

HP-41 EXTENDED FUNC. REV. C

HP VASH Listings

Contained in this package are source listings for proprietary Hewlett-Packard products. This information has been released for distribution on the basis of responsible dissemination by PPC. In order to ensure the continuation of our good-faith arrangement for distribution of this type of documentation it is vital that the user understands that no manufacturer support will be provided for use of these documents.

By opening this package, the user accepts that this information is supplied as is, and understands that all documentation is strictly Not Manufacturer Supported (NOMAS). Hewlett-Packard bears no responsibility for support or correctness of this documentation, and use of the information contained within is at the users risk.

Any questions regarding the enclosed material may be addressed to:

PPC
P.O. Box 9599
Fountain Valley, CA 92728-9599 USA
(714) 754-6226

Please respect the responsibility that acceptance of this information entails. Only through the actions of each of our members can PPC continue to support our members by providing such services as NOMAS listings.

NOMAS

VASM ROM ASSEMBLY

REV. 6/81A

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

OPTIONS: L C S

2		FILE	SCAPIB	
3	0	31	CON	25
4	1	60	CON	48
5	2	0	DEFR4K	APHEAD
5	3	0		
6	4	0	DEFR4K	ALENG
6	5	0		
7	6	0	DEFR4K	ANUM
7	7	0		
8	10	0	DEFR4K	APPCHR
8	11	0		
9	12	0	DEFR4K	APPREC
9	13	0		
10	14	0	DEFR4K	ARCLRC
10	15	0		
11	16	0	DEFR4K	AROTAT
11	17	0		
12	20	0	DEFR4K	ATOX
12	21	0		
13	22	0	DEFR4K	CLRFL
13	23	0		
14	24	0	DEFR4K	CLKEYS
14	25	0		
15	26	0	DEFR4K	CRFLAS
15	27	0		
16	30	0	DEFR4K	CRFLD
16	31	0		
17	32	0	DEFR4K	DELCHR
17	33	0		
18	34	0	DEFR4K	DELREC
18	35	0		
19	36	0	DEFR4K	EMDIR
19	37	0		
20	40	0	DEFR4K	FLSIZE
20	41	0		
21	42	0	DEFR4K	GETAS
21	43	0		
22	44	0	DEFR4K	GETKEY
22	45	0		
23	46	0	DEFR4K	GETP
23	47	0		
24	50	0	DEFR4K	GETR
24	51	0		
25	52	0	DEFR4K	GETREC
25	53	0		
26	54	0	DEFR4K	GETRX
26	55	0		
27	56	0	DEFR4K	GETSUB
27	57	0		
28	60	0	DEFR4K	GETXX
28	61	0		
29	62	0	DEFR4K	INSCHR
29	63	0		
30	64	0	DEFR4K	INSREC
30	65	0		

ADVANCED PROGRAMMING ROM ID
OF FUNCTIONS
ROM HEADER - "ADVANCED 1X"

