDAMF	ERROR -- SI DAMAGE (VIA FUNDEF).
	DEF	ERRDEFN	ERROR -- DEFN ERR.
	DEF	LSCANERR	ERROR -- LINESCAN ERR.
	DEF	BBADFL	GO USE BAD FUN LINE, MAYBE.
	DEF	ERRFDSYM	ERROR -- SYM TBL FULL (VIA FUNDEF).
	DEF	ERRFDWS	ERROR -- WS FULL (VIA FUNDEF).
	DEF	BRFUNDEF	RETURN TO FUNDEF MODULE.
	DEF	CSERRH	HANDLE CODESTRING ERROR.
	DEF	BREPROMT	BRANCH TO REPROMPT.
	DEF	BCONTOFF	GO FOR)CONTINUE & OFF.
	DEF	B4CLOSE	GO FORCE CLOSE OF OPEN FUNCTION.
	DEF	OBSERVEZ	RESUME CODE EXEQ AFTER OBSERVATION.
	DEF	BCSN	GO TO CODESTRINGER FOR AUTOSTART.
	DEF	BAQCONLY	GO ACQUIRE CONSTANT ONLY.
	DEF	EBALS	TBL OF ENTRIES TO THE ERROR MODULE.
	DEF	ERRN	GO TO ERROR MOD. FOR ERR NOS.
	DEF	ERRF	GO TO ERROR MOD. FOR ERR FN'S NAME.
	DEF	ERRX	GO TO ERROR MOD. FOR I/O ERR CODE.
	DEF	CMDERRH	HANDLE COMMAND ERROR.
	DEF	ERRSET	HANDLE ERR IN EXECUTION.
	DEF	OBSERVER	GO TO ERROR MOD. FOR AN OBSERVATION.
	DEF	UNREF	GO TO ERROR MOD. FOR EXEC.ERR RETURN
SPACE 2			
*			
	REF	ACQCONST	REFS TO PROCEDURE:
	REF	ACQIT	ACQUIRE CONSTANT ROUTINE
	REF	ACQXNB	ACQUIRES NAME OR NUMERIC ITEM.
	REF	CINTIM	ACQUIRES NEXT NON-BLANK CHAR.
	REF	CLOSR	CONVERT INTEGER TO IMAGE.
	REF	CMDRETRN	CLOSE AND RELEASE
	REF	CONTOFF	RESUMES IN CMD MODULE.
	REF	CSN	DOES)CONTINUE & OFF.
	REF	DUMPLING	CODESTRINGER
	REF	ENTEREM	LINE OUTPUT ROUTINE.
	REF	ERXEQ	THE ENTRY PT INTO ERROR MODULE.
			ERROR VIA EXECUTE-OPERATION.

REF	FORCLOSE	FORCE CLOSE ON OPEN FUNCTION
REF	FOROPEN	FORCE OPEN THE FORCE-CLOSED FUNCTION
REF	INPDIR	DIRECT INPUT PROCESSOR.
REF	INPEVAL	EVALUATED INPUT DRIVER
REF	KCLEAR	CLEAR COMMAND PROCESSOR.
REF	OBSERVED	RESUME IN CODEXEQ MOD. AFTER OBSERV.
REF	RCSIDAM	RESUME COPYING AFTER SI DAMAGE.
REF	REPROMPT	FUNCTION DEF-ISSUE REPROMPT
REF	RERASE	RESUME ERASING AFTER 'NOT ERASED'.
REF	RESIDAM	RESUME ERASING AFTER SI DAMAGE.
REF	RFSIDAM	RESUME FUN DEFN AFTER SI DAMAGE.
REF	RFUNDEF	RESUME FUN DEFN.
REF	RNOTCPY	RESUME AFTER 'NOT COPIED'.
REF	RNOTFND	RESUME AFTER 'NOT FOUND'.
REF	SAVEREL	RESUME AFTER CS FOR AUTOSTART.
REF	USEBADFL	USE BAD FUNCTION LINE, PERHAPS.
REF	XEQNIL	SUCCESSFUL XEQ-OPER, USE EMPTY RES.
REF	@CATCH	CATCH COMMAND PROCESSOR.
REF	@CONTINU	CONTINUE COMMAND PROCESSOR.
REF	@COPY	COPY COMMAND PROCESSOR.
REF	@DIGITS	DIGITS COMMAND PROCESSOR.
REF	@DROP	DROP COMMAND PROCESSOR.
REF	@ERASE	ERASE COMMAND PROCESSOR.
REF	@FNS	FNS COMMAND PROCESSOR.
REF	@GROUP	GROUP COMMAND PROCESSOR.
REF	@GRP	GRP COMMAND PROCESSOR.
REF	@GRPS	GRPS COMMAND PROCESSOR.
REF	@LIB	LIB COMMAND PROCESSOR.
REF	@LOAD	LOAD COMMAND PROCESSOR.
REF	@OBSERVE	OBSERVE COMMAND PROCESSOR.
REF	@OFF	OFF COMMAND PROCESSOR.
REF	@OPR	OPR COMMAND PROCESSOR.
REF	@OPRN	OPRN COMMAND PROCESSOR.
REF	@ORIGIN	ORIGIN COMMAND PROCESSOR.
REF	@PCOPY	PCOPY COMMAND PROCESSOR.
REF	@SAVE	SAVE COMMAND PROCESSOR.
REF	@SI	SI COMMAND PROCESSOR.
REF	@SIV	SIV COMMAND PROCESSOR.
REF	@SYMBOLS	SYMBOLS COMMAND PROCESSOR.
REF	@TABS	TABS COMMAND PROCESSOR.
REF	@TERM	TERMINAL COMMAND PROCESSOR.
REF	@VARS	VARS COMMAND PROCESSOR.
REF	@WIDTH	WIDTH COMMAND PROCESSOR.
REF	@WSID	WSID COMMAND PROCESSOR.
*		REFS TO CONTEXT:
REF	BRNVAL	ERR-CTRL BRANCH VALUE OR NEG.
REF	CMNDTYPE	COMMAND TYPE.
REF	CONSTBUF	CONSTANT BUFFER.
REF	CONSTTYP	CONSTANT TYPE
REF	COPYSAVE	SCRATCH DBLWD.
REF	EREGS	REG. SAVE AREA FOR ERROR WORK.
REF	ERRORID	ERROR I.D. HOLDER.
REF	F:WS	F:WS DCB.
REF	HICOMMON	HIGHEST ADDR. IN COMMON REGION.
REF	IMAGE	IMAGE BUFFER
REF	LINKOLAY	LINK HOLDER FOR OVERLAY SWITCHING.
REF	LINKWS	SAVE LOC FOR LINK
REF	MODE	EXECUTION MODE.
REF	NAMEBUF	HOLDS ITEM ACQUIRED BY ACQIT.
REF	ON%OFF	ON-LINE/OFF-LINE FLAG
REF	STATEPTR	PTS AT TOP STATE-ENTRY IN XEQ STACK.
REF	SYMT	PTS AT 1ST WD OF SYM TBL.
REF	TOPOSTAK	PTS AT TOP OF EXECUTION STACK.
*		REFS TO CONSTANTS:
REF	BITPOS	TBL OF BITS--BITPOS-K HAS ONLY BIT K
REF	IDBADCMD	ERROR I.D. FOR -- BAD COMMAND.
REF	IDSYMFUL	ERROR I.D. FOR -- SYM TBL FULL.
REF	IDWSFULL	ERROR I.D. FOR -- WS FULL.
REF	MAXREAL	X'7FFFFFFF'
*		
*	STANDARD EQU'S	
*		REGISTERS
R1	EQU	1
R2	EQU	2
R3	EQU	3
R4	EQU	4
R5	EQU	5
R6	EQU	6

```

R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
*   OTHER EQU'S
*
SYMTOFF EQU      20      OFFSET FROM LODYN TO SYMT
*                        NOTE-- MUST BE AN EVEN NUMBER --
SPACE
STACKOFF EQU      32    OFFSET FROM END OF COMMON TO THE EXECUTION STACK.
SPACE
EFLAG   EQU      X'10000' MARKS 'EXECUTE-OPER' TYPE OF Q-STATE
*
*   CONSTANTS
*
PAGE
*
*   COMMAND RECOGNITION DRIVER
*
*   PROCESSES COMMAND OR GIVES APPROPRIATE ERROR MESSAGE
*   EXITS VIA CMDEXIT, ACCORDING TO MODE
*   INDIVIDUAL COMMAND PROCESSORS ARE LABELLED: @-----
*
CMDREC  LI, R4      0
        STW, R4     ERRORID      CLEAR ERROR IDENTIFIER.
        BAL, R4     ACQNXNB      ACQUIRE NEXT NON-BLANK
        BAL, R14    ACQIT        ACQUIRE CMD IDENTIFIER.
        LW, R8      NAMEBUF      GET 1ST 4 LETTERS OR WHATEVER.
        LI, R6      -NCMDS       = - NO.OF CMD IDENTIFIERS TABLED.
CMDQ     CW, R8      CMDTBL+NCMDS, R6 HUNT FOR A MATCH...
        BNE        CMDLOOP      NOT YET
        STW, R6     CMNDTYPE      OK-SAVE TYPE FOR POSSIBLE REF.
        B          CMDB+NCMDS, R6 VECTOR OUT
CMDLOOP  BIR, R6     CMDQ         LOOP
        B          ERBADCMD      NO MATCH -- INCORRECT COMMAND.
CMDTBL   RES        0           NOTE--TABLE OF CMD ID'S, ASSOCIATE W/ CMDB
TEXTCLEA TEXT        'CLEA'
TEXTOFF  TEXT        'OFF '
        TEXT        'SAVE'      SAVE
        TEXT        'WIDT'
        TEXT        'ORIG'
        TEXT        'DIGI'
        TEXT        'TABS'
        TEXT        'CONT'
        TEXT        'QLOA'      QUIET LOAD
        TEXT        'LOAD'
        TEXT        'WSID'
        TEXT        'QCOP'      QUIET COPY
        TEXT        'COPY'
        TEXT        'QPCO'      QUIET PCOPY
        TEXT        'PCOP'
        TEXT        'DROP'
        TEXT        'LIB '
        TEXT        'FNS '
        TEXT        'VARS'
        TEXT        'GRPS'
        TEXT        'GROU'
        TEXT        'GRP '
        TEXT        'ERAS'
        TEXT        'SYMB'
        TEXT        'SI '
        TEXT        'SIV '
        TEXT        'OPR '
        TEXT        'OPRN'
        TEXT        'TERM'
        TEXT        'CATC'      CATCH
        TEXT        'OBSE'      OBSERVE
NCMDS    EQU        %-CMDTBL    = NO.OF CMD ID'S.
CMDB     RES        0           NOTE--BRANCH TBL ASSOCIATED W/ CMDTBL.
@@CLEAR  B          KCLEAR
@@OFF    B          @OFF
@@SAVE   B          @SAVE

```

```

@@WIDTH B @WIDTH
@@ORIGIN B @ORIGIN
@@DIGITS B @DIGITS
@@TABS B @TABS
@@CONT B @CONTINU
MTW,1 CMNDTYPE USE @LOAD
@@LOAD B @LOAD
@@WSID B @WSID
MTW,1 CMNDTYPE USE @COPY
@@COPY B @COPY
MTW,1 CMNDTYPE USE @PCOPY
@@PCOPY B @PCOPY
@@DROP B @DROP
@@LIB B @LIB
@@FNS B @FNS
@@VARS B @VARS
@@GRPS B @GRPS
@@GROUP B @GROUP
@@GRP B @GRP
@@ERASE B @ERASE
@@SYMB B @SYMBOLS
@@SI B @SI
@@SIV B @SIV
@@OPR B @OPR
@@OPRN B @OPRN
@@TERM B @TERM
@@CATCH B @CATCH
@@OBSERV B @OBSERVE
PAGE
*
* XWLOCGLB-PERFORM LOCAL-GLOBAL EXCHANGE ON SYMBOL TABLE
* USED WHEN COMMANDS REQUIRE SURFACING- AND SUBSEQUENT
* RESUBMERGING-OF GLOBAL SYMBOLS SHADOWED BY LOCALS
*
* R14=LINK R6,R7,R8,R9,R10,R11 USED-NOT SAVED
*
XWLOCGLB LW,R11 HICOMMON
AI,R11 -STACKOFF
SW,R11 TOPOSTAK ESTABLISH LOOP COUNT
BLEZ *R14 EMPTY STACK-NO PROBLEM
LW,R9 MAXREAL MASK OFF SIGN BIT
LW,R6 TOPOSTAK
CHKSTAK LW,R8 1,R6 GET STACK ENTRY
BGEZ NOSHADOW NOT A SHADOWED GLOBAL
LW,R7 0,R6 EXCHANGE CANDIDATE-GET SYMT PTR
LW,R10 *SYMT,R7 GET SYMT REFERENT
OR,R10 BITPOS-0 MERGE 'G' BIT
STW,R10 1,R6 SET SYMT ENTRY IN STACK
STS,R8 *SYMT,R7 SET STACK ENTRY LESS 'G' BIT-IN SYMT
AI,R6 1 KICK STACK POINTER
AI,R11 -1 DECREMENT LOOP COUNT
NOSHADOW AI,R6 1 KICK STACK POINTER
BDR,R11 CHKSTAK LOOP
B *R14 RETURN
PAGE
*
* ACQCONLY-ACQUIRE CONSTANT(S) ONLY
* USED IN SCAN OF SYSTEM COMMANDS
*
* R14=LINK, ACQCONST USED
*
* MOST REGISTERS VOLATILE
*
ACQCONLY STW,R14 LINKWS SAVE LINK
STW,R14 CONSTBUF
BAL,R4 ACQCONST GET CONSTANT(S)
B ERBADCMD OVERFLOW
CI,R2 X'15'
BNE ERBADCMD RESIDUAL GARBAGE
LW,R14 CONSTTYP
BLZ ERBADCMD NON-INTEGER CONST. FOUND
LW,R7 CONSTBUF GET FIRST CONSTANT
BLZ ERBADCMD NEGATIVE-BAD SCENE
CI,R6 1 CHECK HOW MANY CONSTANTS FOUND
B *LINKWS RETURN WITH CC'S SET
PAGE
*
* OUTORANG -- DUAL RETURN PT FOR PARAMETER-CHANGE COMMANDS.

```

```

*
* VALOUT  -- ALT.ENTRY, PUTS VALUE IN R8 IN IMAGE, DISPLAYS, & EXITS.
* ISVAL   -- ALT.ENTRY TO DISPLAY 'IS ' & VALUE.
* GENIS   -- ALT.ENTRY TO DISPLAY 'IS ' USING VALUE IN R7.
*
OUTORANG B      ERBADCMD      @ RETURN PT IF -- BAD COMMAND.
          LW,R3   WASMSG      @ RETURN PT IF OK. GET 'WAS ' MSG.
MSGIM     STW,R3  IMAGE       PUT MSG IN IMAGE.
          LI,R3   4           SET POSITION IN IMAGE.
          LW,R8   R7          GET VALUE IN R8.
VALOUT    BAL,R13 CINTIM      PUT VALUE IN IMAGE.
          BAL,R12 DUMPLING    DISPLAY MSG NOW IN IMAGE.
          B       CMDEXIT     COMMAND EXIT.
ISVAL     LW,R7   COPYSAVE+1  GET VALUE IN R7.
GENIS     LW,R3   ISMSG      GET 'IS ' MSG.
          B       MSGIM
WASMSG    TEXT   'WAS '
ISMSG     TEXT   'IS '
          PAGE
*
* ERBADCMD-ILLEGAL COMMAND-AND CMDEXIT-COMMAND EXIT
*
*
* RETURNS TO DRIVER INDICATED BY MODE:FORCE OPEN
*                               DEFINITION REPROMPT
*                               DIRECT INPUT
*                               EVALUATED NPUT
*
ERBDCEMDA LI,R5   F:WS        CLOSE
          BAL,R6   CLOSR      OPEN DCB
ERBADCMD  LI,R8   IDBADCMD    ERROR I.D. FOR 'BAD COMMAND'.
CMDERR    BAL,R7  CMDERRH    HANDLE COMMAND ERROR.
CMDEXIT   LCW,R8  ERRORID     CK ERROR I.D...
          BEZ      CMDEXITM   ZERO -- NO ERROR.
          BLZ      CMDEXITC   POS -- ERR-CTRL TESTED ALREADY.
          BAL,R7  IDECTEST    @ NEG -- ERR DISPLAYED, CK ERR-CTRL:
          B        CTRL      @ CONTROL THIS ERROR.
          B        CMDEXITO   @ UNCONTROLLED.
CMDEXITC  LW,R8   BRNVAL      WAS ERROR CONTROLLED...
          BGEZ     CTRL      YES.
CMDEXITO  LW,R8   ON%OFF      DID ERROR OCCUR IN OFF-LINE OPERATN.
          BEZ      BCONTOFF   YES -- SIMULATE )CONTINUE CMD.
CMDEXITM  LW,R7   MODE        EXIT CMD ACCORDING TO THE MODE:
          B        CMDXVECT,R7
          B        FOROPEN    -1 FORCE OPEN THE FUN FORCED CLOSE.
CMDXVECT  B        REPROMPT   0 FUN DEFN MODE.
          B        INPDIR     1 DIRECT INPUT MODE.
          LI,R7   EFLAG       2 EVAL-INPUT OR EXECUTE-OPERATOR MODE:
          CW,R7   *STATEPTR
          BAZ      INPEVAL     EVALUATED INPUT.
          LW,R7   ERRORID     EXECUTE-OP, DID ERROR OCCUR...
          BEZ      XEQNIL      NO -- EMPTY VECTOR RESULTS.
          B        ERXEQ       YES.
DONTSAVE  EQU     DONTSAVH    HANDLE 'NOT SAVED, ETC.' ERROR.
ERRERASE  STW,R1  EREGS       SAVE R1.
          BAL,R14  DERASE      DISPLAY 'NAME NOT ERASED'.
          LW,R1    EREGS       RESTORE R1.
          B        RERASE      RESUME ERASING; DO ERR-CTRL LATER.
ERNOTFND  BAL,R14 DNOTFND     DISPLAY 'NAME NOT FOUND'.
          B        RNOTFND     RESUME COPYING; DO ERR-CTRL LATER.
ERNOTCPY  BAL,R14 DNOTCPY     DISPLAY 'NAME NOT COPIED'.
          B        RNOTCPY     RESUME COPYING; DO ERR-CTRL LATER.
BSIDAM    B        DSIDAM     *BREF*
SIDAME    BAL,R7  BSIDAM      DISPLAY 'SI DAMAGE'.
          B        RESIDAM     RESUME ERASING; DO ERR-CTRL LATER.
SIDAMC    BAL,R7  BSIDAM      DISPLAY 'SI DAMAGE'.
          B        RCSIDAM     RESUME COPYING; DO ERR-CTRL LATER.
SIDAMF    BAL,R7  BSIDAM      DISPLAY 'SI DAMAGE'.
          B        RFSIDAM     RESUME FUN DEF; DO ERR-CTRL LATER.
ERRDEFN   EQU     DEFNERRH    HANDLE 'DEFN ERR' ERROR.
LSCANERR  EQU     LSCANERRH   HANDLE 'LINESCAN ERR' ERROR.
BBADFL    B        USEBADFL   *BREF* USE BAD FUNCTION LINE, MAYBE.
ERRFDSYM  LI,R8   IDSYPFUL    = ERROR I.D. FOR 'SYM TBL FULL'.
          B        ERRFUNDF
ERRFDWS   LI,R8   IDWSFULL    = ERROR I.D. FOR 'WS FULL'.
ERRFUNDF  BAL,R7  FUNERRH     HANDLE FUN DEFN ERROR 6
BRFUNDEF  B        RFUNDEF    *BREF* RETURN TO FUNDEF MODULE.
CSERRH    LI,R5   F:WS        CLOSE & RELEASE IN CASE THIS

```

	LI, R6	CMDERR		CODESTRING ERR OCCURRED DURING
*	B	CLOSR		AN AUTOCONTINUE SAVE; HANDLE ERR
				AS IF A CMD OR FUN DEF MISTAKE.
BREPROMT	EQU	CMDXVECT		(TO SAVE A BRANCHING REF)
BCONTOFF	LI, R2	-1		FLAG FOR ABORT
	B	CONTOFF	*BREF*	AND GO TO 'CONTINUE'
B4CLOSE	STW, R14	LINKOLAY		SAVE LINK BACK TO CMD MODULE.
	BAL, R14	FORCLOSE	*BREF*	FORCE CLOSE OF OPEN FUNCTION.
CMDVIA14	STCF	LINKOLAY		SAVE COND.CODE.
	LW, R14	LINKOLAY		RESTORE LINK BACK TO CMD MODULE.
	LC	LINKOLAY		RECOVER COND.CODE.
	B	CMDRETRN	*BREF*	RETURN TO CMD MODULE.
OBSERVEZ	B	OBSERVED	*BREF*	RETURN TO CODEXEQ MODULE.
BCSN	BAL, R12	CSN		CODESTRING (FOR AUTOSTART), THEN
	B	SAVEREL		RESUME THE SAVE COMMAND WORK.
BAQCONLY	STW, R14	LINKOLAY		SAVE LINK BACK TO CMD MODULE.
	LI, R14	CMDVIA14		PREPARE TO RESUME AFTER
	B	ACQCONLY		ACQUIRING A CONSTANT.

PAGE

*

* COMMUNICATION REGION -- FROM ROOT INTO ERROR MODULE -- VIA R15.

*

EBALS	RES	0	ERROR-ENTRY BAL'S. (ORDERING MUST
ERRN	BAL, R15	BENTEREM	BE MATCHED IN ERROR MODULE).
ERRF	BAL, R15	BENTEREM	
ERRX	BAL, R15	BENTEREM	
CMDERRH	BAL, R15	BENTEREM	
IDECTEST	BAL, R15	BENTEREM	
CTRL	BAL, R15	BENTEREM	
DONTSAVH	BAL, R15	BENTEREM	
DERASE	BAL, R15	BENTEREM	
DNOTFND	BAL, R15	BENTEREM	
DNOTCPY	BAL, R15	BENTEREM	
DSIDAM	BAL, R15	BENTEREM	
DEFNERRH	BAL, R15	BENTEREM	
LSCANERH	BAL, R15	BENTEREM	
UNREF	BAL, R15	BENTEREM	
ERRSET	BAL, R15	BENTEREM	
OBSERVER	BAL, R15	BENTEREM	
FUNDERRH	BAL, R15	BENTEREM	

SPACE 2

BENTEREM	B	ENTEREM	ENTER ERROR MODULE WITH R15 SET.
----------	---	---------	----------------------------------

PAGE

SPACE 2

Z	SET	%-RCMD@	SIZE OF RCMD MODULE IN HEX.
---	-----	---------	-----------------------------

SPACE

Z	SET	Z+Z/10*6+Z/100*96+Z/1000*1536	SIZE IN DECIMAL.
---	-----	-------------------------------	------------------

SPACE 3

END

	TITLE	'SCMD-B00,08/22/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	
SCMD@	RES	0	CONTROL CARDS NOT PRINTED.
			ORIGIN OF SIMPLE-CMD MODULE.
*			
*	REF'S	AND	DEF'S
*			
	DEF	SCMD@	= START OF 'SCMD' MODULE.
	DEF	@WIDTH	WIDTH COMMAND PROCESSOR.
	DEF	@DIGITS	DIGITS COMMAND PROCESSOR.
	DEF	@ORIGIN	ORIGIN COMMAND PROCESSOR.
	DEF	@ERASE	ERASE COMMAND PROCESSOR.
	DEF	RESIDAM	RESUME ERASING AFTER SI DAMAGE.
	DEF	RERASE	RESUME ERASING.
	DEF	@GROUP	GROUP COMMAND PROCESSOR.
	DEF	@GRP	GRP COMMAND PROCESSOR.
	DEF	@FNS	FNS COMMAND PROCESSOR.
	DEF	@GRPS	GRPS COMMAND PROCESSOR.
	DEF	@VARS	VARS COMMAND PROCESSOR.
	DEF	@SIV	SIV COMMAND PROCESSOR.
	DEF	@SI	SI COMMAND PROCESSOR.
	DEF	@CATCH	CATCH COMMAND PROCESSOR.
	DEF	@OBSERVE	OBSERVE COMMAND PROCESSOR.
	DEF	FOROPEN	FORCED OPEN OF CLOSED FN.
	SPACE 2		
*			
	REF	ACQCONLY	REFS TO PROCEDURE: ACQUIRES CONSTANT ONLY.
	REF	ACQIT	ACQUIRES NAME OR NUMERIC ITEM.
	REF	ACQNAME	ACQUIRES A NAME.
	REF	ACQNB	ACQUIRES NON-BLANK CHAR.
	REF	ACQNXCC	ACQUIRES NEXT CHAR & CODE.
	REF	ALOCNONX	ALLOC DATA BLK: N WDS + HDR, EVEN SIZE. NON-EXECUTION MODE.
*			
	REF	CMDERR	COMMAND ERROR.
	REF	CMDEXIT	COMMAND EXIT.
	REF	DREF	DE-REFERENCER.
	REF	DUMPLING	LINE OUTPUT ROUTINE.
	REF	ERBADCMD	ERROR -- BAD COMMAND.
	REF	ERRERASE	ERROR -- ... NOT ERASED.
	REF	FERASECK	CK FOR ERASE OF FUNC BEING DEFINED.
	REF	FINDNAME	FINDS A NAME, BUT WON'T CREATE NEW.
	REF	FUNLDISP	PUTS FUNC. NAME & LINE NO. IN IMAGE
	REF	GENCHAR	GENERATE CHARACTER (OR MNEMONIC)
	REF	GENNAME	GEN. NAME (EXPAND MNEM. IF PRESENT)
	REF	GENNAME0	(DITTO, BUT WITHOUT INDENTATIONS).
	REF	INPDIR	DIRECT INPUT HANDLER.
	REF	INPRET	INPUT DRIVER ENTRY PT.
	REF	ISVAL	DISPLAYS 'IS ' & PARAMETER VALUE.
	REF	MAYDREF	DEREFERENCE IF R4 PTS AT A DATA BLK.
	REF	OUTORANG	EXIT (DUAL) FOR COMMON CMDS.
	REF	SETDIGIT	SET DIGITS,GET OLD VALUE
	REF	SETORG	SET NEW ORIGIN-GET OLD VALUE
	REF	SETWIDTH	SET WIDTH,GET OLD VALUE
	REF	SICLR	CLEARs STATE-INDICATOR TO GO-STATE.
	REF	SIDAME	SI DAMAGE DURING ERASE.
	REF	SQUEEZER	ENTRY IN APLINPUT TO SQUEEZE MNEM'S
	REF	XWLOCGLB	EXCHANGES LOCALS & GLOBALS.
*			
	REF	BREAKFLG	REFS TO CONTEXT: BREAK FLAG.
	REF	CATCHTBL	CATCH TABLE.
	REF	CONSTBUF	CONSTANT BUFFER.
	REF	COPYSAVE	SCRATCH DBLWD.
	REF	DIGITS	
	REF	GOSTATE	PT AT STATE-ENTRY TO CLEAR DOWN TO.
	REF	HICOL	HIGH COLUMN INDICATOR.
	REF	HICOMMON	HIGHEST ADDR. IN COMMON REGION.
	REF	IMAGE	IMAGE BUFFER.
	REF	MODE	EXECUTION MODE.
	REF	NAMEBUF	NAME BUFFER.
	REF	OBSERVE	OBSERVATION SETTING.
	REF	OBSFLAG	OBSERVE FLAG.
	REF	OPENFN	POINTER TO FNCT NAME OF FORCED CLOSE
	REF	ORIGIN	INDEX ORIGIN
	REF	RESULT	PTS AT A NEW DATA BLK.
	REF	SAVE312	SAVE AREA.
	REF	SICTRL	STATE-INDICATOR CONTROL SETTING.
	REF	STATEPTR	PTS AT TOP STATE-ENTRY IN XEQ STACK.

```

REF      SYMT      PTS AT SYM TBL.
REF      SYMTSIZE  NO.OF ENTRIES IN SYM TBL.
REF      WIDTH

*
REF      BITPOS    REFS TO CONSTANTS:
REF      BLANKS    TBL OF BITS--BITPOS-K = K-TH BIT.
REF      FUNTYPES  BLANKS.
REF      IDNOTGRP  DBLWD -- RANGE OF USER-DEFD FUN TYPES
REF      IDSYMFUL  ERROR I.D. FOR -- NOT GROUPED.
REF      IDWSFULL  ERROR I.D. FOR -- SYM TBL FULL.
REF      MAXREAL   ERROR I.D. FOR -- WS FULL.
REF      STACKOFF  X'7FFFFFFF'
REF      TEXTCLEA  OFFSET FROM HI COMMON TO XEQ STACK.
REF      TEXTOFF   'CLEA'
REF      X1FFFF    'OFF '
REF      X1FFFF    X'1FFFF'

*
* STANDARD EQU'S
*
* REGISTERS
R1      EQU      1
R2      EQU      2
R3      EQU      3
R4      EQU      4
R5      EQU      5
R6      EQU      6
R7      EQU      7
R8      EQU      8
R9      EQU      9
R10     EQU      10
R11     EQU      11
R12     EQU      12
R13     EQU      13
R14     EQU      14
R15     EQU      15
*
* OTHER EQU'S
*
*
CATF     EQU      10      FUNCTION-STATE CATEGORY, STACK-ENTRY
QUAD     EQU      X'53'   INTERNAL (EBCDIC) CHAR -- QUAD.
NAMECODE EQU      23      CODESTRING DESIG -- ORDINARY NAME.
LASTCSV  EQU      138     LAST CODESTRING DESIGNATION = DUMMY.
PENDFLAG EQU      X'8000' MARKS PENDENT STATE-ENTRIES IN STACK
TYPELOGL EQU      1       DATA BLK TYPE = LOGICAL.
TYPELIST EQU      6       DATA BLK TYPE = LIST.
TYPEGRP  EQU      X'11'   DATA BLK TYPE = GROUP.
*
* CONSTANTS
*
*
PAGE
*
* @WIDTH-CHANGE WIDTH AND(OR) REPORT OLD VALUE
*
*
@WIDTH   LI,R6      SETWIDTH      = WIDTH PARAMETER ROUTINE LOC.
          LW,R7      WIDTH          CURRENT WIDTH SETTING.
PARAMCMD STD,R6      COPYSAVE      SAVE PARAM. ROUTINE LOC & VALUE.
          BAL,R14    ACQCONLY      ACQ 1 CONSTANT ONLY, IF ANY...
          BL         ISVAL         NONE, DISPLAY 'IS ' & VALUE.
          BG         ERBADCMD      TOO MANY -- BAD COMMAND.
          LI,R5      OUTORANG      OK, SET DUAL RETURN LOC AND
          B          *COPYSAVE     GO TO PARAMETER SET ROUTINE.
          PAGE
*
* @DIGITS-CHANGE DIGITS AND(OR) REPORT OLD VALUE
*
*
@DIGITS  LI,R6      SETDIGIT      = DIGITS PARAMETER ROUTINE LOC.
          LW,R7      DIGITS        CURRENT DIGITS SETTING.
          B          PARAMCMD
          PAGE
*
* @ORIGIN-CHANGE ORIGIN AND(OR) REPORT OLD VALUE
*
*
@ORIGIN  LI,R6      SETORG        = ORIGIN PARAMETER ROUTINE LOC.
          LW,R7      ORIGIN        CURRENT ORIGIN SETTING.
          B          PARAMCMD
          PAGE
*
* @ERASE-DRIVER TO ERASE NAMED GLOBAL OBJECTS FROM USERS WORKSPACE
*
*
@ERASE   LI,R14     0             PRESET NAME COUNTER.

```


	STW, R14	CONSTBUF	DOE A NAME FOLLOW ')ERASE'...
	CI, R3	LASTCSV	NO -- INCORR. CMND.
ERASEA	BLE	ERBADCMD	FIND THE NAME...
	BAL, R8	FINDNAME	NOT FOUND.
	B	ERASEB	FOUND, BUMP NAME COUNT.
	MTW, 1	CONSTBUF	
	LW, R7	CONSTBUF	
ERASEB	STW, R6	CONSTBUF, R7	PUT ITS NAME PTR IN BUFFER.
	CI, R3	LASTCSV	DOES ANOTHER NAME FOLLOW...
	BG	ERASEA	YES, LOOP BACK.
	AI, R2	-X'15'	NO, VERIFY END OF STMT...
	BNEZ	ERBADCMD	OOPS -- INCORR. CMND.
	LW, R2	MODE	IS THIS FUNC. DEFN MODE...
	BNEZ	ERASEC	NO.
ERASEC	BAL, R14	FERASECK	YES, MAY ERASE THE OPEN FUNCTION.
	BAL, R14	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.
	LI, R1	0	PRE-SET INDICATING
	STW, R1	SAVE312	NOT ON A GROUP.
	LW, R1	CONSTBUF	= # NAMES TO ERASE.
ERASED	LW, R6	CONSTBUF, R1	GET A NAME PTR.
	LW, R4	BITPOS-0	EXCHANGE ITS REFERENT-INDICATOR WITH
	XW, R4	*SYMT, R6	A CLEAR (GLOBAL-BIT ONLY) ONE,
	STW, R4	CONSTBUF, R1	AND SAVE THE REF-INDIC.
	AND, R4	X1FFFF	IS THERE A REFERENT...
	BEZ	ERASEJ	NO -- SKIP IT.
	LB, R14	*R4	YES.
	CI, R14	TYPEGRP	IS IT A GROUP...
	BE	ERASEGRP	YEP.
	CLM, R14	FUNTPES	IS IT A FUNCTION DESCRIPTOR...
	BCS, 9	ERASEI	HOORAY, IT IS NOT.
	LW, R14	1, R4	DAMN, GET ITS REF-COUNT.
	AI, R14	-1	DECR IT; IF ONLY 1 REF, WE CAN
	BEZ	ERASEI	ERASE THE FUNCTION EASILY.
	LI, R12	PENDFLAG	OTHERWISE, CK FOR IT BEING PENDENT.
	LI, R5	X'1FFFF'	
	LW, R2	STATEPTR	
	B	ERASEF	
ERASEE	LI, R3	X'7FFF'	
	AND, R3	0, R2	
	BEZ	PENDENT	NUTS, PENDENT (DYADIC WHOSE LEFT
	AW, R2	R3	ARG IS UNRESOLVED).
ERASEF	CS, R4	1, R2	
	BNE	ERASEE	
	CW, R12	0, R2	
	BANZ	PENDENT	NUTS, PENDENT (CALLED FOR SOMETHIG
	BDR, R14	ERASEE	& NOT RETURNED TO).
	MTW, -1	1, R4	OK, SUSPENDED ONLY, DECR REF-CNT
	LW, R2	STATEPTR	MOMENTARILY.
	B	ERASEH	DAMAGE EACH SUSPENSION.
ERASEG	LI, R3	X'7FFF'	
	AND, R3	0, R2	
	AW, R2	R3	
ERASEH	CS, R4	1, R2	
	BNE	ERASEG	
	STW, R14	1, R2	(DAMAGED)
	MTW, -1	1, R4	(DECR REF-COUNT)
	BGZ	ERASEG	
	MTW, 1	1, R4	SET REF-COUNT BACK TO 1, FOR SYM TBL
	BAL, R7	DREF	AND DE-REF (DELETE) THE FUNCTION.
	B	SIDAME	ERROR -- SIDAMAGE,
RESIDAM	EQU	ERASEJ	RETURN AFTER DISPLAY OF ERR MSG.
PENDENT	LW, R4	CONSTBUF, R1	RESTORE THE ORIGINAL REFERENT-INDIC
	STW, R4	*SYMT, R6	BACK INTO THE FUNC.NAME SYM TBL
*			ENTRY.
	B	ERRERASE	ERROR -- NAME NOT ERASED,
RERASE	EQU	ERASEJ	RETURN AFTER DISPLAY OF ERR MSG.
ERASEI	BAL, R7	DREF	DE-REFERENCE THE ERASED REFERENT.
ERASEJ	BDR, R1	ERASED	LOOP TILL ERASE LIST EXHAUSTED.
	LW, R7	SAVE312	ARE WE WORKING ON A GROUP...
	BEZ	RXEXIT	NO.
	AI, R7	-1	YES, DECR ITS NAME COUNT.
ERASEGQ	STW, R7	SAVE312	SAVE THAT COUNT & IF NOT ZERO
	BGZ	ERASEG1	ERASE ANOTHER NAME IN GROUP.
	LW, R1	SAVE312+2	DONE, RESTORE ORIG. ERASE LIST COUNT
	LW, R14	SAVE312+3	AND ITS 1ST NAME PTR.
	STW, R14	CONSTBUF+1	
	LW, R6	SAVE312+1	GET GROUP'S NAME PTR, AND EXCHANGE
	LW, R4	BITPOS-0	(AGAIN) ITS SYM TBL REF-INDIC WITH A

	XW, R4	*SYMT, R6	CLEAR (GLOBAL-BIT ONLY) ONE.
	B	ERASEI	THEN GET RID OF THE GROUP ITSELF.
ERASEGRP	LW, R14	SAVE312	IS THIS A GROUP IN A GROUP...
	BNEZ	ERASEI	YES -- DISPERSE IT.
	STW, R1	SAVE312+2	NO, SAVE CURRENT ERASE LIST COUNT.
	STW, R4	*SYMT, R6	PUT GROUP'S DB PTR BACK INTO SYM TBL
	STW, R6	SAVE312+1	SAVE NAME PTR TO THE GROUP.
	LW, R14	CONSTBUF+1	SAVE ERASE LISTS 1ST NAME PTR.
	STW, R14	SAVE312+3	
	AI, R4	2	PT AT COUNT FIELD OF THE GROUP
	LH, R7	*R4	GET ITS COUNT.
	B	ERASEGQ	
ERASEG1	LW, R2	SAVE312+1	NAME PTR FOR GROUP ITSELF.
	LW, R3	*SYMT, R2	GET GROUP'S DATA BLK PTR AGAIN.
	AI, R3	2	OFFSET TO ITS COUNT FIELD.
	LH, R6	*R3, R7	GET ONE OF ITS NAME PTRS. AND PUT
	STW, R6	CONSTBUF+1	IT IN 1ST POS.OF 'ERASE LIST'.
	LI, R1	1	CLAIM ONLY 1 NAME PTR.
	LW, R14	MODE	IS THIS FUN DEFN MODE...
	BNEZ	ERASED	NO--ERASE THAT ONE.
	STW, R1	CONSTBUF	YES, ERASE AFTER CHECKING FOR
	LI, R14	ERASED	MATCH OF THE OPEN FUNC. NAME PTR
	B	FERASECK	
RXEXIT	LI, R14	CMDEXIT	TAKE COMMAND EXIT AFTER
	B	XWLOCGLB	RE-EXCHANGING LOCALS & GLOBALS.
PAGE			
*			
* @GROUP-DRIVER TO GATHER LISTED NAMES INTO A GROUP OR DISPERSE A GROUP			
*			
@GROUP	LI, R14	-1	PRESET NAME COUNTER.
	STW, R14	CONSTBUF	
	CI, R3	LASTCSV	DOES A NAME-START FOLLOW ')GROUP'...
	BLE	ERBADCMD	NO -- INCORR. CMND.
GRPA	BAL, R12	ACQNAME	ACQ NAME IF POSSIBLE...
	B	GRPSYMFL	OOPS -- SYM TBL FULL.
	B	GRPWSFL	OOPS -- WS FULL.
	AI, R13	-NAMECODE	OK, VERIFY ORDINARY NAME...
	BNEZ	ERBADCMD	STOP OR TRACE NAME -- INCORR.CMD
	MTW, 1	CONSTBUF	BUMP NAME COUNTER.
	LW, R7	CONSTBUF	
	STW, R6	CONSTBUF+1, R7	SAVE ITS NAME PTR.
	CI, R3	LASTCSV	DOES ANOTHER NAME FOLLOW...
	BG	GRPA	YES.
	AI, R2	-X'15'	NO, VERIFY END OF LINE...
	BNEZ	ERBADCMD	NOPE -- INCORR. CMND.
	BAL, R14	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.
	LW, R6	CONSTBUF+1	SET R6 = GROUP'S NAME PTR.
	LI, R5	X'1FFFF'	SET FOR SELECTIVE LOAD.
	LI, R4	0	
	LW, R7	CONSTBUF	= NO.OF NAMES MINUS 1.
	BGZ	GRPB	(CREATE, REPLACE, OR EXPAND A GROUP)
	LS, R4	*SYMT, R6	(DISPERSE) GET REFERENT-INDICATOR...
	BEZ	RXEXIT	NONE -- ASSUME DISPERSED ALREADY.
	LB, R2	*R4	OK, VERIFY IT'S A GROUP...
	AI, R2	-TYPEGRP	
	BEZ	DISPERSE	BINGO -- DE-REFERENCE IT.
NOTGROUP	LI, R14	ERNOTGRP	ISSUE DIAGNOSTIC AFTER
	B	XWLOCGLB	RE-EXCHANGING LOCALS & GLOBALS.
GRPWSFUL	BAL, R14	XWLOCGLB	RE-EXCHANGE LOCALS & GLOBALS.
GRPWSFL	LI, R8	IDWSFULL	= ERROR I.D. FOR 'WS FULL'.
	B	CMDERR	CMD ERROR EXIT.
GRPSYMFL	LI, R8	IDSYMFUL	= ERROR I.D. FOR 'SYM TBL FULL'.
	B	CMDERR	CMD ERROR EXIT.
GRPB	LS, R4	*SYMT, R6	GET REFERENT-INDICATOR...
	BEZ	GRPC	NONE -- CREATE NEW GROUP.
	LB, R14	*R4	SOME -- REPLACE OR EXPAND GROUP.
	AI, R14	-TYPEGRP	VERIFY IT'S A GROUP...
	BNEZ	NOTGROUP	OOPS.
GRPC	LW, R11	CONSTBUF	= # NAMES IN THIS)GROUP LIST.
	LI, R10	0	CLEAR GROUP-NAME REPETITION COUNT.
*			
* (SO WE CAN HANDLE EVEN WEIRDOS			
LIKE ')GROUP G A G B G C G').			
GRPD	CW, R6	CONSTBUF+1, R7	CK FOR REPEATED GROUP NAME PTR...
	BNE	GRPE	NO.
	AI, R10	-1	YES, DECR REPETITION COUNT AND
	STW, R10	CONSTBUF+1, R7	REPLACE NAME PTR BY IT.
GRPE	BDR, R7	GRPD	WORK BACK THROUGH THE LIST.
	AW, R11	R10	= # NAMES GROUPED BY THE LIST.

