

```

000000          LST
000000          !          HED MATRIX ROM
000000          !! updated JFG March2020
000000          !! to match Matrix ROM 00087-15004 REV.A
000000          !! changes marked with !!
000000          !
000000          !*****
000000          !***      MATRIX ROM - 1      ***
000000          !*****
000000          !
000000          GLO GLOBAL
060000          ABS 60000
060000 260      BYT 260          !ROM NUMBER
060001 120      BYT 120          !INVERSE OF ROM #
060002          !
060002          !*****
060002          !***      SYSTEM TABLES      ***
060002          !*****
060002          !
060002 326 326      DEF RUNTIM          !RUNTIME TABLE
060004 326 326      DEF ASCIIS          !ASCII TABLE
060006 326 326      DEF BASICS          !PARSE TABLE
060010 326 326      DEF ERMSG          !ERROR MESSAGES
060012 326 326      DEF INIT          !INITIALIZATION
060014          !
060014          !*****
060014          !***      INITIALIZATION      ***
060014          !*****
060014          !
060014 230      INIT      BIN
060015 132 260 326      LDBD R32,=ROMFL          !GET ROMFL
060020 326
060021 366 336      JNZ INIOUT          !JIF NOT POWER ON
060023 251 326 326      LDM R32,=R60K          !SET START ADDR=24K
060026 316 326 326      JSB =RSUM8K          !SELF CHECKSUM
060031 367 336      JZR INIT+          !JIF CHECKSUM OK
060033          !!      JMP INIT+          !JIF CHECKSUM OK
060033 316 326 326      JSB =ERROR+
060036 316 326 326      JSB =ERROR
060041 014      BYT 12D
060042 236      RTN
060043 136 261 326  INIT+      LDMD R36,=FWBIN
060046 326
060047 263 326 326      STMD R36,=MBASE          !RAM BASE ADDRESS
060052 313 326 326      ADM R36,=RAMAMT          !RAM FOR MATRIX ROM
060055 263 326 326      STMD R36,=FWBIN          !RESET FWBIN
060060 263 326 326      STMD R36,=LWAMEM          !RESET LAST WORD AVAIL. MEMORY
060063 261 326 326      LDMD R36,=MBASE          !GET STOLEN RAM POINTER
060066 140 223      CLM R40
060070 213      DCM R40          !8 377'S
060071 036 266 326      STBD R40,X36,DETTAG          !TAG DETR TO NULL DATA
060074 326
060075 267 326 326      STMD R40,X36,ROWMIN          !TAG ROWMIN TO NULL DATA
060100      BSZ 0          ! " COLMIN " " "
060100      BSZ 0          ! " ROWMAX " " "
060100      BSZ 0          ! " COLMAX " " "
060100 267 326 326      STMD R40,X36,ROWMAB          ! " ROWMAB " " "
060103      BSZ 0          ! " COLMAB " " "
060103      BSZ 0          ! " RNORMX " " "
060103      BSZ 0          ! " CNORMX " " "
060103      SETROW      BSZ 0
060103 140 222      CLB R40
060105      SETCOL      BSZ 0
060105 141 223      CLM R41
060107 136 261 326      LDMD R36,=MBASE          !GET TEMP STORE PTR
060112 326
060113 140 036 266      STBD R40,X36,ROWFLG          !0 OUT ROWFLG
060116 326 326
060120 146 267 326      STMD R46,X36,ARYFLG          !0 OUT ARYFLG
060123 326
060124      BSZ 0          !0 OUT RDMFLG

```

060124	236		INIRTN	RTN		
060125	310	012	INIOUT	CMB R# ,=12	!DECOMPILING?	
060127	366	352		JNZ SETROW	!JIF NO	
060131	165	271	326	LDMI R65 ,=PTR1	!GET RUNTIME TOK, ROM# , ROM TOK	
060134	326					
060135	167	310	370	CMB R67 ,=370	!TALKING TO A ROM ?	
060140	366	362		JNZ INIRTN	!JIF NO	
060142	166	310	000	CMB R66 ,=ROM#	!TALKING TO ME?	
060145	366	355		JNZ INIRTN	!JIF NO	
060147	165	006	345	PUMD R65 ,+R6	!SAVE RUNTIME TOK, ROM# , ROM TOK	
060152	316	326	326	JSB =ROMJSB		
060155	326	326		DEF UNSTAK	!DECOMPILE TOKENS	
060157	000			BYT 0		
060160	177	250	000	LDB R77 ,=BLANK	!ASSUME 110 TOK (MAT PRINT A)	
060163	165	006	343	POMD R65 , -R6	!GET RUNTIME TOK, ROM# , ROM TOK	
060166	220			TSB R65	!TEST RUNTIME TOK (110 OR 111)	
060167	362	336		JOD INIPU	!IF (RUNTIME TOK = 111)	
060171	177	250	000	LDB R77 ,=SLASH	! MAT PRINT A/	
060174				BSZ 0	!END IF	
060174	130	213		DCM R30	!POINT TO DECOMPILE BUFFER	
060176	177	030	344	PUBD R77 ,+R30	!PUSH OUT SLASH OR BLANK	
060201	236			RTN		
060202				!		
060202				!*****		
060202				!*** ASCII TABLE ***		
060202				!*****		
060202				!		
060202	115	101	124	ASCIIS ASP "MAT READ"	!MAT READ	1
060205	040	122	105			
060210	101	304				
060212	115	101	124	ASP "MAT INPUT"	!MAT INPUT	2
060215	040	111	116			
060220	120	125	324			
060223	103	117	116	ASP "CONT"	!CONT	3
060226	324					
060227	111	116	326	ASP "INV"	!INV	4
060232	123	131	323	ASP "SYS"	!SYS	5
060235	124	122	316	ASP "TRN"	!TRN	6
060240	132	105	322	ASP "ZER"	!ZER	7
060243	103	117	316	ASP "CON"	!CON	10
060246	103	123	125	ASP "CSUM"	!CSUM	11
060251	315					
060252	122	123	125	ASP "RSUM"	!RSUM	12
060255	315					
060256	111	104	316	ASP "IDN"	!IDN	13
060261	103	122	117	ASP "CROSS"	!CROSS	14
060264	123	323				
060266	104	105	124	ASP "DETL"	!DETL	15
060271	314					
060272	104	105	324	ASP "DET"	!DET	16
060275	104	117	324	ASP "DOT"	!DOT	17
060300	114	102	116	ASP "LBND"	!LBND	20
060303	304					
060304	125	102	116	ASP "UBND"	!UDIM	21
060307	304					
060310	101	115	101	ASP "AMAXROW"	!AMAXROW	22
060313	130	122	117			
060316	327					
060317	101	115	101	ASP "AMAXCOL"	!AMAXCOL	23
060322	130	103	117			
060325	314					
060326	101	115	101	ASP "AMAX"	!AMAX	24
060331	330					
060332	101	115	111	ASP "AMINROW"	!AMINROW	25
060335	116	122	117			
060340	327					
060341	101	115	111	ASP "AMINCOL"	!AMINCOL	26
060344	116	103	117			
060347	314					
060350	101	115	111	ASP "AMIN"	!AMIN	27
060353	316					

060354	115	101	130	ASP "MAXABROW"	!MAXABROW	30
060357	101	102	122			
060362	117	327				
060364	115	101	130	ASP "MAXABCOL"	!MAXABCOL	31
060367	101	102	103			
060372	117	314				
060374	115	101	130	ASP "MAXAB"	!MAXAB	32
060377	101	302				
060401	101	102	123	ASP "ABSUM"	!ABSUM	33
060404	125	315				
060406			!	ASP "NORMAL"	!NORMAL	35
060406	106	116	117	ASP "FNORM"	!FNORM	34
060411	122	315				
060413	122	116	117	ASP "RNORMROW"	!RNORMROW	35
060416	122	115	122			
060421	117	327				
060423	122	116	117	ASP "RNORM"	!RNORM	36
060426	122	315				
060430	103	116	117	ASP "CNORMCOL"	!CNORMCOL	37
060433	122	115	103			
060436	117	314				
060440	103	116	117	ASP "CNORM"	!CNORM	40
060443	122	315				
060445	122	117	327	ASP "ROW"	!ROW	41
060450	103	117	314	ASP "COL"	!COL	42
060453	122	105	104	ASP "REDIM"	!REDIM	43
060456	111	315				
060460	123	125	315	ASP "SUM"	!SUM	44
060463	115	101	324	ASP "MAT"	!MAT	45
060466	105	122	122	ASP "ERROM"	!ERROM	46
060471	117	315				
060473	377			BYT 377	!	47
060474				!*****PARSER QUITs !SEARCHING HERE*****		
060474	252			BYT 252	!...A+()*B	50
060475	252			BYT 252	!()*A+B...	51
060476	252			BYT 252	!*	52
060477	253			BYT 253	!+	53
060500	275			BYT 275	!DUMMY=	54
060501	255			BYT 255	!-	55
060502	256			BYT 256	!.	56
060503	257			BYT 257	!/	57
060504	255			BYT 255	!MONADIC -	60
060505	275			BYT 275	!=	61
060506	124	122	316	ASP "TRN"	!TRN()	62
060511	103	117	316	ASP "CON"	!CON	63
060514	103	117	316	ASP "CON"	!CON()	64
060517	103	117	316	ASP "CON"	!CON(,)	65
060522	124	122	316	ASP "TRN"	!DUMMY TRN	66
060525	111	104	316	ASP "IDN"	!IDN	67
060530	111	104	316	ASP "IDN"	!IDN()	70
060533	111	104	316	ASP "IDN"	!IDN(,)	71
060536	200			BYT 200	!72	
060537	132	105	322	ASP "ZER"	!ZER	73
060542	132	105	322	ASP "ZER"	!ZER()	74
060545	132	105	322	ASP "ZER"	!ZER(,)	75
060550	075	250		BYT 75,250	!=()	76
060552	075	250		BYT 75,250	!=()*+-/B	77
060554	252			BYT 252	!TRN()*B	100
060555	252			BYT 252	!B*TRN()	101
060556	257			BYT 257	!SCALAR /	102
060557	255			BYT 255	!SCALAR -	103
060560	253			BYT 253	!SCALAR +	104
060561	252			BYT 252	!SCALAR *	105
060562	254			BYT 254	!COMMA	106
060563	247			BYT 247	!SEMI	107
060564	257			BYT 257	!SLASH	110
060565	240			BYT 240	!BLANK	111
060566	252			BYT 252	!INV()*B	112
060567	111	116	326	ASP "INV"	!DUMMY INV	113
060572	111	116	326	ASP "INV"	!INV()	114
060575	053	250		BYT 53,250	!()*A+()*B	115

060577	123 131 323	ASP "SYS"	!SYS	116
060602	103 123 125	ASP "CSUM"	!CSUM	117
060605	315			
060606	122 123 125	ASP "RSUM"	!RSUM	120
060611	315			
060612	103 122 117	ASP "CROSS"	!CROSS	121
060615	123 323			
060617	240	BYT 240	!DUMMY	122
060620	272	BYT 272	!C(I:J)	123
060621	!			
060621	!	*****		
060621	!	*** CONSTANTS ***		
060621	!	*****		
060621	!			
060621	ROM#	EQU 260		
060621	PIVK	EQU 0		
060621	PIVL	EQU 3		
060621	HR	EQU 6		
060621	MSIZE	EQU 11		
060621	DETR	EQU 13		
060621	PIVI	EQU 23		
060621	PIVJ	EQU 26		
060621	PIVOT	EQU 31		
060621	ROWFLG	EQU 41		
060621	ROWCTR	EQU 42		
060621	COLCTR	EQU 44		
060621	MAXCOL	EQU 46		
060621	MAXROW	EQU 50		
060621	ARYFLG	EQU 52		
060621	RDMFLG	EQU 53		
060621	DSPFLG	EQU 54		
060621	ROWMIN	EQU 55		
060621	COLMIN	EQU 57		
060621	ROWMAX	EQU 61		
060621	COLMAX	EQU 63		
060621	ROWMAB	EQU 65		
060621	COLMAB	EQU 67		
060621	RNORMX	EQU 71		
060621	CNORMX	EQU 73		
060621	TTYPB	EQU 75		
060621	TINCRB	EQU 76		
060621	ASTATS	EQU 101		
060621	RAMAMT	EQU 111	!RAM AMOUNT FOR THIS ROM	
060621	!			
060621	!			
060621	DETTAG	EQU 17		
060621	COLON	EQU 45		
060621	ZERO	EQU 0		
060621	ATOK	EQU 137		
060621	CDMTOK	EQU 63		
060621	CLNTOK	EQU 123		
060621	COLTOK	EQU 42		
060621	COMTOK	EQU 106		
060621	CRSTOK	EQU 121		
060621	CRTOK	EQU 354	!TOKENS	
060621	CSMTOK	EQU 117		
060621	CTOK	EQU 125		
060621	DETTOK	EQU 15		
060621	DSTOK	EQU 126		
060621	DUMTOK	EQU 66		
060621	EQU TOK	EQU 54		
060621	EOLTOK	EQU 354		
060621	EVNTOK	EQU 124		
060621	IDMTOK	EQU 67		
060621	IDNTOK	EQU 13		
060621	INPTOK	EQU 156		
060621	INVTOK	EQU 4		
060621	IVSTOK	EQU 114		
060621	I*BTOK	EQU 112		
060621	MULTOK	EQU 52		
060621	ODDTOK	EQU 125		

```

060621      PR TOK      EQU 153
060621      PRNTOK     EQU 155
060621      RDMTOK     EQU 153
060621      REDTOK     EQU 152
060621      ROWTOK     EQU 41
060621  260      BYT 260      !ROM NUMBER
060622      RSMTOK     EQU 120
060622      SCLTOK     EQU 77
060622      SCMTOK     EQU 105
060622      SLNTOK     EQU 50
060622      SMPTOK     EQU 51
060622      SPPTOK     EQU 115
060622      STOTOK     EQU 61
060622      SYSTOK     EQU 116
060622      TRNTOK     EQU 62
060622      TRTOK      EQU 6
060622      T*BTOK     EQU 100
060622      B*TTOK     EQU 101
060622      ULNTOK     EQU 347
060622      ULBTOK     EQU 345
060622      UMNTOK     EQU 60
060622      USTOK      EQU 313
060622      ZDMTOK     EQU 73
060622      !
060622      !*****
060622      !***      PARSE TABLE      ***
060622      !*****
060622      !
060622  326 326  BASICS  DEF DUMMY      !ILLEGAL      0
060624  326 326      DEF MATRED      !MAT READ      1
060626      !      INTERNAL ROM USE ONLY
060626  326 326      DEF MATINP      !MAT INPUT      2
060630  326 326      DEF SYCONT      !CONT           3
060632  326 326      DEF INV          !INV            4
060634  326 326      DEF SYS          !SYS            5
060636  326 326      DEF TRN          !TRN            6
060640  326 326      DEF ZER          !ZER            7
060642  326 326      DEF CON          !CON            10
060644  326 326      DEF CSUM         !CSUM           11
060646  326 326      DEF RSUM         !RSUM           12
060650  326 326      DEF IDN          !IDN            13
060652  326 326      DEF CROSS        !CROSS          14
060654  326 326      DEF DETL         !DETL           15
060656  326 326      DEF DET          !DET            16
060660  326 326      DEF DOT          !DOT            17
060662  326 326      DEF LDIM         !LDIM           20
060664  326 326      DEF UDIM         !UDIM           21
060666  326 326      DEF RAMAX        !RAMAX          22
060670  326 326      DEF CAMAX        !CAMAX          23
060672  326 326      DEF AMAX         !AMAX           24
060674  326 326      DEF RAMIN        !RAMIN          25
060676  326 326      DEF CAMIN        !CAMIN          26
060700  326 326      DEF AMIN         !AMIN           27
060702  326 326      DEF RMAXAB       !RMAXAB         30
060704  326 326      DEF CMAXAB       !CMAXAB         31
060706  326 326      DEF MAXAB        !MAXAB          32
060710  326 326      DEF ABSUM        !ABSUM          33
060712      !      DEF SYNORM         !NORMAL         35
060712  326 326      DEF NORM         !EUCLNORM       34
060714  326 326      DEF NORMRW       !RNORMROW       35
060716  326 326      DEF RNORM        !RNORM          36
060720  326 326      DEF NORMCL       !CNORMCOL       37
060722  326 326      DEF CNORM        !CNORM          40
060724  326 326      DEF ROW          !ROW            41
060726  326 326      DEF COL          !COL            42
060730  326 326      DEF REDIM        !REDIM          43
060732  326 326      DEF SUM          !SUM            44
060734      !
060734      !*****
060734      !***      RUNTIME TABLE      ***
060734      !*****

```

060734	!				
060734	RUNTIM	BSZ 0			
060734 326 326		DEF MAT	!MAT	45	
060736 326 326		DEF MREAD.	!MAT READ	1	
060740 326 326		DEF INPUT.	!MAT INPUT	2	
060742 326 326		DEF CONTR	!CONT	3	
060744 326 326		DEF OTHER	!INV	4	
060746 326 326		DEF OTHER	!SYS	5	
060750 326 326		DEF OTHER	!TRN	6	
060752 326 326		DEF OTHER	!ZER	7	
060754 326 326		DEF OTHER	!CON	10	
060756 326 326		DEF OTHER	!CSUM	11	
060760 326 326		DEF OTHER	!RSUM	12	
060762 326 326		DEF OTHER	!IDN	13	
060764 326 326		DEF OTHER	!CROSS	14	
060766 326 326		DEF DET.	!DETL	15	
060770 326 326		DEF DETA.	!DET (A)	16	
060772 326 326		DEF DOT.	!DOT	17	
060774 326 326		DEF LDIM10	!LDIM	20	
060776 326 326		DEF UDIM10	!UDIM	21	
061000 326 326		DEF RAMAX.	!AMAXROW	22	
061002 326 326		DEF CAMAX.	!AMAXCOL	23	
061004 326 326		DEF AMAX.	!AMAX	24	
061006 326 326		DEF RAMIN.	!AMINROW	25	
061010 326 326		DEF CAMIN.	!AMINCOL	26	
061012 326 326		DEF AMIN.	!AMIN	27	
061014 326 326		DEF RMXAB.	!MAXABROW	30	
061016 326 326		DEF CMXAB.	!MAXABCOL	31	
061020 326 326		DEF MAXAB.	!MAXAB	32	
061022 326 326		DEF ABSUM.	!ABSUM	33	
061024	!	DEF NORMR	!NORMAL	35	
061024 326 326		DEF NORM.	!NORM	34	
061026 326 326		DEF NORMR.	!RNORMROW	35	
061030 326 326		DEF RNORM.	!RNORM	36	
061032 326 326		DEF NORMC.	!CNORMCOL	37	
061034 326 326		DEF CNORM.	!CNORM	40	
061036 326 326		DEF ROW.	!M PRT ROW	41	
061040 326 326		DEF COL.	!M PRT COL	42	
061042 326 326		DEF RDIM.	!REDIM	43	
061044 326 326		DEF SUM.	!SUM	44	
061046 326 326		DEF MAT.	!MAT	45	
061050 326 326		DEF EROM.	!ERROM	46	
061052 326 326		DEF DUMMY	!DUMMY	47	
061054 326 326		DEF LIN.	!...A+()*B	50	
061056 326 326		DEF SCLDUM	!()*A+B...	51	
061060 326 326		DEF MATMUL	!MAT A*B	52	
061062 326 326		DEF MATADD	!MAT A+B	53	
061064 326 326		DEF DUMMY=	!DUMMY =	54	
061066 326 326		DEF MATSUB	!MAT A-B	55	
061070 326 326		DEF PER.	!MAT A.B	56	
061072 326 326		DEF SLASH.	!MAT A/B	57	
061074 326 326		DEF UMIN.	!UNARY -	60	
061076 326 326		DEF A.	!MAT C=A	61	
061100 326 326		DEF TRN10	!TRN (A)	62	
061102 326 326		DEF CON1	!CON	63	
061104 326 326		DEF CDIM1V	!CON (M)	64	
061106 326 326		DEF CDIM2V	!CON (M,N)	65	
061110 326 326		DEF DUMTRN	!TRN NOP	66	
061112 326 326		DEF IDN10	!IDN	67	
061114 326 326		DEF IDIM1V	!IDN (M)	70	
061116 326 326		DEF IDIM2V	!IDN (M,N)	71	
061120 326 326		DEF INCOM.	!MAT INPUT	72	
061122 326 326		DEF ZER.	!ZER	73	
061124 326 326		DEF ZDIM1V	!ZER (M)	74	
061126 326 326		DEF ZDIM2V	!ZER (M,N)	75	
061130 326 326		DEF SCL.	!A=()	76	
061132 326 326		DEF SCLDUM	!A=()*+/-/B	77	
061134 326 326		DEF TRN*A.	!TRN()*A	100	
061136 326 326		DEF B*TRN.	!B*TRN()	101	
061140 326 326		DEF SCLDIV	!(C)/A	102	
061142 326 326		DEF SCLSUB	!(C)-A	103	

061144	326	326	DEF SCLADD	! (C)+A	104
061146	326	326	DEF SCLMUL	! (C)*A	105
061150	326	326	DEF COM.	!MAT PRT A,	106
061152	326	326	DEF SEM.	!MAT PRT A;	107
061154	326	326	DEF SLSH.	!MAT PRT A/	110
061156	326	326	DEF COMM.	!MAT PRT A	111
061160	326	326	DEF INV*B.	!INV()*B	112
061162	326	326	DEF DUMINV	!INV NOP	113
061164	326	326	DEF INV10	!INV(A)	114
061166	326	326	DEF SCLDUM	!()*A+()*B	115
061170	326	326	DEF SYS10	!SYS	116
061172	326	326	DEF CSUM.	!CSUM	117
061174	326	326	DEF RSUM.	!RSUM	120
061176	326	326	DEF CROSS.	!CROSS	121
061200	326	326	DEF DUMMY	!DUMMY	122
061202	326	326	DEF COLON.	!C(I:J,K:L)	123
061204	326	326	DEF ATTR32	!TWODIM NOP	124
061206	326	326	DEF ATTR32	!ONEDIM NOP	125
061210	326	326	DEF CI.	!C(I)	126
061212	326	326	DEF CIJ.	!C(I:J)	127
061214	326	326	DEF CIJ*.	!C(I:J,)	130
061216	326	326	DEF CIJK.	!C(I:J,K)	131
061220	326	326	DEF CIJKL.	!C(I:J,K:L)	132
061222	326	326	DEF CK.	!C(,K)	133
061224	326	326	DEF CKL.	!C(,K:L)	134
061226	326	326	DEF CI*.	!C(I,)	135
061230	326	326	DEF CIK.	!C(I,K)	136
061232	326	326	DEF CIKL.	!C(I,K:L)	137
061234	326	326	DEF AI.	!A(I)	140
061236	326	326	DEF AIJ.	!A(I:J)	141
061240	326	326	DEF AIJ*.	!A(I:J,)	142
061242	326	326	DEF AIJK.	!A(I:J,K)	143
061244	326	326	DEF AIJKL.	!A(I:J,K:L)	144
061246	326	326	DEF AK.	!A(,K)	145
061250	326	326	DEF AKL.	!A(,K:L)	146
061252	326	326	DEF AI*.	!A(I,)	147
061254	326	326	DEF AIK.	!A(I,K)	150
061256	326	326	DEF AIKL.	!A(I,K:L)	151
061260	326	326	DEF READ.	!MATREAD	152
061262	326	326	DEF RDIM1	!REDIM1V	153
061264	326	326	DEF RDIM2	!REDIM2V	154
061266	326	326	DEF SCLPRN	!()	155
061270	326	326	DEF INPUN.	!MAT INPUT	156
061272			!***** MATREAD,REDIM,CONT, RUNTIME *****		
061272			!***** ATTRIBUTES !TABLE *****		
061272	241		BYT 241	!ROM BASIC STMT	
061273			!*****		
061273			MREAD.	BSZ 0	
061273			RDIM.	BSZ 0	
061273			CONTR	BSZ 0	
061273	236		RTN		
061274			!***** SCALAR, COLON RUNTIME *****		
061274	000	051	BYT 0,51		
061276			!*****		
061276			COLON.	BSZ 0	
061276	236		SCLDUM	RTN	!JUST GET ATTRIBUTES
061277			!***** SCLPRN, ATTR32 RUNTIME *****		
061277	032		BYT 32		
061300			!*****		
061300			ATTR32	BSZ 0	
061300	236		SCLPRN	RTN	!JUST GET ATTRIBUTES
061301			!***** CONT !PARSETIME *****		
061301	114	250	110 SYCONT	LDB R14,=110	!SET UP CONT TOKEN
061304	316	326	326 SYSCOM	JSB =ROMJSB	
061307	326	326		DEF G01N	
061311	000			BYT 0	
061312	360	336		JMP REXIT	
061314			!***** NORMAL !PARSETIME *****		
061314			!*SYNORM	LDB R14,=175	!SET UP NORMAL TOKEN
061314			!*	JMP SYSCOM	
061314			!***** DUMMY !TRN,INV RUNTIME *****		

061314	024	055		BYT	24,55
061316			!	*****	
061316			DUMINV	BSZ	0
061316	236		DUMTRN	RTN	



061317		!	HED MAT PARSE		
061317	316	326	326	MAT JSB =PUSH-	!PUSH MAT ROM TOKEN
061322	316	326	326	JSB =SCANM	
061325	114	310	153	CMB R14,=PRTOK	
061330	367	336		JZR PRINT	
061332	310	126		CMB R14,=DSTOK	
061334	367	336		JZR PRINT	!JIF DISPLAY TOKEN
061336	104	251	377	GTO NAR	
061341	377				
061342	272	326	326	PRINT STBI R#,=PTR2-	!PUSH PRINT TOKEN
061345	132	222		CLB R32	!FLAG PLAIN PRINT
061347	121	010	244	LDBD R21,R10	!GET NEXT CHARACTER
061352	120	311	125	CMM R20,=125,123	!SEE IF HAVE "US"
061355	123				
061356	367	336		JZR USING	!JIF YES - DEMAND USING TOKEN
061360	311	165	163	CMM R20,=165,163	!SEE IF HAVE "us"
061363	366	336		JNZ PLIST	!JIF NO - CAN'T BE "USING" OR "using"
061365				BSZ 0	
061365	316	326	326	USING JSB =SCANM	!NOW DEMAND USING TOKEN
061370	114	310	313	CMB R14,=USTOK	!PRINT USING?
061373	366	336		JNZ ERR89	!JIF NO
061375	024	242		STB R14,R24	!SAVE USING TOK
061377	250	347		LDB R14,=ULNTOK	!USING LINE # TOKEN
061401	121	250	345	LDB R21,=ULBTOK	!USING LINE LABEL TOKEN
061404	132	006	344	PUBD R32,+R6	!SAVE PRINT FLAG
061407	316	326	326	JSB =ROMJSB	
061412	326	326		DEF GLN/LB	!GET THE LINE # OR LINE TOKEN
061414	000			BYT 0	
061415	132	006	342	POBD R32,-R6	!RESTORE PRINT FLAG
061420	370	336		JEN USEX1	!JIF FOUND LINE #
061422	102	270	326	LDBI R2,=PTR2+	!ELSE TRASH TOKEN
061425	326				
061426	316	326	326	JSB =ROMJSB	!GET IN-LINE USING
061431	326	326		DEF STREXP	
061433	000			BYT 0	
061434	370	336		JEN USEX1-	!JIF THERE
061436	316	326	326	ERR89 JSB =ERROR	
061441	131			BYT 89D	
061442	360	336		REXIT JMP EOPRT+	
061444	124	272	326	USEX1- STBI R24,=PTR2-	!PUSH USING
061447	326				
061450	132	210		USEX1 ICB R32	!FLAG PRINT USING
061452	114	310	000	CMB R14,=SEMI	!IS THERE A PLIST?
061455	366	336		JNZ TSTOUT	!JIF NO, DONE
061457	272	326	326	STBI R14,=PTR2-	!PUSH SEMI TOKEN
061462	316	326	326	PLIST JSB =SMYROM	!SEARCH MY ROM-ROW,COL
061465	371	336		JEZ PLIST+	!JIF NOTHING FOUND
061467	143	310	041	CMB R43,=ROWTOK	!ROW TOKEN?
061472	367	336		JZR TOKOK	!JIF YES
061474	310	042		CMB R43,=COLTOK	!COL TOKEN?
061476	366	336		JNZ ERR89	!JIF NO - ERROR
061500	316	326	326	TOKOK JSB =PUSH-	!PUSH ROW OR COL TOK
061503	316	326	326	JSB =NARE+	!SCAN & DEMAND ARRAY
061506	371	336		JEZ EOPRT+	!ERROR IF NO ARRAY
061510	360	336		JMP LODCOM	!GO LOAD COMMA TOKEN
061512	316	326	326	PLIST+ JSB =SCANM	!SCAN PAST SEMI OR ROW,COL TOK
061515	114	310	001	CMB R14,=1	
061520	366	336		JNZ TSTOUT	!PUSH EOL IF NO ARRAY
061522	316	326	326	JSB =NARRE!	!CHECK FOR ARRAY
061525	153	250	106	LODCOM LDB R53,=COMTOK	
061530	114	310	000	CMB R14,=COMMA	!COMMA?
061533	367	336		JZR DONE	!JIF YES
061535	153	210		ICB R53	!READY TO CHECK FOR SEMI
061537	114	310	000	CMB R14,=SEMI	
061542	367	336		JZR DONE	!JIF SEMI FOUND
061544	153	210		ICB R53	!READY TO CHECK FOR SLASH
061546	114	310	000	CMB R14,=SLASH	
061551	367	336		JZR DONE	
061553	153	210		ICB R53	!MAKE IT A BLANK
061555	316	326	326	JSB =PSHROM	
061560	316	326	326	EOPRT+ JSB =ROMEX	!AND RETURN

061563	316	326	326	DONE	JSB =PSHROM	
061566	120	310	000		CMB R20,=CR	!CR TOKEN?
061571	367	336			JZR TSTRTN	!JIF YES.
061573	310	000			CMB R20,=BANG	! TOKEN?
061575	367	336			JZR TSTRTN	!JIF YES.
061577	310	100			CMB R20,=100	!@ TOKEN?
061601	366	257			JNZ PLIST	!JIF NO.
061603	316	326	326	TSTRTN	JSB =SCANM	!SCAN PAST , SEMI OR
061606	360	350			JMP EOPRT+	
061610	153	250	354	TSTOUT	LDB R53,=EOLTOK	
061613	272	326	326		STBI R53,=PTR2-	
061616	360	340			JMP EOPRT+	
061620	136	250	156	MATINP	LDB R36,=INPTOK	
061623	360	336			JMP MREAD	
061625	136	250	152	MATRED	LDB R36,=REDTOK	
061630	316	326	326	MREAD	JSB =PUSH-	!PUSH MAT READ TOK.
061633	316	326	326	RLIST	JSB =NARE+	!DEMAND AN ARRAY.
061636	371	320			JEZ EOPRT+	!JIF NO MORE ARRAYS
061640	316	326	326		JSB =PUSH--	!PUSH READ TOKEN
061643	114	310	000		CMB R14,=COMMA	!COMMA?
061646	367	363			JZR RLIST	!JIF YES, MORE ARRAYS
061650	360	306			JMP EOPRT+	
061652	114	310	002	NAR	CMB R14,=2	!TEST FOR C (
061655	366	336			JNZ EQCK	!JIF C=
061657	316	326	326		JSB =SPSH45	
061662	136	250	125		LDB R36,=CTOK	!C(I:J) CASE TOKEN
061665	316	326	326		JSB =SUBARY	
061670	114	310	000		CMB R14,=EQUALS	!MUST BE =.
061673	366	336			JNZ GERR89	!ERROR IF NOT THERE.
061675	316	326	326		JSB =SCANM	!GET PAST =.
061700	114	310	002		CMB R14,=2	!A(?)
061703	366	336			JNZ GETA	
061705	104	251	377		GTO APUSH	
061710	377					
061711	316	326	326	GETA	JSB =NARRE!	!DEMAND ARRAY A
061714	104	251	377		GTO STORE	!GO WIND UP
061717	377					
061720				!* LEFT SIDE		
061720	316	326	326	EQCK	JSB =NARRE!	!DEMAND ARRAY C
061723	114	310	000		CMB R14,=EQUALS	!MUST BE EQUAL
061726	366	336			JNZ GERR89	!ERROR
061730				!* RIGHT SIDE		
061730	316	326	326		JSB =SCANM	
061733	114	310	370		CMB R14,=370	!ROM TOKEN
061736	367	336			JZR NAR1	!JIF YES
061740	104	251	377		GTO MATEXP	
061743	377					
061744	141	310	260	NAR1	CMB R41,=ROM#	!MY ROM?
061747	367	336			JZR CONT	
061751	104	251	035	GERR89	GTO ERR89	
061754	143					
061755	230			CONT	BIN	!SET MODE.
061756	143	310	004		CMB R43,=INVTOK	!TOK<4 ILLEGAL.
061761	372	366			JNC GERR89	!JIF < 4.
061763	310	015			CMB R43,=DETTOK	!TOK>15 ILLEGAL HERE.
061765	373	362			JCY GERR89	
061767	132	223			CLM R32	
061771	043	240			LDB R32,R43	
061773	204				LLB R32	!TOKEN * 2
061774	032	265	222		LDMD R32,X32,BASICS	!ADDR
061777	141					
062000	213				DCM R32	!ADJUST FOR GOTO
062001	004	243			STM R32,R4	!GO THERE
062003	316	326	326	REDIM	JSB =PUSH-	!PUSH IT OUT
062006	316	326	326	ALOOP	JSB =SCANM	
062011	114	310	002		CMB R14,=2	!DEMAND ARRAY VAR
062014	366	336			JNZ ER74JM	
062016	316	326	326		JSB =SPSH45	
062021	132	250	153		LDB R32,=RDMTOK	!TOKEN FOR REDIM 1 VAR
062024	316	326	326		JSB =SUBSCR	
062027	371	336			JEZ ER74JM	

062031	114	310	000		CMB R14,=COMMA	!ANY MORE ARRAYS?
062034	367	350			JZR ALOOP	
062036	360	336			JMP LVROM	!EXIT
062040	316	326	326	SUBSCR	JSB =NVAL	!LOG EXP-1ST SUBSCRIPT
062043	371	336			JEZ ER74EX	!ERR IF NO LOGICAL EXP
062045	114	310	000		CMB R14,=COMMA	!2 VAR?
062050	366	336			JNZ RTPAR	!JIF 1 VAR
062052	132	210			ICB R32	!TOK FOR REDIM 2 VAR=36
062054	316	326	326		JSB =NVA+	!GET 2ND LOGICAL EXP
062057	371	336			JEZ ER74EX	!DEMAND 2ND VAR
062061	114	310	000	RTPAR	CMB R14,=CLOSE	!DEMAND RT PAREN
062064	235				CLE	
062065	366	336			JNZ ER74EX	
062067	316	326	326		JSB =SCANM	!GET BY IT
062072	032				ARP R32	
062073	316	326	326		JSB =PUSH	!PUSH OUT REDIM TOK
062076	235				CLE	
062077	234				ICE	!NO ERROR FLAG
062100	236			ER74EX	RTN	!EXIT SUBSCR SUBROUTINE
062101	360	246		ER74JM	JMP GERR89	!PASS IT ON!
062103	143	250	117	CSUM	LDB R43,=CSMTOK	!CSUM TOKEN
062106	360	336			JMP SUM	
062110	143	250	120	RSUM	LDB R43,=RSMTOK	!RSUM TOKEN
062113				AMAX	BSZ 0	
062113				AMIN	BSZ 0	
062113				MAXAB	BSZ 0	
062113				DET	BSZ 0	
062113				NORM	BSZ 0	
062113				CNORM	BSZ 0	
062113				RNORM	BSZ 0	
062113				ABSUM	BSZ 0	
062113	143	036	242	SUM	STB R43,R36	!SAVE 43
062116	316	326	326		JSB =GTOPN	
062121	360	336			JMP FUN1A	
062123	143	250	116	SYS	LDB R43,=SYSTOK	!SYS TOKEN
062126	360	336			JMP DOT	
062130	143	250	121	CROSS	LDB R43,=CRSTOK	!CROSS TOKEN
062133	143	036	242	DOT	STB R43,R36	!SAVE 43
062136	316	326	326		JSB =GTOPN	
062141	316	326	326		JSB =NARRE!	
062144	114	310	000		CMB R14,=COMMA	
062147	366	336			JNZ ER74	!NEED 1 MORE
062151	316	326	326	FUN1A-	JSB =SCANM	!CHEW UP COMMA
062154	316	326	326	FUN1A	JSB =NARRE!	!GET NUMERIC ARRAY
062157	153	036	240		LDB R53,R36	
062162	316	326	326	PSH)CK	JSB =PSHRM-	!PUSH IT AND DUMMY=
062165	114	310	000		CMB R14,=CLOSE	!FIND )?
062170	366	336			JNZ ER74	!JIF NO
062172	316	326	326	SCARTN	JSB =SCANM	!GET PAST IT
062175	360	336		LVROM	JMP PRM--	!DONE
062177	316	326	326	PSHRM-	JSB =PSHRM	!PUSH TOKEN
062202	153	250	054	PSHDM=	LDB R53,=EQUOTOK	!DUMMY =
062205	316	326	326		JSB =PSHROM	!PUSH DUMMY =
062210	236				RTN	!DONE
062211	132	250	063	CON	LDB R32,=CDMTOK	!REDIM BEFORE CON TOKEN
062214	360	336			JMP ZER+	
062216	132	250	067	IDN	LDB R32,=IDMTOK	!REDIM BEFORE IDN TOKEN
062221	360	336			JMP ZER+	
062223	132	250	073	ZER	LDB R32,=ZDMTOK	!REDIM BEFORE ZER TOKEN
062226	132	006	344	ZER+	PUBD R32,+R6	!SAVE R32 TOKEN
062231	316	326	326		JSB =SCANM	
062234	114	310	000		CMB R14,=OPEN	!REDIM FIRST?
062237	366	336			JNZ PRM-	!JIF NO
062241	132	210			ICB R32	!NEXT LARGER TOKEN
062243	316	326	326		JSB =SCANM	!GET PAST (.
062246	170	006	342		POBD R70,-R6	!CLEAN UP STACK
062251	316	040	144		JSB =SUBSCR	!GO REDIM ARRAY
062254	371	336			JEZ ER74	!JIF ERROR
062256	316	202	144	PDUM=	JSB =PSHDM=	!PUSH DUMMY =
062261	360	336			JMP PRM--	
062263	136	006	343	ER74++	POMD R36,-R6	!THROW AWAY RTN

062266	104	251	035	ER74	GTO ERR89	
062271	143					
062272	153	006	342	PRM-	POBD R53,-R6	
062275	316	177	144	PSH&EX	JSB =PSHRM-	!PUSH TOKEN & DUMMY =
062300	316	326	326	PRM--	JSB =ROMEX	
062303	316	326	326	GTOPN	JSB =SCANM	
062306	114	310	000		CMB R14,=OPEN	!MUST BE OPEN
062311	366	350			JNZ ER74++	!JIF NOT OPEN
062313	316	326	326		JSB =SCANM	!ELSE CHEW IT UP
062316	236				RTN	
062317	316	326	326	MONAD-	JSB =NARE+	!SCAN, GET NEXT
062322	153	250	060		LDB R53,=UMNTOK	
062325	360	346			JMP PSH&EX	!PUSH & EXIT.
062327	316	326	326	SCALAR	JSB =GETSCL	!GET SCALAR,CK FOR ).
062332	153	250	077		LDB R53,=SCLTOK	!DECOMP TOK (DUMMY RUNTIME)
062335	136	250	105		LDB R36,=SCMTOK	!()*A RUNTIME TOKEN
062340	114	310	000		CMB R14,=STAR	!(SCL)*A?
062343	367	336			JZR LINEAR	!JIF YES
062345	137	250	051		LDB R37,=51	!PLUS TOKEN LESS 2
062350	136	212		OPLOOP	DCB R36	!()+-/A TOKS IN ORDER
062352	137	210			ICB R37	!PLUS,MINUS,SLASH
062354	210				ICB R37	!IN ORDER
062355	014	300			CMB R37,R14	!ANY MATCH YET?
062357	367	336			JZR SCLOP-	!JIF YES
062361	310	000			CMB R37,=SLASH	!ALL BEEN CONSIDERED?
062363	366	363			JNZ OPLOOP	!LOOP IF NO
062365	153	212			DCB R53	!A=(NUM EXP)
062367	316	326	326		JSB =PSHROM	!PUSH OUT TOKEN
062372	153	250	155		LDB R53,=PRNTOK	!SCALAR PAREN TOKEN
062375	360	336			JMP PSHEX	!GO PUSH AND EXIT
062377	114	310	000	MATEXP	CMB R14,=OPEN	!(?)
062402	367	323			JZR SCALAR	!JIF YES.
062404	310	000			CMB R#,=MINUS	!MONADIC MINUS?
062406	367	307			JZR MONAD-	!JIF YES.
062410	310	002			CMB R14,=2	
062412	366	336			JNZ GETARY	!JIF NO.
062414	316	326	326	APUSH	JSB =SPSH45	
062417	136	250	137		LDB R36,=ATOK	!OFFSET FOR ARRAY A
062422	316	326	326		JSB =SUBARY	
062425	360	227			JMP PDUM=	
062427	316	326	326	LINEAR	JSB =SCLOP	!PUSH PRNTOK, GET ARRAY
062432	114	310	000		CMB R14,=PLUS	!()*A OR ()*A+()*B?
062435	366	336			JNZ SCLOP+	!JIF ()*A
062437	153	250	051		LDB R53,=SMPTOK	!DUMMY * OP FOR 1ST TERM
062442	316	326	326		JSB =PSHROM	!PUSH OUT TOKEN
062445	316	326	326		JSB =SCANM	!GET PAST +
062450	114	310	000		CMB R14,=OPEN	!( ?
062453	366	211			JNZ ER74	!ERROR IF NO
062455	316	326	326		JSB =GETSCL	!GET SCALAR,CK FOR )
062460	153	250	115		LDB R53,=SPPTOK	!+ ( TOKEN
062463	136	250	050		LDB R36,=SLNTOK	!()*A+()*B RUNTIME TOKEN
062466	316	326	326	SCLOP-	JSB =SCLOP	!PUSH PRNTOK, GET ARRAY
062471	153	036	240	SCLOP+	LDB R53,R36	!LOAD UP RUNTIME TOKEN
062474	360	336			JMP PSHEX	!PUSH IT OUT & EXIT
062476	316	326	326	SCLOP	JSB =PSHROM	!PUSH SCLDUM TOK-DECOMP
062501	153	250	155		LDB R53,=PRNTOK	!SCALAR PAREN TOKEN
062504	316	326	326		JSB =PSHROM	!PUSH OUT TOKEN
062507	316	326	326	NARE+	JSB =ROMJSB	!GET NUMERIC ARRAY REF
062512	326	326			DEF NARRE+	
062514	000				BYT 0	
062515	236				RTN	
062516	316	326	326	GETARY	JSB =NARRE!	!NUM ARRAY REF-NO SCAN
062521	114	310	000		CMB R14,=PLUS	!A+B?
062524	367	336			JZR MATEX	!JIF YES.
062526	310	000			CMB R14,=MINUS	!A-B?
062530	367	336			JZR MATEX	!JIF YES.
062532	310	000			CMB R14,=SLASH	!A/B?
062534	367	336			JZR MATEX	!JIF YES.
062536	310	000			CMB R14,=PERIOD	!A.B?
062540	367	336			JZR MATEX	!JIF YES.
062542	310	000			CMB R14,=STAR	!A*B OR A*TRN(B)?

062544	367	336		JZR TRNCK	!JIF MAYBE.
062546	153	250	061	STORE LDB R53,=STOTOK	!PUSH MAT STORE TOKEN
062551	316	326	326	PSHEX JSB =PSHROM	!TO OUTPUT (MAT A=B).
062554	360	336		JMP EXROM	!DONE.
062556	114	310	370	TOKCK CMB R14,=370	!A ROM?
062561	366	336		JNZ GTO89	!ERROR IF NO.
062563	141	310	260	CMB R41,=ROM#	!MINE?
062566	366	336		JNZ GTO89	!ERROR IF NO.
062570	143	310	006	CMB R43,=TRTOK	!PARSING TRN?
062573	366	336		JNZ GTO89	!ERROR IF NO.
062575	316	303	144	JSB =GTOPN	!SEE IF ( IS NEXT.
062600	316	326	326	JSB =NARRE!	!DEMAND ARRAY VAR.
062603	114	310	000	CMB R14,=CLOSE	!FIND ) ?
062606	366	336		JNZ GTO89	!ERROR IF NO.
062610	316	326	326	JSB =SCANM	!GET PAST ).
062613	153	250	066	LDB R53,=DUMTOK	!TRN NOP TOK.
062616	316	326	326	JSB =PSHROM	!PUSH IT OUT.
062621	153	250	101	LDB R53,=B*TTOK	!B*TRN() TOKEN.
062624	360	336		JMP MATEX+	!PUSH IT OUT & EXIT.
062626	316	303	144	TRN JSB =GTOPN	!DEMAND (.
062631	316	326	326	JSB =TRNINV	!DEMAND ARRAY).
062634	153	250	062	LDB R53,=TRNTOK	!IN CASE DOING A=TRN().
062637	114	310	000	CMB R14,=STAR	!* ?
062642	366	336		JNZ MATEX+	!JIF DOING A=TRN().
062644	153	250	066	LDB R53,=DUMTOK	!ELSE DOING A=TRN()*B.
062647	316	326	326	JSB =PSHROM	!PUSH OUT TOKEN.
062652	114	250	100	LDB R14,=T*BTOK	!A=TRN()*B TOKEN.
062655	114	006	344	MATEX PUBD R14,+R6	!SAVE TOKEN
062660	316	107	145	JSB =NARE+	
062663	153	006	342	POBD R53,-R6	!RESTORE TOK TO 53
062666	316	177	144	MATEX+ JSB =PSHRM-	
062671	360	336		EXROM JMP ROMEX+	
062673	104	251	035	GTO89 GTO ERR89	
062676	143				
062677	136	006	343	ROMEX POMD R36,-R6	!TRASH RETURN
062702	104	251	377	ROMEX+ GTO ROMRTN	
062705	377				
062706	316	326	326	TRNINV JSB =NARRE!	!GET ARRAY VAR.
062711	360	336		JMP SCL+	!CONTINUE
062713	316	326	326	TRNCK JSB =SCANM	!GET PAST *.
062716	114	310	001	CMB R14,=1	!ARRAY VAR?
062721	366	233		JNZ TOKCK	!JIF NO.
062723	316	326	326	JSB =NARRE!	!DEMAND ARRAY VAR.
062726	153	250	052	LDB R53,=MULTOK	!STAR-2.
062731	360	333		JMP MATEX+	!GO FINISH EXPRESSION.
062733	316	303	144	INV JSB =GTOPN	!DEMAND ).
062736	316	306	145	JSB =TRNINV	!DEMAND ARRAY).
062741	153	250	114	LDB R53,=IVSTOK	!IN CASE DOING A=INV().
062744	114	310	000	CMB R14,=STAR	!* ?
062747	366	315		JNZ MATEX+	!JIF DOING A=INV().
062751	153	212		DCB R53	!DUMINV TOKEN.
062753	316	326	326	JSB =PSHROM	!PUSH IT OUT.
062756	114	250	112	LDB R14,=I*BTOK	!A=INV()*B TOKEN.
062761	360	272		JMP MATEX	!GO WIND UP.
062763	316	326	326	GETSCL JSB =NVA+	!GET SCALAR
062766	371	336		JEZ INPERR	!ERROR IF NOT THERE
062770	114	310	000	SCL+ CMB R14,=CLOSE	!) ?
062773	366	336		JNZ INPERR	!ERROR IF NOT THERE
062775	316	326	326	JSB =SCANM	!GET BY )
063000	236			RTN	
063001	316	326	326	NARRE! JSB =ROMJSB	!CALL NARREF
063004	326	326		DEF NARREF	
063006	000			BYT 0	
063007	236			RTN	
063010	316	326	326	SUBARY JSB =NVAL	!DEMAND NUM EXP
063013	370	336		JEN GOTNEX	!JIF C(I OR A(I
063015	114	310	000	CMB R14,=COMMA	!TEST FOR C(, OR A(,
063020	366	336		JNZ INPERR	!ERR IF NOT (I OR (,
063022	316	326	326	JSB =SCANM	!GET PAST ,
063025	114	310	000	CMB R14,=CLOSE	!A(,)?
063030	367	336		JZR INPERR	!ERROR IF A(,,)

063032	134	251	000		LDM R34,=0,6	!NULL STRING
063035	006					
063036	273	326	326		STMI R34,=PTR2-	
063041	136	312	005		ADB R36,=5	!TOKEN OFFSET
063044	136	210		CMCK++	ICB R36	!TOKEN +4 OR 9
063046	316	326	326		JSB =NVAL	!DEMAND NUM EXP
063051	371	336			JEZ INPERR	!ERROR IF NOT THERE
063053	114	310	000		CMB R14,=CLOSE	!C(I:J,K) OR A(I:J,K)?
063056	367	336			JZR EVENCK	!JIF YES
063060	310	045			CMB R14,=COLON	!MUST BE C(I:J,K: OR A...
063062	366	336			JNZ INPERR	!ERROR IF NOT
063064	136	210			ICB R36	!TOKEN +5 OR 10
063066	316	326	326		JSB =NVA+	!DEMAND NUM EXP
063071	371	336			JEZ INPERR	!ERROR IF NONE
063073	153	250	123		LDB R53,=CLNTOK	!COLON TOKEN
063076	316	326	326		JSB =PSHROM	!PUSH IT OUT
063101	114	310	000		CMB R14,=CLOSE	!C(I:J,K:L) OR A...?
063104	366	336			JNZ INPERR	!ERROR IF NO
063106	360	336			JMP EVENCK	
063110	210			GOTNEX	ICB R#	!TOKEN+1
063111	114	310	000		CMB R14,=CLOSE	!C(I) OR A(I)?
063114	366	336			JNZ CKCOL	!JIF NO )
063116	153	250	125		LDB R53,=ODDTOK	!ONE DIM ATTRIBUTES
063121	360	336			JMP THRU-	!DONE
063123	136	210		CKCOL	ICB R36	!TOKEN+2
063125	114	310	000		CMB R14,=COMMA	!C(I, OR A(I,?
063130	366	336			JNZ CKCLN	!JIF NO ,
063132	136	312	005		ADB R36,=5	!TOKEN+7
063135	360	336			JMP TWODIM	
063137	114	310	045	CKCLN	CMB R14,=COLON	!C(I: OR A(I:?
063142	366	336			JNZ INPERR	!JIF NO
063144	316	326	326	PSHCLN	JSB =NVA+	!DEMAND NUM EXP
063147	371	336			JEZ INPERR	!ERROR IF NOT THERE
063151	153	250	123		LDB R53,=CLNTOK	!COLON TOKEN
063154	316	326	326		JSB =PSHROM	!PUSH IT OUT
063157	114	310	000		CMB R14,=CLOSE	!C(I:J) OR A(I:J)?
063162	366	336			JNZ TWODIM	!JIF NO
063164	360	336			JMP THRU	!GO FINISH UP
063166	104	251	262	INPERR	GTO ER74++	
063171	144					
063172	136	210		TWODIM	ICB R36	!TOKEN+3
063174	114	310	000		CMB R14,=COMMA	!C(I:J, OR A(I:J,?
063177	366	365			JNZ INPERR	!ERROR IF NO
063201	316	326	326		JSB =SCANM	!GET BY ,
063204	114	310	000		CMB R14,=CLOSE	!C(I:J,) OR A(I:J,)?
063207	366	233			JNZ CMCK++	!JIF NO
063211	134	251	000		LDM R34,=0,6	!NULL STRING
063214	006					
063215	273	326	326		STMI R34,=PTR2-	
063220	136	220		EVENCK	TSB R36	!TOKEN ALREADY EVEN?
063222	363	336			JEV THRU	!JIF YES
063224	153	250	124		LDB R53,=EVNTOK	!TWO DIM ATTRIBUTES
063227	316	326	326	THRU-	JSB =PSHROM	!PUSH IT OUT
063232	316	326	326	THRU	JSB =SCANM	!GET PAST )
063235	036			PUSH--	ARP R36	
063236	360	336			JMP PUSH	
063240	043			PUSH-	ARP R43	
063241	153	240		PUSH	LDB R53,R#	!MOVE TOKEN
063243	151	250	370	PSHROM	LDB R51,=370	!ROM TOKEN
063246	272	326	326		STBI R51,=PTR2-	!PUSH ROM TOKEN
063251	250	260			LDB R51,=ROM#	!ROM NUMBER
063253	272	326	326		STBI R51,=PTR2-	!PUSH ROM #
063256	153	272	326		STBI R53,=PTR2-	!PUSH TOKEN
063261	326					
063262	236				RTN	
063263	316	326	326	SCANM	JSB =ROMJSB	!GO SCAN
063266	326	326			DEF SCAN	
063270	000				BYT 0	
063271	236				RTN	
063272	230			SMYROM	BIN	
063273	110	263	326		STMD R10,=SAVR10	

```

063276 326
063277 120 040 242      STB R20,R40
063302 140 223          CLM R40
063304 020 240          LDB R40,R20          !SAVE 1ST CHAR. OF INPUT
063306 130 010 241      LDM R30,R10          !SAVE POINTER TO NEXT CHAR.
063311 122 223          CLM R22          !CLR R23
063313 250 260          LDB R22,=ROM#
063315 041 242          STB R22,R41
063317 316 326 326      JSB =ROMJSB
063322 326 326          DEF BPSALT
063324 000              BYT 0
063325 236              RTN
063326 316 326 326 SPSH45 JSB =ROMJSB
063331 326 326          DEF PSH45
063333 000              BYT 0
063334 236              RTN
063335 136 006 344 NVAL  PUBD R36,+R6          !SAVE TOKEN
063340 316 326 326      JSB =ROMJSB
063343 326 326          DEF NUMVAL
063345 000              BYT 0
063346 136 006 342 VALPOP POBD R36,-R6
063351 236              RTN
063352 136 006 344 NVA+  PUBD R36,+R6          !SAVE TOKEN
063355 316 326 326      JSB =ROMJSB
063360 326 326          DEF NUMVA+
063362 000              BYT 0
063363 360 361          JMP VALPOP
063365          !**** OTHER ATTRIBUTES !*****
063365 041              BYT 41
063366          !*****
063366 236 OTHER      RTN
063367          !**** DUMMY ATTRIBUTES !*****
063367 000 044          BYT 0,44
063371          !*****
063371 ROW            BSZ 0
063371 COL            BSZ 0
063371 DETL           BSZ 0
063371 RAMAX          BSZ 0
063371 CAMAX          BSZ 0
063371 RAMIN          BSZ 0
063371 CAMIN          BSZ 0
063371 RMAXAB         BSZ 0
063371 CMAXAB         BSZ 0
063371 NORMRW         BSZ 0
063371 NORMCL         BSZ 0
063371 LDIM           BSZ 0
063371 UDIM           BSZ 0
063371 236 DUMMY      RTN          ! DONE

```

```

063372      !          HED RAMIN, CAMIN, RAMAX, CAMAX, RMAXAB, CMAXAB
063372      !****  RAMIN ATTRIBUTES TABLE  *****
063372 000 055      BYT 0,55
063374      !
063374      !*****
063374      !** AMINROW : ROW NUMBER OF SMALLEST ELEMENT IN ARRAY MOST **
063374      !**          RECENTLY NAMED IN AMIN FUNCTION.          **
063374      !** IN   : <--- R12                                     **
063374      !** OUT  : AMINROW                                       **
063374      !**          <--- R12                                     **
063374      !*****
063374      !
063374 136 251 055 RAMIN.   LDM R36,=ROWMIN      !PTR (REL TO MBASE) TO RAMIN.
063377 000
063400 360 336      JMP INDEX                  !GTO COMMON RETRIEVAL CODE.
063402      !
063402      !****  CAMIN ATTRIBUTES TABLE  *****
063402 000 055      BYT 0,55
063404      !
063404      !*****
063404      !** AMINCOL : COLUMN NUMBER OF SMALLEST ELEMENT IN ARRAY **
063404      !**          MOST RECENTLY NAMED IN AMIN FUNCTION.      **
063404      !** IN   : <--- R12                                     **
063404      !** OUT  : AMINCOL                                       **
063404      !**          <--- R12                                     **
063404      !*****
063404      !
063404 136 251 057 CAMIN.   LDM R36,=COLMIN      !PTR (REL TO MBASE) TO CAMIN.
063407 000
063410 360 336      JMP INDEX                  !GTO COMMON RETRIEVAL CODE.
063412      !
063412      !****  RAMAX ATTRIBUTES TABLE  *****
063412 000 055      BYT 0,55
063414      !
063414      !*****
063414      !** AMAXROW : ROW NUMBER OF LARGEST ELEMENT IN ARRAY MOST **
063414      !**          RECENTLY NAMED IN AMAX FUNCTION.          **
063414      !** IN   : <--- R12                                     **
063414      !** OUT  : AMAXROW                                       **
063414      !**          <--- R12                                     **
063414      !*****
063414      !
063414 136 251 061 RAMAX.   LDM R36,=ROWMAX      !PTR (REL TO MBASE) TO RAMAX.
063417 000
063420 360 336      JMP INDEX                  !GTO COMMON RETRIEVAL CODE.
063422      !
063422      !****  CAMAX ATTRIBUTES TABLE  *****
063422 000 055      BYT 0,55
063424      !
063424      !*****
063424      !** AMAXCOL : COLUMN NUMBER OF LARGEST ELEMENT IN ARRAY **
063424      !**          MOST RECENTLY NAMED IN AMAX FUNCTION.      **
063424      !** IN   : <--- R12                                     **
063424      !** OUT  : AMAXCOL                                       **
063424      !**          <--- R12                                     **
063424      !*****
063424      !
063424 136 251 063 CAMAX.   LDM R36,=COLMAX      !PTR (REL TO MBASE) TO CAMAX.
063427 000
063430      !
063430      !*****
063430      !**          COMMON RETRIEVAL CODE          **
063430      !*****
063430      !
063430 230      INDEX      BIN                      !SET MODE.
063431 136 323 326      ADMD R36,=MBASE        !POINT TO ANSWER.
063434 326
063435 036 245      LDMD R36,R36              !GET ANSWER.
063437 364 336      JNG DETERR                !JIF NO ANSWER THERE.
063441 316 326 326      JSB =CONBIN            !ELSE CONVERT ANSWER TO BCD INT.
063444 360 336      JMP DETOK                !GO PUSH ANS ON STACK.