ALPHA LENGTH

ALPHA FORMATTED NUMBER

APPEND ALPHA TO CURRENT RECORD

APPEND ALPHA TO END OF FILE

ALPHA RECALL RECORD

ALPHA ROTATE

SHIFT LEFTMOST ALPHA CHAR TO X

CLEAR FILE

CLEAR ALL KEY ASSIGNMENTS

CREATE AN ASCII FILE

CREATE A DATA FILE

DELETE CHAR BY X

DELETE RECORD

EXT.MEM DIRECTORY

ASK FILE SIZE

GET ASCII FILE FROM MASS.MEM

GET KEY

GET PROGRAM

GET REGISTERS

GET RECORD FROM TEXT FILE

GET REGISTERS BY X

GET SUBROUTINE

GET A REGISTER FROM A REG FILE TO X

INSERT CHAR

INSERT RECORD

31	66	0	DEFR4K	PASH	PROGRAMMABLE ASSIGN
31	67	0			
32	70	0	DEFR4K	PCLPS	PROGRAMMABLE CLEAR PROGRAMS
32	71	0			
33	72	0	DEFR4K	POSA	POSITION ALPHA
33	73	0			
34	74	0	DEFR4K	POSFL	POSITION FILE
34	75	0			
35	76	0	DEFR4K	PSIZE	PROGRAMMABLE SIZE
35	77	0			
36	100	0	DEFR4K	PURFL	PURGE A FILE IN EXT.MEM
36	101	0			
37	102	0	DEFR4K	RCLFLG	RECALL FLAGS
37	103	0			
38	104	0	DEFR4K	RCLPT	RECALL POINTER
38	105	0			
39	106	0	DEFR4K	RCLPTA	RECALL POINTER BY ALPHA NAME
39	107	0			
40	110	0	DEFR4K	REGMOV	REGISTERS MOVE
40	111	0			
41	112	0	DEFR4K	REGSWP	REGISTERS SWAP
41	113	0			
42	114	0	DEFR4K	SAVEAS	SAVE ASCII FILE TO MASS.MEM
42	115	0			
43	116	0	DEFR4K	SAVEP	SAVE PROGRAM
43	117	0			
44	120	0	DEFR4K	SAVER	SAVE ALL REGISTERS
44	121	0			
45	122	0	DEFR4K	SAVERX	SAVE REGISTERS BY X
45	123	0			
46	124	0	DEFR4K	SAVEX	SAVE X REGISTER TO A REG FILE
46	125	0			
47	126	0	DEFR4K	SEKPT	SEEK POINTER
47	127	0			
48	130	0	DEFR4K	SEKPTA	SEEK POINTER BY ALPHA FILE NAME
48	131	0			
49	132	0	DEFR4K	SIZE?	ASK CURRENT SIZE OF THE MACHINE
49	133	0			
50	134	0	DEFR4K	STOFLG	RESTORE FLAGS
50	135	0			
51	136	0	DEFR4K	X<>F	X EXCHANGE WITH FLAG 0-7
51	137	0			
52	140	0	DEFR4K	XTOA	X APPEND TO ALPHA REGISTER
52	141	0			
53	142	0	CON	0	
54	143	0	CON	0	

*
*

57	144	203	CON	@203	C
58	145	61	CON	@61	I
59	146	40	CON	@40	
60	147	16	CON	@16	N
61	150	3	CON	@03	C
62	151	6	CON	@06	F
63	152	40	CON	@40	
64	153	24	CON	@24	T
65	154	30	CON	@30	X
66	155	5	CON	@05	E
67	156	55	CON	@55	-
68			APHEAD	ENTRY	APHEAD

 * CLKEYS - CLEAR ALL KEY ASSIGNMENT *

74	ENTRY	CLKEYS	
76 157	223 CON	0223	S
77 160	31 CON	031	Y
78 161	5 CON	005	E
79 162	13 CON	013	K
80 163	14 CON	014	L
81 164	3 CON	003	C
82 165	CLKEYS 1770 C=REGN	15	CLEAR BIT MAP FIRST
83 166	136 C=0	S	
84 167	132 C=0	M	
85 170	1750 REGN=C	15	LEAVES LINE NUMBER ALONE
86 171	1250 REGN=C	10	DOESN'T LEAVE SCRATCH CLEAR
87 172	1570 C=REGN	13	GET FINAL END ADDR
88 173	346 BC EX	X	AND SAVE IN B.X
89 174	460 LDI		
90 175	300 CON2	12 0	0000
91 176	406 A=C	X	A.X= BOTTOM MEMORY ADDR
92 177	CLK10 1446 ? A<B	X	REACHED FINAL END YET ?
93 200	143 GONC	CLK20 (214)	YES, DONE
94 201	246 C=A	X	A<>C SBX
94 202	406		A= C SBX
95 203	1160 DADD=C		
96 204	70 C=DATA	REAL DATA	
97 205	1076 C=C+1	S	IS THIS A KEY REG ?
98 206	63 GONC	CLK20 (214)	NO, CLEARED THEM ALL
99 207	116 C=0	W ALL	
100 210	1176 C=C-1	S FB	CLEAR THE KEY ASSIGNMENT IN REG
101 211	1360 DATA=C		
102 212	546 A=A+1	X	
103	LEGAL		
104 213	1643 GOTO	CLK10 (177)	
105 214	CLK20 1 GOSUB	GTFEND	NOW CLEAR ALL THE USER PROG ASSIGNMEN
105 215	0		
106 216	1074 RCR	2	
107 217	CLK30 1 GOSUB	UPLINK	
107 220	0		
108 221	1076 C=C+1	S	ALPHA LABEL ?
109 222	173 GONC	CLK40 (241)	NO, IT IS AN END
110 223	160 N=C		SAVE THE LINK IN N
111 224	256 C=A	W	
111 225	416		
112 226	530 M=C		SAVE THE ADDR IN M
113 227	1 GOSUB	INCAD2	GET 3RD BYTE OF THE CHAIN
113 230	0		
114 231	1 GOSUB	INCADA	
114 232	0		
115 233	106 C=0	X	
116 234	1 GOSUB	PTBYTA	ZERO THE KEY CODE
116 235	0		
117 236	630 C=M		
118 237	416 A=C		
119 240	260 C=N		
120 241	CLK40 1346 ? C#0	X	REACHED CHAIN END ?
121 242	1557 GOC	CLK30 (217)	NO YET