	AI, R10	0	ANY GROUP-NAME REPETITION...
	BEZ	GRPF	NO.
	AI, R4	0	YES, DOES ITS GROUP ALREADY EXIST.
	BEZ	GRPF	NO (STUPID GROUP CMND).
	AI, R4	2	YES, ADD IN THE # NAMES IN THE
	AH, R11	*R4	OLD GROUP.
GRPF	STH, R11	CONSTBUF	SET LENGTH OF NEW GROUP.
	AI, R11	2	ACCT FOR LENGTH HALFWD & ROUND UP.
	SLS, R11	-1	= SIZE NEEDED FOR NEW GROUP DATA BLK
	BAL, R14	ALOCNONX	ALLOC IT & DB HDR...
	B	GRPWSFUL	OOPS -- WS FULL.
	STW, R4	RESULT	FINE, SAVE PTR TO NEW DATA BLK.
	LI, R14	TYPEGRP	MAKE IT A GROUP TYPE DATA BLK.
	STB, R14	*RESULT	
	LI, R5	4	PLUG IN THE NEW GROUP'S LENGTH.
	LH, R10	CONSTBUF	
	STH, R10	*RESULT, R5	
	INT, R7	CONSTBUF	GET LENGTH OF)GROUP CMND'S LIST.
GRPG	LW, R10	CONSTBUF+1, R7	(FROM RT TO LF) GET LIST'S NAME PTR
	BLZ	GRPI	OR REPETITION INDICATOR.
	AI, R5	1	NOTE--PLUG IN NAME PTRS IN REVERSE ORDER
	STH, R10	*RESULT, R5	WITH RESPECT TO CMND'S LIST.
GRPH	BDR, R7	GRPG	
	LI, R2	0	CLEAR & GET PTR TO NEW GROUP.
	XW, R2	RESULT	
	LW, R4	*SYMT, R6	GET OLD REF-INDICATOR FOR THAT NAME.
DISPERSE	LI, R3	X'1FFFF'	REPLACE OLD REF-INDIC BY PTR TO NEW
	STS, R2	*SYMT, R6	GROUP (BY ZERO IF A DISPERSE).
	LI, R7	RXEXIT	GO TO 'RXEXIT' AFTER DE-REFERENCING
	B	MAYDREF	ANY OLD GROUP INDICATED.
GRPI	AI, R10	1	IS IT 1ST REPETITION...
	BNEZ	GRPH	NO, SKIP IT.
	LI, R2	0	
	LI, R3	X'1FFFF'	YES, IS THERE AN OLD GROUP OF THE
	LS, R2	*SYMT, R6	SAME GROUP-NAME REPEATED...
	BEZ	GRPH	NOPE, SKIP IT.
	LI, R3	4	YEP.
	LH, R14	*R2, R3	GET # NAME PTRS IN OLD GROUP.
	BEZ	GRPH	NONE, SKIP IT.
GRPJ	AI, R3	1	
	LH, R10	*R2, R3	MOVE NAME PTRS FOR OLD GROUP INTO
	AI, R5	1	THE NEW GROUP (RETAINING SAME
	STH, R10	*RESULT, R5	ORDER AS FOR OLD GROUP -- WHICH
	BDR, R14	GRPJ	WAS BACKWARDS FOR OLD GROUP'S
	B	GRPH	CMND LIST).
ERNOTGRP	LI, R8	IDNOTGRP	ERROR I.D. FOR 'NOT GROUPED'.
	B	CMDERR	CMD ERROR EXIT.
PAGE			

*			*
*	GAP -- PUTS BLANKS INTO IMAGE BUFFER FOR SPACING & COLUMNARIZATION		*
*	UNLESS -- WE ARE AT LEFT MARGIN OR REACH CURRENT WIDTH.		*
*	REGS:	R13 -- LINK, EXIT VIA *R13.	*
*		R3 -- COLUMN INDICATOR.	*
*		R7 IS VOLATILE	*
*			*
GAP	AI, R3	0	ARE WE AT LEFT MARGIN...
	BEZ	*R13	YES -- EXIT.
	LI, R7	' '	NO, GET A BLANKETY-BLANK BLANK.
GAPW	CW, R3	WIDTH	HAVE WE HIT WIDTH-SETTING...
	BGE	*R13	YES -- EXIT.
	STB, R7	IMAGE, R3	NO, PLUG IN A BLANK.
	AI, R3	1	INCR COLUMN INDICATOR.
	CI, R3	3	IS IT A MULTIPLE OF 4...
	BANZ	GAPW	NOT YET.
	B	*R13	YES -- EXIT.
PAGE			
* @GRP-DRIVER TO LIST NAMES OF OBJECTS IN A GROUP			
* @GRP			
	CI, R3	LASTCSV	DOES A NAME-START FOLLOW ')GRP'...
	BLE	ERBADCMD	NO -- INCORR. CMND.
	BAL, R8	FINDNAME	YES, FIND THE NAME...
	B	CMDEXIT	NOT FOUND -- EXIT.
	AI, R2	-X'15'	FOUND, VERIFY THAT ENDS THE CMND
	BNEZ	ERBADCMD	OOPS -- INCORR. CMND.
	STW, R6	CONSTBUF	OK, SAVE THE NAME PTR.
	BAL, R14	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.

```

        LW,R6      CONSTBUF      GET THE REFERENT-INDICATOR FOR THE
        LI,R1      X'1FFFF'      NAMED ITEM...
        AND,R1     *SYMT,R6
        BEZ        RXEXIT        NONE -- SKIP IT.
        LB,R14     *R1           OK, VERIFY REFERENT IS A GROUP.
        AI,R14     -TYPEGRP
        BNEZ       NOTGROUP      YUCK -- NOT GROUPED.
        AI,R1      2             PT AT 1ST WD OF THE GROUPING, AND
        STW,R1     CONSTBUF      SAVE THAT LOC.
        LH,R1      *CONSTBUF     GET LENGTH OF THE GROUP...
        BEZ        RXEXIT        ZERO -- SKIP IT.
        LI,R3      0             PRESET COLUMN INDICATOR.
GRPOUT  BAL,R13    GAP           GIVE SPACING & COLUMNARIZATION.
        LH,R8      *CONSTBUF,R1  FROM LAST NAME PTR TO FIRST, DELIVER
        STW,R1     CONSTBUF+1    TO 'GENNAME0' WHICH PUTS THE NAMES
        BAL,R13    GENNAME0      IN IMAGE BUFFER (NO INDENTATIONS)
        LW,R1      CONSTBUF+1    IN PROPER FORM, OUTPUTTING AS
        BDR,R1     GRPOUT        WHOLE LINES ARE FILLED UP.
        AI,R3      0             WAS LAST LINE OUTPUT ALREADY...
        BEZ        RXEXIT        YES.
        LI,R12     RXEXIT        NO, GO TO 'RXEXIT' AFTER
        B          DUMPLING      OUTPUTTING LAST LINE.

PAGE
*****
*
* QNAME -- COMPARES A TEST NAME-STRING AGAINST THE STRING IN CONSTBUF.
* REGS:   R7 -- LINK, EXITS: 0,R7 = HIGH TEST STRING
*                               1,R7 = EQUAL TEST STRING
*                               2,R7 = LOW TEST STRING
*
* R6 -- (EXIT) = NO.OF WDS TO CONTAIN THE TEST STRING.
* R1 -- (ENTRY) PTS AT A SYMBOL TABLE NAME-INDICATOR WD
*                               AND HAS A 1 IN BYTE 0. FOR A SHORT NAME*
*                               THE NAME-INDIC. WD CONTAINS IT.
*                               FOR A LONG NAME THE NAME-INDIC. WD
*                               CONTAINS A PTR TO THE 1ST WD OF THE
*                               NAME & BYTE 0 CONTAINS THE NO.OF
*                               WDS USED TO CONTAIN THE NAME.
*
* R9 -- (ENTRY) CONTAINS ALL BLANKS.
* R2 & R3 ARE VOLATILE.
*
*
QNAME   LW,R6      BREAKFLG      TEST FOR A BREAK OR HANG-UP...
        BNEZ       QBREAK        YES -- BREAK EXIT.
        LB,R6      *R1           GET BYTE 0 OF NAME-INDICATOR WD.
        CI,R6      20            IS IT LESS THAN A NAME-START CHAR...
        BLE        QNAMEL        YES, IT'S LONG NAME WD COUNT.
        LI,R6      1             NO, SHORT NAME, USE WD COUNT OF 1.
        LW,R3      R1           SET 'DESTINATION' REG (WD RESOLUT'N)
        B          QNAMES
QNAMEL  LW,R3      0,R1          SET 'DESTINATION' REG (WD RESOLUT'N)
QNAMES  SLS,R3     2            USE BYTE RESOLUTION.
        LI,R2      BA(CONSTBUF)  SET 'SOURCE' REG.
        CBS,R2     0            TEST SOURCE VS DESTINATION STRINGS.
        BL        0,R7          LO -- EXIT, TEST-STRING IS HIGH.
        BG        2,R7          HI -- EXIT, TEST-STRING IS LOW.
        CW,R9      CONSTBUF,R6   EQUAL, IS 'SOURCE' A LONGER NAME.
        BNE       2,R7          YES -- EXIT, TEST-STRING IS LOW.
        B         1,R7          NO -- EXIT, NAMES ARE IDENTICAL.

PAGE
*
* @FNS-DRIVER TO LIST NAMES OF FUNCTIONS IN USERS WORKSPACE
*
* @GRPS-DRIVER TO LIST NAMES OF GROUPS IN USERS WORKSPACE
*
* @VARS-DRIVER TO LIST NAMES OF GLOBAL VARIABLES IN USERS WORKSPACE
*
@FNS    LD,R10     FUNTYPES      = RANGE OF USER-DEFD FUNCTION TYPES.
        AI,R11     3            EXPAND TO COVER INTRINSIC FUN TYPES.
        B          QSET
@GRPS   LI,R10     TYPEGRP       = RANGE OF GROUP TYPE DATA BLK TYPES
QSETGRP LI,R11     TYPEGRP
        B          QSET
@VARS   LI,R10     TYPELOGL      = RANGE OF VARIABLE TYPES.
        LI,R11     TYPELIST
QSET    STD,R10    CONSTBUF+44   SAVE RANGE OF DB TYPES OF INTEREST.
        LI,R5      -11
        LW,R9      BLANKS       BLANK 22 WDS EACH OF THE
QSETA   STD,R9     CONSTBUF+22,R5 LO COMPARE STRING AND
        STD,R9     CONSTBUF+44,R5 HI COMPARE STRING.

```

	BIR, R5	QSETA	
	STW, R5	CONSTBUF+47	ZERO (LEFT MARGIN) COL.INDIC. HOLDER
	LI, R5	'9'	SET THE HI COMPARE STRING BIGGER
	STB, R5	CONSTBUF+22	THAN ANY BREADBOX.
	CI, R2	X'15'	DOES CMND LINE END ALREADY...
	BE	QX	YES.
QSETB	LI, R5	-80	NO, MOVE UP TO 80 NEW CHARS INTO
	STB, R2	CONSTBUF+20, R5	THE LO COMPARE STRING.
	BAL, R4	ACQNXCC	
	CI, R2	' '	QUIT ON BLANK OR LESS.
	BLE	QSETC	
	BIR, R5	QSETB	
	B	ACQNXCC	SKIP IF OVER 80 CHARS.
QSETC	BAL, R4	ACQNB	GET 1ST NON-BLANK AFTER LO STRING.
	CI, R2	X'15'	DOES CMND LINE END NOW...
	BE	QX	YES.
	LI, R5	-80	NO, MOVE UP TO 80 NEW CHARS INTO
QSETD	STB, R2	CONSTBUF+42, R5	THE HI COMPARE STRING.
	BAL, R4	ACQNXCC	
	CI, R2	' '	QUIT ON BLANK OR LESS.
	BLE	QSETE	
	BIR, R5	QSETD	
	B	ACQNXCC	SKIP IF OVER 80 CHARS.
QSETE	BAL, R4	ACQNB	GET 1ST NON-BLANK AFTER HI STRING.
	AI, R2	-X'15'	VERIFY THAT ENDS THE CMND.
	BNEZ	ERBADCMD	OOPS -- INCORR. CMND.
QX	BAL, R14	XWLOCGLB	EXCHANGE LOCALS & GLOBALS.
	BAL, R7	QINIT	INIT. FOR SYMBOL COMPARISONS.
QSYMA	AI, R1	2	PT AT A NAME-INDICATOR WD.
	LI, R5	X'1FFFF'	DOES THAT SYMBOL TABLE ENTRY HAVE A
	AND, R5	-1, R1	REFERENT...
	BEZ	QSYML	NO, LEAVE ITS HIT-BIT = 0.
	LB, R10	*R5	YES, GET REFERENT'S D.B. TYPE.
	CLM, R10	CONSTBUF+44	IS IT A TYPE WE ARE INTERESTED IN...
	BCS, 9	QSYML	NO, LEAVE HIT-BIT = 0.
	BAL, R7	QNAME	@ YES, COMPARE NAME TO LO STRING...
	B	%+1	@ ABOVE (SET HIT-BIT = 1).
	AWM, R13	-1, R1	@ EQUAL (SET HIT-BIT = 1).
*			@ BELOW (LEAVE HIT-BIT = 0).
QSYML	BIR, R8	QSYMA	SWEEP THE WHOLE SYMBOL TABLE.
	LI, R5	-10	
QSYMH	LD, R8	CONSTBUF+42, R5	REPLACE LO COMPARE STRING BY HI
	STD, R8	CONSTBUF+20, R5	COMPARE STRING.
	BIR, R5	QSYMH	
	BAL, R7	QINIT	INIT. FOR SYMBOL COMPARISONS.
QSYMB	AI, R1	2	PT AT A NAME-INDICATOR WD.
	CW, R13	-1, R1	TEST HIT-BIT...
	BAZ	QSYMZ	0 -- WE AREN'T INTERESTED.
	BAL, R7	QNAME	@ 1 -- COMPARE NAME TO HI STRING...
	B	QHITOFF	@ ABOVE (SET HIT-BIT BACK = 0).
	B	%+1	@ EQUAL.
*			@ BELOW.
QSYMZ	BIR, R8	QSYMB	SWEEP THE WHOLE SYMBOL TABLE.
*			
* NOTE	--AT THIS	PT ONLY SYM TBL ENTRIES WHOSE HIT-BIT = 1 ARE OF THE	
*		RIGHT TYPE AND USE NAMES IN THE CORRECT RANGE. EACH OF THOSE	
*		NAMES WILL BE DISPLAYED IN (EBCDIC) ORDER. WHEN DISPLAYED	
*		THE CORRESPONDING HIT-BIT WILL BE SET = 0 TO AVOID USING	
*		THAT NAME AGAIN. WE QUIT WHEN ALL HIT-BITS ARE SET = 0.	
QHUNT	LI, R7	QHUNTA	EXIT FROM QINIT TO QHUNTA.
QINIT	LW, R9	BLANKS	INSURE R9 = ALL BLANKS.
	LI, R13	X'40000'	INSURE R13 = HIT-BIT.
	LCW, R8	SYMTSIZE	= - NO.OF SYM TBL ENTRIES.
	LW, R1	SYMT	PT AT SYM TBL.
	STW, R1	CONSTBUF+46	SET 'CANDIDATE' FLAG POSITIVE.
	AI, R1	-1	PT 2 WDS AHEAD OF 1ST NAME-INDICATOR
	AW, R1	BITPOS-7	SET R1'S BYTE 0 TO A ONE (= NO.OF
*			WORDS NEEDED FOR SHORT NAMES).
	B	0, R7	EXIT.
QHITOFF	EOR, R13	-1, R1	TURN OFF HIT-BIT.
	STW, R13	-1, R1	RESTORE R13 = HIT-BIT.
	LI, R13	X'40000'	
	B	QSYMZ	
QHUNTA	AI, R1	2	PT AT A NAME-INDICATOR WD.
	CW, R13	-1, R1	TEST HIT-BIT...
	BAZ	QHUNTZ	0 -- WE AREN'T INTERESTED.
	BAL, R7	QNAME	@ 1, COMPARE NAME & LATEST CANDIDATE

	B	QHUNTZ	@	ABOVE (NOT INTERESTED YET).
	B	%+1	@	EQUAL (ONLY FAST HIT OF HI ONE).
*	STW, R8	CONSTBUF+46	@	BELOW (SET 'CANDIDATE' FLAG TO MINUS SYMTSIZE + COUNT).
	LI, R3	QMOVE		GO TO QMOVE AFTER SETTING UP BLANKS.
QRESETHI	LI, R5	-10		
	LW, R9	BLANKS		
QBLANK	STD, R9	CONSTBUF+20, R5		BLANK THE CANDIDATE STRING.
	BIR, R5	QBLANK		
	LI, R5	'9'		SET 1ST BYTE ABOVE ALL NAMES.
	STB, R5	CONSTBUF		
	B	0, R3		EXIT.
QMOVE	CI, R6	1		IS NEW CANDIDATE LONG OR SHORT...
	BG	QLONG		LONG.
	LW, R10	0, R1		SHORT, GET THAT NAME AND
	B	QCANDY		PUT IT IN CANDIDATE STRING.
QLONG	LW, R5	0, R1		PT AT 1ST WD OF LONG NAME.
	AI, R5	-1		BACK UP FOR INDEXING THRU THE NAME.
QLONGET	LW, R10	*R5, R6		GET NAME WDS (FROM LAST TO FIRST).
QCANDY	STW, R10	CONSTBUF-1, R6		PUT NAME WD IN CANDIDATE STRING.
	BDR, R6	QLONGET		LOOP TILL 1ST NAME WD REPLACES THE 'HI' WD OF CANDIDATE STRING.
*				
QHUNTZ	BIR, R8	QHUNTA		SWEEP THE WHOLE SYMBOL TABLE.
	LW, R8	CONSTBUF+46		GET LATEST CANDIDATE FLAG...
	BLZ	QSCORE		MINUS -- GOTCHA.
	LW, R3	CONSTBUF+47		NOT, GET COLUMN INDICATOR...
	BEZ	RXEXIT		0 -- RE-EXCHANGE & EXIT.
	LI, R12	RXEXIT		NZ, GO TO RXEXIT AFTER.
	B	DUMPLING		DISPLAYING LAST LINE.
QBREAK	LW, R8	SYMTSIZE		ON BREAK OR HANG-UP,
	LW, R1	SYMT		PUT ALL HIT-BITS = 0.
	EOR, R13	MAXREAL+1		(MAXREAL+1 = -1) GET ALL BUT HIT-BIT
	LI, R12	0		
QGBACK	LS, R12	0, R1		
	STW, R12	0, R1		
	AI, R1	2		
	BDR, R8	QGBACK		
	B	RXEXIT		EXIT TO RE-EXCHANGE LOCALS & GLOBALS
QSCORE	AW, R8	SYMTSIZE		GET ENTRY NO. FOR CANDIDATE.
	SLS, R8	1		MAKE IT A NAME PTR.
	LW, R3	R8		
	EOR, R13	*SYMT, R3		
	STW, R13	*SYMT, R3		SET ITS HIT-BIT BACK = 0.
	LI, R13	X'40000'		
	LW, R3	CONSTBUF+47		GET LATEST COLUMN INDICATOR.
	BAL, R13	GENNAME0		PUT NAME IN IMAGE BUFFER, NO INDENTS
	BAL, R13	GAP		PUT IN SPACING & COLUMNARIZATION.
	STW, R3	CONSTBUF+47		SAVE NEW COLUMN INDICATOR.
	LI, R3	QHUNT		START ANOTHER SYM TBL SWEEP AFTER
	B	QRESETHI		RESETTING THE CANDIDATE STRING TOO HIGH (SO ANY NAME WILL DO).
*				
	PAGE			
*				
*	@SIV-DRIVER TO LIST STATE VECTOR AND ASSOCIATED LOCAL VARIABLES			
*				
*	@SI-DRIVER TO LIST STATE VECTOR			
*				
@SIV	LI, R3	0		FLAG THAT THIS IS AN)SIV CMND.
@SI	STW, R3	CONSTBUF		SAVE)SI VS)SIV FLAG.
	AI, R2	-X'15'		VERIFY END-OF-STMT.
	BNEZ	SIOPTION		NO, CK FOR SI CONTROL OPTION.
	LW, R1	STATEPTR		PT AT TOP STATE-ENTRY IN STACK.
SIA	STW, R1	CONSTBUF+1		SAVE CURRENT STATE PTR.
	LW, R2	BREAKFLG		HAS A BREAK OCCURRED...
	BNEZ	CMDEXIT		YES -- EXIT
	LI, R2	X'7FFF'		GET WD COUNT TO NEXT STATE-ENTRY.
	AND, R2	0, R1		
	BEZ	CMDEXIT		ZERO (FINAL ENTRY) -- EXIT.
	STW, R2	CONSTBUF+2		SAVE IT.
	LI, R3	0		PRESET COLUMN INDICATOR.
	LB, R14	*CONSTBUF+1		WHAT CATEGORY IS CURRENT STATE...
	AI, R14	-CATF		
	BEZ	SIB		FUNCTION ENTRY.
	LI, R7	QUAD		EVALUATED-INPUT ENTRY.
	LI, R13	SIF		GO TO 'SIF' AFTER
	B	GENCHAR		PUTTING A QUAD IN IMAGE BUFFER.
SIB	LW, R14	1, R1		IS THIS A DAMAGED FUNCTION ENTRY...
	BEZ	SIC		YES, USE A BLANK FUNCTION NAME &

```

*                               LINE NUMBER.
SIC    BAL,R14  FUNLDISP      NO, SHOW FUNC.NAME & LINE NUMBER.
SID    LI,R13   SID          WE MAY LOOP VIA 'GENCHAR'.
      LI,R7     ' '         SET A BLANK.
      CI,R3     6           HAVE WE REACHED COLUMN 6...
      BL       GENCHAR      NO, PUT BLANK & LOOP.
      LI,R12   PENDFLAG     IS THIS A PENDENT FUNCTION...
      CW,R12   *CONSTBUF+1
      BANZ     SIE          YES, PUT BLANK.
      LI,R7    '*'         NO, PUT * TO SHOW SUSPENSION.
SIE    BAL,R13  GENCHAR
      LW,R13   CONSTBUF
      BEZ      SIV         TEST FLAG FOR )SIV CMND...
SIF    AI,R3    0          IT IS -- SHOW ANY SHADOWED NAMES.
      BEZ      SIG         ARE WE AT LEFT MARGIN...
      BAL,R12  DUMPLING    YES.
      LW,R1    CONSTBUF+1  NO, OUTPUT LAST IMAGE.
      AW,R1    CONSTBUF+2  PT AT NEXT STATE-ENTRY IN STACK.
      B        SIA
SIV    LW,R6    CONSTBUF+1  PT AT THE FUNCTION STATE-ENTRY.
      LW,R14   3,R6        GET NO.OF SHADOWED NAMES...
      BEZ      SIF         NONE.
      AI,R6    2
SIW    BAL,R13  GAP        GIVE SPACING & COLUMNARIZATION.
      AI,R6    2          PT AT SHADOWED NAME PTR.
      LW,R8    0,R6        GET THAT NAME PTR.
      BAL,R13  GENNAME
      BDR,R14  SIW
      B        SIF
SIOPTION BAL,R14 ACQIT      ACQUIRE OPTION NAME.
      LW,R8    NAMEBUF     GET OPTION...
      CW,R8    TEXTCLEA
      BE       SICLEAR     CLEAR.
      CW,R8    TEXTOFF
      BE       SIOFF       OFF.
      SW,R8    TEXTON      ON...
      BNEZ     ERBADCMD    (OOPS -- BAD COMMAND)
SIOFF   STW,R8  SICTRL     ON=0 & OFF=NZ=DON'T SUSP.FUN ON ERR.
      B        CMDEXIT
TEXTON  TEXT    'ON '
SICLEAR LW,R8    HICOMMON   PT AT LAST WD IN COMMON REGION.
      AI,R8    -STACKOFF   OFFSET TO 'FINAL' STATE ENTRY IN THE
      STW,R8    GOSTATE    STATE-INDICATOR.
      LI,R14   INPDIR      REQUEST DIRECT INPUT AFTER
      B        SICLR       CLEARING THE STATE-INDICATOR.

PAGE
*
* @CATCH-DRIVER TO 'CATCH' ASSIGNMENTS TO A VARIABLE NAME
*
@CATCH  CI,R2    X'15'      CK FOR END-OF-STMT...
      BNE      CATCH      NO, SET A CATCH.
      STW,R6    CATCHTBL   YES, CLEAR ANY CATCHES.
      STW,R6    CATCHTBL+2 NOTE--R6 IS NEGATIVE.
      B        CMDEXIT    EXIT.
CATCH   CI,R3    LASTCSV    DOES A NAME-START FOLLOW )CATCH...
      BLE      ERBADCMD   NO -- BAD COMMAND.
      BAL,R12  ACQNAME     ACQ VARIABLE NAME IF POSSIBLE...
      B        GRPSYMFL   OOPS -- SYM TBL FULL.
      B        GRPWSFL    OOPS -- WS FULL.
      STW,R6    CONSTBUF   OK, SAVE NAME PTR.
      BAL,R14  ACQIT      ACQ 'VIA'...
      LW,R8    NAMEBUF
      CW,R8    TEXTVIA
      BNE      ERBADCMD   DIDN'T -- BAD COMMAND.
      CI,R3    LASTCSV    DID, DOES NAME-START FOLLOW VIA...
      BLE      ERBADCMD   NO -- BAD COMMAND.
      BAL,R12  ACQNAME     ACQ FUNCTION NAME IF POSSIBLE...
      B        GRPSYMFL   OOPS -- SYM TBL FULL.
      B        GRPWSFL    OOPS -- WS FULL.
      LW,R7    CONSTBUF   OK, GET VARIABLE NAME PTR AGAIN.
      LI,R5    -4         = SIZE OF CATCHTBL.
CATCHN  LW,R12  CATCHTBL+4,R5 IS CATCH TBL ENTRY IN USE...
      BLZ      CATCHV     NO, FILL IT IN.
      CW,R7    CATCHTBL+4,R5 YES, FOR SAME VARIABLE NAME...
      BE       CATCHF     YEP, JUST CHANGE FUNC.NAME PTR.
      AI,R5    2          NOPE, TRY NEXT ENTRY.
      BLZ      CATCHN
      B        ERBADCMD   NO ROOM -- BAD COMMAND.

```

```

CATCHV STW,R7 CATCHTBL+4,R5 FILL VARIABLE NAME PTR ENTRY.
CATCHF STW,R6 CATCHTBL+5,R5 FILL FUNCTION NAME PTR ENTRY.
      B CMDEXIT EXIT.
TEXTVIA TEXT 'VIA '
PAGE
*
* @OBSERVE-DRIVER TO 'OBSERVE' INTERMEDIATE RESULTS
*
@OBSERVE AI,R2 -X'15' VERIFY END-OF-STMT.
      BNEZ ERBADCMD OOPS -- BAD COMMAND.
      LI,R2 -2 PREPARE FOR DIRECT INPUT MODE.
      LI,R5 PENDFLAG
      CW,R5 *STATEPTR IS TOP STATE PENDENT...
      BAZ OBSERVES NO, DIRECT INPUT COMING UP.
      LI,R2 -1 YES, EXECUTION NOW IN PROGRESS.
      STW,R2 OBSERVE TURN ON OBSERVATION SETTING.
OBSERVES STW,R2 OBSFLAG SET OBSERVE-CMD FLAG: -2 OR -1
      B CMDEXIT SO IT WILL BE -1 DURING NEXT EXEC.
PAGE
*
* FOROPEN-FORCED OPEN-ROUTINE TO FORE REOPEN OF FUNCTION WHICH
* HAS BEEN FORCED CLOSE
*
* ISSUES MESSAGE- DEL AND FUNCTION NAME (WITH CR)
* SIMULATES INPUT OF ISSUED MESSAGE
* GOES TO DIRECT INPUT DRIVER TO HANDLE INPUT
* (CODESTRINGER WILL RECOGNIZE OPENING DEL AND OPEN FN)
*
* OW ENTRY, OPENFN HAS NAME OF CLOSED FUNCTION
*
* REGISTERS ARE USED PROLIFICALLY AND NOT SAVED
*
FOROPEN LW,R7 BLANKS BLANK PROMPT
      STD,R7 IMAGE
      LI,R3 6
      LI,R7 X'7F' APL-DEL
      BAL,R13 GENCHAR OUTPUT IT (OR MNEMONIC)
      LI,R7 X'40' BLANK
      BAL,R13 GENCHAR
      LW,R8 OPENFN GET NAME PTR OF FN TO BE OPENED
      BAL,R13 GENNAME
      STW,R3 HICOL SAVE CHARACTER COUNT FOR SQUEEZER
      BAL,R12 DUMPLING OUTPUT THE LINE
      LB,R11 OPENFN RECOVER OLD MODE, U17-0010
      BNEZ REMODE IF PRESENT; U17-0011
      LI,R11 1 OTHERWISE, ASSUME DIRECT MODE. U17-0012
REMODE STW,R11 MODE U17-0013
      LI,R11 INPRET+1
      STW,R11 SAVE312+8 SET EXIT FROM 'APLINPUT'(SQUEEZER)
      B SQUEEZER USE TAIL-END OF APLINPUT ROUTINE
      TO REDUCE MNEMONIC IF PRESENT
PAGE
*****
SPACE 2
Z SET %-SCMD@ SIZE OF SCMD MODULE IN HEX.
SPACE
Z SET Z+Z/10*6+Z/100*96+Z/1000*1536 SIZE IN DECIMAL.
SPACE 3
END

```


	TITLE	'ERROR-B00,08/22/73,DWG702985'	
	SYSTEM	SIG7F	
	CSECT	1	
	PCC	0	CONTROL CARDS NOT PRINTED.
ERROR@	RES	0	ORIGIN OF ERROR MODULE.
*			
*	REF'S	AND	DEF'S
*			
	DEF	ERROR@	= START OF 'ERROR' MODULE.
	DEF	ENTEREM	PT OF ENTRY INTO THE ERROR MODULE.
SPACE 3			
*			REFS TO PROCEDURE:
	REF	EBALS	TBL OF BAL,R15 EXITS INTO ENTEREM.
	REF	ALOCBLK	ALLOCATES A DATA BLOCK.
	REF	ALOCNW	ALLOCS. DATA BLK W/HDR & EVENS SIZE.
	REF	DREF	DE-REFERENCES A DATA BLOCK.
	REF	MAYDREF	DE-REFS. DATA BLK, IF NON-ZERO PTR.
	REF	SICLR	CLEARs STATE-INDICATOR TO GO-STATE.
	REF	NIRETURN	RETURN AFTER NILADIC INTRINSIC OPER.
	REF	BCBRANCH	CLR BREAK & START ERR-CTRL'D BRANCH.
	REF	ECBRANCH	START ERR-CONTROLLED BRANCH.
	REF	BBADFL	EXIT TO USE BAD FUN.LINE (MAYBE).
	REF	BCONTOFF	EXIT AS IF A)CONTINUE CMD ISSUED.
	REF	CMDEXITM	EXIT ACCORDING TO MODE.
	REF	CMDEXITO	EXIT VIA MODE UNLESS OFF-LINE RUN.
	REF	INPDIR	EXIT FOR DIRECT INPUT.
	REF	INPEVAL	EXIT FOR EVALUATED INPUT.
	REF	INPLSCER	EXIT FOR INPUT AFTER LINESCAN ERR.
	REF	GENNAME	GENERATE A NAME IN IMAGE BUFFER.
	REF	GENNAME0	DITTO (ALT.ENTRY).
	REF	TEXTC2I	MOVE TEXTC MSG TO IMAGE.
	REF	FUNLDIS%	DISPLAY FUNCTION LINE.
	REF	EDECODOP	DECODE CODESTRING & DISPLAY ERROR.
	REF	DUMPLINP	DISPLAY LINE AS A PROMPT.
	REF	EDUMLIN	DISPLAY LINE IN ERROR.
	REF	EDUMLIG	DITTO (ALT.ENTRY).
	REF	EWRTXTC	WRITE TEXTC ERROR MSG.
	REF	EWROUTWB	WRITE ERROR INFO AT WORD BOUNDARY.
	REF	CLOSR	CLOSE & RELEASE FILE.
	REF	OBSERVEZ	EXIT PT AFTER AN OBSERVATION.
	REF	SINGOUT	DISPLAY VALUE OF OBSERVATION.
*			REFS TO CONTEXT:
	REF	ELINK	SOMETIME LINK TO ERROR ROUTINE.
	REF	ERRORID	INTERNAL I.D. FOR ERROR.
	REF	EREGS	@ R1-R6 SAVED AT ERR-CTRL TEST.
	REF	ERRLOC	@ ERROR LOC (CURRLNO & NAME PTR).
	REF	ERRNUM	@ ERROR NUMBER.
	REF	SICTRL	STATE-INDICATOR CTRL:
*			0 = ON -- SUSPEND, IF APPROPR.
*			NZ = OFF -- DON'T SUSPEND ON ERR.
	REF	STATEPTR	PTS AT TOP STATE-ENTRY.
	REF	GOSTATE	PTS AT GO-STATE ENTRY.
	REF	BRNVAL	ERR-CTRL BRANCH VALUE (OR NEGATIVE).
	REF	MODE	EXECUTION MODE:
*			-1 = FORCED CLOSE OF FUN DEFN.
*			0 = FUN DEFN.
*			1 = DIRECT INPUT.
*			2 = EVAL-INPUT OR EXECUTE-OPER.
	REF	OLDMODE	MODE WHEN FUN DEFN MODE BEGAN.
	REF	IOERCODE	I/O ERROR CODE,SUBCODE,DCB ADDR WD.
	REF	ON%OFF	ON OR OFF-LINE RUN FLAG.
	REF	BSPFLAG	BACKSPACE FLAG FOR TERMINAL TYPE.
	REF	ERRORCHR	ERROR MARKER CHAR FOR TERMINAL TYPE.
	REF	OUTRANST	OUTPUT TRANSLATION TBL FOR TERMINAL.
	REF	OBSAVE	REG.SAVE AREA DURING AN OBSERVATION.
	REF	SAVE14	LINKAGE HOLDER.
	REF	F:WS	DCB FOR WS FILE ACTIVITIES.
	REF	BREAKFLG	BREAK FLAG: <0 IF HANG-UP, =0 IF NO BREAK DETECTED, >0 IF BREAK DETECTED.
*			
*			
	REF	CURRCS	PTR TO CURRENT CODESTRING, IF ANY.
	REF	OFFSET	OFFSET.
	REF	WHATERR	DATA CONCERNING EXECUTION ERROR.
	REF	IMAGE	IMAGE BUFFER.
	REF	HICOL	INDICATES HIGH COLUMN IN IMAGE.
	REF	IMAGEPOS	POSITION INDICATOR FOR IMAGE BUFFER.
	REF	INBUF	INPUT OR OUTPUT BUFFER.
	REF	ERRCOL	COLUMN INDICATOR FOR ERROR.