```



```

063446      !
063446      !****  RMAXAB ATTRIBUTES TABLE  *****
063446 000 055      BYT 0,55
063450      !
063450      !*****
063450      !** MAXABROW : ROW NUMBER OF ELEMENT WITH LARGEST ABSOLUTE **
063450      !**          VALUE IN ARRAY MOST RECENTLY NAMED IN MAXAB **
063450      !** IN  : <--- R12 **
063450      !** OUT : MAXABROW **
063450      !**          <--- R12 **
063450      !*****
063450      !
063450 136 251 065 RMXAB.  LDM R36,=ROWMAB      !PTR (REL TO MBASE) TO RMAXAB.
063453 000
063454 360 352      JMP INDEX      !GTO COMMON RETRIEVAL CODE.
063456      !
063456      !****  CMAXAB ATTRIBUTES TABLE  *****
063456 000 055      BYT 0,55
063460      !
063460      !*****
063460      !** MAXABCOL : COLUMN NUMBER OF ELEMENT WITH LARGEST ABSO- **
063460      !**          LUTE BALUE IN ARRAY MOST RECENTLY NAMED IN **
063460      !**          MAXAB FUNCTION. **
063460      !** IN  : <--- R12 **
063460      !** OUT : MAXABCOL **
063460      !**          <--- R12 **
063460      !*****
063460      !
063460 136 251 067 CMXAB.  LDM R36,=COLMAB      !PTR (REL TO MBASE) TO CMAXAB.
063463 000
063464 360 342      JMP INDEX      !GTO COMMON RETRIEVAL CODE.
063466      !
063466      !****  RNORMROW ATTRIBUTES TABLE *****
063466 000 055      BYT 0,55
063470      !
063470      !*****
063470      !** RNORMROW : ROW NUMBER WITH LARGEST SUM OF ABSOLUTE VALUES **
063470      !**          IN ARRAY MOST RECENTLY NAMED IN RNORM. **
063470      !** IN  : <--- R12 **
063470      !** OUT : RNORMROW **
063470      !**          <--- R12 **
063470      !*****
063470      !
063470 136 251 071 NORMR.  LDM R36,=RNORMX      !PTR (REL TO MBASE) TO RNORMX.
063473 000
063474 360 332      JMP INDEX      !GTO COMMON RETRIEVAL CODE.
063476      !
063476      !****  CNORMCOL ATTRIBUTES TABLE *****
063476 000 055      BYT 0,55
063500      !
063500      !*****
063500      !** CNORMCOL : COLUMN NUMBER WITH LARGEST SUM OF ABSOLUTE VALUES **
063500      !**          IN ARRAY MOST RECENTLY NAMED IN RNORM FUNCTION. **
063500      !** IN  : <--- R12 **
063500      !** OUT: CNORMCOL **
063500      !**          <--- R12 **
063500      !*****
063500      !
063500 136 251 073 NORMC.  LDM R36,=CNORMX      !PRT (REL TO MBASE) TO CNORMX.
063503 000
063504 360 322      JMP INDEX      !GTO COMMON RETRIEVAL CODE.
063506      !
063506      !*****  ERROM RUNTIME *****
063506 000 055      BYT 0,55
063510      !
063510      !*****
063510      !** ERROM : RETURNS NUMBER OF LAST PLUG-IN ROM TO GENERATE AN ERROR **
063510      !**          MESSAGE. IF ERROR ORIGINATED FROM SYSTEM ROMS OR IF **
063510      !**          NO ERROR HAS OCCURRED, FUNCTION RETURNS VALUE 0. **
063510      !** IN  : <--- R12 **
063510      !** OUT : 0 OR ROM# **

```

```

063510      !***          <--- R12                                     **
063510      !*****
063510      !
063510 136 223      EROM.      CLM R36
063512 260 326 326      LDBD R36,=ERROM#
063515 316 326 326      JSB =CONBIN
063520 360 336      JMP DETOK

```

```

063522      !          HED INVERSE, DET, SYSTEM SOL ROUTINES
063522      !****  DET ATTRIBUTES TABLE  *****
063522 000 055      BYT 0,55
063524      !
063524      !*****
063524      !** DETL : DETERMINANT OF LAST MATRIX INVERTED IN MAT ... INV  **
063524      !**          STATEMENT, OR SPECIFIED AS 1st ARGUMENT IN      **
063524      !**          MAT ... SYS STATEMENT.                          **
063524      !** IN   : <--- R12                                           **
063524      !** OUT  : DETERMINANT                                         **
063524      !**          <--- R12                                           **
063524      !*****
063524      !
063524 230      DET.      BIN          !SET MODE.
063525 146 261 326      LDMD R46,=MBASE      !TEMP STORAGE BASE ADDR.
063530 326
063531 140 046 265      LDMD R40,X46,DETR      !GET DET.
063534 013 000
063536 147 310 377      CMB R47,=377          !ANY DETERMINANT FOUND YET?
063541 372 336      JNC DETOK          !JIF YES.
063543 316 326 326 DETERR JSB =ERROR      !ERROR-NO PRECEDING INVERSION.
063546 007      BYT 7D
063547 140 223      ZERDET CLM R40          !DEFAULT RESULT=0.
063551 144 250 377      LDB R44,=377          !MAKE IT AN INTEGER.
063554 140 012 345 DETOK PUMD R40,+R12      !PUSH IT ON STACK.
063557 236      EXIT      RTN
063560      !
063560      !****  DET(A) ATTRIBUTES TABLE  *****
063560 024 055      BYT 24,55
063562      !*****
063562      !*** DET(A) ***
063562      !***      : DETERMINANT OF OPERAND MATRIX A ***
063562      !*** IN   : RELATIVE ADDRESS A ***
063562      !***          <--- R12 ***
063562      !*** OUT  : DETERMINANT OF A ***
063562      !***          <--- R12 ***
063562      !*** ***
063562      !*** TEMPORARY STORAGE NEEDED FOR: ***
063562      !***      POSITION VECTOR = 2 * Ma ***
063562      !***      W -- WORKING COPY OF A = 8 * Ma * Na ***
063562      !***      MAXIMUM EXPONENT VECTOR = 2 * Na ***
063562      !*** ***
063562      !*** TEMPORARY VARIABLES: ***
063562      !***      TMP1  = BASE ADDRESS OF A : Ba ***
063562      !***      TMP1+ = BASE ADDRESS OF W : Bw ***
063562      !***      TMP1++ = HEADER OF A ***
063562      !***      TMP2+ = BASE ADDRESS OF POSITION : Bp ***
063562      !***      TMP3  = BASE ADDRESS OF MAX-EXP : Bx ***
063562      !*** ***
063562      !*** THE METHOD OF FINDING THE DETERMINANT IS BASED ***
063562      !*** ON THE PLU DECOMPOSITION OF OPERAND ARRAY A. ***
063562      !*** GIVEN OPERAND ARRAY A. THEN THERE EXISTS: ***
063562      !*** ***
063562      !*** 1). LOWER TRIANGULAR ARRAY L (ASSUMING A 3X3) ***
063562      !*** ***
063562      !***      L(1,1)      0      0 ***
063562      !***      L = L(2,1) L(2,2)      0 ***
063562      !***      L(3,1) L(3,2) L(3,3) ***
063562      !*** ***
063562      !*** 2). UPPER TRIANGULAR ARRAY U (ASSUMING A 3X3) ***
063562      !*** ***
063562      !***      1      U(1,2) U(1,3) ***
063562      !***      U = 0      1      U(2,3) ***
063562      !***      0      0      1 ***
063562      !*** ***
063562      !*** 3). POSITION VECTOR P WHICH ACCOUNTS FOR ***
063562      !*** ROW INTERCHANGES ***
063562      !*** ***
063562      !*** SUCH THAT: ***
063562      !*** ***
063562      !***      A = P * L * U ***

```

```

063562      !***                                     ***
063562      !*** APPARENTLY THIS FACTORIZATION IS UNIQUE, I.E. ***
063562      !*** THERE EXISTS ONLY 1 P, L & U SUCH THAT A = PLU. ***
063562      !***                                     ***
063562      !*** IF A IS DIMENSIONED M X M, THEN ***
063562      !***                                     ***
063562      !*** DETERMINANT OF A = L(1,1) * L(2,2) * ... * L(M,M) ***
063562      !*****
063562      !
063562 316 326 326 DETA.      JSB =IDS10      !COMMON SETUP FOR INV,DET,SYS.
063565 366 336      JNZ LU3      !IF (N = 0)
063567 316 326 326 DET=1      JSB =FTR61      ! GEN 1 IN R40
063572 360 360      JMP DETOK      ! PUSH IT OUT & EXIT
063574      LU3      BSZ 0      !END IF
063574      !
063574      ! - GEN W -- (REAL) -
063574      ! - WORKING COPY OF A -
063574      !
063574 316 326 326      JSB =MNMUL2      !FIND 8N * N = SIZE W
063577 316 326 326      JSB =RESMEM      !MEM FOR WORK COPY A -- W
063602 370 353      JEN EXIT      !JIF NO ROOM -- EXIT
063604 165 012 345 LU1      PUMD R65,+R12      !PUSH NAME & Bw
063607 263 326 326      STMD R65,=TMP4+      !SAVE NAME & Bw
063612 260 326 326      LDBD R65,=TMP1++      !GET HEADER
063615 344      PUBD R65,+R12      !SAVE OLD HEADER
063616 261 326 326      LDMD R65,=TYPC      !GET TYPC, INCR.
063621 345      PUMD R65,+R12      !SAVE OLD TYPC, INCR
063622 223      CLM R65
063623 262 326 326      STBD R65,=TMP1++      !FAKE HEADER
063626 262 326 326      STBD R65,=TRCFLG      !SUPPRESS TRACE
063631 166 250 010      LDB R66,=10      !FAKE TYPE & INCR
063634 165 263 326      STMD R65,=TYPC      !STORE IT FOR STOV
063637 326
063640 263 326 326      STMD R65,=TYPB      !STORE IT FOR DOT PRODUCT
063643 316 326 326      JSB =EQUA10      !GEN WORK COPY OF A -- W
063646 165 012 343      POMD R65,-R12      !GET OLD TYPC, INCR
063651 263 326 326      STMD R65,=TYPC      !RESTORE THEM
063654 342      POBD R65,-R12      !GET OLD HEADER
063655 262 326 326      STBD R65,=TMP1++      !RESTORE IT
063660 223      CLM R65
063661 166 026 241      LDM R66,R26      !TYPE & 8N - INCRA
063664 165 263 326      STMD R65,=TYPA      !FOR DOT PRODUCT
063667 326
063670 316 326 326      JSB =TRCRST      !RESTORE TRACE FLAG
063673 156 261 326      LDMD R56,=MBASE      !TEMP STORAGE BASE ADDR
063676 326
063677 165 012 343      POMD R65,-R12      !GET Bw
063702 056 345      PUMD R65,+R56      !INIT PIVK=Bw-8NK (Bw)
063704 247      STMD R65,R56      !INIT PIVL=Bw-8K (Bw)
063705 263 326 326      STMD R65,=TMP1+      !Bw FOR FACTOR LOOP
063710 122 222      CLB R22      !CLEAR FOR 3 BYTE SUBTRACT
063712 155 020 241      LDM R55,R20      !N
063715 303      ADM R55,R20      !2N
063716 316 326 326      JSB =RESMEM      !FOR MAX EXP IN COLS OF W
063721 370 234      JEN EXIT      !JIF NO ROOM -- EXIT
063723      !
063723      ! *****
063723      ! ** FIND MAXIMUM EXPONENT OF EACH COLUMN OF **
063723      ! ** THE MATRIX **
063723      ! *****
063723      ! ** EXPONENT FOR EACH COLUMN INITIALIZED TO **
063723      ! ** -500 AND ADJUSTED BY -12 AT END OF **
063723      ! ** COLUMN LOOP. HENCE, THE RANGE OF MAX **
063723      ! ** EXPONENTS IS -512 TO 487. IF A MAX EXP **
063723      ! ** IS -512 THEN THE PIVOT ELEMENT IS 0. **
063723      ! ** NOTE THAT EXPONENTS ARE IN BCD MODE. **
063723      ! ** TO GET -500: **
063723      ! **          9 9 9 9 **
063723      ! **          - 0 5 0 0 **
063723      ! **          ----- **
063723      ! **          9 4 9 9 - NINE'S COMP **

```

```

063723      !      **      + 1      **
063723      !      **      -----      **
063723      !      **      9 5 0 0 - TEN'S COMP      **
063723      !      **      *****      **
063723      !
063723 316 326 326      JSB =SVPTRS      !SAVE PTR1
063726 165 263 326      STMD R65,=TMP3      !SAVE Bx
063731 326
063732 263 326 326      STMD R65,=PTR1-      !SET PTR1 TO Bx
063735 100 020 241      LDM R0,R20      !COLS = N
063740      PMON10      BSZ 0      !REPEAT
063740 130 020 241      LDM R30,R20      ! N TO 30
063743 000 305      SBM R30,R0      ! N - COLS
063745 066 243      STM R30,R66      ! FOR MNMUL3
063747 176 251 010      LDM R76,=10,0      ! FOR MNMUL3
063752 000
063753 316 326 326      JSB =MNMUL3      ! OFFSET NEXT COL
063756 165 261 326      LDMD R65,=TMP1+      ! GET Bw
063761 326
063762 055 305      SBM R65,R55      ! PT TO COL
063764 134 251 000      LDM R34,=0,95C      ! BIGEXP = -500
063767 225
063770 110 020 241      LDM R10,R20      ! ROWS = N
063773      PMON20      BSZ 0      ! REPEAT
063773 165 263 326      STMD R65,=PTR2-      ! SET PTR2
063776 326
063777 140 271 326      LDMI R40,=PTR2-      ! NXT COL ELE
064002 326
064003 367 336      JZR PMON40      ! IF (COL ELE <> 0)
064005 231      BCD      ! MODE FOR SEP10
064006 316 326 326      JSB =SEP10      ! SEPERATE EXP
064011 134 036 301      CMM R34,R36      ! BIGEXP & EXP
064014 365 336      JPS PMON30      ! IF (BIGEXP < EXP)
064016 241      LDM R34,R36      ! NEW BIGEXP
064017      PMON30      BSZ 0      ! END IF
064017 230      BIN      ! RESET MODE
064020      PMON40      BSZ 0      ! END IF
064020 130 222      CLB R30
064022 165 026 305      SBM R65,R26      ! PT NXT COL ELE
064025 110 213      DCM R10      ! ROWS = ROWS - 1
064027 366 342      JNZ PMON20      ! UNTIL (ROWS = 0)
064031 231      BCD
064032 134 315 022      SBM R34,=12C,0      ! ADJUST BIGEXP
064035 000
064036 230      BIN
064037 273 326 326      STMI R34,=PTR1-      ! STORE BIGEXP(COLS)
064042 100 213      DCM R0      ! COLS = COLS - 1
064044 366 272      JNZ PMON10      !UNTIL (COLS = 0)
064046      !      -----
064046      !      -- INITIALIZE DETRM. --
064046      !      -----
064046 140 223      CLM R40
064050 146 012 345      PUMD R46,+R12      !INIT EXP EXCESS COUNTER=0
064053 147 250 020      LDB R47,=10C      !INITIAL DETRM = 1
064056 140 345      PUMD R40,+R12      !MANT DET TO STACK
064060 145 345      PUMD R45,+R12      !EXP, SGN DET TO STACK
064062      !
064062      !      *****
064062      !      ** FACTOR MATRIX INTO UPPER AND LOWER TRIANGULAR MATRICIES **
064062      !      **      A = LU      **
064062      !      *****
064062      !      ** R0 = K : COUNTER FOR OUTER FACTOR LOOP, COUNTS NUM ROWS **
064062      !      ** R22 = I : COUNTER FOR PIVOT ELE LOOP, COUNTS NUM ROWS **
064062      !      ** R76 = M : POINTS TO ROW OF COL CONTAINING THE PIVOT ELE. **
064062      !      ** THE LU DECOMPOSITION IS DONE BY THE FOLLOWING STEPS IN **
064062      !      ** EACH ITERATION OF FACTOR LOOP: **
064062      !      **      (1). DIVIDE MATRIX W INTO A SUBMATRIX WHERE W(K,K) **
064062      !      **      IS THE 1st ELEMENT. **
064062      !      **      (2). SEARCH EACH COL OF THE SUBMATRIX FOR THE **
064062      !      **      LARGEST ELEMENT -- THE PIVOT ELEMENT. **
064062      !      **      (3). IF THE ROW CONTAINING THE PIVOT (AND POINTED **

```

```

064062      !      **      TO BY M) IS NOT THE TOP ROW OF THE SUBMATRIX, **
064062      !      **      THEN EXCHANGE THE 2 ROWS WITHIN THE SUBMATRIX. **
064062      !      **      (4). CALCULATE INDIVIDUAL L,U BY THE FORMULAS: **
064062      !      **
064062      !      **      OPTION BASE <= I <= NUM ROWS A **
064062      !      **      OPTION BASE <= K <= NUM COLS A **
064062      !      **
064062      !      **      FOR I < K : **
064062      !      **      Min(I,K)-1 **
064062      !      **      U(I,K) = [A(I,K) - Sum L(I,J)*U(J,K)] / L(I,I) **
064062      !      **      J=1 **
064062      !      **      FOR I >= K: **
064062      !      **      K-1 **
064062      !      **      L(I,K) = A(I,K) - Sum L(I,J)*U(J,K) **
064062      !      **      J=1 **
064062      !      **
064062      !      **      (5). AT THE END OF THE LU DECOMPOSITION, THE **
064062      !      **      DETERMINANT IS FOUND BY MULTIPLYING THE **
064062      !      **      DIAGONAL ELEMENTS OF THE MATRIX. **
064062      !      *****
064062      !
064062      DET5      BSZ 0      !LOOP (FACTOR LOOP)
064062      166 000 241      LDM R66,R0      ! K TO 66
064065      176 026 241      LDM R76,R26      ! 8N TO 76
064070      316 326 326      JSB =MNMUL3      ! FIND 8N X K
064073      145 055 241      LDM R45,R55      ! MOVE IT
064076      176 251 010      LDM R76,=10,0      ! 8 TO 76
064101      000
064102      316 326 326      JSB =MNMUL3      ! FIND 8 X K
064105      155 070 243      STM R55,R70      ! SAVE IT
064110      165 261 326      LDMD R65,=TMP1+      ! Bw
064113      326
064114      305      SBM R65,R70      ! Bw - 8K
064115      045 305      SBM R65,R45      ! Bw - 8K - 8NK
064117      136 261 326      LDMD R36,=MBASE      ! TEMP STORE PTR
064122      326
064123      313 031 000      ADM R36,=PIVOT      ! ADD IN PIVOT OFFSET
064126      140 223      CLM R40      ! PIVOT MANTISSA = 0
064130      036 247      STMD R40,R36      ! PIVOT EXP = 0
064132      122 000 241      LDM R22,R0      ! I = K
064135      076 243      STM R22,R76      ! M = K
064137      !
064137      !      - FIND COL PIVOT ELE -
064137      !
064137      DET7      BSZ 0      ! REPEAT
064137      165 263 326      STMD R65,=PTR2-      ! SET PTR2
064142      326
064143      140 271 326      LDMI R40,=PTR2-      ! GET W(I,K)
064146      326
064147      006 345      PUMD R40,+R6      ! SAVE VALUE W(I,K)
064151      165 345      PUMD R65,+R6      ! SAVE ADDR W(I,K)
064153      136 345      PUMD R36,+R6      ! SAVE PIVOT ADDR
064155      150 036 245      LDMD R50,R36      ! GET PIVOT
064160      231      BCD      ! SET MODE FOR ABS
064161      151 206      LRB R51      ! SHIFT OUT SIGN
064163      204      LLB R51      ! ABS(PIVOT)
064164      141 206      LRB R41      ! SHIFT OUT SIGN
064166      204      LLB R41      ! ABS(W(I,K))
064167      316 326 326      JSB =SUB10      ! ABS(PIVOT) - ABS(W(I,K))
064172      136 006 343      POMD R36,-R6      ! GET PIVOT ADDR
064175      165 343      POMD R65,-R6      ! GET ADDR W(I,K)
064177      140 343      POMD R40,-R6      ! GET VALUE W(I,K)
064201      150 012 343      POMD R50,-R12      ! GET SUBTRACT RESULT
064204      230      BIN      ! RESET MODE
064205      151 220      TSB R51      ! TEST RESULT
064207      376 336      JRZ DET12      ! IF (PIVOT < W(I,K))
064211      140 036 247      STMD R40,R36      ! PIVOT = W(I,K)
064214      122 076 243      STM R22,R76      ! M = I
064217      DET12      BSZ 0      ! END IF
064217      130 222      CLB R30
064221      165 026 305      SBM R65,R26      ! PT NXT W(I,K)

```

064224	122	211		ICM R22	!	I = I + 1
064226	020	301		CMM R22,R20	!	COMPARE I & N
064230	366	305		JNZ DET7	!	UNTIL (I = N)
064232	176	000	301	CMM R76,R0	!	COMPARE M & K
064235	367	336		JZR DET30	!	IF (M <> K)
064237			!	-----		
064237			!	- EXCHANGE ROWS -		
064237			!	-----		
064237	231			BCD	!	MODE FOR NCB
064240	140	012	342	POBD R40,-R12	!	SGN(DET)
064243	216			NCB R40	!	DET = -DET
064244	344			PUBD R40,+R12	!	SAVE DET
064245	230			BIN	!	RESET MODE
064246	176	066	243	STM R76,R66	!	COPY M
064251	166	205		LLM R66	!	FIND 2M
064253	155	223		CLM R55		
064255	166	055	243	STM R66,R55	!	2M TO 55
064260	165	261	326	LDMD R65,=TMP2+	!	Bp
064263	326					
064264	305			SBM R65,R55	!	Bp - 2M
064265	263	326	326	STMD R65,=PTR2-	!	SET UP PTR2
064270	134	000	241	LDM R34,R0	!	K TO 34
064273	205			LLM R34	!	2K
064274	055	243		STM R34,R55	!	2K TO 55
064276	157	222		CLB R57		
064300	165	261	326	LDMD R65,=TMP2+	!	GET Bp
064303	326					
064304	305			SBM R65,R55	!	Bp - 2K
064305	263	326	326	STMD R65,=PTR1-	!	SET UP PTR1
064310	130	271	326	LDMI R30,=PTR1-	!	POS(K)
064313	326					
064314	132	271	326	LDMI R32,=PTR2-	!	POS(M)
064317	326					
064320	273	326	326	STMI R32,=PTR1+	!	POS(K) = POS(M)
064323	130	273	326	STMI R30,=PTR2+	!	POS(M) = POS(K)
064326	326					
064327	126	066	243	STM R26,R66	!	8N FOR MNMUL3
064332	316	326	326	JSB =MNMUL3	!	FIND 8N * M
064335	165	271	326	LDMI R65,=MBASE	!	PIVK = Bw - 8NK
064340	326					
064341	263	326	326	STMD R65,=PTR2-	!	RESET PTR2
064344	261	326	326	LDMD R65,=TMP1+	!	Bw
064347	055	305		SBM R65,R55	!	Bw - 8N * M
064351	263	326	326	STMD R65,=PTR1-	!	PUT IT IN PTR1
064354			DET20	BSZ 0	!	REPEAT
064354	140	271	326	LDMI R40,=PTR2-	!	GET W(K,I)
064357	326					
064360	160	271	326	LDMI R60,=PTR1-	!	GET W(M,I)
064363	326					
064364	273	326	326	STMI R60,=PTR2+	!	W(K,I) = W(M,I)
064367	140	273	326	STMI R40,=PTR1+	!	W(M,I) = W(K,I)
064372	326					
064373	271	326	326	LDMI R40,=PTR1-	!	BACK UP PTR1
064376	271	326	326	LDMI R40,=PTR2-	!	BACK UP PTR2
064401	122	213		DCM R22	!	NUMROW = NUMROW - 1
064403	366	347		JNZ DET20	!	UNTIL (NUMROW = 0)
064405			DET30	BSZ 0	!	END IF
064405	230			BIN		
064406	165	271	326	LDMI R65,=MBASE	!	GET PIVK
064411	326					
064412	070	305		SBM R65,R70	!	Bw - 8K (L(K,K)ADDR)
064414	263	326	326	STMD R65,=PTR2-	!	SET UP PTR2
064417	100	211		ICM R0	!	K=K+1
064421	133	012	342	POBD R33,-R12	!	SGN(DET)
064424	134	343		POMD R34,-R12	!	EXP(DET)
064426	150	343		POMD R50,-R12	!	MANT(DET)
064430	140	271	326	LDMI R40,=PTR2-	!	GET L(K,K) VALUE
064433	326					
064434	231			BCD	!	MODE FOR SEP10
064435	316	326	326	JSB =SEP10	!	SEPARATE L(K,K)
064440	140	221		TSM R40	!	TEST L(K,K)

064442	366	336		JNZ DET32	!	IF (L(K,K) = 0)
064444	166	012	343	POMD R66,-R12	!	EXP EXCESS COUNTER
064447	100	020	301	CMM R0,R20	!	LAST COL OF A?
064452	367	336		JZR DET33	!	JIF YES & SET DET=0
064454	166	012	345	PUMD R66,+R12	!	SAVE EXP EXCESS CTR
064457	136	000	241	LDM R36,R0	!	GET K+1
064462	230			BIN	!	RESET MODE
064463	213			DCM R36	!	GIVES K
064464	316	326	326	JSB =DET88	!	GEN NON-0 L(K,K) PIVOT
064467	136	311	210	CMM R36,=88C,94C	!	PIVOT = 0 IF EXP = -512
064472	224					
064473	366	336		JNZ DET31	!	IF (EXP = -512)
064475	223			CLM R36	!	EXP=0
064476	140	223		CLM R40	!	MANT=0
064500			DET31	BSZ 0	!	END IF
064500			DET32	BSZ 0	!	END IF
064500	100	006	345	PUMD R0,+R6	!	SAVE R0
064503	316	326	326	JSB =MPY30	!	CALC NEW DET VALUE
064506	100	006	343	POMD R0,-R6	!	RESTORE R0
064511	166	012	343	POMD R66,-R12	!	EXP EXCESS CTR
064514	156	251	000	LDM R56,=0C,10C	!	MAX FOR TESTING
064517	020					
064520	137	310	120	CMB R37,=50C	!	EXP(DET) NEG?
064523	373	336		JCY NEGEXP	!	IF (EXP(DET) < 0)
064525	136	056	301	CMM R36,R56	!	EXP(DET) & 1000
064530	372	336		JNC DET35	!	IF(EXP(DET)>=1000)
064532	305			SBM R36,R56	!	EXP - 1000
064533	166	211		ICM R66	!	INCR EXP EXCESS
064535			DET35	BSZ 0	!	END IF
064535	360	336		JMP DET33		
064537			NEGEXP	BSZ 0	!	ELSE - (EXP(DET)>=0)
064537	136	311	001	CMM R36,=1C,90C	!	DET(EXP) & -1000
064542	220					
064543	373	336		JCY DET36	!	IF(EXP(DET)<=-1000)
064545	056	303		ADM R36,R56	!	EXP + 1000
064547	166	213		DCM R66	!	DECR EXP EXCESS
064551			DET36	BSZ 0	!	END IF
064551			DET33	BSZ 0	!	END IF
064551	230			BIN	!	RESET MODE FOR COMPARE
064552	100	020	301	CMM R0,R20	!	K & N
064555	366	336		JNZ DET34	!	IF (K = N)
064557			!	-----		
064557			!	- LAST COL W -		
064557			!	-----		
064557	231			BCD	!	MODE FOR EXP ADJ
064560	166	221		TSM R66	!	TEST EXP EXC CTR
064562	367	336		JZR EXP0K	!	IF(EXCESS <> 0)
064564	364	336		JNG NEGEX	!	IF(EXCESS > 0)
064566			POSLOP	BSZ 0	!	REPEAT
064566	136	056	303	ADM R36,R56	!	ADD 1000
064571	311	000	005	CMM R36,=0C,5C	!	EXP>=500?
064574	373	336		JCY EXP0K	!	JIF YES
064576	166	213		DCM R66		
064600	366	364		JNZ POSLOP	!	UNTIL(EXCESS=0)
064602	360	336		JMP EXP0K1		
064604			NEGEX	BSZ 0	!	ELSE-(EXCESS<=0)
064604			NEGLOP	BSZ 0	!	REPEAT
064604	136	056	305	SBM R36,R56	!	SUB 1000
064607	311	231	224	CMM R36,=99C,94C	!	EXP<=-502?
064612	372	336		JNC EXP0K	!	JIF YES
064614	166	211		ICM R66		
064616	366	364		JNZ NEGLOP	!	UNTIL(EXCESS=0)
064620			EXP0K1	BSZ 0	!	END IF
064620			EXP0K	BSZ 0	!	END IF
064620	316	326	326	JSB =RONF5	!	ROUND AND PACK DET
064623	140	012	345	PUMD R40,+R12	!	STACK IT
064626	316	326	326	JSB =RSPTRS	!	RESTORE PTR1
064631	230			BIN		
064632	235			CLE	!	FLAG NORMAL RTN
064633				BSZ 0	!	-----
064633	236			RTN	!	--- EXIT ---



```

064634          BSZ 0          !      -----
064634          DET34  BSZ 0          !      ELSE - (K <> N)
064634 166 012 345      PUMD R66,+R12      !      EXP EXCESS CTR
064637 140 345          PUMD R40,+R12      !      MANT(DET)
064641 136 345          PUMD R36,+R12      !      EXP(DET)
064643 132 344          PUBD R32,+R12      !      SGN(DET)
064645 110 261 326      LDMD R10,=MBASE      !      TEMP STORE ADDR
064650 326
064651 165 010 245      LDMD R65,R10      !      GET PIVK = Bw - 8NK
064654 130 222          CLB R30
064656 165 026 305      SBM R65,R26      !      NEW Bw - 8NK
064661 010 345          PUMD R65,+R10      !      SAVE IT
064663 245              LDMD R65,R10      !      PIVL = Bw - 8K
064664 315 010 000      SBM R65,=10,0,0      !      NEW Bw - 8K
064667 000
064670 247              STMD R65,R10      !      SAVE IT
064671 261 326 326      LDMD R65,=TMP1+      !      INITIAL Bw
064674 110 223          CLM R10      !      I = 0
064676          !
064676          !      *****
064676          !      ** FOR I <= I < K FIND:      **
064676          !      **                      Min(I,K)-1      **
064676          !      ** U(I,K) = [A(I,K) - Sum L(I,J) * U(J,K)] / L(I,I)      **
064676          !      **                      J=1      **
064676          !      *****
064676          !
064676          DET40  BSZ 0          !      REPEAT
064676 110 014 243      STM R10,R14      !      CTR FOR DOT
064701 316 326 326      JSB =DET66      !      DOT SETUP
064704 316 326 326      JSB =DET77      !      FIND NEW U(I,K)
064707 000 301          CMM R#,R0      !      COMPARE I & K
064711 366 363          JNZ DET40      !      UNTIL (I = K)
064713          !
064713          !      *****
064713          !      ** FOR NUM ROWS A >= I >= K FIND:      **
064713          !      **                      K-1      **
064713          !      ** L(I,K) = A(I,K) - Sum L(I,J) * U(J,K)      **
064713          !      **                      J=1      **
064713          !      *****
064713          !
064713          DET50  BSZ 0          !      REPEAT
064713 100 014 243      STM R0,R14      !      CTR FOR DOT
064716 316 326 326      JSB =DET66      !      DOT SETUP
064721 155 006 345      PUMD R55,+R6      !      ADDR A(I,K)
064724 316 326 326      JSB =RONF5      !      ROUND & PACK
064727 155 006 343      POMD R55,-R6      !      ADDR A(I,K)
064732 263 326 326      STMD R55,=PTR2-      !      SET UP PTR2
064735 140 273 326      STMI R40,=PTR2-      !      NEW L(I,K)
064740 326
064741 230              BIN      !      RESET MODE
064742 110 211          ICM R10      !      I = I + 1
064744 020 301          CMM R10,R20      !      COMPARE I & N
064746 366 343          JNZ DET50      !      UNTIL (I = N)
064750          BSZ 0          !      END IF (K = N)
064750 104 251 061      GTO DET5      !END LOOP (FACTOR LOOP)
064753 150
064754          !
064754          !*****
064754          !*** DET66 : SUMMATION ROUTINE TO FIND L & U IN LU DECOMPOSITION. ***
064754          !***          ROUTINE FINDS SUM:      ***
064754          !***                      R14+1      ***
064754          !***          A(I,K) - Sum L(I,J) * U(J,K)      ***
064754          !***                      J=1      ***
064754          !***      ***
064754          !*** IN      : R14 = SUMMATION COUNTER (UP BND - LOW BND)      ***
064754          !***          R26 = 8N      ***
064754          !***          R65 = Bw - 8NI : USED TO CALCULATE ADDR A(I,K)      ***
064754          !***          TMP1+ Bw : BASE ADDRESS OF W      ***
064754          !***          PIVL ADDRESS OF W(K,K) THE 1st ELE OF SUBMATRIX      ***
064754          !*** OUT      : R40 = MANTISSA OF SUM      ***
064754          !***          R36 = EXPONENT OF SUM      ***

```

```

064754      !***          R32  = SIGN OF SUM          ***
064754      !***          R65  = Bw -8NI - 8N FOR NEXT ADDR A(I,K)      ***
064754      !*****
064754      !
064754      DET66      BSZ 0
064754 146 261 326      LDMD R46,=MBASE          !TEMP STORAGE BASE ADDR
064757 326
064760 211          ICM R46
064761 211          ICM R46
064762 211          ICM R46          !POINT TO PIVL.
064763 155 046 245      LDMD R55,R46          !GET PIVL = Bw - 8K
064766 263 326 326      STMD R55,=TMP1          !AFETCH ADDR FOR DOT (COL)
064771 325 326 326      SBMD R55,=TMP1+          !-8K = Bw - 8K - Bw
064774 065 303      ADM R55,R65          !ADDR A(I,K) = Bw - 8NI - 8K
064776 263 326 326      STMD R55,=PTR2-          !SET UP PTR2
065001 140 271 326      LDMI R40,=PTR2-          !GET A(I,K)
065004 326
065005 165 263 326      STMD R65,=TMP2          !BFETCH ADDR FOR DOT (ROW)
065010 326
065011 316 326 326      JSB =DOT38          !A(I,K) - Sum L(I,J) * U(J,K)
065014 230          BIN
065015 130 222          CLB R30
065017 165 026 305      SBM R65,R26          !NEXT ADDR FOR A(I,K)
065022 236          RTN
065023      !
065023      !*****
065023      !*** DET77 : FINDS U(I,K) FOR LU DECOMPOSITION      ***
065023      !***          FINDS Y(I) FOR SYS(A,B)      ***
065023      !***          DIVIDES SUM IN R32,R36,R40 BY L(I,I)      ***
065023      !***          INSURES L(I,I) IS NON-ZERO.      ***
065023      !*** IN      : R10  = I : ROW COUNTER FOR U(I,K), C(I), L(I,I)      ***
065023      !***          R32  = SIGN OF SUM      ***
065023      !***          R36  = EXPONENT OF SUM      ***
065023      !***          R40  = MANTISSA OF SUM      ***
065023      !***          TMP1 = ADDRESS OF A(I,K)      ***
065023      !***          TMP2 = ADDRESS OF L(I,I) THE DIVISOR      ***
065023      !*** OUT      : U(I,K) (R40) STORED OVER A(I,K) (TMP1)      ***
065023      !***          R10  = I ROW COUNTER INCREMENTED      ***
065023      !***          DRP SET TO R10      ***
065023      !*****
065023      !
065023      DET77      BSZ 0
065023 165 070 243      STM R65,R70          !MOVE PTR TO NEXT COL
065026 132 033 242      STB R32,R33          !MOVE SIGN.
065031 136 034 243      STM R36,R34          !MOVE EXP.
065034 140 050 243      STM R40,R50          !MOVE MANTISSA.
065037 165 261 326      LDMD R65,=TMP2          !GET ADDR L(I,I)
065042 326
065043 263 326 326      STMD R65,=PTR2-          !SET PTR2
065046 140 271 326      LDMI R40,=PTR2-          !GET VALUE L(I,I)
065051 326
065052 231          BCD          !MODE FOR SEP10
065053 316 326 326      JSB =SEP10          !SEPARATE IT.
065056 140 221          TSM R40          !TEST MANTISSA OF L(I,I)
065060 366 336          JNZ DET78          !IF (L(I,I) = 0)
065062 136 010 241      LDM R36,R10          ! GET I
065065 230          BIN          ! RESET MODE.
065066 316 326 326      JSB =DET88          ! GET AN L(I,I) <> 0
065071      DET78      BSZ 0          !END IF
065071 231          BCD          !MODE FOR DIV14
065072 316 326 326      JSB =DIV14          !NEW U(I,K) OR C(I).
065075 140 012 343      POMD R40,-R12          !GET U(I,K) OR C(I).
065100 165 261 326      LDMD R65,=TMP1          !GET ADDR A(I,K)
065103 326
065104 263 326 326      STMD R65,=PTR2-          !SET PTR2
065107 140 273 326      STMI R40,=PTR2-          !SAVE OVER A(I,K) IF=U(I,K)
065112 326
065113 165 070 241      LDM R65,R70          !RESTORE PTR TO NEXT COL
065116 230          BIN          !RESET MODE.
065117 110 211          ICM R10          !I = I + 1
065121 236          RTN

```

```

065122      !
065122      !*****
065122      !*** DET88 : FINDS A NON-ZERO DIVISOR FOR DET77      ***
065122      !*** IN      : R36      = I : ROW COUNTER & POINTER TO MAX EXP VEC.  ***
065122      !***          R40      = 0      ***
065122      !***          TMP3     = Bx THE BASE ADDRESS OF MAX EXP VECTOR      ***
065122      !*** OUT      : R36      = BIG EXPONENT OF COLUMN CONTAINING L(I,I)  ***
065122      !***          THE DIVISOR FOR DET77.      ***
065122      !***          R40      = 1      ***
065122      !*****
065122      !
065122      DET88      BSZ 0
065122      136 205      LLM R36      !2I
065124      165 261 326      LDMD R65,=TMP3      !GET Bx
065127      326
065130      036 305      SBM R65,R36      !POINT BIGEXP FOR COL I
065132      263 326 326      STMD R65,=PTR2-
065135      136 271 326      LDMI R36,=PTR2-      !GET BIGEXP(I) .
065140      326
065141      147 250 020      LDB R47,=10C      !CORRESPONDING MANTISSA = 1
065144      231      BCD      !SET MODE FOR MPY30 OR DIV14.
065145      236      RTN      !RETURN WITH A NON ZERO L(I,I) .
065146      !
065146      !*****
065146      !*** IDS10 : COMMON SET UP FOR DETERMINANT, INVERSE AND SYSTEM ***
065146      !***          SOLUTION. INSURES OPERAND MATRIX IS SQUARE,      ***
065146      !***          INITIALIZES POSITION VECTOR, AND PROVIDES      ***
065146      !***          INFORMATION ABOUT OPERAND MATRIX A.      ***
065146      !*** IN      : RELATIVE ADDRESS A      ***
065146      !***          <--- R12      ***
065146      !*** OUT      : R22      = Ma -- NUMBER OF ROWS IN A      ***
065146      !***          R24      = Na -- NUMBER OF COLUMNS IN A      ***
065146      !***          R20      = Na -- NUMBER OF COLUMNS IN A      ***
065146      !***          R26      = 8 * Na (COLUMN INCREMENT FOR A)      ***
065146      !***          R76      = 8 * Na (COLUMN INCREMENT FOR A)      ***
065146      !***          TMP1     = BASE ADDRESS OF A : Ba      ***
065146      !***          TMP1+    = BASE ADDRESS OF A : Ba      ***
065146      !***          TMP1++   = HEADER OF A      ***
065146      !***          TMP2+    = BASE ADDRESS OF POSITION VECTOR : Bp      ***
065146      !*****
065146      !
065146      316 326 326      IDS10      JSB =LOCSZ      !Ba, Ma, Na
065151      122 024 301      CMM R22,R24      !COMPARE Ma & Na
065154      367 336      JZR IDS12      !IF (Ma <> Na)
065156      316 326 326      ERR14      JSB =ERROR+
065161      316 326 326      JSB =ERROR      ! MATRIX NOT SQUARE
065164      016      BYT 014D      ! REPORT ERROR AND ...
065165      360 336      JMP POPRS      ! EXIT
065167      IDS12      BSZ 0      !ELSE - (Ma = Na)
065167      155 223      CLM R55
065171      174 022 241      LDM R74,R22      ! SAVE N
065174      176 205      LLM R76      ! 2N
065176      055 243      STM R76,R55      ! 2N FOR RESMEM
065200      205      LLM R76      ! 4N
065201      205      LLM R76      ! 8N
065202      026 243      STM R76,R26      ! SAVE 8N
065204      316 326 326      JSB =RESMEM      ! FOR POSITION VECTOR
065207      370 336      JEN POPRS      ! JIF NO ROOM -- EXIT
065211      165 263 326      STMD R65,=TMP2+      ! BASE ADDR POSITION VECTOR: Bp
065214      326
065215      263 326 326      STMD R65,=PTR2-      ! ADDR FOR PTR2 IN LOOP
065220      316 326 326      JSB =ZERTST      ! SEE IF NULL ARRAY
065223      367 336      JZR IDS15      ! IF (NOT NULL ARRAY)
065225      114 223      CLM R14      ! K = 0
065227      IDSLP      BSZ 0      ! REPEAT
065227      273 326 326      STMI R#,=PTR2-      ! POS(K+1) = K
065232      211      ICM R#      ! K = K + 1
065233      022 301      CMM R#,R22      ! COMPARE K & N
065235      366 370      JNZ IDSLP      ! UNTIL (K = N)
065237      IDSLP      BSZ 0      ! END IF
065237      020 243      STM R#,R20      ! SAVE N IN R20

```

```

065241 126 076 243          STM R26,R76          ! 8N TO 76
065244          BSZ 0          !END IF
065244 236          RTN
065245          !
065245          !*****
065245          !*** RDIM-:  R76 = ROW OR COL PARAMETER          ***
065245          !***          FALL THRU RDIMS AND RDIM          ***
065245          !*****
065245          !
065245 176          RDIM-      DRP R76
065246          !
065246          !*****
065246          !*** RDIMS:  DRP POINTS TO ROW OR COL PARAMETER          ***
065246          !***          ARP POINTS TO R22 OR R24: ROW OR COL POSITION          ***
065246          !***          FALL THRU RDIM          ***
065246          !*****
065246          !
065246 243          RDIMS      STM R#,R#
065247          !
065247          !*****
065247          !*** RDIM :  REDIMENSIONS RESULT ARRAY          ***
065247          !*** IN   :  RELATIVE ADDR C          ***
065247          !***          <--- R12          ***
065247          !***          R22 = ROW PARAMETER          ***
065247          !***          R24 = COL PARAMETER          ***
065247          !*** OUT  :  REL ADDR C POPPED OFF STACK          ***
065247          !***          DRP = 22 THE ROW PARAMETER          ***
065247          !*****
065247          !
065247 235          RDIM      CLE          !CLEAR FLAG.
065250 233          DCE          !FLAG FOR TYPE C ARRAY
065251 165 012 343          POMD R65,-R12          !GET REL ADDR OF ARRAY
065254 316 326 326          JSB =REDIM.          !REDIMENSION ARRAY.
065257 316 326 326          JSB =VECFLG          !SETUP TRCFLG FOR TRACE LATER.
065262 117 310 300 ERRCK    CMB R17,=300          !CHECK FOR ERRORS IN REDIM
065265 372 336          JNC CKOUT          !IF (REDIM ERRORS)
065267 100 006 343 POPRS    POMD R0,-R6          ! TRASH 1 RTN
065272          CKOUT      BSZ 0          !END IF
065272 122 221          TSM R22          !TEST NUM ROWS IN ARRAY
065274 236          RTN
065275          !
065275          !*****
065275          !*** RQCOPY : COPIES B TYPE ARRAY (INCLUDING HEADER) INTO          ****
065275          !***          TEMPORARY RESERVED MEMORY.  USED IN SYS(A,B) .          ****
065275          !*** IN   : TMP2 = Bb : BASE ADDR B          ****
065275          !***          R22 = Mb : NUM ROWS B          ****
065275          !***          R24 = Nb : NUM COLS B          ****
065275          !*** OUT  : REL ADDR R OR Q          ****
065275          !***          <--- R6          ****
065275          !***          R75 = TOTAL SIZE OF ELEMENTS          ****
065275          !***          R65 = REL ADDR R OR Q          ****
065275          !*****
065275          !
065275          RQCOPY      BSZ 0
065275 130 251 010          LDM R30,=10,0          !ELEMENT SIZE
065300 000
065301 316 326 326          JSB =NUMBYT          !FIND NUM BYTES OF ELEMENTS
065304 145 251 013          LDM R45,=13,0,0          !HEADER SIZE
065307 000 000
065311 155 050 243          STM R55,R50          !SAVE SIZE OF ELEMENTS
065314 045 303          ADM R55,R45          !TOTAL NUM BYTES IN ARRAY
065316 316 326 326          JSB =RESMEM          !RESERVE MEMORY FOR SCRATCH ARRAY
065321 370 344          JEN POPRS          !IF (NOT ENOUGH ROOM)
065323          BSZ 0          ! TRASH 1 RTN & EXIT
065323          BSZ 0          !ELSE - (ENOUGH ROOM)
065323 316 326 326          JSB =SVPTRS          ! SAVE PTR1
065326 165 263 326          STMD R65,=PTR1          ! POINT TO BASE OF SCRATCH ARRAY
065331 326
065332 155 261 326          LDMD R55,=TMP2          ! GET Bb
065335 326
065336 045 303          ADM R55,R45          ! POINT TO HEADER OF B

```

```

065340 263 326 326      STMD R55,=PTR2      !   SET PTR2 TO HEADER B
065343      RQLOOP      BSZ 0                !   REPEAT
065343 155 270 326      LDBI R55,=PTR2-      !       GET ARRAY HEADER INFO
065346 326
065347 272 326 326      STBI R55,=PTR1-      !       COPY ARRAY HEADER INFO
065352 145 213          DCM R45              !       NUM BYTES = NUM BYTES - 1
065354 366 365          JNZ RQLOOP           !       UNTIL (NUM BYTES = 0)
065356 316 326 326      JSB =RSPTRS         !       RESTORE PTR1
065361 165 075 243      STM R65,R75         !       SAVE ABS ADDR OF HEADER
065364 325 326 326      SBMD R65,=FWCURR    !       MAKE SCRATCH ADDR RELATIVE
065367 175 263 326      STMD R75,=PTR2-      !       ABS ADDR HEADER R & W
065372 326
065373 050 241          LDM R75,R50          !       GET ELE SIZE
065375 136 250 100      LDB R36,=100        !       HEADER FOR R & W
065400 272 326 326      STBI R36,=PTR2-      !       STORE HEADER
065403 006 343          POMD R36,-R6         !       GET RETURN ADDR
065405 165 345          PUMD R65,+R6         !       PUSH REL ADDR R & W
065407 136 345          PUMD R36,+R6         !       REPLACE RETURN ADDR
065411 155 271 326      LDMI R55,=PTR2-      !       MOVE PAST ASCII NAME PTR
065414 326
065415 175 273 326      STMI R75,=PTR2-      !       STORE TOTAL SIZE OF ELE
065420 326
065421      RQEND      BSZ 0                !END IF
065421 236              RTN
065422      !**** INV (MAT) ATTRIBUTES TABLE *****
065422 024 055          BYT 24,55
065424      !*****
065424      !*** MAT C = INV (A) :                ***
065424      !*** STORES INVERSE OF MATRIX A INTO MATRIX C.          ***
065424      !*** IF A IS SINGULAR (DET(A)=0), THEN THE ROUTINE        ***
065424      !*** WILL STILL TRY TO FIND THE INVERSE OF A.              ***
065424      !*** IN      : RELATIVE ADDRESS C                          ***
065424      !*** RELATIVE ADDRESS A                                    ***
065424      !*** <--- R12                                              ***
065424      !*** OUT      : STACK POPPED & INVERSE ASSIGNED TO RESULT ARRAY ***
065424      !*** (REDIMENSIONED IF NECESSARY).                        ***
065424      !*** TEMPORARY STORAGE NEEDED FOR:                          ***
065424      !*** SAME AS DETERMINANT (SEE DETA.)                        ***
065424      !*** TEMPORARY VARIABLES :                                  ***
065424      !*** SAME AS DETERMINANT (SEE DETA.)                        ***
065424      !*** TMP4      = BASE ADDRESS RESULT ARRAY : Bc          ***
065424      !***
065424      !*** THE INVERSE OF A IS BASED ON THE EQUATION: A = P * L * U ***
065424      !*** P = POSITION VECTOR                                    ***
065424      !*** A = OPERAND ARRAY                                       ***
065424      !*** L = LOWER TRIANGLE OF LU DECOMPOSITION                ***
065424      !*** U = UPPER TRIANGLE OF LU DECOMPOSITION                ***
065424      !***
065424      !*** MATHEMATICALLY, THE INVERSE IS FOUND LIKE THIS:      ***
065424      !*** A = P * L * U                                           ***
065424      !*** A * INV(A) = P * L * U * INV(A)                       ***
065424      !*** INV(P) = INV(P) * P * L * U * INV(A)                  ***
065424      !*** INV(L) * INV(P) = INV(L) * L * U * INV(A)             ***
065424      !*** INV(L) * INV(P) = U * INV(A)                           ***
065424      !*** INV(U) * INV(L) * INV(P) = INV(U) * U * INV(A)       ***
065424      !*** INV(U) * INV(L) * INV(P) = INV(A)                     ***
065424      !***
065424      !*** SO THE INVERSE IS FOUND BY THE FOLLOWING STEPS:      ***
065424      !*** 1). FIND LU DECOMPOSITION OF A                          ***
065424      !*** 2). FIND V = INV(L)                                     ***
065424      !*** 3). FIND R = INV(U)                                     ***
065424      !*** 4). FIND RV = R * V                                    ***
065424      !*** a). LOWER TRIANGLE                                    ***
065424      !*** b). UPPER TRIANGLE                                    ***
065424      !*** 5). FIND INV(A) = RV * INV(P)                         ***
065424      !*****
065424      !
065424      LST
065424 316 146 152 INV10 JSB =IDS10            !COMMON SETUP FOR OPERAND ARRAY -- A
065427 316 247 152      JSB =RDIM            !REDIMENSION C TO N X N.
065432      !