122 243
123 244
123 245

1160 DADD=C
1 GOLONG PKIOAS
2

ENABLE CHIP 00

*
*

* PSIZE - PROGRAMMABLE SIZE FUNCTION *
* THE SIZE IS TAKEN FROM THE X REGISTER *

*

131 ENTRY PSIZE

*

133	246	205	CON	0205	E
134	247	32	CON	032	Z
135	250	11	CON	011	I
136	251	23	CON	023	S
137	252	20	CON	020	P
138	253	PSIZE	1	GOSUB	GOSUB
138	254		0		GET INT(X)
139	255		0	XDEF	X<999
140	256		160	N=C	
141	257	1370	C=REGN	11	SAVE USER RTH STACK IN REG.9&8
142	260	1150	REGN=C	9	
143	261	1070	C=REGN	8	
144	262	134	PT=	4	
145	263	130	G=C		SAVE REG.8[5:4] IN G
146	264	34	PT=	3	
147	265	416	A=C	W	
148	266	1470	C=REGN	12	
149	267	252	AC EX	WPT	
150	270	1050	REGN=C	8	
151	271	260	C=N		
152	272	1104	S9=	0	
153	273	1	GOSUB	SIZSUB	TRY TO DO THE NEW SIZE
153	274		0		
154	275	1070	C=REGN	8	RESTORE PART OF REG.8 FIRST
155	276	134	PT=	4	
156	277	330	CG EX		
157	300	1050	REGN=C	8	
158	301	1114	?S9=1		ENOUGH ROOM TO DO THE NEW SIZE
159	302	253	GONC	PSIZ10 (327)	YES

*

161		ENTRY	NORMEX	
162	303	NORMEX	1	GOSUB
162	304		0	LDSST0
163	305	1314	?S13=1	RUNNING ?
164	306	47	GOC	NORM (312)
165	307	114	?S4=1	YES
166	310	1	GOLNC	PACKE
166	311		2	SST ?
				NO, PACK AND SAY "TRY AGAIN"

*

169		ENTRY	NORM	
169	312	NORM	1	GOSUB
169	313		0	GOSUB3
170	314		0	XDEF
171	315	16	CON	016
172	316	17	CON	017
173	317	40	CON	040
174	320	22	CON	022
175	321	17	CON	017

```

176 322      17 CON      Q17      0
177 323     1015 CON     Q1015    M
178 324      1 GOSUB    GOL3
178 325      0
179 326      0 XDEF     APEREX

```

* WHEN RETURN FROM "SIZSUB", M.X HAS THE THE DISPLACEMENT BETWEEN THE
* OLD AND NEW SIZE IN BINARY. NOW WE WILL PUT THE OLD RTN STACK BACK
* FIRST, AND THEN UPDATE IT ONE AT A TIME. WE WILL ROTATE THE RTN STACK
* TO THE RIGHT ONE BY ONE AND ADD THE DISPLACEMENT TO IT IF IT IS A
* RAM ADDRESS.

```

186 327 PSIZ10 1170 C=REGN 9      UPDATE AND RESTORE RTN STACK
187 330      1350 REGN=C 11
188 331      1470 C=REGN 12
189 332      416 A=C      W
190 333      1070 C=REGN 8
191 334      330 CG EX      RESTORE PART OF THE ADDR FROM G
192 335      34 PT=      3
193 336      252 AC EX WPT      PC ALREADY UPDATED BY "SIZSUB"
194 337      1450 REGN=C 12
195 340      340 SEL Q
196 341      1234 PT=      7
197 342      240 SEL P
198 343 PSIZ20 34 PT=      3
199 344      1470 C=REGN 12
200 345      416 A=C      W
201 346      1370 C=REGN 11
202 347      252 AC EX WPT
203 350      174 RCR      4
204 351      1350 REGN=C 11
205 352      256 AC EX W
206 353      174 RCR      4
207 354      1450 REGN=C 12
208 355      416 A=C      W
209 356      340 SEL Q
210 357      1724 DEC PT
211 360      1624 ? PT= 0      ALL DONE ?
212 361      1540 RTN C      YES
213 362      240 SEL P
214 363      1352 ? C#0 WPT      RTN ADDR ZERO ?
215 364      1573 GONC PSIZ20 ( 343) YES, CONTINUE TO NEXT ONE
216 365      1342 ? C#0 PT      A ROM ADDRESS ?
217 366      1557 GOC PSIZ20 ( 343) YES, DON'T TOUCH IT
218 367      630 C=M      GET THE DISPLACEMENT
219 370      506 A=A+C X      ADD IT
220 371      256 AC EX W
221 372      1450 REGN=C 12      DONE WITH ONE RTN ADDR
222 373      1503 GOTO PSIZ20 ( 343)