	REF	WSIDNAME	WS I.D. NAME (TEXTC).
	REF	RESULT	PTR TO RESULT DATA BLK.
	REF	STRAYS	STRAY DATA BLK PTRS.
	REF	SYMT	PTS AT SYMBOL TABLE.
*			REFS TO CONSTANTS:
	REF	BLANKS	BLANK DBLWD.
*	REF	HEXTBL	TBL OF EBCDIC CHARS FOR HEX DIGITS.
			REFS TO EQU'S:
	REF	NSTRAYS	NO.OF NORMAL EXECUTION STRAYS.
*	REF	DWSIZIM	DBLWD SIZE OF IMAGE.
			INTERNAL ERROR I.D.'S FOR:
	REF	IDLSCAN	LINESCAN ERR
	REF	IDDEFN	DEFN ERR
	REF	IDNOTSAV	NOT SAVED, THIS WS IS ...
	REF	IDSIDAM	SI DAMAGE
	REF	IDNERASE	... NOT ERASED
	REF	IDNOTCPY	... NOT COPIED
	REF	IDNOTFND	... NOT FOUND
	REF	IDXEQERR	ERR DURING AN 'EXECUTE' OPERATION.
*			
*	STANDARD EQU'S		
*		REGISTERS	
R0	EQU	0	
R1	EQU	1	
R2	EQU	2	
R3	EQU	3	
R4	EQU	4	
R5	EQU	5	
R6	EQU	6	
R7	EQU	7	
R8	EQU	8	
R9	EQU	9	
R10	EQU	10	
R11	EQU	11	
R12	EQU	12	
R13	EQU	13	
R14	EQU	14	
R15	EQU	15	
*			
*	OTHER EQU'S		
*			
CATQ	EQU	7	QUAD-STATE CATEGORY.
CATF	EQU	10	FUNCTION-STATE CATEGORY.
EFLAG	EQU	X'10000'	MARKS 'EXECUTE' TYPE OF Q-ENTRY.
TYPETEXT	EQU	2	DATA BLK TYPE = TEXT.
TYPEINTG	EQU	3	DATA BLK TYPE = INTEGER.
FNOFF	EQU	5	FUN.DESCR. OFFSET TO FUN.NAME PTR.
XSIZEOFF	EQU	2	FUN.DESCR. OFFSET TO XSIZE WD.
PENDFLAG	EQU	X'8000'	MARKS PENDENT STATES.
LOCKFLAG	EQU	X'10000'	MARKS LOCKED FUNCTIONS.
IDLE	EQU	X'16'	IDLE CHARACTER.
*			
*	CONSTANTS		
*			
PENDFUN	DATA	X'A008000'	PENDENT FUNCTION STATE BITS.
PAGE			
*			
*	ERROR MESSAGES:		
*			
M%SYNTAX	TEXTC	'SYNTAX ERR'	
M%UNDEF	TEXTC	'UNDEFINED'	
M%NORES	TEXTC	'NO RESULT'	
M%IOERR	TEXTC	'I/O ERR '	
M%DOMAIN	TEXTC	'DOMAIN ERR'	
M%RANK	TEXTC	'RANK ERR'	
M%LENGTH	TEXTC	'LENGTH ERR'	
M%WSFULL	TEXTC	'WS FULL'	
M%SINGMX	TEXTC	'SING. MATRIX'	
M%FMFSYN	TEXTC	'FORMAT SYNTAX ERR'	
M%INDEX	TEXTC	'INDEX ERR'	
M%BADCH	TEXTC	'BAD CHAR'	
M%TRUNC	TEXTC	'TRUNCATED INPUT'	
M%OPENQT	TEXTC	'OPEN QUOTE'	
M%SYMFUL	TEXTC	'SYM TBL FULL'	
M%LSCAN	TEXTC	'LINESCAN ERR'	
M%DEFN	TEXTC	'DEFN ERR'	
M%SIDAM	TEXTC	'SI DAMAGE'	
M%BADCMD	TEXTC	'BAD COMMAND'	

M%NOTGRP	TEXTC	'NOT GROUPED'
M%2BIGLD	TEXTC	'TOO BIG TO LOAD'
M%FILREF	TEXTC	'BAD FILE REF'
M%FILBSY	TEXTC	'FILE IN USE'
M%WSNOTF	TEXTC	'WS NOT FOUND'
M%NOTSAV	TEXTC	'NOT SAVED, THIS WS IS '
M%NERASE	TEXTC	' NOT ERASED'
M%NOTCPY	TEXTC	' NOT COPIED'
M%NOTFND	TEXTC	' NOT FOUND'
M%TOOBIG	TEXTC	'TOO BIG'
M%2NAMEY	TEXTC	'TOO MANY SYMBOLS'
M%FILSPC	TEXTC	'FILE SPACE TOO LOW'
M%TERMAL	TEXTC	'WRONG TERMINAL'
M%FILNAM	TEXTC	'FILE NAME ERR'
M%NOTAPL	TEXTC	'NOT APL FILE'
M%FTFULL	TEXTC	'FILE TBL FULL'
M%FILACC	TEXTC	'FILE ACCESS ERR'
M%FILTIE	TEXTC	'FILE TIE ERR'
M%NOPACK	TEXTC	'PRIVATE PACK UNAVAIL, CALL OPR.'
M%FILIDX	TEXTC	'FILE INDEX ERR'
M%FILDAM	TEXTC	'FILE DAMAGE'
M%FIOERR	TEXTC	'FILE I/O ERR '

PAGE

IDMSGTBL	EQU	%+1	ERROR I.D.	VS MESSAGE LOCATION:
	DATA	0	-1	BREAK (NO MESSAGE)
	DATA	M%SYNTAX	0	LINESCAN ERR = SYNTAX ERR
	DATA	M%SYNTAX	1	SYNTAX ERR
	DATA	M%UNDEF	2	UNDEFINED
	DATA	M%NORES	3	NO RESULT
	DATA	-M%IOERR	4	I/O ERR
	DATA	M%DOMAIN	5	DOMAIN ERR
	DATA	M%RANK	6	RANK ERR
	DATA	M%LENGTH	7	LENGTH ERR
	DATA	M%WSFULL	8	WS FULL
	DATA	M%SINGMX	9	SING. MATRIX
	DATA	M%FMTSYN	10	FORMAT SYNTAX ERR
	DATA	M%INDEX	11	INDEX ERR
	DATA	0	12	XEQ ERR (NO MESSAGE)
	DATA	M%BADCH	13	BAD CHAR
	DATA	M%TRUNC	14	TRUNCATED INPUT
	DATA	M%OPENQT	15	OPEN QUOTE
	DATA	M%SYMFUL	16	SYM TBL FULL
	DATA	M%LSCAN	17	LINESCAN ERR
	DATA	M%DEFN	18	DEFN ERR
	DATA	M%SIDAM	19	SI DAMAGE
	DATA	M%BADCMD	20	BAD COMMAND
	DATA	M%NOTGRP	21	NOT GROUPED
	DATA	M%2BIGLD	22	TOO BIG TO LOAD
	DATA	M%FILREF	23	BAD FILE REF
	DATA	M%FILBSY	24	FILE IN USE
	DATA	M%WSNOTF	25	WS NOT FOUND
	DATA	M%NOTSAV	26	NOT SAVED, THIS WS IS ...
	DATA	M%NERASE	27	... NOT ERASED
	DATA	M%NOTCPY	28	... NOT COPIED
	DATA	M%NOTFND	29	... NOT FOUND
	DATA	M%TOOBIG	30	TOO BIG
	DATA	M%2NAMEY	31	TOO MANY SYMBOLS
	DATA	M%FILSPC	32	FILE SPACE TOO LOW
	DATA	M%TERMAL	33	WRONG TERMINAL
	DATA	M%FILNAM	34	FILE NAME ERR
	DATA	M%NOTAPL	35	NOT APL FILE
	DATA	M%FTFULL	36	FILE TBL FULL
	DATA	M%FILACC	37	FILE ACCESS ERR
	DATA	M%FILTIE	38	FILE TIE ERR
	DATA	M%NOPACK	39	PRIVATE PACK UNAVAIL, CALL OPR.
	DATA	M%FILIDX	40	FILE INDEX ERR
	DATA	M%FILDAM	41	FILE DAMAGE
	DATA	-M%FIOERR	42	FILE I/O ERR

PAGE

ERROR#TBL	EQU	%+1	ERROR I.D.	VS ERROR NUMBER TABLE:
	DATA	100	-1	BREAK
	DATA, 1	2	0	LINESCAN ERR (SYNTAX ERR ASSUMED)
	DATA, 1	2	1	SYNTAX ERR
	DATA, 1	3	2	UNDEFINED
	DATA, 1	8	3	NO RESULT
	DATA, 1	30	4	I/O ERR
	DATA, 1	4	5	DOMAIN ERR
	DATA, 1	5	6	RANK ERR

DATA,1	6	7	LENGTH ERR
DATA,1	1	8	WS FULL
DATA,1	15	9	SINGULAR MATRIX
DATA,1	16	10	FORMAT SYNTAX ERR
DATA,1	7	11	INDEX ERR
DATA,1	0	12	XEQ ERR
DATA,1	20	13	BAD CHAR
DATA,1	22	14	TRUNCATED INPUT
DATA,1	23	15	OPEN QUOTE
DATA,1	9	16	SYM TBL FULL
DATA,1	21	17	LINESCAN ERR
DATA,1	35	18	DEFN ERR
DATA,1	36	19	SI DAMAGE
DATA,1	40	20	BAD COMMAND
DATA,1	51	21	NOT GROUPED
DATA,1	45	22	TOO BIG TO LOAD
DATA,1	43	23	BAD FILE REF
DATA,1	42	24	FILE IN USE
DATA,1	44	25	WS NOT FOUND
DATA,1	41	26	NOT SAVED, THIS WS IS ...
DATA,1	50	27	... NOT ERASED
DATA,1	48	28	... NOT COPIED
DATA,1	49	29	... NOT FOUND
DATA,1	46	30	TOO BIG
DATA,1	47	31	TOO MANY SYMBOLS
DATA,1	70	32	FILE SPACE TOO LOW
DATA,1	31	33	WRONG TERMINAL
DATA,1	73	34	FILE NAME ERR
DATA,1	74	35	NOT APL FILE
DATA,1	75	36	FILE TBL FULL
DATA,1	76	37	FILE ACCESS ERR
DATA,1	77	38	FILE TIE ERR
DATA,1	78	39	PRIVATE PACK UNAVAIL, CALL OPR.
DATA,1	79	40	FILE INDEX ERR
DATA,1	72	41	FILE DAMAGE
DATA,1	71	42	FILE I/O ERR

SPACE 2

BOUND 4

PAGE

```

*****
*
*  DSTRAYS -- DEREFERENCES THE NORMAL STRAY DATA BLK PTRS ASSOCIATED
*           WITH EXECUTION.
*  REGS:    R14 LINK, EXIT VIA *R14
*           R4,R5,R7 ARE VOLATILE.
*
DSTRAYS  LI,R5    NSTRAYS          = NO.OF NORMAL EXEC. STRAYS.
DSTRAY   LI,R4    0
          XW,R4    STRAYS-1,R5
          BAL,R7   MAYDREF          DEREFERENCE, IF NON-ZERO PTR.
          BDR,R5   DSTRAY
          B        *R14             EXIT.
PAGE
*****
*
*  ECTEST -- ERROR CONTROL TESTER.
*
*  IDECTEST -- ALT.ENTRY PT. ENTERED WITH THE ERROR I.D. IN R8. THIS
*             IS SAVED IN ERRORID & R8 IS GIVEN THE ERROR NUMBER.
*
*  REGS:      R8 (ENTRY) CONTAINS THE ERROR NUMBER (BYTE SIZE).
*             R7 LINK -- EXIT 0,R7 IF CONTROLLED, BRNVAL SET MAX.OF
*             ERR-CTRL LINE NO. VALUE OR ZERO.
*             EXIT 1,R7 IF UNCONTROLLED, BRNVAL SET NEG.
*             ALL REGS. PRESERVED.
*
IDECTEST  STW,R8    ERRORID          SAVE ERROR I.D.
          STW,R7    ERRLOC           SAVE R7.
          LW,R7     ERRORID
          LB,R8     ERRO#TBL,R7      GET ERROR NUMBER.
          LW,R7     ERRLOC           RESTORE R7.
ECTEST    LCI      8                SAVE R1-R6, USE R7 TO 'CLEAR' ERRLOC
          STM,R1    EREGS            AND PUT R8 IN ERRNUM.
          LW,R1     STATEPTR         PT AT TOP STATE ENTRY.
          LI,R2     EFLAG            SET FOR 'EXECUTE' STATE TESTS.
          B         ECTESTS
ECTESTNS  LI,R6     X'7FFF'          PT AT THE NEXT STATE ENTRY.
          AND,R6    0,R1

```

ECTESTS	AW, R1	R6		TEST FOR 'EXECUTE'...
	CW, R2	0, R1		YES, TRY AGAIN.
	BANZ	ECTESTNS		
	LI, R3	X'F8000'		
	AND, R3	0, R1		TEST FOR PENDENT FUNCTION...
	CW, R3	PENDFUN		YES.
	BE	ECTESTPF		NO, SAVE PTR TO SUSP. OR EVAL-INP.
	STW, R1	GOSTATE		
	LI, R2	-1		SET BRNVAL NEG. = UNCONTROLLED ERROR
	STW, R2	BRNVAL		
	LCI	6		
	LM, R1	EREGS		RESTORE USED REGS.
	B	1, R7		EXIT (UNCONTROLLED).
ECTESTPF	LW, R5	1, R1		GET LINE # & FDEFPTR WD.
	CW, R7	ERRLOC		HAS THE ERROR LOC BEEN UPDATED...
	BNE	ECTESTCT		YES.
	LI, R4	X'E0000'		NO, SET IT NOW --
	AND, R4	1, R1		CURRLINO &
	AW, R4	FNOFF, R5		FUNCTION NAME PTR.
	STW, R4	ERRLOC		
ECTESTCT	LW, R4	XSIZOFF, R5		DOES PEND FUNC HAVE ERR-CTRL TBL...
	CI, R4	X'1FFFF'		
	BL	ECTESTNS		NO, TRY AGAIN.
	AI, R4	2		YES, PT AT # OF ROWS.
	LW, R6	0, R4		GET # OF ROWS.
ECTESTCR	AI, R4	2		PT AT NEXT ROW OF ERR-CTRL TBL.
	LW, R5	1, R4		GET ERR-CTRL NO...
	BEZ	ECTESTC		ZERO -- CONTROL THIS ERROR.
	CW, R5	ERRNUM		NZ, DOES IT MATCH THE ERROR NO...
	BE	ECTESTC		YES, CONTROL THIS ERROR.
	BDR, R6	ECTESTCR		NO, LOOP THRU ERR-CTRL TBL...
	B	ECTESTNS		NO SCORE, TRY AGAIN.
ECTESTC	LW, R2	0, R4		GET ERR-CTRL LINE NO. VALUE...
	BGEZ	ECTESTB		
	LI, R2	0		USE ZERO IF THAT VALUE IS NEG.
ECTESTB	STW, R2	BRNVAL		SET BRNVAL = BRANCH VALUE FOR ERROR.
	STW, R1	GOSTATE		SAVE PTR TO ERR-CTRL'D FUN STATE.
	LCI	6		
	LM, R1	EREGS		RESTORE USED REGS.
	B	0, R7		EXIT (CONTROLLED).
PAGE				
* COMMUNICATIONS REGION -- FROM ROOT INTO ERROR MODULE -- VIA R15.				
* IT IS INTENDED THAT ALL ENTRIES TO THIS MODULE BE CHanneled THRU HERE				
ENTEREM	STW, R7	ELINK		SAVE R7 FOR PROCESSES NEEDING THAT.
	AI, R15	BE-EBALS		CALC. COMMUNICATION BRANCH.
BE	B	*R15	@	ENTER APPROPRIATE PROCESS.
	B	ERRN	@	
	B	ERRF	@	
	B	ERRX	@	
	B	CMDERRH	@	
	B	IDECTEST	@	
	B	CTRL	@	
	B	DONTSAVH	@	
	B	DERASE	@	
	B	DNOTFND	@	
	B	DNOTCPY	@	
	B	DSIDAM	@	
	B	DEFNERRH	@	
	B	LSCANERH	@	
	B	UNREF	@	
	B	ERRSET	@	
	B	OBSERVER	@	
	B	FUNDERRH	@	
* FUNDERRH	RES	0	@	
	BAL, R7	IDECTEST	@	RECORD ERROR & TEST ERR-CTRL...
	B	*ELINK	@	CONTROLLED -- RESUME FUNDEF AWHILE
	B	EIDTEXTC	@	UNCONTROLLED -- DISPLAY & RESUME.
LSCANERH	LI, R5	F:WS		RELEASE WS IF OPEN DUE TO ERROR ON
	BAL, R6	CLOSR		AN AUTOCONTINUE CODESTRING ERR.
	LI, R8	IDLSCAN		= ERROR I.D. FOR 'LINESCAN ERR'.
	BAL, R7	IDECTEST	@	RECORD ERROR & TEST ERR-CTRL...
	B	BBADFL	@	CONTROLLED -- MUST BE BAD FUN.LINE
	LW, R7	ERRCOL	@	UNCONTROLLED.
	BAL, R11	GENCARMS		DISPLAY MSG WITH AN ERROR MARKER.
	LW, R7	MODE		CK MODE...

	BLEZ	BBADFL	FUN.DEFN -- MAY USE BAD FUN.LINE.
	LW,R10	ON%OFF	IF HANG-UP OCCURRED.
	BEZ	BCONTOFF	DO LIKE A CONTINUE CMD.
	LW,R10	BSPFLAG	DOES TERMINAL ADMIT BACKSPACING...
	BEZ	CMDEXITM	NO, EXIT VIA MODE AS IF A CMD.
	LW,R3	ERRCOL	YES, DISPLAY ACCEPTABLE
	LI,R12	INPLSCER	PORTION OF LINE
	B	DUMPLINP	AS A PROMPT.
DEFNERRH	LI,R8	IDDEFN	= ERROR I.D. FOR 'DEFN ERR'.
	BAL,R7	IDECTEST @	RECORD ERROR & TEST ERR-CTRL...
	B	CMDEXITM @	CONTROLLED -- EXIT VIA MODE.
	STW,R1	ERRCOL @	UNCONTROLLED.
	LI,R11	EFLAG	WAS THIS AN 'EXECUTED' FUN DEFN...
	CW,R11	OLDMODE	
	BAZ	DEFNERRM	NO.
	LI,R3	-BA(IMAGE)	YES, DISPLAY THE OFFENDING LINE.
	AW,R3	HICOL	
	BAL,R12	EDUMPLIN	
DEFNERRM	LW,R7	ERRCOL	
	LI,R11	CMDEXITO	GO TO CMDEXITO AFTER
*	B	GENCARMS @	DISPLAYING ERR MSG & ERROR MARKER.
GENCARMS	AI,R7	-BA(IMAGE) @	
	STW,R7	ERRCOL	SET ERROR POSITION IN THE LINE.
	LW,R6	ERRORID	
	LW,R6	IDMSGTBL,R6	GET LOC OF (TEXTC) ERR MSG.
	LB,R8	*R6	GET BYTE COUNT OF ERR MSG.
	LW,R10	R8	SAVE BYTE COUNT FOR LOOP
	AI,R8	1	ADD 1 TO ERR MSG SIZE FOR FITTING.
	SLS,R6	2	
	AI,R6	1	= BA(ERROR MSG STRING).
	LI,R5	DWSIZIM	
	LW,R9	BLANKS	
BLANKIT	STD,R9	INBUF-2,R5	BLANK THE OUTPUT BUFFER.
	BDR,R5	BLANKIT	
	LI,R5	IDLE	
	STB,R5	IMAGE,R7	PUT IDLE IN BAD SPOT IN IMAGE.
	LB,R5	ERRORCHR	
	LB,R5	OUTRANST,R5	GET TERMINAL'S ERROR MARKER &
	STB,R5	INBUF,R7	PUT IT IN THE OUTPUT BUFFER.
	LW,R9	R7	
	SW,R7	R8	CK IF MESSAGE FITS ON LEFT...
	BGEZ	SETMESS	YES
	AW,R7	R8	NO, MOVE TO RIGHT OF MARKER.
	AI,R7	2	
	AW,R9	R8	ADJUST SIZE.
SETMESS	LB,R5	0,R6	
	LB,R5	OUTRANST,R5	TRANSLATE BYTE TO OUTPUT FORM
	STB,R5	INBUF,R7	AND PUT IN INBUF
	AI,R6	1	
	AI,R7	1	
	BDR,R10	SETMESS	LOOP
	LI,R8	INBUF	
	AI,R9	1	SIZE.
	LW,R7	R11	EXIT FROM GENCARMS AFTER
	B	EWROUTWB	DISPLAYING THE DIAGNOSTIC.
DONTSABV	LI,R8	IDNOTSAV	= ERROR I.D. FOR 'NOT SAVED, ETC.'.
	BAL,R7	IDECTEST @	RECORD ERROR & TEST ERR-CTRL...
	B	CTRL @	CONTROLLED.
	LW,R7	ERRORID @	UNCONTROLLED.
	LW,R8	IDMSGTBL,R7	GET LOC OF (TEXTC) ERR MSG.
	LI,R3	0	INIT. CHAR POSITION.
	BAL,R13	TEXTC2I	MOVE ERR MSG INTO IMAGE BUFFER.
	LI,R8	WSIDNAME	PT AT (TEXTC) WS I.D.
	BAL,R13	TEXTC2I	APPEND THAT TO ERR MSG IN IMAGE.
	LI,R12	CMDEXITO	GO TO CMDEXITO AFTER
	B	EDUMLIG	DISPLAYING THE DIAGNOSTIC.
CMDERRH	BAL,R7	IDECTEST @	RECORD ERROR & TEST ERR-CTRL...
	B	CTRL @	CONTROLLED.
EIDTEXTC	LW,R7	ERRORID @	UNCONTROLLED, GET ERROR I.D. AGAIN
MSGTEXTC	LW,R8	IDMSGTBL,R7	GET LOC OF (TEXTC) ERR MSG.
	LW,R7	ELINK	EXIT AFTER
*	B	WRTEXTCQ	@ DISPLAYING THE ERR MSG.
WRTEXTCQ	AI,R8	0	@ IS R8 POSITIVE...
	BGZ	EWRTXTXC	YES, NORMAL MSG.
	LCW,R8	R8	NO, I/O OR FILE I/O ERR MSG.
	LI,R3	0	INIT. CHAR POSITION.
	BAL,R13	TEXTC2I	MOVE ERR MSG INTO IMAGE BUFFER.
	LW,R12	R7	PUT LINK IN R12 SO EDUMLIG IS EXIT.

HEXIO	LI, R14	WRTEXTCX	EXIT FROM 'HEXIO' TO 'WRTEXTCX'.
	LW, R7	IOERCODE	GET I/O ERR CODE & SUBCODE.
	SLD, R6	8	MOVE CODE OVER.
	SLS, R7	-1	RIGHT-JUSTIFY SUBCODE.
	SLD, R6	-8	MOVE CODE BACK.
	LI, R11	4	PRODUCE 4 HEX DIGITS.
HEXCHAR	LI, R6	0	
	SLD, R6	4	
	LB, R6	HEXTBL, R6	GET EBCDIC VERSION OF A HEX DIGIT.
	B	*R14	
WRTEXTCX	STB, R6	IMAGE, R3	PUT HEX CHAR IN IMAGE.
	AI, R3	1	PT AT NEXT POS.
	BDR, R11	HEXCHAR	LOOP TILL LAST HEX CHAR PUT.
	B	EDUPLIG	DISPLAY DIAG., LINK IS R12.
DSIDAM	LI, R8	-IDSIDAM	RECORD NEG. ERROR I.D. (DISPLAY NOW,
	STW, R8	ERRORID	TEST ERR-CTRL LATER).
	LI, R7	IDSIDAM	= ERROR I.D. FOR 'SI DAMAGE'.
	B	MSGTEXTC	DISPLAY & RESUME PROCESSING.
DERASE	LI, R8	-IDNERASE	RECORD NEG. ERROR I.D. (DISPLAY NOW,
	STW, R8	ERRORID	CK ERR-CTRL LATER)--'NOT ERASED'.
	LI, R3	0	INIT. CHAR POSITION.
	LW, R8	R6	GET NAME PTR, I.E. SYMT OFFSET.
	BAL, R13	GENNAME	GEN. NAME INTO IMAGE BUFFER.
NAMEGEND	LCW, R7	ERRORID	GET ERROR I.D.
	LW, R8	IDMSGTBL, R7	GET LOC OF (TEXTC) ERR MSG.
	BAL, R13	TEXTC2I	APPEND ERR MSG TO NAME IN IMAGE.
	LW, R12	R14	EXIT AFTER
	B	EDUPLIG	DISPLAYING DIAGNOSTIC.
DNOTCPY	LI, R8	-IDNOTCPY	RECORD NEG. ERROR I.D. (DISPLAY NOW,
	STW, R8	ERRORID	CK ERR-CTRL LATER)--'NOT COPIED'.
	LI, R3	0	INIT. CHAR POSITION.
	LW, R8	R1	GET NAME PTR, I.E. SYMT OFFSET.
	LI, R13	NAMEGEND	GO TO NAMEGEND AFTER
	B	GENNAME0	GEN. NAME INTO IMAGE BUFFER.
DNOTFND	LI, R8	-IDNOTFND	RECORD NEG. ERROR I.D. (DISPLAY NOW,
	STW, R8	ERRORID	CK ERR-CTRL LATER)--'NOT FOUND'.
	LI, R8	' '	NAME IS IN IMAGE BUFFER WITH A
	LI, R3	-1	TRAILING BLANK.
DNOTFND1	AI, R3	1	FIND THE CHAR POSITION
	CB, R8	IMAGE, R3	OF THAT BLANK.
	BNE	DNOTFND1	
	B	NAMEGEND	

PAGE

```

*
* ERRSET IS ENTERED VIA SETERR IN THE INTRINS MODULE. IT HANDLES THE
* END ACTIONS TO BE TAKEN ON AN EXECUTION ERROR. THERE ARE
* REALLY TWO TYPES OF EXECUTION ERRORS -- THE USUAL ONES (E.G.
* DOMAIN, RANK, SYNTAX, ETC.) ASSOCIATED WITH EXECUTION AND
* COMMAND OR FUNCTION DEFINITION ERRORS RESULTING VIA AN
* EXECUTE-OPERATOR (THESE ARE CALLED XEQ ERRORS). IN THE XEQ
* ERROR CASE, ERROR-CONTROL HAS ALREADY BEEN TESTED.
*

```

ERRSET	STW, R2	ERRORID	SAVE EXECUTION-ERROR I.D. VALUE.
	LB, R8	ERRO#TBL, R2	GET ERROR NUMBER...
	BNEZ	TESTCTRL	USUAL TYPE.
	LW, R8	BRNVAL	XEQ ERR, IS IT A CONTROLLED ERR...
	BGEZ	CTRL	YES.
	B	UNCTRL	NO.
TESTCTRL	BAL, R7	ECTEST	@ TEST ERROR CONTROL...
	B	CTRL	@ CONTROLLED ERROR.
UNCTRL	LB, R4	*STATEPTR	@ UNCONTROLLED ERROR.
	STB, R4	WHATERR	SAVE F OR Q STATE CATEGORY...
	AI, R4	-CATQ	
	BNEZ	ERRFSTAT	F -- FUNCTION OR FINAL STATE.
	LI, R4	EFLAG	Q -- EVAL-INPUT OR EXECUTE...
	CW, R4	*STATEPTR	
	BAZ	ERRS	EVAL-INPUT STATE.
	LW, R4	CURRCS	EXECUTE STATE.
	BNEZ	ERRXEQCS	CODESTRING IS AVAILABLE FOR ERR DIAG
	LW, R4	STATEPTR	CODESTRING NOT YET AVAILABLE.
	LI, R3	X'1FFFF'	
	AND, R3	1, R4	PT AT CODESTRING CONTAINING THIS
	STW, R3	CURRCS	EXECUTE-OPERATOR.
	MTW, 1	-1, R3	BUMP REF-COUNT OF CODESTRING DATBLK
	LW, R3	1, R4	
	SLS, R3	-17	RESTORE BYTE OFFSET USED WHEN THIS
	STW, R3	OFFSET	EXECUTE WAS RECOGNIZED.

	LI, R3	X'7FFF'	
	AND, R3	0, R4	UPDATE TO NEXT STATE ENTRY.
	AWM, R3	STATEPTR	
	B	UNCTRL	
ERRXEQCS	BAL, R14	EXERRLNK	HANDLE ERR DIAGNOSTIC & DEREf CURRCS
	LW, R2	ERRORID	WAS IT A BREAK 'MESSAGE'...
	BLZ	UNCTRL	YES, RETAIN BREAK I.D.
	LI, R2	IDXEQERR	
	STW, R2	ERRORID	NO, SWITCH TO XEQ ERROR I.D.
	B	UNCTRL	
ERRFSTAT	LI, R4	PENDFLAG	IS F-ENTRY PENDENT...
	CW, R4	*STATEPTR	
	BANZ	ERRP	YES.
	STB, R4	WHATERR	NO, SET CAT=0 -- DIRECT LINE ERR.
	B	ERRS	
ERRP	LW, R4	ERRORID	WAS THIS A BREAK...
	BLZ	ERRSUSP	YES, SUSPEND THE FUNCTION.
	LW, R4	SICTRL	TEST STATE INDICATOR CONTROL...
	BNEZ	ERRS	OFF -- DON'T SUSPEND THE FUNCTION.
ERRSUSP	LI, R4	-PENDFLAG-1	ON -- SUSPEND IT.
	AND, R4	*STATEPTR	
	STW, R4	*STATEPTR	
	LW, R4	STATEPTR	MAKE IT THE 'GO' STATE.
	STW, R4	GOSTATE	
ERRS	BAL, R14	DSTRAYS	DEREF ANY EXECUTION STRAYS.
	LI, R14	EXERR	ISSUE LAST ERR DIAGNOSTIC AFTER
	B	SICLR	CLEARING THE STATE INDICATOR DOWN
			TO THE 'GO' STATE.
*			
PAGE			
*****			*****
*			*
*	EXERR	-- EXECUTION ERROR DIAGNOSTIC DRIVER. USUALLY DISPLAYS AN	*
*		ERROR MSG & OFFENDING LINE WITH ERROR MARKER. EXITS TO	*
*		INPDIR OR INPEVAL.	*
*			*
*	EXERRLNK	-- ALT. ENTRY PT CALLED FOR 'EXECUTE' OPERATOR ERRORS	*
*		(LINK VIA R14).	*
*			*
*	ALL REGS	SHOULD BE CONSIDERED VOLATILE.	*
*			*
EXERRLNK	STW, R14	SAVE14	SAVE LINK FOR RETURN VIA UNREF.
	LW, R7	BREAKFLG	CK FOR BREAK OR HANGUP...
	BNEZ	UNREF	YES, NO DISPLAY.
	B	EXERR1	NO.
EXERR	LI, R14	EXERREX	SET RETURN FROM UNREF.
	STW, R14	SAVE14	
	LCW, R7	BREAKFLG	CK FOR BREAK...
	BGEZ	EXERR1	NOT ON OR ELSE HANGUP.
	AWM, R7	BREAKFLG	ON, RESET BREAK FLAG.
EXERR1	LW, R7	ERRORID	GET ERROR IDENTIFIER.
	LW, R8	IDMSGTBL, R7	GET ERR MSG LOC...
	BEZ	EXNOMSG	NONE (BREAK OR XEQ ERR).
	BAL, R7	WRTEXTCQ	DISPLAY ERR MSG.
EXNOMSG	LB, R7	WHATERR	GET STATE CATEGORY FOR ERROR PT...
	AI, R7	-CATF	
	BEZ	FLINERR	FUNCTION LINE.
	LW, R7	BLANKS	DIRECT OR EVAL INPUT LINE OR XEQ.
	STD, R7	IMAGE	BLANK BYTES D-7 OF IMAGE.
	LI, R3	6	SET FOR COLUMN 7.
SETIMPOS	STW, R3	IMAGEPOS	= START PT FOR DECODE OPS.
	LW, R7	ERRORID	CK FOR BREAK I.D...
	BGEZ	EDECODOP	NO, DISPLAY ERR LINE & MARKER.
UNREF	LW, R7	SAVE14	SET RETURN.
	CI, R7	OBSLINE+1	WAS THIS AN OBSERVATION...
	BE	0, R7	YES -- RETURN.
	LI, R4	0	NO, CLEAR & GET CODESTRING PTR.
	XW, R4	CURRCS	
	AI, R4	-2	PT AT CODESTRING DATA BLK HEADER.
*			RETURN IS VIA R7.
			DE-REFERENCE THE CODESTRING DATA BLK
EXERREX	B	DREF	
	LI, R11	0	
	STW, R11	ERRORID	CLEAR THE ERROR I.D.
	LW, R11	ON%OFF	ON OR OFF-LINE...
	BEZ	BCONTOFF	OFF LINE -- DO LIKE CONTINUE CMD.
	LB, R11	*STATEPTR	ON LINE.
	AI, R11	-CATQ	IS EVAL-INPUT THE TOP STATE NOW...
	BNEZ	INPDIR	NO -- DO DIRECT INPUT.
	B	INPEVAL	YES -- DO EVAL INPUT AGAIN.