```

```

065432      !      *****
065432      !      ** TEST FOR NULL ARRAY **
065432      !      *****
065432      !
065432 366 336      JNZ INV11      !IF (N = 0)
065434 316 326 326 DETL=1 JSB =FTR61      ! GEN 1 IN R40
065437 100 261 326 DINIT LDMD R0,=MBASE      ! GET TEMP STORE PTR
065442 326
065443 140 000 267      STMD R40,X0,DETR      ! PUT DET IN DETR
065446 013 000
065450 236      RTN      ! EXIT
065451      INV11      BSZ 0      !END IF
065451      !
065451      !      *****
065451      !      ** GET A = LU DECOMPOSITION **
065451      !      ** USING DETERMINANT ROUTINE **
065451      !      *****
065451      !
065451 145 261 326      LDMD R45,=TYPC      !GET TYPC, INCR
065454 326
065455 220      TSB R45      !TEST FOR REAL
065456 367 336      JZR INV14      !IF (NOT REAL)
065460 160 012 345      PUMD R60,+R12      ! SAVE NAME & Bc
065463 316 174 147      JSB =LU3      ! A=LU USING TEMP STORE
065466 360 336      JMP INV16
065470      INV14      BSZ 0      !ELSE - (REAL)
065470 316 204 147      JSB =LU1      ! A=LU&STORE OVER C OR A
065473      INV16      BSZ 0      !END IF
065473 370 324      JEN RQEND      !QUIT IF NOT ENOUGH ROOM
065475 140 012 343      POMD R40,-R12      !GET DETERMINANT
065500 316 037 153      JSB =DINIT      !PUT DET IN DETR
065503 316 326 326      JSB =TRCRST      !RESTORE TRACE FLAG
065506      !
065506      !      *****
065506      !      ** FIND MATRIX -- V -- SUCH THAT: I = L * V WHERE      **
065506      !      ** I IS THE IDENTITY MATRIX      **
065506      !      ** L IS THE MATRIX WITH THE LOWER TRIANGLE OF THE      **
065506      !      ** LU DECOMPOSITION, AND AN UPPER TRIANGLE OF ZEROS.      **
065506      !      ** V IS THE INVERSE OF MATRIX L      **
065506      !      ** FOR EXAMPLE, ASSUMING 3 X 3 MATRICIES      **
065506      !      **      **
065506      !      ** 1 0 0 L(1,1) 0 0 V(1,1) 0 0      **
065506      !      ** 0 1 0 = L(2,1) L(2,2) 0 * V(2,1) V(2,2) 0      **
065506      !      ** 0 0 1 L(3,1) L(3,2) L(3,3) V(3,1) V(3,2) V(3,3)      **
065506      !      **      **
065506      !      **      I      **
065506      !      ** LET D(I,K) = Sum L(I,J) * V(J,K)      **
065506      !      ** J=K      **
065506      !      **      **
065506      !      ** IF I = K THEN D(I,K) = 1 (DIAG ELE. OF I)      **
065506      !      ** IF I # K THEN D(I,K) = 0 (NON-DIAG ELE. OF I)      **
065506      !      **      **
065506      !      **      I-1      **
065506      !      ** THEN V(I,K) = [D(I,K) - Sum L(I,J) * V(J,K)] / L(I,I)      **
065506      !      ** J=K      **
065506      !      **      **
065506      !      ** IN THE NEXT SECTION, L IS CONTAINED IN W -- THE WORKING COPY      **
065506      !      ** OF A FROM THE DETERMINANT ROUTINE. ELEMENTS OF V ARE STORED      **
065506      !      ** OVER CORRESPONDING ELEMENTS OF L IN W.      **
065506      !      ** R0 = COL POINTER -- K      **
065506      !      ** R10 = ROW POINTER -- I      **
065506      !      ** R14 = SUMMATION COUNTER (UP BND - LOW BND)      **
065506      !      ** TMP1 = FETCH ADDR FOR L(I,J) (COL PTR DOT PRD)      **
065506      !      ** TMP2 = FETCH ADDR FOR V(J,K) (ROW PTR DOT PRD)      **
065506      !      ** (MBASE) POINTER TO DIAGONAL ELEMENTS OF U      **
065506      !      *****
065506      !
065506 100 223      CLM R0      !K = 0
065510 165 261 326      LDMD R65,=TMP1+      !GET Bw
065513 326
065514 360 336      JMP INV20

```

```

065516          INV24    BSZ 0          !REPEAT
065516 130 222          CLB R30
065520 165 271 326      LDMI R65,=MBASE    !   GET DIAG PTR
065523 326
065524 026 305          SBM R65,R26
065526 315 010 000      SBM R65,=10,0,0    !   NEXT DIAG ELE
065531 000
065532 273 326 326 INV20 STMI R#,=MBASE    !   RESET DIAG PTR
065535 100 010 243      STM R0,R10        !   I = K
065540          INV21    BSZ 0          !   LOOP
065540 114 010 241      LDM R14,R10        !   SUM CTR = I
065543 000 305          SBM R14,R0        !   SUM CTR = I - K
065545 155 271 326      LDMI R55,=MBASE    !   GET DIAG ELE
065550 326
065551 263 326 326      STMD R55,=TMP1    !   DOT COL PTR
065554 140 223          CLM R40          !   ACCUM = 0
065556 100 010 301      CMM R0,R10        !   COMPARE I & K
065561 366 336          JNZ INV22        !   IF (I = K)
065563 147 250 020      LDB R47,=10C      !   ACCUM = 1
065566          INV22    BSZ 0          !   END IF
065566 165 263 326      STMD R65,=TMP2    !   DOT ROW PTR
065571 326
065572 316 326 326      JSB =DOT38        !   D(I,K) - Sum L(I,J) * V(J,K)
065575 316 023 152      JSB =DET77        !   DIVIDE BY L(I,I) TO GET V(I,K)
065600 020 301          CMM R#,R20        !   COMPARE I & N
065602 367 336          JZR INV23        !   IF (I <> N)
065604 130 222          CLB R30
065606 165 026 305      SBM R65,R26        !   Bw -8NI
065611 360 325          JMP INV21        !   LOOP
065613          INV23    BSZ 0          !   END IF
065613 100 211          ICM R0          !   K = K + 1
065615 020 301          CMM R0,R20        !   COMPARE K & N
065617 366 275          JNZ INV24        !UNTIL (K = N)
065621
065621 !
065621 ! *****
065621 ! ** FIND MATRIX -- R -- SUCH THAT: I = U * R WHERE **
065621 ! ** I IS THE IDENTITY MATRIX **
065621 ! ** U IS THE MATRIX WITH THE UPPER TRIANGLE OF THE **
065621 ! ** LU DECOMPOSITION, AND A LOWER TRIANGLE OF ZEROS WITH **
065621 ! ** ONES ALONG THE DIAGONAL. **
065621 ! ** R IS THE INVERSE OF MATRIX U **
065621 ! ** FOR EXAMPLE, ASSUMING 3 X 3 MATRICIES **
065621 ! ** ** **
065621 ! ** 1 0 1 1 U(1,2) U(1,3) 1 R(1,2) R(1,3) **
065621 ! ** 0 1 0 = 0 1 U(2,3) * 0 1 R(2,3) **
065621 ! ** 0 0 1 0 0 1 0 0 1 **
065621 ! ** K-1 **
065621 ! ** R(I,K) = -U(I,K) - Sum U(I,J) * R(J,K) **
065621 ! ** J=I+1 **
065621 ! ** IN THE NEXT SECTION, U IS CONTAINED IN W -- THE WORKING COPY **
065621 ! ** OF A FROM THE DETERMINANT ROUTINE. ELEMENTS OF R ARE STORED **
065621 ! ** OVER CORRESPONDING ELEMENTS OF U IN W. **
065621 ! ** R0 = COL POINTER -- K **
065621 ! ** R10 = ROW POINTER -- I **
065621 ! ** R14 = SUMMATION COUNTER (UP BND - LOW BND) **
065621 ! ** TMP1 = FETCH ADDR FOR U(I,J) (COL PTR DOT PRD) **
065621 ! ** TMP2 = FETCH ADDR FOR R(J,K) (ROW PTR DOT PRD) **
065621 ! ** (MBASE) POINTER TO DAGONAL ELEMENTS OF U **
065621 ! *****
065621 !
065621 INV30    BSZ 0          !LOOP
065621 130 222          CLB R30
065623 165 026 303      ADM R65,R26        !   DIAG ELE NEXT ROW UP
065626 273 326 326      STMI R65,=MBASE    !   SAVE IT
065631 100 213          DCM R0          !   K = K - 1
065633 367 336          JZR INV40        !   ESCAPE LOOP WHEN K=0
065635 010 243          STM R0,R10        !   I = K
065637          INV32    BSZ 0          !   REPEAT
065637 130 222          CLB R30
065641 155 261 326      LDMD R55,=TMP2
065644 326

```

```

065645 263 326 326      STMD R55,=TMP1      !      DOT COL PTR
065650 026 303          ADM R55,R26          !      ADDR U(I,K)
065652 263 326 326      STMD R55,=PTR2-     !      SET UP PTR2
065655 140 271 326      LDMI R40,=PTR2-     !      GET U(I,K)
065660 326
065661 165 263 326      STMD R65,=TMP2      !      DOT ROW PTR
065664 326
065665 313 010 000      ADM R65,=10,0,0     !      PT NEXT ROW ...
065670 000
065671 303              ADM R65,R26          !      FOR DOT
065672 231              BCD                  !      MODE TO CHS
065673 141 204          LLB R41              !      LEFT DIGIT TO E
065675 216              NCB R41              !      CHANGE SIGN
065676 202              ERB R41              !      RESTORE LEFT DIGIT
065677 230              BIN                  !      RESET MODE
065700 114 000 241      LDM R14,R0           !      SUM CTR = K
065703 010 305          SBM R14,R10          !      SUM CTR = K - I
065705 316 326 326      JSB =DOT38          !      -U(I,K) - Sum U(I,J) * R(J,K)
065710 316 326 326      JSB =RONF5          !      ROUND, OV-UF & PACK
065713 155 261 326      LDMD R55,=TMP2      !      ADDR U(I,K)
065716 326
065717 263 326 326      STMD R55,=PTR2-     !      SET PTR2
065722 140 273 326      STMI R40,=PTR2-     !      R(I,K) OVER U(I,K)
065725 326
065726 230              BIN                  !      RESET MODE
065727 110 213          DCM R10              !      I = I - 1
065731 366 304          JNZ INV32            !      UNTIL (I = 0)
065733 165 271 326      LDMI R65,=MBASE     !      CURRENT DIAG ELE
065736 326
065737 313 010 000      ADM R65,=10,0,0     !      NEXT COL
065742 000
065743 263 326 326      STMD R65,=TMP2      !      SAVE IT
065746 360 251          JMP INV30
065750      INV40      BSZ 0                  !END LOOP
065750      !
065750      ! *****
065750      ! ** FIND MATRIX -- RV -- SUCH THAT: RV = R * V WHERE **
065750      ! ** R = INV(U) [INVERSE OF UPPER TRIANGLE OF LU DECOMP.] **
065750      ! ** V = INV(L) [INVERSE OF LOWER TRIANGLE OF LU DECOMP.] **
065750      ! ** FOR EXAMPLE, ASSUMING 3 X 3 MATRICIES **
065750      ! ** **
065750      ! **      1 R(1,2) R(1,3) V(1,1) 0 0 **
065750      ! **      0 1 R(2,3) * V(2,1) V(2,2) 0 **
065750      ! **      0 0 1 V(3,1) V(3,2) V(3,3) **
065750      ! **
065750      ! ** BUT NOTE THAT BOTH R & V ARE CONTAINED IN W LIKE THIS: **
065750      ! **
065750      ! **      V(1,1) R(1,2) R(1,3) **
065750      ! **      W = V(2,1) V(2,2) R(2,3) **
065750      ! **      V(3,1) V(3,2) V(3,3) **
065750      ! **
065750      ! ** VALUES OF RV ARE CALCULATED IN PLACE AND STORED IN W. **
065750      ! ** SEPERATE LOOPS ARE USED TO FIND THE LOWER AND UPPER **
065750      ! ** TRIANGLES OF RV. **
065750      ! *****
065750      !
065750      ! *****
065750      ! ** FIND LOWER TRIANGLE OF RV: **
065750      ! ** **
065750      ! ** FOR I >= K : **
065750      ! ** N **
065750      ! ** RV(I,K) = V(I,K) + Sum R(I,J) * V(J,K) **
065750      ! ** J=I+1 **
065750      ! ** **
065750      ! ** R20 = NUM COL (& ROWS) OF W (& A) : N **
065750      ! ** R26 = 8N **
065750      ! ** R10 = ROW PTR FOR RV : I **
065750      ! ** R0 = COL PTR FOR RV : K **
065750      ! ** (PIVL)= ADDR DIAGONAL ELE OF R (UPPER TRIANGLE OF RV) **
065750      ! ** (PIVK)= ADDR DIAGONAL ELE OF V (LOWER TRIANGLE OF RV) **
065750      ! ** IN THE EXAMPLE ABOVE, PIVK WILL BE THE ADDRS FOR **

```



```

065750      !      **          V(1,1), V(2,2) & V(3,3) AND PIVL WILL BE THE      **
065750      !      **          ADDRS FOR R(1,2) & R(2,3).                      **
065750      !      ** R14   = SUMMATION COUNTER (UP BND - LOW BND)              **
065750      !      ** TMP1  = FETCH ADDR FOR V IN SUMMATION                    **
065750      !      ** TMP2  = FETCH ADDR FOR R IN SUMMATION                    **
065750      !      *****
065750      !
065750      165 261 326      LDMD R65,=TMP1+      !Bw
065753      326
065754      INV42      BSZ 0
065754      110 020 301      CMM R10,R20      !COMPARE I & N
065757      372 336      JNC INV43
065761      104 251 377      GTO INV50      !IF (I < N)
065764      377
065765      110 211      INV43      ICM R10      ! I = I + 1
065767      114 261 326      LDMD R14,=MBASE      ! POINT TO PIVK
065772      326
065773      130 222      CLB R30      ! CLEAR FOR 3 BYTE SUBTRACT
065775      155 014 341      POMD R55,=R14      ! GET PIVK
066000      026 305      SBM R55,R26      ! PIVK - 8N
066002      014 247      STMD R55,R14      ! NEW PIVL = DIAG ELE OF R FOR COL K
066004      315 010 000      SBM R55,=10,0,0      ! OVER ONE ELEMENT TO THE RIGHT
066007      000
066010      273 326 326      STMI R55,=MBASE      ! NEW PIVK = DIAG ELE OF V FOR COL K
066013      100 223      CLM R0      ! K = 0
066015      INV44      BSZ 0      ! REPEAT
066015      114 020 241      LDM R14,R20      ! SUM CTR = N
066020      010 305      SBM R14,R10      ! SUM CTR = N - I
066022      130 222      CLB R30      ! CLEAR FOR 3 BYTE SUBTRACT
066024      155 065 241      LDM R55,R65      ! ADDR V(I,K+1)
066027      026 305      SBM R55,R26      ! 1st ELT BELOW CURRENT ROW
066031      263 326 326      STMD R55,=TMP1      ! FETCH ADDR OF V (COL PTR DOT PRD)
066034      271 326 326      LDMI R55,=MBASE      ! GET NEW PIVK
066037      263 326 326      STMD R55,=TMP2      ! FETCH ADDR OF R (ROW PTR DOT PRD)
066042      165 263 326      STMD R65,=PTR2-      ! POINT TO V(I,K+1)
066045      326
066046      140 271 326      LDMI R40,=PTR2-      ! GET VALUE OF V(I,K+1)
066051      326
066052      231      BCD      ! MODE TO CHANGE SIGN
066053      141 204      LLB R41      ! LEFT DIGIT TO E
066055      216      NCB R41      ! CHANGE SIGN
066056      202      ERB R41      ! RESTORE LEFT DIGIT
066057      316 326 326      JSB =DOT38      ! -V(I,K+1) - Sum R(I,J) * V(J,K+1)
066062      132 216      NCB R32      ! CORRECT IT'S SIGN
066064      316 326 326      JSB =RONF5      ! ROUND, OV-UF & PACK
066067      165 263 326      STMD R65,=PTR2-      ! POINT TO RV(I,K+1)
066072      326
066073      140 273 326      STMI R40,=PTR2-      ! STORE RV(I,K+1) [I >= K+1]
066076      326
066077      165 261 326      LDMD R65,=PTR2-      ! NEXT V(I,K+1)
066102      326
066103      230      BIN      ! RESET MODE
066104      100 211      ICM R0      ! K = K + 1
066106      010 301      CMM R0,R10      ! COMPARE K & I
066110      366 303      JNZ INV44      ! UNTIL (K = I)
066112      !
066112      ! *****
066112      ! ** FIND UPPER TRIANGLE OF RV:      **
066112      ! **      **
066112      ! ** FOR I < K :      **
066112      ! **          N      **
066112      ! **      RV(I,K) = Sum R(I,J) * V(J,K)      **
066112      ! **          J=K      **
066112      ! **      **
066112      ! ** R20   = NUM COL (& ROWS) OF W (& A) : N      **
066112      ! ** R26   = 8N      **
066112      ! ** R10   = ROW PTR FOR RV : I      **
066112      ! ** R0    = COL PTR FOR RV : K      **
066112      ! ** (PIVL)= ADDR DIAGONAL ELE OF R (UPPER TRIANGLE OF RV)      **
066112      ! ** (PIVK)= ADDR DIAGONAL ELE OF V (LOWER TRIANGLE OF RV)      **
066112      ! ** R14   = SUMMATION COUNTER (UP BND - LOW BND)      **

```

```

066112      !      ** TMP1  = FETCH ADDR FOR V IN SUMMATION      **
066112      !      ** TMP2  = FETCH ADDR FOR R IN SUMMATION      **
066112      !      ****
066112      !
066112      INV47      BSZ 0      !      LOOP
066112      100 020 301      CMM R0,R20      !      COMPARE K & N
066115      367 235      JZR INV42      !      ESCAPE IF K = N (BACKWARD REFERENCE)
066117      136 261 326      LDMD R36,=MBASE      !      PT TEMP STORE PTR
066122      326
066123      313 003 000      ADM R36,=PIVL      !      POINT TO PIVL
066126      130 222      CLB R30      !      CLEAR FOR 3 BYTE SUBTRACT
066130      155 036 245      LDMD R55,R36      !      GET PIVL
066133      315 010 000      SBM R55,=10,0,0      !      PIVL - 8
066136      000
066137      026 305      SBM R55,R26      !      NEW PIVL = PIVL-8-8N (ADDR V(J,K))
066141      036 247      STMD R55,R36      !      UPDATE NEW PIVL
066143      263 326 326      STMD R55,=TMP1      !      FETCH ADDR FOR V (COL PTR DOT PRD)
066146      165 263 326      STMD R65,=TMP2      !      FETCH ADDR FOR R (ROW PTR DOT PRD)
066151      326
066152      114 020 241      LDM R14,R20      !      SUM CTR = N
066155      000 305      SBM R14,R0      !      SUM CTR = N - K
066157      316 326 326      JSB =DOTPRD      !      Sum R(I,J) * V(J,K)
066162      230      BIN      !      RESET MODE
066163      165 263 326      STMD R65,=PTR2-      !      POINT TO RV(I,K+1)
066166      326
066167      140 273 326      STMI R40,=PTR2-      !      STORE RV(I,K+1) [I < K+1]
066172      326
066173      165 261 326      LDMD R65,=PTR2-      !      ADDR FOR NEXT R(I,J)
066176      326
066177      100 211      ICM R0      !      K = K + 1
066201      360 307      JMP INV47      !      END LOOP
066203      INV50      BSZ 0      !END IF
066203      !
066203      !      ****
066203      !      ** FIND INV (A) = RV * INV(P) WHERE :      **
066203      !      **      RV = INV(U) * INV(L)      **
066203      !      **      P = POSITION VECTOR      **
066203      !      ** MATRIX RV IS THE INVERSE OF A WITH THE COLUMNS IN A      **
066203      !      ** DIFFERENT ORDER. THIS STEP UNSCRAMBLES THE COLUMNS      **
066203      !      ** OF RV USING THE POSITION VECTOR. FOR EXAMPLE, THE      **
066203      !      ** CORRECT POSITION OF THE 3rd COLUMN OF RV IS IN THE      **
066203      !      ** CONTENTS OF THE 3rd ELEMENT OF THE POSITION VECTOR.      **
066203      !      ** THE UNSCRAMBLING STARTS FROM THE LAST COLUMN OF MATRIX      **
066203      !      ** RV AND THE LAST ELEMENT OF POSITION VECTOR P.      **
066203      !      ****
066203      !
066203      !
066203      136 020 241      LDM R36,R20      !N
066206      205      LLM R36      !2N
066207      316 326 326      JSB =SVPTRS      !SAVE PTR1
066212      165 261 326      LDMD R65,=TMP2+      !GET Bp: BASE ADDR OF POSITION VECTOR
066215      326
066216      140 223      CLM R40      !CLEAR FOR 3 BYTE SUBTRACT
066220      130 222      CLB R30      !CLEAR FOR 3 BYTE SUBTRACT
066222      165 036 305      SBM R65,R36      !Bp - 2N: LAST ELE OF POSITION VECTOR
066225      155 261 326      LDMD R55,=TMP1+      !GET Bw: BASE ADDR WORKING COPY OF A
066230      326
066231      026 305      SBM R55,R26      !Bw - 8N: 2nd ELE OF 1st COL OF W
066233      INV52      BSZ 0      !REPEAT
066233      !!      DCM R10      !      K = K - 1
066233      100 213      DCM R0      !      K = K - 1
066235      366 336      JNZ INV54      !      IF (K = 0)
066237      !      ****
066237      !      ** DONE UNSCRAMBLING COLUMNS **
066237      !      ****
066237      316 326 326      JSB =RSPTRS      !      RESTORE PTR1
066242      147 260 326      LDBD R47,=TYPC      !      TYPE OF RESULT ARRAY C
066245      326
066246      367 336      JZR CKTRC      !      IF (C NOT REAL)
066250      160 012 343      POMD R60,-R12      !      GET Bc
066253      263 326 326      STMD R60,=TMP4      !      DESTINATION ADDR FOR EQUA09
066256      165 261 326      LDMD R65,=TMP1+      !      GET Bw

```

```

066261 326
066262 263 326 326      STMD R65,=TMP1      !      SOURCE ADDR FOR EQUA09
066265 251 000 010      LDM R65,=0,10,0      !      TYPE & INCR OF W
066270 000
066271 263 326 326      STMD R65,=TYP A      !      STORE THEM FOR EQUA09
066274 122 020 241      LDM R22,R20      !      ROW COUNTER = N FOR EQUA09
066277 124 241      LDM R24,R20      !      COL COUNTER = N FOR EQUA09
066301 132 223      CLM R32      !      DON'T CONVERT TAGGED INT TO REAL
066303 316 326 326      JSB =ROMJSB
066306 326 326      DEF EQUA09      !      MOVE W INTO C
066310 261      BYT 261
066311 236      RTN      !      EXIT FOR INTEGER & SHORT C
066312      CKTRC      BSZ 0      !      END IF
066312 157 260 326      LDBD R57,=TRCFLG      !      TRACE: BIT4=0 ; NO TRACE: BIT4=1
066315 326
066316 374 336      JLZ CKTRTN      !      IF (TRACE)
066320 146 260 326      LDBD R46,=TYP C      !      GET TYPE OF C
066323 326
066324 160 261 326      LDMD R60,=TMP4      !      GET ARRAY NAME POINTER
066327 326
066330 165 261 326      LDMD R65,=TMP3++      !      GET Bc
066333 326
066334 316 326 326      JSB =FETCH      !      GET VALUE C(1,1) .
066337 316 326 326      JSB =STOV      !      STORE C(1,1)
066342      CKTRTN      BSZ 0      !      END IF
066342 236      RTN      !      EXIT FOR REAL C
066343      INV54      BSZ 0      !      END IF
066343 155 313 010      ADM R55,=10,0,0      !      POINT TO Kth COLUMN OF RV
066346 000 000
066350 165 263 326      STMD R65,=PTR2+      !      POINT TO Kth ELE OF POSITION VECTOR
066353 326
066354 211      ICM R65
066355 211      ICM R65      !      NEXT ELEMENT OF POSITION VECTOR
066356 132 271 326      LDMI R32,=PTR2+      !      GET POS(K)
066361 326
066362      INV56      BSZ 0
066362      !!      CMM R32,R10      !      COMPARE POS(K) & K
066362 132 000 301      CMM R32,R0      !      COMPARE POS(K) & K
066365 367 244      JZR INV52      !UNTIL (POS(K) <> K)
066367 136 032 241      LDM R36,R32      !MOVE POS(K)
066372 205      LLM R36      !2 * POS(K)
066373 034 243      STM R36,R34      !SAVE 2 * POS(K)
066375 175 261 326      LDMD R75,=TMP2+      !GET Bp
066400 326
066401 140 222      CLB R40      !CLEAR FOR 3 BYTE SUBTRACT
066403 175 036 305      SBM R75,R36      !Bp - 2 * POS(K) [POS(POS(K)) ]
066406 263 326 326      STMD R75,=PTR2-      !POINT TO POS(POS(K))
066411 136 271 326      LDMI R36,=PTR2-      !GET VALUE OF POS(POS(K))
066414 326
066415      !      *****
066415      !      ** EXCHANGE POSITIONS **
066415      !      ** FIND INV(P)      **
066415      !      *****
066415 132 273 326      STMI R32,=PTR2+      !POS(POS(K)) = POS(K)
066420 326
066421 241      LDM R32,R36      !POS(K) = POS(POS(K))
066422 134 205      LLM R34      !4 * POS(K)
066424 205      LLM R34      !8 * POS(K)
066425 175 261 326      LDMD R75,=TMP1+      !GET Bw: BASE ADDR OF RV
066430 326
066431 136 222      CLB R36      !FOR 3 BYTE SUBTRACT
066433 175 034 305      SBM R75,R34      !Bw - 8 * POS(K)
066436 263 326 326      STMD R75,=PTR2-      !POINT TO POS(K)th COL OF RV
066441 124 020 241      LDM R24,R20      !COUNTER = N
066444 155 263 326      STMD R55,=PTR1-      !POINT TO K th COL OF RV (SEE INV54)
066447 326
066450      INV58      BSZ 0      !LOOP
066450      !      *****
066450      !      ** EXCHANGE COLUMNS **
066450      !      *****
066450 140 271 326      LDMI R40,=PTR2-      !      POS(K) th COLUMN ELEMENT

```

```

066453 326
066454 170 271 326          LDMI R70,=PTR1-          !   K th COLUMN ELEMENT
066457 326
066460 273 326 326          STMI R70,=PTR2+          !   SWAP ...
066463 140 273 326          STMI R40,=PTR1+          !   THEM
066466 326
066467 124 213              DCM R24                  !   COUNTER = COUNTER - 1
066471 367 267              JZR INV56                !   ESCAPE IF COUNTER = 0
066473 130 222              CLB R30                  !   CLEAR FOR 3 BYTE SUBTRACT
066475 175 261 326          LDMD R75,=PTR2+          !   CURRENT POS(K) th COL ADDR
066500 326
066501 026 305              SBM R75,R26              !   ADDR OF POS(K) - 8N
066503 263 326 326          STMD R75,=PTR2-          !   NEXT POS(K) th COL ADDR
066506 261 326 326          LDMD R75,=PTR1+          !   CURRENT K th COL ADDR
066511 305                  SBM R75,R26              !   ADDR OF K - 8N
066512 263 326 326          STMD R75,=PTR1-          !   NEXT K th COL ADDR
066515 360 331              JMP INV58                !END LOOP
066517                      !   UNL
066517                      !
066517                      !**** INV(A) * B ATTRIBUTES TABLE *****
066517 011 051              BYT 11,51
066521                      !
066521                      ! *****
066521                      ! *** MAT X = INV(A) * B ***
066521                      ! *** MULTIPLIES INVERSE OF MATRIX A WITH MATRIX B, ***
066521                      ! *** AND STORES THE PRODUCT IN MATRIX X. ***
066521                      ! *** IN : RELATIVE ADDRESS X ***
066521                      ! *** RELATIVE ADDRESS A ***
066521                      ! *** RELATIVE ADDRESS B ***
066521                      ! *** <--- R12 ***
066521                      ! *** OUT : STACK POPPED AND RESULT MATRIX ASSIGNED ***
066521                      ! *** (AND REDIMENSIONED IF NECESSARY) ***
066521                      ! *****
066521                      !
066521 INV*B.   BSZ 0
066521 230          BIN
066522 120 222          CLB R20                      !CLEAR FLAG.
066524 212          DCB R20                      !FLAG FOR INV(A)*B.
066525 360 336          JMP SYS9                    !GO FIND ANSWER.
066527                      !
066527                      !**** SYS ATTRIBUTES !TABLE *****
066527 045 055          BYT 45,55
066531                      !
066531                      ! *****
066531                      ! *** MAT X = SYS (A,B) ***
066531                      ! *** A = COEFFICIENT MATRIX (MUST BE SQUARE) ***
066531                      ! *** B = CONSTANT MATRIX (MUST HAVE SAME NUM ROWS AS A) ***
066531                      ! *** X = UNKNOWN ARRAY ***
066531                      ! ***
066531                      ! *** STATEMENT USED TO SOLVE A SYSTEM OF N LINEAR EQUATIONS ***
066531                      ! *** WITH N UNKNOWNNS: ***
066531                      ! ***
066531                      ! *** A(1,1) X(1) + A(1,2) X(2) + ... + A(1,N) X(N) = B(1) ***
066531                      ! *** A(2,1) X(1) + A(2,2) X(2) + ... + A(2,N) X(N) = B(2) ***
066531                      ! *** . . . . ***
066531                      ! *** . . . . ***
066531                      ! *** . . . . ***
066531                      ! *** A(N,1) X(1) + A(N,2) X(2) + ... + A(N,N) X(N) = B(N) ***
066531                      ! ***
066531                      ! *** THE SOLUTION TO THIS LINEAR SYSTEM IS BASED ON GAUSSIAN ***
066531                      ! *** ELIMINATION. TWO STEPS ARE INVOLVED: ***
066531                      ! ***
066531                      ! *** 1). REDUCE SYSTEM TO ROW ECHELON FORM ***
066531                      ! ***
066531                      ! *** U(1,1) X(1) + U(1,2) X(2) + ... + U(1,N) X(N) = Y(1) ***
066531                      ! *** U(2,2) X(2) + ... + U(2,N) X(N) = Y(2) ***
066531                      ! *** . . . . ***
066531                      ! *** . . . . ***
066531                      ! *** . . . . ***
066531                      ! *** U(N,N) X(N) = Y(N) ***
066531                      ! ***

```

```

066531      !*** 2). SOLVING X(1), X(2), ..., X(N) BY BACK SUBSTITUTION.      ***
066531      !***                                                                ***
066531      !***          X(N) = Y(N) / U(N,N)                                ***
066531      !***          X(N-1) = [Y(N-1) - U(N-1,N) * X(N)] / U(N-1,N-1)      ***
066531      !***          .                                                    ***
066531      !***          .                                                    ***
066531      !***          .                                                    ***
066531      !***                                                                ***
066531      !*** MATHEMATICALLY, THE SOLUTION TO AX = B LOOKS LIKE THIS:      ***
066531      !***                                                                ***
066531      !***          PLU = A      (FIND PLU DECOMPSITION OF OPERAND A      ***
066531      !***                      PL CONTAINS THE MULTIPLIERS THAT        ***
066531      !***                      KNOCKS OUT THE LOWER TRIANGLE OF        ***
066531      !***                      THE ROW ECHELON FORM; AND                ***
066531      !***                      U CONTAINS THE COEFFICIENTS             ***
066531      !***                      OF THE ROW ECHELON FORM)                 ***
066531      !***          AX = B      (THE GIVEN SYSTEM OF LINEAR EQUATIONS)   ***
066531      !***          PLUX = B     (SUBSTITUTING PLU FOR A)                ***
066531      !***                                                                ***
066531      !*** SOLVE PLY = B FOR Y, THEN                                     ***
066531      !*** SOLVE UX = Y FOR X                                           ***
066531      !***                                                                ***
066531      !*** FOR GREATER ACCURACY, THE RESIDUALS ARE CALCULATED ON THE    ***
066531      !*** FIRST PASS AND ADDED TO THE SOLUTION MATRIX ON THE SECOND    ***
066531      !*** PASS.                                                         ***
066531      !***                                                                ***
066531      !*** IN: RELATIVE ADDRESS X                                       ***
066531      !***          RELATIVE ADDRESS A                                   ***
066531      !***          RELATIVE ADDRESS B                                   ***
066531      !***                                                                ***
066531      !***          <--- R12                                             ***
066531      !*** OUT: STACK POPPED AND RESULT ARRAY ASSIGNED                ***
066531      !***          AND REDIMENSIONED IF NECESSARY                      ***
066531      !*****                                                             ***
066531      !
066531      SYS10      BSZ 0
066531 165 012 343      POMD R65,-R12      !GET REL ADDR ARRAY B
066534 345          PUMD R65,+R12      !RESTORE IT.
066535 235          CLE                !CLEAR FLAG.
066536 234          ICE                !FLAG FOR TYPB.
066537 316 326 326      JSB =LOCSZ+    !GET Bb,Mb,Nb,TYPB, INCRB.
066542 316 275 152      JSB =RQCOPY    !COPY B INTO R
066545 316 275 152      JSB =RQCOPY    !COPY B INTO Q
066550 145 006 343      POMD R45,-R6    !GET REL ADDR Q
066553 155 343          POMD R55,-R6    !GET REL ADDR R
066555 162 012 343      POMD R62,-R12   !GET REL ADDR A & B
066560          BSZ 0                !LEAVE REL ADDR X ON STACK
066560 155 345          PUMD R55,+R12    !PUSH REL ADDR R
066562 145 345          PUMD R45,+R12    !PUSH REL ADDR Q
066564 137 222          CLB R37         !FLAG = 0
066566 344          PUBD R37,+R12       !PUSH FLAG
066567 155 345          PUMD R55,+R12    !PUSH REL ADDR R
066571 345          PUMD R55,+R12       !PUSH REL ADDR R
066572 175 261 326      LDMD R75,=TYPB !GET TYPB, INCRB
066575 326
066576 345          PUMD R75,+R12       !PUSH THEM
066577 261 326 326      LDMD R75,=TMP2  !GET Bb
066602 345          PUMD R75,+R12       !PUSH Bb
066603 155 345          PUMD R55,+R12    !PUSH REL ADDR R
066605 162 345          PUMD R62,+R12    !PUSH REL ADDR A & B
066607 175 343          POMD R75,-R12    !POP OFF B
066611 145 345          PUMD R45,+R12    !PUSH REL ADDR Q
066613 345          PUMD R45,+R12       !PUSH REL ADDR Q
066614 162 345          PUMD R62,+R12    !PUSH REL ADDR A & B
066616 120 222          CLB R20         !CLEAR FLAG.
066620 210          ICB R20            !FLAG FOR SYS(A,B) .
066621          SYS9      BSZ 0
066621 170 012 343      POMD R70,-R12    !GET X,A,B OR Q,A,B.
066624 121 342          POBD R21,-R12    !LAST ADDR BYTE X OR Q
066626 120 345          PUMD R20,+R12    !RESTORE FLAG AND ...
066630 170 345          PUMD R70,+R12    !REST OF ADDRESSES
066632      !

```

```

066632      ! *****
066632      ! ** AT THIS POINT, THE STACK LOOKS LIKE THIS: **
066632      ! ** **
066632      ! ** FOR MAT X = INV(A) * B: **
066632      ! ** **
066632      ! ** 377 (1 BYTE) ---- OPERATION FLAG **
066632      ! ** REL ADDR X ---- RDIM **
066632      ! ** REL ADDR A ---- IDS10 **
066632      ! ** REL ADDR B ---- LOCSZI **
066632      ! ** <--- R12 **
066632      ! ** **
066632      ! ** FOR MAT X = SYS(A,B): **
066632      ! ** **
066632      ! ** REL ADDR X \ **
066632      ! ** REL ADDR R >-- FOR MATADD (ADD RESIDUALS TO SOLUTION) **
066632      ! ** REL ADDR Q / **
066632      ! ** 000 (1 BYTE) ---- OPERATION FLAG 2nd PASS **
066632      ! ** REL ADDR R ---- RDIM 2nd PASS **
066632      ! ** REL ADDR R ---- LOCSZI 2nd PASS **
066632      ! ** TYPB (1 BYTE) \ **
066632      ! ** INCRB (2 BYTES) \ **
066632      ! ** ABS ADDR Bb \ FOR MAT- **
066632      ! ** REL ADDR R / (FIND RESIDUALS) **
066632      ! ** REL ADDR A / **
066632      ! ** REL ADDR Q / **
066632      ! ** 001 (1 BYTE) ---- OPERATION FLAG 1st PASS **
066632      ! ** REL ADDR Q ---- RDIM 1st PASS **
066632      ! ** REL ADDR A ---- IDS10 1st PASS **
066632      ! ** REL ADDR B ---- LOCSZI 1st PASS **
066632      ! ** <--- R12 **
066632      ! *****
066632      !
066632 165 343      POMD R65,-R12      !GET REL ADDR B
066632 263 326 326  STMD R65,=TMP2      !TEMPORARILY SAVE IT.
066632 316 146 152  JSB =IDS10      !COMMON SETUP FOR OPERAND ARRAY -- A
066632 165 261 326  LDMD R65,=TMP2      !GET REL ADDR B
066632 326
066632 012 345      PUMD R65,+R12      !PUSH IT ON STACK FOR LOCSZI
066632      SYS8      BSZ 0
066632 175 261 326  LDMD R75,=TMP2+      !GET Bp: BASE ADDR OF POSITION VECTOR
066632 326
066632 316 326 326  JSB =LOCSZI      !GET Bb, Mb, Nb
066632 175 263 326  STMD R75,=TMP2+      !RESTORE Bp
066632 326
066632      ! *****
066632      ! ** TEST DIMENSIONS OF **
066632      ! ** 1st & 2nd OPERANDS **
066632      ! *****
066632 122 020 301  CMM R22,R20      !COMPARE Mb & N
066632 367 336      JZR SYS12      !IF (MB <> N)
066632 316 326 326  JSB =MISMAT      ! ERR 11-DIM MISMATCH- & EXIT
066632      SYS12      BSZ 0      !END IF
066632 136 261 326  LDMD R36,=MBASE      !TEMP STORE POINTER
066632 326
066632 165 036 267  STMD R65,X36,PIVI      !SAVE Bb
066632 023 000
066632 316 247 152  JSB =RDIM      !REDIM Q TO N X Nb.
066632      ! *****
066632      ! ** TEST FOR NULL ARRAY **
066632      ! *****
066632 366 336      JNZ SYS13      !IF (N = 0)
066632 122 012 342  CLRR12      POBD R22,-R12      ! GET FLAG (0 OR 1= SYS ; 377 = INV*B)
066632 364 336      JNG DETREL      ! IF (FLAG > 0 )
066632 223          CLM R22      ! ROW = 0
066632 112 315 034  SBM R12,=34,0      ! CLEAN UP R12.
066632 000
066632 316 247 152  JSB =RDIM      ! REDIM X TO NULL ARRAY
066632      DETREL      BSZ 0      ! END IF
066632 104 251 033  GTO DETL=1      ! SET DET=1 & EXIT.
066632 153
066632      SYS13      BSZ 0      !END IF

```

```

066732 136 261 326          LDMD R36,=MBASE          !TEMP STORE ADDR
066735 326
066736 165 012 345          PUMD R65,+R12            !SAVE Bq
066741 055 305              SBM R65,R55              !Bq - ARRAY SIZE Q OR X
066743 045 243              STM R65,R45              !MOVE IT
066745 124 040 243          STM R24,R40              !TEMP SAVE Nb.
066750 205                  LLM R24                  !2Nb
066751 205                  LLM R24                  !4Nb
066752 205                  LLM R24                  !8Nb
066753 034 243              STM R24,R34              !TEMP SAVE 8Nb.
066755 014 243              STM R24,R14              !COPY TO SAVE IN TMP2
066757
066757 ! *****
066757 ! ** FIND COLUMN INCREMENT FOR MATRIX Q OR X.          **
066757 ! ** REGISTER CONTENTS:                                **
066757 ! ** R55 = ARRAY SIZE OF Q OR X                        **
066757 ! ** R65 = Bq (OR Bx) - ARRAY SIZE OF Q OR X          **
066757 ! ** R45 = Bq (OR Bx) - ARRAY SIZE OF Q OR X          **
066757 ! ** R40 = Nb                                           **
066757 ! ** R34 = 8 * Nb                                       **
066757 ! ** R24 = 8 * Nb                                       **
066757 ! ** R14 = 8 * Nb                                       **
066757 ! ** COLUMN INCREMENT OF Q OR X STORED IN:            **
066757 ! ** R14                                                **
066757 ! ** PIVK - IN STOLEN RAM                              **
066757 ! *****
066757 !
066757 167 260 326          LDBD R67,=TYPC            !WHAT TYPE OF MATRIX IS Q OR X ?
066762 326
066763 367 336              JZR SYS11                !IF (SHORT OR INTEGER)
066765 114 207              LRM R14                  ! 4Nb: ASSUME Q OR X IS SHORT
066767 167 310 020          CMB R67,=20              ! COMPARE TYPE & SHORT
066772 366 336              JNZ CSHORT              ! IF (INTEGER)
066774 114 040 305          SBM R14,R40              ! 3Nb FOR INTEGER.
066777 CSHORT BSZ 0          ! END IF
066777 134 251 010          LDM R34,=10,0            ! REPLACE 8Nb WITH 8
067002 000
067003 155 026 241          LDM R55,R26              ! 8Na FOR RESMEM VECTOR -- T
067006 157 222              CLB R57                  ! CLEAR MOST SIG. BYTE
067010 316 326 326          JSB =RESMEM              ! TO CALC Q(I,K)'S (COL K).
067013 371 336              JEZ SYS14                ! IF (NO ROOM)
067015 236                  ERROUT RTN              ! QUIT
067016 SYS14 BSZ 0          ! END IF
067016 SYS11 BSZ 0          !END IF
067016 175 223              CLM R75                  !CLEAR MOST SIG. BYTE
067020 114 075 243          STM R14,R75              !MOVE 3Nb, 4Nb, 8Nb
067023 175 263 326          STMD R75,=MTEMP          !MTEMP = 3Nb, 4Nb, 8Nb
067026 326
067027 130 223              CLM R30                  !CLEAR FOR 3 BYTE SUBTRACT
067031 165 026 305          SBM R65,R26              !Bt - 8N IF RESMEM ELSE GARBAGE.
067034 313 010 000          ADM R65,=10,0,0          !Bt - 8(N+1) OR GARBAGE.
067037 000
067040 012 345              PUMD R65,+R12            !SAVE FOR LATER.
067042 134 345              PUMD R34,+R12            !SAVE 8Nb OR 8 FOR LATER.
067044 145 075 303          ADM R45,R75              !Bq - ARRAY SIZE + 8Nb
067047 036 267 026          STMD R45,X36,PIVJ        !PIVJ = W(N,1) ADDR (TOP OF LAST COL OF Q)
067052 000
067053 164 261 326          LDMD R64,=TYPB          !GET TYPB, INCRB.
067056 326
067057 167 222              CLB R67                  !CLEAR MOST SIG. BYTE
067061 164 267 075          STMD R64,X36,TTYPB        !TTYPB = TYPE OF B & INCR OF B
067064 000
067065 !
067065 ! *****
067065 ! ** FIND COLUMN INCREMENT OF MATRIX B **
067065 ! ** IN : R24 = 8 * Nb                                **
067065 ! ** OUT: R24 = 3Nb, 4Nb, 8Nb                          **
067065 ! *****
067065 !
067065 310 020              CMB R64,=20              !WHAT TYPE IS MATRIX B?
067067 372 336              JNC BSET                  !IF (SHORT OR INTEGER)

```

```

067071 124                DRP R24                !   POINT TO 8Nb
067072 367 336            JZR BINT                !   IF (SHORT)
067074 207                LRM R24                !       4Nb
067075 360 336            JMP BSET
067077 170 012 343 CLRREL  POMD R70,-R12
067102 360 205            JMP CLRR12
067104                BINT    BSZ 0                !   ELSE - (INTEGER)
067104 207                LRM R#                  !       4Nb
067105 040 305            SBM R#,R40              !       3Nb
067107                BSZ 0                !   END IF
067107                BSET    BSZ 0              !END IF
067107 140 012 343        POMD R40,-R12
067112 160 342            POBD R60,-R12          !GET FLAG FROM STACK.
067114 344                PUBD R60,+R12          !RESTORE IT.
067115 140 345            PUMD R40,+R12
067117 124 345            PUMD R24,+R12          !SAVE 3Nb, 4Nb, OR 8Nb.
067121 160 220            TSB R60                !TEST OPERATION FLAG
067123 367 336            JZR BSET+              !IF (INV(A)*B OR 1st SYS PASS)
067125                !
067125                ! *****
067125                ! ** FIND PLU DECOMPOSITION & DETERMINANT OF A. **
067125                ! ** ALSO RESERVE MEMORY FOR VECTOR -- Y **
067125                ! *****
067125                !
067125 124 022 241            LDM R24,R22                !   N FOR LU3 (EQUA10).
067130 176 026 241            LDM R76,R26                !   8Na FOR LU3
067133 316 174 147          JSB =LU3                !   A=LU USING TEMP STORE
067136 370 255            JEN ERROUT              !   JIF NO ROOM
067140 155 223            CLM R55                !   CLEAR MOST SIG. BYTE
067142 126 055 243          STM R26,R55            !   8Na
067145 316 326 326          JSB =RESMEM            !   8Na BYTES FOR Y(I)'S.
067150 370 243            JEN ERROUT              !   JIF NO ROOM.
067152 136 261 326          LDMD R36,=MBASE          !   TEMP STORE ADDR
067155 326
067156 313 003 000          ADM R36,=PIVL            !   GET PIVL POINTER
067161 165 036 345          PUMD R65,+R36            !   SAVE PIVL = By
067164 155 345            PUMD R55,+R36            !   SAVE HR = 8Na
067166 120 345            PUMD R20,+R36            !   SAVE MSIZE = Na
067170 140 012 343          POMD R40,-R12            !   GET DETERMINANT
067173 036 247            STMD R40,R36              !   UPDATE DETR IN STOLEN RAM
067175 165 251 000          LDM R65,=0,10,0          !   REAL TYPE & INCR
067200 010 000
067202 263 326 326          STMD R65,=TYPA          !   PUT IT IN TYPA
067205                BSET+    BSZ 0              !END IF
067205 176 012 343          POMD R76,-R12            !GET 3Nb, 4Nb, OR 8Nb.
067210 263 326 326          STMD R76,=TMP3++          !SAVE IT
067213 367 262            JZR CLRREL              !JIF Nb = 0 (NULL ARRAY)
067215 100 260 326          LDBD R0,=TRCFLG          !GET TRACE FLAG.
067220 326
067221 006 344            PUBD R0,+R6              !SAVE FOR LATER.
067223 222                CLB R0                  !TEMP SET TRACE FLAG=0 (TURN OFF TRACE)
067224 262 326 326          STBD R0,=TRCFLG          !SET TRACE FLAG=0.
067227 012 343            POMD R0,-R12            !GET 8Nb OR 8.
067231 165 223            CLM R65                !INITIALIZE K = 0
067233                !
067233                ! *****
067233                ! ** LOOP TO FIND UNKNOWN VALUES OF X OR Q **
067233                ! **
067233                ! ** PIVI = Bb : BASE ADDR OF OPERAND ARRAY B **
067233                ! ** PIVJ = Q(N,1) : **
067233                ! ** PIVK = By : BASE ADDR OF Y VECTOR **
067233                ! ** TTPYB = TYPE & INCR OF OPERAND B **
067233                ! ** HR = 8 * Na **
067233                ! ** MSIZE = Na **
067233                ! ** TMP1+ = Bw : BASE ADDR OF W -- WORKING COPY OF A **
067233                ! ** TMP2+ = Bp : BASE ADDR OF POSITION VECTOR -- POS **
067233                ! ** TMP1 = Ba : BASE ADDR OF OPERAND A **
067233                ! ** TMP2 = Bb : BASE ADDR OF OPERAND B **
067233                ! ** TMP4+ = Bq : BASE ADDR OF Q **
067233                ! ** TMP3++= 3Nb, 4Nb, 8Nb -- COL INCR OF B **
067233                ! ** R76 = 3Nb, 4Nb, 8Nb -- COL INCR OF B **

```



```

067233      !      **      R65      = 8 * K      **
067233      !      **      R00      = 8 * Nb OR 8      **
067233      !      *****
067233      !
067233      SYS15      BSZ 0      !LOOP
067233 165 273 326      STMI R65,=MBASE      !      PIVK = 8K, 4K, 3K
067236 326
067237 261 326 326      LDMD R65,=TMP1+      !      Bw - 8NI (INITIALLY, I = 0)
067242 006 345      PUMD R65,+R6      !      SAVE IT
067244 110 223      CLM R10      !      I = 0
067246      !
067246      !      *****
067246      !      **      SOLVE PLY = B FOR Y      **
067246      !      **      **      **      **
067246      !      **      FOR 1 <= I <= N = NUM ROWS OF A      **
067246      !      **      **      **      **
067246      !      **      I-1      **
067246      !      **      Y(I) = [B(POS(I),K) - Sum L(I,J) * Y(J)] / L(I,I) WHERE      **
067246      !      **      J=1      **
067246      !      **      **      **      **
067246      !      **      L = LOWER TRIANGLE OF W -- WORKING COPY OF A      **
067246      !      **      **      **      **
067246      !      **      L(1,1) U(1,2) U(1,3)      **
067246      !      **      W = L(2,1) L(2,2) U(2,3) (ASSUMING 3 X 3 MATRIX)      **
067246      !      **      L(3,1) L(3,2) L(3,3)      **
067246      !      **      **      **      **
067246      !      **      B = 2nd OPERAND ARRAY      **
067246      !      **      Y = VECTOR SUCH THAT PLY = B & UX = Y      **
067246      !      **      POS = POSITION VECTOR -- P      **
067246      !      **      **      **      **
067246      !      **      R10 = I : ROW POINTER FOR Y      **
067246      !      **      R14 = SUMMATION COUNTER      **
067246      !      **      MSIZE = N : NUMBER OF ROWS OF A      **
067246      !      **      PIVK = 8K, 4K, 3K : DETERMINES Kth COL OF B      **
067246      !      *****
067246      !
067246      SYS16      BSZ 0      !      REPEAT
067246 114 010 241      LDM R14,R10      !      SUM CTR = I
067251 155 241      LDM R55,R10      !      MOVE I
067253 157 222      CLB R57      !      CLEAR MOST SIG. BYTE
067255 155 205      LLM R55      !      2I.
067257 165 261 326      LDMD R65,=TMP2+      !      Bp
067262 326
067263 055 305      SBM R65,R55      !      Bp - 2I
067265 263 326 326      STMD R65,=PTR2-      !      SET UP PTR2
067270 166 271 326      LDMI R66,=PTR2-      !      GET POS(I)
067273 326
067274 176 261 326      LDMD R76,=TMP3++      !      GET 8N
067277 326
067300 316 326 326      JSB =MNMUL3      !      8N * POS(I)
067303 136 261 326      LDMD R36,=MBASE      !      TEMP STORE ADDR
067306 326
067307 145 036 341      POMD R45,+R36      !      GET PIVK = 8K, 4K, 3K
067312 175 341      POMD R75,+R36      !      GET PIVL = By
067314 263 326 326      STMD R75,=TMP1      !      Y(J) FETCH ADDR (COL PTR DOT PRD)
067317 341      POMD R75,+R36      !      GET HR = 8N
067320 165 006 343      POMD R65,-R6      !      GET L(I,J) FETCH ADDR
067323 263 326 326      STMD R65,=TMP2      !      SAVE L(I,J) FETCH ADDR (ROW PTR DOT PRD)
067326 075 305      SBM R65,R75      !      NEXT L(I,J) ADDRESS
067330 006 345      PUMD R65,+R6      !      SAVE IT
067332 136 313 012      ADM R36,=12,0      !      POINT TO PIVI.
067335 000
067336 175 036 341      POMD R75,+R36      !      GET Bb
067341 045 305      SBM R75,R45      !      Bb - 8K
067343 055 305      SBM R75,R55      !      Bb - 8K - (8N * POS(I)) [FROM MNMUL3]
067345 136 261 326      LDMD R36,=MBASE      !      TEMP STORE ADDR
067350 326
067351 313 075 000      ADM R36,=TTYPB      !      POINT TO TYPE B.
067354 146 036 244      LDBD R46,R36      !      GET TYPE B FOR FETCH.
067357 175      DRP R75      !      FETCH ADDR DRP
067360 316 326 326      JSB =FETCH      !      GET B(POS(I),K).