```

*
*

* PASH - PROGRAMMABLE ASSIGN FUNCTION *
* THE FUNCTION OR PROGRAM IS TAKEN FROM ALPHA REGISTER, THE KEY *
* CODE IS TAKEN FROM X REGISTER. THE KEY CODE IS THE SAME AS THEY *
* WILL SEE WHEN EXECUTING THE "ASN" FUNCTION. POSITIVE NUMBER FOR *
* UNSHIFTED KEY, NEGATIVE NUMBER FOR THE SHIFTED KEY. *

233 ENTRY PASH

235	374	216	CON	@216	N
236	375	23	CON	@23	S
237	376	1	CON	@01	A
238	377	20	CON	@20	P
239	400	1	GOSUB	GOSUB1	GET FCN/PROG NAME
239	401	0			
240	402	0	XDEF	GTPRNA	
241	403	630	C=M		
242	404	1150	REGN=C	9	SAVE NAME IN REG.9
243	405	370	C=REGN	3	
244	406	1366	? C#0	XS	X < 1 ?
245	407	163	GONC	PASN10 (425)	NO
246	410	1	GOSUB	GOSUB3	SAY "KEY CODE ERR"
246	411	0			
247	412	0	XDEF	APERMG	
248	413	13	CON	@13	K
249	414	5	CON	@05	E
250	415	31	CON	@31	Y
251	416	3	CON	@03	C
252	417	17	CON	@17	O
253	420	4	CON	@04	D
254	421	1005	CON	@1005	E
255	422	1	GOSUB	GOL3	
255	423	0			
256	424	0	XDEF	DISERR	
257	425	PASN10	416	A=C	W
258	426	1146	C=C-1	X	X < 10 ?
259	427	1617	GOC	PASNER (410)	YES, KEY CODE ERR
260	430	1146	C=C-1	X	X > 99 ?
261	431	1573	GONC	PASNER (410)	YES, KEY CODE ERR
262	432	674	ROR	11	CI[1]= ROW #, CI[0]= COLUMN #
263	433	406	A=C	X	
264	434	1634	PT=	0	
265	435	1342	? C#0	PT	COLUMN # = 0 ?
266	436	1523	GONC	PASNER (410)	YES, KEY CODE ERR
267	437	460	LDI		
268	440	220	CON2	9 0	
269	441	1434	PT=	1	
270	442	1402	? A#C	PT	ROW # <= 8 ?
271	443	1453	GONC	PASNER (410)	NO, SAY "KEY CODE ERR"
272	444	460	LDI		
273	445	61	CON2	3 1	DON'T ALLOW SHIFT KEY
274	446	1552	?A#C	WPT	
275	447	1413	GONC	PASNER (410)	SAY "KEYCODE ERR"
276	450	PASN12	460	LDI	
277	451	106	CON2	4 6	
278	452	1402	? A#C	PT	ROW 1-3 ?
279	453	27	GOC	PASN15 (455)	YES, MAX. COLUMN # IS 5
280	454	1146	C=C-1	X	ROW 4-8, MAX. COL. # IS 4
281	455	PASN15	1634	PT=	0
282	456	1402	? A#C	PT	COL. # TOO BIG ?
283	457	1313	GONC	PASNER (410)	YES, SAY "KEY CODE ERR"
284	460	120	LC	1	
285	461	1434	PT=	1	
286	462	1542	? A#C	PT	ROW 4 ?
287	463	47	GOC	PASN20 (467)	NO
288	464	1552	? A#C	WPT	ENTER KEY ?
289	465	23	GONC	PASN20 (467)	YES, THE REST OF THE ROW 4 KEYS
290	466	552	A=A+1	WPT	IS OFF BY ONE
291	467	PASN20	646	A=A-1	X

```

292 470          246 C=A      X          PUT A[1]=COLUMN #, A[0] = ROW #
292 471          406
293 472          1706 C SR    X
294 473          1746 A SL    X
295 474          242 AC EX    PT
296 475          406 A=C      X
297 476          1536 ? A#0   S          X NEGATIVE ?
298 477          43 GONC      PASN30 ( 503 ) NO, UNSHIFTED KEY
299 500          460 LDI
300 501          10 CON       8
301 502          506 A=A+C    X
302          LEGAL
303 503 PASN30    1 GOLONG XASN
303 504          2