FLINERR	LW, R6	ERRLOC	GET CURRLINO (& FUNCTION NAME PTR).
	LI, R8	X'FFFF'	
	AND, R8	ERRLOC	GET FUNCTION NAME PTR.
	BAL, R14	FUNLDIS%	GEN. FUN NAME & LINE NO.
	LI, R12	UNREF	SET EXIT TO UNREF AFTER DISPLAY.
	LW, R8	ERRORID	CK FOR BREAK I.D...
	BLZ	EDUMLIN	YES -- DISPLAY ONLY FUN NAME & LN.
	LI, R8	LOCKFLAG	IS FUNCTION LOCKED...
	AW, R6	SYMT	
	LW, R6	0, R6	
	CW, R8	0, R6	
	BAZ	SETIMPOS	NO.
	B	EDUMLIN	YES -- DISPLAY ONLY FUN NAME & LN.
PAGE			
* OBSERVER -- DISPLAYS AN OBSERVATION: LINE & MARKER PLUS A VALUE.			
* OBSERVER			
	LW, R15	ERRLOC	SAVE ERROR LOCATION.
	LCI	0	SAVE ALL REGS.
	STM, R0	OBSAVE	
	LI, R2	IDXEQERR	CLAIM THIS IS AN 'XEQ ERR' TO AVOID
	STW, R2	ERRORID	DISPLAYING AN ERROR MSG.
	LB, R4	*STATEPTR	
	STB, R4	WHATERR	SAVE F OR Q STATE CATEGORY...
	AI, R4	-CATQ	
	BEZ	OBSLINE	Q-STATE.
	LI, R4	PENDFLAG	F-STATE.
	CW, R4	*STATEPTR	PENDENT OR SUSPENDED...
	BANZ	OBSFLINE	PENDENT (SHOW FUN.NAME & LN).
	STB, R4	WHATERR	SUSPENDED (DIRECT INPUT LINE).
OBSLINE	BAL, R14	EXERRLNK	@ SHOW LINE & MARKER, DON'T DE-CURRCS.
	LW, R4	OBSAVE+6	@ PT AT DATA BLK TO BE OBSERVED.
	BAL, R14	SINGOUT	@ DISPLAY THAT VALUE.
	B	%+1	@ SCREW ERROR RETURN.
	LCI	0	@ NORMAL RETURN.
	LM, R0	OBSAVE	RESTORE ALL REGS.
	STW, R15	ERRLOC	RESTORE ERROR LOCATION.
	B	OBSERVEZ	EXIT FROM THIS OBSERVATION.
OBSFLINE	LW, R4	STATEPTR	PT AT THE F-STATE.
	LI, R14	X'E0000'	EXTRACT CURR.LINE NO.
	AND, R14	1, R4	
	LW, R4	1, R4	PT AT FUNC.DESCR.
	AW, R14	FNOFF, R4	INCLUDE FUN.NAME PTR.
	STW, R14	ERRLOC	TEMP. SET ERRLOC.
	B	OBSLINE	
PAGE			
CTRL	LI, R4	0	CLEAR & TEST CODESTRING PTR...
	STW, R4	ERRORID	(CLEAR THE ERROR I.D.)
	XW, R4	CURRCS	
	BEZ	CTRLS	NOT CURRENTLY IN USE.
	AI, R4	-2	USED, PT AT CODESTRING D.B. HDR.
	BAL, R7	DREF	DEREFERENCE THAT DATA BLOCK.
CTRLS	BAL, R14	DSTRAYS	DEREF ANY EXECUTION STRAYS.
	BAL, R14	SICLR	CLEAR STATE INDIC. DOWN TO 'GO' STATE
	LCW, R12	BREAKFLG	TEST BREAK FLAG...
	BGEZ	ECBRANCH	OFF OR HANGUP -- DO ERR-CTRL BRNCH
	B	BCBRANCH	ON -- RESET & DO ERR-CTRL BRANCH.
ERRN	LI, R11	6	
	BAL, R7	ALOCBLK	ALLOCATE 6-WD DATA BLK.
	STW, R4	RESULT	SAVE PTR TO RESULT DATA BLK.
	LI, R11	(TYPEINTG**8)+1	TYPE = INTEGER & RANK = 1.
	STH, R11	*RESULT	
	LI, R11	2	LENGTH = 2.
	STW, R11	2, R4	
	LW, R11	ERRNUM	
	STW, R11	3, R4	SET ERROR NUMBER.
	LW, R11	ERRLOC	
	SLS, R11	-17	SET ERROR LINE NUMBER (OR ZERO).
	STW, R11	4, R4	
	B	NIRETURN	RETURN FROM NILADIC INTRINSIC.
ERRF	LI, R12	0	PREPARE FOR EMPTY ERROR LOC.
	LW, R2	ERRLOC	IF FUNC.ERR, GET CURRLINO & NAME PTR
	CI, R2	X'E0000'	IS THERE A CURRENT LINE NO...
	BAZ	EFNALOC	NO -- USE EMPTY FUNCTION NAME.
	AI, R2	1	= OFFSET TO FUN'S NAME INDIC. WD.
	LW, R3	*SYMT, R2	GET NAME INDICATOR WD.
	LB, R12	R3	GET # WDS FOR THAT NAME (UNLESS IT
	CI, R12	20	IS A SHORT NAME).

	BLE	EFNALOC	LONG NAME.
	LI, R12	1	SHORT NAME, 1 WD IN R3 ALREADY.
EFNALOC	LW, R11	R12	= NO.OF WDS FOR FUNCTION NAME.
	AI, R11	1	ADD 1 FOR THE VECTOR LENGTH WD.
	BAL, R7	ALOCHNW	ALLOC DATA BLK, INCLUDING HEADER.
	STW, R4	RESULT	SAVE PTR TO RESULT DATA BLK.
	LI, R11	(TYPETEXT**8)+1	TYPE = TEXT & RANK = 1.
	STH, R11	*RESULT	
	AI, R4	2	PT AT LENGTH WD.
	LW, R11	R12	GET # WDS FOR NAME...
	BEZ	EFNLEN	NONE (EMPTY TEXT VECTOR).
	CI, R11	1	
	BE	EFNMOV	ONE (NAME WD IS ALREADY IN R3).
	LW, R5	*SYMT, R2	LONG, PT AT 1ST WD OF LONG NAME.
EFNGET	LW, R3	0, R5	GET NAME WD.
	AI, R5	1	PT AT NEXT WD OF LONG NAME DATA BLK.
EFNMOV	AI, R4	1	PT AT NEXT WD OF RESULT.
	STW, R3	0, R4	STORE NAME WD.
	BDR, R12	EFNGET	LOOP TILL LAST NAME WD STORED.
	SW, R4	R11	PT AT LENGTH WD AGAIN.
	SLS, R11	2	= NO.OF BYTES MOVED.
EFNBLCK	CB, R3	BLANKS	CK FOR TRAILING BLANK IN NAME...
	BNE	EFNLEN	NO, R11 = # CHARS IN NAME.
	SLS, R3	-8	YES, DROP THAT BLANK.
	BDR, R11	EFNBLCK	DECR. BYTE COUNT & LOOP.
EFNLEN	STW, R11	0, R4	SET TEXT VECTOR'S LENGTH.
	B	NIRETURN	RETURN FROM NILADIC INTRINSIC.
ERRX	LI, R11	4	
	BAL, R7	ALOCBLK	ALLOCATE 4-WD DATA BLK.
	STW, R4	RESULT	SAVE PTR TO RESULT DATA BLK.
	LI, R11	(TYPETEXT**8)+1	TYPE = TEXT & RANK = 1.
	STH, R11	*RESULT	
	LI, R11	4	LENGTH = 4.
	STW, R11	2, R4	
	AI, R4	3	PT AT VALUE WORD.
	SLS, R4	2	PT AT 1ST VALUE BYTE POS.
	BAL, R14	HEXIO	START GETTING I/O ERR CODE & SUBCODE
	STB, R6	0, R4	SET VALUE BYTE (EBCDIC HEX DIGIT).
	AI, R4	1	PT AT NEXT VALUE BYTE.
	BDR, R11	HEXCHAR	LOOP TILL LAST HEX CHAR SET.
	B	NIRETURN	RETURN FROM NILADIC INTRINSIC.

PAGE

SPACE 2

Z SET %-ERROR@ SIZE IN HEX.

SPACE

Z SET Z+Z/10*6+Z/100*96+Z/1000*1536 SIZE IN DECIMAL.

SPACE 3

END

TITLE 'GRAF-B00,09/11/73,DWG702985'
PAGE

EXTERNAL COMMUNICATION

DEFINITIONS

DEF	DELTA GRF	DELTA-GRF FUNCTION
DEF	GRAF@	START OF PROCEDURE
DEF	GRFIOFLG	GRAF TERM I/O FLAG
DEF	GRINCHAR	GRAF TERM INPUT CHAR
DEF	GRXYBYTES	GRAF TERM X/Y COORD BYTES
DEF	NEXTLINE	RESTORE CURSOR POSITION
DEF	QZINPUT	QUAD-ZERO INPUT DRIVER
DEF	QZOUTPUT	QUAD-ZERO OUTPUT DRIVER

REFERENCES

REF	ALOCHNW	ALLOCATE HEADER AND N WORDS
REF	BREAKFLG	BREAK FLAG
REF	CREALBIN	CONVERT REAL TO DEC MANTISSA/EXPON
REF	DIGITS	'DIGITS' VALUE
REF	DXRETURN	DYADIC FUNCTION RETURN
REF	ERDOMAIN	DOMAIN ERROR
REF	ERLENGTH	LENGTH ERROR
REF	ERRANK	RANK ERROR
REF	ERTERMAL	TERMINAL TYPE ERROR
REF	F2I	FLOT TO INTG CONVERSION
REF	GDIGITS	'DIGITS' VALUE FOR AXIS LIMITS
REF	GENDIGS	GEN MANTISSA DIGITS
REF	GENDIGSE	GEN MANTISSA DIGITS, E-FORMAT
REF	GENEXP	GEN EXPONENT CHARS
REF	GETCHAR	GET INPUT CHARACTER/COORDS
REF	GETCOORD	READ CURSOR LOC
REF	GRAFBUF	GRAPHICS I/O BUFFER
REF	GRAFOUT	GRAPHICS OUTPUT ROUTINE
REF	GRAFTEXT	GRAPHICS TEXTUAL OUTPUT
REF	GRBUFLIM	GRAPHICS BUFFER LIMIT
REF	GRAFPNTR	GRAPHICS PARAMETERS DATA BLOCK PNTR
REF	GSCLRVAL	GET SCALAR VALUE
REF	IDTERMAL	IDENT CODE FOR TERMINAL TYPE ERROR
REF	LFARG	LEFT ARG PNTR
REF	LFRANK	LEFT ARG RANK
REF	LFSIZE	LEFT ARG SIZE
REF	LFTYPE	LEFT ARG TYPE
REF	MXRETURN	MONADIC FUNCTION RETURN
REF	OPBREAK	OP BREAK HANDLER LOC
REF	ORIGIN	INDEX ORIGIN (0 OR 1)
REF	OUTRANST	OUTPUT TRANSLATE TABLE
REF	QZINRET	QUAD-ZERO INPUT DRIVER RETURN
REF	QZOUTRET	QUAD-ZERO OUTPUT DRIVER RETURN
REF	RESULT	RESULT DATA BLOCK POINTER
REF	RETURN	RETURN ADDRESS CELL
REF	RTADR	RIGHT ARG ADR CELL
REF	RTARG	RIGHT ARG PNTR
REF	RTRANK	RIGHT ARG RANK
REF	RTSIZE	RIGHT ARG SIZE
REF	RTTYPE	RIGHT ARG TYPE
REF	SETCHAR	PUT CHAR IN BUFFER
REF	SETUPARG	SET UP ARG PARAMETERS
REF	STRAPOPT	GRAF INPUT STRAP OPTION 5-6-OR-7
REF	TERMTYPE	TERMINAL TYPE CODE
REF	XSEGBASE	XSEG BASE
REF	XSEGBRK	XSEG BREAK OK FLAG
REF	XSETUP	SET ADR FOR INDEXED ACCESS

ASSEMBLY PARAMETERS

```

          SYSTEM  SIG5F
PROGSECT CSECT   1
GRAF@    RES     0          START OF PROCEDURE

```

REGISTERS

*			
F	EQU	0	FLAGS REG
XI	EQU	1	GENERAL INDEX REG
K	EQU	2	SUBSCRIPT REG
T	EQU	2	TYPE
M	EQU	2	ARG DIMENS
N	EQU	3	*
CC	EQU	3	OUTPUT CHAR COUNT
XL	EQU	3	XSEG LOC
A	EQU	4	ARG ADR/INDEX
LZ	EQU	4	LINK REG
GP	EQU	4	GRAPHICS DATA BLOCK POINTER
LX	EQU	5	INDEX LINK REG
LX7	EQU	7	INDEX LINK REG
AF	EQU	6	ACCUM FOR FLOT VALUES
AF1	EQU	7	*
AI	EQU	7	ACCUM FOR INTG VALUES
BUF	EQU	7	BUFFER FOR MOVING DATA/CODE GROUPS
R	EQU	8	GENERAL WORK REG
R1	EQU	9	*
R2	EQU	10	*
R3	EQU	11	*
X	EQU	8	POINT COORD REG
Y	EQU	9	*
Z	EQU	10	*
S	EQU	11	SIZE
L3	EQU	12	LINK REG
L2	EQU	13	LINK REG
L1	EQU	14	LINK REG
LO	EQU	14	LINK FOR 'GRAFOUT'
*			
* DATA TYPE CODES			
*			
LOGL	EQU	1	LOGICAL (BIT)
CHAR	EQU	2	CHARACTER (BYTE)
INTG	EQU	3	INTEGER (WORD)
FLOT	EQU	4	FLOATING (DOUBLEWORD)
*			
* FLAG BITS			
*			
DASHFLAG	EQU	X'001'	DASHED CURVE FLAG
RENTFLAG	EQU	X'002'	RE-ENTER FLAG
3DFLAG	EQU	X'004'	3-DIMENSIONAL PLOT FLAG
EDGEFLAG	EQU	X'008'	EDGE-CROSS FLAG
ISCLFLAG	EQU	X'010'	INPUT SCALING FLAG
ASCLFLAG	EQU	X'020'	AUTO-SCALING FLAG
CENTERFG	EQU	X'040'	CENTER SCALING FLAG
VECTORFG	EQU	X'080'	VECTOR-IN-PROGRESS FLAG
OFFSETFLG	EQU	X'100'	OFFSET POINT FLAG
*			
* TERMINAL CONTROL CHARACTERS			
*			
#ESC	EQU	X'1B'	NEXT CHAR IS CONTROL
#FF	EQU	X'0C'	ERASE SCREEN
#GS	EQU	X'1D'	START VECTOR
#US	EQU	X'1F'	END VECTOR
#XHIGH	EQU	X'20'	UPPER HALF OF X DATA
#YHIGH	EQU	X'20'	UPPER HALF OF Y DATA
#XLOW	EQU	X'40'	LOWER HALF OF X DATA
#YLOW	EQU	X'60'	LOWER HALF OF Y DATA
*			
* OFFSETS INTO GRAPHICS PARAMETERS DATA BLOCK			
*			
FLAGS	EQU	3	FLAGS (DASH/RE-ENTER, SCALING)
XMIN	EQU	4	SCALE DATA: X1 UNITS
XMAX	EQU	5	X2 UNITS
YMIN	EQU	6	Y1 UNITS
YMAX	EQU	7	Y2 UNITS
ZMIN	EQU	8	Z1 UNITS
ZMAX	EQU	9	Z2 UNITS
XSCALE	EQU	10	%XPTS/(X2-X1)
YSCALE	EQU	11	%YPTS/(Y2-Y1)
ZSCALE	EQU	12	%ZPTS/(Z2-Z1)
LXWINDOW	EQU	13	WINDOW DATA: LOWER X INCHES
UXWINDOW	EQU	14	UPPER X INCHES
LYWINDOW	EQU	15	LOWER Y INCHES
UYWINDOW	EQU	16	UPPER Y INCHES
XDISPFIX	EQU	17	X'46800XXX'

```

*                                     WHERE          XXX=PPI*LX      POINTS
XRATIO      EQU          18          (UX-LX)/%XINCHS
YDISPFIX    EQU          19          X'46800YYY'
*                                     WHERE          YYY=PPI*LY      POINTS
YRATIO      EQU          20          (UY-LY)/%YINCHS
INXCOORD    EQU          21          X COORD OF QUAD-0 INPUT CHAR
INYCOORD    EQU          22          Y COORD OF QUAD-0 INPUT CHAR
INCHAR      EQU          23          INPUT CHARACTER
SPARE1      EQU          24          SPARES(FOR FUTURE PICTURE SAVE)
SPARE2      EQU          25
*
GRDBSIZE    EQU          26          GRAPHICS DATA BLOCK SIZE
PAGE
*
*
*   P R O C S
*
*
*          (THIS MODULE RESIDES IN AN OVERLAY, SO THE USUAL
*          TEMP ALLOCATION PROCEDURE WILL NOT WORK).
*
TLOC        SET          20
GRFTEMPS    EQU          XSEGBASE
*
TEMP        CNAME        1
DTEMP       CNAME        2
PROC
DO1         NAME=2
TLOC        SET          TLOC+(TLOC&1)
DISP        TLOC
LF          EQU          GRFTEMPS+TLOC
TLOC        SET          TLOC+NAME
PEND
*
*
EVEN        CNAME        0
ODD         CNAME        1
PROC
LF          EQU          %
ERROR,1,(CF(2)+NAME)&1    'REGISTER HAS WRONG PARITY'
PEND
*
*
EQUAL       CNAME
PROC
LF          EQU          %
ERROR,1,1-(CF(2)=CF(3))  'REGISTERS MUST BE EQUAL'
PEND
PAGE
*
*
*   XSEG GEN PROCS
*
*
*
GEN         OPEN          GEN
CNAME
OPEN        M,N,MN,I
PROC
LF          EQU          %
ERROR,1,1-(NUM(CF)=3)    'WRONG NUMBER OF CF ARGS'
M          SET          CF(2)
N          SET          CF(3)
MN         SET          M+N
ERROR,1,1-(NUM(AF)=(M>0)+(N>0)) 'WRONG NUMBER OF AF ARGS'
DO         M>0
I          DO          N
LW,AF(1)+M+I-1    AF(2)+I-1
FIN
I          DO          MN*(MN<3)
STW,AF(1)+I-1    I-1,XL
ELSE
LCI          MN
STM,AF(1)      0,XL
FIN
ELSE
I          DO          MN*(MN<3)
LW,BUF        AF(1)+I-1
STW,BUF        I-1,XL

```

```

        ELSE
            LCI          N
            LM,BUF       AF(1)
            STM,BUF      0,XL
        FIN
    FIN
    AI,XL      MN
    PEND
    CLOSE     M,N,MN,I
    PAGE

*
*
*   OUTPUT BYTES PROC
*
*
OUTPUT    OPEN      I
          CNAME
          PROC
LF        CW,CC      GRBUFLIM
          BLE        %+2
          BAL,L0     GRAFOUT
I         DO         NUM(AF)
          DO         AFA(I)
          STB,AF(I)  GRAFBUF,CC
          ELSE
          LI,AI      AF(I)
          STB,AI     GRAFBUF,CC
          FIN
          AI,CC      1
          FIN
          PEND
          CLOSE     I
          PAGE

*
*
*   SET/RESET BREAK FLAG PROC
*
SETBRK   CNAME      OPBREAK
RESETBRK CNAME      0
          PROC
LF        LI,L2      NAME
          STW,L2     XSEGBRK
          PEND
          PAGE

*
*
*   F.S. ARITHMETIC PROC
*
*
FSADD    FNAME      1          F.S. ADD      FUNCTION
FSSUB    FNAME      2          F.S. SUBTRACT FUNCTION
FSMUL    FNAME      3          F.S. MULTIPLY FUNCTION
FSDIV    FNAME      4          F.S. DIVIDE   FUNCTION
          PROC
          LOCAL      X,E,I,S,Z,ZE,ZS,AD,ML,DV,SB,NORM
X         SET        0,0      ARG FRACTIONS
E         SET        0,0      ARG EXPONENTS
S         SET        +1,+1    ARG SIGNS
I         DO         2        UNPACK EACH ARG:
X(I)      SET        AF(I)
          DO         X(I)<0
X(I)      SET        -X(I)
S(I)      SET        -1          SIGN (+1 OR -1)
          FIN
E(I)      SET        (X(I)**-24)-X'40'    EXPONENT (UNBIASED),
X(I)      SET        X(I)&X'00FFFFFF'    POSITIVE FRACTION (X'00XXXXXX').
          FIN

*
          GOTO,NAME  AD,SB,ML,DV      DO SPECIFIED OP
SB         BOUND     1          SUBTRACT:
S(2)      SET        -S(2)        REVERSE 2ND ARG SIGN, DO ADD
*
AD         BOUND     1          ADD:
I         SET        (E(1)-E(2))*4    ALIGN FRACTIONS,
          DO         I<0          SET RESULT EXPONENT
X(1)      SET        X(1)**I
ZE        SET        E(2)
          ELSE

```

```

X(2)    SET      X(2)**(-I)
ZE      SET      E(1)
      FIN
ZS      SET      S(1)          SET RESULT SIGN
      DO      S(1)=S(2)
Z       SET      X(1)+X(2)      SIGNS AGREE: ADD FRACTIONS
      ELSE
Z       SET      X(1)-X(2)      SIGNS DIFFER: SUBTRACT FRACS
      DO      Z<0              AND REVERSE IF NECESSARY.
Z       SET      -Z
ZS      SET      -ZS
      FIN
      FIN
      GOTO     NORM            GO NORMALIZE
*
ML      BOUND     1            MULTIPLY:
Z       SET      (X(1)*X(2))**-20  MULTIPLY FRACTIONS
ZE      SET      E(1)+E(2)-1      ADD EXPONENTS
ZS      SET      S(1)*S(2)        MULTIPLY SIGNS
      GOTO     NORM            NORMALIZE
*
DV      BOUND     1            DIVIDE:
Z       SET      (X(1)**24)/X(2)  DIVIDE FRACTIONS
ZE      SET      E(1)-E(2)        SUBTRACT EXPONENTS
ZS      SET      S(1)*S(2)        MULTIPLY SIGNS
*
NORM    BOUND     1            NORMALIZE:
      DO      Z>0              IF Z NONZERO,
      WHILE   Z<X'00100000'
Z       SET      Z**4            SHIFT UP
ZE      SET      ZE-1
      FIN
      WHILE   Z>=X'01000000'
Z       SET      Z**-4            OR DOWN,
ZE      SET      ZE+1
      FIN
Z       SET      (Z+(ZE+X'40')**24)*ZS  AND PACK IN F.S. FORM.
      FIN
      PEND     Z              RETURN RESULT
      PAGE
*
*
*   D E V I C E   D E P E N D E N T   D A T A
*
*
*   F.S. PARAMETERS
*
%XINCHS EQU      +FS'8.25'      SCREEN SIZE, IN INCHES
%YINCHS EQU      +FS'6.29'      *
%ASPECT  EQU      FSDIV(%YINCHS,%XINCHS)  ASPECT RATIO
%XPTS    EQU      +FS'1023.999'  SCREEN SIZE, POINTS
%YPTS    EQU      FSMUL(%XPTS,%ASPECT)    *
%ZPTS    EQU      %XPTS          *
%PPI     EQU      FSDIV(%XPTS,%XINCHS)    POINTS PER INCH
%EPS     EQU      +FS'.001'
%NTICS   EQU      +FS'5'        NUMBER OF TIC MARKS ON LONG SIDE
      PAGE
*
*
*   PLANAR PROJECTION PARAMETERS
*
*
*   CONVERSION OF A 3-D POINT (X,Y,Z) TO A 2-D POINT (X',Y')
*   IS DONE BY A LINEAR TRANSFORMATION COMPOSED OF (1) AN
*   ORTHOGONAL PROJECTION ONTO THE X'Y' PLANE, AND (2) RE-
*   SCALING TO MAKE THE ENTIRE PROJECTED BOX (0<=X<=XP,
*   0<=Y<=YP, 0<=Z<=ZP) FIT INTO THE RECTANGLE (0<=X'<=XP,
*   0<=Y'<=YP). (NOTE: XP/YP/ZP ARE ABBREVIATIONS FOR
*   %XPNTS, ETC.) THE TRANSFORMATION DEPENDS ON TWO ANGULAR
*   PARAMETERS: ALPHA AND BETA, THE AZIMUTH (TO RIGHT OF
*   Z-AXIS) AND ELEVATION (ABOVE XZ PLANE) OF THE 'VANTAGE
*   POINT'. THESE ANGLES ARE INTRODUCED IN TERMS OF THEIR
*   SINES AND COSINES:
*
%SA      EQU      +FS'.34202'    = SIN 20 DEGREES
%CA      EQU      +FS'.93969'    = COS 20 DEGREES
%SB      EQU      +FS'.70711'    = SIN 45 DEGREES

```

```

%CB      EQU      +FS'.70711'      = COS 45 DEGREES
*
*      THE TRANSFORM IS
*
*      
$$\begin{aligned} X' &= (X*CA + (ZP-Z)*SA) *XP / \\ &\quad (XP*CA + ZP*SA) \\ Y' &= (Y*CB + (XP-X)*SA*SB + (ZP-Z)*CA*SB) *YP / \\ &\quad (YP*CB + XP*SA*SB + ZP*CA*SB) \end{aligned}$$

*
*      WHICH WE MESSAGE INTO THE FORM
*
*      
$$\begin{aligned} X' &= P10 + P11*X + P13*Z \\ Y' &= P20 + P21*X + P22*Y + P23*Z \end{aligned}$$

*
*      BY MEANS OF THE FOLLOWING:
*
DX      OPEN      DX,DY,DZ,SASB,CASB,F,DXZ
SET      FSMUL(%XPTS,%CA) = XP*CA
DZ      SET      FSMUL(%ZPTS,%SA) = ZP*SA
F      SET      FSDIV(%XPTS,; = XP/(XP*CA + ZP*SA)
FSADD(DX,DZ))
%P10     EQU      FSMUL(F,DZ) = ZP*SA*F = CONSTANT IN X'
%P11     EQU      FSMUL(F,%CA) = CA*F = COEFF OF X IN X'
%P13     EQU      FSMUL(F,-%SA) = -SA*F = COEFF OF Z IN X'
*
SASB     SET      FSMUL(%SA,%SB) = SA*SB
CASB     SET      FSMUL(%CA,%SB) = CA*SB
DX      SET      FSMUL(%XPTS,SASB) = XP*SA*SB
DY      SET      FSMUL(%YPTS,%CB) = YP*CB
DZ      SET      FSMUL(%ZPTS,CASB) = ZP*CA*SB
DXZ      SET      FSADD(DX,DZ) = XP*SA*SB + ZP*CA*SB
F      SET      FSDIV(%YPTS,; = YP/(YP*CB + XP*SA*SB + ZP*CA*SB)
FSADD(DXZ,DY))
%P20     EQU      FSMUL(F,DXZ) = (XP*SA*SB+ZP*CA*SB)*F= CONST IN Y'
%P21     EQU      FSMUL(F,-SASB) = -SA*SB*F = COEFF OF X IN Y'
%P22     EQU      FSMUL(F,%CB) = CB*F = COEFF OF Y IN Y'
%P23     EQU      FSMUL(F,-CASB) = -CA*SB*F = COEFF OF Z IN Y'
CLOSE    DX,DY,DZ,DXZ,F,SASB,CASB
PAGE
*
*
*      Q U A D   Z E R O   I N P U T / O U T P U T
*
*
*      USECT      PROGSECT
*
*
*      QUAD-ZERO OUTPUT
*
*      GIVEN ARG DATA BLOCK POINTER IN RTARG, AND GRAPHICS
*      PARAMETER DATA BLOCK POINTER IN GRAFPNTR. THE LATTER
*      MAY BE ZERO IF THIS IS THE FIRST ENTRY INTO THE GRAPHICS
*      PACKAGE, IN WHICH CASE WE ALLOCATE THE BLOCK (PUTTING
*      ITS POINTER IN GRAFPNTR) AND INITIALIZE SCALING AND
*      WINDOW DATA THEREIN. UPON EXIT (TO QZOUTRET), BOTH
*      POINTERS ARE STILL IN EXISTENCE.
*
QZOUTPUT EQU      %
LI,R      QZOUTRET      SET RETURN ADDRESS
STW,R     RETURN
BAL,LX    INITGRAF      SET UP GRAFPNTR
BAL,LX    GETCOORD      REMEMBER CURRENT CURSOR LOC
LI,A      1
BAL,LX    SETUPARG      SET UP RT ARG PARAMS
LW,T      RTTYPE
CI,T      CHAR          DON'T ALLOW CHAR ARG
BE        ERDOMAIN
LW,XI     RTARG          GET DB POINTER
LW,R      RTRANK         CHECK ARG RANK
BNEZ      4Z1
BAL,LZ    GSCLRVAL,T     SCALAR: GET ITS VALUE IN INTG
BEZ       *RETURN        IF =0, DO NOTHING
LI,CC     0              INIT OUTPUT
OUTPUT    *AI            OUTPUT ARG VALUE
B         NEXTLINE       FINISH UP, EXIT
4Z1       LI,A          1
CI,R      2              CHECK ARG RANK

```


	BG	ERRANK	>2	ERROR
	BE	4Z2	=2	MATRIX
	LW,N	2,XI	=1	VECTOR: GET ITS DIMEN (N),
	BEZ	*RETURN		(IGNORE EMPTY ARG)
	LI,M	1		PRETEND IT'S A 1*N MATRIX.
	B	4Z3		
4Z2	EQU	%		MATRIX:
	LW,M	2,XI		GET 1ST DIMEN (M)
	BEZ	*RETURN		(IGNORE EMPTY ARG)
	LW,N	3,XI		GET 2ND DIMEN (N)
	BEZ	*RETURN		(IGNORE EMPTY ARG)
	EQUAL,N,M+1			
	CLR,M	=3		COMPARE BOTH DIMENS WITH 3
	BLE	4Z3		M<=3: COLUMN-MAJOR
*	BCS,8	ERLENGTH		M>3 AND N>3: ERROR
				M>3 AND N<=3: ROW-MAJOR
	STW,N	INCELMT		ROW-MAJOR: ELEMENT INCREMENT = N,
	STW,A	INCCOORD		COORDINATE INCREMENT = 1,
	XW,M	N		SWAP M/N TO MAKE M<=3.
	B	4Z4		
4Z3	STW,A	INCELMT		COL-MAJOR: ELEMENT INCREMENT = 1,
	STW,N	INCCOORD		COORDINATE INCREMENT = N.
4Z4	STW,M	DIMEN		M = DATA DIMENSION = 1, 2, OR 3.
	ODD,N			
	MW,N	INCELMT		N(SIZE)*(ELMT INCR) = TERMINATION
	STW,N	STOPVAL		...VALUE
*				
*	GENERATE XSEG			
*				
	LI,A	1		SET UP RTADR FOR INDEXED LOAD
	BAL,LX	XSETUP		INIT XSEG LOC
	LI,XL	XSEGBASE		GEN INIT CODE
	GEN,0,4	CODE1		GET DATA TYPE
	LW,T	RTTYPE		
	LW,R3	AIK0INST		SET UP 'AI,K INCC' INST
	AW,R3	INCCOORD		SET UP 'LW,X AF' INST
	LW,R2	CODE2		GET DIMEN COUNT (1,2,3)
	LW,XI	DIMEN		FOR EACH DIMEN GEN:
	LD,R	LOADFSTB,T		LOAD ARG(K)
	AW,R	RTADR		CONVFS
4Z5	GEN,4,0	R		LW,XYZ AF
	AW,R2	=X'00100000'		AI,K INCC
	BDR,XI	4Z5		DISCARD LAST 'AI,K INCC'
	AI,XL	-1		GET -DIMEN
	LCW,AI	DIMEN		GET 1-DIMEN
	BIR,AI	4Z6		DIMEN=1: CHANGE 'LW,X' TO 'LW,Y'
	STW,R2	-1,XL		AND GEN CODE TO COMPUTE X DATA.
	GEN,0,3	CODE3		(GEN CLOBBERED AI=0).
	LI,AI	0		GEN: AI,K INCE-(D-1)*INCC
4Z6	MW,AI	INCCOORD		BAL,LX SENDPNT
	AW,AI	INCELMT		
	AND,AI	=X'000FFFFF'		AND LOOP CONTROL CODE.
	AW,AI	AIK0INST		
	GEN,1,4	AI, CODE4		
*				
*	EXECUTE XSEG			
*				
	LW,GP	GRAFPNTR		SET UP GRAPHIC PARAMS POINTER
	LW,F	DIMEN		COMPUTE 3-DIMENSION FLAG
	AI,F	1		(3DFLAG IS THE 4-BIT)
	EQUAL,3DFLAG,4			
	AND,F	=3DFLAG		FETCH OTHER FLAGS
	OR,F	FLAGS,GP		IF AUTO-SCALING FLAG SET,
	CI,F	ASCLFLAG		DO AUTOMATIC SCALING ON GIVEN DATA
	BAZ	4Z7		
	BAL,LX	AUTOSCAL		
4Z7	LI,CC	0		INIT OUTPUT CHAR COUNT
	SETBRK			ALLOW BREAKS DURING XSEG EXECUTION
	BAL,L2	XSEGBASE		EXECUTE XSEG
	RESETBRK			STOP BREAKS AFTERWARDS
QUIT	BAL,LX	ENDVEC		DONE: TERMINATE CURRENT VECTOR
	LI,F	0		
	LI,F+1	DASHFLAG		RESET 'DASH' FLAG: ONLY ONE CURVE
	STS,F	FLAGS,GP		IS TO BE DASHED.
*				
*	RESTORE ORIGINAL CURSOR POSITION, EXIT			
*				
NEXTLINE	EQU	%		RESTORE CURSOR LOC READ BY GETCOORD

```

        LW,R1      GRXYBYTES      GET INPUT BYTES 2-5 (COORD BYTES)
        SCS,R1     16              PUT Y-COORD TO LEFT OF X-COORD
        AND,R1     =X'1F1F1F1F'    GET RID OF CODE BITS SENT BY DEVICE
        OR,R1      YHYLXHL        INSTALL CORRECT COORD CODE BITS
*                                     WE NOW HAVE THE 4-BYTE COORD IN R1
        OUTPUT     #GS            SEND 'START VECTOR' COMMAND
        ODD,R1
        EQUAL,R+1,R1
11Z1    SLD,R      8              SEND THE 4 BYTES (ONE-POINT VECTOR)
        OUTPUT     *R
        AI,R1      0
        BNEZ       11Z1
        OUTPUT     #US            SEND 'END VECTOR' COMMAND
        BAL,L0     GRAFOUT        DUMP OUTPUT BUFFER
        B          *RETURN        EXIT
*
YHYLXHL DATA,1  #YHIGH,#YLOW,#XHIGH,#XLOW
*
*      LOCAL DATA/TEMPS
*
INCELMT TEMP      INCREMENT TO NEXT ELEMENT
INCCOORD TEMP     INCREMENT TO NEXT COORDINATE
DIMEN    TEMP     DATA DIMENSIONS (1, 2, OR 3)
INDEXVAL TEMP     GENERATED X-VALUE (FOR DIMEN=1)
STOPVAL  TEMP     TERMINATING INDEX-VALUE
*
*
AIK0INST AI,K      0
        BOUND      8
LOADFSTB EQU       %-2*LOGL      LOAD/CONVERT INST TABLE
        BAL,LX     0             LOGL LOAD (0 OR -1)
        LW,AF      FS01,AI       LOGL CONVERT TO FS
        RES        2             CHAR (NOT ALLOWED)
        LW,AI      0             INTG LOAD
        BAL,LX     IT0FS         INTG CONVERT TO FS
        LD,AF      0             FLOT LOAD
        BAL,LX     FLT0FS        FLOT CONVERT TO FS
        BAL,LX     0             ISEQ LOAD
        BAL,LX     IT0FS         ISEQ CONVERT TO FS
*
FS01    EQU        %+1           TABLE TO CONVERT LOGL VALUE (-1/0)
        DATA      FS'1.0',0    TO F.S. VALUE (1.0/0.0)
*
*
*      XSEG CODE:
*
CODE1    LI,K       0             0 INIT ARG INDEX
        LCW,AI      ORIGIN        1 INIT SUPPLIED-X VALUE
        LW,AI       FS01,AI       2 TO INDEX ORIGIN (0 OR 1)
        STW,AI      INDEXVAL     3 IN CASE DIMEN=1.
*
*      - IF DIMEN=2/3 -
*
CODE2    LOAD       ARG(K)         GET X VALUE
        CONVERT    TO F.S.        CONVERT
        LW,X       AF             PUT IN X
        AI,K       INCC           BUMP INDEX TO NEXT (Y) COORD
*
*      - ANY DIMEN
*
        LOAD       ARG(K)         GET Y VALUE
        CONVERT    TO F.S.        CONVERT
        LW,Y       AF             PUT IN Y
*
*      - IF DIMEN=3 -
*
        AI,K       INCC           BUMP INDEX TO NEXT (Z) COORD
        LOAD       ARG(K)         GET Z VALUE
        CONVERT    TO F.S.        CONVERT
        LW,Z       AF             PUT IN Z
*
*      - IF DIMEN=1 -
*
CODE3    LW,X       INDEXVAL       1-DIMENSIONAL DATA: WE SUPPLY
        FAS,X      =FS'1.0'      X-VALUES=ORIGIN+ 0.0, 1.0, 2.0,...
        XW,X       INDEXVAL
*