```

```

067363 140 012 345      PUMD R40,+R12      !      PUSH VALUE FOR ONER
067366 316 326 326      JSB =ONER      !      DEMAND IT TO BE REAL.
067371 316 326 326      JSB =DOT38      !      B(POS(I),K) - Sum L(I,J) * Y(J)
067374 316 023 152      JSB =DET77      !      DIVIDE BY L(I,I) FOR Y(I)
067377 136 261 326      LDMD R36,=MBASE !      TEMP STORE ADDR
067402 326
067403 313 011 000      ADM R36,=MSIZE  !      POINT TO MSIZE
067406 110 036 331      CMMD R10,R36    !      COMPARE I & (MSIZE) = N
067411 366 233          JNZ SYS16      !      UNTIL (I = N)
067413 165 006 343      POMD R65,-R6    !      GET RID OF L(I,J) FETCH ADDR
067416
067416 !
067416 ! *****
067416 ! ** SOLVE UX = Y FOR X (INV*B) OR UQ = Y FOR Q (SYS) **
067416 ! ** **
067416 ! ** X(I,K) = Y(I) - Sum U(I,J) * X(J,K) **
067416 ! ** J>I **
067416 ! ** I = N, N-1, ... , 2, 1 **
067416 ! ** **
067416 ! ** R10 = I : ROW COUNTER FOR X **
067416 ! ** R14 = SUMMATION COUNTER **
067416 ! ** MSIZE = N : NUM ROWS OF A **
067416 ! ** PIVK = 8K, 4K, 3K : DETERMINES Kth COL OF X **
067416 ! *****
067416 !
067416 261 326 326      LDMD R65,=TYPA    !      GET TYPE AND INCR OF A
067421 345            PUMD R65,+R6      !      SAVE THEM
067422 100 263 326      STMD R0,=INCRA   !      8Nb = INCR AMT A FOR DOT38
067425 326
067426 110 223          CLM R10         !      I = 0
067430 136 213          DCM R36         !      POINT ...
067432 213            DCM R36         !      TO ...
067433 213            DCM R36         !      HR
067434 165 261 326      LDMD R65,=TMP2   !      L(N,N) POINTER
067437 326
067440 036 333          ADMD R65,R36     !      FETCH ADDR U(N-1,N) .
067442 136 313 020      ADM R36,=20,0   !      POINT TO PIVJ.
067445 000
067446 175 261 326      LDMD R75,=TMP1   !      POINTER TO Y(N) .
067451 326
067452 150 261 326      LDMD R50,=INCRC   !      GET INCREMENT C
067455 326
067456 152 222          CLB R52         !      FOR 3 BYTE SUBTRACT
067460 155 245          LDMD R55,R36     !      PIVJ = X(N,K) CURR COL ADDR.
067462 012 345          PUMD R55,+R12    !      PTR TO FINAL X(N,K) LOC.
067464 050 305          SBM R55,R50     !      PTR TO X(N,K+1)-NEXT PASS
067466 036 247          STMD R55,R36     !      SAVE IT IN PIVJ.
067470 223            CLM R55         !      FLAG REAL RESULT ARRAY
067471 320 326 326      CMBD R55,=TYPC   !      COMPARE TYPE OF RESULT ARRAY
067474 367 336          JZR SYS19        !      IF (INTEGER OR SHORT)
067476 152 012 343      POMD R52,-R12    !      Bq+(3,4)Nb(N-1) . &
067501          BSZ 0                  !      Bt-8(N+1) .
067501 345            PUMD R52,+R12      !      PUT Bt-8(N+1) BACK.
067502 155 343          POMD R55,-R12
067504 152 345          PUMD R52,+R12    !
067506 155 343          POMD R55,-R12
067510          BSZ 0                  !      END IF
067510 006 345          PUMD R#,+R6      !      0,=X REAL ELSE Bx-(3,4)Nb(N+1) .
067512 360 336          JMP SYS22
067514          BSZ 0                  !      REPEAT
067514 110 014 243      STM R10,R14      !      SUM CTR = I
067517 134 000 241      LDM R34,R0       !      MOVE 8Nb
067522 136 222          CLB R36         !      CLEAR FOR 3 BYTE SUBTRACT
067524 155 034 303      ADM R55,R34     !      ADDR NEXT X(J,K)
067527 012 345          PUMD R55,+R12    !      SAVE IT.
067531 136 261 326      LDMD R36,=MBASE !      TEMP STORE ADDR
067534 326
067535 313 006 000      ADM R36,=HR      !      HR POINTER.
067540 165 263 326      STMD R65,=TMP2   !      FETCH ADDR FOR X(J,K)
067543 326
067544 036 333          ADMD R65,R36     !      X(J,K) - 8Nb
067546 313 010 000      ADM R65,=10,0,0 !      X(J,K) - 8Nb - 8

```

```

067551 000
067552 175 263 326      STMD R75,=PTR2+      !      POINT TO Y(N-I)
067555 326
067556 140 271 326      LDMI R40,=PTR2+      !      GET Y(N-I) .
067561 326
067562 175 261 326      LDMD R75,=PTR2+      !      SAVE PTR2
067565 326
067566 316 326 326      JSB =DOT38           !      Y(N-I) - Sum U(N-I,J) * X(J,K)
067571 316 326 326      JSB =RONF5           !      ROUND,OV/UF,PACK
067574 230               BIN                  !      RESET MODE.
067575 155 012 343 SYS22 POMD R55,-R12       !      PTR TO X(N-I,K) .
067600 263 326 326      STMD R55,=TMP1       !      FETCH ADDR FOR U(N-I,J)
067603 263 326 326      STMD R55,=PTR2-      !      SET UP PTR2
067606 140 273 326      STMI R40,=PTR2-      !      SAVE X(N-I,K)
067611 326
067612 165 345           PUMD R65,+R12       !      SAVE 65
067614 006 343           POMD R65,-R6        !      0 OR Bq-(3,4)Nb((N+1)+J) .
067616 367 336           JZR SYS23           !      IF (X NOT REAL)
067620 263 326 326      STMD R65,=TMP4+      !      STORE ADDRESS
067623 345               PUMD R65,+R6        !      SAVE IT UNTIL AFTER STORE
067624 316 326 326      JSB =STOV            !      X(N,J) TO FINAL LOC
067627 165 006 343      POMD R65,-R6        !      GET OLD STORE ADDRESS
067632 323 326 326      ADMD R65,=MTEMP      !      NEXT ADDR TO X(N,J+1) .
067635                SYS23 BSZ 0             !      END IF
067635 006 345           PUMD R#,+R6         !      ADDR OR 0 BACK ON STACK.
067637 012 343           POMD R#,-R12        !      RESTORE 65
067641 136 261 326      LDMD R36,=MBASE      !      TEMP STORE ADDR.
067644 326
067645 313 011 000      ADM R36,=MSIZE       !      MSIZE POINTER.
067650 110 211          ICM R10              !      I=I+1.
067652 036 331          CMMD R10,R36         !      COMPARE I & N
067654 366 236          JNZ SYS20            !      UNTIL (I = N)
067656 155 006 343      POMD R55,-R6        !      RID R6 OF 0 OR X(I,J)
067661 343              POMD R55,-R6        !      GET TYPE & INCR OF A
067662 263 326 326      STMD R55,=TYPA      !      RESTORE THEM
067665 136 261 326      LDMD R36,=MBASE      !      TEMP STORE ADDR
067670 326
067671 313 076 000      ADM R36,=TINCRB      !      WHERE INCRB IS.
067674 165 271 326      LDMI R65,=MBASE      !      GET PIVK=8K.
067677 326
067700 036 333          ADMD R65,R36         !      8(K+1) (RESP 3,4) .
067702 176 261 326      LDMD R76,=TMP3++    !      8(K+1) (RESP 3,4) .
067705 326
067706 065 301          CMM R76,R65         !      8K = 8Nb?
067710 367 336          JZR SYS24           !      ESCAPE IF ALL COLS OF X FOUND.
067712 104 251 232      GTO SYS15           !END LOOP
067715 156
067716 155 012 343 SYS24 POMD R55,-R12       !RID R12 OF Bt-8(N-1) .
067721 343              POMD R55,-R12       !GET Bq
067722 146 342          POBD R46,-R12       !GET FLAG.
067724 364 336          JNG SYS25           !IF (NOT INV(A)*B)
067726 212              DCB R46             ! DECREMENT FLAG.
067727 367 336          JZR RESDUL          ! JIF 1st PASS (FIND RESIDUALS)
067731 170 006 342      POBD R70,-R6        ! GET TRACE FLAG.
067734 262 326 326      STBD R70,=TRCFLG    ! RESTORE TRACE FLAG.
067737 104 251 377      GTO MATADD          ! X = R + Q (ADD RESIDUALS TO SOLUTION)
067742 377
067743                SYS25 BSZ 0             !END IF
067743 170 006 342      POBD R70,-R6        !RID R6 OF TRCFLG.
067746 262 326 326      STBD R70,=TRCFLG    !RESTORE TRCFLG.
067751 316 312 154      JSB =CKTRC          !CHECK TRACE.
067754 236              RTN
067755      !
067755      ! *****
067755      ! ** RESDUL : FIND RESIDUAL ERROR **
067755      ! **
067755      ! ** LET Q BE THE COMPUTED SOLUTION FOR LINEAR SYSTEM AX = B. **
067755      ! ** THE ERROR OF THE COMPUTED SOLUTION IS : **
067755      ! **
067755      ! ** E = X - Q **
067755      ! **

```

```

067755      !  ** THE RESIDUAL ERROR IS :                               **
067755      !  **                                                              **
067755      !  **      R = AX - AQ  OR                                     **
067755      !  **                                                              **
067755      !  **      R = B  - AQ                                         **
067755      !  **                                                              **
067755      !  ** THE RESIDUAL MEASURES HOW WELL Q SATISFIES THE LINEAR SYSTEM **
067755      !  ** AX = B.                                                 **
067755      !  ****                                                              **
067755      !
067755      RESDUL  BSZ 0
067755 170 222      CLB R70
067757 212      DCB R70                      !FLAG = -1 (R=B-AQ)
067760 120 006 345      PUMD R20,+R6          !N
067763 126 345      PUMD R26,+R6          !8N
067765 165 261 326      LDMD R65,=TYP A      !TYPE & INCR OF A
067770 326
067771 345      PUMD R65,+R6
067772 261 326 326      LDMD R65,=TMP2+      !Bp
067775 345      PUMD R65,+R6
067776 261 326 326      LDMD R65,=TMP1+      !Bw
070001 345      PUMD R65,+R6
070002 316 326 326      JSB =MAT-          !FIND R = B - A * Q
070005 165 006 343      POMD R65,-R6
070010 263 326 326      STMD R65,=TMP1+      !Bw
070013 343      POMD R65,-R6
070014 263 326 326      STMD R65,=TMP2+      !Bp
070017 343      POMD R65,-R6
070020 263 326 326      STMD R65,=TYP A      !TYPE & INCR A
070023 126 343      POMD R26,-R6          !8N
070025 120 343      POMD R20,-R6          !N
070027 170 342      POBD R70,-R6          !TRACE FLAG
070031 262 326 326      STBD R70,=TRCFLG      !RESTORE TRCFLG.
070034 104 251 247      GTO SYS8
070037 155
070040      !
070040      !****  LINK NEXT SEGMENT          *****

```

```

070040      !          HED ()*A+()*B !ROUTINE
070040      !****      ()*A+()*B ATTRIBUTES TABLE      *****
070040 000 051          BYT 0,51
070042      !
070042      !*****
070042      !*** MAT C = (SCALOR A) * A + (SCALOR B) * B      ***
070042      !***      RESULTS OF 2 SCALOR MULTIPLICATIONS ADDED IN 1 STATEMENT      ***
070042      !*** IN : REL ADDR C      ***
070042      !***      SCALOR A      ***
070042      !***      REL ADDR A      ***
070042      !***      SCALOR B      ***
070042      !***      REL ADDR B      ***
070042      !***      <--- R12      ***
070042      !*** OUT : STACK POPPED & ARRAY ASSIGNED      ***
070042      !***      (RESULT ARRAY REDIMENSIONED IF NECESSARY)      ***
070042      !*****
070042      !
070042 316 326 326 LIN.      JSB =LOCSZI      !GET Bb, Mb, Nb
070045 140 012 343      POMD R40,-R12      !GET SCALOR B.
070050 273 326 326      STMI R40,=MBASE      !TEMP SAVE IT.
070053 316 326 326      JSB =DIMCK      !CHECK FOR MATCHING DIMENSIONS.
070056 316 326 326      JSB =REDIMC      !REDIM C & CALC COUNTER.
070061 316 262 152      JSB =ERRCK      !CHECK FOR REDIM ERRORS
070064 124 221      TSM R24      !TEST (Ma * Na )
070066 367 336      LINLOP      JZR LINEND      !WHILE (Ma * Na > 0)
070070 316 326 326      JSB =STKAIJ      !      STACK A(I,J) & SCL A
070073 316 326 326      JSB =MPYR70      !      (SCL A) * A(I,J)
070076 132 012 344      PUBD R32,+R12      !      SGN OF PROD TO STACK
070101 140 345      PUMD R40,+R12      !      MANTISSA TO STACK
070103 136 345      PUMD R36,+R12      !      EXP TO STACK
070105 140 271 326      LDMI R40,=MBASE      !      GET SCL B
070110 326
070111 345      PUMD R40,+R12      !      STACK IT
070112 316 326 326      JSB =BFET-      !      GET B(I,J)
070115 316 326 326      JSB =MPYR70      !      (SCL B) * B(I,J)
070120 316 326 326      JSB =RUNSUM      !      (SCL A) * A(I,J) + (SCL B) * B(I,J)
070123 316 326 326      JSB =STOV--      !      STORE AS C(I,J)
070126 170 012 343      POMD R70,-R12      !      GET SCL A OFF STACK
070131 316 326 326      JSB =NXTAB      !      NEXT A(I,J) & B(I,J)
070134 124 213      DCM R24      !      DECR (Ma * Na)
070136 360 326      JMP LINLOP      !      LOOP FOR TEST
070140      LINEND      BSZ 0      !END WHILE
070140 236      RTN
070141      !
070141      LST
070141 170 006 345 MPYR70      PUMD R70,+R6      !PROTECT R70 FROM MPYR.
070144      !      UNL
070144 316 326 326      JSB =MPYR      !DO THE MULTIPLY.
070147 170 006 343      POMD R70,-R6      !RECOVER R70.
070152 236      RTN

```

```

070153      !          HED MAT C = A +-. / B ROUTINE
070153      !****   MAT DIV ATTRIBUTES TABLE   ****
070153 010 051      BYT 10,51
070155      !*****
070155      !*** MAT C = A / B ***
070155      !***   PERFORMS DIVISION BETWEEN CORRESPONDING ELEMENTS ***
070155      !***   OF A AND B AND PLACES RESULT IN C. ***
070155      !*** IN   : REL ADDR C ***
070155      !***   REL ADDR A ***
070155      !***   REL ADDR B ***
070155      !***   <--- R12 ***
070155      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
070155      !***   (REDIMENSIONED IF NECESSARY) ***
070155      !*****
070155      !
070155 114 251 326 SLASH.   LDM R14,=A/B          !DIVIDE JSB ADDR.
070160 326
070161 360 336      JMP MADD10          !GTO COMMON ROUTINE.
070163      !
070163      !****   MAT TIMES ATTRIBUTES TABLE   ****
070163 010 051      BYT 10,51
070165      !*****
070165      !*** MAT C = A . B ***
070165      !***   PERFORMS MULTIPLICATION BETWEEN CORRESPONDING ***
070165      !***   ELEMENTS OF A AND B AND PLACES RESULT IN C. ***
070165      !*** IN   : REL ADDR C ***
070165      !***   REL ADDR A ***
070165      !***   REL ADDR B ***
070165      !***   <--- R12 ***
070165      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
070165      !***   (REDIMENSIONED IF NECESSARY) ***
070165      !*****
070165      !
070165 114 251 326 PER.     LDM R14,=MPROI+        !MULTIPLY JSB ADDR.
070170 326
070171 360 336      JMP MADD10          !GTO COMMON ROUTINE.
070173      !
070173      !****   MAT SUB ATTRIBUTES TABLE   ****
070173 010 051      BYT 10,51
070175      !*****
070175      !*** MAT C = A - B ***
070175      !***   PERFORMS SUBTRACTION BETWEEN CORRESPONDING ***
070175      !***   ELEMENTS OF A AND B AND PLACES RESULT IN C. ***
070175      !*** IN   : REL ADDR C ***
070175      !***   REL ADDR A ***
070175      !***   REL ADDR B ***
070175      !***   <--- R12 ***
070175      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
070175      !***   (REDIMENSIONED IF NECESSARY) ***
070175      !*****
070175      !
070175 114 251 326 MATSUB  LDM R14,=SBROI+        !SUBTRACT JSB ADDR.
070200 326
070201 360 336      JMP MADD10          !GTO COMMON ROUTINE.
070203      !
070203      !****   MAT ADD ATTRIBUTES TABLE   ****
070203 010 051      BYT 10,51
070205      !*****
070205      !*** MAT C = A + B ***
070205      !***   PERFORMS ADDITION BETWEEN CORRESPONDING ***
070205      !***   ELEMENTS OF A AND B AND PLACES RESULT IN C. ***
070205      !*** IN   : REL ADDR C ***
070205      !***   REL ADDR A ***
070205      !***   REL ADDR B ***
070205      !***   <--- R12 ***
070205      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
070205      !***   (REDIMENSIONED IF NECESSARY) ***
070205      !*****
070205      !
070205 114 251 326 MATADD  LDM R14,=ADROI+        !ADD JSB ADDR.
070210 326

```

```

070211      !
070211      !*****
070211      !***      COMMON CODE FOR MATRIX ARITHMETIC ROUTINES      ***
070211      !*****
070211      !
070211  316 326 326 MADD10  JSB =LOCSZI      !GET Bb, Mb, Nb
070214  316 326 326      JSB =DIMCK      !CHECK FOR DIMENSION MATCH
070217  316 247 152      JSB =RDIM      !REDIM C TO Ma X Na
070222  316 326 326      JSB =MNMUL2    !COMPUTE Ma * Na
070225  122 055 241      LDM R22,R55      !SAVE Ma * Na
070230  367 336      MADLOP  JZR MADEND    !WHILE (Ma * Na > 0)
070232  316 326 326      JSB =GET2RI     ! GET A(I,J) & B(I,J)
070235  014 306 326      JSB X14,ZRO     ! GOTO ARITH ROUTINE
070240  326
070241  140 012 343      POMD R40,-R12    ! GET C(I,J) FROM STACK
070244  316 326 326      JSB =STOV      ! STORE C(I,J)
070247  316 326 326      JSB =NXTAB      ! NEXT A(I,J) & B(I,J)
070252  122 213      DCM R22      ! DECR (Ma * Na)
070254  360 352      JMP MADLOP      ! LOOP FOR TEST
070256      MADEND  BSZ 0      !END WHILE
070256  236      RTN
070257      !
070257      !*****
070257      !*** A/B : DIVIDES ARRAY ELEMENT A(I,J) BY ARRAY ELEMENT B(I,J) **
070257      !*** IN : R50 = ELEMENT A(I,J) **
070257      !***      R40 = ELEMENT B(I,J) **
070257      !*** OUT : A(I,J) / B(I,J) **
070257      !***      <--- R12 **
070257      !*****
070257      !
070257  371 336  A/B      JEZ RLDIV      !JIF A(I,J), B(I,J) BOTH REAL.
070261  150 012 345      PUMD R50,+R12    ! INT A(I,J) TO STACK.
070264  140 345      PUMD R40,+R12    ! INT B(I,J) TO STACK.
070266  316 326 326      JSB =TWOR      ! DEMAND REAL A(I,J), B(I,J).
070271  316 326 326      JSB =SEP15      ! SEPARATE THEM.
070274  316 326 326 RLDIV JSB =DIV14      !A(I,J)/B(I,J).
070277  236      RTN
070300      !
070300      !*****
070300      !*** DIMCK : CHECKS TO SEE IF NUM ROWS A = NUM ROWS B AND ***
070300      !***      NUM COLS A = NUM COLS B. ERROR 11 IF NOT. ***
070300      !*** IN : REL ADDR A ***
070300      !***      <--- R12 ***
070300      !***      R22 = Mb : NUM ROWS B ***
070300      !***      R24 = Nb : NUM COLS B ***
070300      !*** OUT : REL ADDR A POPPED OFF STACK & DIMENSION CHECK MADE. ***
070300      !*****
070300      !
070300      DIMCK  BSZ 0
070300  174 022 241      LDM R74,R22      !SAVE Mb & Nb
070303  316 326 326      JSB =LOCSZ      !GET Ba, Ma, Na
070306  174 022 301      CMM R74,R22      !Ma = Mb & Na = Nb ?
070311  367 336      JZR QUIT      !IF (DIMENSIONS DON'T MATCH UP)
070313  316 326 326 MISMAT JSB =ERROR+
070316  316 326 326      JSB =ERROR      ! ERROR 11 - DIM MISMATCH
070321  013      BYT 011D
070322  112 261 326      LDMD R12,=TOS    ! CLEAN STACK
070325  326
070326  130 006 343      POMD R30,-R6      ! TRASH 1 RETURN.
070331      QUIT  BSZ 0      !END IF
070331  236      RTN
070332      !
070332      !*****
070332      !*** NXTAB : GIVEN CURRENT ADDR A(I,J) & B(I,J), ***
070332      !***      ROUTINE WILL FIND NEXT ADDR A(I+1,J), B(I+1,J) OR ***
070332      !***      A(I,J+1), B(I,J+1) ***
070332      !*** IN : TMP1 = CURRENT ADDR A(I,J) ***
070332      !***      TMP2 = CURRENT ADDR B(I,J) ***
070332      !***      INCRA = ROW OR COL INCREMENT OF A ***
070332      !***      INCRB = ROW OR COL INCREMENT OF B ***
070332      !*** OUT : TMP1 = ADDR A(I+1,J) OR A(I,J+1) ***

```

```

070332      !***      TMP2      = ADDR B(I+1,J) OR B(I,J+1)      ***
070332      !*****
070332      !
070332      LST
070332      NXTAB      BSZ 0
070332      316 326 326      JSB =NXTB
070335      316 326 326      JSB =NXTA
070340      236      RTN
070341      !
070341      !*****
070341      !*** NXTB : GIVEN CURRENT ADDR B(I,J),      ***
070341      !***      ROUTINE WILL FIND NEXT ADDR B(I+1,J) OR B(I,J+1)      ***
070341      !*** IN : TMP2 = CURRENT ADDR B(I,J)      ***
070341      !***      INCRB = ROW OR COL INCREMENT OF B      ***
070341      !*** OUT : TMP2 = ADDR B(I+1,J) OR B(I,J+1)      ***
070341      !***      R65 = ADDR B(I+1,J) OR B(I,J+1)      ***
070341      !***      R55 = ROW OR COL INCREMENT OF B      ***
070341      !*****
070341      !
070341      165 261 326 NXTB      LDMD R65,=TMP2      !GET ADDR OF B(J,I)
070344      326
070345      155 261 326      LDMD R55,=INCRB      !GET ELE SIZE OF B
070350      326
070351      316 326 326      JSB =NXTELE      !POINT TO NEXT B(J,I)
070354      165 263 326      STMD R65,=TMP2      !SAVE POINTER TO NEXT B(J,I)
070357      326
070360      236      RTN
070361      !
070361      !*****
070361      !*** NXTA : GIVEN CURRENT ADDR A(I,J),      ***
070361      !***      ROUTINE WILL FIND NEXT ADDR A(I+1,J) OR A(I,J+1)      ***
070361      !*** IN : TMP1 = CURRENT ADDR A(I,J)      ***
070361      !***      INCRA = ROW OR COL INCREMENT OF A      ***
070361      !*** OUT : TMP1 = ADDR A(I+1,J) OR A(I,J+1)      ***
070361      !***      R65 = ADDR A(I+1,J) OR A(I,J+1)      ***
070361      !***      R55 = ROW OR COL INCREMENT OF A      ***
070361      !*****
070361      !
070361      165 261 326 NXTA      LDMD R65,=TMP1      !GET ADDR OF A(I,J)
070364      326
070365      155 261 326      LDMD R55,=INCRA      !GET ELE SIZE OF A
070370      326
070371      316 326 326      JSB =NXTELE      !POINT TO NEXT A(I,J)
070374      165 263 326      STMD R65,=TMP1      !SAVE POINTER TO NEXT A(I,J)
070377      326
070400      236      RTN
070401      !
070401      !*****
070401      !*** NXTELE : FINDS NEXT ARRAY ELEMENT      ***
070401      !*** IN : R55 = ROW OR COL INCREMENT OF ARRAY      ***
070401      !***      R65 = ADDR OF CURRENT ARRAY ELEMENT      ***
070401      !*** OUT : R55 = ROW OR COL INCREMENT OF ARRAY      ***
070401      !***      R65 = ADDR OF NEXT ARRAY ELEMENT      ***
070401      !*****
070401      !
070401      NXTELE      BSZ 0
070401      !      UNL
070401      230      BIN      !MODE FOR SUBTRACTION
070402      157 222      CLB R57      !CLEAR FOR 3 BYTE SUBTRACT
070404      165 055 305      SBM R65,R55      !POINTER TO NEXT ARRAY ELEMENT
070407      236      RTN

```



```

070410      !          HED (R,C,E)NORM, ABSUM, SUM, CSUM, RSUM ROUTINES
070410      !**** ENORM ATTRIBUTES TABLE *****
070410 024 055      BYT 24,55
070412      !*****
070412      !
070412      NORM.      BSZ 0
070412 316 326 326      JSB =ROMJSB
070415 326 326      DEF NORM2
070417 261      BYT 261
070420 236      RTN
070421      !**** ABSUM ATTRIBUTES TABLE *****
070421 024 055      BYT 24,55
070423      !*****
070423      ABSUM.      BSZ 0
070423 316 326 326      JSB =ROMJSB
070426 326 326      DEF ABSUM2
070430 261      BYT 261
070431 236      RTN
070432      !**** SUM ATTRIBUTES !TABLE *****
070432 024 055      BYT 24,55
070434      !*****
070434      SUM.      BSZ 0
070434 316 326 326      JSB =ROMJSB
070437 326 326      DEF SUM2
070441 261      BYT 261
070442 236      RTN
070443      !**** CSUM ATTRIBUTES !TABLE *****
070443 024 055      BYT 24,55
070445      !*****
070445      CSUM.      BSZ 0
070445 316 326 326      JSB =ROMJSB
070450 326 326      DEF CSUM2
070452 261      BYT 261
070453 236      RTN
070454      !**** RSUM ATTRIBUTES !TABLE *****
070454 024 055      BYT 24,55
070456      !*****
070456      RSUM.      BSZ 0
070456 316 326 326      JSB =ROMJSB
070461 326 326      DEF RSUM2
070463 261      BYT 261
070464 236      RTN
070465      !**** RNORM ATTRIBUTES TABLE *****
070465 024 055      BYT 24,55
070467      !*****
070467      RNORM.      BSZ 0
070467 316 326 326      JSB =ROMJSB
070472 326 326      DEF RNORM2
070474 261      BYT 261
070475 236      RTN
070476      !**** CNORM ATTRIBUTES TABLE *****
070476 024 055      BYT 24,55
070500      !*****
070500      CNORM.      BSZ 0
070500 316 326 326      JSB =ROMJSB
070503 326 326      DEF CNORM2
070505 261      BYT 261
070506 236      RTN
070507      !
070507      !*****
070507      !*** TEST FOR NULL ARRAY      ***
070507      !*** IN   : R22  = M : NUMBER OF ROWS      ***
070507      !***      R24  = N : NUMBER OF COLS      ***
070507      !*** OUT  : R00  = 0 \ IF NULL ARRAY      ***
070507      !***      DRP  = 0 /      ***
070507      !***      DRP  = 22 OR 24 IF NOT NULL      ***
070507      !*****
070507      !
070507      LST
070507 231      ZTST-      BCD      !FOR UPCOMING SHIFT.
070510 130 200      ELB R30      !MOVE VEC-MAT INDICATOR INTO R30.

```

070512	262	326	326		STBD R30,=TMP4	!SAVE IT.
070515	230				BIN	!RESET MODE.
070516	122	221		ZERTST	TSM R22	!TEST NUMBER OF ROWS
070520				!	UNL	
070520	367	336			JZR RNULL	!JIF (NUM ROWS = 0)
070522	124	221			TSM R24	! TEST NUMBER OF COLS
070524	366	336			JNZ NRTN	! JIF (NUM COLS <> 0)
070526	100	223		RNULL	CLM R0	! FLAG NULL ARRAY
070530	236			NRTN	RTN	!DONE
070531				!		

```

070531      !          HED MATRIX UNARY MINUS ROUTINE
070531      !****  MONAD - ATTRIBUTES TABLE  ****
070531 007 050      BYT 7,50
070533      !*****
070533      !*** MAT C = - A          ***
070533      !***      CHANGE ALL THE SIGNS OF OPERAND ARRAY AND STORE IN  ***
070533      !***      RESULT ARRAY          ***
070533      !*** IN  : REL ADDR C          ***
070533      !***      REL ADDR A          ***
070533      !***      <--- R12          ***
070533      !*** OUT : STACK POPPED AND ARRAY ASSIGNED          ***
070533      !***      (REDIMENSIONED IF NECESSARY)          ***
070533      !*****
070533      !
070533 165 012 343 UMIN.      POMD R65,-R12          !GET REL ADDR A
070536 316 147 147      JSB =ZERDET          !PUSH 0 ON R12
070541 165 345      PUMD R65,+R#          !REPLACE REL ADDR A
070543 360 336      JMP SCLSUB          !FIND C = (0) - A
070545      !

```

```

070545      !          HED MAT C = (SCL) * + - / A ROUTINES
070545      !****  SCLDIV ATTRIBUTES TABLE  *****
070545 011 051      BYT 11,51
070547      !*****
070547      !*** MAT C = (SCALOR) / A          ***
070547      !***      PERFORMS DIVISION BETWEEN A SCALOR AND          ***
070547      !***      EACH ELEMENT OF THE OPERAND ARRAY.          ***
070547      !*** IN   : REL ADDR C          ***
070547      !***      SCALOR          ***
070547      !***      REL ADDR A          ***
070547      !***      <--- R12          ***
070547      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED          ***
070547      !***      (AND REDIMENSIONED IF NECESSARY)          ***
070547      !*****
070547      !
070547 114 251 257 SCLDIV   LDM R14,=A/B          !DIVIDE JSB ADDR.
070552 160
070553 360 336      JMP SCLCOM          !GTO COMMON ROUTINE.
070555      !
070555      !****  SCLSUB ATTRIBUTES TABLE  *****
070555 011 051      BYT 11,51
070557      !*****
070557      !*** MAT C = (SCALOR) - A          ***
070557      !***      PERFORMS SUBTRACTION BETWEEN A SCALOR AND          ***
070557      !***      EACH ELEMENT OF THE OPERAND ARRAY.          ***
070557      !*** IN   : REL ADDR C          ***
070557      !***      SCALOR          ***
070557      !***      REL ADDR A          ***
070557      !***      <--- R12          ***
070557      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED          ***
070557      !***      (AND REDIMENSIONED IF NECESSARY)          ***
070557      !*****
070557      !
070557 114 251 326 SCLSUB   LDM R14,=SBROI+          !SUBTRACT JSB ADDR.
070562 326
070563 360 336      JMP SCLCOM          !GTO COMMON ROUTINE.
070565      !
070565      !****  SCLADD ATTRIBUTES TABLE  *****
070565 011 051      BYT 11,51
070567      !*****
070567      !*** MAT C = (SCALOR) + A          ***
070567      !***      PERFORMS ADDITION BETWEEN A SCALOR AND          ***
070567      !***      EACH ELEMENT OF THE OPERAND ARRAY.          ***
070567      !*** IN   : REL ADDR C          ***
070567      !***      SCALOR          ***
070567      !***      REL ADDR A          ***
070567      !***      <--- R12          ***
070567      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED          ***
070567      !***      (AND REDIMENSIONED IF NECESSARY)          ***
070567      !*****
070567      !
070567 114 251 326 SCLADD   LDM R14,=ADROI+          !ADD JSB ADDR.
070572 326
070573 360 336      JMP SCLCOM          !GTO COMMON ROUTINE.
070575      !
070575      !****  SCLMUL ATTRIBUTES TABLE  *****
070575 011 051      BYT 11,51
070577      !*****
070577      !*** MAT C = (SCALOR) * A          ***
070577      !***      PERFORMS MULTIPLICATION BETWEEN A SCALOR AND          ***
070577      !***      EACH ELEMENT OF THE OPERAND ARRAY.          ***
070577      !*** IN   : REL ADDR C          ***
070577      !***      SCALOR          ***
070577      !***      REL ADDR A          ***
070577      !***      <--- R12          ***
070577      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED          ***
070577      !***      (AND REDIMENSIONED IF NECESSARY)          ***
070577      !*****
070577      !
070577 114 251 326 SCLMUL   LDM R14,=MPROI+          !MULTIPLY JSB ADDR.
070602 326

```

```

070603      !
070603      !*****
070603      !***      COMMON CODE FOR SCALOR - ARRAY ARITHMETIC      ***
070603      !*****
070603      !
070603  316 326 326 SCLCOM   JSB =LOCSZ           !GET Ba, Ma, Na
070606  316 326 326         JSB =REDIMC          !REDIM C & CALC COUNTER.
070611  316 262 152         JSB =ERRCK          !LOOK FOR REDIM ERROR & NULL ARRAY
070614  124 221           TSM R24              !TEST (Ma * Na)
070616  367 336         SMULOP  JZR SMUEND        !WHILE (Ma * Na > 0)
070620  316 326 326         JSB =STKAIJ         !  STACK SCL & A(I,J)
070623  014 306 326         JSB X14,ZRO         !  GOTO ARITH ROUTINE
070626  326
070627  140 012 343         POMD R40,-R12        !  PRODUCT FROM STACK
070632  316 326 326         JSB =STOV           !  STORE C(I,J)
070635  170 012 343         POMD R70,-R12        !  GET SCALOR
070640  316 361 160         JSB =NXTA           !  POINT NEXT A(I,J)
070643  124 213           DCM R24              !  DECR (Ma * Na)
070645  360 347           JMP SMULOP            !  LOOP FOR TEST
070647         SMUEND   BSZ 0                  !END WHILE
070647  236           RTN
070650      !
070650      !*****
070650      !*** REDIMC : GET SCALOR FROM STACK AND REDIMENSION      ***
070650      !***      RESULT ARRAY      ***
070650      !*** IN  : SCALOR      ***
070650      !***      <--- R12      ***
070650      !***      R22 = Ma : NUM ROWS A      ***
070650      !***      R24 = Na : NUM COLS A      ***
070650      !*** OUT  SCALOR POPPED FROM R12      ***
070650      !***      R24 = Ma * Na      ***
070650      !***      R55 = Ma * Na      ***
070650      !***      R70 = SCALOR      ***
070650      !*****
070650      !
070650  170 012 343 REDIMC   POMD R70,-R12        !GET SCALOR.
070653  316 247 152 REDC+   JSB =RDIM           !REDIM C TO Ma X Na.
070656  170 006 345         PUMD R70,+R6        !TEMP SAVE SCALOR
070661  316 326 326         JSB =MNMUL         !FIND Ma * Na
070664  124 055 241         LDM R24,R55         !INITIALIZE COUNTER TO Ma * Na.
070667  170 006 343         POMD R70,-R6        !GET SCALOR
070672  236           RTN

```

```
070673      !      HED MAT C = CROSS(B,A) ROUTINE
070673      !****  CROSS ARRIBUTES TABLE  *****
070673 045 055      BYT 45,55
070675      !*****
070675      CROSS.  BSZ 0
070675 316 326 326      JSB =ROMJSB
070700 326 326      DEF CROSS2
070702 261      BYT 261
070703 236      RTN
```

```

070704      !          HED DOT ROUTINE (C = B DOT A)
070704      !**** DOT ATTRIBUTES !TABLE *****
070704 045 055      BYT 45,55
070706      !*****
070706      DOT.      BSZ 0
070706 316 326 326      JSB =ROMJSB
070711 326 326      DEF DOT2
070713 261      BYT 261
070714 236      RTN
070715      !
070715      !*****
070715      !*** IF A & B ARE VECTORS, THIS ROUTINE WILL FIND THE DOT PRODUCT ***
070715      !*** OF THE VECTORS: ***
070715      !*** NUM ROWS A ***
070715      !*** Sum A(I) * B(I) ***
070715      !*** I=1 ***
070715      !*** IF A & B ARE MATRICIES, THE ROUTINE WILL FIND DOT PRODUCT ***
070715      !*** OF A ROW OF A WITH A COLUMN OF B: ***
070715      !*** NUM COLS A ***
070715      !*** Sum A(I,J) * B(J,I) ***
070715      !*** J=1 ***
070715      !*** EXIT WITH DOT PRODUCT IN R40. ***
070715      !*****
070715      !
070715      LST
070715      DOTPRD BSZ 0
070715      !      UNL
070715 144 223      CLM R44      !INITIALIZE SUM=0.
070717 250 377      LDB R44,=377      !ASSUME INTEGER.
070721 316 326 326      JSB =DOTACC      !GOTO ACCUMULATE PRODUCT LOOP
070724      LST
070724 144 310 377 RDNF5- CMB R44,=377      !SEE IF AN INTEGER
070727      !      UNL
070727 373 336      JCY RNFRTN      !IF (NOT INTEGER)
070731 231      RDNF5 BCD      ! MODE SETTING FOR ROU10
070732 316 326 326      JSB =ROU10      ! ROUND ANSWER
070735 316 326 326      JSB =NFR5      ! OV/UF CHECK & PACK ANS
070740      RNFRTN BSZ 0      !END IF
070740 236      RTN
070741      !
070741      !*****
070741      !*** R14 ***
070741      !*** DOT38 : FINDS SUM: C(I,K) - Sum A(I,J) * B(J,K) ***
070741      !*** J=1 ***
070741      !*** IN : R40 = VALUE OF C(I,K) ***
070741      !*** OUT : R40 = VALUE OF ABOVE FORMULA ***
070741      !*** FOR OTHER PARAMETERS SEE DOTACC ***
070741      !*****
070741      !
070741 231      DOT38 BCD      !MODE FOR NEXT INSTRUCTION.
070742 316 326 326      JSB =SEP10      !SEP INITIAL SUM VALUE IN R40.
070745 132 216      NCB R32      !CHANGE SIGN OF INITIAL SUM.
070747 316 326 326      JSB =DOTACC      !GO FIND VALUE ROW DOT COL.
070752 132 216      NCB R32      !CHANGE SIGN OF FINAL SUM.
070754 236      RTN      !END.
070755      !
070755      !*****
070755      !*** R14 ***
070755      !*** DOTACC : ACCUMULATES THE SUM: Sum A(I,J) * B(J,K) ***
070755      !*** J=1 ***
070755      !*** IN : R14 = UPPER BOUND OF SUMMATION ***
070755      !*** TMP1 = ADDR OF A(I,J) (COL PTR FOR DOT PRODUCT) ***
070755      !*** TMP2 = ADDR OF B(J,K) (ROW PTR FOR DOT PRODUCT) ***
070755      !*** OUT : R14 = 0 ***
070755      !*** R40 = MANTISSA OF SUM ***
070755      !*** R36 = EXPONENT OF SUM ***
070755      !*** R32 = SIGN OF SUM ***
070755      !*****
070755      !
070755      LST
070755      DOTACC BSZ 0

```

```

070755      !          UNL
070755 100 006 345      PUMD R0,+R6      !PROTECT REGISTERS FROM ...
070760 122 344      PUBD R22,+R6      !  MPYR
070762 170 345      PUMD R70,+R6
070764 160 345      PUMD R60,+R6
070766 150 345      PUMD R50,+R6
070770 114 221      TSM R14      !TEST NUM ROWS
070772 367 336      DOTLOP  JZR DOTEND      !WHILE (NUM ROWS > 0)
070774 132 012 344      PUBD R32,+R12      !  SAVE SIGN OF TEMP SUM
070777 140 345      PUMD R40,+R12      !  SAVE MANT OF TEMP SUM
071001 136 345      PUMD R36,+R12      !  SAVE EXP OF TEMP SUM
071003 316 326 326      JSB =GET2RI      !  GET TWO INTS OR SEP REALS
071006 316 326 326      JSB =MPYR      !  MULTIPLY 2 RLS OR 2 INTS
071011 316 326 326      JSB =RUNSUM      !  CALC RUNNING SUM
071014 316 332 160      JSB =NXTAB      !  POINT TO NEXT A(I,J) & B(J,K)
071017 114 213      DCM R14      !  NUM ROWS = NUM ROWS -1
071021 360 347      JMP DOTLOP      !  LOOP FOR TEST
071023      DOTEND  BSZ 0      !END WHILE
071023 150 006 343      POMD R50,-R6      !RESTORE REGISTERS ...
071026 160 343      POMD R60,-R6
071030 170 343      POMD R70,-R6
071032 122 342      POBD R22,-R6
071034 100 343      POMD R0,-R6
071036 231      BCD      !RESET MODE FOR CALLING ROUTINE.
071037 236      RTN
071040      !
071040      !*****
071040      !*** RUNSUM : ACCUMULATES PRODUCTS OF ELEMENTS OF A & B      ***
071040      !*** IN  : SIGN OF OLD RUNNING SUM      ***
071040      !***      MANTISSA OF OLD RUNNING SUM      ***
071040      !***      EXPONENT OF OLD RUNNING SUM      ***
071040      !***      <--- R12      ***
071040      !***      R40 = ELE OF A * ELE OF B      ***
071040      !*** OUT : STACK POPPED      ***
071040      !***      R32 SIGN OF NEW RUNNING SUM      ***
071040      !***      R40 = MANTISSA OF NEW RUNNING SUM      ***
071040      !***      R36 = EXPONENT OF NEW RUNNING SUM      ***
071040      !*****
071040      !
071040      LST
071040 134 012 343 RUNSUM  POMD R34,-R12      !EXP OF RUNNING SUM.
071043      !          UNL
071043 150 343      POMD R50,-R12      !MANT OF RUNNING SUM.
071045 133 342      POBD R33,-R12      !SGN OF RUNNING SUM.
071047 144 310 377      CMB R44,=377      !INT OR REAL PRODUCT A(I,J)*B(J,K)?
071052 372 336      JNC RLPROD      !JIF REAL.
071054 154 310 377      CMB R54,=377      !INT OR REAL RUNNING SUM?
071057 372 336      JNC RLSUM      !JIF REAL.
071061 235      CLE      !CLEAR FLAG.
071062 234      ICE      !SAYS A(I,J)*B(J,K) & SUM BOTH INT.
071063 360 336      JMP BOTHIN      !GO ADD INT PROD TO INT SUM.
071065 140 060 243 RLSUM  STM R40,R60      !GET READY TO CONV PROD TO REAL.
071070 316 326 326      JSB =INTORL      !CONVERT INTEGER TO REAL.
071073 140 060 241      LDM R40,R60      !MOVE BACK TO R40.
071076 316 326 326      JSB =SEP10      !SEPARATE IT.
071101 360 336      JMP BOTHRL      !FLAG THAT 2 RLS THEN ADD THEM.
071103 154 310 377 RLPROD  CMB R54,=377      !IS RUNNING SUM REAL?
071106 372 336      JNC BOTHRL      !JIF YES, HAVE 2 RLS- ADD THEM.
071110 150 060 243      STM R50,R60      !READY TO CONVERT SUM TO REAL.
071113 136 006 345      PUMD R36,+R6      !PROTECT R36 FROM INTORL.
071116 316 326 326      JSB =INTORL      !CONVERT SUM.
071121 136 006 343      POMD R36,-R6      !RESTORE R36.
071124 150 060 241      LDM R50,R60      !MOVE REAL SUM TO R50.
071127 316 326 326      JSB =SEP20      !SEPARATE IT.
071132 235      BOTHRL  CLE      !FOR ADSU- SAYS ADD 2 REALS.
071133 316 326 326 BOTHIN  JSB =ADSU      !GO DO ADD.
071136 236      RTN
071137      !
071137      !*****
071137      !*** GETS 1 ARRAY ELEMENT EACH FROM A & B AND PREPARES IT ***
071137      !*** FOR MULTIPLICATION.      ***

```



```

071137      !*** IN   : TMP1  = ADDR ELE OF A          ***
071137      !***      TMP2  = ADDR ELE OF B          ***
071137      !*** OUT  : R40   = INTEGER OR SEP REAL ELE OF A      ***
071137      !***      R50    = INTEGER OR SEP REAL ELE OF B      ***
071137      !*****
071137      !
071137      GET2RI    BSZ 0
071137 165 223      CLM R65                                !INDICATE FETCH FROM TMP1
071141 316 326 326 JSB =AFETCH                            !GET VALUE OF A(I,J)
071144 140 012 345 PUMD R40,+R12                          !PUT ON STACK FOR TWOSEP.
071147 165 261 326 BFET-  LDMD R65,=TMP2                 !GET ADDR OF B(J,K)
071152 326
071153 146 260 326 BFETCH  LDBD R46,=TYPB                !GET TYPE OF B FOR FETCH
071156 326
071157 165      DRP R65
071160 316 326 326 JSB =FETCH                            !GET VALUE OF B(J,K)
071163 140 012 345 BFET+  PUMD R40,+R12                  !PUT ON STACK FOR TWOSEP.
071166 231      BCD
071167 316 326 326 JSB =TWOSEP                            !GET TWO INTS OR SEP'D REALS.
071172 236      RTN
071173      !
071173      !*****
071173      !*** GETS 1 ARRAY ELEMENT AND A SCALOR AND PREPARES IT FOR ***
071173      !*** MULTIPLICATION. ***
071173      !*** IN   : TMP1  = ADDR ELE OF A          ***
071173      !***      R70    = SCALOR VALUE            ***
071173      !*** OUT  : SCALOR ***
071173      !***      <--- R12 ***
071173      !***      R40   = INTEGER OR SEP REAL VALUE OF A      ***
071173      !***      R50   = INTEGER OR SEP REAL VALUE OF SCALOR ***
071173      !*****
071173      !
071173      STKAIJ    BSZ 0
071173 165 223      CLM R65                                !INDICATE FETCH FROM TMP1
071175 316 326 326 JSB =AFETCH                            !GET VALUE OF A(I,J)
071200 170 012 345 PUMD R70,+R12                          !SAVE A COPY OF SCL
071203 345      PUMD R70,+R12                          !COPY TO MULTIPLY
071204 360 355      JMP BFET+                            !PUSH & SEPARATE

```

```

071206      !          HED MAT A = IDENTITY ROUTINE
071206      !****  IDN (MAT) ATTRIBUTES TABLE  *****
071206 000 055      BYT 0,55
071210      !*****
071210      IDN10      BSZ 0
071210 316 326 326      JSB =ROMJSB
071213 326 326      DEF IDN2
071215 261      BYT 261
071216 236      RTN
071217      !****  IDN (M) ATTRIBUTES TABLE  *****
071217 020 055      BYT 20,55
071221      !*****
071221 316 326 326 IDIM1V      JSB =DUP1V      !DUP REL ADDR ARRAY ON R12
071224 316 326 326      JSB =RDIM1      !REDIM TARGET ARRAY FIRST.
071227 360 357      JMP IDN10      !NOW GO SET IT=IDN.
071231      !****  IDN (M,N) ATTRIBUTES TABLE  *****
071231 040 055      BYT 40,55
071233      !*****
071233      !***  MAT C = IDN (M,N)      ***
071233      !***      RESULT ARRAY C REDIMENSIONED TO M BY N MATRIX      ***
071233      !***      THEN 1'S ASSIGNED TO ALL DIAGONAL ELEMENTS AND      ***
071233      !***      0'S ASSIGNED TO ALL NON DIAGONAL ELEMENTS.      ***
071233      !***      C MUST BE REDIMENSIONED TO A SQUARE MATRIX (M = N).      ***
071233      !***  IN  : REL ADDR C      ***
071233      !***      SCALOR M      ***
071233      !***      SCALOR N      ***
071233      !***      <--- R12      ***
071233      !***  OUT : STACK POPPED AND ARRAY REDIMENSIONED AND ASSIGNED.      ***
071233      !*****
071233      !
071233 316 326 326 IDIM2V      JSB =DUP2V      !DUP REL ADDR ARRAY ON R12
071236 316 326 326      JSB =RDIM2      !REDIM TARGET ARRAY.
071241 360 345      JMP IDN10      !NOW GO SET IT=IDN.
071243      !
071243      !*****
071243      !***  DUP1V : DUPLICATES ADDRESS OF VECTOR ON R12 STACK      ****
071243      !***  IN  : REL ADDR C      ****
071243      !***      SCALOR M      ****
071243      !***      <--- R12      ****
071243      !***  OUT : REL ADDR C      ****
071243      !***      REL ADDR C      ****
071243      !***      SCALOR M      ****
071243      !***      <--- R12      ****
071243      !*****
071243      !
071243      DUP1V      BSZ 0      !DUP REL ADDR WITH 1 SUBSCRIPT
071243 140 012 343      POMD R40,-R12      !GET SUBSCRIPT
071246 165 343      POMD R65,-R12      !GET REL ADDR ARRAY
071250 345      PUMD R65,+R12      !RESTORE REL ADDR ARRAY
071251 345      PUMD R65,+R12      !ANOTHER COPY
071252 140      DRP R40      !POINT TO SUBSCRIPT
071253 012 345      DUPSH      PUMD R#,+R12      !RESTORE SUBSCRIPT
071255 236      RTN
071256      !
071256      !*****
071256      !***  DUP2V : DUPLICATES ADDRESS OF MATRIX ON R12 STACK      ****
071256      !***  IN  : REL ADDR C      ****
071256      !***      SCALOR M      ****
071256      !***      SCALOR N      ****
071256      !***      <--- R12      ****
071256      !***  OUT : REL ADDR C      ****
071256      !***      REL ADDR C      ****
071256      !***      SCALOR M      ****
071256      !***      SCALOR N      ****
071256      !***      <--- R12      ****
071256      !*****
071256      !
071256      DUP2V      BSZ 0      !DUP REL ADDR ARRAY WITH 2 SUBSCRIPTS
071256 150 012 343      POMD R50,-R12      !GET ONE OF THE SUBSCRIPTS
071261 316 243 162      JSB =DUP1V      !DUP REL ADDR & RESTORE SUBSCRIPT
071264 150      DRP R50      !POINT TO OTHER SUBSCRIPT

```

```

071265 360 364          JMP DUPSH          !RESTORE OTHER SUBSCRIPT
071267          !
071267          EQUA10  BSZ 0
071267 316 326 326      JSB =ROMJSB
071272 326 326          DEF EQUA2
071274 261              BYT 261
071275 236              RTN
071276          !
071276          LST
071276          STOV--  BSZ 0
071276 316 324 161      JSB =RONF5-          !ROUND & PACK IT UP.
071301          STOV    BSZ 0
071301          !
071301          !      UNL
071301          !*****
071301          !* IN : R40/47 - VALUE TO BE STORED          *
071301          !*      TMP4 - ABS ADDR OF ARRAY NAME (3 BYTES) &          *
071301          !*      ABS ADDR OF ARRAY ELE (3 BYTES)          *
071301          !*      TRCFLG - BIT 4 = 0 - NO TRACE          *
071301          !*      BIT 4 = 1 - TRACE          *
071301          !*      BIT 0 = 0 - MATRIX          *
071301          !*      BIT 0 = 1 - VECTOR          *
071301          !* OUT: VALUE IN R40/47 STORED AT ADDR IN TMP4          *
071301          !*      TMP4 = NAME & ABS ADDR OF NEXT ARRAY ELE          *
071301          !*****
071301          !
071301 230              BIN          !SET MODE
071302 316 326 326      JSB =PUTREG          !PROTECT REGS FROM PRINT DRIVER
071305 160 261 326      LDMD R60,=TMP4          !POINT TO CURRENT ARRAY ELE
071310 326
071311 165 012 345      PUMD R65,+R12          !ABS ADDR 1ST ARRAY ELE ON STACK
071314 155 261 326      LDMD R55,=INCR          !GET ELEMENT SIZE
071317 326
071320 316 001 161      JSB =NXTELE          !POINT TO NEXT ARRAY ELE
071323 160 263 326      STMD R60,=TMP4          !SAVE POINTER
071326 326
071327 156 250 375      LDB R56,=375          !ASSUME NO TRACE MASK
071332 157 260 326      LDBD R57,=TRCFLG          !TRACING C?
071335 326
071336 374 336          JLZ NOTRC          !IF (TRACE FLAG SET)
071340 156 250 377      LDB R56,=377          ! SET TRACE MASK
071343 130 251 377      LDM R30,=377,377
071346 377
071347 012 345          PUMD R30,+R12          ! PUSH ROW FOR TRACE
071351 345              PUMD R30,+R12          ! PUSH COL FOR TRACE
071352 157 317 357      ANM R57,=357          ! CLEAR TRACE BUT NOT VECTOR INFO
071355 262 326 326      STBD R57,=TRCFLG          ! RESTORE FLAG WITH VECTOR INFO
071360 344              PUBD R57,+R12          ! ODD=VECTOR ; EVEN=MATRIX
071361          NOTRC  BSZ 0          !END IF
071361 160 012 345      PUMD R60,+R12          !ABS ADDR OF ARRAY NAME ON STACK
071364 163 343          POMD R63,-R12          !MOVE POINTER BACK TO ARRAY NAME
071366 156 327 326      ANMD R56,=TMP1++          !GET HEADER WITH TRACE ON/OFF
071371 326
071372 344              PUBD R56,+R12          !PUSH HEADER ON STACK
071373 140 345          PUMD R40,+R12          !ARRAY ELE VALUE ON STACK
071375          !
071375 316 326 326      JSB =ROMJSB          !GOING TO STOSV.
071400 326 326          DEF STOSV
071402 000              BYT 0
071403 316 326 326      JSB =GETREG          !RESTORE CPU REGISTERS.
071406 236              RTN
071407          !
071407          AFETCH BSZ 0
071407 232              SAD          !SAVE DRP, ARP
071410 146 260 326      LDBD R46,=TYPA          !GET MATRIX TYPE
071413 326
071414 237              PAD          !RESTORE DRP, ARP
071415 221              TSM R#          !TEST DRP REGISTER
071416 366 336          JNZ FETCH          !JIF FETCH ADDR ALREADY HERE
071420 261 326 326      LDMD R#,=TMP1          !ELSE GET FETCH ADDR
071423          !