```

*

*

```

*****
* ALENG - ALPHA LENGTH                                     *
* RECALL THE ALPHA LENGTH TO THE X REGISTER               *
*****

```

*

```

311 505          207 CON      @207          G
312 506          16 CON      @16           N
313 507          5 CON       @05           E
314 510          14 CON      @14           L
315 511          1 CON       @01           A
316          ENTRY ALENG
317 512 ALENG     1 GOSUB     GOSUB
317 513          0
318 514          0 XDEF      ALEN
319 515          453 GOTO     ATOX20 ( 562 ) PUT THE NUMBER TO X

```

*

```

* ALEN - ROUTINE TO COMPUTE THE ALPHA LENGTH
* INPUT : DON'T CARE
* OUTPUT : A.X=C.X= ALPHA LENGTH
* USED A[3:0], B[3:0], C, PT=3, S2, S3 +2 SUB LEVEL

```

*

```

327          ENTRY ALEN

```

*

```

329 516 ALEN     1 GOSUB     GOSUB          GET ADDR OF 1ST CHAR OF ALPHA
329 517          0
330 520          0 XDEF      FAHED
331 521          1014 ?S2=1          ALPHA EMPTY ?
332 522          33 GONC      ALEN10 ( 525 ) NO
333 523          6 A=0        X
334 524          113 GOTO     ALEN20 ( 535 )
335 525 ALEN10   116 C=0      W
336 526          460 LDI
337 527          5 CON       5          C[3:0]= ADDR OF ALPHA END
338 530          352 BC EX    WPT
339 531          1 GOSUB     GOSUB          COUNT THE # OF BYTES BETWEEN THE
339 532          0
340 533          0 XDEF      CNTBY7        TWO ADDRESSES
341 534          546 A=A+1    X
342 535 ALEN20   246 C=A      X
342 536          406
343 537          1740 RTN

```

*

```

* ATOX - ALPHA TO X
* SHIFT THE LEFTMOST CHAR OF THE ALPHA REGISTER INTO THE X REGISTER
* IF ALPHA EMPTY, IT WILL LEAVE A ZERO IN THE X REGISTER
*****

```

```

351 540      230 CON    @230      X
352 541      17 CON    @17       O
353 542      24 CON    @24       T
354 543       1 CON    @01       A

```

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

```

356          ENTRY  ATOX
357          ENTRY  ATOX20

```

```

359 544 ATOX      1 GOSUB  GOSUB      GET THE LEFTMOST CHAR ADDR
359 545          0
360 546          0 XDEF    FAHED      THE 1ST CHAR WILL BE IN G
361 547          1014 ?S2=1.        ALPHA EMPTY
362 550          33 GONC    ATOX10 (< 553) NO
363 551          56 B=0     W
364 552          133 GOTO   ATOX30 (< 565)
365 553 ATOX10    106 C=0     X
366 554          1 GOSUB    PTBYTA     SHIFT OFF THE CHAR
366 555          0
367 556          1634 PT=    0
368 557          230 C=G
369 560          126 C=0     XS
370 561 ATOX15    406 A=C     X
371 562 ATOX20    1 GOSUB    GOSUB      CONVERT THE BINARY TO DECIMAL
371 563          0
372 564          0 XDEF    BIN-D
373 565 ATOX30    1 GOLONG  RCL        RECALL THE NUMBER TO X
373 566          2

```

```

*****
* FLSIZE - FILE SIZE
* GET THE FILE SIZE OF AN EXTERNAL MEMORY FILE. THE FILE NAME IS
* TAKEN FROM THE ALPHA REGISTER. IF ALPHA IS EMPTY, IT WILL
* DEFAULT TO THE CURRENT FILE.
*****

```

```

383          ENTRY  FLSIZE
384 567      205 CON    @205      E
385 570      32 CON    @32       Z
386 571      11 CON    @11       I
387 572      23 CON    @23       S
388 573      14 CON    @14       L
389 574       6 CON    @06       F
390 575 FLSIZE 1610 S0=    1
391 576       1 GOSUB    GOSUB2
391 577       0
392 600       0 XDEF    FLSHAP     GET FILE ENTRY
393 601      260 C=N
394 602     1573 GOTO   ATOX15 (< 561)

```

```

*****
* SIZE? - RECALL THE SIZE TO THE X REGISTER
*****

```

```

400          ENTRY  SIZE?