```

```

*           - ALL CASES -
*
* CODE4      AI,K      INCE-(D-1)*INCC  BACK TO X-COORD, BUMP TO NEXT ELMT
*            BAL,LX    SENDPNT          PROCESS THE POINT (X,Y,Z)
*            CW,K      STOPVAL
*            BL        XSEGBASE+4        LOOP UNTIL ALL POINTS PROCESSED
*            B          *L2              DONE: EXIT FROM XSEG
*            PAGE
*
*
* QUAD-ZERO INPUT
*
*           GIVEN GRAPHICS PARAMETER DATA BLOCK POINTER IN GRAFPNTR
*           (WHICH IS ZERO AT FIRST ENTRY; SEE 'QZOUTPUT'). UPON
*           EXIT (TO QZINRET), GRAFPNTR EXISTS, AND THE RESULT DATA
*           BLOCK POINTER IS IN 'RESULT'. THE RESULT IS A
*           CHARACTER SCALAR (=THE INPUT CHARACTER). THE
*           X AND Y COORDINATES OF THE CROSSHAIR CURSOR ARE
*           SAVED; THEY CAN BE RECOVERED VIA '11 #GRF 3'.
*           THESE COORDINATES ARE EITHER
*           UNSCALED IN 'POINTS' (IF ISCLFLAG=0), OR SCALED ACCORDING
*           TO THE LATEST X/Y SCALING PARAMETERS (ISCLFLAG=1). IN
*           THE LATTER CASE, CHARACTERS FALLING OUTSIDE THE 'WINDOW'
*           ARE IGNORED.
*
* QZINPUT    EQU      %
*            LI,A      QZINRET
*            STW,A      RETURN          SET UP RETURN ADDRESS
*            BAL,LX    INITGRAF        SET UP GRAFPNTR, COPY TO GP
*            BAL,LX    GETCOORD        GET CURSOR LOC
*            LW,X      GRXYBYTS        SAVE CURSOR COORDS
*            STW,X      XYSAVE
*
*
* 12Z1       SETBRK    ALLOW BREAKS DURING INPUT WAIT
*            LW,L2     BREAKFLG        RESPOND IMMEDIATELY TO
*            BNEZ      OPBREAK          EXISTING BREAKS.
*            BAL,LX    GETCHAR          DISPLAY CROSSHAIRS, GET CHAR & COORDS
*            RESETBRK  STOP BREAKS AFTERWARDS
*            LW,GP      GRAFPNTR        RESTORE GP (BECAUSE OF RE-START)
*            LW,XI      GRAFPNTR        EXTRA COPY TO BE BUMPED BY 1
*            LI,K       X'7F'
*            AND,K      GRINCHAR        GET 7-BIT ASCII APL CHAR.
*            LB,R       INPTRANS,K      TRANSLATE TO INTERNAL CODE
*            STW,R      INCHAR,GP       STORE IN GRAPHICS DATA BLOCK
*            LI,F       ISCLFLAG        SET F SO THAT 'BDR,F' WILL BRANCH
*            AND,F      FLAGS,GP        IF ISCLFLAG=1.
*            LI,K       -2              PROCESS BOTH COORDS:
* 12Z2       LW,R1      GRXYBYTS        PACK A COORD:
*            SLS,R1     3                DISCARD HIGH-BYTE CONTROL BITS
*            LI,R       0
*            SLD,R      5                GET UPPER 5 BITS
*            SLS,R1     3                DISCARD LOW-BYTE CONTROL BITS
*            SLD,R      5                APPEND LOWER 5-BITS
*            STW,R1     GRXYBYTS        SAVE WHAT'S LEFT FOR NEXT COORD
*            OR,R       =X'46000000'    CONVERT TO F.S. VALUE IN R
*            SFS,R      6                IN 'POINTS'.
*            BDR,F      12Z3            TEST ISCLFLAG
*            B          12Z4            =0: STORE AS IS
* 12Z3       EQU      %                  =1: DO SCALING & TESTING
*            FDS,R      =%PPI           CHANGE UNIT TO INCHES FOR
*            CW,R       UXWINDOW,GP     WINDOWING TEST.
*            BG         12Z1            MUST BE <= UPPER LIMIT (INCHES)
*            FSS,R      LXWINDOW,GP     AND >= LOWER LIMIT; IF NOT
*            BLZ        12Z1            IN WINDOW, REJECT CHAR.
*            FDS,R      XRATIO,GP       RESCALE TO FULL-SCREEN BOUNDS
*            FMS,R      =%PPI           CONVERT BACK TO 'POINTS'
*            FDS,R      XSCALE,XI       RESCALE ACCORDING TO SCALE
*            FAS,R      XMIN,GP         PARAMS: PUTS IT IN (MIN,MAX).
* 12Z4       STW,R      INXCOORD,XI     STORE PROCESSED X/Y COORD
*            AI,GP      2                NEXT TIME USE 'Y' PARAMS
*            AI,XI      1
*            BIR,K      12Z2            PROCESS NEXT (Y) COORD
*
*
*            LI,S      6                ALLOCATE BLOCK FOR COORDINATES
*            BAL,LX7    ALOCHNW
*            STW,A      RESULT          SAVE POINTER
*            LI,R       FLOT**8+1      FLOATING POINT VECTOR
*            STH,R      *RESULT

```

```

        LI,R      2
        STW,R     2,A      SET LENGTH=2
        LI,R      0
        STW,R     5,A      SET LOW ORDER VALUE FOR FLOATING
        STW,R     7,A      LONG COORDINATES(COMPUTED FS)
        LW,XI     GRAFPNTR
        LW,R      INXCOORD,XI
        STW,R     4,A      SET HIGH ORDER VALUES IN RESULT
        LW,R      INYCOORD,XI
        STW,R     6,A
*
CNXTLINE EQU      %
        LW,X      XYSAVE
        STW,X     GRXYBYTES
        LI,CC     0        IINIT OUTPUT CHAR COUNT
        B         NEXTLINE RESTORE CURSOR AND EXIT
*
*
GRINCHAR TEMP     GRAPHICS TERM INPUT CHAR
GRXYBYTES TEMP    GRAPHICS TERM X/Y COORD BYTES
JUNKTEMP  TEMP    PLUS JUNK BYTES
XYSAVE    TEMP    SAVE CELL FOR CURSOR COORDS
*
*
INPTRANS EQU      %      QUAD-ZERO INPUT CHAR TRANSLATION TABLE
II        DO      32
        DATA,1   II-1
        FIN
        DATA     X'40715D4C'
        DATA     X'747E6EB5'
        DATA     X'79707BFB'
        DATA     X'6B4E4B61'
        DATA     '0123'
        DATA     '4567'
        DATA,2   '89',X'4DB4'
        DATA     X'5EFA7AB1'
        DATA     X'727C4250'
        DATA     X'44456D7F'
        DATA     X'48495A7D'
        DATA     X'534F5556'
        DATA     X'5C6F6C62'
        DATA     X'5F645B66'
        DATA     X'676A4AFD'
        DATA     X'EBFC7660'
        DATA,1   X'EA','A','B','C'
        TEXT      'DEFGHIJKLMNOPQRSTUVWXYZ'
        DATA,1   'X','Y','Z',X'ED'
        DATA     X'ECEE43FF'
        PAGE
*
*
*   G R A P H I C S   S T A T E   C O N T R O L
*
*   DELTA-GRF FUNCTION
*
*       GIVEN ARG DATA BLOCK POINTERS IN LFARG & RTARG, AND
*       GRAPHICS PARAMETERS DATA BLOCK POINTER IN 'GRAFPNTR'
*       (WHICH IS ZERO UPON FIRST ENTRY: SEE 'QZOUTPUT').
*       UPON EXIT (TO DXRETURN) THESE POINTERS STILL EXIST,
*       AND A POINTER TO THE RESULT (WHICH IS USUALLY NULL) IS
*       IN 'RESULT'.
*
*
*       THE FOLLOWING IS A LIST OF ALL THE FUNCTIONS PERFORMED
*       BY DELTAGRF (THE 'DELTA' HAS BEEN DROPPED FOR BREVITY):
*
*       0 GRF 1      TURN OFF INPUT SCALING
*       0 GRF 2      TURN ON  INPUT SCALING
*       1 GRF 3      SET TERMINAL DEVICE TYPE
*       2 GRF NUL    SET DEFAULT WINDOW
*       2 GRF LX,UX,LY,UY
*                       SET UP GIVEN WINDOW LIMITS
*       3 GRF NUL    SET 'TRANSPARENT' SCALING
*       3 GRF X1,X2,Y1,Y2,(Z1,Z2)
*                       SET UP GIVEN SCALING
*       4 GRF NUL    RETURN CURSOR TO NEXT LINE
*       4 GRF X,Y,(Z) SET CURSOR TO GIVEN POINT

```

*		5	GRF	NUL	DRAW AXES
*		6	GRF	1	SET AUTO-SCALING MODE
*		6	GRF	2	SET CENTERED AUTO-SCALING MODE
*		6	GRF	3	KILL AUTO-SCALING (FIX CURRENT SCALE)
*		7	GRF	NUL	DASH NEXT CURVE
*		8	GRF	NUL	ERASE & HOME SCREEN
*		8	GRF	1	ERASE SCREEN
*		8	GRF	2	HOME SCREEN
*		9	GRF	NUL	DRAW BOX
*		10	GRF	1	TURN OFF RE-ENTRANCE
*		10	GRF	2	TURN ON RE-ENTRANCE
*		11	GRF	1	RETURN SCALING PARAMS AS RESULT
*		11	GRF	2	RETURN WINDOW PARAMS AS RESULT
*		11	GRF	3	RETURN INPUT COORDS AS RESULT
*		11	GRF	4	RETURN INPUT CHARACTER AS RESULT
*		11	GRF	5	SET FOR 5-CHAR INPUT STRAP OPTION.
*		11	GRF	6	6
*		11	GRF	7	7
*					
*					
*					
DELTAGR	EQU	%			
	LI,A	DXNULRTN			
	STW,A	RETURN		SET UP RETURN ADDRESS	
	BAL,LX	INITGRAF		SET UP GRAFPNTR	
	LI,A	0			
	BAL,LX	SETUPARG		SET UP LEFT ARG PARAMS	
	LI,A	1			
	BAL,LX	SETUPARG		SET UP RIGHT ARG PARAMS	
	LI,S	1			
	CW,S	LFSIZE		MAKE SURE LEFT ARG IS 1 ELEMENT	
	BNE	ERLENGTH			
	LW,XI	LFARG		GET LEFT ARG PNTR	
	AW,XI	LFRANK		SKIP OVER DIMENS	
	LW,T	LFTYPE		GET ITS TYPE	
	BAL,LZ	GSCLRVAL,T		GET ITS VALUE (= FUNCTION CODE)	
	BLZ	ERDOMAIN		MUST BE >=0	
	CI,AI	11		...AND <=11	
	BG	ERDOMAIN			
	STW,AI	FUNCCODE		SAVE IT FOR LATER	
*					
	LW,S	RTSIZE		NOW CHECK RIGHT ARG SIZE	
	BEZ	FUNC0		IF SIZE=0, IT'S A NUL-FUNC	
	BDR,S	FUNCN		IF SIZE>1, IT'S A MULTI-PARAM FUNC	
FUNC1	LW,XI	RTARG		SIZE=1: IT'S A 1-PARAM FUNC	
	AW,XI	RTRANK		POINT TO RT ARG PAST DIMENS	
	LW,T	RTTYPE		GET TYPE	
	BAL,LZ	GSCLRVAL,T		GET ITS (SINGLE) INTEGER VALUE	
	LW,K	AI		PUT IT IN K	
	BLEZ	ERDOMAIN		MUST BE >0 ...	
	LW,XI	FUNCCODE			
	CB,K	LIMTBL1,XI		AND <= LIMIT FOR FUNCTION;	
	BG	ERDOMAIN			
	LW,GP	GRAFPNTR		SET UP GP AND ENTER SELECTED	
	BAL,LX	JUMPTBL1,XI		1-PARAM ROUTINE: PARAM IN K.	
	B	*RETURN		EXIT	
*					
FUNC0	LW,GP	GRAFPNTR		SIZE=0: GO TO SELECTED 0-PARAM	
	BAL,LX	JUMPTBL0,AI		ROUTINE.	
	B	*RETURN		EXIT	
*					
FUNCN	CLM,AI	2T04		SIZE>1: ALLOWED ONLY FOR	
	BCS,9	ERLENGTH		FUNCTIONS 2, 3, OR 4.	
	LI,F	0		(FOR SET/SCALE: ASSUME 2D)	
	CB,S	LIMTBLN,AI		SIZE MUST MATCH SIZE REQUIREMENTS	
	BE	7Z4		OF SELECTED FUNCTION.	
	CB,S	LIMTBLNN,AI			
	BNE	ERLENGTH			
	LI,F	3DFLAG		(FOR SET/SCALE WITH 3/6 PARAMS: SET 3D)	
7Z4	LI,A	1			
	BAL,LX	XSETUP		SET UP RTADR	
	LI,XL	XSEGBASE		BEGIN XSEG GEN	
	LW,T	RTTYPE			
	LD,R	LOADFSTB,T		GEN XSEG TO FETCH ARG VALUES,	
	AW,R	RTADR		CONVERT 'EM TO F.S., AND STORE	
	GEN,2,4	R,CODE6		'EM IN 'PARAMS' BUFFER.	
	LI,R	0		IN CASE OF 2-DIMEN SCALE-SETTING,	
	STW,R	PARAMS+4		SET Z-SCALE PARAMS TO 0.	

	STW, R	PARAMS+5	
	LI, K	0	START WITH 1ST PARAM
	LW, N	RTSIZE	GET SIZE
	BAL, L2	XSEGBASE	GET PARAMS TO BUFFER
	LW, GP	GRAFPNTR	SET UP GP
	LW, AI	FUNCCODE	
	BAL, LX	JUMPTBLN, AI	ENTER N-PARAM FUNC
	B	*RETURN	EXIT
*			
*			
DXNULRTN	EQU	%	SET NUL RESULT, EXIT TO DXRETURN
	LI, S	1	ALLOCATE RESULT DATA BLOCK:
	BAL, LX7	ALOCHNW	RESULT = NULL
	STW, A	RESULT	STORE DATA BLOCK POINTER
	LI, R	INTG**8+1	
	STH, R	*RESULT	SET TYPE=INTG, RANK=1
	LI, R	0	
	STW, R	2, A	SET DIMEN=0
	B	DXRETURN	EXIT
*			
*			
*			
	LOCAL DATA/TEMPS		
FUNCCODE	TEMP		HOLDS LEFT ARG VAL (FUNCTION CODE)
PARAMS	DTEMP		STORAGE FOR UP TO SIX PARAMETERS
	DTEMP		
	DTEMP		
*			
*			
LIMTBL1	DATA, 1	2, 3, 0, 0, 0, 0, ; 3, 0, 2, 0, 2, 7	PARAM LIMITS FOR 1-PARAM FUNCTIONS
	BOUND	4	
LIMTBLN	DATA, 1	0, 0, 3, 3, 1	SIZE LIMITS (-1) FOR N-PARAM FUNCS
	BOUND	4	
LIMTBLNN	DATA, 1	0, 0, 3, 5, 2	ALTERNATE SIZES THEREFOR (3D)
	BOUND	8	
2T04	DATA	2, 4	N-PARAM FUNC CODE RANGE
*			
*			
*			
*			
	JUMP TABLES		
JUMPTBL0	EQU	%	0-PARAMETER FUNCTIONS:
	B	ERLENGTH	
	B	ERLENGTH	
	B	NOWINDOW	2 DEFAULT WINDOW
	B	NOSCALE	3 TRANSPARENT SCALING
	B	CNXTLINE	4 RETURN CURSOR
	B	DRAWAXES	5 DRAW AXES
	B	ERLENGTH	
	B	DASHMODE	7 DASH NEXT CURVE
	B	ERASHOME	8 ERASE/HOME SCREEN
	B	DRAWBOX	9 DRAW BOX
	B	ERLENGTH	
	B	ERLENGTH	
*			
JUMPTBL1	EQU	%	1-PARAMETER FUNCTIONS:
	B	ISCLMODE	0: K SET QUAD-0 INPUT MODE
	B	SETDEV	1: K SET DEVICE TYPE
	B	ERLENGTH	
	B	ERLENGTH	
	B	ERLENGTH	
	B	ERLENGTH	
	B	SCALMODE	6: K SET SCALING MODE
	B	ERLENGTH	
	B	SCREENCT	8: K SCREEN CONTROL
	B	ERLENGTH	
	B	RENTMODE	10: K SET RE-ENTRANCE MODE
	B	PARAMVAL	11: K RETURN SCALE/WINDOW PARAMS
*			
JUMPTBLN	EQU	%-2	N-PARAMETER FUNCTIONS:
	B	SETWINDO	2: LX, UX, LY, UY SET WINDOW
	B	SETSCALE	3: X1, X2, Y1, Y2, (Z1, Z2) SET SCALING
	B	SETCOORD	4: X, Y, (Z) SET COORDS
*			
*			
*			
*			
*			
	XSEG CODE:		
*			
*			
*			
*			
	LOAD	RTARG(K)	0 GET PARAM

```

*          CONVERT TO F.S.      1  CONVERT TO F.S.
CODE6     STW,AF  PARAMS,K      STORE IN BUFFER
          AI,K    1             BUMP INDEX
          BDR,N   XSEGBASE      LOOP
          B       *L2           EXIT
          PAGE

```

```

*
*
*  P O I N T   P L O T T I N G   R O U T I N E S
*
*

```

```

*  SEND POINT
*

```

```

*          THIS ROUTINE, CALLED ONCE FOR EACH 2- OR 3-DIMENSIONAL
*          DATA POINT, PLOTS THE CURVE JOINING THE POINTS.  IT
*          PERFORMS THE FOLLOWING FUNCTIONS:
*

```

- * 1. SCALING
- * 2. PLANAR PROJECTION OF 3-DIMENSIONAL DATA
- * 3. REDUCTION TO SPECIFIED WINDOW
- * 4. PRODUCTION OF DASHED CURVE, IF 'DASHFLAG' SET
- * 5. RANGE CHECKING WITH INTERPOLATION TO EDGE OF WINDOW
- * FOR OUT-OF-RANGE POINTS; SUCH CURVES ARE ALLOWED
- * TO RE-ENTER THE WINDOW IF 'RENTFLAG' IS SET.
- * 6. ADDITION OF OFFSET (IN REG 'S') TO FINAL X AND
- * Y COORDS IF OFSETFLG=1 (ONLY USED BY 'DRAWAXES').

```

*          SENDPNT IS CALLED WITH GRAFPNTR IN GP, FLAGS IN F,
*          AND THE POINT COORDINATES IN X/Y (OR X/Y/Z). LINK IS LX.
*          'EDGEFLAG' AND 'VECTORFG' MUST NOT BE ALTERED BY THE
*          CALLER BETWEEN CALLS; ON THE INITIAL CALL THEY MUST
*          BE IN THE RESET STATE.
*

```

```

* SENDPNT EQU      %
*

```

```

*  SCALING
*

```

```

*          FSS,X    XMIN,GP      CONVERT X TO RASTER POINTS
*          FMS,X    XSCALE,GP
*          FSS,Y    YMIN,GP      CONVERT Y TO RASTER POINTS
*          FMS,Y    YSCALE,GP
*

```

```

*  PROJECTION
*

```

```

*          CI,F     3DFLAG      DO THIS STUFF ONLY FOR 3-D DATA
*          BAZ      1Z1
*          FSS,Z    ZMIN,GP      FINISH SCALING: CONVERT Z TO POINTS
*          FMS,Z    ZSCALE,GP
*          FMS,X    =%P21        P21*X
*          FMS,Y    =%P22        P22*Y
*          FMS,Z    =%P23        P23*Z
*          FAS,Y    X            NEW Y = P20 +P21*X +P22*Y +P23*Z
*          FAS,Y    Z
*          FAS,Y    =%P20
*          FMS,X    =FSDIV(%P11,%P21)  P11*X
*          FMS,Z    =FSDIV(%P13,%P23)  P13*Z
*          FAS,X    Z            NEW X = P10 +P11*X +P13*Z
*          FAS,X    =%P10
1Z1 EQU      %                NOW THE POINT IS 2-DIMENSIONAL
*

```

```

*  RANGE CHECKING
*

```

```

*          CLM,X    XLIMITS      IS X IN RANGE ?
*          BCS,9    1Z2          NO
*          CLM,Y    YLIMITS      YES, IS Y IN RANGE ?
*          BCR,9    1Z4          YES
1Z2 CI,F     EDGEFLAG      NO: NEW PNT OFF SCREEN; WAS LAST PNT
*          BAZ      1Z3          ALSO OFF SCREEN ?
*          STD,X    LASTPNT      YES - WE'RE WAITING TO RE-ENTER,
*          B        0,LX          DON'T PLOT IT.
1Z3 STW,LX   SNDPTRET      NO - WE JUST WENT OFF SCREEN:
*          BAL,LX   EDGEPLT      SET 'EDGEFLAG', INTERPOLATE TO
*                                EDGE, PLOT THE EDGE POINT.
*          AND,F    =-1-VECTORFG  BREAK OFF OLD VECTOR/START NEW
*          CI,F     RENTFLAG      SEE IF RE-ENTRANCE BEING ALLOWED
*          BANZ     *SNDPTRET     YES, KEEP GETTING POINTS AND HOPING
*          B        QUIT          NO, STOP ALTOGETHER
1Z4 CI,F     EDGEFLAG      NEW PNT ON SCREEN: HOW ABOUT LAST

```

	BAZ	PLOT	PNT ? IF BOTH ON, PLOT NEW ONE.
	STD,X	SAVEPNT	NEW ON, LAST OFF: WE JUST NOW
	STW,LX	SNDPTRET	RE-ENTERED: RESET 'EDGEFLAG',
	BAL,LX	EDGEPLLOT	INTERPOLATE TO EDGE, PLOT EDGE
	LW,LX	SNDPTRET	POINT,
	LD,X	SAVEPNT	
	B	PLOT	THEN PLOT NEW DATA POINT.
*			
*			
EDGEPLLOT	EQU	%	
	EOR,F	=EDGEFLAG	REVERSE EDGE FLAG
	CI,F	VECTORFG	IF THE VERY FIRST POINT IS OFF
	BAZ	0,LX	SCREEN, THERE'S NO 'LASTPNT'.
*			
*	INTERPOLATE	TO EDGE	
*			
	LD,X+2	LASTPNT	WE'VE GOT X1/Y1, GET X2/Y2
	STD,X	LASTPNT	REMEMBER NEW POINT
	XW,Y	X+2	PUT X'S TOGETHER, Y'S TOGETHER
	LCI	4	REMEMBER X1, X2, Y1, Y2
	STM,X	TEMPX1	
	FSS,X+1	X	
	STW,X+1	DELX	SAVE X2-X1
	FSS,X+3	X+2	
	STW,X+3	DELY	SAVE Y2-Y1
	LI,X	0	TRY LEFT EDGE FIRST: X = 0 (LEFT)
	LW,Z	TEMPX1	SEE IF X1/X2 HAVE OPPOSITE SIGNS
	EOR,Z	TEMPX2	(I.E. 0 IS BETWEEN X1/X2).
	BLZ	ZZ1	YES
	LW,X	UPPERX	NO, TRY RIGHT EDGE: X = UPPERX (RT)
	CLM,X	TEMPX1X2	DOES RIGHT EDGE LIE BETWEEN X1/X2
	BCR,9	ZZ1	(IN EITHER DIRECTION) ?
	BCS,6	ZZ2	NO: CAN ONLY LIE ON TOP OR BOTTOM
ZZ1	EQU	%	YES, WE HAVE X2-X1 NONZERO; THE
*			EDGE POINT MAY BE ON TOP, BOTTOM,
*			OR LEFT (IF X=0, RIGHT IF X>0)
*			EDGE.
	LW,Y	X	COMPUTE POTENTIAL Y-VALUE OF
	FSS,Y	TEMPX1	LEFT (RIGHT) EDGE POINT:
	FDS,Y	DELX	$Y = Y1 + (Y2-Y1)*(X-X1)/(X2-X1).$
	FMS,Y	DELY	
	FAS,Y	TEMPY1	
	BLZ	ZZ2	WE'VE GOT THE CORRECT EDGE POINT
	CW,Y	UPPERY	ONLY IF (1) Y LIES ON THE SCREEN,
	BG	ZZ2	
	CLM,Y	TEMPY1Y2	AND (2) Y LIES BETWEEN Y1/Y2.
	BCR,9	PLOT1	
	BCR,6	PLOT1	
ZZ2	EQU	%	THE REQUIRED EDGE POINT LIES ON
*			THE TOP OR BOTTOM EDGE;
*			Y2-Y1 IS NONZERO.
	LI,Y	0	IT'S THE BOTTOM (Y=0) IF AND ONLY
	LW,Z	TEMPY1	IF Y1/Y2 HAVE OPPOSITE SIGNS
	EOR,Z	TEMPY2	(0 LIES BETWEEN Y1/Y2).
	BLZ	ZZ3	
	LW,Y	UPPERY	OTHERWISE, IT'S ON THE TOP
ZZ3	LW,X	Y	COMPUTE CORRESPONDING X VALUE:
	FSS,X	TEMPY1	$X = X1 + (X2-X1)*(Y-Y1)/(Y2-Y1).$
	FDS,X	DELY	
	FMS,X	DELX	
	FAS,X	TEMPX1	
	B	PLOT1	
*			
*			
PLOT	EQU	%	PLOT POINT
	STD,X	LASTPNT	REMEMBER LAST POINT PLOTTED
*			
*	WINDOWING		
*			
PLOT1	EQU	%	
	FMS,X	XRATIO,GP	REDUCE TO WINDOW SIZE
	FMS,Y	YRATIO,GP	
	FAS,X	XDISPFX,GP	SHIFT TO WINDOW LOCATION, AND 'FIX'
	FAS,Y	YDISPFX,GP	COORDINATE VALUES; I.E., THEY ARE
*			NO LONGER FLOATING-SHORT, BUT
*			=X'46800XXX' (XXX IS A 10-BIT
*			INTEGER).
	CI,F	DASHFLAG	ARE WE TO DASH THIS CURVE ?


```

        BAZ      OUTPOINT      NO, JUST CONTINUE CURRENT VECTOR
        STW,LX   PLOTRET        YES, BREAK IT UP: FOR EACH POINT
        AND,F    =-1-VECTORFG   SEND NEW 'START VECTOR' COMMAND,
        BAL,LX   OUTPOINT      FOLLOWED BY POINT (X,Y) FOLLOWED
        LW,X     TEMPYYXX      BY SLIGHTLY DISPLACED POINT
        AI,X     X'00010001'   (X+1,Y+1).
        CI,X     1024          CHECK FOR SCREEN WRAPAROUND
        BAZ      20Z1          NO
        AI,X     -2            YES-DISPLACE TO LEFT.
20Z1      LW,LX   PLOTRET
        B        OUT2NDPT
*
*
OUTPUTPOINT EQU      %      OUTPUT POINT
        CI,F      VECTORFG   IF NOT ALREADY IN VECTOR MODE,
        BANZ      3Z1
        AI,F      VECTORFG   GET INTO IT BY SENDING
        OUTPUT    #GS        'START VECTOR';
        LI,Z      -1
        STW,Z     LASTXHI     ALSO INIT HIGH-ORDER X/Y MEMORY.
        STW,Z     LASTYHI
3Z1      STH,Y     X          COMBINE Y AND X COORDS INTO 1 WORD
        STW,X     TEMPYYXX   SAVE FOR 'PLOT' IN DASH MODE
OUT2NDPT EQU      %      (FOR 'PLOT' IN DASH MODE)
        CI,F      OFFSETFLG  IF OFFSET FLAG =1,
        BAZ      3Z4
        AW,X      S          ADD X/Y OFFSETS TO X/Y.
3Z4      EQU      %
        SLS,X     6          DISCARD EXTRANEIOUS BITS
        LI,Y     #YHIGH**-5  COMPUTE HIGH-ORDER Y BYTE
        SCD,X     5
        CW,Y     LASTYHI     IF DIFFERENT FROM LAST ONE,
        BE       3Z2
        STW,Y     LASTYHI
        OUTPUT    *Y
3Z2      LI,Y     #YLOW**-5   OUTPUT IT.
        SCD,X     5          COMPUTE LOW-ORDER Y BYTE
        OUTPUT    *Y
        SLS,X     6          OUTPUT IT
        LI,Y     #XHIGH**-5  DISCARD EXTRANEIOUS BITS
        SCD,X     5          COMPUTE HIGH-ORDER X BYTE
        CW,Y     LASTXHI     IF DIFFERENT FROM LAST ONE,
        BE       3Z3
        STW,Y     LASTXHI
        OUTPUT    *Y
3Z3      LI,Y     #XLOW**-5   OUTPUT IT.
        SCD,X     5          COMPUTE LOW-ORDER X BYTE
        OUTPUT    *Y
        B        0,LX        OUTPUT IT
*                                     EXIT
*
*      LOCAL DATA/TEMPS
*
LASTPNT  DTEMP
SAVEPNT  DTEMP
TEMPX1X2 DTEMP
TEMPY1Y2 DTEMP
TEMPX1   EQU      TEMPX1X2
TEMPX2   EQU      TEMPX1X2+1
TEMPY1   EQU      TEMPY1Y2
TEMPY2   EQU      TEMPY1Y2+1
DELX     TEMP
DELY     TEMP
LASTXHI  TEMP
LASTYHI  TEMP
TEMPYYXX EQU      TEMPX1X2
SNDPTRET TEMP
PLOTRET  TEMP
*
*
        BOUND      8
XLIMITS  DATA     0,%XPTS   SCREEN LIMITS FOR X COORD (IN PTS)
YLIMITS  DATA     0,%YPTS   SCREEN LIMITS FOR Y COORD (IN PTS)
UPPERX   EQU      XLIMITS+1  UPPER X BOUND
UPPERY   EQU      YLIMITS+1  UPPER Y BOUND
        PAGE
*
*
*      DRAW AXES

```

```

*
*
*           CORRESPONDS TO
*           5 #GRF NUL
*           CALLED WITH GP SET UP.  EXITS VIA 'RETURN'.
*
DRAWAXES EQU      %
          BAL, LX  GETCOORD          REMEMBER CURSOR
          LI, CC   0                INIT OUTPUT
          LW, F    ZSCALE, GP       DECIDE WHETHER 2-D OR 3-D
          BEZ      13Z1
          LI, F    3DFLAG
13Z1      STW, F    AXFLAGS          3DFLAG IS SET ACCORDINGLY IN AXFLAGS
          BEZ      13Z2
          LW, R    ZMIN, GP         IF IT'S 3-D, X AND Y AXES WILL BE
          EOR, R    ZMAX, GP         DRAWN ONLY IF Z=0 IS IN RANGE
          BGEZ     13Z5             (I.E., ZMIN<0<=ZMAX).
13Z2      LW, R    YMIN, GP         IF Y=0 (AND Z=0, IF 3D) IS
          EOR, R    YMAX, GP         IN RANGE,
          BGEZ     13Z3
          LI, K     0                DRAW X-AXIS.
          BAL, L2   AXIS
13Z3      LW, R    XMIN, GP         IF X=0 (AND Z=0, IF 3D) IS
          EOR, R    XMAX, GP         IN RANGE,
          BGEZ     13Z4
          LI, K     1                DRAW Y-AXIS.
          BAL, L2   AXIS
13Z4      CI, F    3DFLAG          IF 3D,
          BAZ      13Z6
13Z5      LW, R    XMIN, GP         AND X=0 IS IN RANGE,
          EOR, R    XMAX, GP
          BGEZ     13Z6
          LW, R    YMIN, GP         AND Y=0 IS IN RANGE,
          EOR, R    YMAX, GP
          BGEZ     13Z6
          LI, K     2                DRAW Z-AXIS.
          BAL, L2   AXIS
13Z6      BAL, LX  ENDVEC          END LATEST VECTOR
          B         NEXTLINE       RESTORE CURSOR, EXIT.
          PAGE

*
*
*
*
*           PROCESS AXIS SELECTED BY K
*
*           AXIS SELECTION IN K, LINK IN L2.
*
AXIS      EQU      %
          LB, R     BIASTBL, K      GET BIAS OF XMIN, YMIN, OR ZMIN
          LCI       2
          LM, R     *R, GP          FETCH MIN/MAX OF SELECTED AXIS
          STD, R    KMINMAX        REMEMBER THEM
          LI, R     0
          STW, R    XCOORD          INIT POINT = ORIGIN (0,0,0)
          STW, R    YCOORD
          STW, R    ZCOORD
          LW, F     AXFLAGS        INIT FLAGS: OFFSET=VECTOR=0; 3D=0/1

*
*
*           AXIS LINE
*
          LW, R     KMIN
          BAL, LX   KAXISPT        SEND LOWEST POINT ON K-AXIS
          LW, R     KMAX
          BAL, LX   KAXISPT        SEND HIGHEST POINT ON K-AXIS

*
*
*           TIC MARKS
*
          AI, F     OFFSETFLG      FOR TICS AND LIMITS WE NEED TO
                                   APPLY POINT OFFSETS.
                                   THERE ARE 2 CASES, DEPENDING ON
                                   WHETHER OR NOT 0 IS IN RANGE
                                   (I.E., KMIN<0<=KMAX).
                                   CASE 1: ZERO IS OFF SCREEN ---
                                   STARTING AT MAX POINT ON K-AXIS,
                                   PRODUCE TICS WITH SEPARATION
                                   =(MIN-MAX)/N ALONG K-AXIS.
14Z1      LW, R     KMIN
          EOR, R     KMAX
          BLZ       14Z1
          LW, R     KMIN
          FSS, R     KMAX
          FDS, R     =%NTICS
          B         14Z3
          LAW, R     KMIN
          LAW, R1    KMAX
          CW, R      R1
          BGE       14Z2
          BGE       14Z2

```

```

14Z2      LAW,R      KMAX
FDS,R      =%NTICS

*
          STW,R      XCOORD,K
          BAL,L3     SENDTICS
          LCW,R      KDELTA
          STW,R      XCOORD,K
*
14Z3      BAL,L3     SENDTICS
*
*          AXIS LIMITS
*
          LW,R      KMIN
          LW,S      LIM10FST,K
          BAL,L3     SENDLIM
          LW,R      KMAX
          LW,S      LIM20FST,K
          BAL,L3     SENDLIM
          B          *L2
*
*
*          LOCAL DATA/TEMPS
*
KMINMAX    DTEMP
KMIN        EQU      KMINMAX
KMAX        EQU      KMINMAX+1
AXFLAGS     TEMP
XCOORD      TEMP
YCOORD      TEMP
ZCOORD      TEMP
KDELTA      TEMP
*
BIASTBL     DATA,1  XMIN,YMIN,ZMIN
            BOUND    4
*
*          POINT OFFSET TABLES
*
OFST        COM,32   AF(2)**16+AF(1)
*
*          TIC MARKS
*
TICOFSET    EQU      %
            OFST      0,5
            OFST      5,0
            OFST      -1,5
*
LIM10FST    EQU      %
            OFST      0,9
            OFST      12,0
            OFST      12,-18
*
LIM20FST    EQU      %
            OFST      -12,9
            OFST      12,-18
            OFST      12,0
            PAGE
*
*
*          SEND A SERIES OF TIC MARKS
*
*          AXIS SELECTION IN K, SEPARATION VALUE IN R, 1ST POINT
*          IN X/Y/ZCOORD. LINK IS L3.
*
SENDTICS    EQU      %
            STW,R      KDELTA
            LW,R      XCOORD,K
15Z1        CLM,R      KMINMAX
            BCS,9      *L3
            LW,S      TICOFSET,K
            BAL,LX     KAXISPT1
            LW,X      TEMPPYXX
            LCW,S      TICOFSET,K
            BAL,LX     OUT2NDPT
            LW,R      XCOORD,K
            FAS,R      KDELTA
            B          15Z1
*
*
*          STARTING AT THE ORIGIN, PRODUCE
*          TICS ALONG POSITIVE K-AXIS;
*
*          AGAIN STARTING AT ORIGIN, MAKE
*          TICS ON NEGATIVE K-AXIS
*          BY NEGATING SEPARATION VALUE.
*
*          GO TO MIN END OF K-AXIS
*          + FIXED OFFSET
*          POSITION BEAM & WRITE LOWER LIMIT
*          GO TO MAX END OF K-AXIS
*          + FIXED OFFSET
*          POSITION BEAM & WRITE UPPER LIMIT
*          EXIT
*
*          MIN/MAX VALUE ON K-AXIS
*          MIN VALUE ON K-AXIS
*          MAX VALUE ON K-AXIS
*          AXES FLAGS INITIALIZER
*          X COORDINATE OF K-AXIS POINT
*          Y COORDINATE OF K-AXIS POINT
*          Z COORDINATE OF K-AXIS POINT
*          DISTANCE BETWEEN TIC MARKS ON K-AXIS
*
*          AXIS SELECTION BIASES
*
*          XMIN (LEFT): DIRECTLY ABOVE
*          YMIN (BOTTOM): DIRECTLY TO RIGHT
*          ZMIN (TOP RIGHT): RIGHT AND BELOW
*
*          XMAX (RIGHT): ABOVE, A BIT TO LEFT
*          YMAX (TOP): TO RIGHT, A BIT BELOW
*          ZMAX (BOT LEFT): DIRECTLY TO RIGHT
*
*          STORE INCREMENT
*          TEST 1ST POINT
*          IF POINT IS OFF SCREEN,
*          STOP AND EXIT.
*          ON SCREEN: SEND THE K-AXIS POINT
*          BUT SPECIFY FIXED OFFSET.
*          NOW REPEAT THE SAME POINT BUT
*          NEGATE THE OFFSET.
*
*          INCREMENT POINT ALONG K-AXIS
*          AND TEST IT.