```

```

071423      FETCH      BSZ 0
071423      !
071423      !*****
071423      !* IN : R# - 3 BYTE ABS ADDR OF ARRAY ELE TO BE FETCHED *
071423      !*      R46 - TYPE OF ARRAY *
071423      !* OUT: R40/47 - VALUE OF ARRAY ELEMENT *
071423      !*****
071423      !
071423 263 326 326      STMD R#,,=PTR2      !SET UP FETCH ADDR
071426      LST
071426 316 326 326  FETCH-  JSB =PUTREG      !SAVE REGISTER FROM PRINT DRIVER
071431      !      UNL
071431 316 326 326      JSB =ROMJSB
071434 326 326      DEF FETSVX      !GET 8 BYTE VALUE IN R60
071436 000      BYT 0
071437 160 040 243      STM R60,R40      !MOVE FETCHED VALUE TO R40
071442 316 326 326      JSB =GETREG      !RESTORE REGISTERS
071445 230      BIN
071446 236      RTN
071447      !
071447      !*****  MAT RUNTIME  *****
071447 241      BYT 241
071450      !*****
071450 170 223  MAT.      CLM R70
071452 360 336      JMP SAVSIZ

```

```

071454      !          HED MAT C(I:J,K:L) = A(I:J,K:L) ROUTINE
071454      !**** DUMMY= ATTRIBUTES TABLE *****
071454 005 051      BYT 5,51
071456      !*****
071456 236      DUMMY= RTN
071457      !***** C(I) ATTRIBUTES TABLE *****
071457 044      BYT 44
071460      !*****
071460 316 326 326 CI.      JSB =ONEB          !GET I.
071463 146      DRP R46
071464 316 326 326 CI+      JSB =ADOP-          !I+OPTBAS, J+OPTBAS.
071467 370 336      JEN CIOK          !JIF C IS A VECTOR.
071471 104 251 377 RDR33      GTO RDER33          !ATTEMPT TO CHANGE # DIMS ERROR.
071474 377
071475      !***** C(I:J) ATTRIBUTES TABLE *****
071475 032      BYT 32
071476      !*****
071476 316 326 326 CIJ.      JSB =TWOB          !GET I AND J.
071501 156      DRP R56          !POINT TO I.
071502 360 360      JMP CI+
071504      !***** C(I:J,) ATTRIBUTES TABLE *****
071504 032      BYT 32
071505      !*****
071505 143 012 343 CIJ*.      POMD R43,-R12          !TRASH NULL STRING.
071510 316 326 326      JSB =TWOB          !GET I AND J.
071513 156      DRP R56          !POINT TO I.
071514 360 336      JMP CI*+
071516 174 223      CIOK      CLM R74          !READY FOR K AND L.
071520 210      ICB R74          !K=1.
071521 176 210      ICB R76          !L=1.
071523 316 326 326 CKROW      JSB =CKRSIZ          !ROW NO.S IN BOUNDS?
071526 136 261 326 SAVSIZ      LDMD R36,=MBASE          !TEMP AREA BASE.
071531 326
071532 170 036 267      STMD R70,X36,ASTATS          !SAVE I, J, K, L.
071535 101 000
071537 236      RTN
071540      !***** C(I:J,K) ATTRIBUTES TABLE *****
071540 044      BYT 44
071541      !*****
071541 316 326 326 CIJK.      JSB =ONEB          !GET K.
071544 146      DRP R46          !POINT TO I.
071545 074 243      CIJK+      STM R#,R74          !SAVE K.
071547 146 076 243      STM R46,R76          !SAVE L (OR K).
071552 174 006 345      PUMD R74,+R6          !PROTECT R74, R76.
071555 316 326 326      JSB =TWOB          !GET I AND J.
071560 174 006 343      POMD R74,-R6          !RETRIEVE R74, R76.
071563 156      DRP R56          !POINT TO I.
071564 360 336      JMP CIK++
071566      !***** C(I:J,K:L) !ATTRIBUTES TABLE *****
071566 032      BYT 32
071567      !*****
071567 316 326 326 CIJKL.      JSB =TWOB          !GET K AND L.
071572 156      DRP R56
071573 360 350      JMP CIJK+
071575      !***** C(I,) ATTRIBUTES TABLE *****
071575 044      BYT 44
071576      !*****
071576 143 012 343 CI*.      POMD R43,-R12          !TRASH NULL STRING.
071601 316 326 326      JSB =ONEB          !GET I.
071604 146      DRP R46          !POINT TO I.
071605 316 326 326 CI*+      JSB =ADOP-          !I+OPTBAS, J+OPTBAS.
071610 370 257      CI*++      JEN RDR33          !JIF C IS ONEDIM.
071612 174 223      CLM R74          !READY FOR K AND L.
071614 210      ICB R74          !K=1.
071615 176 024 241      LDM R76,R24          !L=N.
071620 360 301      RELAYR      JMP CKROW          !ROW NO.S IN BOUNDS?
071622      !***** C(,K) ATTRIBUTES TABLE *****
071622 044      BYT 44
071623      !*****
071623 316 326 326 CK.      JSB =ONEB          !GET K.
071626 146      DRP R46          !POINT TO K.

```

```

071627 074 243      CK+      STM R#,R74      !SAVE K.
071631 146 076 243      STM R46,R76      !SAVE L (OR K).
071634 163 012 343      POMD R63,-R12      !TRASH NULL STRING.
071637 316 326 326      JSB =ADDOPT      !K+OPTBAS, L+OPTBAS.
071642 370 225      JEN RDR33      !ERROR IF C IS A VECTOR.
071644 122 072 243 CK++      STM R22,R72      !J=M.
071647 170 250 001      LDB R70,=1      !I=1.
071652 171 222      CLB R71
071654 316 326 326      JSB =CKCSIZ      !COL NO.S IN BOUNDS?
071657 360 245      JMP SAVSIZ      !SAVE RESULTS.
071661 316 326 326 ASUB      JSB =MINFO      !Ba, Ma, Na, VECa TO R12.
071664 174 223      CLM R74      !READY FOR K=1.
071666 210      ICB R74      !K=1.
071667 124 076 243      STM R24,R76      !L = Na.
071672 360 350      JMP CK++      !I=1, J=MA, & STORE ASTATS.
071674      !***** C(,K:L) ATTRIBUTES TABLE *****
071674 032      BYT 32
071675      !*****
071675 316 326 326 CKL.      JSB =TWOB      !GET K AND L.
071700 156      DRP R56      !POINT TO K.
071701 360 324      JMP CK+
071703      !***** C(I,K) ATTRIBUTES TABLE *****
071703 032      BYT 32
071704      !*****
071704 316 326 326 CIK.      JSB =ONEB      !GET K.
071707 146      DRP R46      !POINT TO K.
071710 074 243      CIK+      STM R#,R74      !SAVE K.
071712 146 076 243      STM R46,R76      !SAVE L (OR K).
071715 174 006 345      PUMD R74,+R6      !PROTECT R74, R76.
071720 316 326 326      JSB =ONEB      !GET I.
071723 174 006 343      POMD R74,-R6      !RETRIEVE R74, R76.
071726 146      CIK+-      DRP R46      !POINT TO I.
071727 316 326 326 CIK++      JSB =ADOP-      !I+OPTBAS, J (OR I)+OPTBAS.
071732 371 336      JEZ CIKOK      !JIF C IS TWODIM.
071734 360 252      JMP CI+++      !ERROR IF C IS A VECTOR.
071736 316 326 326 CIKOK      JSB =CKCSIZ      !COL NO.S IN BOUNDS?
071741 360 255      JMP RELAYR      !GO CHECK ROW NO.S.
071743      !***** C(I,K:L) ATTRIBUTES TABLE *****
071743 044      BYT 44
071744      !*****
071744 316 326 326 CIKL.      JSB =TWOB      !GET K AND L.
071747 156      DRP R56
071750 360 336      JMP CIK+
071752 070 243      ADOP-      STM R#,R70      !SAVE I.
071754 146 072 243      STM R46,R72      !SAVE J.
071757 316 326 326 ADDOPT      JSB =MINFO      !BASE, M, N, OPTION BASE, VECINFO.
071762 150 223      CLM R50      !READY FOR COPIES OF OPTBAS.
071764 136 215      TCM R36      !COMP OPTION BASE.
071766 211      ICM R36      !GIVES OPTBAS.
071767 026 243      STM R36,R26      !SAVE OPTBAS.
071771 170 030 243      STM R70,R30      !COPY I, J, K, AND L.
071774 130 026 303      ADM R30,R26      !I+OPTBAS.
071777 132 303      ADM R32,R26      !J+OPTBAS.
072001 134 303      ADM R34,R26      !K+OPTBAS.
072003 136 303      ADM R36,R26      !L+OPTBAS.
072005 170 030 241      LDM R70,R30      !MOVE RESULTS BACK TO R70.
072010 236      RTN
072011 235      MINFO      CLE      !FOR ARRAY A (LAST ARRAY THRU).
072012 165 012 343 MINFO+      POMD R65,-R12      !ARRAY OFFSET.
072015 345      PUMD R65,+R12      !COPY IT BACK FOR LATER.
072016 316 326 326      JSB =RUDIM      !GET M, N, OPTION BASE.
072021 230      BIN      !SET MODE.
072022 150 020 241      LDM R50,R20      !MOVE BASE, M, N, VEC FLAG.
072025 156 223      CLM R56      !ASSUME C NOT A VECTOR.
072027 371 336      JEZ MPUSH      !PUSH OUT MATRIX INFO.
072031      !
072031 212      DCB R56      !FLAG C AS A VECTOR.
072032 150 012 345 MPUSH      PUMD R50,+R12      !SAVE FOR WHEN GET TO ACOM.
072035 236      RTN
072036      !*
072036 130 070 241 CKRSIZ      LDM R30,R70      !I.

```

```

072041 132 072 241          LDM R32,R72          !J.
072044 122 024 243          STM R22,R24          !MOVE M TO R24.
072047 360 336              JMP CKSIZ+
072051 174 030 243 CKCSIZ  STM R74,R30          !MOVE K AND L.
072054 132 221          CKSIZ+ TSM R32          !NEED L>=0.
072056 364 336              JNG BNDERR          !JIF NEG.
072060 130 221          TSM R30          !NEED K>=0.
072062 364 336              JNG BNDERR          !JIF NEG.
072064 311 001 000          CMM R30,=1,0        !NEED K>=1.
072067 373 336              JCY KOK            !JIF K IS OK.
072071 316 326 326 BNDERR  JSB =ERROR
072074 067                  BYT 55D
072075 236                  RTN
072076 124 032 301 KOK      CMM R24,R32          !N>=L?
072101 372 366              JNC BNDERR          !ERROR IF NOT.
072103 132 030 301          CMM R32,R30          !L>=K?
072106 372 336              JNC CKREV          !JIF NO.
072110 311 001 000          CMM R32,=1,0        !L>=1?
072113 372 354              JNC BNDERR          !ERROR IF NO.
072115 124 030 301 CKK      CMM R24,R30          !N>=K?
072120 372 347              JNC BNDERR          !ERROR IF NO.
072122 236                  RTN
072123 124 211          CKREV  ICM R24          !N+1.
072125 360 366              JMP CKK
072127          !***** A ATTRIBUTES !TABLE *****
072127 005 051              BYT 5,51
072131          !*****
072131 136 251 261 A.       LDM R36,=ASUB
072134 163
072135 360 336              JMP ACOM
072137          !***** A(I) ATTRIBUTES TABLE *****
072137 044              BYT 44
072140          !*****
072140 136 251 060 AI.       LDM R36,=CI.          !ASTATS, Ba, Ma, Na, VECa PTR.
072143 163
072144 360 336              JMP ACOM
072146          !***** A(I:J) ATTRIBUTES TABLE *****
072146 032              BYT 32
072147          !*****
072147 136 251 076 AIJ.       LDM R36,=CIJ.          !ASTATS, Ba, Ma, Na, VECa PTR.
072152 163
072153 360 336              JMP ACOM
072155          !***** A(I:J,) ATTRIBUTES TABLE *****
072155 032              BYT 32
072156          !*****
072156 136 251 105 AIJ*.      LDM R36,=CIJ*.          !ASTATS, Ba, Ma, Na, VECa PTR.
072161 163
072162 360 336              JMP ACOM
072164          !***** A(I:J,K) ATTRIBUTES TABLE *****
072164 044              BYT 44
072165          !*****
072165 136 251 141 AIJK.      LDM R36,=CIJK.          !ASTATS, Ba, Ma, Na, VECa PTR.
072170 163
072171 360 336              JMP ACOM
072173          !***** A(I:J,K:L) !ATTRIBUTES TABLE *****
072173 032              BYT 32
072174          !*****
072174 136 251 167 AIJKL.     LDM R36,=CIJKL.          !ASTATS, Ba, Ma, Na, VECa PTR.
072177 163
072200 360 336              JMP ACOM
072202          !***** A(,K) ATTRIBUTES TABLE *****
072202 044              BYT 44
072203          !*****
072203 136 251 223 AK.        LDM R36,=CK.          !ASTATS, Ba, Ma, Na, VECa PTR.
072206 163
072207 360 336              JMP ACOM
072211          !***** A(,K:L) ATTRIBUTES TABLE *****
072211 032              BYT 32
072212          !*****
072212 136 251 275 AKL.       LDM R36,=CKL.          !ASTATS, Ba, Ma, Na, VECa PTR.
072215 163

```

```

072216 360 336          JMP ACOM
072220          !***** A(I,) ATTRIBUTES TABLE *****
072220 044          BYT 44
072221          !*****
072221 136 251 176 AI*.    LDM R36,=CI*.          !ASTATS, Ba, Ma, Na, VECa PTR.
072224 163
072225 360 336          JMP ACOM
072227          !***** A(I,K) ATTRIBUTES TABLE *****
072227 032          BYT 32
072230          !*****
072230 136 251 304 AIK.    LDM R36,=CIK.          !ASTATS, Ba, Ma, Na, VECa PTR.
072233 163
072234 360 336          JMP ACOM
072236          !***** A(I,K:L) ATTRIBUTES TABLE *****
072236 044          BYT 44
072237          !*****
072237 136 251 344 AIKL.  LDM R36,=CIKL.          !ASTATS, Ba, Ma, Na, VECa PTR.
072242 163
072243 126 261 326 ACOM   LDMD R26,=MBASE          !TEMP AREA BASE.
072246 326
072247 170 026 265          LDMD R70,X26,ASTATS    !GET CSTATS.
072252 101 000
072254 006 345          PUMD R70,+R6              !SAVE CSTATS.
072256 036 306 000        JSB X36,ZERO            !FIND ASTATS.
072261 000
072262 230              BIN                      !SET MODE.
072263 140 012 343        POMD R40,-R12            !GET Ba, Ma, Na, aVEC.
072266 165 343          POMD R65,-R12            !TRASH A OFFSET.
072270 170 006 343        POMD R70,-R6            !GET CSTATS.
072273 366 336          JNZ ACOM+                !JIF CSTATS KEYED IN.
072275 140 345          PUMD R40,+R6              !PROTECT R40 FROM RUDIM IN MINFO.
072277 235              CLE
072300 233              DCE                      !SAYS DOING TARGET ARRAY C.
072301 316 012 164        JSB =MINFO+            !GENERATE CSTATS.
072304 140 006 343        POMD R40,-R6            !RECOVER R40.
072307 126 261 326 ACOM+ LDMD R26,=MBASE          !TEMP AREA BASE.
072312 326
072313 150 012 343        POMD R50,-R12            !GET Bc, Mc, Nc, cVEC.
072316 160 026 265        LDMD R60,X26,ASTATS    !IA, JA, KA, LA.
072321 101 000
072323 316 326 326        JSB =SIZE              !FIND # A ROWS TO MOVE.
072326 022 243          STM R#,R22              !# A ROWS TO MOVE.
072330 164              DRP R64
072331 316 326 326        JSB =SIZE              !FIND # A COLS TO MOVE.
072334 024 243          STM R#,R24              !# A COLS TO MOVE.
072336 170 221          TSM R70
072340 367 336          JZR CREDIM              !JIF 0 - GO REDIM C.
072342 316 326 326        JSB =SIZE              !FIND # ROWS C SPECIFIED.
072345 036 243          STM R#,R36              !SAVE # ROWS C SPECIFIED.
072347 174              DRP R74
072350 316 326 326        JSB =SIZE              !FIND # COLS C SPECIFIED.
072353 024 301          CMM R#,R24              !COLS C = COLS A?
072355 366 336          JNZ UNEQUL              !JIF NO.
072357 122 036 301        CMM R22,R36            !ROWS A = ROWS C?
072362 367 336          JZR EQUAL              !JIF YES, SIZES MATCH.
072364 156 046 300 UNEQUL CMB R56,R46          !BOTH VECs OR MATS?
072367 367 336          JZR FAIL                !JIF YES - SIZE MISMATCH.
072371 132 022 301        CMM R32,R22            !COLS C = ROWS A?
072374 366 336          JNZ FAIL                !JIF #, NO CHANCE OF MATCHING.
072376 136 024 301        CMM R36,R24            !ROWS C = COLS A?
072401 366 336          JNZ FAIL                !JIF NO - #VECELTS # #MATELTS.
072403 156 220          TSB R56                !IS C THE VECTOR?
072405 367 336          JZR ASWAP              !JIF NO, SWAP ASTATS.
072407 174 006 345        PUMD R74,+R6            !SWAP CSTATS I:J & K:L TO MAKE
072412 170 345          PUMD R70,+R6            !INCRS COME OUT RIGHT IN
072414 174 343          POMD R74,-R6            !NEWBAS.
072416 170 343          POMD R70,-R6
072420 360 336          JMP EQUAL              !CONTINUE.
072422 164 006 345 ASWAP  PUMD R64,+R6            !SWAP ASTATS I:J & K:L TO MAKE
072425 160 345          PUMD R60,+R6            !INCRS COME OUT RIGHT IN
072427 164 343          POMD R64,-R6            !NEWBAS.

```



072431	160	343		POMD R60,-R6	
072433	136	022	243	STM R36,R22	!EFFECTIVELY SWAPPING R22 & R24.
072436	132	024	243	STM R32,R24	!FINISH SWAPPING R22 & R24.
072441	122	006	345	EQUAL PUMD R22,+R6	!SAVE # A ROWS.
072444	223			CLM R22	!LIKE RUDIM DOES.
072445	213			DCM R22	
072446	124	345		PUMD R24,+R6	!SAVE # A COLS.
072450	223			CLM R24	!LIKE RUDIM DOES.
072451	360	336		JMP CRED+	!FIND Bc, Mc, Nc, TPC, INCR.
072453	210		CREDIM	ICB R#	!IC=1.
072454	174	210		ICB R74	!KC=1.
072456	122	072	243	STM R22,R72	!JC = # ROWS TO MOVE.
072461	006	345		PUMD R22,+R6	!SAVE # ROWS TO MOVE.
072463	124	345		PUMD R24,+R6	!SAVE # COLS TO MOVE.
072465	076	243		STM R24,R76	!LC=# COLS TO MOVE.
072467	156	046	300	CMB R56,R46	!VEC-VEC OR MAT-MAT?
072472	367	336		JZR CRED	!JIF YES, READY TO REDIM C.
072474	220			TSB R#	!IS C THE VECTOR?
072475	367	336		JZR CRED	!JIF NO, A IS, GO REDIM C.
072477	124	311	001	CMM R24,=1,0	!ONE OF THE DIMS MUST = 1.
072502	000				
072503	367	336		JZR CRED	!JIF # COLS = 1, READY TO REDIM.
072505	122	311	001	CMM R22,=1,0	!# ROWS = 1?
072510	000				
072511	367	336		JZR SZOKAY	!JIF ROWS=1 OR COLS=1.
072513	144	006	343	SZFAIL POMD R44,-R6	!CLEAN UP R6.
072516	316	313	160	FAIL JSB =MISMAT	!SIZE MISMATCH ERROR EXIT.
072521	122	014	243	SZOKAY STM R22,R14	!COLS=1 THEN REDIM VECTOR C.
072524	024	241		LDM R22,R24	
072526	114	243		STM R14,R24	!FINISHED SWAPPING.
072530	124	054	243	CRED STM R24,R54	!NEW Nc (COL SIZE) TO R54.
072533	120	012	343	CRED+ POMD R20,-R12	!GET C OFFSET.
072536	127	342		POBD R27,-R12	
072540	140	345		PUMD R40,+R12	!PROTECT Ma, Na, aVEC.
072542	150	345		PUMD R50,+R12	!PROTECT Mc, Nc, cVEC.
072544	160	345		PUMD R60,+R12	!PROTECT ASTATS FROM REDIM.
072546	120	066	243	STM R20,R66	!STORE BC
072551	127	065	242	STB R27,R65	
072554	235			CLE	!CLEAR ARRAY SPECIFIER FLAG.
072555	233			DCE	!SPECIFIES TARGET=C IN REDIM.
072556	316	326	326	JSB =REDIM.	!SET INCR, TPC, & MAY REDIM C.
072561	316	326	326	JSB =VECFLG	!C VECTOR INFO.
072564	160	012	343	POMD R60,-R12	!RECOVER ASTATS.
072567	150	343		POMD R50,-R12	!RECOVER Mc, Nc, cVEC.
072571	140	343		POMD R40,-R12	!RECOVER Ma, Na, aVEC.
072573	114	261	326	LDMD R14,=INCR	!NEEDED IN NEWBAS FOR NEW Bc.
072576	326				
072577	120	222		CLB R20	!FLAG STORE ADDR
072601	316	326	326	JSB =NEWBAS	!CALC 1ST STORE LOCATION (NEW Bc).
072604	114	261	326	LDMD R14,=INCRA	!FOR NEWBAS.
072607	326				
072610	160	070	243	STM R60,R70	!MOVE ASTATS FOR NEWBAS.
072613	140	050	243	STM R40,R50	!OTHER A INFO FOR NEWBAS.
072616	120	210		ICB R20	!FLAG FETCH ADDR
072620	316	326	326	JSB =NEWBAS	!CALC 1ST FETCH LOCATION.
072623	124	006	343	POMD R24,-R6	!# ROWS TO MOVE.
072626	122	343		POMD R22,-R6	!# COLS TO MOVE.
072630	316	262	152	JSB =ERRCK	!ERROR OUT IF REDIM ERROR.
072633	126	223		CLM R26	!NO TEMP AREA USED IF STAYS 0.
072635	165	261	326	LDMD R65,=TMP1	!GET Ba
072640	326				
072641	321	326	326	CMMD R65,=TMP4+	!COMPARE Ba & Bc
072644	366	336		JNZ TRANSF	!IF ( C = A)
072646	130	261	326	LDMD R30,=INCR	! GET ELE SIZE OF C
072651	326				
072652	316	326	326	JSB =NUMBYT	! FIND NUM BYTES IN ARRAY
072655	316	326	326	JSB =RESMEM	! MEMORY FOR SCRATCH ARRAY
072660	370	336		JEN RETURN	! IF (NO ROOM) RETURN
072662	126	210		ICB R26	! FLAG TEMP MEMR USED
072664	141	012	343	POMD R41,-R12	! GET A INFO OFF STACK.
072667	171	343		POMD R71,-R12	! GET C INFO OFF STACK.

072671	345		PUMD R71,+R12	!	PUT C INFO BACK ON STACK
072672	165	071	STM R65,R71	!	MOVE TEMP AREA BASE Bt
072675	171	012	PUMD R71,+R12	!	PUT T INFO ON STACK.
072700	345		PUMD R71,+R12	!	T INFO AGAIN FOR FINAL MOVE.
072701	141	345	PUMD R41,+R12	!	REPLACE A INFO.
072703			BSZ 0	!	END IF
072703	126	006	PUMD R26,+R6	!	SAVE FLAG.
072706	316	326	JSB =MOVE	!	MOVE DATA, FINALLY.
072711	126	006	POMD R26,-R6	!	GET FLAG.
072714	367	336	JZR RETURN	!	IF (FLAG <> 0)
072716	316	326	JSB =TRCRST	!	RESET TRACE FLAG
072721	360	336	JMP MOVE	!	COPY SCRATCH ARRAY INTO C
072723			BSZ 0	!	END IF
072723	236		RTN		
072724			!		
072724	030	243	SIZE STM R#,R30	!	WORKING COPY OF I (OR K).
072726	132	211	ICM R32	!	J+1 (OR L+1).
072730	301		CMM R32,R30	!	NO MOVING IF I=J+1 OR K=L+1.
072731	373	336	JCY JBIGR	!	JIF J+1>=I (OR L+1>=K).
072733	130	006	PUMD R30,+R6	!	TEMP PUSH WHILE SWAP.
072736	032	241	LDM R30,R32	!	SWAP J+1 AND I OR L+1 AND K.
072740	132	006	POMD R32,-R6	!	FINISH SWAP.
072743	030	305	JBIGR SBM R#,R30	!	CALC # ROWS (OR COLS) TO MOVE.
072745	236		RTN		
072746			!		
072746	170	030	NEWBAS STM R70,R30	!	MOVE I, J, K, L.
072751	166	006	PUMD R66,+R6	!	TEMP SAVE LA.
072754	070	241	LDM R66,R70	!	IC (OR IA).
072756	213		DCM R66	!	IC-1.
072757	176	054	LDM R76,R54	!	Nc (OR Na).
072762	006	345	PUMD R76,+R6	!	NEEDED FOR ROW INCR CALC.
072764	114	000	STM R14,R0	!	ANOTHER COPY OF INCR (INCRA).
072767	132	030	CMM R32,R30	!	J>=I?
072772	373	336	JCY JGEI	!	JIF YES.
072774	166	213	DCM R66	!	ADJUST INIT COL.
072776	114	215	TCM R14	!	GIVES -INCR (OR -INCRA).
073000	316	326	JGEI JSB =MNMUL3	!	Nc*(IC-1).
073003	176	055	LDM R76,R55	!	MOVE PRODUCT.
073006	213		DCM R76	!	Nc*(IC-1)-1.
073007	074	303	ADM R76,R74	!	Nc*(IC-1)-1+KC.
073011	166	000	LDM R66,R0	!	INCR (OR INCRA).
073014	136	034	CMM R36,R34	!	L>=K?
073017	373	336	JCY LGEK	!	JIF YES.
073021	176	213	DCM R76	!	ADJUST INIT ROW.
073023	100	215	TCM R0	!	GIVES -INCR (OR -INCRA).
073025	316	326	LGEK JSB =MNMUL3	!	(Nc*(IC-1)-1+KC)*INCR.
073030	175	261	LDMD R75,=TMP4+	!	GET STORE ADDR
073033	326				
073034	120	220	TSB R20	!	STORE OR FETCH ?
073036	367	336	JZR LGEK1	!	JIF STORE
073040	175	261	LDMD R75,=TMP1	!	GET FETCH ADDR
073043	326				
073044			LGEK1 BSZ 0		
073044	175	055	305 SBM R75,R55	!	Bc-(Nc*(IC-1)-1+KC)*INCR.
073047	012	345	PUMD R75,+R12	!	SAVE NEW Bc (OR Bc).
073051	166	014	241 LDM R66,R14	!	MOVE +-INCR.
073054	176	006	343 POMD R76,-R6	!	Nc.
073057	316	326	326 JSB =MNMUL3	!	+INCR*Nc.
073062	166	006	343 POMD R66,-R6	!	RESTORE LA.
073065	100	012	345 PUMD R0,+R12	!	SAVE +-INCR (OR+-INCRA).
073070	316	326	326 JSB =SHIF54	!	MOVE PRODUCT.
073073	156	012	345 PUMD R56,+R12	!	SAVE ON STACK.
073076	236		RTN		
073077			!		
073077			!*****		
073077			!*** MOVE : MOVES A GROUP OF ELEMENTS FROM ARRAY A INTO A SECTION ***		
073077			!*** OF ARRAY C.		***
073077			!*** IN : 1st STORE ADDR (C)		***
073077			!*** STORE COL INCR (C)		***
073077			!*** STORE ROW INCR (C)		***
073077			!*** 1st FETCH ADDR (A)		***

```

073077      !***      FETCH COL INCR (A)      ***
073077      !***      FETCH ROW INCR (A)      ***
073077      !***      <--- R12      ***
073077      !***      R22 = Mc : NUM ROWS OF C      ***
073077      !***      R24 = Nc : NUM COLS OF C      ***
073077      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED.      ***
073077      !*****
073077      !
073077      MOVE      BSZ 0
073077 174 012 343      POMD R74,-R12      !FETCH ROW & COL INCREMENTS.
073102 155 343      POMD R55,-R12      !1st FETCH ADDRESS (NEW Ba).
073104 144 343      POMD R44,-R12      !STORE ROW & COL INCREMENTS.
073106 070 243      STM R44,R70      !MOVE STORE INCREMENTS.
073110 155 012 345      PUMD R55,+R12      !SAVE 1st FETCH ADDR
073113 122 036 243      STM R22,R36      !ROWS = Mc
073116 367 336      BEGROW      JZR MOEND1      !WHILE (ROWS <> 0)
073120 155 012 343      POMD R55,-R12      ! PTR FETCH ROW
073123 165 343      POMD R65,-R12      ! PTR STORE ROW
073125 345      PUMD R65,+R12      ! SAVE CURRENT STORE ROW
073126 155 345      PUMD R55,+R12      ! SAVE CURRENT FETCH ROW
073130 124 030 243      STM R24,R30      ! COLS = Nc
073133 367 336      MOVLOP      JZR MOEND2      ! WHILE (COLS <> 0)
073135 155      DRP R55      ! PTR TO FETCH ADDR
073136 316 007 163      JSB =AFETCH      ! GET VALUE TO MOVE.
073141 165 012 345      PUMD R65,+R12      ! SAVE STORE ADDR
073144 263 326 326      STMD R65,=TMP4+      ! SET STORE ADDR
073147 316 301 162      JSB =STOV      ! STORE VALUE IN C OR T
073152 165 012 343      POMD R65,-R12      ! GET OLD STORE ADDR
073155 132 074 241      LDM R32,R74      ! GET FETCH ROW INCR ...
073160 316 326 326      JSB =INCTST      ! AND TEST IT
073163 155 032 305      SBM R55,R32      ! NEXT FETCH ADDR
073166 132 070 241      LDM R32,R70      ! GET STORE ROW INCR ...
073171 316 326 326      JSB =INCTST      ! AND TEST IT
073174 165 032 305      SBM R65,R32      ! NEXT STORE ADDR
073177 130 213      DCM R30      ! COLS = COLS - 1
073201 360 330      JMP MOVLOP      ! LOOP
073203      MOEND2      BSZ 0      ! END WHILE
073203 132 076 241      LDM R32,R76      ! GET FETCH COL INCR ...
073206 316 326 326      JSB =INCTST      ! AND TEST IT
073211 155 012 343      POMD R55,-R12      ! GET CURRENT FETCH ROW.
073214 032 305      SBM R55,R32      ! NEXT FETCH ROW
073216 132 072 241      LDM R32,R72      ! STORE COL INCR
073221 316 326 326      JSB =INCTST      ! AND TEST IT
073224 165 012 343      POMD R65,-R12      ! CURRENT STORE ROW.
073227 032 305      SBM R65,R32      ! NEXT STORE ROW
073231 012 345      PUMD R65,+R12      ! SAVE STORE ROW
073233 155 345      PUMD R55,+R12      ! SAVE FETCH ROW
073235 136 213      DCM R36      ! ROWS = ROWS - 1
073237 360 255      JMP BEGROW      ! LOOP
073241      MOEND1      BSZ 0      !END WHILE
073241 162 012 343      POMD R62,-R12      !CLEAN UP R12 STACK
073244 236      RTN
073245      !
073245      INCTST      BSZ 0
073245 134 222      CLB R34      !ASSUME INCREMENT > 0
073247 132 221      TSM R32      !TEST INCREMENT
073251 365 336      JPS ENDINC      !IF (INCREMENT < 0)
073253 134 212      DCB R34      ! MSB NEGATIVE
073255      ENDINC      BSZ 0      !END IF
073255 236      RTN
073256      !
073256      !***** LINK NEXT SEGMENT *****

```

```

073256      !! updated JFG March2020
073256      !! to match Matrix ROM 1 00087-15004 REV.A
073256      !          HED MAT C = B*TRN(A), TRN(B)*A ROUTINES
073256      !***** ATTRIBUTES TABLE *****
073256 011 051          BYT 11,51
073260      !*****
073260      !*** MAT C = B * TRN(A) ***
073260      !***      MULTIPLY AN ARRAY BY THE TRANSPOSE OF AN ARRAY ***
073260      !*** IN   : REL ADDR C ***
073260      !***      REL ADDR B ***
073260      !***      REL ADDR A ***
073260      !***      <--- R12 ***
073260      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
073260      !***      (AND REDIMENSIONED IF NECESSARY) ***
073260      !*****
073260      !
073260      B*TRN.      BSZ 0
073260 170 222          CLB R70
073262 210          ICB R70          !FLAG B*TRN(A)
073263 360 336          JMP MAT-          !FIND B*TRN(A)
073265      !
073265      !***** ATTRIBUTES TABLE *****
073265 011 051          BYT 11,51
073267      !*****
073267      !*** MAT C = TRN(B) * A ***
073267      !***      MULTIPLY A TRANPOSE OF AN ARRAY BY AN ARRAY ***
073267      !*** IN   : REL ADDR C ***
073267      !***      REL ADDR B ***
073267      !***      REL ADDR A ***
073267      !***      <--- R12 ***
073267      !*** OUT : STACK POOPED AND RESULT ARRAY ASSIGNED ***
073267      !***      (AND REDIMENSIONED IF NECESSARY) ***
073267      !*****
073267      !
073267      TRN*A.      BSZ 0
073267 170 250 002      LDB R70,=2          !FLAG TRN(B)*A
073272 360 336          JMP MAT-          !FIND TRN(B)*A
073274      !

```

```

073274      !          HED MATRIX C = BA MULTIPLY ROUTINE
073274      !****  MAT MUL ATTRIBUTES TABLE  ****
073274  011 051      BYT 11,51
073276      !*****
073276      !*** MAT C = B * A ***
073276      !***      CALCULATES THE PRODUCT OF TWO ARRAYS.  THE VALUE ***
073276      !***      OF EACH ELEMENT OF THE RESULT ARRAY IS DETERMINED ***
073276      !***      ACCORDING TO THE USUAL RULES OF MATRIX MULTIPLICATION. ***
073276      !***      Nb ***
073276      !***      C(I,K) = Sum B(I,J) * A(J,K) ***
073276      !***      J=1 ***
073276      !***      THE NUMBER OF COLUMNS IN THE 1st OPERAND ARRAY (B) ***
073276      !***      MUST BE THE SAME AS THE NUMBER OF ROWS IN THE 2nd ***
073276      !***      OPERAND ARRAY (A).  THE RESULT ARRAY (C) HAS THE SAME ***
073276      !***      NUMBER OF ROWS AS THE 1st OPERAND ARRAY AND THE SAME ***
073276      !***      NUMBER OF COLS AS THE 2st OPERAND ARRAY. ***
073276      !***      EITHER (BUT NOT BOTH) OF THE OPERAND ARRAYS CAN BE ***
073276      !***      VECTORS. ***
073276      !*** ***
073276      !***      THE MULTIPLICATION ROUTINE IS ALSO USED TO CALCULATE ***
073276      !***      THE RESIDUAL MATRIX -- R -- FOR THE SYS(A,B) STATEMENT. ***
073276      !***      Na ***
073276      !***      R(I,K) = B(I,K) - Sum A(I,J) * Q(J,K) ***
073276      !***      J=1 ***
073276      !*** ***
073276      !*** IN : FOR B*A & B*TRN(A) & TRN(B)*A: ***
073276      !*** ***
073276      !***      REL ADDR C ***
073276      !***      REL ADDR B ***
073276      !***      REL ADDR A ***
073276      !***      <--- R12 ***
073276      !*** ***
073276      !***      FOR R=B-AQ: ***
073276      !*** ***
073276      !***      TYPB (1 BYTE) ***
073276      !***      INCRB (2 BYTES) ***
073276      !***      ABS ADDR Bb ***
073276      !***      REL ADDR R ***
073276      !***      REL ADDR A ***
073276      !***      REL ADDR Q ***
073276      !***      <--- R12 ***
073276      !*** ***
073276      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED ***
073276      !***      (AND REDIMENSIONED IF NECESSARY). ***
073276      !*****
073276      !
073276      MATMUL  BSZ 0          !ENTRY FOR C = B * A
073276  170 222      CLB R70      !FLAG C = B * A
073300      MAT-   BSZ 0          !ENTRY FOR B-AQ,TRN(B)*A,B*TRN(A)
073300  316 326 326  JSB =LOCSZ      !Ba, Ma, Na
073303  170 310 001  CMB R70,=1      !LOOK AT OPERATION FLAG
073306  366 336      JNZ NOTRNA      !IF (B * TRN(A))
073310  316 326 326  JSB =SWAPEM      ! SWAP Ma & Na
073313      NOTRNA BSZ 0          !END IF
073313  174 022 241  LDM R74,R22      !SAVE Ma, Na
073316  316 326 326  JSB =LOCSZI      !Bb, Mb,Nb
073321  170 310 002  CMB R70,=2      !LOOK AT OPERATION FLAG
073324  366 336      JNZ NOTRNB      !IF (TRN(B) * A)
073326  316 326 326  JSB =SWAPEM      ! SWAP Mb & Nb
073331      NOTRNB BSZ 0          !END IF
073331  124 074 301  CMM R24,R74      !Nb (COLS B) = Ma (ROWS A) ?
073334  367 336      JZR MMEND1      !IF (Nb <> Ma)
073336  316 313 160  JSB =MISMAT      ! ERROR -- EXIT
073341      MMEND1 BSZ 0          !END IF
073341  120 024 241  LDM R20,R24      !SAVE NUM COL OF B (Nb)
073344  316 245 152  JSB =RDIM-      !REDIM C TO Mb X Na
073347  316 116 161  JSB =ZERTST      !TEST FOR NULL ARRAY
073352  366 336      JNZ MMCONT      !IF (NULL ARRAY)
073354  236          RTN          ! EXIT
073355      MMCONT BSZ 0          !END IF
073355  316 326 326  JSB =C=AORB      !SEE IF TEMP MEM NEEDED

```

```

073360 170 006 344          PUBD R70,+R6          !SAVE OPERATION FLAG
073363 122 345          PUMD R22,+R6          !SAVE Mc
073365 155 261 326        LDMD R55,=TMP2        !GET Bb
073370 326
073371 345          PUMD R55,+R6          !SAVE IT
073372 261 326 326        LDMD R55,=TYPB        !GET TYPE & INCR OF A
073375 345          PUMD R55,+R6          !SAVE THEM
073376 261 326 326        LDMD R55,=TYPB        !GET TYPE & INCR OF B
073401 345          PUMD R55,+R6          !SAVE THEM
073402          ! *****
073402          ! * MATRIX MULTIPLICATION LOOP *
073402          ! *****
073402 124 345          PUMD R24,+R6          !SAVE Na : NUM COLS A
073404 076 243          STM R24,R76          !MOVE Na FOR INCSUB
073406 316 326 326        JSB =ROMJSB
073411 326 326          DEF COLINT          !INIT COL PTR TO A
073413 261          BYT 261
073414 124 026 241        LDM R24,R26          !GET OLD Na OR Nb
073417 076 243          STM R24,R76          !MOVE IT FOR INCSUB
073421 170 310 001        CMB R70,=1
073424 366 336          JNZ NOCOLA          !IF (B * TRN(A))
073426 156 063 241        LDM R56,R63          ! GET ELE SIZE A
073431 263 326 326        STMD R56,=INCRA        ! INCRA FOR NEXT ROW ELE
073434 316 326 326        JSB =ACOLEL          ! GET COL INCR FOR A
073437 124 263 326        STMD R24,=TMP3        ! SAVE IN TMP3 FOR COLNXT
073442 326
073443          NOCOLA BSZ 0
073443 170 310 002        CMB R70,=2
073446 366 336          JNZ NOCOLB          !ELSE IF (TRN(B) * A)
073450 316 326 326        JSB =BCOLEL          ! GET COL INCR FOR B
073453 124 263 326        STMD R24,=INCRB        ! INCRB FOR NEXT COL ELE
073456 326
073457          NOCOLB BSZ 0          !END IF
073457 124 006 343        POMD R24,-R6        !RESTORE Na
073462          MMROW BSZ 0          !REPEAT
073462 124 026 243        STM R24,R26          ! COLS = Na
073465          MMCOL BSZ 0          ! REPEAT
073465 114 020 241        LDM R14,R20          ! ROWS = Nb
073470 170 220          TSB R70          ! TEST OPERATION FLAG
073472 364 336          JNG MMELS3          ! IF (B * A)
073474 316 315 161        JSB =DOTPRD          ! Sum B(I,J) * A(J,K)
073477 360 336          JMP MMEND3
073501          MMELS3 BSZ 0          ! ELSE - (R=B-AQ)
073501 152 012 343        POMD R52,-R12        ! GET TYPB, INCRB, ADDR B(I,K)
073504 046 242          STB R52,R46          ! SET TYPB
073506 155          DRP R55          ! SET FETCH ADDR FOR B(I,K)
073507 316 023 163        JSB =FETCH          ! FETCH VALUE B(I,K)
073512 175 053 241        LDM R75,R53          ! MOVE INCRB
073515 177 222          CLB R77          ! CLEAR FOR 3 BYTE SUBTRACT
073517 155 075 305        SBM R55,R75          ! NEXT ADDR B(I,K)
073522 152 012 345        PUMD R52,+R12        ! NEXT TYPB, INCRB, ADDR B(I,K)
073525 140 345          PUMD R40,+R12        ! PUSH VALUE OF B(I,K)
073527 316 326 326        JSB =ONER          ! CONVERT IT TO REAL
073532 316 341 161        JSB =DOT38          ! B(I,K) - Sum A(I,J) * Q(J,K)
073535 316 324 161        JSB =RONF5-          ! ROUND RESULT
073540          MMEND3 BSZ 0          ! END IF
073540 316 301 162        JSB =STOV          ! STORE C(I,K) (OR R(I,K)
073543 316 326 326        JSB =ROMJSB
073546 326 326          DEF COLNXT          ! NEXT COL OF A
073550 261          BYT 261
073551 165 261 326        LDMD R65,=TMP2        ! GET PTR ROW B
073554 326
073555 175 261 326        LDMD R75,=TMP2+        ! OLD PTR ROW B
073560 326
073561 263 326 326        STMD R75,=TMP2        ! DO ROW AGAIN
073564 126 213          DCM R26          ! COLS = COLS - 1
073566 366 275          JNZ MMCOL          ! UNTIL (COLS = 0)
073570 170 310 002        CMB R70,=2          ! LOOK AT OPERATION FLAG
073573 366 336          JNZ NCOLB          ! IF (TRN(B) * A)
073575 166 006 343        POMD R66,-R6        ! GET OLD INCRB
073600 345          PUMD R66,+R6          ! PUT IT BACK ON STACK

```

```

073601 055 243          STM R66,R55          !      MOVE INCRB
073603 157 222          CLB R57              !      CLEAR FOR 3 BYTE SUBTRACT
073605 175 305          SBM R75,R55          !      NEXT COLUMN OF B
073607 065 243          STM R75,R65          !      MOVE IT
073611                  NCOLB BSZ 0           !      END IF
073611 165 263 326      STMD R65,=TMP2       !      POINT NEXT ROW B
073614 326
073615 263 326 326      STMD R65,=TMP2+      !      SAVE IT
073620 261 326 326      LDMD R65,=TMP1+      !      GET Ba
073623 263 326 326      STMD R65,=TMP1       !      RESET PTR COL A
073626 263 326 326      STMD R65,=TMP3+      !      FOR COLNXT
073631 122 213          DCM R22              !      ROWS = ROWS - 1
073633 366 225          JNZ MMROW             !UNTIL (ROWS = 0)
073635                  ! *****
073635                  ! ** CLEAN UP STACK **
073635                  ! *****
073635 165 006 343      POMD R65,-R6          !GET TYPE & INCR B
073640 263 326 326      STMD R65,=TYPB       !RESTORE THEM
073643 343              POMD R65,-R6          !GET TYPE & INCR A
073644 263 326 326      STMD R65,=TYPB       !RESTORE THEM
073647 343              POMD R65,-R6          !GET Bb
073650 263 326 326      STMD R65,=TMP2+      !SAVE IT
073653 122 343          POMD R22,-R6          !GET Mc
073655 316 326 326      JSB =COPYAB          !SEE IF COPY NEEDED
073660 170 006 342      POBD R70,-R6          !GET OPERATION FLAG
073663 220              TSB R70              !TEST FLAG
073664 365 336          JPS MMEND4            !IF (FLAG < 0)
073666 152 012 343      POMD R52,-R12         ! GET RID OF TYPB, INCRB, ADDR B(I,K)
073671                  MMEND4 BSZ 0           !END IF
073671 236              RTN
073672                  !
073672                  ! *****
073672                  ! *** IN : R22 = Ma (OR Mb) ***
073672                  ! *** R24 = Na (OR Nb) ***
073672                  ! *** OUT : R22 = Na (OR Nb) ***
073672                  ! *** R24 = Ma (OR Mb) ***
073672                  ! *** R26 = OLD Na (OR Nb) ***
073672                  ! *****
073672                  !
073672                  ! SWAPEM BSZ 0
073672 122 026 243      STM R22,R26           !SAVE ROW
073675 024 241          LDM R22,R24           !SWAP COL
073677 126 243          STM R26,R24           !SWAP ROW
073701 022 241          LDM R26,R22           !SAVE OLD COL
073703 236              RTN
073704                  !
073704                  ! *****
073704                  ! *** C=AORB : IF THE RESULT ARRAY IS EQUAL TO AN OPERAND ARRAY ***
073704                  ! *** THE ROUTINE WILL RESERVE MEMORY FOR THE RESULT. ***
073704                  ! *** IN : TMP1 = Ba ***
073704                  ! *** TMP2 = Bb ***
073704                  ! *** R65 = Bc ***
073704                  ! *** OUT : R71 = 0 IF C <> A AND C<>B ***
073704                  ! *** R71 = 1 IF C =A ***
073704                  ! *** R71 = 377 IF C = B ***
073704                  ! *** R30 = ELEMENT SIZE OF A OR B IF C = A OR B ***
073704                  ! *****
073704                  !
073704                  ! LST
073704                  ! C=AORB BSZ 0
073704                  ! UNL
073704 171 222          CLB R71              !ASSUME ADDR'S DIFFERENT
073706 165 321 326      CMMD R65,=TMP1
073711 326
073712 366 336          JNZ C&B              !IF (C = A)
073714 130 261 326      LDMD R30,=INCRA      ! GET ELEMENT SIZE OF A
073717 326
073720 316 326 326      JSB =BASE=           ! RESERVE MEMORY TO COPY A
073723 171 210          ICB R71              ! FLAG C = A
073725 360 336          JMP C=END
073727                  C&B BSZ 0

```

```

073727 165 321 326          CMMD R65,=TMP2
073732 326
073733 366 336              JNZ C=END                !ELSE IF (C = B)
073735 130 261 326          LDMD R30,=INCRB          !   GET ELEMENT SIZE OF B
073740 326
073741 316 326 326          JSB =BASE=              !   RESERVE MEMORY TO COPY B
073744 171 212              DCB R71                  !   FLAG C = B
073746                      C=END                    BSZ 0          !END IF
073746 236                  RTN
073747                      !
073747                      !*****
073747                      !*** BASE= : RESERVES MEMORY FOR RESULT ARRAY IF THE RESULT ***
073747                      !***          ARRAY IS EQUAL TO AN OPERAND ARRAY.          ***
073747                      !*** IN   : R22 = Ma OR Mb          ***
073747                      !***          R24 = Na OR Nb          ***
073747                      !***          R30 = ELEMENT SIZE A OR B (INCRA OR INCRB) ***
073747                      !*** OUT  : TMP3++ = BASE ADDR OF RESULT ARRAY IN TEMP MEMORY ***
073747                      !***          TMP4+  = BASE ADDR OF RESULT ARRAY IN TEMP MEMORY ***
073747                      !***          R65    = BASE ADDR OF RESULT ARRAY IN TEMP MEMORY ***
073747                      !*****
073747                      !
073747                      BASE=      BSZ 0
073747 316 326 326          JSB =NUMBYT              !FIND NUM BYTES IN ARRAY
073752 316 326 326          JSB =RESMEM              !RESERVE MEMORY
073755 165 263 326          STMD R65,=TMP4+          !RESET Bc
073760 326
073761 263 326 326          STMD R65,=TMP3++          !SAVE IT FOR COPYAB
073764 371 336              JEZ BAS=EN              !IF (NOT ENOUGH ROOM)
073766 164 006 343          POMD R64,-R6             !   TRASH 2 RETURNS
073771                      BAS=EN      BSZ 0          !END IF
073771 236                  RTN
073772                      !
073772                      !*****
073772                      !*** NUMBYT : FINDS NUMBER OF BYTES IN THE ELEMENTS OF AN ARRAY ***
073772                      !***          (DOES NOT INCLUDE BYTES OF THE ARRAY HEADER) ***
073772                      !*** IN   : R22 = M : NUMBER OF ROWS IN ARRAY          ****
073772                      !***          R24 = N : NUMBER OF COLUMNS IN ARRAY      ****
073772                      !***          R30 = SIZE OF ELEMENTS IN ARRAY (8, 4 OR 3 BYTES) ****
073772                      !*** OUT  : R55 = NUMBER OF BYTES IN THE ELEMENTS OF THE ARRAY ****
073772                      !*****
073772                      !
073772                      NUMBYT    BSZ 0
073772 316 326 326          JSB =MNMUL              !FIND M * N
073775 166 055 241          LDM R66,R55              !MOVE M * N
074000 176 030 241          LDM R76,R30              !MOVE ELEMENT SIZE
074003 316 326 326          JSB =MNMUL3            !FIND NUM BYTES IN ARRAY ELEMENTS
074006 236                  RTN
074007                      !
074007                      !*****
074007                      !*** COPYAB : COPIES SCRATCH RESULT ARRAY INTO AN OPERAND ARRAY ***
074007                      !*** IN   : R71  = 377 IF C = B (RESULT = 1st OPERAND) ***
074007                      !***          R71  = 1   IF C = A (RESULT = 2nd OPERAND) ***
074007                      !***          R71  = 0   IF C <> A AND C <> B ***
074007                      !***          TMP1  = Ba : BASE ADDR OF A ***
074007                      !***          TMP2  = Bb : BASE ADDR OF B ***
074007                      !***          TMP3++= Bc : BASE ADDR OF C ***
074007                      !***          TYPA  = TYPE AND ELEMENT SIZE OF A ***
074007                      !***          TYPB  = TYPE AND ELEMENT SIZE OF B ***
074007                      !*** OUT  : RESULT ARRAY COPIED INTO OPERAND ARRAY ***
074007                      !*****
074007                      !
074007                      LST
074007                      COPYAB    BSZ 0
074007                      !
074007 171 220              TSB R71                  !TEST COPY FLAG
074011 367 336              JZR CABED5              !IF (C = B OR A)
074013 365 336              JPS CABEL3              !   IF (C = B)
074015 155 260 326          LDBD R55,=TYPB          !   GET TYPE OF ARRAY B
074020 326
074021 262 326 326          STBD R55,=TYPB          !   PUT IT IN TYPB FOR MOVE
074024 261 326 326          LDMD R55,=TMP2+          !   GET Bb