```

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

```

401 603      277 CON      @277      ?
402 604      5  CON      @05       E
403 605      32 CON      @32       Z
404 606      11 CON      @11       I
405 607      23 CON      @23       S
406 610 SIZE?      1 GOSUB  FNDEND
406 611          0
407 612          116 C=0
408 613      1160 DADD=C
409 614      1570 C=REGN 13
410 615          74 RCR      3
411 616          706 A=A-C  X
412 617      1433 GONC    ATQX20 ( 562) SIZE = 0
413 620          16 A=0
414 621      1413 GOTO    ATQX20 ( 562)

```

*
*

```

* X<256- ROUTINE TO GET INT(X) AND CHECK IF INT(X) < 256
*   INPUT : CHIP 00 ENABLE
*           IF INT(X) >= 256 WILL EXIT TO "DATA ERROR"
*           IF X HAS A STRING, WILL SAY "ALPHA DATA"
* X<999 - GET INT(X) AND CHECK IF INT(X) < 1000
*   INPUT : CHIP 0 ENABLE
*   OUTPUT : A,X = C,X = INT(DECIMAL NUMBER)
*   USED A,B,X,C,S8      +2 SUB LEVEL

```

*

```

427          ENTRY  X<256
428          ENTRY  X<999

```

*

```

430 622 X<999      370 C=REGN 3
431 623          416 A=C      W
432 624          460 LDI
433 625          1750 CON      1000
434 626          53 GOTO      INT10 ( 633)
435 627 X<256      370 C=REGN 3
436 630          416 A=C      W
437 631          460 LDI
438 632          400 CON      256
439 633 INT10      346 BC EX  X
440 634          256 AC EX  W
441 635          1  GOSUB    CHK#S      SEE IF IT IS A NUMBER
441 636          0
442 637          1140 SETHEX
443 640          416 A=C      W
444 641          1526 ? A#0  XS      IS THE NUMBER < 1 ?
445 642          77  GOC      INT20 ( 651) YES, ITS INTEGER IS ZERO ANYWAY
446 643          460 LDI
447 644          3  CON      3
448 645          1406 ? A<C  X      IS THE NUMBER < 1000 ?
449 646 INTER      1  GOLNC  ERRDE    NO, SAY "DATA ERROR"
449 647          2
450 650          256 AC EX  W
451 651 INT20      1  GOSUB    BCDBIN    CONVERT THE NUMBER TO BINARY
451 652          0
452 653          406 A=C      X      A,X = THE BINARY OF THE NUMBER
453 654          1446 ? A<B  X      IS THE NUMBER TOO BIG ?
454 655          1713 GONC    INTER ( 646)
455 656          1740 RTN

```

*

* BIN-D - ROUTINE TO CONVERT A BINARY NUMBER INTO A NORMALIZED NUMBER

* INPUT : A.X = BINARY NUMBER

* OUTPUT : B = NORMALIZED DECIMAL NUMBER

* CHIP 00 ENABLE

* USED A, B, C, S0 +1 SUB LEVEL

* BIN-D3 - SPECIAL ENTRY WITH S0=1 AS INPUT

*

*

466		ENTRY BIN-D
467		ENTRY BIN-D3

*

AFH	469	657	BIN-D	1604	S0=	0	
	470	660		36	A=0	S	
BIN	471	661	BIN-D3	460	LDI		
	472	662		20	CON	Q20	
	473	663		1160	DADD=C		UNSELECT RAM
	474	664		106	C=0	X	
	475	665		1760	PFAD=C		UNSELECT PERIPHERALS
	476	666		1	GOSUB	GENNUM	CONVERT TO DECIMAL DIGITS
	476	667		0			

* NOW B.S = # OF DECIMAL DIGITS

* A.M = LEFT JUSTIFIED DECIMAL DIGITS

	479	670		116	C=0	W	
	480	671		1160	DADD=C		SELECT CHIP 00
	481	672		1614	?S0=1		COME IN FROM BIN-D3 ?
	482	673		1540	RTN C		YES, RETURN HERE
	483	674		336	C=B	S	
	484	675		634	PT=	11	
	485	676		1176	C=C-1	S	COMPUTE THE EXP OF THE NUMBER
	486	677	BIND10	1176	C=C-1	S	
	487	700		47	GOC	BIND20 (704)	1C4
	488	701		1046	C=C+1	X	
	489	702		1724	DEC PT		
	490				LEGAL		
	491	703		1743	GOTO	BIND10 (677)	1BF
	492	704	BIND20	12	A=0	WPT	
	493	705		406	A=C	X	A.X= EXP OF THE NUMBER
	494						
	495		ENTRY	BIND25			(CALLED BY "ASLEFT" IN 410X)
	496						
	497	706	BIND25	1534	PT=	12	SHIFT OUT LEADING ZEROS
	498	707	BIND30	1502	? A#0	PT	
	499	710		57	GOC	BIND40 (715)	1CD
	500	711		1772	A SL	M	
	501	712		646	A=A-1	X	
	502	713		1743	GONC	BIND30 (707)	1C7
	503	714		16	A=0	W	
	504	715	BIND40	36	A=0	S	
	505	716		216	B=A	W	
	506	717		1740	RTN		