```

```

KAXISPT1 AND,F      ==-1-VECTORFG      RESET VECTOR FLAG (START NEW VEC)
KAXISPT  STW,R      XCOORD,K          STORE K-AXIS COMPONENT
      LCI           3
      LM,X          XCOORD            GET X/Y/Z COORDS OF K-AXIS PT
      B            SENDPNT           SEND IT; EXIT VIA LX.
      PAGE

*
*
* SEND AXIS LIMIT
*
*          POSITIONS BEAM ON K-AXIS AT VALUE GIVEN IN 'R'
*          OFFSET BY VALUE IN 'S'; WRITES OUT THE GIVEN
*          LIMIT VALUE (R) STARTING AT THAT BEAM POSITION.
*          LINK IS L3.
*
SENDLIM  EQU        %
      BAL,LX        KAXISPT1          SEND K-AXIS POINT R, OFFSET BY S
      OUTPUT        #US              END GRAPH MODE
      BAL,L0        GRAFOUT          AND DUMP OUT GRAPH MODE STUFF.
*                                     NOW WE'RE IN ALPHA MODE FOR LIMIT
      LW,R          XCOORD,K          GET BACK THE LIMIT VALUE

*
*
*          CODE TO CONVERT F.S. VALUE IN 'R' TO TEXT STRING
*          IN GRAFBUF. WIDTH IS CONTROLLED BY 'GDIGITS'
*          INSTEAD OF 'DIGITS'. DEFAULT VALUE OF 'GDIGITS'
*          IS 3 (INTRINSIC COULD BE ADDED TO CHANGE THIS).
*
*          THIS CODE INTERFACES WITH EXTERNAL CONVERSION
*          ROUTINES THAT EMPLOY A DIFFERENT REGISTER
*          NAMING CONVENTION; HENCE.....
*
      OPEN          R0,R4,R5,R7,R8,R9,R11,R12
R0        EQU        0                (F)
R4        EQU        4                (A,GP)
R5        EQU        5                (LX)
R7        EQU        7                (LX7, AF1)
R8        EQU        8                (R)
R9        EQU        9                (R1)
R11       EQU        11               (S)
R12       EQU        12               (L3)
*
      STW,K          KSAVE            SAVE K
      STW,GP         GPSAVE          SAVE GP
      STD,L3         L3L2SAVE        SAVE L3 AND L2
      XW,R0          GDIGITS         SAVE R0,GET GDIGITS
      XW,R0          DIGITS          SWITCH DIGITS TO GDIGITS
      EQUAL,R,R8     FS LIMIT VALUE IS IN 'R' (R8)
      LI,R9          0               CONVERT FS TO FL
      BAL,R11        CREALBIN        CONVERT TO INTEGER-EXPONENT
      LI,R5          BA(GRABUF)      TARGET ADDRESS
      CI,R7          X'72'           CHECK FOR NEGATIVE VALUE
      BNE            16Z1            NO
      BAL,R11        SETCHAR
16Z1      CI,R12      -4              CHECK IF E-FORM NEEDED
      BLE            16Z2            YES
      CW,R12         DIGITS
      BG             16Z2            YES
      BAL,R4         GENDIGS        NO
      B             16Z3
16Z2      BAL,R4     GENDIGSE        E-FORM GEN. DIGITS
      BAL,R4         GENEXP          AND EXPONENT
16Z3      AI,R5      -BA(GRABUF)     GET BYTE COUNT
      LW,CC          R5
16Z4      AI,R5      -1
      BLZ            16Z5
      LB,R4          GRAFBUF,R5
      LB,R4          OUTRANST,R4     TRANSLATE TO TERMINAL CHAR.
      STB,R4         GRAFBUF,R5
      B             16Z4
16Z5      LD,L3      L3L2SAVE        LOOP
      LW,GP         GPSAVE          RESTORE L3 AND L2
      LW,K          KSAVE          RESTORE GP
      XW,R0         DIGITS         RESTORE K
      XW,R0         GDIGITS        DIGITS
*
      CLOSE          R0,R4,R5,R7,R8,R9,R11,R12

```

```

*
*
*      BAL, LO   GRAFTEXT      OUTPUT BUFFER IN ALPHA MODE
*      B        *L3          EXIT
*
*
KSAVE   TEMP
GPSAVE  TEMP
L3L2SAVE DTEMP
        PAGE
*
*
*   DRAW BOX
*
*           CORRESPONDS TO
*           9 #GRF NUL
*           CALLED WITH GP SET UP.  RETURNS VIA 'RETURN'.
*
DRAWBOX EQU      %
        BAL, LX  GETCOORD      REMEMBER CURSOR LOC
        LI, CC   0             INIT OUTPUT
        LI, F    RENTFLAG      INIT FLAGS FOR SENDPNT
        LW, X    XMIN, GP      START WITH LOWER LEFT CORNER
        LW, Y    YMIN, GP
        BAL, LX  SENDPNT
        LW, X    XMIN, GP
        LW, Y    YMAX, GP      MOVE UP
        BAL, LX  SENDPNT
        LW, X    XMAX, GP      MOVE RIGHT
        LW, Y    YMAX, GP
        BAL, LX  SENDPNT
        LW, X    XMAX, GP
        LW, Y    YMIN, GP      MOVE DOWN
        BAL, LX  SENDPNT
        LW, X    XMIN, GP      MOVE LEFT
        LW, Y    YMIN, GP
        BAL, LX  SENDPNT
        BAL, LX  ENDVEC        END VECTOR
        B        NEXTLINE      RESET CURSOR, EXIT
        PAGE
*
*
*   SET COORDINATE
*
*           CORRESPONDS TO
*           4 #GRF X, Y          ..OR
*           4 #GRF X, Y, Z
*           THE TWO OR THREE COORDINATES ARE IN 'PARAMS' BUFFER;
*           GR AND 3D-FLAG ARE SET UP.  SETS CURSOR TO (X,Y) OR
*           (X,Y,Z).  EXITS VIA 'RETURN'.
*
SETCOORD EQU      %
        BAL, LX  GETCOORD      SAVE CURRENT CURSOR LOC IN CASE
*                                     OF LATER '4 GRF NUL'.
        LW, X    GRXYBYTS      SAVE CURSOR COORDS
        STW, X   XYSAVE
        LCI      3
        LM, X    PARAMS        GET X, Y, (Z)
        LI, CC   0             INIT OUTPUT
        BAL, LX  SENDPNT      SEND ONE POINT (X,Y,(Z))
ENDOUT   BAL, LX  ENDVEC        STOP VECTOR
        BAL, LO  GRAFOUT      DUMP OUTPUT BUFFER
        B        *RETURN      EXIT
        PAGE
*
*
*   ERASE AND HOME
*
*           CORRESPONDS TO
*           8 #GRF NUL
*           ERASES SCREEN AND 'HOMES UP' CURSOR.  EXITS VIA 'RETURN'.
*
ERASHOME EQU      %
        LI, K    1
        BAL, LX  SCREENCT      DO A #GRF 1
        LI, K    2
*   FALLS INTO  SCREENCT      THEN #GRF 2 AND EXIT

```

```

PAGE
*
*
* SCREEN CONTROL
*
* CORRESPONDS TO
*      8 #GRF K
* CALLED WITH K IN 'K'. THIS ROUTINE EITHER
* SENDS 'ERASE' COMMAND (K=1) OR SENDS CURSOR
* TO 'HOME' POSITION (K=2). FOR K=1, RETURNS
* VIA LINK IN LX; FOR K=2, RETURNS VIA 'RETURN'.
*
SCREENCT EQU %
        LI,CC 0 INIT OUTPUT
        LI,F 0 INIT FLAGS (ONLY VECTORFG USED)
        BDR,K 9Z1 WHICH FUNCTION IS IT ?
        OUTPUT #ESC,#FF K=1: ERASE - SEND 'ERASE' COMMAND;
        BAL,L0 GRAFOUT DUMP OUTPUT BUFFER;
CALWT13 CAL1,8 FPTWT13 DELAY FOR ERASE OF SCREEN
        B 0,LX EXIT (LINK).
FPTWT13 DATA X'0F000002' DELAY 2 UNITS
9Z1 LD,X HOMEPOS K=2: HOME - GET COORDS (IN PTS) OF
        BAL,LX OUTPOINT 'HOME' POSITION; SEND IT;
        B ENDOUT END VECTOR; DUMP OUTPUT; EXIT.
*
        BOUND 8
HOMEPOS DATA 0,FSADD(X'46800000',%YPTS)-19
        PAGE
*
*
* END VECTOR MODE
*
* CALLED WITH FLAGS IN F. LINK IS LX.
*
ENDVEC EQU %
        CI,F VECTORFG IF NOT IN VECTOR MODE,
        BAZ 0,LX SKIP IT.
        AI,F -VECTORFG IN: GET OUT
        OUTPUT #US
        B 0,LX EXIT
        PAGE
*
*
* P A R A M E T E R S E T U P R O U T I N E S
*
* INITIALIZE PARAM DATA BLOCK
*
* IF 'GRAFPNTR' IS ZERO, THIS ROUTINE ALLOCATES THE DATA
* BLOCK AND INITIALIZES IT TO DEFAULT PARAMS. RETURNS
* GRAFPNTR IN GP. LINK IS LX.
* ALSO TESTS TERMINAL TYPE TO MAKE SURE USER
* SPECIFIED 'TERM 13' (TEK 4013 TERMINAL).
*
INITGRAF EQU %
        LI,R 13
        CW,R TERMTYPE IS IT TERM 13?
        BE 17Z1 YES
        LI,2 IDTERMAL NO, SIGNAL TERMINAL TYPE ERROR
        B 17Z1 ERTERMAL
        LI,R 0
        STW,R GRFIOFLG INIT I/O FLAG (SEE ERRGRFIO IN UTSG)
        STW,R GRXYBYTS INIT CURSOR LOC (--DITTO--)
        LW,GP GRAFPNTR GET PARAM DATA BLOCK PNTR
        BNEZ 0,LX IF ALREADY IN EXISTENCE, RETURN IT.
        STW,LX LINKTEMP
        LI,S GRDBSIZE-2 ELSE, CREATE ONE (SIZE IN 'S'
        BAL,LX7 ALOCHNW EXCLUDES HEADER). ALOCHNW RETURNS
        EQUAL,A,GP PNTR IN 'A', WHICH IS ALSO 'GP'.
        STW,GP GRAFPNTR SAVE PNTR
        LI,R INTG**8+1 SET UP PHONY TYPE AND RANK
        STH,R *GRAFPNTR IN DATA BLOCK HEADER.
        LI,S GRDBSIZE-3 LENGTH OF PHONY INTEGER VECTOR
        STW,S 2,GP SET IN DATA BLOCK HEADER
        LCI 9 COPY DEFAULT PARAMS
        LM,R-1 DEFAULTS INTO NEW DATA BLOCK.
        STM,R-1 FLAGS,GP
        LM,R-1 DEFAULTS+9

```

```

STM,R-1  FLAGS+9,GP
LCI      5
LM,R-1   DEFAULTS+18
STM,R-1  FLAGS+18,GP
B        *LINKTEMP      EXIT W/PNTR IN 'GP'

*
LINKTEMP TEMP
GRFIOFLG TEMP
*
*   DEFAULT PARAMETERS
*
DEFAULTS EQU      %           DEFAULT PARAMS
DEFFLAGS EQU      %           FLAGS:
DATA      CENTERFG+;         SPECIFY CENTERED AUTOMATIC SCALING,
          ASCLFLAG+;
          ISCLFLAG+;         INPUT COORDINATE SCALING,
          RENTFLAG          AND CURVE RE-ENTRANCE.
DEFSCALE EQU      %           SCALING:
DATA      0,%XPTS           X LIMITS
DATA      0,%YPTS           Y LIMITS
DATA      0,0              Z LIMITS
DATA      FS'1',FS'1',0     X/Y/Z MULTIPLIERS
DEFWINDO EQU      %           WINDOW:
%XDISP    EQU      FSSUB(%XINCHS,%YINCHS)
DATA      %XDISP,%XINCHS    LARGEST SQUARE AT RIGHT
DATA      0,%YINCHS
DATA      FSADD(X'46800000',; WITH CORRESPONDING 'DISPFIX' VALS
          FSMUL(%XDISP,%PPI))
DATA      %ASPECT          AND RATIOS.
DATA      X'46800000'
DATA      FS'1'
DATA      0,0,64,0,0      INXCOORD,INYCOORD,INCHAR,SPARE1,SPARE2
PAGE

*
*
*   SCALING PARAM MODIFICATION
*
*   SCALING PARAMS ARE CHANGED IN ONE THE FOLLOWING WAYS:
*
*   1. ORIGINALLY, 'INITGRAF' SETS THEM ALL TO ZERO AND
*      SETS THE AUTOMATIC SCALING FLAG (SEE 3 BELOW).
*
*   2. USER CAN SET THEM EXPLICITLY VIA
*      3 #GRF X1,X2,Y1,Y2      OR
*      3 #GRF X1,X2,Y1,Y2,Z1,Z2  OR
*      3 #GRF NUL .
*      THE FIRST TWO SET THE X AND Y (AND Z) LIMITS AS
*      GIVEN AND RESETS THE 'AUTOMATIC SCALING' FLAG SO
*      THAT THEY WON'T GET CHANGED BY THE CURVE PLOTTER;
*      THESE FORMS ARE HANDLED BY THE ROUTINE 'SETSCALE'.
*      THE THIRD FORM ABOVE, HANDLED BY 'NOSCALE', SETS X/Y
*      SCALING TO USE X AND Y DIRECTLY AS POINT COORDINATES
*      WHERE THE X RANGE IS (0,%XPTS) AND THE Y RANGE
*      IS (0,%YPTS); 'NOSCALE' ALSO RESETS 'AUTOSCALE' FLAG
*      AND SETS THE WINDOW PARAMS TO FULL-SCREEN SIZE.
*
*   3. IF THE 'AUTOSCALE' FLAG IS SET WHEN THE CURVE
*      PLOTTER IS CALLED, THE SCALING PARAMETERS ARE
*      SET ACCORDING TO THE MIN/MAX OF THE DATA IN EACH
*      OF ITS COORDINATES (ALSO MAKING USE OF THE 'CENTER
*      SCALING' FLAG SETTING). THIS IS HANDLED BY THE
*      ROUTINE 'AUTOSCAL' CALLED BY 'QZOUTPUT'.
*
*   SET SCALING ACCORDING TO USER'S PARAMS
*
*   CORRESPONDS TO
*      3 #GRF X1,X2,Y1,Y2      OR
*      3 #GRF X1,X2,Y1,Y2,Z1,Z2
*   THE FOUR (SIX) PARAMS ARE IN 'PARAMS'.
*   RESETS AUTO- AND CENTER-SCALING FLAGS.
*   CALLED WITH GP AND 3DFLAG SET UP. LINK IS LX.
*
SETSCALE EQU      %
LCI      6
LM,R     PARAMS      GET X1,X2,Y1,Y2,(Z1,Z2)
STM,R    XMIN,GP     STORE IN X/Y/Z LIMIT PARAMS
FSS,R+1  R           COMPUTE X SPAN

```

```

BNEZ      5Z1
FAS,R     =%EPS
STW,R     XMAX,GP
LW,R+1    =%EPS
5Z1       FSS,R+3  R+2
BNEZ      5Z2
FAS,R+2   =%EPS
STW,R+2   YMAX,GP
LW,R+3    =%EPS
5Z2       FSS,R+5  R+4
BNEZ      5Z3
FAS,R+4   =%EPS
STW,R+4   ZMAX,GP
LW,R+5    =%EPS
5Z3       LCI      3
LM,AF     XSIZEPTS
FDS,AF     R+1
FDS,AF+1   R+3
FDS,AF+2   R+5
CI,F       3DFLAG
BANZ      5Z4
LI,AF+2    0
5Z4       LCI      3
STM,AF     XSCALE,GP
AI,LX      0
BLZ        0,LX
RESETASC  LI,F     0
LI,F+1     ASCLFLAG+CENTERFG
STS,F      FLAGS,GP
B           0,LX

*
*   FIXED DATA
*
XSIZEPTS  DATA    %XPTS
YSIZEPTS  DATA    %YPTS
ZSIZEPTS  DATA    %ZPTS
PAGE
*
*
*   SET UP TRANSPARENT SCALING AND FULL-SCREEN WINDOW
*
*   CORRESPONDS TO
*   3 #GRF NUL
*   SETS X/Y LIMITS TO INTERPRET (X,Y) AS POINT COORDINATES.
*   SETS WINDOW LIMITS TO FULL SCREEN.  RESETS AUTO-SCALE
*   FLAGS.  CALLED WITH GP SET UP.  LINK IS LX.
*
NOSCALE   EQU      %
LCI        9
LM,R-1     TRANSCAL
STM,R-1    XMIN,GP
LCI        8
LM,R       FULLWIND
STM,R      LXWINDOW,GP
B          RESETASC

*
*   TRANSPARENT SCALE PARAMETERS
*
TRANSCAL   EQU      DEFSCALE
*
*   FULL-SCREEN WINDOW PARAMETERS
*
FULLWIND   EQU      %
DATA       0,%XINCHS
DATA       0,%YINCHS
DATA       X'46800000'
DATA       FS'1'
DATA       X'46800000'
DATA       FS'1'
PAGE
*
*
*   AUTOMATIC SCALING
*
*   SETS UP MIN/MAX OR ORIGIN-CENTERED SCALING BASED ON
*   THE DATA BEING PLOTTED.  IT IS CALLED FROM 'QZOUTPUT'
*   AFTER THE DATA-FETCHING XSEG HAS BEEN BUILT.  'AUTOSCAL'
*   USES THIS XSEG TO ACCESS THE DATA POINTS.  CALLED WITH

```

```

IF SPAN IS ZERO, SET MAX=MIN+EPS
SO THAT SPAN WILL BE >0.

```

```

SAME FOR Y

```

```

SAME FOR Z

```

```

GET X/Y/Z SCREEN SIZES (IN PTS)

```

```

XSCALE = XSIZEPTS/(X2-X1)
YSCALE = YSIZEPTS/(Y2-Y1)
ZSCALE = ZSIZEPTS/(Z2-Z1)
IF 2D SCALE BEING SET UP,
SET ZSCALE=0 FOR THE
BENEFIT OF 'DRAWAXES'.

```

```

STORE X/Y/Z SCALE FACTORS
IF ENTERED FROM 'AUTOSCAL',
DON'T TOUCH FLAGS.
RESET AUTO SCALING FLAGS

```

```

SCREEN SIZE, IN POINTS
*
*

```

```

MOVE 'TRANSPARENT' SCALE LIMITS
AND MULTIPLIERS TO PARAM DB..
MOVE FULL-SCREEN WINDOW PARAMS
TO DB.
RESET AUTO-SCALE FLAGS, EXIT

```

```

TRANSPARENT SCALE PARAMS

```

```

FULL SCREEN WINDOW PARAMS
X LIMITS
Y LIMITS
X DISPLACEMENT (FIX)
X RATIO
Y DIPLACEMENT (FIX)
Y RATIO

```



```

*           F, GP, AND XL (LAST XSEG LOC+1) SET UP. LINK IS LX.
*
AUTOSCAL EQU      %
          STW,LX   ASCLRET      SAVE LINK (XSEG USES LX)
          LW,R     CODE5        MODIFY XSEG TEMPORARILY: CHANGE
          STW,R    -4,XL       'BAL SENDPNT' TO 'BAL SCALPNT'.
          SETBRK   ALLOW BREAKS DURING XSEG EXECUTION
          BAL,L2   SCALPNT      START XSEG; AFTER LAST POINT,
*                               XSEG RETURNS HERE...
          RESETBRK STOP BREAKS AFTERWARDS
          LW,R     CODE4        RESTORE XSEG AS IT WAS UPON ENTRY
          STW,R    -4,XL
          LW,LX    ASCLRET      RESTORE LINK
          AI,LX    -1**17      TELL SETSCALE TO LEAVE FLAGS ALONE
          CI,F     3DFLAG       IF 2-DIMENDIONAL DATA,
          BAZ      6Z0
          LI,R     0            SET Z SCALE PARAMS TO 0.
          STW,R    PARAMS+4
          STW,R    PARAMS+5
6Z0 EQU      %
          CI,F     CENTERFG     IF NO CENTERING TO BE DONE,
          BAZ      SETSCALE     GO SET SCALING FROM MINS/MAXES.
          LI,M     4            CENTER THE ORIGIN: FOR EACH COORD...
6Z1 LAW,R      PARAMS,M        LET L/U BE LOWER/UPPER LIMITS: L<U
          LAW,R1   PARAMS+1,M   FETCH AL/AU = ABS(L)/ABS(U)
          CW,R     R1           AL:AU
          BE       6Z3         AL=AU: THUS L=-U, ALREADY CENTERED
          BL       6Z2
          STW,R    PARAMS+1,M   AL>AU: THUS L<0, CHANGE LIMITS
          B        6Z3         TO (L,AL) = (-AL,AL) CENTERED.
6Z2 LCW,R1     R1             AL<AU: THUS U>0, CHANGE LIMITS
          STW,R1   PARAMS,M    TO (-AU,U) = (-AU,AU) CENTERED.
6Z3 AI,M       -2            DO ALL COORDS
          BGEZ     6Z1
          B        SETSCALE    THEN SET SCALING TO CENTERED PARAMS
*
*
SCALPNT EQU      %
          BAL,L3   XSEGBASE     SET L3 SO 1ST POINT WILL BE
          STW,X     PARAMS+0     PROCESSED HERE:
          STW,X     PARAMS+1     XMIN=XMAX=X
          STW,Y     PARAMS+2
          STW,Y     PARAMS+3     YMIN=YMAX=Y
          STW,Z     PARAMS+4
          STW,Z     PARAMS+5     ZMIN=ZMAX=Z
          BAL,L3   0,LX        SET L3 SO THAT 2ND AND LATER
*                               POINTS WILL BE PROCESSED HERE:
          LI,AF     -6          AF RUNS -6, -4, -2 AS AI RUNS
          LI,AI     -3          -3, -2, -1.
6Z4 LW,X         X+3,AI        GET X/Y/Z COORD (CALL IT X)
          CLM,X     PARAMS+6,AI COMPARE MIN:X:MAX
          BCR,9     6Z6         MIN<=X<=MAX, NO CHANGE
          BL       6Z5
          STW,X     PARAMS+7,AF X>MAX, UPDATE MAX
          B        6Z6
6Z5 STW,X         PARAMS+6,AF  X<MIN, UPDATE MIN
6Z6 AI,AF         2           BUMP AF BY 2, AI BY 1
          BIR,AI    6Z4
          B         0,LX      DONE, RETURN TO XSEG FOR NEXT POINT
*
*
ASCLRET TEMP      AUTOSCAL LINK SAVE CELL
CODE5    BAL,LX   *L3      (PLACED IN XSEG TEMPORARILY)
          PAGE
*
*
* WINDOW PARAM MODIFICATION
*
* WINDOW PARAMS ARE CHANGED IN ONE OF THE FOLLOWING WAYS:
*
* 1. ORIGINALLY, 'INITGRAF' SETS THEM TO THE LARGEST
*    POSSIBLE SQUARE WINDOW TOUCHING BOTTOM AND RIGHT
*    EDGES OF SCREEN.
*
* 2. USER CAN SET THEM VIA
*    2 #GRF LX,UX,LY,UY      OR
*    2 #GRF NUL
*    THE FIRST (HANDLED BY 'SETWINDO') SETS THEM AS

```

```

*
*      GIVEN.  THE SECOND ('NOWINDOW') SETS THEM TO THE
*      LARGEST SQUARE WINDOW, AS IN 1 ABOVE.
*
*      3.  THEY ARE SET TO FULL SCREEN SIZE BY 'NOSCALE'
*          IN RESPONSE TO
*          3 #GRF NUL
*
*      SET WINDOW ACCORDING TO USER'S PARAMS
*
*          CORRESPONDS TO
*          2 #GRF LX,UX,LY,UY
*          THE FOUR PARAMS ARE IN 'PARAMS'.  CALLED WITH GP SET
*          UP.  LINK IS LX.
*
SETWINDO EQU      %
          LW,R      PARAMS          LX
          BLZ      ERDOMAIN          TOO LOW
          LW,R+1    PARAMS+1        UX
          CW,R+1    =%XINCHS
          BG      ERDOMAIN          TOO HIGH
          CW,R+1    PARAMS
          BLE      ERDOMAIN          OUT OF ORDER
          LW,R+2    PARAMS+2        LY
          BLZ      ERDOMAIN
          LW,R+3    PARAMS+3        UY
          CW,R+3    =%YINCHS
          BG      ERDOMAIN          TOO HIGH
          CW,R+3    PARAMS+2
          BLE      ERDOMAIN          OUT OF ORDER
          LCI      4
          STM,R     LXWINDOW,GP      STORE 'EM IN D8
          FSS,R+1    PARAMS          UX-LX
          FDS,R+1    =%XINCHS        (UX-LX)/XPTS = XRATIO
          FSS,R+3    PARAMS+2        UY-LY
          FDS,R+3    =%YINCHS        (UY-LY)/YPTS = YRATIO
          FMS,R      =%PPI           LX (IN PTS)
          FAS,R      =X'46800000'    XDISPFIX (= X'46800XXX')
          FMS,R+2    =%PPI           LY (IN PTS)
          FAS,R+2    =X'46800000'    YDISPFIX (=X'46800YYY')
          LCI      4                STORE RATIOS
          STM,R     XDISPFIX,GP      AND DISPLACEMENTS.
          B         0,LX            EXIT
          PAGE
*
*
*      SET UP DEFAULT WINDOW
*
*          CORRESPONDS TO
*          2 #GRF NUL
*          CALLED WITH GP SET UP.  LINK IS LX.
*
NOWINDOW EQU      %
          LCI      8
          LM,R      DEFWINDO        COPY DEFAULT WINDOW PARAMS
          STM,R     LXWINDOW,GP      TO WINDOW DATA.
          B         0,LX            EXIT
          PAGE
*
*
*      FLAG MODIFICATION
*
*          SCALING-MODE FLAGS ARE CHANGED AS FOLLOWS:
*
*      1.  ORIGINALLY, 'INITGRAF' SETS AUTO-SCALE AND CENTERING
*          FLAGS.  THIS IS THE 'DEFAULT' SCALING MODE.
*
*      2.  THESE FLAGS ARE RESET BY ANY COMMAND OF THE FORM
*          3 #GRF NUL ...
*          SCALING IS THEREBY FIXED AT THE USER'S LIMITS OR
*          AT THE 'TRANSPARENT' VALUES.  THE ROUTINES 'SETSCALE'
*          AND 'NOSCALE' DO THIS.
*
*      3.  THE USER CAN CONTROL THE SCALING MODE FLAGS WITH:
*          6 #GRF K
*          FOR K=1, WE SET AUTO-SCALE FLAG, BUT NOT CENTERING
*          FLAG;  THUS EACH FOLLOWING CURVE WILL BE SCALED
*          ACCORDING TO ITS MINIMA/MAXIMA.

```

```

*           FOR K=2, WE SET BOTH FLAGS SO THAT FOLLOWING CURVES
*           WILL EACH BE CENTER-SCALED.
*           FOR K=3, WE RESET BOTH FLAGS SO THAT SCALING WILL
*           REMAIN FIXED AT ITS LAST SETTING.
*           THESE FUNCTIONS ARE PERFORMED BY THE FOLLOWING
*           ROUTINE, CALLED WITH THE K-VALUE IN AI, LINK IN LX,
*           AND GP SET UP:
*
SCALMODE EQU      %
          LB,F     SCLFGTBL,K      GET FLAG BITS
          LI,F+1   ASCLFLAG+CENTERFG GET FLAG BIT MASK
          STS,F    FLAGS,GP        SET APPROPRIATE FLAGS
          B        0,LX            EXIT
*
SCLFGTBL DATA,1  0,ASCLFLAG,ASCLFLAG+CENTERFG,0
PAGE
*
*
*           DASHED-CURVE MODE FLAG IS CHANGED AS FOLLOWS:
*
*           1. ORIGINALLY, 'INITGRAF' RESETS DASHFLAG.
*
*           2. IT IS SET BY 'DASHMODE', BELOW, IN RESPONSE TO:
*              7 #GRF NUL
*
*           3. IT IS RESET AT THE END OF 'QZOUTPUT' AFTER PLOTTING
*              A CURVE.
*
DASHMODE EQU      %
          LI,F     DASHFLAG          SET DASHFLAG
          LI,F+1   DASHFLAG
          STS,F    FLAGS,GP
          B        0,LX            EXIT
PAGE
*
*
*           CURVE RE-ENTRANCE MODE FLAG IS CHANGED AS FOLLOWS:
*
*           1. ORIGINALLY, 'INITGRAF' SETS IT.
*
*           2. THE USER MAY SET OR RESET IT WITH:
*              10 #GRF K
*              WHERE K=1 FOR 'RESET', 2 FOR 'SET'. THE FOLLOWING
*              SUBROUTINE HANDLES THIS (K IS IN K, LINK IS LX,
*              GP IS SET UP):
*
RENTMODE EQU      %
          LI,F     RENTFLAG          GET RE-ENTRANCE MODE BIT
          LI,F+1   RENTFLAG          AND MASK.
FLAGSETK EQU      %
          SLS,F    -2,K              IF K<2 SHIFT BIT OFF ITS MARK
          STS,F    FLAGS,GP          SET OR RESET RENTFLAG
          B        0,LX            EXIT
PAGE
*
*
*           INPUT SCALING MODE FLAG IS CHANGED AS FOLLOWS:
*
*           1. ORIGINALLY, 'INITGRAF' SETS IT.
*
*           2. THE USER MAY SET OR RESET IT WITH:
*              0 #GRF K
*              WHERE K=1 FOR 'RESET', K=2 FOR 'SET'. THE FOLLOWING
*              ROUTINE HANDLES THIS (K IS IN K, GP IS SET UP,
*              LINK IS LX):
*
ISCLMODE EQU      %
          LI,F     ISCLFLAG          SPECIFY INPUT-SCALE FLAG
          LI,F+1   ISCLFLAG
          B        FLAGSETK          SET FLAG ACCORDING TO K VALUE; EXIT
PAGE
*
*
*           SET DEVICE
*
*           CORRESPONDS TO
*              1 #GRF K
*           THIS IS ACCEPTED ONLY FOR PURPOSES OF COMPATIBILITY

```

```

*           WITH BERKELEY'S SYSTEM.  ONLY K=3 IS ALLOWED.  LINK IS LX.
*
SETDEV  EQU      %
        CI,K     3           IF K=3 (MEANING 'TEK 4013 TERMINAL')
        BE      0,LX        IT'S OK.
        B       ERDOMAIN    ELSE ERROR
        PAGE

*
*
* RETURN PARAMETER VALUES
*   OR SET GRAPHICS INPUT TO 5,6,OR 7 CHARACTER READ
*   CORRESPONDING TO STRAPPING OPTION FOR TERMINAL
*   (DEFAULT IS 7)
*
*   CORRESPONDS TO
*       11 #GRF K
*   FOR K=1, RETURNS SCALING INFO AS 4- OR 6-WORD FLOT
*   VECTOR.
*   FOR K=2, RETURNS WINDOW INFO AS 4-ELEMENT FLOT VECTOR.
*   FOR K=3, RETURNS X/Y COORDS OF QUAD-ZERO INPUT
*   AS 2-ELEMENT FLOT VECTOR.
*   FOR K=4, RETURNS CHARACTER OF QUAD-0 INPUT AS SCALAR
*   EXITS DIRECTLY TO 'DXRETURN'.
*   FOR K=5,6,OR 7,SETS 'STRAPOPT'
*
PARAMVAL EQU      %
        CI,K     4
        BG      18Z4        SET STRAPOPT
        BE      18Z3        RETURN INCHAR
        CI,K     1
        BNE     18Z1
        LW,R    ZSCALE,GP   IF K=1 AND ZSCALE>0,
        BEZ     18Z1
        AI,K    -1          MODIFY CODE TO SAY '3-D SCALING'.
18Z1    STW,K    KSAVE       SAVE INDEX
        LB,S    PARCOUNT,K COMPUTE DATA SIZE =
        SLS,S    1          N DOUBLEWORDS
        AI,S    2          + DIMEN WORD + GAP WORD.
        BAL,LX7 ALOCHNW     ALOC RESULT DATA BLOCK
        STW,A    RESULT     STORE RESULT POINTER
        LI,R    FLOT**8+1   SET TYPE=FLOT, RANK=1
        STH,R    *RESULT
        LW,K    KSAVE       RESTORE K VALUE (0-3)
        LB,XI   PARBIAS,K   GET 1ST PARAM BIAS
        LB,K    PARCOUNT,K GET PARAM COUNT
        STW,K    2,A        SET DIMEN = COUNT
18Z2    LW,R    *GRAFPNTR,XI GET A PARAM (F.S.)
        LI,R1   0          CONVERT TO F.L.
        STW,R    4,A        STORE IN RESULT DATA BLOCK
        STW,R1   5,A
        AI,XI    1
        AI,A     2
        BDR,K    18Z2       REPEAT FOR EACH PARAM
18Z3    B       DXRETURN    EXIT
        LI,S     1
        BAL,LX7 ALOCHNW     ALLOCATE RESULT DATA BLOCK
        STW,A    RESULT
        LI,R     CHAR**8+0   CHARACTER SCALAR
        STH,R    *RESULT
        LW,R-1   GRAFPNTR    GET CHARACTER
        LW,R     INCHAR,R-1
        AI,A     2
        STB,R    *A         STORE IN RESULT
18Z4    B       DXRETURN
        STW,K    STRAPOPT   SET STRAP OPTION
        B       DXNULRTN

*
*
PARBIAS DATA,1  XMIN,XMIN,LXWINDOW,INXCOORD
PARCOUNT DATA,1 6,4,4,2
PAGE

*
*
* M I S C E L L A N E O U S   S U B R O U T I N E S
*
*
* CONVERT INTEGER TO F.S. VALUE

```

```

*
*          CALLED WITH INTEGER IN AI.  RETURNS WITH F.S. VALUE
*          IN AF.  LINK IS LX.
*
ITOF5      EQU          %
           AI,AI        0          TEST SIGN
           BGEZ         10Z1
           LCW,AI        AI        NEGATIVE: NEGATE IT;
           ODD,AI        AI        CONVERT IT TO F.L. AS A
           EQUAL,AI,AF+1          POSITIVE VALUE SO THAT WE
           LW,AF         =X'4E000000' MAY PROPERLY TRUNCATE TO F.S.;
           SFL,AF        14        NORMALIZE;
           LCW,AF        AF        TRUNCATE TO F.S.; RE-NEGATE;
           B             0,LX      EXIT.
10Z1       LW,AF         =X'4E000000' POSITIVE: APPEND EXPONENT;
           SFL,AF        14        NORMALIZE;
           B             0,LX      EXIT WITH F.S. VALUE IN AF.
*
*
*  CONVERT F.L. TO F.S.
*
*          CONVERTS F.L. VALUE IN AF/AF1 TO F.S. VALUE IN AF.
*          LINK IS LX.
*
FLT5OF5    EQU          %
           AI,AF        0          TEST SIGN: IF >=0, IT'S OK TO
           BGEZ         0,LX      TRUNCATE VALUE.
           LCD,AF        AF        <0: NEGATE F.L.,
           LCW,AF        AF        TRUNCATE POS VAL, RE-NEGATE F.S.
           B             0,LX      EXIT
*
*
18Z        END

```



```

        TITLE      'UTSG-B00,09/11/73,DWG702985'
        PAGE

*
*
*  E X T E R N A L      C O M M U N I C A T I O N
*
*  DEFINITIONS
*
        DEF      FPTWR2      FPT TO WRITE F:Q0G
        DEF      GETCHAR      GET INPUT CHARACTER
        DEF      GETCOORD      GET CURSOR COORDS
        DEF      GRAFOUT      GRAPHICS OUTPUT ROUTINE
        DEF      GRAFTEXT      GRAPHICS TEXTUAL OUTPUT
        DEF      GRBUFLIM      GRAPHICS I/O BUFFER LIMIT
        DEF      UTSG@        START OF PROCEDURE

*
*  REFERENCES
*
        REF      ERI0          GRAPHICS PKG I/O ERROR EXIT
        REF      GRAFBUF        GRAPHICS I/O BUFFER
        REF      GRFIOFLG       GRAPHICS TERM I/O FLAG
        REF      GRINCHAR       GRAPHICS TERM INPUT CHAR
        REF      GRXYBYTES      GRAPHICS TERM X/Y COORD BYTES
        REF      IOERCODE       I/O ERROR CODE WORD
        REF      NEXTLINE       RESTORE CURSOR LOC
        REF      RETURN         RETURN ADR CELL
        PAGE
        REF      STRAPOPT       4013 STRAP OPTION,5,6,OR 7 CHARS.

*
*
*  A S S E M B L Y      P A R A M E T E R S
*
        SYSTEM      BPM
        SYSTEM      SIG5
PROGSECT CSECT      1
UTSG@    EQU        %          START OF PROCEDURE

*
*  REGISTERS
*
CC      EQU        3          OUTPUT CHAR COUNT
LX      EQU        5          LINK REG
R       EQU        8          GENERAL WORK REG
R1      EQU        9          *
LO      EQU        14         LINK FOR 'DUMPOUT'

*
*  TERMINAL CONTROL CHARACTERS
*
#ESC    EQU        X'1B'      NEXT CHAR IS CONTROL
#ENQ    EQU        X'05'      ENQUIRY, ENTER GIN MODE
#SUB    EQU        X'1A'      DISPLAY CROSSHAIR, ENTER GIN MODE

*
*  MISCELLANEOUS
*
ORECSIZE EQU        130        MAX SIZE OF OUTPUT RECORD (BYTES)
PAGE

*
*
*  T E R M I N A L      I / O
*
*
*      USECT      PROGSECT
*
*  GET CURSOR COORDINATES
*
*      READS TERMINAL STATUS BYTE INTO GRSTBYTE AND CURSOR
*      COORDINATE BYTES INTO GRXYBYTES.  LINK IS LX.
*
GETCOORD EQU        %
LI,R     #ESC**8+#ENQ
STH,R    GRAFBUF
CAL1,1   FPTWR2
CAL1,1   FPTRD
LW,R     GRSTBYTE
CI,R     2
BAZ      0,LX
LW,R1    =X'10000000'        TURN ON MOST SIGNIFICANT BIT