```



```

074027 360 336          JMP CABED6
074031          CABEL3  BSZ 0          ! ELSE - (C = A)
074031 155 261 326      LDMD R55,=TMP1+ ! GET Ba
074034 326
074035          CABED6  BSZ 0          ! END IF
074035 012 345          PUMD R#,+R12    ! PUSH Ba OR Bb
074037 316 326 326      JSB =RCINC      ! ROW & COL INCR ON STACK FOR A OR B
074042 165 261 326      LDMD R65,=TMP3++ ! GET Bc
074045 326
074046 012 345          PUMD R65,+R12    ! PUSH Bc
074050 154 345          PUMD R54,+R12    ! PUSH ROW & COL INCR FOR C
074052 316 326 326      JSB =TRCRST    ! RESTORE TRACE FLAG
074055 316 077 166      JSB =MOVE       ! COPY SCATCH ARRAY
074060          CABED5  BSZ 0          !END IF
074060 236          RTN
074061          !
074061          !*****
074061          !*** RCINC : PUSHES ROW AND COL INCREMENT OF RESULT ARRAY ON R12 ***
074061          !*** IN : INCR = ELEMENT SIZE OF C (8, 4 OR 3 BYTES) ***
074061          !*** R24 = N : NUMBER OF COLUMNS IN A OR B ***
074061          !*** OUT : ROW INCREMENT FOR A OR B ***
074061          !*** COL INCREMENT FOR A OR B ***
074061          !*** <--- R12 ***
074061          !*****
074061          !
074061          RCINC      BSZ 0
074061 176 261 326      LDMD R76,=INCR      !GET ROW INCREMENT
074064 326
074065 166 024 241      LDM R66,R24          !MOVE NUM COLS
074070 316 326 326      JSB =MNMUL3        !FIND COL INCREMENT
074073 316 326 326      JSB =SHIF54        !MOVE COL INCREMENT
074076 176 054 243      STM R76,R54        !MOVE ROW INCREMENT
074101 154 012 345      PUMD R54,+R12      !PUSH ROW & COL INCR FOR A OR B
074104 236          RTN
074105          !
074105          !*****
074105          !*** BCOLEL : FINDS COLUMN INCREMENT OF OPERAND ARRAY B, I.E. THE ****
074105          !*** INCREMENT NEEDED TO GET FROM B(I,J) TO B(I+1,J). ****
074105          !*****
074105          !
074105          BCOLEL   BSZ 0
074105 146 260 326      LDBD R46,=INCRB      !GET ELE SIZE OF B
074110 326
074111 360 336          JMP INCSUB          !FIND NEXT COL ELE INCR
074113          !
074113          !*****
074113          !*** ACOLEL : FINDS COLUMN INCREMENT OF OPERAND ARRAY A, I.E. THE ****
074113          !*** INCREMENT NEEDED TO GET FROM A(I,J) TO A(I+1,J). ****
074113          !*****
074113          !
074113          LST
074113          ACOLEL   BSZ 0
074113          !      UNL
074113          LST
074113 146 260 326      LDBD R46,=INCRA      !GET ELE SIZE A
074116 326
074117          !
074117          !*****
074117          !*** INCSUB : FINDS COLUMN INCREMENT OF AN ARRAY ****
074117          !*** IN : R46 = ELEMENT SIZE OF THE ARRAY ****
074117          !*** R24 = NUMBER OF COLUMNS IN THE ARRAY ****
074117          !*** R76 = NUMBER OF COLUMNS IN THE ARRAY ****
074117          !*** OUT : R76 = NUMBER OF COLUMNS IN THE ARRAY ****
074117          !*** R24 = (8, 4 OR 3) * NUM COLS = THE COLUMN INCREMENT ****
074117          !*****
074117          !
074117          INCSUB   BSZ 0
074117 124 205          LLM R24          !R24 X 2
074121 205          LLM R24          !R24 X 4
074122 147 250 004      LDB R47,=4      !ASSUME SHORT
074125 046 300          CMB R47,R46      !COMPARE WITH ELE SIZE

```

074127	367	336		JZR	INCEND	!IF (NOT SHORT)
074131	124			DRP	R24	! SET DRP
074132	373	336		JCY	INC3	! IF (NOT INTEGER)
074134	205			LLM	R24	! R24 X 8 -- REAL
074135	236			RTN		! END IF
074136	076	305	INC3	SBM	R#,R76	! R24 X 3 -- INTEGER
074140			INCEND	BSZ	0	!END IF
074140	236			RTN		
074141			!	UNL		

```

074141      !          HED MAT C = TRANSPOSE(A) ROUTINE
074141      !****   TRN (MAT) ATTRIBUTES TABLE   ****
074141 024 055      BYT 24,55
074143      !*****
074143          LST
074143      TRN10    BSZ 0
074143 316 326 326    JSB =ROMJSB
074146 326 326      DEF TRNPS2
074150 261          BYT 261
074151 236          RTN
074152      !          UNL

```

```

074152      !          HED LBND AND UBND ROUTINES
074152      !****  LBND ATTRIBUTES !TABLE  *****
074152 044 055      BYT 44,55
074154      !*****
074154      LDIM10    BSZ 0
074154 316 326 326      JSB =ROMJSB
074157 326 326      DEF LDIM2
074161 261      BYT 261
074162 236      RTN
074163      !
074163      !****  UBND ATTRIBUTES !TABLE  *****
074163 044 055      BYT 44,55      !ONE ARRAY, ONE NUMERIC
074165      !*****
074165      UDIM10    BSZ 0
074165 316 326 326      JSB =ROMJSB
074170 326 326      DEF UDIM2
074172 261      BYT 261
074173 236      RTN

```

```

074174      !          HED MAT A = ZERO/CONSTANT/(SCL) ROUTINES
074174      !**** CON(M) ATTRIBUTES TABLE *****
074174 020 055      BYT 20,55
074176      !*****
074176      !*** MAT C = CON(M) *****
074176      !***      REDIMENSIONS RESULT ARRAY C INTO A VECTOR THEN *****
074176      !***      ASSIGNS VALUE 1 TO ALL ELEMENTS OF C. *****
074176      !*** IN   : REL ADDR C *****
074176      !***      SCALOR M *****
074176      !***      <--- R12 *****
074176      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED & ASSIGNED *****
074176      !*****
074176      !
074176 316 243 162 CDIM1V JSB =DUP1V      !DUP REL ADDR ARRAY ON R12
074201 316 326 326 JSB =RDIM1      !NOW REDIM TARGET ARRAY.
074204 360 336      JMP CON1      !NOW GO SET IT = CON.
074206      !
074206      !**** CON(M,N) ATTRIBUTES TABLE *****
074206 040 055      BYT 40,55
074210      !*****
074210      !*** MAT C = CON(M,N) *****
074210      !***      REDIMENSIONS RESULT ARRAY C TO M X N THEN *****
074210      !***      ASSIGNS VALUE 1 TO ALL ELEMENTS OF C. *****
074210      !*** IN   : REL ADDR C *****
074210      !***      SCALOR M *****
074210      !***      SCALOR N *****
074210      !***      <--- R12 *****
074210      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED & ASSIGNED *****
074210      !*****
074210      !
074210 316 256 162 CDIM2V JSB =DUP2V      !DUP REL ADDR ARRAY ON R12
074213 316 326 326 JSB =RDIM2      !REDIM TARGET ARRAY.
074216 360 336      JMP CON1      !GO SET IT = CON.
074220      !
074220      !**** CON (MAT) ATTRIBUTES TABLE *****
074220 000 055      BYT 0,55
074222      !*****
074222      !*** MAT C = CON *****
074222      !***      ASSIGNS VALUE 1 TO ALL ELEMENTS OF C. *****
074222      !*** IN   : REL ADDR C *****
074222      !***      <--- R12 *****
074222      !*** OUT : STACK POPPED AND ARRAY ASSIGNED *****
074222      !*****
074222      !
074222 CON1      BSZ 0
074222 144 251 377 LDM R44,=377,1,0,0      !TAGGED INTEGER 1
074225 001 000 000
074230 360 336      JMP ZCON10      !GO STORE CONST 1'S IN MEMORY.
074232      !
074232      !**** (NUM EXP) ATTRIBUTES TABLE *****
074232 000 051      BYT 0,51
074234      !*****
074234      !*** MAT C = (SCALOR) *****
074234      !***      ASSIGNS VALUE OF SCALOR TO ALL ELEMENTS OF C. *****
074234      !*** IN   : REL ADDR C *****
074234      !***      SCALOR *****
074234      !***      <--- R12 *****
074234      !*** OUT : STACK POPPED AND ARRAY ASSIGNED *****
074234      !*****
074234      !
074234 SCL.      BSZ 0
074234 140 012 343 POMD R40,-R12      !GET SCALAR.
074237 360 336      JMP ZCON10      !GO STORE VAL IN WHOLE ARRAY.
074241      !
074241      !**** ZER(M) ATTRIBUTES TABLE *****
074241 020 055      BYT 20,55
074243      !*****
074243      !*** MAT C = ZER(M) *****
074243      !***      REDIMENSIONS RESULT ARRAY C INTO A VECTOR THEN *****
074243      !***      ASSIGNS VALUE 0 TO ALL ELEMENTS OF C. *****
074243      !*** IN   : REL ADDR C *****

```

```

074243      !***          SCALOR M                      ****
074243      !***          <--- R12                      ****
074243      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED & ASSIGNED ****
074243      !*****
074243      !
074243 316 243 162 ZDIM1V   JSB =DUP1V                   !DUP REL ADDR ARRAY ON R12
074246 316 326 326      JSB =RDIM1                     !REDIM TARGET ARRAY.
074251 360 336      JMP ZER.                          !GO SET IT = ZERO.
074253      !
074253      !**** ZER(M,N) ATTRIBUTES TABLE *****
074253 040 055      BYT 40,55
074255      !*****
074255      !*** MAT C = ZER(M,N)                      ****
074255      !*** REDIMENSIONS RESULT ARRAY C TO M X N THEN ****
074255      !*** ASSIGNS VALUE 0 TO ALL ELEMENTS OF C. ****
074255      !*** IN : REL ADDR C                      ****
074255      !*** SCALOR M                      ****
074255      !*** SCALOR N                      ****
074255      !***          <--- R12                      ****
074255      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED & ASSIGNED ****
074255      !*****
074255      !
074255 316 256 162 ZDIM2V   JSB =DUP2V                   !DUP REL ADDR ARRAY ON R12
074260 316 326 326      JSB =RDIM2                     !REDIM TARGET ARRAY.
074263 360 336      JMP ZER.                          !GO SET IT = ZERO.
074265      !
074265      !**** ZER (MAT) ATTRIBUTES TABLE *****
074265 000 055      BYT 0,55
074267      !*****
074267      !*** MAT C = ZER                      ****
074267      !*** ASSIGNS VALUE 0 TO ALL ELEMENTS OF C. ****
074267      !*** IN : REL ADDR C                      ****
074267      !***          <--- R12                      ****
074267      !*** OUT : STACK POPPED AND ARRAY ASSIGNED ****
074267      !*****
074267      !
074267      ZER.      BSZ 0
074267 144 223      CLM R44
074271 250 377      LDB R44,=377                      !TAGGED INTEGER 0
074273      !
074273      !*****
074273      !*** COMMON CODE FOR CON AND ZER ROUTINES ****
074273      !*****
074273      !
074273      ZCON10     BSZ 0
074273 117 311 300      CMM R17,=300                      !LOOK AT ERROR FLAG
074276 372 336      JNC ZCONT                          !IF (REDIMENSION ERROR)
074300 165 012 343      POMD R65,-R12                    ! TRASH REL ADDR OF ARRAY
074303 236      RTN                                      ! EXIT
074304      ZCONT     BSZ 0                              !END IF
074304 140 006 345      PUMD R40,+R6                      !SAVE VALUE TO BE STORED
074307 316 326 326      JSB =LOCSZ-                      !GET M,N,BASE,INCR,C,TYP,C,BIN.
074312 140 006 343      POMD R40,-R6                      !GET VALUE TO BE STORED
074315 316 116 161      JSB =ZERTST                      !NULL ARRAY?
074320 367 336      JZR ZCEND                          !IF (NOT NULL ARRAY)
074322 316 301 162      JSB =STOV                        ! STORE VALUE IN C(1,1)
074325 165 006 345      PUMD R65,+R6                      ! SAVE Bc
074330 263 326 326      STMD R65,=TMP4+                  ! RESET STORE ADDRESS
074333 012 345      PUMD R65,+R12                        ! STORE ADDR FOR MOVE
074335 316 061 170      JSB =RCINC                      ! ROW & COL INCR ON STACK
074340 223      CLM R#                                    ! 0 OUT ROW & COL INCR
074341 165 006 343      POMD R65,-R6                      ! RESTORE FETCH ADDR
074344 012 345      PUMD R65,+R12                        ! FETCH ADDRESS FOR MOVE
074346 154 345      PUMD R54,+R12                        ! FETCH ROW & COL INCR ON STACK
074350 260 326 326      LDBD R#,,=TYP C                  ! GET TYPE OF C
074353 262 326 326      STBD R#,,=TYP A                  ! STORE IN TYP A FOR MOVE
074356 316 077 166      JSB =MOVE                        ! COPY VALUE TO ALL ELE OF ARRAY
074361 316 312 154      JSB =CKTRC                      ! CHECK FOR TRACE
074364      ZCEND     BSZ 0                              !END IF
074364 236      RTN
074365      !

```

```

074365      !*****
074365      !*** MNMUL : R55 = R22 * R24      ***
074365      !*****
074365      !
074365      LST
074365      MNMUL      BSZ 0
074365 124 076 243      STM R24,R76      !POSITION N FOR MULTIPLY.
074370      !
074370      !*****
074370      !*** MNMUL2: R55 = R22 * R76      ***
074370      !*****
074370      !
074370      MNMUL2      BSZ 0
074370 122 066 243      STM R22,R66      !POSITION M FOR MULTIPLY.
074373      !
074373      !*****
074373      !*** MNMUL3: R55 = R66 * R76      ***
074373      !*****
074373      !
074373      MNMUL3      BSZ 0
074373 154 223      CLM R54      !INITIALIZE PRODUCT AREA.
074375 316 326 326      JSB =INTMUL      !FORM M X N
074400 231      SHIF54      BCD
074401 154 205      LLM R54      !SHIFT PRODUCT ...
074403 205      LLM R54      !TWO DIGITS TO THE RIGHT
074404 230      BIN
074405 236      RTN
074406      !
074406      !*****
074406      !*** LOCSZ- : FINDS Bc, Mc, Nc, SETS BINARY MODE      ****
074406      !*** IN : REL ADDR C      ****
074406      !***      <--- R12      ****
074406      !*** OUT : REL ADDR C POPPED OFF STACK      ****
074406      !***      R22 = Mc : NUM ROWS OF C      ****
074406      !***      R24 = Nc : NUM COLS OF C      ****
074406      !***      R65 = Bc : BASE ADDRESS OF C      ****
074406      !***      R36 = OPTION BASE      ****
074406      !***      IF C A VECTOR  LSB TRACE FLAG = 1      ****
074406      !***      IF C A MATRIX  LSB TRACE FLAG = 0      ****
074406      !*****
074406      !
074406      LOCSZ-      BSZ 0
074406 235      CLE      !CLEAR FLAG.
074407 233      DCE      !FLAG SETTING FOR RESULT ARRAY
074410 316 326 326      JSB =LOCSIZ      !GET Mc,Nc,Bc, SET BINARY.
074413 371 336      VECFLG      JEZ LOCRTN      !IF (C IS A VECTOR)
074415 147 260 326      LDBD R47,=TRCFLG      ! GET TRACE FLAG
074420 326
074421 210      ICB R47      ! TAG AS VECTOR
074422 262 326 326      STBD R47,=TRCFLG      ! RESTORE FLAG
074425      LOCRTN      BSZ 0      !END IF
074425 236      RTN
074426      !
074426      !*****
074426      !*** LOCSZI : FINDS Bb, Mb, Nb, SETS BINARY MODE      ****
074426      !*** IN : REL ADDR B      ****
074426      !***      <--- R12      ****
074426      !*** OUT : REL ADDR B POPPED OFF STACK      ****
074426      !***      R22 = Mb : NUM ROWS OF B      ****
074426      !***      R24 = Nb : NUM COLS OF B      ****
074426      !***      R65 = Bb : BASE ADDRESS OF B      ****
074426      !***      R36 = OPTION BASE      ****
074426      !*****
074426      !
074426 235      LOCSZI      CLE      !CLEAR FLAG.
074427 234      ICE      !SET FLAG.
074430 360 336      JMP LOCSIZ      !CONTINUE - TYPE B.
074432      !
074432      !*****
074432      !*** LOCSZ : FINDS Ba, Ma, Na, SETS BINARY MODE      ****
074432      !*** IN : REL ADDR A      ****

```

```

074432      !***          <--- R12          ****
074432      !*** OUT : REL ADDR A POPPED OFF STACK      ****
074432      !***          R22 = Ma : NUM ROWS OF A      ****
074432      !***          R24 = Na : NUM COLS OF A      ****
074432      !***          R65 = Ba : BASE ADDRESS OF A    ****
074432      !***          R36 = OPTION BASE              ****
074432      !*****
074432      !
074432 235      LOCSZ      CLE                      !TYPE A.
074433 165 012 343 LOCSIZ      POMD R65,-R12          !GET RELATIVE ADDRESS
074436 316 326 326 LOCSZ+     JSB =RUDIM             !R22,=R24=0 THEN REDIM.
074441 230                      BIN                  !SET MODE.
074442 236                      RTN
074443      !          UNL

```



```

074443      !          HED MAXAB(A), AMIN(A)/AMAX(A) ROUTINES
074443      !*****  MAXAB ATTRIBUTES TABLE  *****
074443 024 055      BYT 24,55
074445      !*****
074445      !*** MAXAB (A) ***
074445      !***  FINDS LARGEST ABSOLUTE VALUE OF ANY ELEMENT IN ARRAY ***
074445      !*** IN  : REL ADDR A ***
074445      !***          <--- R12 ***
074445      !*** OUT : LARGEST ABSOLUTE VALUE ***
074445      !***          <--- R12 ***
074445      !*****
074445      !
074445 126 251 000 MAXAB.  LDM R26,=0,90C      !FLAG FOR THIS ROUTINE.
074450 220
074451 360 336      JMP MNMX10      !GO TO COMMON ROUTINE
074453      !
074453      !*****  MIN ATTRIBUTES TABLE  *****
074453 024 055      BYT 24,55
074455      !*****
074455      !*** AMIN (A) ***
074455      !***  FINDS VALUE OF SMALLEST ELEMENT IN THE ARRAY. ***
074455      !*** IN  : REL ADDR A ***
074455      !***          <--- R12 ***
074455      !*** OUT : SMALLEST VALUE OF ARRAY ***
074455      !***          <--- R12 ***
074455      !*****
074455      !
074455 126 251 011 AMIN.  LDM R26,=9C,0      !FLAG SETTING FOR AMIN(A) .
074460 000
074461 360 336      JMP MNMX10      !GO TO COMMON ROUTINE.
074463      !
074463      !*****  MAX ATTRIBUTES TABLE  *****
074463 024 055      BYT 24,55
074465      !*****
074465      !*** AMAX (A) ***
074465      !***  FINDS VALUE OF LARGEST ELEMENT IN THE ARRAY. ***
074465      !*** IN  : REL ADDR A ***
074465      !***          <--- R12 ***
074465      !*** OUT : LARGEST VALUE OF ARRAY ***
074465      !***          <--- R12 ***
074465      !*****
074465      !
074465 126 223      AMAX.  CLM R26      !FLAG SETTING FOR AMAX(A) .
074467      !
074467      !*****
074467      !*** COMMON CODE FOR MAXAB, AMIN, AMAX ***
074467      !*****
074467      !
074467 316 032 171 MNMX10  JSB =LOCSZ      !FIND Ba,Ma,Na, (OPTION BASE IN R36)
074472 136 215      TCM R36      !WANT IT 1 IF 0, 0 IF 1.
074474 211      ICM R36
074475 263 326 326      STMD R36,=TMP3      !SAVE OPTION BASE.
074500 316 107 161      JSB =ZTST-      !TEST FOR NULL ARRAY.
074503 366 336      JNZ NOTNUL      !IF (NULL ARRAY)
074505 122 223      CLM R22      ! FOR ROWMIN,ROWMAX,ROWMAB
074507 124 223      CLM R24      ! FOR COLMIN,COLMAX,COLMAB
074511 316 326 326      JSB =WINDUP      ! ZERO ANS FOR ROW, COL #
074514 231      BCD
074515 235      CLE
074516 126 221      TSM R26      ! TEST FOR FUNCTION
074520 375 336      JLN ZERANS      ! IF (NOT MAXAB)
074522 366 336      JNZ POSINF      ! IF (NOT AMIN)
074524 233      DCE      ! SIGN FOR NEG INF
074525      POSINF BSZ 0      ! END IF
074525 316 326 326      JSB =FTR99      ! ANS= + OR - INF
074530      ZERANS BSZ 0      ! END IF
074530 104 251 146      GTO ZERDET      ! ANS= 0 & EXIT
074533 147
074534      NOTNUL BSZ 0      !ELSE - (NOT NULL ARRAY)
074534 316 326 326      JSB =GETELT      ! PUT Ba ON STACK
074537 110 022 241      LDM R10,R22      ! ROWS = Ma

```

```

074542 114 024 241          LDM R14,R24          ! COLS = Na
074545 316 326 326          JSB =INIVAL          ! SAVE POSITON OF Ba
074550                      MNMROW BSZ 0          ! REPEAT
074550 114 024 241          LDM R14,R24          ! COLS = Na
074553                      MNMCOL BSZ 0          ! REPEAT
074553 160 012 343          POMD R60,-R12         ! GET BEST ANS SO FAR
074556 345                  PUMD R60,+R12         ! ANSWER COPY
074557 345                  PUMD R60,+R12         ! WORKING COPY
074560 316 326 326          JSB =GETELT          ! GET A(I,J)
074563 316 326 326          JSB =COMRC           ! FIND NEW MAX-MIN
074566 316 361 160          JSB =NXTA           ! POINT NEXT A(I,J)
074571 114 213              DCM R14              ! COLS = COLS -1
074573 366 356              JNZ MNMCOL           ! UNTIL (COLS = 0)
074575 110 213              DCM R10              ! ROWS = ROWS - 1
074577 366 347              JNZ MNMROW           ! UNTIL (ROWS = 0)
074601 122 000 305          SBM R22,R0           ! ADJUST Ma SINCE DOWNCOUNT
074604 211                  ICM R22              ! OFFSET BY 1
074605 325 326 326          SBMD R22,=TMP3       ! ADJUST FOR OPTION BASE
074610 114 271 326          LDMI R14,=MBASE      ! GET COL VALUE -- Na
074613 326
074614 124 014 305          SBM R24,R14          ! ADJUST SINCE DOWNCOUNT
074617 211                  ICM R24              ! OFFSET BY 1
074620 325 326 326          SBMD R24,=TMP3       ! ADJUST FOR OPTION BASE
074623 100 261 326 WINDUP LDMD R0,=MBASE        ! TEMP AREA BASE PTR
074626 326
074627 130 260 326          LDBD R30,=TMP4       ! GET MATRIX-VECTOR FLAG
074632 326
074633 376 336              JRZ YESCOL           ! IF (VECTOR)
074635 125 250 377          LDB R25,=377         ! NULL FOR COL VALUE
074640                      BSZ 0                 ! END IF
074640 000                  ARP R0               ! COMMON ARP TO SAVE CODE
074641 126 221              TSM R26              ! TEST FOR FUNCTION
074643 122                  DRP R22              ! COMMON DRP TO SAVE CODE
074644 367 336              JZR SAVMAX
074646 374 336              JLZ SAVMIN            ! IF (MAXAB)
074650 267 065 000          STMD R#,X#,ROWMAB    ! RMAXAB ANSWER
074653 124 267 067          STMD R24,X#,COLMAB   ! CMAXAB ANSWER
074656 000
074657 236                  RTN                  ! ELSE IF (AMIN)
074660 267 055 000 SAVMIN STMD R#,X#,ROWMIN      ! RAMIN ANSWER
074663 124 267 057          STMD R24,X#,COLMIN   ! CAMIN ANSWER
074666 000
074667 236                  RTN                  ! ELSE - (AMAX)
074670 267 061 000 SAVMAX STMD R#,X#,ROWMAX      ! RAMAX ANSWER
074673 124 267 063          STMD R24,X#,COLMAX   ! CAMAX ANSWER
074676 000
074677                      BSZ 0                 ! END IF
074677                      BSZ 0                 !END IF
074677 236                  RTN
074700                      !
074700                      ! *****
074700                      ! *** COMRC : FINDS NEW MAXIMUM OR MINIMUM VALUE. ***
074700                      ! *** IN : BEST ANSWER ***
074700                      ! *** BEST ANSWER ***
074700                      ! *** VALUE A(I,J) ***
074700                      ! *** <--- R12 ***
074700                      ! *** R10 = I : ROW COUNTER ***
074700                      ! *** R14 = J : COL COUNTER ***
074700                      ! *** R26 = FUNCTION FLAG (1 BYTE) ***
074700                      ! *** 0 MAXAB, AMAX, MAXRC (FOR NORM) ***
074700                      ! *** 9C AMIN ***
074700                      ! *** OUT : NEW BEST ANSWER ***
074700                      ! *** <--- R12 ***
074700                      ! *****
074700                      !
074700                      LST
074700 070 243 COMRC STM R#,R70 !EXTRA COPY IN CASE = ANS.
074702 316 326 326 COMRC- JSB =SUBROI !DIFF OF LAST 2 VALUES ON STACK.
074705                      ! UNL
074705                      LST
074705 316 326 326 JSB =ONER !DEMAND REAL ANS.

```

```

074710 230          BIN          !RESET MODE.
074711 160 221      TSM R60      !BEST ANS SO FAR SAME AS A(I,J) .
074713 367 336      JZR COMEN1   !IF (BEST ANS <> A(I,J))
074715 161 026 226   XRB R61,R26 ! CHS IF AMIN-NO CHANGE IF AMAX.
074720 376 336      JRZ COMEN2   ! IF (STACK HAS OLD ANS)
074722 160 012 343   POMD R60,-R12 ! RID STACK OF VALUE
074725 170 345      PUMD R70,+R12 ! REPLACE WITH NEW MAX (MIN) .
074727 110 000 243 INIVAL STM R10,R0 ! NEW ROW I OF CURRENT MAX (MIN) .
074732 263 326 326   STMD R10,=TMP2+ ! FOR 2nd MATRIX ROM
074735 114 273 326   STMI R14,=MBASE ! NEW COL J OF CURR MAX (MIN) .
074740 326
074741          COMEN2 BSZ 0      ! END IF
074741          COMEN1 BSZ 0      !END IF
074741 236          COMRTN RTN
074742          ! UNL
074742          !
074742          !*****
074742          !*** GETELT : GETS A(I,J) VALUE AND PUSHES IT ON R12 STACK. ***
074742          !*** WILL TAKE ABSOLUTE VALUE OF A(I,J) IF NECESSARY (MAXAB) ***
074742          !*** IN : . ***
074742          !*** . ***
074742          !*** . ***
074742          !*** <--- R12 ***
074742          !*** R27 = FUNCTION FLAG ***
074742          !*** 0 : AMIN, AMAX ***
074742          !*** 90C : MAXAB ***
074742          !*** OUT : . ***
074742          !*** . ***
074742          !*** . ***
074742          !*** [ABS] VALUE A(I,J) ***
074742          !*** <--- R12 ***
074742          !*****
074742          !
074742          LST
074742 127 070 242 GETELT STB R27,R70 !PUT FLAG IN COMMON LOC.
074745          GELT BSZ 0
074745          ! UNL
074745 165 223      CLM R65      !INDICATE FETCH FROM TMP1
074747 316 007 163   JSB =AFETCH !GET A(I,J)
074752 231          BCD          !MODE FOR SIGN SHIFT
074753 170 220      TSB R70      !TEST FUNCTION FLAG
074755 367 336      JZR GEEND1   !IF (ABSOLUTE VALUE)
074757 141 206      LRB R41      ! SHIFT OF SIGN FOR REAL
074761 204          LLB R41      ! ABS(A(I,J)) IF REAL
074762 144 310 377   CMB R44,=377 ! A(I,J) REAL?
074765 372 336      JNC GEEND2   ! IF (INTEGER)
074767 145 221      TSM R45      ! TEST FOR NEG INT
074771 365 336      JPS GEEND3   ! IF (NEG INT)
074773 215          TCM R45      ! ABS(A(I,J))
074774          GEEND3 BSZ 0      ! END IF
074774          GEEND2 BSZ 0      ! END IF
074774          GEEND1 BSZ 0      !END IF
074774 140 012 345   PUMD R40,+R12 !VALUE IN 40 AND STACK
074777 236          RTN

```

```

075000      !          HED MAT PRINT ROUTINE
075000      !*****  MAT PRINT A; ATTRIBUTES TABLE  *****
075000 036      BYT 36
075001      !*****
075001      !***      /PRINT\          /STATEMENT NO.\          ***
075001      !*** MAT <          > [USING <          >;] A;          ***
075001      !***      \DISP /          \FORMAT STRING/          ***
075001      !***
075001      !*** IN   : REL ADDR A          ***
075001      !***          <--- R12          ***
075001      !***          USING? <> 0 THEN HAVE USING CLAUSE          ***
075001      !***          = 0 THEN DO NOT HAVE USING CLAUSE          ***
075001      !***
075001      !*** OUT : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED ***
075001      !***          WITH CLOSE SEPERATION          ***
075001      !*****
075001      !
075001      SEM.      BSZ 0
075001 230          BIN
075002 120 223      CLM R20          !CLEAR FLAGS.
075004 320 326 326  CMBD R20,=USING?  !SEE IF PRINT USING
075007 366 336      JNZ COM.          !JIF PRINT USING
075011 210          ICB R20          !FOR TSTUS
075012 360 336      JMP COM.++          !GO OUTPUT ELEMENTS.
075014      !
075014      !*****  MAT PRINT A/ ATTRIBUTES TABLE  *****
075014 057      BYT 57
075015      !*****
075015      !***      /PRINT\          /STATEMENT NO.\          ***
075015      !*** MAT <          > [USING <          >;] A/          ***
075015      !***      \DISP /          \FORMAT STRING/          ***
075015      !***
075015      !*** IN   : REL ADDR A          ***
075015      !***          <--- R12          ***
075015      !***          USING? <> 0 THEN HAVE USING CLAUSE          ***
075015      !***          = 0 THEN DO NOT HAVE USING CLAUSE          ***
075015      !***
075015      !*** OUT : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED ***
075015      !***          ONE ELEMENT PER LINE          ***
075015      !*****
075015      !
075015      SLSH.      BSZ 0
075015 230          BIN
075016 121 222      CLB R21          !READY TO INCREMENT.
075020 320 326 326  CMBD R21,=USING?  !SEE IF PRINT USING
075023 366 336      JNZ COM.          !JIF PRINT USING
075025 212          DCB R21          !TO FORCE THRU TSTUS
075026 360 336      JMP COM.+          !GO OUTPUT ELEMENTS.
075030      !
075030      !*****  MAT PRINT A ATTRIBUTES TABLE  *****
075030 057      BYT 57
075031      !*****
075031      !***      /PRINT\          /STATEMENT NO.\          ***
075031      !*** MAT <          > [USING <          >;] A          ***
075031      !***      \DISP /          \FORMAT STRING/          ***
075031      !***
075031      !*** IN   : REL ADDR A          ***
075031      !***          <--- R12          ***
075031      !***          USING? <> 0 THEN HAVE USING CLAUSE          ***
075031      !***          = 0 THEN DO NOT HAVE USING CLAUSE          ***
075031      !***
075031      !*** OUT : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED ***
075031      !***          WITH WIDE SEPERATION          ***
075031      !*****
075031      !
075031 360 336      COMM.      JMP COM.          !NEED DIFF ATTRIB'S FOR DECOMPILING.
075033      !
075033      !*****  MAT PRINT A, ATTRIBUTES TABLE  *****
075033 036      BYT 36
075034      !*****
075034      !***      /PRINT\          /STATEMENT NO.\          ***

```

```

075034      !*** MAT <          > [USING <          >;] A1,      ***
075034      !***      \DISP /          \FORMAT STRING/      ***
075034      !***      ***
075034      !*** IN   : REL ADDR A      ***
075034      !***      <--- R12      ***
075034      !***      USING? <> 0 THEN HAVE USING CLAUSE      ***
075034      !***      = 0 THEN DO NOT HAVE USING CLAUSE      ***
075034      !***      ***
075034      !*** OUT  : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED ***
075034      !***      WITH WIDE SEPERATION      ***
075034      !*****
075034      !
075034      COM.      BSZ 0
075034      230      BIN
075035 121 222      CLB R21      !FLAG SETTING TO BYPASS TSTUS
075037 120 260 326 COM.+  LDBD R20,=USING?      !FOR TSTUS
075042 326
075043 263 326 326 COM.++ STMD R#,=TMP2      !SAVE FOR TSTUS
075046      LST
075046 316 326 326      JSB =ROMJSB
075051 326 326      DEF FIND#/      !FIND # AND TRAILING / IN FORMAT STRING
075053 261      BYT 261
075054      !      UNL
075054      !
075054      !      *****
075054      !      *** GET OPERAND INFORMATION      ***
075054      !      *** AND TEST FOR NULL ARRAY      ***
075054      !      *****
075054      !
075054 316 032 171      JSB =LOCSZ      !GET Ma,Na,Ba, SET BINARY.
075057 316 116 161      JSB =ZERTST      !CHECK FOR NULL ARRAY.
075062 366 336      JNZ PRTLOP      !IF (NULL ARRAY)
075064 236      RTN      !      EXIT
075065      PRTLOP  BSZ 0      !END IF
075065      !
075065      !      *****
075065      !      *** LOOP TO FILL PRINT BUFFER ***
075065      !      *** R20 = I : ROW COUNTER      ***
075065      !      *** R22 = Ma: NUM ROWS A      ***
075065      !      *** R24 = Na: NUM COLS A      ***
075065      !      *** R26 = J : COL COUNTER      ***
075065      !      *****
075065      !
075065 120 223      CLM R20      !I = 0
075067 126 223      CLM R26      !J = 0
075071      PLOOP+  BSZ 0      !REPEAT
075071      PLOOP  BSZ 0      !      REPEAT
075071 124 066 243      STM R24,R66      !      MOVE Na
075074 120 076 243      STM R20,R76      !      MOVE I
075077 316 373 170      JSB =MNMUL3      !      Na * I.
075102 176 055 241      LDM R76,R55      !      MOVE PRODUCT
075105 026 303      ADM R76,R26      !      (Na * I) + J
075107 166 261 326      LDMD R66,=INCRA      !      GET ELEMENT SIZE (8, 4 OR 3)
075112 326
075113 316 373 170      JSB =MNMUL3      !      ((Na * I) + J) * ELE SIZE)
075116 165 261 326      LDMD R65,=TMP1      !      GET Ba
075121 326
075122 055 305      SBM R65,R55      !      ADDR A(I,J)th ELE OF ARRAY
075124 316 007 163      JSB =AFETCH      !      GET VALUE OF A(I,J)
075127 316 326 326      JSB =SAVR20      !      SAVE R20 REGISTERS
075132 140 012 345      PUMD R40,+R12      !      VALUE A(I,J) ON STACK FOR TSTUS
075135 120 260 326      LDBD R20,=TMP2      !      GET USING FLAG FOR TSTUS
075140 326
075141 165 261 326      LDMD R65,=RMEM      !      AMOUNT PREVIOUSLY RESERVED MEMORY
075144 326
075145 006 345      PUMD R65,+R6      !      SAVE IT
075147 223      CLM R65      !      RMEM = 0 ...
075150 263 326 326      STMD R65,=RMEM      !      INITIALIZED FOR PRINTING
075153 316 326 326      JSB =ROMJSB
075156 326 326      DEF TSTUS      !      PLACE VALUE A(I,J) IN PRINT BUFFER
075160 000      BYT 0

```

075161	316	326	326		JSB =RELMEM	!	RELEASE MEM RESERVED BY PRINTING
075164	165	006	343		POMD R65,-R6	!	RESTORE AMOUNT MEMORY RESERVED ...
075167	263	326	326		STMD R65,=RMEM	!	PRIOR TO THE PRINT BUFFER
075172	316	326	326		JSB =RSTR20	!	RESTORE R20 REGISTERS
075175	230				BIN	!	RESET MODE
075176	117	310	300		CMB R17,=300	!	CHECK FOR ERRORS
075201	372	336			JNC PLEND	!	IF (IMAGE ERRORS)
075203	112	261	326		LDMD R12,=TOS	!	CLEAN UP R12
075206	326						
075207	236				RTN	!	EXIT
075210				PLEND	BSZ 0	!	END IF
075210	136	261	326		LDMD R36,=TMP2	!	GET TERMINATOR FLAG
075213	326						
075214	365	336			JPS PNEXT	!	IF (TERMINATOR IS A SLASH)
075216	316	326	326		JSB =COMDMP	!	PRINT THE ELEMENT
075221				PNEXT	BSZ 0	!	END IF
075221	156	261	326		LDMD R56,=MBASE	!	TEMP AREA BASE POINTER.
075224	326						
075225	056	264	041		LDBD R56,X56,ROWFLG	!	GET ROW FLAG.
075230	000						
075231	366	336			JNZ PCOL	!	IF (OUTPUT BY ROWS)
075233	126	211			ICM R26	!	J = J + 1
075235	024	301			CMM R26,R24	!	COMPARE J & Na
075237	360	336			JMP PLEND1		
075241				PCOL	BSZ 0	!	ELSE - (OUTPUT BY COLUMNS)
075241	120	211			ICM R20	!	I = I + 1
075243	022	301			CMM R20,R22	!	COMPARE I & Ma
075245				PLEND1	BSZ 0	!	END IF
075245	372	222			JNC PLOOP	!	UNTIL (J=Na [ROWS] OR I=Ma [COLS])
075247	316	326	326		JSB =COMDMP	!	PRINT CURRENT ROW OR COLUMN
075252	156	220			TSB R56	!	LOOK AT ROW FLAG AGAIN
075254	366	336			JNZ PCOL2	!	IF (OUTPUT BY ROWS)
075256	126	223			CLM R26	!	J = 0
075260	120	211			ICM R20	!	I = I + 1
075262	022	301			CMM R20,R22	!	COMPARE I & Ma
075264	360	336			JMP PLEND2		
075266				PCOL2	BSZ 0	!	ELSE - (OUTPUT BY COLUMNS)
075266	120	223			CLM R20	!	I = 0
075270	126	211			ICM R26	!	J = J + 1
075272	024	301			CMM R26,R24	!	COMPARE J & Na
075274				PLEND2	BSZ 0	!	END IF
075274	373	336			JCY PLEND3		
075276	104	251	070		GTO PLOOP+	!	UNTIL (I=Ma [ROWS] OR J=Na [COLS])
075301	172						
075302				PLEND3	BSZ 0		
075302	260	326	326		LDBD R#,=TMP2+	!	GET CR,LF SUPPRESS FLAG
075305	367	336			JZR PRTEND	!	IF (CR,LF SUPPRESS)
075307	316	103	140		JSB =SETROW	!	DEFAULT TO ROW OUTPUT
075312	236				RTN	!	EXIT
075313				PRTEND	BSZ 0	!	END IF
075313	235				CLE	!	E = 0 : ASSUME NOT PRINT USING
075314	260	326	326		LDBD R#,=USING?	!	GET USING FLAG
075317	367	336			JZR USING-	!	IF (PRINT USING)
075321	234				ICE	!	E = 1 : PRINT USING
075322				USING-	BSZ 0	!	END IF
075322	140	270	326		LDBI R40,=PTR1-+	!	GET NEXT TOKEN
075325	326						
075326	310	002			CMB R40,=2	!	AT END OF ARRAY LIST?
075330	367	336			JZR BLANKL	!	IF (END OF ARRAY LIST)
075332	316	103	140		JSB =SETROW	!	DEFAULT TO ROW OUTPUT.
075335	371	336			JEZ ETRTN	!	IF (PRINT USING)
075337	316	326	326		JSB =ROMJSB		
075342	326	326			DEF PRLINE	!	PROCESS EOLN
075344	000				BYT 0		
075345				ETR TN	BSZ 0	!	END IF
075345	122	223			CLM R22		
075347	273	326	326		STMI R22,=P.PTR	!	RESET PRINTER POINTER
075352	236				RTN	!	EXIT
075353				BLANKL	BSZ 0	!	ELSE - (BETWEEN ARRAYS)
075353	370	336			JEN BLRTN	!	IF (NOT PRINT USING)
075355	231				BCD	!	SET MODE

```

075356 144 223          CLM R44          !      INDICATE ...
075360 211              ICM R44          !      ONE BLANK LINE.
075361 316 326 326      JSB =ROMJSB
075364 326 326          DEF SENDCR          !      PRINT 1 BLANK LINE
075366 000              BYT 0
075367          BLRTN      BSZ 0          !      END IF
075367          BSZ 0          !END IF
075367 236              RTN
075370          !
075370          !      *****
075370          !      *** PRINT CONTENTS OF BUFFER ***
075370          !      *****
075370          !
075370          COMDMP      BSZ 0
075370 110 260 326      LDBD R10,=TMP2+      !GET CR,LF SUPPRESS FLAG
075373 326
075374 366 336          JNZ SUPRES          !IF (NO CR,LF SUPPRESS)
075376 316 326 326      JSB =PUTREG          !      PROTECT REGISTERS ...
075401 140 012 345      PUMD R40,+R12          !      INCLUDING R40
075404 122 270 326      LDBI R22,=P.PTR          !      GET PRINTER POINTER
075407 326
075410 366 336          JNZ DBUFF          !      IF (PRINT BUFFER EMPTY)
075412 250 000          LDB R22,=BLANK          !      GET A BLANK
075414 272 326 326      STBI R22,=P.BUFF          !      PUT IT IN PRINT BUFFER
075417 270 326 326      LDBI R22,=P.FLAG          !      GET PRINTER FLAG
075422 367 336          JZR FULIN          !      JIF FULL LINE JUST PRINTED.
075424          DBUFF      BSZ 0          !      END IF
075424 316 326 326      JSB =ROMJSB
075427 326 326          DEF WRTLIN          !      PRINT CONTENTS OF BUFFER
075431 000              BYT 0
075432 230              FULIN      BIN
075433 122 260 326      LDBD R22,=TMP4          !      GET TRAILING / FLAG
075436 326
075437 367 336          JZR DBUF1          !      IF (TRAILING /)
075441 250 000          LDB R22,=BLANK          !      GET A BLANK
075443 272 326 326      STBI R22,=P.BUFF          !      PUT IT IN PRINT BUFFER
075446 223              CLM R22          !      INDICATE ...
075447 211              ICM R22          !      1 CHAR IN ...
075450 273 326 326      STMI R22,=P.PTR          !      P.PTR
075453          DBUF1      BSZ 0          !      END IF
075453 140 012 343      POMD R40,-R12          !      RESTORE R40
075456 316 326 326      JSB =GETREG          !      RESTORE REGISTERS
075461          SUPRES      BSZ 0          !END IF
075461 236              RTN
075462          !
075462          !
075462          SAVR20      BSZ 0
075462 160 020 241      LDM R60,R20          !MOVE R20 REGISTERS
075465 263 326 326      STMD R60,=TMP3          !SAVE THEM
075470 236              RTN
075471          !
075471          RSTR20      BSZ 0
075471 160 261 326      LDMD R60,=TMP3          !GET SAVED R20 REGISTERS
075474 326
075475 020 243          STM R60,R20          !RESTORE THEM
075477 236              RTN
075500          !***** MAT PRINT ROW ATTRIBUTES TABLE *****
075500 041              BYT 41
075501          !*****
075501          !***      /PRINT\          /STATEMENT NO.\          ***
075501          !*** MAT <          > [USING <          >;] ROW A          ***
075501          !***      \DISP /          \FORMAT STRING/          ***
075501          !***          ***
075501          !*** IN : REL ADDR A          ***
075501          !***          <--- R12          ***
075501          !***          USING? <> 0 THEN HAVE USING CLAUSE          ***
075501          !***          = 0 THEN DO NOT HAVE USING CLAUSE          ***
075501          !***          ***
075501          !*** OUT : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED          ***
075501          !***          BY ROWS (NORMAL DISPLAY)          ***
075501          !*****

```

```

075501      !
075501 236   ROW.      RTN                      !NOTHING TO DO.
075502      !
075502      !***** MAT PRINT COL ATTRIBUTES TABLE *****
075502 041   BYT 41
075503      !*****
075503      !***      /PRINT\                      /STATEMENT NO.\      ***
075503      !*** MAT <      > [USING <      >;] COL A      ***
075503      !***      \DISP /                      \FORMAT STRING/      ***
075503      !***      ***
075503      !*** IN : REL ADDR A      ***
075503      !***      <--- R12      ***
075503      !***      USING? <> 0 THEN HAVE USING CLAUSE      ***
075503      !***      = 0 THEN DO NOT HAVE USING CLAUSE      ***
075503      !***      ***
075503      !*** OUT : STACK POPPED AND OPERAND ARRAY PRINTED OR DISPLAYED      ***
075503      !***      BY COLUMNS      ***
075503      !*****
075503      !
075503      COL.      BSZ 0
075503 230      BIN
075504 140 222      CLB R40                      !READY TO SET FLAG.
075506 210      ICB R40                      !SET FLAG TO DESIGNATE COL OUTPUT.
075507 104 251 104      GTO SETCOL          !GO STORE FLAG.
075512 140

```



```

075513      !          HED MAT READ ROUTINE
075513      !*****          !MAT READ ATTRIBUTES TABLE *****
075513 044      BYT 44
075514      !*****
075514      !*** MAT READ C ***
075514      !*** ELEMENTS OF RESULT ARRAY C ASSIGNED VALUES FROM A ***
075514      !*** LIST OF NUMBERS IN A DATA STATEMENT (MUST BE NUMBERS ***
075514      !*** AND NOT STRINGS). ARRAY ELEMENTS ASSIGNED VALUES ***
075514      !*** IN ORDER FROM LEFT TO RIGHT ON EACH ROW, FROM THE ***
075514      !*** FIRST ROW TO THE LAST. ***
075514      !*** PROGRAMABLE STATEMENT ONLY. ***
075514      !*** ***
075514      !*** IN : REL ADDR C ***
075514      !*** <--- R12 ***
075514      !*** OUT : STACK POPPED AND RESULT ARRAY ASSIGNED. ***
075514      !*****
075514      !
075514      READ.      BSZ 0
075514 116 310 001      CMB R16,=1          !LOOK AT CSTAT
075517 366 336      JNZ RDRUN          !IF (CALCULATOR MODE)
075521 360 336      JMP ERR88          ! MAT READ ILLEGAL IN CALC MODE
075523      RDRUN      BSZ 0          !END IF
075523 316 006 171      JSB =LOCSZ-      !GET Bc, Mc, Nc
075526 122 221      TSM R22          !ROWS = Mc
075530 367 336      RWHIL1      JZR REND1      !WHILE (ROWS <> 0)
075532 124 026 243      STM R24,R26          ! COLS = Nc
075535 367 336      RWHIL2      JZR REND2      ! WHILE (COLS <> 0)
075537 316 326 326      JSB =PUTREG      ! SAVE REGS FROM READDT
075542 165 012 345      PUMD R65,+R12      ! ABS ADDR OF ARRAY 1ST ELE
075545 160 345      PUMD R60,+R12      ! ABS ADDR OF ARRAY NAME
075547 163 343      POMD R63,-R12
075551 120 260 326      LDBD R20,=TYPC      ! TYPE HEADER (TRACE OFF)
075554 326
075555 344      PUBD R20,+R12
075556 222      CLB R20          ! FLAG NOT A STRING
075557 316 326 326      JSB =ROMJSB      ! OFF TO READ DATA STMT.
075562 326 326      DEF READDT      ! VALUE OF ARRAY ELE ON STACK
075564 000      BYT 0
075565 316 326 326      JSB =GETREG      ! RESTORE REGS
075570 316 262 152      JSB =ERRCK      ! WERE THERE ANY ERRORS?
075573 155 261 326      LDMD R55,=INCR      ! GET ELEMENT SIZE
075576 326
075577 316 001 161      JSB =NXTELE      ! GET NEXT ELE ADDR
075602 126 213      DCM R26          ! COLS = COLS - 1
075604 360 327      JMP RWHIL2      ! LOOP
075606      BSZ 0          ! END WHILE
075606 122 213      DCM R22          ! ROWS = ROWS - 1
075610 360 316      JMP RWHIL1      ! LOOP
075612      REND1      BSZ 0          !END WHILE
075612 316 312 154      JSB =CKTRC      !CHECK TRACE.
075615 236      RTN

```

```

075616      !          HED MAT INPUT ROUTINE
075616      !*****  TOKEN 2 ATTRIBUTES TABLE  *****
075616 241      BYT 241
075617      !*****
075617      !*** MAT INPUT C ***
075617      !*** 3 ROUTINES ARE USED TO IMPLEMENT THE MAT INPUT STATEMENT ***
075617      !*** 1). INPUT. -- DISPLAYS PROMPT ON SCREEN ***
075617      !*** 2). INPUN. -- EVALUATES EXPRESSION IN CALCULATOR MODE ***
075617      !*** 3). INCOM. -- STORES EVALUATED EXPRESSION IN ARRAY ***
075617      !***
075617      !*** THE STATEMENT -- MAT INPUT C -- IS PARSED LIKE THIS: ***
075617      !***
075617      !*** 156 \ ***
075617      !*** 260 > INPUN. ***
075617      !*** 370 / ***
075617      !*** 103 -- ARRAY NAME ***
075617      !*** 001 -- LENGTH OF ARRAY NAME ***
075617      !*** 000 \ ***
075617      !*** 000 > ADDR OF ARRAY ***
075617      !*** 000 / ***
075617      !*** 002 -- PLACE ADDR OF ARRAY ON R12 STACK ***
075617      !*** 002 \ ***
075617      !*** 260 > INPUT. ***
075617      !*** 370 / ***
075617      !***
075617      !*** BEFORE THE MAT INPUT TOKEN (002 260 370) IS EXECUTED, ***
075617      !*** THE ARRAY ADDRESS MUST BE ON THE R12 STACK FIRST. THIS ***
075617      !*** IS ACCOMPLISHED BY THE FLAG -- ARYFLG. ***
075617      !***
075617      !*** IF ARYFLG = 0 AND INPUT.(002 260 370), THEN ***
075617      !*** THE ADDR OF THE ARRAY IS NOT ON THE STACK YET. ***
075617      !*** ACTION: RETURN TO SYSTEM. ***
075617      !***
075617      !*** IF ARYFLG = 0 AND INPUN. (156 260 370), THEN ***
075617      !*** ADDR OF ARRAY ON STACK AND INPUT. NOT EXECUTED ***
075617      !*** ACTION: MOVE TOKEN POINTER TO 002 260 370 (TO EXECUTE ***
075617      !*** INPUT) AND RETURN TO SYSTEM. ***
075617      !***
075617      !*** IF ARYFLG <> 0 AND INPUT., THEN ***
075617      !*** ARRAY ADDR ON STACK. ***
075617      !*** ACTION: EXECUTE INPUT. CODE ***
075617      !***
075617      !*** IF ARYFLG <> 0 AND INPUN., THEN ***
075617      !*** R12 STACK CLEAN AND INPUT. HAS BEEN EXECUTED ***
075617      !*** ACTION: EXECUTE INPUN. CODE ***
075617      !***
075617      !*** IN : REL ADDR C \ ASSUMING ARYFLG <> 0 ***
075617      !*** <--- R12 / ***
075617      !*** OUT : ARRAY PROMPT ON SCREEN ***
075617      !*****
075617      !
075617      LST
075617      INPUT. BSZ 0
075617 230      BIN
075620 120 261 326      LDMD R20,=MBASE !GET TEMP STORE PTR
075623 326
075624 155 020 264      LDBD R55,X20,ARYFLG !GET ARRAY FLAG
075627 052 000
075631 366 336      JNZ FLGSET !IF (ARYFLG = 0)
075633 316 326 326      JSB =SVPTRS ! SAVE PTR1
075636 236      RTN ! RETURN TO SYSTEM
075637      FLGSET BSZ 0 !END IF
075637 261 326 326      LDMD R#,=PTR1- !POINT TO FIRST ARRAY TOKEN
075642 213      DCM R# !POINT TO 002 TOKEN
075643 263 326 326      STMD R#,=TMP2 !SAVE IT FOR FETVAR
075646 316 326 326      JSB =SFLG=0 !INIT DSPFLG TO 0.
075651 116 310 001      CMB R16,=1 !LOOK AT CSTAT
075654 366 336      JNZ INDISP !IF (CALCULATOR MODE)
075656 316 326 326 ERR88      JSB =ERROR ! INPUT ILLEGAL ...
075661 130      BYT 88D ! IN CALCULATOR MODE
075662 236      RTN ! EXIT