*

* FAHED - ROUTINE TO FIND THE ADDR OF THE 1ST CHAR OF THE ALPHA REGISTER

* INPUT : DON'T CARE

* OUTPUT : IF S2 = 1, ALPHA REGISTER EMPTY

* IF S2 = 0, THEN :

* A[3:0] = ADDR OF FIRST CHARACTER

* G = 1ST CHAR IN THE ALPHA REGISTER

* PT = 3

* USED A[3:0], C, PT, S2, +1 SUB LEVEL

* 518 ENTRY FAHD

```

520 720 FAHD      1 GOSUB  GOSUB      LOAD ADDR HEX 6008
520 721          0
521 722          0 XDEF    ADR608
522 723          1004 S2=    0
523 724 FAHD10    1 GOSUB  INCADA      SKIP OVER LEADING NULLS
523 725          0
524 726          1 GOSUB  GTBYTA
524 727          0
525 730          126 C=0     XS
526 731          1346 ?C#0   X
527 732          77 GOC     FAHD20 ( 741)
528 733          460 LDI
529 734          5 CON      5
530 735          102 C=0     PT
531 736          1552 ? A#C   WPT      REACHED ALPHA END ?
532 737          1657 GOC     FAHD10 ( 724) NOT YET
533 740          1010 S2=    1
534 741 FAHD20    1634 PT=    0
535 742          130 G=C      SAVE 1ST CHAR IN G
536 743 FAHD30    34 PT=     3
537 744          1070 C=REGN  8      SAVE ALPHA HEAD IN REG.8[13:10]
538 745          374 RCR      10
539 746          252 C=A      WPT
539 747          412
540 750          174 RCR      4
541 751          1050 REGN=C  8
542 752          1740 RTN

```

* 544 ENTRY ADR608

```

546 753 ADR608    116 C=0
547 754          1160 DADD=C
548 755          1070 C=REGN  8
549 756          1334 PT=    13
550 757          620 LC      6
551 760          20 LC      0
552 761          20 LC      0
553 762          1020 LC      8
554 763          1050 REGN=C  8
555 764          34 PT=     3
556 765          374 RCR      10
557 766          412 A=C      WPT
558 767          1740 RTN

```

*

* ANUM - ALPHA FORMATTED NUMBER *
* EXTRACT AN ALPHA FORMATTED NUMBER FROM THE ALPHA REGISTER TO X. *
* IF A NUMBER IS FOUND, IT WILL SET FLAG 22 AND RECALL THE NUMBER *
* TO X. IF NO NUMBER IS FOUND, THEN NO ACTION WILL BE TAKEN. *

* 567 ENTRY ANUM

```

569 770          215 CON      @215      M
570 771          25 CON      @25        U
571 772          16 CON      @16        N