```

```

        STS,R1    GRXYBYTS
        B        0,LX                EXIT
*
GRSTBYTE EQU      GRINCHAR          GRAPHICS TERM STATUS BYTE
PAGE
*
*
*   READ CHAR AND CROSSHAIR COORDS
*
*           READS CHAR INTO GRINCHAR AND CROSSHAIR CURSOR COORDINATE
*           BYTES INTO GRXYBYTS.  LINK IS LX.
*
GETCHAR  EQU      %
        LI,R      #ESC**8+#SUB
        STH,R     GRAFBUF
        CAL1,1    FPTWR2            SEND 'DISPLAY CROSSHAIR' COMMAND
        CAL1,1    FPTRD             READ CHAR AND COORD BYTES
        B        0,LX              EXIT
        PAGE
*
*
*   WRITE OUTPUT BUFFER
*
*           OUTPUTS (CC) GRAPHIC TERMINAL CONTROL BYTES FROM
*           'GRAFBUF'.  RESETS CC TO ZERO.
*           LINK IS LO.
*
GRAFOUT  EQU      %
        AI,CC     0                  IF CC=0, THERE'S
        BEZ      *LO                 NOTHING TO OUTPUT.
        CAL1,1    FPTWRGRF          OUTPUT GRAFBUF TO F:Q0G
        LI,CC     0                  RESET CC=0
        B        *LO                EXIT
*
GRBUFLIM DATA    ORECSIZE-4        BUF SIZE
PAGE
*
*
*   WRITE TEXTUAL OUTPUT
*
*           OUTPUTS (CC) ALPHANUMERIC CHARACTERS FROM
*           'GRAFBUF'.  RESETS CC TO ZERO.
*           LINK IS LO.
*
GRAFTEXT EQU      %
        AI,CC     0                  IF CC=0, THERE'S
        BEZ      *LO                 NOTHING TO OUTPUT.
        CAL1,1    FPTWRALF          OUTPUT GRAFBUF TO F:Q0A
        LI,CC     0                  RESET CC=0
        B        *LO                EXIT
        PAGE
*
*
*   ERROR CONTROL
*
ERRGRFIO EQU      %
        STW,10    IOERCODE          SAVE ERROR STATUS INFO
        LI,R      -1                SET I/O FLAG AND TEST ITS
        XW,R      GRFIOFLG          FORMER STATE.
        BLZ      ERIO              IF IT WAS ALREADY SET, THEN WE
*                                   WERE TRYING TO RESTORE CURSOR
*                                   POSITION IN RESPONSE TO A PRIOR
*                                   I/O ERROR - SO FORGET IT.
        LW,R      GRXYBYTS          1ST ERROR: HAD THE CURSOR LOC
        BEZ      ERIO              BEEN SAVED? (IF NOT, EXIT).
        LI,R      ERIO              YES, SET EXIT ADDRESS TO 'ERIO'
        STW,R     RETURN
        LI,CC     0                  INIT CHAR COUNT,
        B        NEXTLINE          AND RESTORE CURSOR LOC.
        PAGE
*
*
*   FPT'S
*
F        COM,1,7,7,17 AFA(1),CF(2),0,AF(1)

```



```

*
FPTWR2  F,X'11'  F:Q0G      M:WRITE  F:Q0G,
        DATA    X'34000000'
        F        GRAFBUF      (BUF, GRAFBUF),
        F        2            (SIZE, 2),
        F        0            (BTD, 0)
*
FPTRD   F,X'10'  F:Q0G      M:READ   F:Q0G,
        DATA    X'34000010'  (WAIT),
        F        GRSTBYTE     (BUF, GRSTBYTE),
        F        *STRAPOPT    (SIZE, 7, 6, OR 5),
        F        3            (BTD, 3)
*
FPTWRGRF F,X'11'  F:Q0G      M:WRITE  F:Q0G,
        DATA    X'34000000'
        F        GRAFBUF      (BUF, GRAFBUF),
        F        *CC          (SIZE, *CC),
        F        0            (BTD, 0)
*
FPTWRALF F,X'11'  F:Q0A      M:WRITE  F:Q0A,
        DATA    X'34000000'
        F        GRAFBUF      (BUF, GRAFBUF),
        F        *CC          (SIZE, *CC),
        F        0            (BTD, 0)
        PAGE
*
*
*   DCB'S
*
*       GRAPHICS MODE DCB:
*
F:Q0G    DSECT    1
F:Q0G    M:DCB    (DEVICE, 'UC'), (BIN), (DRC), ;
              (ERR, ERRGRFIO), (ABN, ERRGRFIO)
*
*       ALPHA MODE DCB:
*
F:Q0A    DSECT    1
F:Q0A    M:DCB    (DEVICE, 'UC'), (BCD), (NODRC), ;
              (ERR, ERRGRFIO), (ABN, ERRGRFIO)
*
*       USECT      PROGSECT
*
END

```


+1,1
+185,185 TITLE 'CONSTS-B00,07/10/74,DWG702985'
APLSINON TEXTC 'APL B00 005'
+END

U02-0001
U02-0002
U02-0003
U02-0004
U02-0005

+1,1	TITLE	'OPR-B00,03/22/74,DWG702985'	U08-0001
*	INCLUDES	TECH. BULLETIN NO. 001 AND 003	U08-0002
+485	SIDR NO.	21930, FIX MONADIC MINUS FOR OVERFLOW	U08-0003
	STW,OP	OPER	U08-0004
		SAVE IN CASE OF OVERFLOW	U08-0005
+1887,1887	SIDR#20360	CORRECT MAX. FL. VAL. TO ALLOW FUZZING.	U08-0006
FLOTINF	DATA	X'7FFFFFFF',X'BFFFFFFF'	U08-0007
+END			U08-0008

+1,1	TITLE	'MIX-B00,07/09/74,DWG702985'	U09-0001
*	INCLUDES TECH. BULLETINS NO. 001,002, AND 005		U09-0002
+381,393	SIDR#20362	GENERATE INTG. VECT. FOR IOTA 1 OR 0	U09-0003
	BAL,LZ	INTSCALR	U09-0004
	LW,X	AI	U09-0005
	BLZ	ERDOMAIN	U09-0006
	LI,S	3	U09-0007
	LI,R	ISEQ**8+1	U09-0008
	LCW,T	ORGADJ	U09-0009
	LI,XL	1	U09-0010
	AI,AI	-1	U09-0011
	BGZ	MINDX1	U09-0012
	LI,R	INTG**8+1	U09-0013
	LI,S	2	U09-0014
	LW,T	ORIGIN	U09-0015
MINDX1	BAL,LX7	ALOCHNW	U09-0016
	STW,A	RESULT	U09-0017
	STH,R	*RESULT	U09-0018
	STW,X	2,A	U09-0019
	STW,T	3,A	U09-0020
	AI,X	-1	U09-0021
	BLEZ	*RETURN	U09-0022
+697	SIDR#20917	CHECK FOR CHAR. LEFT ARG. ON RESHAPE	U09-0023
	AI,T	-CHAR	U09-0025
	BEZ	ERDOMAIN	U09-0026
+2198	SIDR NO. 22622	YES-THATS A NO-NO	U09-0027
	CI,X	4	U09-0028
	BE	ERDOMAIN	U09-0029
		NO-ERROR EXIT	U09-0030
+END			U09-0031

+1,1	TITLE	'INDX-B00,07/09/74,DWG702985'	U10-0001	
*	INCLUDES TECH. BULLETINS NO. 001,003, AND 005		U10-0002	
			U10-0003	
+541,544	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANSU	U10-0004	
	CW,A	RTRANK	CHECK RANK	U10-0005
	BGE	MNOP	VECTOR OR SCALAR IS NOP	U10-0006
+545	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANSU	U10-0007	
	LW,S	RTRANK	GET RANK	U10-0008
	BAL,L2	INDEXB	ALLOCATE LOOP CONTROL BLOCK	U10-0009
+1073	SIDR	NO. 21930, FIX TAKE ERR ON (-2)*31	U10-0010	
	BLZ	ERDOMAIN	CATCH '80000000'	U10-0011
+1114,1114	SIDR	NO. 22620 FIX TAKE WITH PECULIAR ARG.	U10-0012	
17Z10	BLZ	ERDOMAIN	CATCH '80000000'	U10-0013
	STW,S	*DIMENPTR,X	TAKE:R(J)=ABS(A(J));STORE IT AS	U10-0014
+1684,1684	SIDR	NO. *****, ALLOW COORD SPEC. OF 1 WITH SCALAR	U10-0015	
	AI,K	-1	GREATER THAN 1	U10-0016
	BGZ	ERCOORD	MAY BE SPECIFIED	U10-0017
+END			U10-0018	

+1,1			U13-0001
	TITLE	'APLOUT-B00,07/09/74,DWG702985'	U13-0002
*	INCLUDES	TECH. BULLETINS NO. 002,003 AND 005	U13-0003
+574,575	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0004
CINTIM	LI,R2	9 PRESET COUNT	U13-0005
	LAW,R9	R8 GET ABS VALUE IN R9	U13-0006
	BLZ	SIZFOUND CATCH WEIRDO	U13-0007
+612,612	SIDR NO.	21930	U13-0008
+614	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0009
	LAW,R9	R8 GET ABS VALUE IN R9	U13-0010
	BLZ	SIZFOUND CATCH WEIRDO	U13-0011
+646,646	SIDR NO.	22624 FIX PRINT BUG ON NEG. UNDRFLO NO'S	U13-0012
	BGZ	POSVAL +	U13-0013
+649,649	SIDR NO.	22624 FIX PRINT BUG ON NEG. UNDRFLO NO'S	U13-0014
PLUSREAL	AI,R8	0 CHECK FOR UNDERFLOW	U13-0015
	BEZ	ZEROREAL YES	U13-0016
POSVAL	LW,R12	R5 GET SIGNIFICANCE	U13-0017
+659,659		FIX ROUND ERR IN DFMT FOR 0<N<1	U13-0018
	AI,R5	0 CHECK SPECIAL CASE (DFMT)	U13-0019
	BGZ	ADJUST NO	U13-0020
	LI,R5	1 YES, CHANGE TO 1 DIGIT	U13-0021
ADJUST	LD,R8	PRANGE-2,R5	U13-0022
+924,924	SIDR NO.	21051	U13-0023
+1077	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0025
	BLZ	WEIRDO (-2)*31	U13-0026
+1089	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0027
WEIRDO	LI,R6	10 SET 10 DIGITS	U13-0028
	B	SETDMAX	U13-0029
+1201	SIDRS NO.	21051 AND 22623	U13-0030
*			U13-0031
*	INVALID DATA TYPE		U13-0032
*			U13-0033
INVALID	CI,R2	6 CHECK FOR LIST	U13-0034
	BG	LASTROW NO-SKIP OUTPUT	U13-0035
	LI,R6	BA(LISTMSG)	U13-0036
	LI,R10	12 YES-OUTPUT MESSAGE	U13-0037
	LI,R13	LASTROW	U13-0038
	B	GENTEXTM	U13-0039
LISTMSG	TEXT	'**LIST ERR**'	U13-0040
+END			U13-0041


```

+1,1 U08-0001
      TITLE 'OPR-B00,11/16/73,DWG702985' U08-0002
* INCLUDES TECH. BULLETIN NO. 001 U08-0003
+1887,1887 SIDR#20360 CORRECT MAX. FL. VAL. TO ALLOW FUZZING. U08-0004
FLOTINF DATA X'7FFFFFFF',X'BFFFFFFF' U08-0005
+END U08-0006
+1,1 U08-0001
      TITLE 'OPR-B00,03/22/74,DWG702985' U08-0002
* INCLUDES TECH. BULLETIN NO. 001 AND 003 U08-0003
+485 SIDR NO. 21930, FIX MONADIC MINUS FOR OVERFLOW U08-0004
      STW,OP OPER SAVE IN CASE OF OVERFLOW U08-0005
+1887,1887 SIDR#20360 CORRECT MAX. FL. VAL. TO ALLOW FUZZING. U08-0006
FLOTINF DATA X'7FFFFFFF',X'BFFFFFFF' U08-0007
+END U08-0008
+1,1 U10-0001
      TITLE 'INDX-B00,11/16/73,DWG702985' U10-0002
* INCLUDES TECH. BULLETIN NO. 001 U10-0003
+541,544 SIDR#20361 CORRECT MEMORY MGMT. ERROR IN MTRANSU10-0004
      CW,A RTRANK CHECK RANK U10-0005
      BGE MNOP VECTOR OR SCALAR IS NOP U10-0006
+545 SIDR#20361 CORRECT MEMORY MGMT. ERROR IN MTRANSU10-0007
      LW,S RTRANK GET RANK U10-0008
      BAL,L2 INDEXB ALLOCATE LOOP CONTROL BLOCK U10-0009
+1,1 U09-0001
      TITLE 'MIX-B00,11/16/73,DWG702985' U09-0002
* INCLUDES TECH. BULLETIN MO. 001 U09-0003
+381,393 SIDR#20362 GENERATE INTG. VECT. FOR IOTA 1 OR 0 U09-0004
      BAL,LZ INTSCALR GET AND VALIDITY CHECK RIGHT ARG U09-0005
      LW,X AI SAVE N=LENGTH U09-0006
      BLZ ERDOMAIN QUIT IF NEGATIVE. U09-0007
      LI,S 3 SIZE FOR ISEQ DATA BLOCK U09-0008
      LI,R ISEQ**8+1 INDEX SEQUENCE VECTOR U09-0009
      LCW,T ORGADJ BASE=ORIGIN-1 U09-0010
      LI,XL 1 STEP=1 U09-0011
      AI,AI -1 CHECK FOR SPECIAL CASE U09-0012
      BGZ MINDX1 NO U09-0013
      LI,R INTG**8+1 YES-INTEGER VECTOR U09-0014
      LI,S 2 CHANGE DATA BLOCK SIZE U09-0015
      LW,T ORIGIN VALUE=ORIGIN U09-0016
MINDX1 BAL,LX7 ALOCHNW GET RESULT DATA BLOCK U09-0017
      STW,A RESULT U09-0018
      STH,R *RESULT SET TYPE-RANK U09-0019
      STW,X 2,A LENGTH U09-0020
      STW,T 3,A BASE (OR VALUE) U09-0021
      AI,X -1 CHECK IF SPECIAL CASE U09-0022
      STW,XL 4,A NO-SET STEP=1 U09-0024
+END U09-0025
!JOB
!LIMIT (TIME,3),(CORE,14)
!ASSIGN M:CI,(FILE,APLC08)
!ASSIGN M:SI,(FILE,APLU08)
!ASSIGN M:B0,(FILE,APLB08),(OUT),(SAVE)
!METASYM SI,CI,B0,LO,LU,DC
!JOB
!LIMIT (TIME,5)
!LOAD (LMN,APL),;
!(MAP),(SL,F),(PERM),(ABS),(BREF,60),;
!(EF,(:J0,:SYS),(APLB01),(APLB02),(APLB03),(APLB04),;
!(APLB05),(APLB06),(APLB07),(APLB08),(APLB09),;
!(APLB10),(APLB11),(APLB12),(APLB13),(APLB14),;
!(APLB15),(APLB16),(APLB17),(APLB18),(APLB19),;
!(APLB20),(APLB21),(APLB22),(APLB23),(APLB24),;
!(APLB25),(APLB26),(APLB27))
!TREE :J0-APLB03-APLB14-APLB27-APLB12-APLB13-;
! APLB23-APLB07-APLB08-APLB09-APLB10-APLB11-;
! APLB02-APLB05-APLB06-APLB19-;
! (APLB16-APLB17-APLB18,;
! APLB01-APLB04-APLB15-APLB24,;
! APLB20-APLB21-APLB22,;
! APLB25-APLB26)
+1,1 U02-0001
      TITLE 'CONSTS-B00,01/28/74,DWG702985' U02-0002
+185,185 U02-0003
APLSINON TEXTC 'APL 01/28/74' U02-0004
+END U02-0005
+1,1 U06-0001
      TITLE 'WMAQ-B00,01/28/74,DWG702985' U06-0002
* INCLUDES TECH. BULLETIN NO. 002 U06-0003

```

```

+678,678      ALLOW X'OD' AS EQUIV. TO X'15'                                U06-0004
+END                                                  U06-0006
+1,1                                                  U09-0001
*             TITLE      'MIX-B00,01/09/74,DWG702985'                    U09-0002
*             INCLUDES TECH. BULLETINS NO. 001 AND 002                    U09-0003
+381,393 SDR#20362      GENERATE INTG. VECT. FOR IOTA 1 OR 0U09-0004
      BAL,LZ      INTSCALR      GET AND VALIDITY CHECK RIGHT ARG      U09-0005
      LW,X        AI            SAVE N=LENGTH                        U09-0006
      BLZ ERDOMAIN      QUIT IF NEGATIVE.                            U09-0007
      LI,S          3           SIZE FOR ISEQ DATA BLOCK            U09-0008
      LI,R          ISEQ**8+1    INDEX SEQUENCE VECTOR              U09-0009
      LCW,T          ORGADJ      BASE=ORIGIN-1                      U09-0010
      LI,XL         1           STEP=1                              U09-0011
      AI,AI         -1          CHECK FOR SPECIAL CASE              U09-0012
      BGZ MINDX1      NO                                              U09-0013
      LI,R          INTG**8+1    YES-INTEGER VECTOR                U09-0014
      LI,S          2           CHANGE DATA BLOCK SIZE            U09-0015
      LW,T          ORIGIN      VALUE=ORIGIN                        U09-0016
MINDX1      BAL,LX7  ALOCHNW      GET RESULT DATA BLOCK            U09-0017
      STW,A          RESULT      U09-0018
      STH,R          *RESULT      SET TYPE-RANK                    U09-0019
      STW,X          2,A          LENGTH                            U09-0020
      STW,T          3,A          BASE (OR VALUE)                  U09-0021
      AI,X           -1          CHECK IF SPECIAL CASE              U09-0022
      BLEZ           *RETURN      YES-DONE                          U09-0023
      STW,XL         4,A          NO-SET STEP=1                    U09-0024
+697      SDR#20917      CHECK FOR CHAR. LEFT ARG. ON RESHAPEU09-0025
      AI,T           -CHAR      CHECK LEFT ARG FOR TEXT            U09-0026
      BEZ ERDOMAIN      YES-THATS A NO-NO                          U09-0027
+END                                                  U09-0028
+1,1                                                  U09-0001
*             TITLE      'MIX-B00,07/09/74,DWG702985'                    U09-0002
*             INCLUDES TECH. BULLETINS NO. 001,002, AND 005              U09-0003
+381,393 SDR#20362      GENERATE INTG. VECT. FOR IOTA 1 OR 0U09-0004
      BAL,LZ      INTSCALR      GET AND VALIDITY CHECK RIGHT ARG      U09-0005
      LW,X        AI            SAVE N=LENGTH                        U09-0006
      BLZ ERDOMAIN      QUIT IF NEGATIVE.                            U09-0007
      LI,S          3           SIZE FOR ISEQ DATA BLOCK            U09-0008
      LI,R          ISEQ**8+1    INDEX SEQUENCE VECTOR              U09-0009
      LCW,T          ORGADJ      BASE=ORIGIN-1                      U09-0010
      LI,XL         1           STEP=1                              U09-0011
      AI,AI         -1          CHECK FOR SPECIAL CASE              U09-0012
      BGZ MINDX1      NO                                              U09-0013
      LI,R          INTG**8+1    YES-INTEGER VECTOR                U09-0014
      LI,S          2           CHANGE DATA BLOCK SIZE            U09-0015
      LW,T          ORIGIN      VALUE=ORIGIN                        U09-0016
MINDX1      BAL,LX7  ALOCHNW      GET RESULT DATA BLOCK            U09-0017
      STH,R          *RESULT      SET TYPE-RANK                    U09-0019
      STW,X          2,A          LENGTH                            U09-0020
      STW,T          3,A          BASE (OR VALUE)                  U09-0021
      AI,X           -1          CHECK IF SPECIAL CASE              U09-0022
      BLEZ           *RETURN      YES-DONE                          U09-0023
      STW,XL         4,A          NO-SET STEP=1                    U09-0024
+697      SDR#20917      CHECK FOR CHAR. LEFT ARG. ON RESHAPE      U09-0025
      AI,T           -CHAR      CHECK LEFT ARG FOR TEXT            U09-0026
      BEZ ERDOMAIN      YES-THATS A NO-NO                          U09-0027
+2198      SDR NO. 22622      OBSCURE INNER PRODUCT BUG            U09-0028
      CI,X          4           CHECK IF IDENTITY ELEMENT EXISTS    U09-0029
      BE            ERDOMAIN      NO-ERROR EXIT                    U09-0030
+END                                                  U09-0031
+1,1                                                  U13-0001
*             TITLE      'APLOUT-B00,01/09/74,DWG702985'                U13-0002
*             INCLUDES TECH. BULLETIN NO. 002                            U13-0003
+924,924 SDR NO. 21051      U13-0004
      BG            INVALID      INVALID DATA TYPE                U13-0005
+1201      SDR NO. 21051      U13-0006
*                                                  U13-0007
* INVALID DATA TYPE                                              U13-0008
*                                                  U13-0009
      BG            SYSTERR      NO-CRASH                          U13-0011
      LI,R6         BA(LISTMSG)  U13-0012
      LI,R10        12           YES-OUTPUT MESSAGE              U13-0013
      LI,R13        LASTROW      U13-0014
      B             GENTEXTM      U13-0015
LISTMSG      TEXT      '***LIST ERR**'                            U13-0016
+END                                                  U13-0017
+1,1                                                  U14-0001
*             TITLE      'APLUTSI-B00,01/14/74,DWG702985'              U14-0002

```

*	INCLUDES TECH. BULLETIN MO. 002		U14-0003
+528	SIDR 21086	SET PLATEN TO 0,0	U14-0004
	LB,R1	X'2B'	U14-0005
	CI,R1	X'70'	U14-0006
	BL	CALSTRMB	U14-0007
		NO	U14-0008
CALSTRMP	CAL1,8	FPTSTRMP	U14-0009
+7078707	SIDR 21086	TURN UPPER CASE RESTRICT MODE OFF	U14-0010
	DATA	X'A0000000'	U14-0011
	DATA	X'04'	U14-0012
+708	SIDR 21086	SET PLATEN TO 0,0	U14-0013
FPTSTRMP	DATA	X'06200000'	U14-0014
	DATA	X'18000000'	U14-0015
	DATA	255,0	U14-0017
FDBLOCK	TEMP	48	U14-0018
+1608	SIDR 21085	FUNCTION DEF. MODE STORAGE BLOCK	U14-0019
	LW,R13	R10	U14-0020
	MI,R13	360	U14-0021
	AW,R11	R13	U14-0022
	AI,R7	-1	U14-0023
+1612,1612	SIDR 21085	CORRECT TIMODAY	U14-0024
	LB,R13	R9,R7	U14-0025
+1616,1616	SIDR 21085	CORRECT TIMODAY	U14-0026
T60S	DATA	216000,3600	U14-0027
+END			U15-0001
+1,1			U15-0002
	TITLE	'FUNDEF-B00,01/14/74,DWG702985'	U15-0003
*	INCLUDES TECH. BULLETIN NO. 002		U15-0004
+27	SIDR NO. 21109		U15-0005
	DEF FDSICLR	ENTRY FROM)SI CLEAR	U15-0006
+49	SIDR NO. 21109		U15-0007
	REF SICLR\$	CLEAR STATE INDICATOR	U15-0008
+116	SIDR 21050	ADD LINE COUNT CHECK	U15-0009
LNCHECK	BLOCK	1	U15-0010
+358	SIDR 21050	INDICATE LINE VALUE SET	U15-0011
	MTW,1	LNCHECK	U15-0012
	STW,R4	LNCHECK	U15-0013
+524	SIDR NO. 21109	(INDICATE NO LINE VALUE SET)	U15-0014
*		(UNLESS,PERHAPS, AN SI CLEAR	U15-0015
*		CMD. IS ISSUED. SEE 'FDSICLR'	U15-0016
+631	SIDR 21050	INDICATE LINE VALUE SET	U15-0017
	MTW,1	LNCHECK	U15-0018
+701	SIDR 21050	RESET EDIT CHECK FLAG	U15-0019
	STW,R1	EDITCK	U15-0020
+735,735	SIDR 21050	ADD CHECK OF DIRECTIVE COUNT	U15-0021
DELETE	AI,R9	0	U15-0022
	BLEZ	NXTPROMT	U15-0023
	LW,R12	LINVAL	U15-0024
+758,758	SIDR 21050	ADD CHECK IF LINE VALUE SET	U15-0025
DOPROMPT	LW,R12	LNCHECK	U15-0026
	BEZ	NXTPROMT	U15-0027
	LI,R12	INPF	U15-0028
+1212	SIDR NO. 21109	REQUEST FUNCTION INPUT AFTER	U15-0029
*			U15-0030
*	FDSICLR-)SI CLEAR	COMMAND EXECUTED IN FUNCTION DEFINITION MODE.	U15-0031
*		LINE CHAIN ENTRY MUST BE MOVED,NOT CLEARED.	U15-0032
*		OLDMODE AND GLOBLOC MAY REQUIRE CHANGING	U15-0033
*			U15-0034
*	EXIT IS TO REPRMPT.		U15-0036
*	MANY REGISTERS USED, NONE SAVED		U15-0037
*			U15-0038
FDSICLR	LW,R1	HILINE	U15-0039
	AI,R1	3	U15-0040
	LW,R8	TOPOSTAK	U15-0041
	BAL,R14	SICLR\$	U15-0042
	LW,R2	HILINE	U15-0043
	AI,R2	3	U15-0044
SICLOOP	AI,R2	-3	U15-0045
	AI,R1	-3	U15-0046
	LCI	3	U15-0047
	LM,R3	0,R2	U15-0048
	STM,R3	0,R1	U15-0049
	CW,R2	R8	U15-0050
	BG	SICLOOP	U15-0051
	STW,R1	TOPOSTAK	U15-0052
	LW,R2	STATEPTR	U15-0053
	AI,R2	-3	U15-0054
	STW,R2	HILINE	U15-0055
	LW,R2	GLOBLOC	U15-0056
		MOVE A LINE CHAIN ENTRY	
		CHECK IF DONE(R8=TOPOSTAK)	
		SET TO CLEAR BELOW LINE CHAIN	
		SAVE TOP OF STACK	
		CLEAR STACK(EXCEPT LINE-CHAIN)	
		SET LOOP TO MOVE LINE-CHAIN	
		SET NEW TOPOSTAK	
		SET NEW HILINE	
		GET GLOBLOC	

	BEZ	GLOBLOK	OK, NO CHANGE	U15-0057
	AW, R2	SYMT		U15-0059
GLOBLOK	STW, R2	GLOBLOC	GLOBLOC NOW PTS TO SYMT ENTRY	U15-0060
	LI, R2	1		U15-0061
	STW, R2	OLDMODE	SET OLDMODE DIRECT	U15-0062
	B	REXPROMPT	GO TO REXPROMPT	U15-0063
+END				U15-0064
+1, 1				U17-0001
	TITLE	'CMD-B00, 01/23/74, DWG702985'		U17-0002
*	INCLUDES	TECH. BULL. NO. 002		U17-0003
+976	SIDR NO.	21132		U17-0004
	LW, R8	BREAKFLG	CHECK FOR HANGUP	U17-0005
	CI, R8	-5		U17-0006
	BE	SETQUIT		U17-0007
+979, 979	SIDR NO.	21132		U17-0008
SETQUIT	MTW, -1	QUITFLAG	AVOID REPEATED ATTEMPTS	U17-0009
+END				U17-0010
+1, 1				U20-0001
	TITLE	'APLFIO-B00, 01/25/74, DWG702985'		U20-0002
*	INCLUDES	TECH. BULLETIN NO. 002		U20-0003
+1088	SIDR#21362	FIX TRAP ON 25-26-27		U20-0004
	LW, R3	RTARG	RESTORE RTARG POINTER-R3	U20-0005
+1340, 1340	SIDR#20885	CORRECT ERROR MESSAGES		U20-0006
	GEN, 1, 8, 70, X'75', X'01'	18		U20-0007
+1366	SIDR#20885	CORRECT ERROR MESSAGES		U20-0008
	GEN, 1, 8, 7 0, 0, 22	45		U20-0009
	GEN, 1, 8, 7 0, 0, 23	46		U20-0010
	GEN, 1, 8, 7 0, 0, 24	47		U20-0011
+1368, 1368	SIDR#20885	CORRECT ERROR MESSAGES		U20-0012
NCASES	EQU HA(\$)-HA(CASECODE)-1	# SP. CASES OF INTEREST.		U20-0013
+1417	SIDR#20885	CORRECT ERROR MESSAGES		U20-0014
	DATA, 1	IDFILTIE	45	U20-0015
	DATA, 1	IDFILTIE	46	U20-0016
	DATA, 1	IDFILTIE	47	U20-0017
+END				U20-0018
+1, 1				U24-0001
	TITLE	'SCMD-B00, 01/23/74, DWG702985'		U24-0002
*	INCLUDES	TECH. BULL. NO. 002		U24-0003
+40	SIDR NO.	21109		U24-0004
	REF FDSICLR		SI CLEAR IN FUNDEF MODE	U24-0005
+668, 669	SIDR NO.	21109		U24-0006
	LI, R14	INPDIR	SET RETURN	U24-0007
	LW, R8	MODE	TEST FOR FUNDEF MODE	U24-0008
	BNEZ	SICLR	NO-GOOD	U24-0009
	B	FDSICLR	YES-MORE WORK	U24-0010
+END				U24-0011
+1, 1				U10-0001
	INCLUDES	TECH. BULLETIN NO. 001 AND 003		U10-0003
+541, 544	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANS		U10-0004
	CW, A	RTRANK	CHECK RANK	U10-0005
	BCE	MNOP	VECTOR OR SCALAR IS NOP	U10-0006
+545	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANS		U10-0007
	LW, S	RTRANK	GET RANK	U10-0008
	BAL, L2	INDEXB	ALLOCATE LOOP CONTROL BLOCK	U10-0009
+1073	SIDR NO.	21930, FIX TAKE ERR ON (-2)*31		U10-0010
	BLZ	ERDOMAIN	CATCH '80000000'	U10-0011
+1684, 1684	SIDR NO.	*****, ALLOW COORD SPEC. OF 1 WITH SCALAR		U10-0012
	AI, K	-1	GREATER THAN 1	U10-0013
	BGZ	ERCOORD	MAY BE SPECIFIED	U10-0014
+END				U10-0015
+1, 1				U11-0001
	TITLE	'EVAL-B00, 03/22/74, DWG702985'		U11-0002
+154, 154	SIDR NO.	21931	FIX ERROR IN LAMINATION	U11-0003
	LI, AI	-2	NO, SET AI NEGATIVE (-2)	U11-0004
+163	SIDR NO.	21931	FIX ERROR IN LAMINATION	U11-0005
	MTW, 0	X	CHECK FOR NEGATIVE VALUE	U11-0006
	BGEZ	0, LX	NO, USE INTEGER PART	U11-0007
	AI, AI	-1	YES, ADJUST INTEGER VALUE	U11-0008
+END				U11-0009
	TITLE	'APLOUT-B00, 03/22/74, DWG702985'		U13-0002
*	INCLUDES	TECH. BULLETIN NO. 002 AND 003		U13-0003
+574, 575	SIDR NO.	21930, FIX '-0' PRINTOUT		U13-0004
CINTIM	LI, R2	9	PRESET COUNT	U13-0005
	LAW, R9	R8	GET ABS VALUE IN R9	U13-0006
	BLZ	SIZFOUND	CATCH WEIRDO	U13-0007
+612, 612	SIDR NO.	21930		U13-0008
+614	SIDR NO.	21930, FIX '-0' PRINTOUT		U13-0009
	LAW, R9	R8	GET ABS VALUE IN R9	U13-0010

	BLZ	SIZEFOUND	CATCH WEIRDO	U13-0011
+659,659		FIX ROUND ERR IN DFMT FOR 0<N<1		U13-0012
	AI,R5	0	CHECK SPECIAL CASE (DFMT)	U13-0013
	BGZ	ADJUST	NO	U13-0014
	LI,R5	1	YES, CHANGE TO 1 DIGIT	U13-0015
ADJUST	LD,R8	PRANGE-2,R5		U13-0016
+924,924	SIDR	NO. 21051		U13-0017
	BG	INVALID	INVALID DATA TYPE	U13-0018
+1077	SIDR NO.	21930, FIX '-0'	PRINTOUT	U13-0019
	BLZ	WEIRDO	(-2)*31	U13-0020
+1089	SIDR NO.	21930, FIX '-0'	PRINTOUT	U13-0021
WEIRDO	LI,R6	10	SET 10 DIGITS	U13-0022
	B	SETDMAX		U13-0023
*				U13-0025
*		INVALID DATA TYPE		U13-0026
*				U13-0027
INVALID	CI,R2	6	CHECK FOR LIST	U13-0028
	BG	SYSTERR	NO-CRASH	U13-0029
	LI,R6	BA(LISTMSG)		U13-0030
	LI,R10	12	YES-OUTPUT MESSAGE	U13-0031
	LI,R13	LASTROW		U13-0032
	B	GENTEXTM		U13-0033
LISTMSG	TEXT	'**LIST ERR**'		U13-0034
+END				U13-0035
+1,1				U17-0001
	TITLE	'CMD-B00,03/04/74,DWG702985'		U17-0002
*	INCLUDES	TECH. BULL. NO. 002 AND 003		U17-0003
+364,364	SIDR NO.	21594	FIX SAVE WITH AUTOSTART OPTION	U17-0004
+378,378	SIDR NO.	21594	FIX SAVE WITH AUTOSTART OPTION	U17-0005
SAVEREL	BAL,R8	GARBCOLL	GARBAGE COLLECT	U17-0006
	BAL,R11	RELEASER	RELEASE UNUSED MEMORY	U17-0007
+575	SIDR NO.	21595	FIX COPY PROBLEM IN GRAPHICS WORKSPACES	U17-0008
	LW,R5	HICOMMON	RESTORE	U17-0009
	LW,R5	-18,R5	GRAPHICS SPECIAL DATA BLOCK	U17-0010
	STW,R5	GRAFPNTR	POINTER (FOR FILE COPY)	U17-0011
	STW,R6	SAVE312	SET SAVE312 TO SKIP IN SNEWTABS	U17-0013
+976	SIDR NO.	21132		U17-0014
	LW,R8	BREAKFLG	CHECK FOR HANGUP	U17-0015
	CI,R8	-5		U17-0016
	BE	SETQUIT		U17-0017
+979,979	SIDR NO.	21132		U17-0018
SETQUIT	MTW,-1	QUITFLAG	AVOID REPEATED ATTEMPTS	U17-0019
+END				U17-0020
+1,1				U21-0001
	TITLE	'APLFMT-B00,03/22/74,DWG702985'		U21-0002
*	INCLUDES	TECH. BULLETIN NO 3		U21-0003
+217	SIDR NO.	21561	FIX FOR EMPTY DATA SETS	U21-0004
	LI,R12	0	INITIALIZE COUNT OF N BY 0 ITEMS	U21-0005
+223	SIDR NO.	21561	FIX FOR EMPTY DATA SETS	U21-0006
	AI,R9	0	CHECK FOR N BY 0 DATA SET	U21-0007
	BNEZ	FMT2A	NO	U21-0008
	AI,R12	1	YES-KICK COUNTER	U21-0009
FMT2A	RES	0		U21-0010
+232	SIDR NO.	21561	FIX FOR EMPTY DATA SETS	U21-0011
	AW,R1	R12	THROW IN ONE COLUMN PER N BY 0 ITEM	U21-0012
+861	SIDR NO.	21561	FIX FOR EMPTY DATA SETS	U21-0013
	BEZ	XFORMGEN	IF N BY 0 SET, DO BLANK INSERT	U21-0014
	LI,R2	0		U21-0016
+923	SIDR NO.	21930, FIX '-0'		U21-0017
	BLZ	CONVINT2	WEIRDO '-0'	U21-0018
+960,964	FIX	ROUNDOFF FOR 0<N<1		U21-0019
	BAL,R11	PLUSREAL	CONVERT	U21-0020
+967,968	FIX	ROUNDOFF FOR 0<N<1		U21-0021
	B	CONVRL5	YES	U21-0022
+END				U21-0023
+1,1				U26-0001
	TITLE	'GRAF-B00,03/31/74.DWG702985'		U26-0002
+969	SIDR NO.	21983	ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0003
*		CURRPSF AND LASTPSF ARE FLAGS INDICATING POSITIONS		U26-0004
*		OF CURRENT AND LAST POINTS IF THESE POINTS ARE OFF		U26-0005
*		SCALE: 1=LOW X, 2=HIGH X, 4=LOW Y, 8=HIGH Y		U26-0006
*				U26-0007
CURRPSF	TEMP			U26-0008
LASTPSF	TEMP			U26-0009
+997,997	SIDR NO.	21983	ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0010
*	SECTOR	CHECKING		U26-0011
+999,1014	SIDR NO.	21983	ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0012
	STW,LX	SNDPTRET	SAVE LINK	U26-0013