```

075663		INDISP	BSZ 0	!END IF
075663	316 006 171		JSB =LOCSZ-	!Bc, Mc, Nc
075666	316 116 161		JSB =ZERTST	!TEST FOR NULL ARRAY.
075671	366 336		JNZ NULDSP	!IF (NULL ARRAY)
075673	165 261 326		LDMD R65,=TMP2	! GET POINTER TO ARRAY NAME ADDR
075676	326			
075677	263 326 326		STMD R65,=PTR1	! POINT TO IT
075702	164 271 326		LDMI R64,=PTR1-	! GET ARRAY NAME LENGTH
075705	326			
075706		NAMLEN	BSZ 0	! REPEAT
075706	167 270 326		LDBI R67,=PTR1-	! SCAN PAST ARRAY NAME
075711	326			
075712	164 212		DCB R64	! LENGTH = LENGTH - 1
075714	366 370		JNZ NAMLEN	! UNTIL (LENGTH = 0)
075716	165 271 326		LDMI R65,=PTR1-	! POINT TO NEXT ARRAY
075721	326			
075722	261 326 326		LDMD R65,=PTR1-	! GET IT'S ADDRESS
075725	126 222		CLB R26	! READY TO SET FLAG.
075727	210		ICB R26	! SET FLAG.
075730	104 251 377		GTO NEWAR1	! GO STORE FLAG & EXIT.
075733	377			
075734		NULDSP	BSZ 0	!END IF
075734	164 271 326		LDMI R64,=PTR2+	!GET MAXCOL & MAXROW
075737	326			
075740	136 060 243		STM R36,R60	!INIT ROW = OPTION BASE
075743	062 243		STM R36,R62	!INIT COL = OPTION BASE
075745	160 020 267		STMD R60,X20,ROWCTR	!INIT ROW,INIT COL,MAX COL,MAX ROW
075750	042 000			
075752		DSPAII	BSZ 0	
075752	165 261 326		LDMD R65,=LAVAIL	!GET LAVAIL
075755	326			
075756	263 326 326		STMD R65,=SAVLAV	!AND SAVE IT
075761	261 326 326		LDMD R65,=TMP2	!GET ARRAY NAME ADDRESS
075764	263 326 326		STMD R65,=PTR1	!POINT TO ARRAY NAME ADDRESS
075767	123 250 002		LDB R23,=2	!VARIABLE TOKEN FOR FETVAR.
075772	316 326 326		JSB =FETVAR	!STACK UP ARRAY NAME.
075775		!	UNL	
075775	155 261 326		LDMD R55,=PTR1-	!GET PTR TO 156 260 370
076000	326			
076001	263 326 326		STMD R55,=SAVPC	!MAKES 156 TOK EXECUTED NEXT
076004	124 250 000		LDB R24,=OPEN	!READY TO STACK (.
076007	012 344		PUBD R24,+R12	!STACK IT.
076011	250 004		LDB R24,=4	!REAL INDICATOR.
076013	344		PUBD R24,+R12	!STACK IT.
076014	136 020 265		LDMD R36,X20,ROWCTR	!GET CURRENT ROW NO.
076017	042 000			
076021	316 326 326		JSB =CONBIN	!CONVERT IT TO REAL.
076024	230		BIN	!RESET MODE.
076025	140 012 345		PUMD R40,+R12	!STACK IT.
076030	260 326 326		LDBD R40,=TRCFLG	!VECTOR IF RT BIT =1.
076033	362 336		JOD ONEPAR	!IF (MATRIX)
076035	136 250 000		LDB R36,=COMMA	! READY TO STACK.
076040	344		PUBD R36,+R12	! STACK IT.
076041	250 004		LDB R36,=4	! REAL INDICATOR.
076043	344		PUBD R36,+R12	! STACK IT.
076044	020 265 044		LDMD R36,X20,COLCTR	! GET CURRENT COL.
076047	000			
076050	316 326 326		JSB =CONBIN	! MAKE IT REAL.
076053	230		BIN	! RESET MODE.
076054	140 012 345		PUMD R40,+R12	! STACK IT.
076057		ONEPAR	BSZ 0	!END IF
076057	140 250 000		LDB R40,=CLOSE	!NEED TO STACK A ).
076062	012 344		PUBD R40,+R12	!STACK IT.
076064	250 077		LDB R40,=77	!QUESTION MARK.
076066	344		PUBD R40,+R12	!STACK IT.
076067	130 251 326		LDM R30,=ERRBUF	!TEMP AREA TO UNSTACK IN.
076072	326			
076073	316 326 326		JSB =ROMJSB	
076076	326 326		DEF UNSTAK	!DECOMPILE R12
076100	000		BYT 0	
076101	020 264 054		LDBD R#,X20,DSPFLG	!GET DSPFLG.

```

076104 000
076105 364 336          JNG SFLG=0          !IF (NO MORE ELE IN BUF)
076107 316 326 326      JSB =ROMJSB
076112 326 326          DEF DISP.          !   FIND WHERE CRT IS
076114 000              BYT 0
076115 126 251 326      LDM R26,=ERRBUF     !   BEGINNING OF STRING
076120 326
076121 136 030 241      LDM R36,R30         !   END OF STRING
076124 026 305          SBM R36,R26         !   FIND LENGTH OF STRING
076126 316 326 326      JSB =ROMJSB
076131 326 326          DEF DRV12.          !   PRINT "A(I,J)? "
076133 000              BYT 0
076134 110 251 326      LDM R10,=INPBUF     !   USE INPBUF FOR INPUT
076137 326
076140 116 250 004      LDB R16,=4          !   SET IDLE FOR INPUT
076143 316 326 326      JSB =SET240         !   INPUT ARRAY VALUE
076146          SFLG=0      BSZ 0            !END IF
076146 126 222          CLB R26             !DSPFLG=0.
076150 104 251 377      GTO NEWARY          !GO STORE IT.
076153 377
076154          !
076154          !***** TOKEN 156 ATTRIBUTES TABLE *****
076154 044              BYT 44
076155          !*****
076155          !*** EVALUATES INPUT ENTERED FROM KEYBOARD IN CALCULATOR MODE. ***
076155          !*** NUMERIC EXPRESSIONS, CR, AND COMMA ONLY LEGITAMATE INPUTS ***
076155          !*** ANY ERRORS ARE REPORTED AND INPUT PROMPTED AGAIN. ROUTINE ***
076155          !*** ALSO CAUSES INCOM. CODE TO BE EXECUTED NEXT. ***
076155          !*** IN : CALCULATOR MODE (ASSUME ARYFLG <> 0) ***
076155          !*** OUT : VALUE OF EXPRESSION ***
076155          !*** <--- R12 ***
076155          !*****
076155          !
076155          INPUN.      BSZ 0
076155 120 261 326      LDMD R20,=MBASE       !GET TEMP STORE PTR
076160 326
076161 155 020 264      LDBD R55,X20,ARYFLG  !GET ARRAY FLAG
076164 052 000
076166 366 336          JNZ FLSET           !IF (ARYFLG = 0)
076170 316 326 326      JSB =RSPTRS         !   RESTORE PTR1
076173 155 271 326      LDMI R55,=PTR1+     !   MOVE IT BACK TO 002 260 370
076176 326
076177 020 266 052      STBD R55,X20,ARYFLG  !   SET FLAG TO NON-ZERO VALUE
076202 000
076203 236              RTN                 !   RETURN TO SYSTEM
076204          LST
076204          FLSET      BSZ 0             !END IF
076204 020 264 054      LDBD R#,X20,DSPFLG   !GET DSPFLG.
076207 000
076210 367 336          JZR IPELSE          !IF (1ST OF NEW ARRAY)
076212 104 251 262      GTO INDISP          !   GO DO INITIAL DISPLAY.
076215 173
076216          !
076216          IPELSE      BSZ 0            !ELSE - (NOT 1ST OF NEW ARRAY)
076216          IPLOOP     BSZ 0            !   LOOP
076216 126 251 326      LDM R26,=ERBEND     !   TEMP AREA FOR ...
076221 326
076222 065 243          STM R26,R65         !   CALC. MODE EVALUATION ...
076224 167 222          CLB R67             !   AND INCOM. INSTRUCTION
076226 165 263 326      STMD R65,=PTR2-     !   SET PTR2
076231 326
076232 263 326 326      STMD R65,=STSIZE    !   SAVE FOR PARSER
076235 316 326 326      JSB =GCHAR         !   GET CHARACTER
076240 316 326 326      JSB =ROMJSB
076243 326 326          DEF NUMVA+         !   PARSE NUMERIC VALUE
076245 000              BYT 0
076246 371 336          JEZ ER43            !   IF (NUMERIC INPUT)
076250 114 310 000      CMB R14,=CR         !   ONLY CR AND , VALID NEXT
076253 367 336          JZR INPCO          !   IF (NOT CR)
076255 310 000          CMB R14,=COMMA      !   COMMA ?
076257 366 336          JNZ ER43            !   NOT COMMA GOTO ER43

```

```

076261          INPCO      BSZ 0          !          END IF
076261 164 251 016      LDM R64,=16,72,260,370 !          TRICK TO GTO INCOM.
076264 072 260 370
076267 273 326 326      STMI R64,=PTR2-
076272 114 012 345      PUMD R14,+R12          !          SAVE R14.
076275 120 310 000      CMB R20,=CR          !          CR?
076300 367 336          JZR INPN1.          !          IF (NOT CR)
076302 110 213          DCM R10          !          FOR NEXT PASS.
076304          INPN1.      BSZ 0          !          END IF
076304 116 250 001      LDB R16,=1          !          SET CALC MODE
076307 316 326 326      JSB =SET240          !          FORCE EXIT.
076312          BSZ 0          !          -----
076312 236          RTN          !          -- RETURN --
076313          BSZ 0          !          -----
076313          ER43      BSZ 0          !          ELSE - (NON-NUMERIC INPUT)
076313 316 326 326      JSB =ERROR
076316 053          BYT 43D          !          NEED NUMERIC VALUE
076317          LST
076317          BADINP      BSZ 0
076317 316 326 326      JSB =RELMEM          !          IN CASE TEMP MEM FOR INPUT.
076322 316 326 326      JSB =ROMJSB
076325 326 326          DEF REPORT          !          REPORT ANY ERRORS
076327 000          BYT 0
076330 120 261 326      LDMD R20,=MBASE          !          GET TEMP STORE POINTER
076333 326
076334 165 261 326      LDMD R65,=ERGOTO          !          GET ON ERROR FLAG
076337 326
076340 367 336          JZR REPDSP          !          IF (ON ERROR)
076342 222          CLB R#          !          CLEAR ARRAY FLAG ...
076343 020 266 052      STBD R#,X20,ARYFLG          !          FOR NEW MAT INPUT
076346 000
076347          !          *****
076347          !          *** TOP OF R6:          ***
076347          !          *** 0155 - CALCRU          ***
076347          !          *** 0721 - INTERL          ***
076347          !          *** 6207 - ROM:GO          ***
076347          !          *****
076347 164 006 343      POMD R64,-R6          !          TRASH 2 RETURNS
076352 166 345      PUMD R66,+R6          !          PUT 1 BACK
076354          !          *****
076354          !          *** TOP OF R6:          ***
076354          !          *** 0155 - CALCRU          ***
076354          !          *** 6207 - ROM:GO          ***
076354          !          *****
076354 236          RTN          !          RETURN & BRANCH TO ERROR
076355          LST
076355          REPDSP      BSZ 0          !          END IF
076355 104 251 351      GTO DSPAIJ          !          REDO A(I,J) DISPLAY
076360 173
076361          !          UNL
076361          BSZ 0          !          END IF
076361 136 211          MORROW      ICM R36          !          UP THE ROW COUNT.
076363 020 267 042      STMD R36,X20,ROWCTR          !          SAVE NEW ROW COUNT.
076366 000
076367 130 211          MORCOL      ICM R30          !          UP THE COL COUNT.
076371 020 267 044      STMD R30,X20,COLCTR          !          UPDATE COL COUNT.
076374 000
076375 114 310 000      CMB R14,=CR          !          MORE IN BUFFER?
076400 367 353          JZR REPDSP          !          JIF NO.
076402 104 251 215      GTO IPLOOP          !          LOOP IF MORE IN BUFFER.
076405 174
076406          BSZ 0          !          END LOOP
076406          BSZ 0          !          END IF
076406          !
076406          LST
076406          ER45      BSZ 0
076406 160 261 326      LDMD R60,=TMP4          !          LAST UPDATED STORE ADDR.
076411 326
076412 155 261 326      LDMD R55,=INCR          !          ELE SIZE OF ARRAY
076415 326
076416 157 222          CLB R57

```

```

076420 165 055 303      ADM R65,R55      !MOVE ADDR BACK ONE
076423 160 263 326      STMD R60,=TMP4    !SAVE AS NEXT STORE ADDR
076426 326
076427 316 326 326      JSB =ERROR
076432 055              BYT 45D          !TOO MANY INPUTS
076433 147 260 326      LDBD R47,=TMP1++  !GET ARRAY HEADER
076436 326
076437 206              LRB R47          !GET TRACE BIT IN CY ...
076440 206              LRB R47          !CY = 0 NO TRACE ; CY = 1 TRACE
076441 372 336          JNC ER45TC        !IF (TRACE)
076443 117 312 010      ADB R17,=010      ! MAKE R17 = 310
076446                ER45TC      BSZ 0    !END IF
076446 360 247          JMP BADINP        !REPORT ERROR AND PROMPT AGAIN
076450                !      UNL
076450                !***** STORE INPUT ATTRIBUTES TABLE *****
076450 044              BYT 44
076451                !*****
076451                !*** RESETS PROGRAM MODE.  GETS VALUE OF EXPRESSION ENTERED ***
076451                !*** FROM KEYBOARD AND PLACES IT IN THE CORRECT ARRAY ELEMENT. ***
076451                !***
076451                !*** IN  : EXPRESSION VALUE ***
076451                !*** <--- R12 ***
076451                !*** OUT : STACK POPPED & ARRAY ELEMENT ASSIGNED ***
076451                !*****
076451                !
076451 230      INCOM.      BIN              !RESET MODE TO BINARY.
076452 117 222          CLB R17          !CLEAR CALCULATOR MODE
076454 116 250 002      LDB R16,=2      !SET RUN FOR WHEN NEW DISP.
076457 140 012 343      POMD R40,-R12    !GET VALUE FROM CALC MODE
076462 114 343          POMD R14,-R12    !RESTORE R14.
076464 112 263 326      STMD R12,=TOS    !RESET TOP OF STACK
076467 326
076470 316 301 162      JSB =STOV        !STORE INPUT VAL IN ARRAY.
076473 316 326 326      JSB =RELMEM      !IN CASE FOR-NEXT WAS USED.
076476 165 261 326      LDMD R65,=CALVRB !TRASH ANY CALC VARIABLES.
076501 326
076502 263 326 326      STMD R65,=LAVAIL
076505 120 261 326      LDMD R20,=MBASE   !TEMP STORAGE BASE.
076510 326
076511 164 020 265      LDMD R64,X20,MAXCOL !GET MAXCOL & MAXROW
076514 046 000
076516 220              TSB R#
076517 364 336          JNG VECINP        !JIF VECTOR.
076521 130 265 044      LDMD R30,X20,COLCTR !GET CURRENT COL COUNT.
076524 000
076525 064 301          CMM R30,R64      !COMPARE IT TO MAXCOL VALUE.
076527 366 236          JNZ MORCOL        !JIF MORE TO CURRENT ROW.
076531 261 326 326      LDMD R30,=PGMOPT  !NEED TO RESET COL COUNT.
076534 213              DCM R30          !ADJUST IT TO ACTUAL VALUE.
076535 215              TCM R30          !ADJUST IT TO ACTUAL VALUE.
076536 213              DCM R30          !OFFSET IT TEMPORARILY.
076537                LST
076537 136 020 265 VECINP LDMD R36,X20,ROWCTR !GET CURRENT ROW COUNT.
076542 042 000
076544 066 301          CMM R36,R66      !DONE WITH THIS ARRAY?
076546 366 211          JNZ MORROW
076550 165 261 326      LDMD R65,=SAVPC   !GET OLD TOKEN POINTER
076553 326
076554 213              DCM R65
076555 213              DCM R65
076556 213              DCM R65          !POINT TO 156 TOKEN
076557 263 326 326      STMD R65,=PTR1-  !RESTORE TOKEN POINTER
076562 120 270 326      LDBI R20,=PTR1-+ !GET NEXT TOKEN
076565 326
076566 126 222          CLB R26          !CLEAR DISPLAY FLAG
076570 114 310 000      CMB R14,=CR      !BUFFER EMPTY ?
076573 367 336          JZR FINARY        !IF (MORE CHARS IN BUFFER)
076575 316 326 326      JSB =TSTEN+      ! CHECK FOR CR, @, & !.
076600 371 204          JEZ ER45          ! JIF TOO MANY INPUTS.
076602 310 034          CMB R#,=34        ! CHECK FOR ELSE
076604 367 200          JZR ER45          ! JIF TOO MANY INPUTS

```

```

076606 126 212          DCB R26          !   SET DISPLAY FLAG NEGATIVE
076610 360 336          JMP NEWAR1
076612          FINARY   BSZ 0          !ELSE - (NO MORE CHARS IN BUFFER)
076612          !! --- changes:
076612          !!       JSB =TSTEN+      !   CHECK FOR CR, @, !
076612          !!       JEZ CLRAR1      !   JIF TOKEN CR, @, !
076612          !!       CMB R#,,=34      !   CHECK FOR ELSE
076612 120 310 016      CMB R20,,=16      !   CHECK FOR
076615 366 336          JNZ CLRARY      !   IF (TOKEN = CR, @, !, ELSE) THEN
076617          !! CLRAR1   BSZ 0
076617          !! --- end of changes
076617 261 326 326      LDMD R20,,=MBASE      !       GET TEMP STORE POINTER
076622 126 020 266      STBD R26,X20,ARYFLG      !       CLEAR FLAG FOR NEXT MAT INPUT
076625 052 000
076627          CLRARY   BSZ 0          !   END IF
076627 126 210          ICB R26          !   SET DISPLAY FLAG TO 1
076631          NEWAR1   BSZ 0          !END IF
076631 165 213          DCM R65          !POINT TO ADDR OF NAME OF NEXT ARRAY
076633 263 326 326      STMD R65,,=TMP2      !SAVE IT FOR FETVAR
076636          NEWARY   BSZ 0
076636 120 261 326      LDMD R20,,=MBASE      !GET TEMP STORE PTR
076641 326
076642 126 020 266      STBD R26,X20,DSPFLG      !STORE IT.
076645 054 000
076647 236          RTN
076650          !          UNL

```

```

076650      !      HED REDIMENSION ROUTINES
076650      !***** REDIM 1 SUBSCRIPT ROUTINE *****
076650 032      BYT 32
076651      !*****
076651      !*** REDIM C(M) ***
076651      !*** SPECIFIES NEW UPPER BOUND OF VECTOR. ***
076651      !*** NUMBER OF SUBSCRIPTS MUST BE SAME AS IN ORIGINAL DIM. ***
076651      !*** TOTAL NUMBER OF REDIM ELEMENTS CAN NOT EXCEED NUMBER ***
076651      !*** ORIGINALLY DIMENSIONED. ***
076651      !*** IN : REL ADDR C ***
076651      !*** SUBSCRIPT M ***
076651      !*** <--- R12 ***
076651      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED. ***
076651      !*****
076651      !
076651      RDIM1      BSZ 0
076651 316 326 326      JSB =ONEB      !POP STACK & INT TO R46
076654 146 076 243      STM R46,R76      !SAVE ROW PARAMETER
076657 316 326 326      JSB =VECMAT      !SEE IF VECTOR OR MATRIX
076662 124 223      CLM R24
076664 211      ICM R24      !COL 1 FOR VECTOR
076665 122      DRP R22      !POINTS TO ROW POSITION
076666 076      ARP R76      !POINTS TO ROW PARAMETER
076667 316 326 326      JSB =NEGDIM      !SEE IF DIMENSIONS ARE NEGATIVE
076672 370 336      JEN RD1END      !IF (2-DIMENSIONAL MATRIX)
076674 104 251 326      LDM R4,=RDER33      ! ERROR -- CHANGE # DIMS
076677 326
076700      RD1END      BSZ 0      !END IF
076700 360 336      JMP RD2END      !GO AND REDIM ARRAY
076702      !
076702      !***** REDIM 2 VAR ROUTINE *****
076702 032      BYT 32
076703      !*****
076703      !*** REDIM C(M,N) ***
076703      !*** SPECIFIES NEW UPPER BOUNDS OF MATRIX. ***
076703      !*** NUMBER OF SUBSCRIPTS MUST BE SAME AS IN ORIGINAL DIM. ***
076703      !*** TOTAL NUMBER OF REDIM ELEMENTS CAN NOT EXCEED NUMBER ***
076703      !*** ORIGINALLY DIMENSIONED. ***
076703      !*** IN : REL ADDR C ***
076703      !*** SUBSCRIPT M ***
076703      !*** SUBSCRIPT N ***
076703      !*** <--- R12 ***
076703      !*** OUT : STACK POPPED AND ARRAY REDIMENSIONED. ***
076703      !*****
076703      !
076703      RDIM2      BSZ 0
076703 316 326 326      JSB =TWOB      !ROW,COL BIN INTS-R46,R56.
076706 146 074 243      STM R46,R74      !SAVE COL PARAMETER
076711 156 076 243      STM R56,R76      !SAVE ROW PARAMETER
076714 316 326 326      JSB =VECMAT      !SEE IF VECTOR OR MATRIX
076717 122      DRP R22      !POINT TO ROW POSITION
076720 076      ARP R76      !POINT TO ROW PARAMETER
076721 316 326 326      JSB =NEGDIM      !SEE IF DIMENSION IS NEGATIVE
076724 124      DRP R24      !POINT TO COL POSITION
076725 074      ARP R74      !POINT TO COL PARAMETER
076726 316 326 326      JSB =NEGDIM      !SEE IF DIMENSION IS NEGATIVE
076731 371 336      JEZ RD2END      !IF (1-DIMENSIONAL VECTOR)
076733 104 251 326      LDM R4,=RDER33      ! ERROR -- CHANGE # DIMS
076736 326
076737      RD2END      BSZ 0      !END IF
076737 235      CLE
076740 233      DCE      !FLAG AS C TYPE MATRIX
076741 360 336      JMP REDIM.      !REDIMENSION THE MATRIX
076743      !
076743      ! *****
076743      ! *** TEST FOR NEGATIVE SUBSCRIPT ***
076743      ! *****
076743      !
076743      !
076743      NEGDIM      BSZ 0
076743 241      LDM R#,R#      !LOAD ROW OR COL PARAM
076744 365 336      JPS NEGDEND      !IF (DIM < 0)

```



```

076746 120 006 343          POMD R20,-R6          !   TRASH 1 RETURN
076751 316 326 326          JSB =ERROR
076754 131                  BYT 89D              !   REPORT ERROR 89 -- ILLEGAL PARAM
076755 236                  RTN
076756                      NEGEND   BSZ 0          !END IF
076756 036 303              ADM R#,R36            !ADD IN THE OPTION BASE
076760 236                  RTN
076761                      !
076761                      ! *****
076761                      ! ***   SEE IF VECTOR OR MATRIX   ***
076761                      ! *****
076761                      !
076761                      !
076761                      !
076761                      !
076761                      !
076761 120 261 326          VECMAT   BSZ 0
076761                      LDMD R20,=MBASE        !GET TEMP STORE POINTER
076764 326
076765 165 223              CLM R65
076767 210                  ICB R65              !FLAG # DIMS ALREADY CHECKED
076770 020 266 053          STBD R65,X20,RDMFLG   !SET REDIM FLAG
076773 000
076774 012 343              POMD R65,-R12         !GET REL ADDR OF ARRAY
076776 345                  PUMD R65,+R12         !COPY FOR REDIM.
076777 345                  PUMD R65,+R12         !REL ADDR FOR LOCSZ-
077000 316 006 171          JSB =LOCSZ-          !1-D OR 2-D MATRIX
077003 165 012 343          POMD R65,-R12         !GET REL ADDR OF MATRIX
077006 136 213              DCM R36              !0 IF OPTION BASE 1 AND ...
077010 215                  TCM R36              !1 IF OPTION BASE 0
077011 236                  RTN
077012                      !
077012                      ! *****
077012                      ! *   SUBROUTINE REDIM                               *
077012                      ! *                                                                 *
077012                      ! * THIS ROUTINE IS CALLED TO REDIMENSION AN ARRAY          *
077012                      ! * OR TO RETURN THE CURRENT DIMENSIONS.                  *
077012                      ! *                                                                 *
077012                      ! * INPUT:                                                  *
077012                      ! *                                                                 *
077012                      ! * R65   = 3 BYTE REL. ADDR. TO ARRAY HEADER              *
077012                      ! * R22   = MAX ROW OR ZERO                             *
077012                      ! * R24   = MAX COL OR ZERO                             *
077012                      ! *                                                                 *
077012                      ! * OUTPUTS                                                *
077012                      ! *                                                                 *
077012                      ! * IF R22 = -1 & R24 = 0:                                *
077012                      ! *   -- R22 = Ma : MAX ROW OF OPERAND ARRAY A OR          *
077012                      ! *           Mb : MAX ROW OF OPERAND ARRAY B OR          *
077012                      ! *           Mc : MAX ROW OF RESULT  ARRAY C            *
077012                      ! *   -- R24 = Na : MAX COL OF OPERAND ARRAY A OR          *
077012                      ! *           Nb : MAX COL OF OPERAND ARRAY B OR          *
077012                      ! *           Nc : MAX COL OF RESULT  ARRAY C            *
077012                      ! *   -- R36 = OPTION BASE (1 OPTION BASE 1; 0 OPTION BASE 0) *
077012                      ! *   -- R55 = ARRAY SIZE                                *
077012                      ! *   -- R60 = ABS ADDR OF ARRAY NAME                    *
077012                      ! *   -- R65 = Ba : BASE ADDR (1st ARRAY ELE) OF OPERAND ARRAY A OR *
077012                      ! *           Bb : BASE ADDR (1st ARRAY ELE) OF OPERAND ARRAY B OR *
077012                      ! *           Bc : BASE ADDR (1st ARRAY ELE) OF RESULT  ARRAY C *
077012                      ! *   -- PTR2- POINTS TO BASE ADDRESS OF ARRAY (Ba, Bb or Bc) *
077012                      ! *   -- TYPE INFO (1 BYTE) IN TYPB,TYPEB,TYPEC          *
077012                      ! *   -- ELE SIZE (2 BYTES) IN INCRA, INCRB, INCRB        *
077012                      ! *   -- TRACE & VECTOR INFO (1 BYTE) IN TRCFLG FOR C-TYPE ARRAY *
077012                      ! *   -- E = 0 IF MATRIX                                *
077012                      ! *   -- E # 0 IF VECTOR                                *
077012                      ! *                                                                 *
077012                      ! * IF R22 & R24 ARE NON-NEG:                            *
077012                      ! *   -- ARRAY WILL BE REDIMENSIONED                      *
077012                      ! *           (NEW ROW AND COL INFO IN ARRAY HEADER)        *
077012                      ! *   -- R60 = ABS ADDR OF ARRAY NAME                    *
077012                      ! *   -- R65 = Ba : BASE ADDR (1st ARRAY ELE) OF OPERAND ARRAY A OR *
077012                      ! *           Bb : BASE ADDR (1st ARRAY ELE) OF OPERAND ARRAY B OR *
077012                      ! *           Bc : BASE ADDR (1st ARRAY ELE) OF RESULT  ARRAY C *
077012                      ! *   -- R55 = NEW ARRAY SIZE                          *
077012                      ! *           (OLD ARRAY SIZE STILL IN ARRAY HEADER)      *

```

```

077012      !*      -- R24 = 1,0 IF VECTOR      *
077012      !*      -- PTR2- POINTS TO 1ST ELEMENT IN ARRAY      *
077012      !*      -- TYPE INFO (1 BYTE) IN TYPB, TYPB, TYPB      *
077012      !*      -- ELE SIZE (2 BYTES) IN INCRA, INCRB, INCRB      *
077012      !*      -- TRACE & VECTOR INFO (1 BYTE) IN TRCFLG FOR C-TYPE ARRAY      *
077012      !*      *
077012      !*      ALL ROW & COL DIMENSIONS ARE REL TO OPTION      *
077012      !*      BASE 1.      *
077012      !*      *
077012      !*      ENTRY REDIM. WILL USE THE OPTION BASE      *
077012      !*      OF THE PROGRAM WITHIN WHICH THE ARRAY      *
077012      !*      RESIDES.      *
077012      !*      *
077012      !*      ENTRY RUDIM WILL SET R22=-1, CLR R24, & REDIM.      *
077012      !*****
077012      !
077012 122 223      RUDIM      CLM R22
077014 213          DCM R22
077015 124 223          CLM R24
077017 360 336          JMP REDIM.
077021      !****      TOKEN 157 ATTRIBUTE TABLE      *****
077021 000 241          BYT 0,241
077023          LST
077023 232      REDIM.      SAD
077024          !          UNL
077024 170 006 345          PUMD R70,+R6          !SAVE R70
077027 316 326 326          JSB =FETSVA          !PTR2 ARRAY SIZE,R70 ARRAY NAME,R46 HEADER
077032 160 070 241          LDM R60,R70          !MOVE ABS ADDR ARRAY NAME
077035 176 251 010          LDM R76,=10,0          !INCREMENT AMOUNT IF REAL.
077040 000
077041 146 035 242          STB R46,R35          !MOVE HEADER FOR TESTING
077044 135 317 060          ANM R35,=60          !ISOLATE TYPE
077047 175 240          LDB R75,R35          !MOVE TYPE.
077051 310 020          CMB R75,=20          !REAL TYPE?
077053 176          DRP R76          !SET ARP FOR LATER.
077054 372 336          JNC ABCTST          !IF (SHORT OR INTEGER)
077056 367 336          JZR INTINC          ! IF (SHORT)
077060 250 004          LDB R#,=4          ! INC AMT FOR SHORT
077062 360 336          JMP INTEND
077064          INTINC      BSZ 0          ! ELSE - (INTEGER)
077064 250 003          LDB R#,=3          ! INC AMT FOR INTEGER
077066          INTEND      BSZ 0          ! END IF
077066          ABCTST      BSZ 0          !END IF
077066 165 261 326          LDMD R65,=PTR2-          !GET ARRAY SIZE ADDR
077071 326
077072 315 007 000          SBM R65,=7,0,0          !FIND ADDR 1ST ARRAY ELE
077075 000
077076      !
077076      !=====
077076      !====      ARRAYS ARE OF THREE TYPES: A, B OR C      ====
077076      !====          C = B * A          =====
077076      !====      TYPE A: 1ST OPERAND ARRAY          =====
077076      !====      TYPE B: 2ND OPERAND ARRAY          =====
077076      !====      TYPE C: RESULT ARRAY          =====
077076      !=====
077076      !
077076 370 336          JEN BCTST          !IF (TYPE A ARRAY)
077100 175 263 326          STMD R75,=TYPB          ! SAVE TYPB AND INCRA
077103 326
077104 165 263 326          STMD R65,=TMP1          ! SAVE Ba
077107 326
077110 263 326 326          STMD R65,=TMP1+          ! ANOTHER COPY
077113 360 336          JMP DOLOC          !
077115 233      BCTST      DCE
077116 370 336          JEN CTYPE          !ELSE IF (TYPE B ARRAY)
077120 175 263 326          STMD R75,=TYPB          ! SAVE TYPB AND INCRB
077123 326
077124 165 263 326          STMD R65,=TMP2          ! SAVE Bb
077127 326
077130 263 326 326          STMD R65,=TMP2+          ! ANOTHER COPY
077133 360 336          JMP DOLOC

```

```

077135          CTYPE      BSZ 0          !ELSE (TYPE C ARRAY)
077135 160 263 326      STMD R60,=TMP4      !   SAVE Bc
077140 326
077141 165 263 326      STMD R65,=TMP3++      !   ANOTHER COPY
077144 326
077145 175 263 326      STMD R75,=TYPC      !   SAVE TYPC AND INCRC
077150 326
077151 146 262 326      STBD R46,=TMP1++      !   SAVE TARGET HEADER
077154 326
077155 316 326 326      JSB =TRCRST          !   ISOLATE TRACE FLAG
077160          DOLOC      BSZ 0          !END IF
077160 151 271 326      LDMI R51,=PTR2-      !GET SIZE, ROWS & COLS FROM HEADER
077163 326
077164 134 223          CLM R34
077166 211          ICM R34          !ASSUME OPTION BASE 0
077167 165 261 326      LDMD R65,=PTR2      !GET LOCATION OF ARRAY
077172 326
077173 321 326 326      CMMD R65,=NXTMEM      !CALC VAR < NXTMEM & PRGM VAR > NXTMEM
077176 372 336      JNC OPTEND      !IF (PROGRAM VARIABLE)
077200 134 261 326      LDMD R34,=PGMOPT      !   GET PROGRAM MODE OPTION BASE
077203 326
077204          OPTEND      BSZ 0          !END IF
077204 136 022 241      LDM R36,R22      !LOOK AT ROW ...
077207 024 225      ORM R36,R24      !AND COL PARAMETERS
077211 365 336      JPS REDIM1      !IF (REDIM--NO REDIMENSION)
077213 122 053 241      LDM R22,R53      !   GET MAX ROW
077216 034 303      ADM R22,R34      !   ADD IN OPTION BASE TO NUM ROWS
077220 235      CLE      !   FLAG AS 2-D MATRIX
077221 124 051 241      LDM R24,R51      !   GET MAX COL
077224 365 336      JPS RED5      !   IF (VECTOR)
077226 223      CLM R24
077227 211      ICM R24      !       1 IN COL PARAM
077230 233      DCE      !       FLAG AS VECTOR
077231 360 336      JMP RED6
077233          RED5      BSZ 0          !   ELSE - (2-D MATRIX)
077233 034 303      ADM R#,R34      !       ADD IN OPTION BASE TO NUM COLS
077235          RED6      BSZ 0          !   END IF
077235 134 213      DCM R34      !   1 IF OPTION BASE 1 AND ...
077237 215      TCM R34      !   0 IF OPTION BASE 0
077240 036 243      STM R34,R36      !   RETURN AS PARAMETER IN R36
077242 104 251 377      GTO REDEND
077245 377
077246          !
077246          RDER33      BSZ 0
077246 316 326 326      JSB =ERROR+
077251 316 326 326      JSB =ERROR      !   ATTEMPT TO CHANGE ...
077254 011      BYT 009D      !   NUMBER OF DIMENSIONS
077255 236      RTN
077256          !
077256          REDIM1      BSZ 0          !ELSE - (REDIMENSION)
077256 134 221      TSM R34      !   LOOK AT OPTION BASE
077260 367 336      JZR REDIM2      !   IF (OPTION BASE 0)
077262 316 116 161      JSB =ZERTST      !       CHECK FOR NULL ARRAY.
077265 367 336      JZR RDIMER      !       JIF NULL ARRAY -- DIM SIZE ERROR
077267          REDIM2      BSZ 0          !   END IF
077267 176 022 241      LDM R76,R22      !   GET ROW PARAMETER
077272 034 305      SBM R76,R34      !   SUBTRACT OPTION BASE
077274 074 243      STM R76,R74      !   MOVE ROW PARAMETER
077276 024 241      LDM R76,R24      !   GET COL PARAMETER
077300 034 305      SBM R76,R34      !   SUBTRACT OPTION BASE
077302 136 261 326      LDMD R36,=MBASE      !   GET TEMP STORE PTR
077305 326
077306 145 036 264      LDBD R45,X36,RDMFLG      !   GET REDIM FLAG
077311 053 000
077313 366 336      JNZ RDMF#0      !   IF (REDIM FLAG = 0)
077315 164 053 241      LDM R64,R53      !       MOVE HEADER ROWS
077320 166 051 241      LDM R66,R51      !       MOVE HEADER COLS
077323 311 377 377      CMM R66,=377,377      !       COMPARE WITH VECTOR INDICATOR
077326 366 336      JNZ NTVC1      !       IF (VECTOR)
077330 124 311 001      CMM R24,=1,0      !       LOOK AT COL PARAM
077333 000

```

077334	366	336		JNZ NTVC1	!	JIF YES
077336	176	066	241	LDM R76,R66		
077341				BSZ 0	!	END IF
077341	164	074	227	NTVC1 XRM R64,R74		
077344	166	064	227	XRM R66,R64		
077347	317	000	200	ANM R66,=000,200		
077352	367	336		JZR RDMF1	!	IF (VECTOR <-- MATRIX)
077354	170	006	343	POMD R70,-R6	!	CLEAN UP STACK
077357	237			PAD	!	" " "
077360	360	264		JMP RDER33	!	# DIMS ERROR
077362				BSZ 0	!	END IF
077362				RDMF#0 BSZ 0	!	ELSE - (REDIM FLAG <> 0)
077362	222			CLB R#	!	SET REDIM FLAG TO 0
077363	036	266	053	STBD R#,X36,RDMFLG		
077366	000					
077367				RDMF1 BSZ 0	!	END IF
077367	155	006	345	PUMD R55,+R6	!	SAVE HEADER SIZE
077372	170	345		PUMD R70,+R6	!	SAVE REDIM ROW & COL
077374	130	261	326	LDMD R30,=INCR	!	GET ELEMENT SIZE
077377	326					
077400	316	372	167	JSB =NUMBYT	!	SIZE OF REDIM MATRIX
077403	170	006	343	POMD R70,-R6	!	RESTORE REDIM ROW & COL
077406	165	343		POMD R65,-R6	!	RESTORE HEADER SIZE
077410	055	301		CMM R65,R55	!	COMPARE HEADER SIZE & REDIM SIZE
077412	365	336		JPS REDIM3	!	IF (HEADER SIZE < REDIM SIZE)
077414	316	326	326	RDIMER JSB =ERROR+		
077417	316	326	326	JSB =ERROR	!	ATTEMPT TO REDIMENSION ARRAY ...
077422	015			BYT 013D	!	BEYOND ORIGINAL SIZE
077423	360	336		JMP REDIM4	!	
077425				REDIM3 BSZ 0	!	ELSE - (HEADER SIZE >= REDIM SIZE)
077425	235			CLE	!	FLAG AS 2-D MATRIX
077426	166	271	326	LDMI R66,=PTR2	!	GET COL OF C
077431	326					
077432	311	377	377	CMM R66,=377,377	!	COMPARE WITH VECTOR INDICATOR
077435	366	336		JNZ REDEX1	!	IF (VECTOR)
077437	076	243		STM R66,R76	!	MOVE VECTOR INDICATOR
077441	233			DCE	!	FLAG AS VECTOR
077442				REDEX1 BSZ 0	!	END IF
077442	176	030	243	STM R76,R30	!	SAVE COL
077445	174	032	243	STM R74,R32	!	SAVE ROW
077450	030	241		LDM R74,R30	!	GET REARRANGED COL & ROW
077452	273	326	326	STMI R74,=PTR2	!	NEW COL & ROW IN ARRAY HEADER
077455				REDIM4 BSZ 0	!	END IF
077455				REDEND BSZ 0	!	END IF
077455	165	261	326	LDMD R65,=PTR2	!	GET ABS ADDR OF 1ST ELE
077460	326					
077461	170	006	343	POMD R70,-R6	!	RESTORE R70
077464	237			PAD		
077465	236			RTN		
077466				!		
077466	136	222		ERROR+ CLB R36		
077470	320	326	326	CMBD R36,=ERRORS	!	ALREADY SET?
077473	366	336		JNZ DOERR+	!	IF (NO OTHER ERROR FLAGGED)
077475	250	260		LDB R36,=ROM#	!	SELECT MY ROM ...
077477	262	326	326	STBD R36,=ERRROM	!	FOR ERRORS
077502				DOERR+ BSZ 0	!	END IF
077502	236			RTN		
077503				!		
077503				TRCRST BSZ 0		
077503	135	260	326	LDBD R35,=TMP1++	!	GET HEADER
077506	326					
077507	317	002		ANM R35,=2	!	ISOLATE TRACE FLAG
077511	204			LLB R35	!	PUT IT IN LEFT NIBBLE
077512	204			LLB R35		
077513	204			LLB R35		
077514	262	326	326	STBD R35,=TRCFLG	!	SAVE IT IN TRACE FLAG
077517	236			RTN		
077520				!		
077520				LST		
077520				PUTREG BSZ 0		
077520				!		
077520				UNL		

```

077520 150 012 345      PUMD R50,+R12
077523 160 345          PUMD R60,+R12
077525 170 345          PUMD R70,+R12
077527 174 000 241      LDM R74,R0
077532 012 345          PUMD R74,+R12
077534 176 014 241      LDM R76,R14
077537 012 345          PUMD R76,+R12
077541 170 020 241      LDM R70,R20
077544 012 345          PUMD R70,+R12
077546 030 241          LDM R70,R30
077550 012 345          PUMD R70,+R12
077552 236              RTN
077553                  !
077553                  LST
077553      GETREG      BSZ 0
077553                  !
077553      UNL
077553 170 012 343      POMD R70,-R12
077556 030 243          STM R70,R30
077560 012 343          POMD R70,-R12
077562 020 243          STM R70,R20
077564 176 012 343      POMD R76,-R12
077567 014 243          STM R76,R14
077571 174 012 343      POMD R74,-R12
077574 000 243          STM R74,R0
077576 170 012 343      POMD R70,-R12
077601 160 343          POMD R60,-R12
077603 150 343          POMD R50,-R12
077605 236              RTN
077606                  !
077606                  !
077606      !*****
077606      !***      ERROR MESSAGE TABLE      ***
077606      !*****
077606      !
077606      !
077606      ! ROM      BIN
077606      !      ! (DEC)      (OCT)
077606      !
077606 200      ERMSG      BYT 200      !      00      377
077607 200      BYT 200      !      01      376
077610 200      BYT 200      !      02      375
077611 200      BYT 200      !      03      374
077612 200      BYT 200      !      04      373
077613 200      BYT 200      !      05      372
077614 200      BYT 200      !      06      371
077615 200      BYT 200      !      07      370
077616 200      BYT 200      !      08      367
077617 043 040 104      ASP "# DIMS"      !      09      366
077622 111 115 323
077625 116 117 124      ASP "NOT A 3-VECTOR"      !      10      365
077630 040 101 040
077633 063 055 126
077636 105 103 124
077641 117 322
077643 104 111 115      ASP "DIM MISMATCH"      !      11      364
077646 040 115 111
077651 123 115 101
077654 124 103 310
077657 115 101 124      ASP "MATRIX ROM"      !      12      363
077662 122 111 130
077665 040 122 117
077670 315
077671 104 111 115      ASP "DIM SIZE"      !      13      362
077674 040 123 111
077677 132 305
077701 116 117 124      ASP "NOT SQUARE"      !      14      361
077704 040 123 121
077707 125 101 122
077712 305
077713 116 117 116      ASP "NON-VECTOR"      !      15      360
077716 055 126 105
077721 103 124 117

```

```

077724 322
077725 377          BYT 377
077726      !
077726      !*****
077726      !***      EXTERNAL LABEL TABLE      ***
077726      !*****
077726      !
077726      CALSCN      DAD 72352          !SCANN WITH REG. PROTECT
077726      PSH45       DAD 24476          !PUSHES TOKEN, 3 BLANKS, LENGTH
077726      FTR61       DAD 55514          !PUTS 1 IN R40
077726      SETR22      DAD 01255          !USES INPBUF FOR INPUT
077726      TSTUS       DAD 72277          !PRINT NUMERIC ROUTINE
077726      MPROI+      DAD 53522          !MPY SEPID REAL OR INT
077726      READDT      DAD 76162          !READ DATA STATEMENT
077726      WRTLIN      DAD 71750          !SEND MESSAGE TO CRT
077726      SALT       DAD 21727          !TO SEARCH ASCII TABLES OF ROM
077726      TMP1+      DAD 104465         !3 BYTES PAST TMP1
077726      TMP1++     DAD 104470         !6 BYTES PAST TMP1
077726      TMP2+      DAD 104475         !3 BYTES PAST TMP2
077726      TMP3+      DAD 104504         !2 BYTE PAST TMP3
077726      TMP3++     DAD 104507         !5 BYTES PAST TMP3
077726      TMP4+      DAD 104517         !5 BYTES PAST TMP4
077726      PGMOPT     DAD 104214         !OPTION BASE
077726      LST
077726      !
077726      !      -----
077726      !      --- ROM2 ---
077726      !      -----
077726      NORM2      DAD 60017          !NORM
077726      ABSUM2     DAD 60051          !SUM OF ABSOLUTE VALUE
077726      SUM2       DAD 60056          !SUM OF ARRAY ELEMENTS
077726      CSUM2      DAD 60123          !SUM OF COLUMN ELEMENTS
077726      RSUM2      DAD 60204          !SUM OF ROW ELEMENTS
077726      RNORM2     DAD 60373          !NORM OF ROW ELEMENTS
077726      CNORM2     DAD 60422          !NORM OF COL ELEMENTS
077726      CROSS2     DAD 60667          !CROSS PRODUCT
077726      DOT2       DAD 61103          !DOT PRODUCT
077726      COLINT     DAD 60474          !INITIALIZE COLUMN
077726      COLNXT     DAD 60523          !NEXT COLUMN
077726      IDN2       DAD 61234          !IDENTITY
077726      EQUA2      DAD 61340          !INT TO REAL CONVER
077726      EQUA09     DAD 61343          !NO INT TO REAL CONVER
077726      FIND#/     DAD 61454          !FIND # A AND FINAL /
077726      TRNPS2     DAD 61646          !TRANSPOSE
077726      LDIM2      DAD 62355          !LOWER BOUND
077726      UDIM2      DAD 62363          !UPPER BOUND

```

```

000000          LST
000000      !*****
000000      !***          MATRIX ROM - 2          ***
000000      !*****
000000      !
000000      !*****
000000      !***          CONSTANTS          ***
000000      !*****
000000      !
000000          LST
000000      RNORMX      EQU 71
000000      CNORMX      EQU 73
000000      TOK#         EQU 43
000000      QUOTE        EQU 5
000000      USLNO        EQU 347
000000      QUOT         EQU 42
000000      RTPARN       EQU 51
000000      LTPARN       EQU 50
000000      COMMA        EQU 54
000000      SLASH        EQU 57
000000      X            EQU 130
000000      PRTOK        EQU 153
000000      DSTOK        EQU 126
000000      !           UNL
000000      !
000000      !*****
000000      !***          ROM INFORMATION          ***
000000      !*****
000000      !
000000          GLO GLOBAL
060000          ABS 60000                      !PLUG IN ROM ADDRESS
060000      !
060000      261          BYT 261                      !ROM NUMBER
060001      117          BYT 117                  !INVERSE ROM NUMBER
060002      !
060002      !*****
060002      !***          SYSTEM TABLES          ***
060002      !*****
060002      !
060002      326 326      DEF RUNTIM
060004      326 326      DEF PARSE
060006      326 326      DEF TOKS
060010      326 326      DEF ERMSG
060012      326 326      DEF INIT
060014      !
060014      !*****
060014      !***          SHELL          ***
060014      !*****
060014      !
060014      RUNTIM      BSZ 0
060014      PARSE       BSZ 0
060014      236      INIT      RTN
060015      377      TOKS      BYT 377
060016      377      ERMSG     BYT 377
060017      !

```

```

060017      !          HED (R,C,F)NORM, ABSUM, SUM, CSUM, RSUM ROUTINES
060017      !*****
060017      !*** FNORM (A): SQUARE ROOT OF THE SUM OF THE SQUARES ***
060017      !***          OF THE ELEMENTS IN THE ARRAY          ***
060017      !*** IN          : RELATIVE ADDRESS A          ***
060017      !***          <--- R12          ***
060017      !*** OUT          : NORM OF A          ***
060017      !***          <--- R12          ***
060017      !*****
060017      !
060017      LST
060017 170 222      NORM2      CLB R70          !READY TO SET FLAG.
060021      !          UNL
060021      210          ICB R70          !SAYS DOING NORM.
060022 316 326 326      JSB =SUM.++          !GO FIND SUM OF SQUARES.
060025 231          BCD          !MODE FOR SQRT.
060026 144 310 377      CMB R44,=377          !INTEGER RESULT?
060031 372 336          JNC RLSQRT          !IF (INTEGER)
060033 140 012 345      PUMD R40,+R12          ! INT RESULT ON STACK
060036 316 326 326      JSB =ONER          ! DEMAND A REAL
060041 316 326 326      JSB =SEP10          ! SEPARATE IT
060044      RLSQRT      BSZ 0          !END IF
060044 316 326 326      JSB =SQR30          !SQRT GIVES THE NORM
060047 360 336          JMP PACKIT
060051      !
060051      !*****
060051      !*** ABSUM (A) : SUM OF THE ABSOLUTE VALUES OF THE ***
060051      !***          ELEMENTS IN A          ***
060051      !*** IN          : RELATIVE ADDRESS A          ***
060051      !***          <--- R12          ***
060051      !*** OUT          : ABSOLUTE SUM OF A          ***
060051      !***          <--- R12          ***
060051      !*****
060051      !
060051      LST
060051 170 222      ABSUM2      CLB R70          !FLAG FOR THIS ROUTINE.
060053      !          UNL
060053 212          DCB R70          !SET IT.
060054 360 336      JMP SUM.+          !GO FIND SUM.
060056      !
060056      !*****
060056      !*** SUM (A) : THE SUM OF THE ELEMENTS IN A          ***
060056      !*** IN          : RELATIVE ADDRESS A          ***
060056      !***          <--- R12          ***
060056      !*** OUT          : SUM OF A          ***
060056      !***          <--- R12          ***
060056      !*****
060056      !
060056      LST
060056 170 222      SUM2      CLB R70          !FLAG SETTING FOR NOT ABSUM.
060060      !          UNL
060060 316 326 326      SUM.+      JSB =SUM.++          !GO FIND SUM.
060063      PACKIT      BSZ 0
060063 316 326 326      JSB =ROMJSB
060066 326 326          DEF RONF5-          !ROUND ANSWER
060070 260          BYT 260
060071 140 012 345      PUMD R40,+R12          !PUSH ANSWER ON STACK.
060074 236          RTN
060075      !
060075      SUM.++      BSZ 0
060075 316 326 326      JSB =LOCSZ2          !Ba, Ma, Na
060100 174 022 241      LDM R74,R22          !SAVE Ma & Na
060103 144 223          CLM R44          !INIT SUM=0.
060105 250 377          LDB R44,=377          !ASSUME INTEGER.
060107 176 221          TSM R76          !TEST Na (NUM COL A)
060111 367 336      SUMWHL      JZR SUMEND          !WHILE (NUM COL A >0)
060113 316 326 326      JSB =CONSUM          ! ADD UP ROWS
060116 176 213          DCM R76          ! DECR NUM COL A
060120 360 367          JMP SUMWHL          ! LOOP FOR TEST
060122      SUMEND      BSZ 0          !END WHILE
060122 236          RTN

```



```

060123      !
060123      !*****
060123      !*** MAT C = CSUM (A) : ***
060123      !***      ADDS THE VALUES OF THE ELEMENTS IN EACH ***
060123      !***      COLUMN OF THE OPERAND ARRAY , THEN ASSIGNS ***
060123      !***      THE SUM TO THE CORRESPONDING ELEMENT OF ***
060123      !***      THE RESULT ARRAY (VECTOR OR 1 ROW MATRIX) ***
060123      !*** IN      : RELATIVE ADDRESS C ***
060123      !***      RELATIVE ADDRESS A ***
060123      !***      <--- R12 ***
060123      !*** OUT      : STACK POPPED & ARRAY C ASSIGNED ***
060123      !***      (AND REDIMENSIONED IF NECESSARY) ***
060123      !*****
060123      !
060123      LST
060123 170 222      CSUM2      CLB R70      !FLAG SETTING FOR NOT ABSUM.
060125      !      UNL
060125 316 326 326      JSB =LOCSZ2      !Ba, Ma, Na
060130 165 012 343      POMD R65,-R12      !GET PTR TO TARGET ARRAY C.
060133 345      PUMD R65,+R12      !PTR ARRAY C BACK ON STACK
060134 345      PUMD R65,+R12      !ONE MORE COPY.
060135 174 022 241      LDM R74,R22      !SAVE Mc & Nc
060140 316 326 326      JSB =ROMJSB
060143 326 326      DEF LOCSZ-      !SEE IF C IS A VECTOR.
060145 260      BYT 260
060146 122      DRP R22      !ASSUME ROW AND ...
060147 024      ARP R24      !COL OF MATRIX
060150 371 336      JEZ CSEND1      !IF (C A VECTOR)
060152 124      DRP R24      !      ROW AND ...
060153 022      ARP R22      !      COL OF VECTOR
060154      CSEND1      BSZ 0      !END IF
060154 223      CLM R#
060155 211      ICM R#      !1 IN ROW OR COL
060156 176 243      STM R76,R#
060160 316 326 326      JSB =RDIM2      !Mc X 1-VEC; 1 X Nc-MATRIX
060163 124 076 241      LDM R24,R76      !GET Na (NUM COL A)
060166 367 336      JZR CSEND2      !IF (NOT NULL ARRAY)
060170 316 326 326      JSB =COLINT      !      INIT COL LOOP
060173      COLSUM      BSZ 0      !      REPEAT
060173 316 326 326      JSB =SUM&ST      !      SUM COLS & STORE VALUE
060176 316 326 326      JSB =COLNXT      !      POINT TO NEXT COL
060201 366 370      JNZ COLSUM      !      UNTIL (NUM COL A = 0)
060203      CSEND2      BSZ 0      !END IF
060203 236      RTN
060204      !
060204      !*****
060204      !*** MAT C = RSUM (A) : ***
060204      !***      ADDS THE VALUES OF THE ELEMENTS IN EACH ***
060204      !***      ROW OF THE OPERAND ARRAY, THEN ASSIGNS THE ***
060204      !***      SUM TO THE CORRESPONDING ELEMENT OF THE ***
060204      !***      RESULT ARRAY (VECTOR OR 1 ROW MATRIX) ***
060204      !*** IN      : RELATIVE ADDRESS C ***
060204      !***      RELATIVE ADDRESS A ***
060204      !***      <--- R12 ***
060204      !*** OUT      : STACK POPPED & ARRAY ASSIGNED ***
060204      !***      (RESULT ARRAY REDIMENSIONED IF NECESSARY) ***
060204      !*****
060204      !
060204      LST
060204 170 222      RSUM2      CLB R70      !FLAG SETTING FOR NOT ABSUM.
060206      !      UNL
060206 316 326 326      JSB =COMIN      !Ba, Ma, Na
060211 124 223      CLM R24
060213 211      ICM R24      !Ma X 1
060214 316 326 326      JSB =RDIM2      !REDIM C TO Ma X 1
060217 176 221      TSM R76      !TEST Ma (NUM ROW A)
060221 367 336      RSWHIL      JZR RSEND      !WHILE (NUM ROW A > 0)
060223 316 326 326      JSB =SUM&ST      !      DO SUM & STORE VALUE
060226 360 371      JMP RSWHIL      !      LOOP FOR TEST
060230      RSEND      BSZ 0      !END WHILE
060230 236      RTN

```

```

060231      !
060231      SUM&ST  BSZ 0
060231 316 326 326 JSB =CLRCOM      !INIT SUM = 0 & DO SUM
060234 316 326 326 JSB =ROMJSB
060237 326 326     DEF STOV--      !STORE RESULT & NXT TAR ADDR
060241 260         BYT 260
060242 176 213     DCM R76        !DECR NUM ROW
060244 236         RTN
060245      !
060245 144 223     CLRCOM  CLM R44      !INIT SUM=0.
060247 250 377     LDB R44,=377      !ASSUME INTEGER.
060251 114 074 241 COMSUM  LDM R14,R74 !INIT NUM ROW(COL) A
060254 367 336     COWHIL  JZR COEND1  !WHILE (NUM ROW(COL) A > 0)
060256 132 012 344     PUBD R32,+R12  !   SAVE RUN SUM SIGN
060261 140 345     PUMD R40,+R12      !   SAVE MANTISSA
060263 136 345     PUMD R36,+R12      !   SAVE EXP
060265 316 326 326 JSB =ROMJSB
060270 326 326     DEF GELT      !   GET A(I,J)
060272 260         BYT 260
060273 140 012 343     POMD R40,-R12   !   GET A(I,J) FROM STACK
060276 231         BCD
060277 170 310 001     CMB R70,=1      !   TEST NORM FLAG
060302 366 336     JNZ COELSE      !   IF (NORM)
060304 140 345     PUMD R40,+R12      !       A(I,J) ON STACK
060306 345         PUMD R40,+R12      !       DO IT AGAIN
060307 316 326 326 JSB =TWOSEP      !       SEPARATE BOTH COPIES
060312 316 326 326 JSB =ROMJSB
060315 326 326     DEF MPYR70      !       SQUARE A(I,J)
060317 260         BYT 260
060320 360 336     JMP COEND2
060322      COELSE  BSZ 0      !   ELSE - (NO NORM)
060322 144 310 377     CMB R44,=377    !       REAL OR INT?
060325 373 336     JCY COEND3      !       IF (NOT INT)
060327 316 326 326 JSB =SEP10      !       SEPERATE IT
060332      COEND3  BSZ 0      !       END IF
060332      COEND2  BSZ 0      !       END IF
060332 316 326 326 JSB =ROMJSB
060335 326 326     DEF RUNSUM      !   ADD TO RUNNING SUM
060337 260         BYT 260
060340 316 326 326 JSB =ROMJSB
060343 326 326     DEF NXTA      !   POINT TO NEXT A(I,J)
060345 260         BYT 260
060346 114 213     DCM R14      !   DECR NUM ROW(COL) A
060350 360 302     JMP COWHIL      !   LOOP FOR TEST
060352      COEND1  BSZ 0      !END WHILE
060352 236         RTN
060353      !
060353      COMIN  BSZ 0
060353 316 326 326 JSB =LOCSZ2      !Ba, Ma, Na
060356 136 215     TCM R36      !WANT 1 IF 0, 0 IF 1.
060360 211     ICM R36
060361 263 326 326 STMD R36,=TMP2    !SAVE OPTION BASE
060364 122 076 243 STM R22,R76      !SAVE Ma (NUM ROW A)
060367 124 074 243 STM R24,R74      !SAVE Na (NUM COL A)
060372 236         RTN
060373      !
060373      !*****
060373      !*** RNORM (A) : THE LARGEST SUM OF THE ABSOLUTE VALUES OF ***
060373      !***           THE ELEMENTS IN EACH ROW OF A           ***
060373      !*** IN           : RELATIVE ADDRESS A           ***
060373      !***           <--- R12           ***
060373      !*** OUT           : ROW NORM OF A           ***
060373      !***           <--- R12           ***
060373      !*****
060373      !
060373      LST
060373 316 326 326 RNORM2 JSB =NORMIN    !INIT Ba, Ma, Na
060376      !
060376      UNL
060376 367 336     RNWHIL  JZR RNEND    !WHILE (NUM ROW A > 0)
060400 316 326 326 JSB =MAXRC      !   FIND NORM(ROW(I))
060403 360 371     JMP RNWHIL      !   LOOP FOR TEST

```

```

060405          RNEND      BSZ 0          !END WHILE
060405 136 261 326      LDMD R36,=MBASE    !TEMP AREA BASE PTR.
060410 326
060411 100 325 326      SBMD R0,=TMP2      !ADJUST ROW# WHERE MAX IS.
060414 326
060415 036 267 071      STMD R0,X36,RNORMX  !STORE RNORMROW ANSWER.
060420 000
060421 236          RTN
060422          !
060422          !*****
060422          !*** CNORM (A) : THE LARGEST SUM OF THE ABSOLUTE VALUES OF ***
060422          !*** THE ELEMENTS IN EACH COLUMN OF A ***
060422          !*** IN : RELATIVE ADDRESS A ***
060422          !*** <--- R12 ***
060422          !*** OUT : COLUMN NORM OF A ***
060422          !*** <--- R12 ***
060422          !*****
060422          !
060422          LST
060422 316 326 326 CNORM2 JSB =NORMIN      !INIT Ba, Ma, Na
060425          ! UNL
060425 367 336          JZR CNEND          !IF (NOT NULL ARRAY)
060427 174 022 241      LDM R74,R22      ! SAVE Ma AND Na
060432 316 326 326      JSB =COLINT      ! INIT COL LOOP
060435          CNLOOP BSZ 0          ! REPEAT
060435 316 326 326      JSB =MAXRC      ! FIND NORM(COL(J))
060440 316 326 326      JSB =COLNXT      ! POINT NEXT COL
060443 366 370          JNZ CNLOOP      ! UNTIL (NUM COL A = 0)
060445 100 325 326      SBMD R0,=TMP2      ! COL# WHERE MAX IS
060450 326
060451          CNEND BSZ 0          !END IF
060451 147 260 326      LDBD R47,=TMP4      !GET VECTOR INDICATOR.
060454 326
060455 376 336          JRZ NOVEC          !IF (VECTOR)
060457 100 223          CLM R0
060461 213          DCM R0          ! 377,377
060462          NOVEC BSZ 0          !END IF
060462 136 261 326      LDMD R36,=MBASE    !TEMP AREA BASE PTR.
060465 326
060466 100 036 267      STMD R0,X36,CNORMX  !STORE CNORMCOL ANSWER.
060471 073 000
060473 236          RTN
060474          !
060474          !*****
060474          !** COLINT : SETS UP VARIABLES AT BEGINNING OF LOOP SO THAT THE ***
060474          !** INCREMENT MOVES DOWN A COLUMN RATHER THAN ACROSS A ***
060474          !** ROW. ***
060474          !** IN : INCRA = ELEMENT SIZE OF A : ROW INCREMENT ***
060474          !** TMP1 = Ba : BASE ADDRESS OF A ***
060474          !** OUT : R24 = 3Na, 4Na, 8Na : COLUMN INCREMENT OF A ***
060474          !** INCRA = 3Na, 4Na, 8Na : COLUMN INCREMENT OF A ***
060474          !** TMP3 = ELEMENT SIZE OF A (2BYTES) & Ba (3 BYTES) ***
060474          !*****
060474          !
060474          LST
060474          COLINT BSZ 0
060474          ! UNL
060474 316 326 326      JSB =ROMJSB
060477 326 326          DEF ACOLEL          !INT = 3Na; SHORT = 4Na; REAL = 8Na
060501 260          BYT 260
060502 163 261 326      LDMD R63,=INCRA      !GET ELE SIZE OF A
060505 326
060506 165 261 326      LDMD R65,=TMP1      !GET Ba
060511 326
060512 163 263 326      STMD R63,=TMP3      !SAVE SIZE AND Ba
060515 326
060516 124 263 326      STMD R24,=INCRA      !RESULT FROM ACOLEL
060521 326
060522 236          RTN
060523          !
060523          !*****

```

```

060523      !** COLNXT : SETS UP VARIABLES AT END OF LOOP SO THAT THE INCREMENT **
060523      !**      MOVES DOWN A COLUMN RATHER THAN ACROSS A ROW.      **
060523      !** IN  : TMP1 = CURRENT ADDRESS OF A COLUMN ELEMENT -- A(I,J) **
060523      !**      TMP3 = ELEMENT SIZE OF A (2 BYTES) & ADDR A(I,J)    **
060523      !** OUT : TMP1 = ADDRESS NEXT COLUMN ELEMENT OF A -- A(I+1,J) **
060523      !**      TMP3 = ELEMENT SIZE OF A (2 BYTES) & ADDR A(I+1,J)    **
060523      !*****
060523      !
060523      LST
060523      COLNXT  BSZ 0
060523      !      UNL
060523      163 261 326      LDMD R63,=TMP3      !GET ELE SIZE AND ADDR A(I,J)
060526 326
060527 155 063 241      LDM R55,R63      !MOVE ELE SIZE
060532 316 326 326      JSB =ROMJSB
060535 326 326      DEF NXTELE      !POINT TO NEXT ROW ELE
060537 260      BYT 260
060540 163 263 326      STMD R63,=TMP3      !SAVE ELE SIZE AND ADDR A(I,J)
060543 326
060544 165 263 326      STMD R65,=TMP1      !SAVE NEXT ADDR A(I,J)
060547 326
060550 176 221      TSM R76      !SEE IF MORE COLS.
060552 236      RTN
060553      !
060553      LST
060553 170 222      NORMIN  CLB R70      !READY TO FLAG.
060555      !      UNL
060555 212      DCB R70      !FLAG TO TAKE ABS VALUE OF A(I,J).
060556 316 353 140      JSB =COMIN      !DO COMMON INITIALIZATION.
060561 144 223      CLM R44      !ASSUME NORM=0.
060563 250 377      LDB R44,=377      !ASSUME INTEGER.
060565 140 012 345      PUMD R40,+R12      !PUSH INIT ANS ON STACK.
060570 110 223      CLM R10      !ASSUME ROW# OR COL# = 1.
060572 211      ICM R10
060573 000 243      STM R10,R0      !INITIALIZE ROW-COL POINTER
060575 231      ZTST-      BCD      !MODE FOR UPCOMING SHIFT
060576 130 200      ELB R30      !MOVE VEC-MAT FLAG INTO R30
060600 262 326 326      STBD R30,=TMP4      !SAVE IT
060603 230      BIN      !RESET MODE
060604      ZERTST  BSZ 0
060604 122 221      TSM R22      !M=0?
060606 367 336      JZR RNULL      !JIF YES
060610 124 221      TSM R24      !N=0?
060612 366 336      JNZ NRTN      !JIF NO - N<>0 AND M<>0
060614 100 223      RNULL  CLM R0
060616 236      NRTN  RTN
060617      !
060617      LST
060617 316 245 140 MAXRC JSB =CLRCOM      !FIND NORM OF CURRENT ROW (OR COL).
060622      !      UNL
060622 160 012 343      POMD R60,-R12      !GET LATEST ANSWER.
060625 345      PUMD R60,+R12      !REPLACE IT.
060626 345      PUMD R60,+R12      !COPY TO COMPARE WITH NEW NORM.
060627 170 006 345      PUMD R70,+R6      !PROTECT R70.
060632 126 344      PUBD R26,+R6      !PROTECT R26.
060634 222      CLB R26      !FLAG NEEDED IN COMRC-
060635 316 063 140      JSB =PACKIT      !PACK UP NEW NORM & PUT ON R12.
060640 140 070 243      STM R40,R70      !READY FOR COMRC-
060643 316 326 326      JSB =ROMJSB
060646 326 326      DEF COMRC-      !COMPARE OLD ANS WITH NEW NORM.
060650 260      BYT 260
060651 126 006 342      POBD R26,-R6      !RESTORE R26.
060654 170 343      POMD R70,-R6      !RESTORE R70.
060656 100 261 326      LDMD R0,=TMP2+      !GET ROW OR COL OF ANSWER
060661 326
060662 110 211      ICM R10      !POINT TO NEXT ROW (OR COL).
060664 176 213      DCM R76      !DECREMENT ROW (OR COL) COUNTER.
060666 236      RTN
060667      !
060667      !*****
060667      !*** MAT C = CROSS (B,A) :      ***

```

```

060667      !***          CALCULATES THE CROSS PRODUCT OF TWO 3 ELEMENT      ***
060667      !***          VECTORS (B & A), AND ASSIGNS RESULTING VECTOR      ***
060667      !***          TO C                                              ***
060667      !*** IN      : RELATIVE ADDRESS C                                ***
060667      !***          RELATIVE ADDRESS B                                ***
060667      !***          RELATIVE ADDRESS A                                ***
060667      !***          <--- R12                                          ***
060667      !*** OUT      : STACK POPPED & ARRAY ASSIGN                      ***
060667      !***          (REDIMENSIONED IF NECESSARY)                      ***
060667      !*****
060667      !
060667      LST
060667 316 326 326 CROSS2 JSB =COMX.          !DO INITIAL CHECK FOR VECTORS.
060672      ! UNL
060672 122 311 003 CMM R22,=3,0          !ARE INPUT VECTORS RIGHT SIZE?
060675 000
060676 367 336      JZR XEND0          !IF (NOT 3-DIMENSIONAL)
060700 316 326 326 JSB =ERROR+
060703 316 326 326 JSB =ERROR          ! ERROR 10 -- EXIT
060706 012      BYT 010D
060707 236      RTN
060710      XEND0      BSZ 0          !END IF
060710 316 326 326 JSB =RDIM2          !REDIM C TO 3 X 1
060713 370 336      JEN XEND1          !IF (C NOT A VECTOR)
060715 316 326 326 JSB =NOVECA          ! ERROR EXIT
060720      XEND1      BSZ 0          !END IF
060720 316 326 326 JSB =ROMJSB
060723 326 326      DEF C=AORB          !SEE IF TEMP ARRAY NEEDED
060725 260      BYT 260
060726 172 022 241 LDM R72,R22          !SAVE I AND J
060731 122 006 345 PUMD R22,+R6          !SAVE NUM ROWS C
060734      XLOOP      BSZ 0          !LOOP
060734 316 326 326 JSB =XINIT          ! INIT I,J,PTR TO A,B
060737 122 213      DCM R22          ! ADJUST SUBSCRIPT
060741 364 336      JNG XENDLP          ! ESCAPE LOOP AFTER 3
060743 074 243      STM R22,R74          ! SWAP I & J
060745 124 072 243 STM R24,R72
060750 144 223      CLM R44          ! INIT SUM=0
060752 250 377      LDB R44,=377          ! MAKE IT INTEGER
060754 316 326 326 JSB =TERMS          ! 1ST PROD IN SUM
060757 316 326 326 JSB =XINIT          ! INIT I,J,PTR TO A,B
060762 316 326 326 JSB =TERMS          ! VALUE C(I)
060765 316 326 326 JSB =ROMJSB
060770 326 326      DEF STOV--          ! STORE C(I)
060772 260      BYT 260
060773 360 337      JMP XLOOP          ! LOOP FOR TEST
060775      XENDLP      BSZ 0          !END LOOP
060775 122 006 343 POMD R22,-R6          !GET NUM ROWS
061000 316 326 326 JSB =ROMJSB
061003 326 326      DEF COPYAB          !COPY IF TEMP ARRAY USED
061005 260      BYT 260
061006 236      RTN
061007      !
061007      XINIT      BSZ 0
061007 172 022 243 STM R72,R22          !INIT I & J
061012 165 261 326 LDMD R65,=TMP1+
061015 326
061016 263 326 326 STMD R65,=TMP1          !INIT PTR TO A
061021 261 326 326 LDMD R65,=TMP2+
061024 263 326 326 STMD R65,=TMP2          !INIT PTR TO B
061027 236      RTN
061030      !
061030 230      TERMS      BIN          !SET MODE.
061031 124 213      TERMB      DCM R24          !DECREMENT ADDR OFFSET COUNT.
061033 364 336      JNG TERM2          !DO NEXT SUBSCRIPT WHEN NEG.
061035 316 326 326 JSB =ROMJSB
061040 326 326      DEF NXTB          !ADDRESS OFFSET-PT TO B(I).
061042 260      BYT 260
061043 360 364      JMP TERMB          !LOOP.
061045      TERMA      BSZ 0
061045 316 326 326 JSB =ROMJSB

```

061050	326	326		DEF NXTA	!ADDRESS OFFSET-PT TO A(J).
061052	260			BYT 260	
061053	122	213	TERM2	DCM R22	!DECREMENT ADDR OFFSET COUNT.
061055	365	366		JPS TERMA	!LOOP TILL COUNT NEG.
061057	114	223		CLM R14	!READY FOR DOT COUNT.
061061	211			ICM R14	!DOT COUNT =1 FOR 1 MULTIPLY.
061062	316	326	326	JSB =ROMJSB	
061065	326	326		DEF DOTACC	!GO CALC B(I)*A(J) OR B(J)*A(I).
061067	260			BYT 260	
061070	231			BCD	!MODE FOR NEXT COMMAND.
061071	132	216		NCB R32	!-BIAJ OR -(-BIAJ+BJAI).
061073	144	310	377	CMB R44,=377	!INTEGER?
061076	372	336		JNC XRTN	!JIF NO.
061100	145	215		TCM R45	!-BIAJ OR -(-BIAJ+BJAI).
061102	236		XRTN	RTN	!RETURN.
061103			!		

```

061103      !          HED DOT ROUTINE (C = B DOT A)
061103      !*****
061103      !*** DOT (B,A) : SUM OF PRODUCTS OF CORRESPONDING ELEMENTS ***
061103      !***          OF VECTORS B & A ***
061103      !*** IN          : RELATIVE ADDRESS B ***
061103      !***          RELATIVE ADDRESS A ***
061103      !***          <--- R12 ***
061103      !*** OUT          : DOT PRODUCT B & A ***
061103      !***          <--- R12 ***
061103      !*****
061103      !
061103      LST
061103 316 326 326 DOT2 JSB =COMX. !DO COMMON DOT, CROSS CALCS.
061106      ! UNL
061106 316 326 326 JSB =ROMJSB
061111 326 326 DEF DOTPRD !GO FIND DOT PRODUCT.
061113 260 BYT 260
061114 140 012 345 PSHRES PUMD R40,+R12 !PUSH RESULT ON STACK.
061117 236 RTN !END.
061120      !
061120      LST
061120      COMX. BSZ 0
061120      ! UNL
061120 316 326 326 JSB =LOCSZ2 !Ba, Ma, Na
061123 371 336 JEZ NOVECA !IF ('A' VECTOR)
061125 122 014 243 STM R22,R14 ! SAVE Ma
061130 316 326 326 JSB =ROMJSB
061133 326 326 DEF LOCSZI ! Bb, Mb, Nb
061135 260 BYT 260
061136 371 336 JEZ NOVECB ! IF('B' VECTOR)
061140 114 022 301 CMM R14,R22 ! COMPARE Mb & Ma
061143 367 336 JZR COMEND ! IF (Mb <> Ma)
061145 316 326 326 JSB =ERROR+ ! ROWS OF 'A' & 'B' ...
061150 316 326 326 JSB =ERROR ! DO NOT MATCH ...
061153 013 BYT 011D ! ERROR 11
061154 130 006 343 POMD R30,-R6 ! TRASH RETURN
061157      COMEND BSZ 0 ! ELSE - (Mb = Ma)
061157 236 RTN ! RETURN
061160      BSZ 0 ! END IF
061160      NOVECB BSZ 0 ! END IF
061160      NOVECA BSZ 0 !END IF
061160 316 326 326 JSB =ERROR+ !ERROR ROUTINE
061163 316 326 326 JSB =ERROR
061166 017 BYT 015D !NON-VECTOR ERROR CODE.
061167 136 006 343 POMD R36,-R6 !TRASH 1 RTN.
061172 236 RTN !DONE
061173      ! UNL
061173      !
061173      LOCSZ2 BSZ 0
061173 316 326 326 JSB =ROMJSB
061176 326 326 DEF LOCSZ !Ba, Ma, Na
061200 260 BYT 260
061201 236 RTN
061202      !
061202      LST
061202      RDIM2 BSZ 0
061202      ! UNL
061202 235 CLE
061203 233 DCE !FLAG TYPE C MATRIX
061204 165 012 343 POMD R65,-R12 !GET REL ADDR OF C
061207 316 326 326 JSB =ROMJSB
061212 326 326 DEF REDIM. !REDIMENSION C
061214 260 BYT 260
061215 316 326 326 JSB =ROMJSB
061220 326 326 DEF VECFLG !SET UP VECTOR-MATRIX FLAG
061222 260 BYT 260
061223 117 310 300 CMB R17,=300 !LOOK FOR REDIM ERROR
061226 372 336 JNC RDMRTN !IF (ERROR)
061230 100 006 343 POMD R0,-R6 ! TRASH 1 RETURN
061233      RDMRTN BSZ 0 !END IF
061233 236 RTN

```

061234  
061234

!  
!\*\*\*\*\* LINK NEXT SEGMENT \*\*\*\*\*



```

061234      !          HED MAT A = IDENTITY ROUTINE
061234      !*****
061234      !*** MAT C = IDN : ASSIGNS VALUE 1 TO ALL DIAGONAL ELEMENTS OF ***
061234      !***          RESULT MATRIX AND ASSIGNS THE VALUE 0 TO ALL ***
061234      !***          OTHER ELEMENTS. ***
061234      !***          RESULT MATRIX MUST BE SQUARE. ***
061234      !*** IN : REL ADDR C ***
061234      !***          <--- R12 ***
061234      !*** OUT : STACK POPPED AND ARRAY ASSIGNED AS IDENTITY MATRIX. ***
061234      !*****
061234      !
061234      LST
061234      IDN2      BSZ 0
061234      !          UNL
061234      117 311 300      CMM R17,=300          !LOOK AT ERROR FLAG
061237 372 336          JNC ICONT          !IF (REDIMENSION ERROR)
061241 165 012 343      POMD R65,-R12      ! TRASH REL ADDR ARRAY C
061244 236              RTN                ! EXIT
061245              ICONT      BSZ 0          !END IF
061245 316 326 326      JSB =ROMJSB
061250 326 326          DEF LOCSZ-          !Bc, Mc, Nc
061252 260              BYT 260
061253 122 024 301      CMM R22,R24          !NUM ROW = NUM COL ? (Mc=Nc)
061256 367 336          JZR ISQUAR          !IF (NOT SQUARE MATRIX)
061260 316 326 326      JSB =ERROR+
061263 316 326 326      JSB =ERROR
061266 016              BYT 014D          ! REPORT ERROR & EXIT
061267 236              RTN
061270              ISQUAR      BSZ 0          !END IF
061270 221              TSM R#              !NULL ARRAY ?
061271 367 336          JZR IDNRTN          !IF (NOT NULL ARRAY)
061273              ICOLLP      BSZ 0          ! REPEAT
061273 122 026 243      STM R22,R26          ! ROWS = Mc
061276              IROWLP      BSZ 0          ! REPEAT
061276 140 223          CLM R40              ! ASSUME NON DIAGONAL
061300 126 024 301      CMM R26,R24          ! NUMROW = NUMCOL ?
061303 366 336          JNZ ISTOP          ! IF (NUMROW = NUMCOL)
061305 316 326 326      JSB =ROMJSB
061310 326 326          DEF FTR61          ! DIAGONAL - 1 IN R40
061312 260              BYT 260
061313              ISTOP      BSZ 0          ! END IF
061313 316 326 326      JSB =ROMJSB
061316 326 326          DEF STOV          ! 1 OR 0 IN C(I,J)
061320 260              BYT 260
061321 126 213          DCM R26              ! ROWS = ROWS - 1
061323 366 351          JNZ IROWLP          ! UNTIL (ROWS = 0)
061325 124 213          DCM R24              ! COLS = COLS - 1
061327 366 342          JNZ ICOLLP          ! UNTIL (COLS = 0)
061331 316 326 326      JSB =ROMJSB
061334 326 326          DEF CKTRC          ! CHECK TRACE
061336 260              BYT 260
061337 236              IDNRTN      RTN          !END IF
061340      !
061340      !*****
061340      !*** EQUA2 : COPIES AN ARRAY -- A -- INTO ARRAY -- C. ***
061340      !*** IN : R24 = Ma : NUM ROWS OF A ***
061340      !***          R24 = Na : NUM COLS OF A ***
061340      !***          R32 = 1 : CONVERT TAGGED INTEGER OF A INTO REALS ***
061340      !***          = 0 : DON'T CONVERT TAGGED INTEGER OF A ***
061340      !***          TMP1 = Ba : BASE ADDRESS OF A ***
061340      !***          TMP4 = BC : BASE ADDRESS OF C ***
061340      !*** OUT : VALUES OF A COPIED INTO C ***
061340      !*****
061340      !
061340      LST
061340      EQUA2      BSZ 0
061340 132 222          CLB R32
061342 210          ICB R32          !FLAG CONVERT TAGGED INT TO REAL
061343              EQUA09      BSZ 0          !ENTRY POINT FOR NO CONVERSION
061343      !          UNL
061343 316 326 326      JSB =ROMJSB

```

```

061346 326 326          DEF PUTREG          !SAVE REGISTERS
061350 260              BYT 260
061351 316 326 326      JSB =ROMJSB
061354 326 326          DEF MNMUL          !FIND Ma * Na
061356 260              BYT 260
061357 130 055 241      LDM R30,R55        !COUNT = Ma * Na
061362                EQULOOP BSZ 0        !REPEAT
061362 165 261 326      LDMD R65,=TMP1      ! GET FETCH ADDR FROM TMP1
061365 326
061366 263 326 326      STMD R65,=PTR2      ! POINT TO VALUE WITH PTR2
061371 146 260 326      LDBD R46,=TYPA      ! GET TYPE OF MATRIX A
061374 326
061375 316 326 326      JSB =ROMJSB
061400 326 326          DEF FETCH-          ! R40 VALUE OF A(I,J)
061402 260              BYT 260
061403 132 220          TSB R32            ! TEST CONVERT FLAG
061405 367 336          JZR EEND1          ! IF (CONVERT)
061407 144 310 377      CMB R44,=377      ! INTEGER ?
061412 372 336          JNC EEND2          ! IF (INTEGER)
061414 140 060 243      STM R40,R60        ! MOVE VALUE
061417 316 326 326      JSB =INTORL        ! CONVERT IT
061422 160 040 243      STM R60,R40        ! MOVE IT BACK
061425                EEND2 BSZ 0          ! END IF
061425                EEND1 BSZ 0          ! END IF
061425 316 326 326      JSB =ROMJSB
061430 326 326          DEF STOV            ! STORE IN C(I,J)
061432 260              BYT 260
061433 316 326 326      JSB =ROMJSB
061436 326 326          DEF NXTA          ! ADDR NEXT A(I,J)
061440 260              BYT 260
061441 130 213          DCM R30            ! COUNT = COUNT - 1
061443 366 315          JNZ EQULOOP        !UNTIL (COUNT = 0)
061445 316 326 326      JSB =ROMJSB
061450 326 326          DEF GETREG          !RESTORE REGISTERS
061452 260              BYT 260
061453 236              RTN
061454                !
061454                !***** LINK NEXT SEGMENT *****

```

```

061454      !
061454      ! *****
061454      ! *** FIND# / : FINDS # AND TRAILING / IN FORMAT STRING ***
061454      ! *** EXIT   : TMP2+ = 0 --> NO # IN FORMAT STRING ***
061454      ! ***          TMP2+ <> 0 --> # PRESENT IN FORMAT STRING ***
061454      ! ***          TMP4  = 0 --> NO FINAL / IN FORMAT STRING ***
061454      ! ***          TMP4  <> 0 --> FINAL / IN FORMAT STRING ***
061454      ! *****
061454      !
061454      !           LST
061454      FIND# /  BSZ 0
061454      !           UNL
061454      !
061454      ! *****
061454      ! *** LOOK FOR MAT PRINT ***
061454      ! *****
061454      !
061454 316 326 326      JSB =SVPTRS          !SAVE PTR1
061457 120 222      CLB R20
061461 262 326 326      STBD R20,=TMP2+      !FLAG NO CR,LF SUPPRESS
061464 262 326 326      STBD R20,=TMP4      !FLAG NO TRAILING SLASH
061467      LOP45      BSZ 0          !REPEAT
061467      LOP45A      BSZ 0          ! LOOP
061467 270 326 326      LDBI R#,=PTR1+      ! GET TOKEN
061472 310 153      CMB R#,,=PRTOK      ! COMPARE WITH PRINT TOKEN
061474 367 336      JZR LOP45B      ! ESCAPE IF PRINT TOKEN
061476 310 126      CMB R#,,=DSTOK      ! COMPARE WITH DISPLAY TOKEN
061500 367 336      JZR LOP45B      ! ESCAPE IF DISPLAY TOKEN
061502 360 363      JMP LOP45A      ! LOOP BACK
061504      LOP45B      BSZ 0          ! END LOOP
061504 165 271 326      LDMI R65,=PTR1      ! GET NEXT 3 TOKENS
061507 326
061510 311 045 260      CMM R65,=045,260,370 ! COMPARE WITH MATRIX ROM MAT TOKEN
061513 370
061514 366 351      JNZ LOP45          !UNTIL (FOUND MATRIX ROM MAT STATEMENT)
061516      !
061516      ! *****
061516      ! *** IF FORMAT STRING, SEARCH FOR ***
061516      ! *** CR, LF SUPPRESS (#) ***
061516      ! *****
061516      !
061516 120 271 326      LDMI R20,=PTR1-      !GET POSSIBLE QUOTE OR USING LINE NUMBER
061521 326
061522 310 005      CMB R20,=QUOTE      !COMPARE WITH QUOTE TOKEN
061524 367 336      JZR FIND#          !JIF TOKEN = QUOTE
061526 310 347      CMB R20,=USLNO      !COMPARE WITH LINE NUMBER TOKEN
061530 366 336      JNZ GOPRT          !IF (TOKEN = LINE NUMBER OR QUOTE)
061532      BSZ 0          ! IF (TOKEN = LINE NUMBER)
061532 316 326 326      JSB =LINEAL      ! ALLOCATE LINE NUMBER
061535 175 323 326      ADMD R75,=FWCURR      ! FIND ABS ADDR LINE NUMBER
061540 326
061541 263 326 326      STMD R75,=PTR1      ! POINT TO IMAGE STATEMENT
061544 162 271 326      LDMI R62,=PTR1-      ! POINT TO UNQUOTED STRING
061547 326
061550      FIND#      BSZ 0          ! END IF
061550 120 270 326      LDBI R20,=PTR1-      ! GET FORMAT STRING LENGTH
061553 326
061554      LOP#      BSZ 0          ! REPEAT
061554 121 270 326      LDBI R21,=PTR1-      ! GET STRING CHARACTER
061557 326
061560 310 043      CMB R21,=43          ! COMPARE WITH #
061562 366 336      JNZ LOP#1          ! IF (CHAR = #)
061564 262 326 326      STBD R21,=TMP2+      ! FLAG CR,LF SUPPRESS
061567      LOP#1      BSZ 0          ! END IF
061567 120 212      DCB R20          ! STRING LENGTH = STRING LENGTH - 1
061571 366 361      JNZ LOP#          ! UNTIL (STING LENGTH = 0)
061573      !
061573      ! *****
061573      ! *** SEARCH FORMAT STRING FOR ***
061573      ! *** TRAILING SLASH ***
061573      ! *****

```

```

061573      !
061573      LOP/      BSZ 0      !      LOOP
061573 121 310 130      CMB R21,=X      !      COMPARE WITH X
061576 367 336      JZR LOP/NX      !      CONTINUE SCAN IF X
061600 310 042      CMB R21,=QUOT      !      COMPARE WITH "
061602 367 336      JZR LOP/NX      !      CONTINUE SCAN IF "
061604 310 051      CMB R21,=RTPARN      !      COMPARE WITH )
061606 367 336      JZR LOP/NX      !      CONTINUE SCAN IF )
061610 310 050      CMB R21,=LTPARN      !      COMPARE WITH (
061612 367 336      JZR LOP/NX      !      CONTINUE SCAN IF (
061614 310 054      CMB R21,=COMMA      !      COMPARE WITH ,
061616 367 336      JZR LOP/NX      !      CONTINUE SCAN IF ,
061620 310 000      CMB R21,=BLANK      !      COMPARE WITH BLANK
061622 367 336      JZR LOP/NX      !      CONTINUE SCAN IF BLANK
061624 360 336      JMP LOP/1      !      ESCAPE IF NON-GARBAGE
061626 270 326 326 LOP/NX      LDBI R#,,=PTR1+      !      GET NEXT IMAGE CHAR
061631 360 340      JMP LOP/      !      END LOOP
061633      LOP/1      BSZ 0
061633 310 057      CMB R#,,=SLASH      !      COMPARE WITH SLASH
061635 366 336      JNZ LOP/2      !      IF (TRAILING SLASH)
061637 262 326 326      STBD R#,,=TMP4      !      FLAG TRAILING SLASH
061642      LOP/2      BSZ 0      !      END IF
061642      GOPRT      BSZ 0      !END IF
061642 316 326 326      JSB =RSPTRS      !RESTORE PTR1
061645 236      RTN
061646      !
061646      !      *****
061646      !      *** LINK NEXT SEGMENT      ***
061646      !      *****

```

```

061646      !          HED MAT C = TRANSPOSE(A) ROUTINE
061646      !*****
061646      !*** MAT C = TRN (A) :          ***
061646      !***      RESULT ARRAY WILL CONTAIN THE SAME ELEMENTS      ***
061646      !***      AS THE OPERAND ARRAY, BUT THE ROWS AND COLUMNS  ***
061646      !***      WILL BE INTERCHANGED.          ***
061646      !*** IN      : RELATIVE ADDRESS C          ***
061646      !***      RELATIVE ADDRESS A          ***
061646      !***      <--- R12          ***
061646      !*** OUT      : STACK POPPED & ARRAY ASSIGNED          ***
061646      !***      (AND REDIMENSIONED IF NECESSARY)          ***
061646      !*****
061646      !
061646      LST
061646      TRNPS2  BSZ 0
061646      !      UNL
061646      !      -----
061646      !      -- GET A INFO & --
061646      !      -- REDIMENSION C --
061646      !      -----
061646  316 173 142      JSB =LOCSZ2          !Ba, Ma, Na
061651  170 022 241      LDM R70,R22          !SAVE Ma & Na
061654  124 014 243      STM R24,R14          !COPY Na
061657  022 241      LDM R24,R22          !Mc = Na
061661  114 243      STM R14,R22          !Nc = Ma
061663  316 202 142      JSB =RDIM2          !REDIM C -- Na X Ma
061666  316 204 141      JSB =ZERTST          !SEE IF NULL ARRAY RESULTS
061671  366 336      JNZ TRNPOS          !IF (NULL ARRAY)
061673  236      RTN          ! QUIT
061674      TRNPOS  BSZ 0          !END IF
061674      !      -----
061674      !      -- COPY A INTO C --
061674      !      -----
061674  316 340 142      JSB =EQUA2          !COPY A INTO C
061677  316 326 326      JSB =ROMJSB
061702  326 326      DEF MNMUL          !FIND MN = Mc * Nc
061704  260      BYT 260
061705  110 055 241      LDM R10,R55          !MN
061710  014 243      STM R10,R14
061712  114 213      DCM R14          !MN - 1
061714  120 261 326      LDMD R20,=INCR          !GET ELE SIZE OF C
061717  326
061720  100 251 070      LDM R0,=70,0          !INIT R0
061723  000
061724  020 304      SBB R0,R20          !60 (REAL), 64 (SHORT), 65 (INTEGER)
061726      !      -----
061726      !      -- TAG BYTE & INCR --
061726      !      -----
061726      BSZ 0          !ASSUME INTEGER OR SHORT ...
061726  126 223      CLM R26          ! TAG INCR = 0
061730  174 250 100      LDB R74,=100          ! TAG = 2nd M.S.BIT OF SIGN
061733  120 310 010      CMB R20,=10          !TEST TYPE OF ARRAY C
061736  366 336      JNZ TRN09          !IF (REAL)
061740  126 250 006      LDB R26,=6          ! TAG INCR = 6
061743  174 250 004      LDB R74,=4          ! TAG = 2nd M.S.BIT OF SIGN
061746      TRN09  BSZ 0          !END IF
061746      !      -----
061746      !      -- IN PLACE CYCLING --
061746      !      -----
061746      CYCLE1  BSZ 0          !REPEAT
061746  316 326 326      JSB =TRNUT2          ! FIND ADDR Pth ELE (R45)
061751  065 243      STM R#,R65          ! SAVE IT
061753  130 222      CLB R30
061755  145 026 305      SBM R45,R26          ! ADDR TAG BYTE Pth ELE
061760  263 326 326      STMD R45,=PTR2          ! STORE ADDR IN PTR2
061763  135 270 326      LDBI R35,=PTR2-          ! GET TAG BYTE
061766  326
061767  034 242      STB R35,R34          ! SAVE IT
061771  074 307      ANM R35,R74          ! Pth ELE TAGGED OR UNTAGGED ?
061773  366 336      JNZ CYCLE4          ! IF (Pth ELE UNTAGGED)
061775  240      LDB R35,R74          ! GET TAG

```

```

061776 034 302          ADB R35,R34          !      TAG THE BYTE
062000 272 326 326      STBI R35,=PTR2      !      STORE TAGGED BYTE
062003 145 065 241      LDM R45,R65         !      RESTORE ADDR Pth ELE
062006 316 326 326      JSB =LDR*          !      Y = [P]
062011          CYCLE2  BSZ 0                !      LOOP
062011 101 012 345      PUMD R*,+R12        !      SAVE Y
062014 316 326 326      JSB =TRNUT1        !      GET NEW P
062017 316 326 326      JSB =TRNUT2        !      GET IT'S ADDR
062022 316 326 326      JSB =LDR*          !      GET [P]
062025 101 006 345      PUMD R*,+R6         !      SAVE [P]
062030 012 343          POMD R*,-R12        !      GET Y
062032 316 326 326      JSB =STR*          !      [P] = Y
062035 101 006 343      POMD R*,-R6         !      Y = [P]
062040 120 310 010      CMB R20,=10         !      TEST TYPE
062043 367 336          JZR TRN08           !      IF (INTEGER OR SHORT)
062045 100 020 302      ADB R0,R20          !      ADD IN ELE SIZE
062050 212              DCB R0              !      R0 = 67
062051 360 336          JMP TRN04
062053          TRN08  BSZ 0                !      ELSE - (REAL)
062053 100 210          ICB R0              !      R0 = 61
062055          TRN04  BSZ 0                !      END IF
062055 135 074 240      LDB R35,R74         !      GET TAG
062060 001 307          ANM R35,R*         !      IS Y TAGGED ?
062062 366 336          JNZ CYCLE3         !      ESCAPE LOOP IF Y TAGGED
062064 101 074 302      ADB R*,R74         !      TAG Y
062067 316 326 326      JSB =RSTR0         !      RESTORE R0
062072 360 315          JMP CYCLE2         !      LOOP
062074          CYCLE3 BSZ 0                !      END LOOP
062074 316 326 326      JSB =RSTR0         !      RESTORE R0
062077          CYCLE4 BSZ 0                !      END IF
062077 110 213          DCM R10             !      P = P - 1
062101 366 243          JNZ CYCLE1         !UNTIL (P = 0)
062103          !      -----
062103          !      -- UNTAG SIGN BYTES --
062103          !      -----
062103 114 211          ICM R14             !MN
062105 174 216          NCB R74             !UNARY NOT THE TAG
062107 130 222          CLB R30
062111 145 261 326      LDMD R45,=TMP3++    !GET Bc
062114 326
062115 026 305          SBM R45,R26         !ADDR TAG BYTE
062117          UNTAG BSZ 0                !REPEAT
062117 145 263 326      STMD R45,=PTR2      !  POINT TO TAG BYTE
062122 326
062123 135 270 326      LDBI R35,=PTR2-     !  GET TAG BYTE
062126 326
062127 074 307          ANM R35,R74         !  CLEAR TAG BYTE
062131 272 326 326      STBI R35,=PTR2     !  STORE CLEARED TAG BYTE
062134 155 261 326      LDMD R55,=INCRc    !  GET ELE SIZE OF C
062137 326
062140 157 222          CLB R57
062142 145 055 305      SBM R45,R55         !  GET NEXT TAG BYTE ADDR
062145 114 213          DCM R14             !  MN = MN - 1
062147 366 346          JNZ UNTAG          !UNTIL (MN = 0)
062151 236              RTN
062152          !
062152          !*****
062152          !* TRNUT1: THE Pth ELE OF AN M X N MATRIX A IS MAPPED BY THE      *
062152          !*      TRANSPOSE TO ELE M(P-1)+1-(MN-1)INT((P-1)/N) OF TRN(A) *
062152          !* ENTRY  : R10/11 = P; R70/71 = M; R72/73 = N; R14/15 = MN-1      *
062152          !* EXIT   : R10/11 = M(P-1)+1-(MN-1)INT((P-1)/N)                *
062152          !*****
062152          !
062152          TRNUT1  BSZ 0
062152 110 213          DCM R10             !P - 1
062154 066 243          STM R10,R66         !MOVE IT
062156 176 070 241      LDM R76,R70         !MOVE M
062161 316 326 326      JSB =INTMUL        !M(P-1)
062164 154 211          ICM R54            !M(P-1)+1
062166 044 243          STM R54,R44         !SAVE IT
062170 166 223          CLM R66

```

```

062172 213          DCM R66          !INIT COUNTER
062173          TRN07  BSZ 0          !REPEAT
062173 166 211          ICM R66          ! BUMP CURRENT INT((P-1)/N)
062175 110 072 305      SBM R10,R72      ! SUBTRACT OUT N
062200 373 371          JCY TRN07        !UNTIL (ANSWER IS NEGATIVE)
062202 114 076 243      STM R14,R76      !MN -1
062205 316 326 326      JSB =INTMUL      ! (MN-1) INT((P-1)/N)
062210 144 054 305      SBM R44,R54      !EXIT VALUE
062213 110 044 241      LDM R10,R44
062216 236          RTN
062217          !
062217          !*****
062217          !* TRNUT2: COMPUTES ADDR OF Pth ELE OF MATRIX A          *
062217          !* ENTRY : R10/11 = P; R20,21 = 8,4 OR 3; TMP3++ = Bc          *
062217          !* EXIT : R45/46/47 = ADDR Pth ELE          *
062217          !*****
062217          !
062217          TRNUT2  BSZ 0
062217 132 010 241      LDM R32,R10          !P
062222 213          DCM R32          !P - 1
062223 034 243      STM R32,R34          !COPY P - 1
062225 205          LLM R32          !2(P-1)
062226 205          LLM R32          !4(P-1)
062227 120 310 004      CMB R20,=4          !TEST TYPE OF ARRAY C
062232 373 336      JCY TRN06          !IF (INTEGER)
062234 132 305      SBM R32,R34          ! 3(P-1)
062236 360 336      JMP TRN05
062240          TRN06  BSZ 0
062240 367 336      JZR TRN05          !ELSE IF (REAL)
062242 132 205      LLM R32          ! 8(P-1)
062244          TRN05  BSZ 0          !END IF
062244 134 222      CLB R34
062246 145 261 326      LDMD R45,=TMP3++          !GET Bc
062251 326
062252 032 305      SBM R45,R32          !ADDR Pth ELE
062254 236          RTN
062255          !
062255          !*****
062255          !* LDR* : LOADS R[R0] WITH VALUE ADDRESSED BY PTR2          *
062255          !* ENTRY: R0 = 60 (REAL), 64 (SHORT), 65 (INT); R45 = ADDR OF ELE          *
062255          !* EXIT : R60 OR 64 OR 65 LOADED WITH APPROPRIATE          *
062255          !* REAL ,SHORT OR INTEGER VALUE          *
062255          !*****
062255          !
062255          LDR*  BSZ 0
062255 145 263 326      STMD R45,=PTR2          !POINT TO ARRAY VALUE
062260 326
062261 100 310 064      CMB R0,=64          !TEST TYPE
062264 373 336      JCY LDR*1          !IF (REAL)
062266 160 271 326      LDMI R60,=PTR2-          ! LOAD 8 BYTES
062271 326
062272 236          RTN
062273 367 336      LDR*1  JZR LDR*2          !ELSE IF (INTEGER)
062275 165 271 326      LDMI R65,=PTR2-          ! LOAD 3 BYTES
062300 326
062301 236          RTN
062302          LDR*2  BSZ 0          !ELSE (SHORT)
062302 164 271 326      LDMI R64,=PTR2-          ! LOAD 4 BYTES
062305 326
062306          BSZ 0          !END IF
062306 236          RTN
062307          !
062307          !*****
062307          !* STR* : STORES R[R0] INTO ARRAY ELE POINTED TO BY PTR2          *
062307          !* ENTRY: R0 = 60 (REAL), 64 (SHORT), 65 (INT); PTR2 = ADDR OF ELE          *
062307          !* EXIT : [PTR2] STORED WITH VALUE CONTAINED IN R60 OR 64 OR 65          *
062307          !*****
062307          !
062307          STR*  BSZ 0
062307 100 310 064      CMB R0,=64          !TEST TYPE
062312 373 336      JCY STR*1          !IF (REAL)

```

```

062314 160 273 326      STMI R60,=PTR2      !   STORE 8 BYTES
062317 326
062320 236              RTN
062321 367 336      STR*1  JZR STR*2          !ELSE IF (INTEGER)
062323 165 273 326      STMI R65,=PTR2      !   STORE 3 BYTES
062326 326
062327 236              RTN
062330              STR*2  BSZ 0              !ELSE (SHORT)
062330 164 273 326      STMI R64,=PTR2      !   STORE 4 BYTES
062333 326
062334              BSZ 0              !END IF
062334 236              RTN
062335      !
062335      !*****
062335      !* RSTR0: RESTORES R0 TO 64 IF SHORT, 65 IF INTEGER      *
062335      !* ENTRY: R0 = 67 (SHORT & INTEGER), 61 (REAL)          *
062335      !*          R20 = 10 (REAL), 4 (SHORT), 3 (INTEGER)      *
062335      !* EXIT : R0 = 60 (REAL), 64 (SHORT), 65 (INTEGER)      *
062335      !*****
062335      !
062335      RSTR0      BSZ 0
062335 120 310 010      CMB R20,=10          !TEST TYPE
062340 367 336      JZR RSTR01          !IF (INTEGER OR SHORT)
062342 100 020 304      SBB R0,R20          !   SUBTRACT ELE SIZE
062345 210          ICB R0              !   R0 = 65 IF INTEGER
062346 360 336      JMP RSTR02          !   R0 = 64 IF SHORT
062350      RSTR01  BSZ 0              !ELSE - (REAL)
062350 100 212      DCB R0              !   R0 = 60
062352      RSTR02  BSZ 0              !END IF
062352 236          RTN
062353      !

```



```

062353      !          HED LBND AND UBND ROUTINES
062353      !****  LBND ATTRIBUTES TABLE  *****
062353 044 055      BYT 44,55
062355      !*****
062355      !*** LBND (A,EXP) : LOWER BOUND OF ARRAY SUBSCRIPT (1 OR 2) ***
062355      !***                      OF ARRAY A, SPECIFIED BY ROUNDED INTEGER ***
062355      !***                      VALUE OF EXPRESSION. EQUAL TO THE ***
062355      !***                      OPTION BASE IN EFFECT. ***
062355      !*** IN              : RELATIVE ADDRESS A ***
062355      !***                      INTEGER VALUE OF EXPRESSION ***
062355      !***                      <--- R12 ***
062355      !*** OUT            : LOWER BOUND OF ARRAY A ***
062355      !***                      <--- R12 ***
062355      !*****
062355      !
062355      LST
062355 170 222      LDIM2      CLB R70          !CLEAR FLAG.
062357      !          UNL
062357 360 336      JMP UDIM+          !PUSH OPTION BASE ON STACK.
062361      !
062361      !****  UBND ATTRIBUTES TABLE  *****
062361 044 055      BYT 44,55          !ONE ARRAY, ONE NUMERIC
062363      !*****
062363      !*** UBND (A,EXP) : UPPER BOUND OF ARRAY SUBSCRIPT (1 OR 2) ***
062363      !***                      OF ARRAY A, SPECIFIED BY ROUNDED INTEGER ***
062363      !***                      VALUE OF EXPRESSION. ***
062363      !*** IN              : RELATIVE ADDRESS A ***
062363      !***                      INTEGER VALUE OF EXPRESSION ***
062363      !***                      <--- R12 ***
062363      !*** OUT            : UPPER BOUND OF ARRAY A ***
062363      !***                      <--- R12 ***
062363      !*****
062363      !
062363      LST
062363 170 222      UDIM2      CLB R70          !CLEAR FLAG .
062365      !          UNL
062365 210          ICB R70          !FLAG SAYING CALC UDIM.
062366 316 326 326 UDIM+      JSB =ONEB      !GET INTEGER N.
062371 146 006 345      PUMD R46,+R6      !SAVE ANSWER FROM LOCSZ
062374 316 173 142      JSB =LOCSZ2      !GET UPPER DIMS & OPTION BASE.
062377 146 006 343      POMD R46,-R6      !RESTORE ANSWER
062402 170 220          TSB R70          !UDIM OR LDIM?
062404 366 336          JNZ NOTLDM        !JIF NOT LDIM.
062406 122 223          CLM R22          !1 IN R22-ADDS TO OPT BASE.
062410 211          ICM R22          !THIS 1 GETS DECR OUT AT UDIM17.
062411 024 243          STM R22,R24      !SAME FOR R24.
062413 146 221          NOTLDM      TSM R46      !N=1 OR 2?
062415 364 336          JNG UDIM22      !JIF NEG; INPUT ERROR.
062417 213          DCM R#          !N=1?
062420 366 336          JNZ UDIM16      !JIF N#1.
062422 136 022 303      ADM R36,R22      !ELSE ADD COL SIZE TO OPT BASE.
062425 360 336          JMP UDIM17      !GO DECR & PUSH ANS ON STACK.
062427 213          UDIM16      DCM R#          !N=2?
062430 366 336          JNZ UDIM22      !JIF N#2; ERROR.
062432 370 336          JEN UDIM22      !ERROR IF A VECTOR.
062434 136 024 303      ADM R36,R24      !ELSE ADD ROW SIZE TO OPT BASE.
062437 213          UDIM17      DCM R#          !ADJUST ANSWER.
062440 316 326 326 UDIM20      JSB =CONBIN      !BINARY INT TO BCD INT.
062443 140 012 345      PUMD R40,+R12      !PUSH ANSWER ONTO STACK.
062446 236          RTN
062447      UDIM22      BSZ 0          !ERROR EXIT
062447 316 326 326      JSB =ERROR
062452 131          BYT 89D
062453      !
062453      !*****
062453      !*** ERROR+ : SETS ERRROM WITH ROM NUMBER IF NO OTHER ***
062453      !***                      ERRORS ARE FLAGGED. ***
062453      !*****
062453      !
062453      ERROR+      BSZ 0
062453 136 222      CLB R36

```

```

062455 320 326 326          CMBD R36,=ERRORS          !ERROR FLAG ALREADY SET
062460 366 336              JNZ DOERR+                !IF (NO OTHER ERRORS FLAGGED)
062462 250 260              LDB R36,=260              !  SELECT MATRIX ROM
062464 262 326 326          STBD R36,=ERRROM          !   FLAG ERROR
062467                DOERR+  BSZ 0                    !END IF
062467 236                  RTN
062470                !
062470                !*****
062470                !***   EXTERNAL LABEL TABLE  ***
062470                !*****
062470                !
062470                LST
062470      TMP1+      DAD 104465          !3 BYTES PAST TMP1
062470      TMP2+      DAD 104475          !3 BYTES PAST TMP2
062470      TMP3++     DAD 104507          !5 BYTES PAST TMP3
062470      FTR61      DAD 55514
062470      LINEAL      DAD 43420          !ALLOCATE LINE NUMBER
062470      !          -----
062470      !          ---  ROM1  ---
062470      !          -----
062470      CKTRC        DAD 66312          !CHECK TRACE
062470      MPYR70        DAD 70141          !MULTIPLICATION
062470      NXTB          DAD 70341          !NEXT ELEMENT OF ARRAY 'B'
062470      NXTA          DAD 70361          !NEXT ELEMENT OF ARRAY 'A'
062470      NXTELE        DAD 70401          !NEXT ELEMENT OF ARRAY
062470      DOTPRD        DAD 70715          !DOT PRODUCT
062470      RONF5-        DAD 70724          !ROUND ANSWER
062470      DOTACC        DAD 70755          !ACCUMULATE DOT PRODUCT
062470      RUNSUM        DAD 71040          !RUNNING SUM
062470      STOV--        DAD 71276          !STORE ELEMENT
062470      STOV          DAD 71301          !STORE ELEMENT
062470      FETCH-        DAD 71426          !FETCH ELEMENT
062470      C=AORB         DAD 73704          !TEST BASE ADDR -- A,B,C
062470      COPYAB        DAD 74007          !COPY ARRAY A OR B
062470      ACOLEL        DAD 74113          !NEXT COL ELE OF 'A' TYPE ARRAY
062470      MNMUL          DAD 74365          !MULTILICATION
062470      LOCSZ-        DAD 74406          !LOCATE 'C' TYPE MATRIX
062470      VECFLG        DAD 74413          !SET VECTOR MATRIX FLAG
062470      LOCSZI        DAD 74426          !LOCATE 'B' TYPE MATRIX
062470      LOCSZ          DAD 74432          !LOCATE 'A' TYPE MATRIX
062470      COMRC-        DAD 74702          !
062470      GELT          DAD 74745          !GET ELEMENT
062470      REDIM.        DAD 77023          !REDIMENSION
062470      PUTREG         DAD 77520          !SAVE REGISTERS
062470      GETREG        DAD 77553          !GET REGISTERS

```