	LI,Z	0	PRESET SECTOR FLAG-IN LIMITS	U26-0014
	BCR,9	1Z7	IN LIMITS	U26-0016
	BCS,1	1Z2	LOW	U26-0017
	AI,Z	1	HIGH	U26-0018
1Z2	AI,Z	1		U26-0019
	CLM,Y	YLIMITS	CHECK Y	U26-0020
	BCR,9	1Z5	IN LIMITS	U26-0021
1Z3	BCS,1	1Z4	LOW	U26-0022
	AI,Z	4	HIGH	U26-0023
1Z4	AI,Z	4		U26-0024
1Z5	CI,F	RENTFLAG	NEW POINT OUT,CHECK IF REENTER	U26-0025
	BAZ	QUIT	NO-QUIT	U26-0026
	STW,Z	CURRPSF	YES-SAVE SECTOR FLAG	U26-0027
	CI,F	EDGEFLAG	CHECK IF OLD POINT ALSO OUT	U26-0028
	BAZ	OUTBOUND	NO-GOING OFFSCREEN	U26-0029
	AND,Z	LASTPSF	CHECK IF LINE IS ENTIRELY OFF	U26-0030
	BEZ	BOTHOUT	PROBABLY NO-TRY FURTHER	U26-0031
1Z6	STD,X	LASTPNT	OFFSCREEN-SAVE LAST POINT AND	U26-0032
1Z6P1	LW,Z	CURRPSF	SECTOR FLAG.	U26-0033
	STW,Z	LASTPSF		U26-0034
	OR,F	=EDGEFLAG	MAKE SURE EDGEFLAG IS SET	U26-0035
	B	*SNDPTRET	RETURN	U26-0036
1Z7	CLM,Y	YLIMITS	X IN LIMS,CHECK Y	U26-0037
	BCR,9	NEWIN	NEW POINT IS INSIDE	U26-0038
	B	1Z3	NEW POINT IS OUT	U26-0039
OUTBOUND	CI,F	VECTORFG	CHECK IF FIRST POINT	U26-0040
	BAZ	1Z6	YES-SAVE AND EXIT	U26-0041
1Z8	STW,Z	LASTPSF	NO-SAVE SECTOR FLAG-CONTROLS	U26-0042
*			DIRECTION OF INTERPOLATION.	U26-0043
	BAL,LX	INTERPOL	INTERPOLATE TO EDGE,OUTBOUND	U26-0044
	B	QUIT	THIS POINT SHOULD NOT BE REACHED	U26-0045
	BAL,LX	PLOT1	PLOT TO THE EDGE POINT	U26-0046
	AND,F	=-1-VECTORFG	START NEW VECTOR	U26-0047
	B	*SNDPTRET	RETURN	U26-0048
NEWIN	CI,F	EDGEFLAG	NEW PT ON SCREEN: HOW ABOUT LAST	U26-0049
+1017,1019	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0050
	BAL,LX	INTERPOL	INBOUND-INTERPOLATE TO EDGE	U26-0051
	B	QUIT	THIS POINT SHOULD NOT BE REACHED	U26-0052
	BAL,LX	PLOT1	PLOT POINT	U26-0053
	LW,LX	SNDPTRET	RESTORE LINK	U26-0054
+1021	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0055
BOTHOUT	STD,X	SAVEPNT	SAVE NEW OUTSIDE POINT	U26-0056
	BAL,LX	INTERPOL	INTERPOLATE FROM OLD POINT TO EDGE	U26-0057
	B	1Z6P1	LINE ENTIRELY OUTSIDE	U26-0058
	BAL,LX	PLOT	PLOT EDGE POINT-GOING IN	U26-0059
	LD,X	SAVEPNT		U26-0060
	LW,Z	CURRPSF	RESTORE NEW POINT AND SECTOR FLAG	U26-0061
+1024,1024	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0063
*		INTERPOL-INTERPOLATES TO AN EDGE OF WINDOW, USING CURRENT AND		U26-0064
*		LAST VALUES, AND LASTPSF TO DETERMINE THE DIRECTION		U26-0065
*		OF INTERPOLATION		U26-0066
*		IF RESULT NOT IN WINDOW, RETURNS TO 0,LX(SHOULD OCCUR ONLY WHEN		U26-0067
*		CALLED FROM 'BOTHOUT')		U26-0068
*		NORMAL RETURN TO 1,LX		U26-0069
				U26-0070
INTERPOL RES		0		U26-0071
+1026,1030	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0072
+1040,1051	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0073
	LI,Z	3	SET MASK	U26-0074
	AND,Z	LASTPSF	CHECK X	U26-0075
	BEZ	INTRPBT	IN RANGE-INTERP TO TOP-BOTTOM	U26-0076
	AI,Z	XLIMITS-1	OUT-GET EDGE VALUE	U26-0077
	LW,X	*Z		U26-0078
+1057,1071	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0079
	BLZ	INTRPBT	NOT IN WINDOW-TRY TOP-BOTTOM	U26-0080
	CW,Y	UPPERY		U26-0081
	BLE 1,LX		OK	U26-0082
INTRPBT	LW,Z	LASTPSF		U26-0083
	SLS,Z	-2	CHECK Y	U26-0084
	BEZ 0,LX		IN WINDOW-NO GOOD	U26-0086
	AI,Z	YLIMITS-1	GET BOTTOM OR TOP VALUE	U26-0087
	LW,Y	*Z		U26-0088
	LW,X	Y		U26-0089
+1076,1076	SIDR NO.	21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0090
	BLZ	0,LX	NOT IN WINDOW	U26-0091
	CW,X	UPPERX		U26-0092
	BLE	1,LX	OK	U26-0093
	B	0,LX		U26-0094

+END			U26-0095
+1,1			U02-0001
	TITLE	'CONSTS-B00,07/10/74,DWG702985'	U02-0002
+185,185			U02-0003
APLSINON	TEXTC	'APL B00 005'	U02-0004
+END			U02-0005
+1,1			U10-0001
	TITLE	'INDX-B00,07/09/74,DWG702985'	U10-0002
*	INCLUDES	TECH. BULLETINS NO. 001,003, AND 005	U10-0003
+541,544	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANSU	U10-0004
	CW,A	RTRANK	U10-0005
	BGE	MNOP	U10-0006
+545	SIDR#20361	CORRECT MEMORY MGMT. ERROR IN MTRANSU	U10-0007
	BAL,L2	INDEXB	U10-0009
+1073	SIDR	NO. 21930, FIX TAKE ERR ON (-2)*31	U10-0010
	BLZ	ERDOMAIN	U10-0011
+1114,1114	SIDR	NO. 22620 FIX TAKE WITH PECULIAR ARG.	U10-0012
17Z10	BLZ	ERDOMAIN	U10-0013
	STW,S	*DIMENPTR,X	U10-0014
+1684,1684	SIDR	NO. *****, ALLOW COORD SPEC. OF 1 WITH SCALAR	U10-0015
	AI,K	-1	U10-0016
	BGZ	ERCOORD	U10-0017
+END			U10-0018
+1,1			U11-0001
	TITLE	'EVAL-B00,07/09/74,DWG702985'	U11-0002
*	INCLUDES	TECH. BULLETINS 003 AND 005	U11-0003
+154,154	SIDR NO.	21931	U11-0004
	LI,AI	-2	U11-0005
+163	SIDR NO.	21931	U11-0006
	MTW,0	X	U11-0007
	BGEZ	0,LX	U11-0008
	AI,AI	-1	U11-0009
+172	SIDR NO.	22618	U11-0010
RESLIMS	DATA	FS'-1E15',FS'1E15'	U11-0011
+363	SIDR NO.	22618	U11-0012
	BCS,9	ERDOMAIN	U11-0014
+END			U11-0015
+1,1			U13-0001
	TITLE	'APLOUT-B00,07/09/74,DWG702985'	U13-0002
*	INCLUDES	TECH. BULLETINS NO. 002,003 AND 005	U13-0003
+574,575	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0004
CINTIM	LI,R2	9	U13-0005
	LAW,R9	R8	U13-0006
	BLZ	SIZFOUND	U13-0007
+612,612	SIDR NO.	21930	U13-0008
+614	SIDR NO.	21930, FIX '-0' PRINTOUT	U13-0009
	LAW,R9	R8	U13-0010
	BLZ	SIZFOUND	U13-0011
+646,646	SIDR NO.	22624	U13-0012
	BGZ	POSVAL	U13-0013
+649,649	SIDR NO.	22624	U13-0014
PLUSREAL	AI,R8	0	U13-0015
	BEZ	ZEROREAL	U13-0016
POSVAL	LW,R12	R5	U13-0017
+659,659			U13-0018
	AI,R5	0	U13-0019
	BGZ	ADJUST	U13-0020
ADJUST	LD,R8	PRANGE-2,R5	U13-0022
+924,924	SIDR	NO. 21051	U13-0023
	BG	INVALID	U13-0024
+1077	SIDR	NO. 21930, FIX '-0' PRINTOUT	U13-0025
	BLZ	WEIRDO	U13-0026
+1089	SIDR	NO. 21930, FIX '-0' PRINTOUT	U13-0027
WEIRDO	LI,R6	10	U13-0028
	B	SETDMAX	U13-0029
+1201	SIDRS	NO. 21051 AND 22623	U13-0030
*			U13-0031
*	INVALID	DATA TYPE	U13-0032
*			U13-0033
INVALID	CI,R2	6	U13-0034
	BG	LASTROW	U13-0035
	LI,R6	BA(LISTMSG)	U13-0036
	LI,R10	12	U13-0037
	LI,R13	LASTROW	U13-0038
	B	GENTEXTM	U13-0039
LISTMSG	TEXT	'**LIST ERR**'	U13-0040
+END			U13-0041
+1,1			U14-0001

	TITLE	'APLUTSI-B00,07/09/74,DWG702985'	U14-0002
+528	SIDR	21086 SET PLATEN TO 0,0	U14-0004
	LB,R1	X'2B'	U14-0005
	CI,R1	X'70'	U14-0006
	BL	CALSTRMB	U14-0007
		NO	U14-0008
CALSTRMP	CAL1,8	FPTSTRMP	U14-0009
		YES-SET PLATEN TO 0,0	U14-0010
+707,707	SIDR	21086 TURN UPPER CASE RESTRICT MODE OFF	U14-0011
	DATA	X'A0000000'	U14-0012
	DATA	X'04'	U14-0013
+708	SIDR	21086 SET PLATEN TO 0,0	U14-0014
FPTSTRMP	DATA	X'06200000'	U14-0015
	DATA	X'18000000'	U14-0016
	DATA	255,0	U14-0017
		PRESENCE BITS FOR PLATEN CONTROL	U14-0018
+755,755	SIDR	21050 INCREASE SIZE OF FDBLOCK (SEE FUNDEF,APLU15)	U14-0019
FDBLOCK	TEMP	48	U14-0020
		FUNCTION DEF. MODE STORAGE BLOCK	U14-0021
+969,969	SIDR	NO. 22619	U14-0022
		FIX BUG IN FILE LISTING OPERATION	U14-0023
	DATA	X'03000000'	U14-0024
		EMPTY NAME,3 BYTES OF ZEROS	U14-0025
+1608	SIDR	21085 CORRECT TIMODAY	U14-0026
	LW,R13	R10	U14-0027
		THOUSANDTHS OF MINUTES	U14-0028
	MI,R13	360	U14-0029
		CHANGE TO SAME BASE AS TMS	U14-0030
	AW,R11	R13	U14-0031
		AND ADD TO TMS	U14-0032
	AI,R7	-1	U14-0033
		(SKIP SECONDS)	U14-0034
+1612,1612	SIDR	21085 CORRECT TIMODAY	U14-0035
	LB,R13	R9,R7	U14-0036
		HOURS,MINUTES	U14-0037
+1616,1616	SIDR	21085 CORRECT TIMODAY	U14-0038
T60S	DATA	216000,3600	U14-0039
+END			U14-0040
+1,1			U15-0001
	TITLE	'FUNDEF-B00,07/09/74,DWG702985'	U15-0002
*	INCLUDES	TECH. BULLETINS NO. 002 AND 005	U15-0003
+27	SIDR NO.	21109	U15-0004
	DEF	FDSICLR	U15-0005
		ENTRY FROM)SI CLEAR	U15-0006
+49	SIDR NO.	21109	U15-0007
	REF	SICLRS	U15-0008
		CLEAR STATE INDICATOR	U15-0009
+68	SIDR NO.	22617	U15-0010
	REF	WIDTH	U15-0011
		LINE WIDTH	U15-0012
+116	SIDR	21050 ADD LINE COUNT CHECK	U15-0013
LNCHECK BLOCK	1	LINE COUNT CHECK	U15-0014
+358	SIDR	21050 INDICATE LINE VALUE SET	U15-0015
	MTW,1	LNCHECK	U15-0016
		INDICATE LINE VALUE HAS BEEN SET	U15-0017
+406	SIDR	21050 RESET LNCHECK	U15-0018
	STW,R4	LNCHECK	U15-0019
		(INDICATE NO LINE VALUE SET)	U15-0020
+524	SIDR NO.	21109	U15-0021
*			U15-0022
*		(UNLESS,PERHAPS, AN SI CLEAR	U15-0023
		CMD. IS ISSUED. SEE 'FDSICLR'	U15-0024
+631	SIDR	21050 INDICATE LINE VALUE SET	U15-0025
	MTW,1	LNCHECK	U15-0026
		INDICATE LINE VALUE HAS BEEN SET	U15-0027
	CW,R11	WIDTH	U15-0028
		CHECK VS WIDTH	U15-0029
	BG	ERDIR	U15-0030
		TOO BIG	U15-0031
+701	SIDR	21050 RESET EDIT CHECK FLAG	U15-0032
	STW,R1	EDITCK	U15-0033
		RESET EDIT CHECK FLAG	U15-0034
+735,735	SIDR	21050 ADD CHECK OF DIRECTIVE COUNT	U15-0035
DELETE	AI,R9	0	U15-0036
		CHECK DIRECTIVE COUNT	U15-0037
	BLEZ	NXTPROMT	U15-0038
		NO DIRECTIVE-NO DELETE	U15-0039
	LW,R12	LINVAL	U15-0040
		FIND THE VALUE IN THE LINE CHAIN	U15-0041
+758,758	SIDR	21050 ADD CHECK IF LINE VALUE SET	U15-0042
DOPROMPT	LW,R12	LNCHECK	U15-0043
		CHECK IF LINE VALUE HAS BEEN SET	U15-0044
	BEZ	NXTPROMT	U15-0045
		NO-USE NXTPROMT	U15-0046
	LI,R12	INPF	U15-0047
		REQUEST FUNCTION INPUT AFTER	U15-0048
+1212	SIDR NO.	21109	U15-0049
*			U15-0050
*			U15-0051
*		FDSICLR-)SI CLEAR COMMAND EXECUTED IN FUNCTION DEFINITION MODE.	U15-0052
*		LINE CHAIN ENTRY MUST BE MOVED,NOT CLEARED.	U15-0053
*		OLDMODE AND GLOBLOC MAY REQUIRE CHANGING	U15-0054
*			U15-0055
*		ENTRY FROM SICLEAR IN MODULE SCMD(SAME OVERLAY)	U15-0056
*		EXIT IS TO REPROMPT.	U15-0057
*		MANY REGISTERS USED, NONE SAVED	U15-0058
*			U15-0059
	AI,R1	3	U15-0060
		SET TO CLEAR BELOW LINE CHAIN	U15-0061
	LW,R8	TOPOSTAK	U15-0062
		SAVE TOP OF STACK	U15-0063
	BAL,R14	SICLR\$	U15-0064
		CLEAR STACK(EXCEPT LINE-CHAIN)	U15-0065
	LW,R2	HILINE	U15-0066
		SET LOOP TO MOVE LINE-CHAIN	U15-0067
	AI,R2	3	U15-0068
SICLOOP	AI,R2	-3	U15-0069
	AI,R1	-3	U15-0070
	LCI	3	U15-0071
	LM,R3	0,R2	U15-0072

	STM,R3	0,R1	MOVE A LINE CHAIN ENTRY	U15-0054
	CW,R2	R8	CHECK IF DONE(R8=TOPOSTAK)	U15-0055
	BG	SICLOOP		U15-0056
	STW,R1	TOPOSTAK	SET NEW TOPOSTAK	U15-0057
	LW,R2	STATEPTR		U15-0058
	AI,R2	-3	SET NEW HILINE	U15-0059
	STW,R2	HILINE		U15-0060
	LW,R2	GLOBLOC	GET GLOBLOC	U15-0061
	BEZ	GLOBLOC	OK,NO CHANGE	U15-0062
	LW,R2	FBLK+2	NAME PTR OFFSET	U15-0063
	AW,R2	SYMT		U15-0064
	STW,R2	GLOBLOC	GLOBLOC NOW PTS TO SYMT ENTRY	U15-0065
GLOBLOC	LI,R2	1		U15-0066
	B	REPRMPT	GO TO REPRMPT	U15-0068
+END				U15-0069
+1,1				U26-0001
	TITLE	'GRAF-B00,07/09/74.DWG702985'		U26-0002
*	INCLUDES	TECH. BULLETINS NO. 003 AND 005		U26-0003
+969	SIDR	NO. 21983,ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0004
*	CURRPSF	AND LASTPSF ARE FLAGS INDICATING POSITIONS		U26-0005
*		OF CURRENT AND LAST POINTS IF THESE POINTS ARE OFF		U26-0006
*		SCALE: 1=LOW X, 2=HIGH X, 4=LOW Y, 8=HIGH Y		U26-0007
*				U26-0008
CURRPSF	TEMP			U26-0009
LASTPSF	TEMP			U26-0010
+997,997	SIDR	NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0011
*	SECTOR	CHECKING		U26-0012
+999,1014	SIDR	NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0013
	STW,LX	SNDPTRET	SAVE LINK	U26-0014
	LI,Z	0	PRESET SECTOR FLAG-IN LIMITS	U26-0015
	CLM,X	XLIMITS	CHECK X	U26-0016
	BCR,9	1Z7	IN LIMITS	U26-0017
	BCS,1	1Z2	LOW	U26-0018
	AI,Z	1	HIGH	U26-0019
1Z2	AI,Z	1		U26-0020
	BCR,9	1Z5	IN LIMITS	U26-0022
1Z3	BCS,1	1Z4	LOW	U26-0023
	AI,Z	4	HIGH	U26-0024
1Z4	AI,Z	4		U26-0025
1Z5	CI,F	RENTFLAG	NEW POINT OUT,CHECK IF REENTER	U26-0026
	BAZ	QUIT	NO-QUIT	U26-0027
	STW,Z	CURRPSF	YES-SAVE SECTOR FLAG	U26-0028
	CI,F	EDGEFLAG	CHECK IF OLD POINT ALSO OUT	U26-0029
	BAZ	OUTBOUND	NO-GOING OFFSCREEN	U26-0030
	AND,Z	LASTPSF	CHECK IF LINE IS ENTIRELY OFF	U26-0031
	BEZ	BOTHOUT	PROBABLY NOT-TRY FURTHER	U26-0032
1Z6	STD,X	LASTPNT	OFFSCREEN-SAVE LAST POINT AND	U26-0033
1Z6P1	LW,Z	CURRPSF	SECTOR FLAG.	U26-0034
	STW,Z	LASTPSF		U26-0035
	OR,F	=EDGEFLAG	MAKE SURE EDGEFLAG IS SET	U26-0036
	B	*SNDPTRET	RETURN	U26-0037
1Z7	CLM,Y	YLIMITS	X IN LIMS,CHECK Y	U26-0038
	BCR,9	NEWIN	NEW POINT IS INSIDE	U26-0039
	B	1Z3	NEW POINT IS OUT	U26-0040
OUTBOUND	CI,F	VECTORFG	CHECK IF FIRST POINT	U26-0041
	BAZ	1Z6	YES-SAVE AND EXIT	U26-0042
1Z8	STW,Z	LASTPSF	NO-SAVE SECTOR FLAG-CONTROLS	U26-0043
	BAL,LX	INTERPOL	INTERPOLATE TO EDGE,OUTBOUND	U26-0045
	B	QUIT	THIS POINT SHOULD NOT BE REACHED	U26-0046
	BAL,LX	PLOT1	PLOT TO THE EDGE POINT	U26-0047
	AND,F	=-1-VECTORFG	START NEW VECTOR	U26-0048
	B	*SNDPTRET	RETURN	U26-0049
NEWIN	CI,F	EDGEFLAG	NEW PT ON SCREEN: HOW ABOUT LAST	U26-0050
+1017,1019	SIDR	NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0051
	BAL,LX	INTERPOL	INBOUND-INTERPOLATE TO EDGE	U26-0052
	B	QUIT	THIS POINT SHOULD NOT BE REACHED	U26-0053
	BAL,LX	PLOT1	PLOT POINT	U26-0054
	LW,LX	SNDPTRET	RESTORE LINK	U26-0055
+1021	SIDR	NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0056
BOTHOUT	STD,X	SAVEPNT	SAVE NEW OUTSIDE POINT	U26-0057
	BAL,LX	INTERPOL	INTERPOLATE FROM OLD POINT TO EDGE	U26-0058
	B	1Z6P1	LINE ENTIRELY OUTSIDE	U26-0059
	BAL,LX	PLOT	PLOT EDGE POINT-GOING IN	U26-0060
	LD,X	SAVEPNT		U26-0061
	LW,Z	CURRPSF	RESTORE NEW POINT AND SECTOR FLAG	U26-0062
	B	1Z8	FINISH WITH 'OUTBOUND' LOGIC.	U26-0063
+1024,1024	SIDR	NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS		U26-0064
*		INTERPOL-INTERPOLATES TO AN EDGE OF WINDOW, USING CURRENT AND		U26-0065

* LAST VALUES, AND LASTPSF TO DETERMINE THE DIRECTION	U26-0066
* IF RESULT NOT IN WINDOW, RETURNS TO 0,LX(SHOULD OCCUR ONLY WHEN	U26-0068
* CALLED FROM'BOTHOUT')	U26-0069
* NORMAL RETURN TO 1,LX	U26-0070
*	U26-0071
INTERPOL RES 0	U26-0072
+1026,1030 SIDR NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0073
+1040,1051 SIDR NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0074
LI,Z 3 SET MASK	U26-0075
AND,Z LASTPSF CHECK X	U26-0076
BEZ INTRPBT IN RANGE-INTERP TO TOP-BOTTOM	U26-0077
AI,Z XLIMITS-1 OUT-GET EDGE VALUE	U26-0078
LW,X *Z	U26-0079
+1057,1071 SIDR NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0080
BLZ INTRPBT NOT IN WINDOW-TRY TOP-BOTTOM	U26-0081
CW,Y UPPERY	U26-0082
BLE 1,LX OK	U26-0083
INTRPBT LW,Z LASTPSF	U26-0084
SLS,Z -2 CHECK Y	U26-0085
AI,Z 0	U26-0086
BEZ 0,LX IN WINDOW-NO GOOD	U26-0087
AI,Z YLIMITS-1 GET BOTTOM OR TOP VALUE	U26-0088
LW,Y *Z	U26-0089
LW,X Y	U26-0090
+1076,1076 SIDR NO. 21983, ADD 'CLIPPING' ALGORITHM TO GRAPHICS	U26-0091
BLZ 0,LX NOT IN WINDOW	U26-0092
CW,X UPPERX	U26-0093
BLE 1,LX OK	U26-0094
B 0,LX	U26-0095
+1535 SIDR NO. 22621 REMOVE TERM 13 RESTRICTION	U26-0096
* OR 'TERM 3' (TEKTRONIX 4010 TERMINAL)	U26-0097
+1538,1539 SIDR NO. 22621 REMOVE TERM 13 RESTRICTION	U26-0098
LW,R TERMTYPE CHECK TERMINAL TYPE	U26-0099
CI,R 13	U26-0100
BE 17Z1	U26-0101
CI,R 3	U26-0102
+END	U26-0103
+2,2	UBAS0001
* PLUS TECH. BULLETINS #001 AND #002	UBAS0002
+68,68 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG'	UBAS0003
M:S0 EQU F:EI SAVE, FILE, LOAD, CHAIN, RENUMBER	UBAS0004
* (DELETES M:S0 AS EXTERNAL,USES F:EI WHICH WAS ADDED FOR 'CATALOG')	UBAS0005
+532,532 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0006
+544,544 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG'	UBAS0007
DATA X'C0000409' ERROR,ABN,NEXTF,ACCT,FILE	UBAS0008
DATA ERR:FILE ERROR ADDRESS	UBAS0009
DATA ERR:FILE ABNORMAL ADDRESS	UBAS0010
+2181,2182 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0012
LI,R4 BA(PTITLE) SET TEXT POINTER	UBAS0013
BAL,R6 PRCAT PRINT OUT ACCOUNT	UBAS0014
MTW,7 MODEF DISABLE BREAK DURING OPEN-CLOSE	UBAS0015
+2186,2187 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0016
MTW,7 MODEF DISABLE BREAK DURING OPEN-CLOSE	UBAS0017
+2191 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0018
MTW,-7 MODEF RE-ENABLE BREAK	UBAS0019
MTW,0 BREAKF CHECK FOR UNSATISFIED BREAK	UBAS0020
BGZ REENTER YES-QUIT	UBAS0021
+2241,2242 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0022
LI,R4 BA(LINE) SET POINTER TO 'LINE'	UBAS0023
BAL,R6 PRCAT OUTPUT THE LINE	UBAS0024
+2251,2251 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0025
BE CLS YES,CLOSE	UBAS0026
+2263,2264 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0027
LI,R4 BA(RLINE) PRINT 'RESTRICTED' INFO	UBAS0028
BAL,R6 PRCAT OUTPUT MESSAGE	UBAS0029
+4603 SIG712376,TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0030
CI,R14 1 CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0031
BANZ 0C31 YES	UBAS0032
+4606,4606 SIG712376,TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0033
+4617 SIG712376 TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0035
CI,R14 1 CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0036
BANZ 0CV15WW YES	UBAS0037
+4658 SIG712376 TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0038
LI,R14 0 INDICATE E-FORMAT NOT FORCED	UBAS0039
+5744,5744 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0040
PRCAT1 LB,R2 0,R4 BYTE COUNT	UBAS0041
+5757 SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0042
* ENTRY TO 'CATALOG' PRINT	UBAS0043

PRCAT	OR,R6	X40	NEW LINE	UBAS0044
	STW,R6	8T5	SAVE LINK	UBAS0045
	B	PRCAT1	JOIN GENERAL TEXT OUTPUT ROUTINE.	UBAS0046
+6091,6091	SIDR	#21027	TECH. BULL. NO. 002 ERROR IN PRINTUSING	UBAS0047
MPR02	LW,R0	8T6	INDEX FOR INNER LOOP	UBAS0048
+6099,6099	SIDR#	21027	TECH. BULL. NO. 002 ERROR IN PRINTUSING	UBAS0049
\$MPUT3	BDR,R0	MPR03	END OF INNER LOOP	UBAS0050
+END				UBAS0051
+2,2				UBAS0001
*			PLUS TECH. BULLETINS #001, #002, AND #003	UBAS0002
+68,68	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG'	UBAS0003
M:S0	EQU F:EI		SAVE,FILE,LOAD,CHAIN,RENUMBER	UBAS0004
*			(DELETES M:S0 AS EXTERNAL,USES F:EI WHICH WAS ADDED FOR 'CATALOG')	UBAS0005
+544,544	SIG712359-21938		TECH BULLETINS 103 ERROR IN 'CATALOG'	UBAS0007
	DATA	X'C1000409'	ERROR,ABN,MODE,NEXTF,ACCT,FILE	UBAS0008
	DATA	ERR:FILE	ERROR ADDRESS	UBAS0009
	DATA	ERR:FILE	ABNORMAL ADDRESS	UBAS0010
	DATA	1	INPUT MODE	UBAS0011
+545,546	SIG-21938,TECH		BULLETIN #003 ERROR IN 'CATALOG'	UBAS0012
	DATA	X'01000101'		UBAS0013
	DATA	X'03000000'		UBAS0014
+556,558	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0015
+2181,2182	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0016
	LI,R4	BA(PTITLE)	SET TEXT POINTER	UBAS0017
	BAL,R6	PRCAT	PRINT OUT ACCOUNT	UBAS0018
	MTW,7	MODEF	DISABLE BREAK DURING OPEN-CLOSE	UBAS0019
+2186,2187	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0020
	MTW,7	MODEF	DISABLE BREAK DURING OPEN-CLOSE	UBAS0021
+2191	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0022
	MTW,-7	MODEF	RE-ENABLE BREAK	UBAS0023
	MTW,0	BREAKF	CHECK FOR UNSATISFIED BREAK	UBAS0024
	BGZ	REENTHER	YES-QUIT	UBAS0025
+2223,2223	SIG-21938,TECH		BULLETIN #003 ERROR IN 'CATALOG' COMMAND.	UBAS0026
	DW,R14	DAYD01		UBAS0027
+2241,2242	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0028
	BAL,R6	PRCAT	OUTPUT THE LINE	UBAS0030
+2251,2251	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0031
	BE	CLS	YES,CLOSE	UBAS0032
+2263,2264	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0033
	LI,R4	BA(RLINE)	PRINT 'RESTRICTED' INFO	UBAS0034
	BAL,R6	PRCAT	OUTPUT MESSAGE	UBAS0035
+4603	SIG712376,TECH.		BULLETIN #001 ERROR IN PRINTUSING	UBAS0036
	CI,R14	1	CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0037
	BANZ	OC31	YES	UBAS0038
+4606,4606	SIG712376,TECH.		BULLETIN #001 ERROR IN PRINTUSING	UBAS0039
	OC31	LI,R12	-6	FORCE POINT AFTER MOST SIG. DIGIT
+4617	SIG712376		TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0041
	CI,R14	1	CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0042
	BANZ	OCV15WW	YES	UBAS0043
+4658	SIG712376		TECH. BULLETIN #001 ERROR IN PRINTUSING	UBAS0044
	LI,R14	0	INDICATE E-FORMAT NOT FORCED	UBAS0045
+5744,5744	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0046
PRCAT1	LB,R2	0,R4	BYTE COUNT	UBAS0047
+5757	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0048
*			ENTRY TO 'CATALOG' PRINT	UBAS0049
PRCAT	OR,R6	X40	NEW LINE	UBAS0050
	STW,R6	8T5	SAVE LINK	UBAS0051
+6091,6091	SIDR	#21027	TECH. BULL. NO. 002 ERROR IN PRINTUSING	UBAS0053
MPR02	LW,R0	8T6	INDEX FOR INNER LOOP	UBAS0054
+6099,6099	SIDR#	21027	TECH. BULL. NO. 002 ERROR IN PRINTUSING	UBAS0055
\$MPUT3	BDR,R0	MPR03	END OF INNER LOOP	UBAS0056
+6755,6755	SIDR	#21509	TECH. BULL. #3 ERROR IN I/O ERROR 57	UBAS0057
NORADERR	BAL,R6	ENDCP08	CLEAR OUTPUT BUFFER (I/O 57)	UBAS0058
	BAL,R2	ERROR		UBAS0059
+11822,11823	SIDR	#20857	TECH BULL #3 UPDATE OF FILE IN ANOTHER ACCT	UBAS0060
+END				UBAS0061
+2,2				UBAS0001
*			PLUS TECH. BULLETINS #001, #002, #003, AND #004	UBAS0002
+65,65	SIDR12359		TECH BULL #004 DCB CHANGE	UBAS0003
	REF M:D0		STREAM 4 GET,PUT,INPUT,AND PRINT	UBAS0004
+67,67	SIDR12359TECH		BULL #004 DCB CHANGE	UBAS0005
+68,68	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG'	UBAS0006
M:S0	EQU	F:EI	SAVE,FILE,LOAD,CHAIN,RENUMBER	UBAS0007
*			(DELETES M:S0 AS EXTERNAL,USES F:EI WHICH WAS ADDED FOR 'CATALOG')	UBAS0008
+364,364	SIDR12359TECH		BULL #004 DCB CHANGE	UBAS0009
PRINTDCB	DATA	M:SL	.SHOWN (NO FILE ASSIGNMENT)	UBAS0010
+532,532	SIG712359,TECH.		BULLETIN #001 ERROR IN 'CATALOG' COMMAND.	UBAS0011
+544,544	SIG712359-21938		TECH BULLETINS 1&3 ERROR IN 'CATALOG'	UBAS0012

	DATA	X'C1000409'	ERROR,ABN,MODE,NEXTF,ACCT,FILE	UBAS0013
	DATA	ERR:FILE	ABNORMAL ADDRESS	UBAS0015
	DATA	1	INPUT MODE	UBAS0016
+545,546		SIG-21938, TECH BULLETIN #003 ERROR IN 'CATALOG'		UBAS0017
	DATA	X'01000101'		UBAS0018
	DATA	X'03000000'		UBAS0019
+556,558		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0020
+2181,2182		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0021
	LI,R4	BA(PTITLE)	SET TEXT POINTER	UBAS0022
	BAL,R6	PRCAT	PRINT OUT ACCOUNT	UBAS0023
	MTW,7	MODEF	DISABLE BREAK DURING OPEN-CLOSE	UBAS0024
+2186,2187		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0025
	MTW,7	MODEF	DISABLE BREAK DURING OPEN-CLOSE	UBAS0026
+2191		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0027
	MTW,-7	MODEF	RE-ENABLE BREAK	UBAS0028
	MTW,0	BREAKF	CHECK FOR UNSATISFIED BREAK	UBAS0029
	BGZ	REENTHER	YES-QUIT	UBAS0030
+2223,2223		SIG-21938, TECH BULLETIN #003 ERROR IN 'CATALOG' COMMAND.		UBAS0031
	DW,R14	DAYD01		UBAS0032
+2241,2242		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND		UBAS0033
	LI,R4	BA(LINE)	SET POINTER TO 'LINE'	UBAS0034
	BAL,R6	PRCAT	OUTPUT THE LINE	UBAS0035
+2251,2251		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0036
	BE	CLS	YES,CLOSE	UBAS0037
+2263,2264		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0038
	LI,R4	BA(RLINE)	PRINT 'RESTRICTED' INFO	UBAS0039
	BAL,R6	PRCAT	OUTPUT MESSAGE	UBAS0040
+2453,2453		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0041
		* FPTWRDO-WRITE RECORD ON M:SL -(OUTPUT ECHO OF AN INPUT LINE)		UBAS0042
+2455,2455		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0043
	FPTWRDO	GEN,8,7,17 X'11',0,M:SL		UBAS0044
+2480,2480		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0045
		* FPTDOWR-WRITE ON M:SL (DIAGNOSTIC ERROR)		UBAS0046
+2482,2482		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0047
	FPTDOWR	GEN,8,7,17 X'11',0,M:SL	WRITE M:SL	UBAS0048
+2488,2488		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0049
		* FPTDOALT-WRITE ON M:SL (ECHO ILLEGAL INPUT FROM M:SI)		UBAS0050
+2490,2490		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0051
	FPTDOALT	GEN,8,7,17 X'11',0,M:SL	WRITE M:SL	UBAS0052
+2571,2571		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0053
	FPTPAGE	GEN,8,7,17 4,0,M:SL		UBAS0054
+2882,2882		SIDR12359 TECH BULL#004 DCB CHANGE		UBAS0055
	CS,R8	M:SL+1	CHECK IF SAME DEVICE	UBAS0056
+4603		SIG712376,TECH. BULLETIN #001 ERROR IN PRINTUSING		UBAS0057
	CI,R14	1	CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0058
	BANZ	OC31	YES	UBAS0059
+4606,4606		SIG712376,TECH. BULLETIN #001 ERROR IN PRINTUSING		UBAS0060
+4617		SIG712376 TECH. BULLETIN #001 ERROR IN PRINTUSING		UBAS0062
	CI,R14	1	CHECK FOR FORCED SCIENTIFIC FORMAT	UBAS0063
	BANZ	OCV15WW	YES	UBAS0064
+4658		SIG712376 TECH. BULLETIN #001 ERROR IN PRINTUSING		UBAS0065
	LI,R14	0	INDICATE E-FORMAT NOT FORCED	UBAS0066
+5629,5629		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0067
	CI,R1	M:SL	CHECK IF NON-FILE	UBAS0068
+5744,5744		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0069
	PRCAT1	LB,R2 0,R4	BYTE COUNT	UBAS0070
+5757		SIG712359,TECH. BULLETIN #001 ERROR IN 'CATALOG' COMMAND.		UBAS0071
		* ENTRY TO 'CATALOG' PRINT		UBAS0072
	PRCAT	OR,R6 X40	NEW LINE	UBAS0073
		STW,R6 8T5	SAVE LINK	UBAS0074
	B	PRCAT1	JOIN GENERAL TEXT OUTPUT ROUTINE.	UBAS0075
+6091,6091		SIDR #21027 TECH. BULL. NO. 002 ERROR IN PRINTUSING		UBAS0076
	MPR02	LW,R0 8T6	INDEX FOR INNER LOOP	UBAS0077
+6099,6099		SIDR# 21027 TECH. BULL. NO. 002 ERROR IN PRINTUSING		UBAS0078
	\$MPUT3	BDR,R0 MPR03	END OF INNER LOOP	UBAS0079
+6669,6670		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0080
	DATA	M:DO	4	UBAS0081
	DATA	M:SL	PRINT ON ME	UBAS0082
+6755,6755		SIDR #21509 TECH. BULL. #3 ERROR IN I/O ERROR 57		UBAS0083
	BAL,R2	ERROR		UBAS0085
+6858,6858		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0086
	LI,R1	M:SL		UBAS0087
+6872,6872		SIDR12359 TECH BULL #004 DCB CHANGE		UBAS0088
	LI,R5	M:SL		UBAS0089
+9260		SIDR #21592 TECH BULL #4 \$STRING SYNTAX ERROR		UBAS0090
	LW,R5	*KPOINTER	CHECK LAST CHAR.	UBAS0091
	CI,R5	KCM	IF COMMA THEN	UBAS0092
	BE	\$TRERR	STRING ERROR	UBAS0093

+10853	SIDR# 22376	FOR NEXT ERROR	UBAS0094
	LW,R13 ONE@MODE	TEST EDIT OR COMPILE	UBAS0095
	BNEZ DR\$FORK	BRANCH FOR COMPILE	UBAS0096
+11822,11823	SIDR #20857 TECH BULL #3	UPDATE OF FILE IN ANOTHER ACCT	UBAS0097
+END			UBAS0098