```

572	773		1 CON	001	A
573	774	ANUM	1 GOSUB	GOSUB	GET ADDR OF ALPHA HEAD
573	775		0		
574	776		0 XDEF	FAHED	
575	777		1014 ?S2=1		ALPHA EMPTY ?
576	1000		1540 RTN C		YES, DON'T DO ANYTHING
577	1001		212 B=A	WPT	SAVE NEXT CHAR ADDR IN B
578	1002	ANUM10	116 C=0	W	INITIALIZE DIGIT ENTRY
579	1003		1156 C=C-1	W	
580	1004		126 C=0	XS	
581	1005		1334 PT=	13	
582	1006		1220 LC	10	
583	1007		1150 REGN=C	9	
584	1010		1 GOSUB	STBT10	
584	1011		0		
585	1012		674 RCR	11	SET MANTISSA NON-ZERO BIT
586	1013		10 S3=	1	
587	1014		1630 C=ST		
588	1015		74 RCR	3	
589	1016		1050 REGN=C	8	
590	1017	ANUM20	1104 S9=	0	
591	1020		34 PT=	3	
592	1021		152 A=B	WPT	GET NEXT CHAR ADDR BACK FROM B
592	1022		212		
593	1023		1 GOSUB	GTBYTA	GET NEXT CHAR
593	1024		0		
594	1025		126 C=0	XS	
595	1026		406 A=C	X	A.X = CHAR
596	1027		460 LDI		
597	1030		72 CON	58	ASCII 0-9 = 48-57
598	1031		1406 ? ACC	X	
599	1032		543 GONC	ANUM60 (1106)	MIGHT BE "E"
600	1033		460 LDI		
601	1034		60 CON	48	
602	1035		1406 ? ACC	X	
603	1036		627 GOC	ANUM40 (1120)	NOT A DIGIT, MIGHT BE ONE OF "+, -, ."
604	1037		460 LDI		
605	1040		40 CON	040	
606	1041		1106 C=A-C	X	CONVERT ASCII DIGIT TO FCN CODE
607	1042	ANUM25	1634 PT=	0	
608	1043		130 G=C		PUT THE FUNCTION CODE TO G
609	1044		1 GOSUB	DIGENT	CALL DIGIT ENTRY ROUTINE
609	1045		0		
610	1046	ANUM30	34 PT=	3	
611	1047		152 AB EX	WPT	GET THE CHAR ADDR BACK FROM B
612	1050		116 C=0	W	
613	1051		460 LDI		
614	1052		5 CON	5	
615	1053		1552 ? A#C	WPT	IS THIS LAST CHAR IN ALPHA REG ?
616	1054		133 GONC	ANUM75 (1067)	YES
617	1055		1 GOSUB	INCADA	POINT TO NEXT CHAR
617	1056		0		
618	1057		152 AB EX	WPT	
619	1060		1114 ?S9=1		LAST CHAR RECOGNIZED ?
620	1061		1217 GOC	ANUM10 (1002)	NO, RE-INITIALIZE DIGIT ENTRY
621	1062		1353 GOTO	ANUM20 (1017)	CONTINUE ON NEXT CHAR
622	1063	ANUM70	1110 S9=	1	THIS CHAR. NOT RECOGNIZED
623	1064		1170 C=REGN	9	SEE IF ANY ENTRY DETECTED AT ALL
624	1065		1072 C=C+1	M	
625	1066		1607 GOC	ANUM30 (1046)	NO, LOOK FOR NEXT CHAR.

626	1067	ANUM75	1170	C=REGN	9	CHECK AGAIN IF ANYTHING WAS DETECTED.
627	1070		1072	C=C+1	M	FOR WHEN THE ALPHA REG IS EMPTIED
628	1071		1540	RTN	C	NO, DON'T CHANGE X OR FLAG 22
629	1072		1670	C=REGN	14	SET NUMERIC INPUT FLAG (F22)
630	1073		474	RCR	8	
631	1074		1730	CST	EX	
632	1075		1410	S1=	1	
633	1076		1730	CST	EX	
634	1077		574	RCR	6	
635	1100		1650	REGN=C	14	
636	1101		614	?S11=1		PUSH FLAG SET ?
637	1102		1	GSUBC	R'SUB	PUSH STACK IF SET
637	1103		1			
638	1104		1	GOLONG	NOREG9	
638	1105		2			
639	1106	ANUM60	460	LDI		
640	1107		105	CON	69	ASCII "E"
641	1110		1546	? A#C	X	
642	1111		1527	GOC	ANUM70 (1063)	UNRECOGNIZED ENTRY
643	1112		1170	C=REGN	9	CHECK IF ANY DIGITS YET
644	1113		1072	C=C+1	M	
645	1114		1477	GOC	ANUM70 (1063)	NO, DON'T RECOGNIZE "E"
646	1115		460	LDI		
647	1116		33	CON	033	FCN CODE FOR EEX
648	1117		1233	GOTO	ANUM25 (1042)	
649	1120	ANUM40	460	LDI		
650	1121		53	CON	43	ASCII "+"
651	1122		1546	? A#C	X	
652	1123	ANUM41	1233	GONC	ANUM30 (1046)	ALWAYS IGNORE "+"
653	1124		1046	C=C+1	X	C.X = ASCII " , "
654	1125		1546	? A#C	X	
655	1126		67	GOC	ANUM45 (1134)	
656	1127		214	?S5=1		DECIMAL POINT FLAG SET ?
657	1130		177	GOC	ANUM48 (1147)	YES, CHECK DIGIT GROUPING FLAG
658	1131	ANUM42	460	LDI		
659	1132		32	CON	032	FCN CODE FOR DECIMAL POINT
660	1133	ANUM44	1073	GOTO	ANUM25 (1042)	
661	1134	ANUM45	1046	C=C+1	X	C.X= ASCII "-"
662	1135		1546	? A#C	X	
663	1136		47	GOC	ANUM47 (1142)	
664	1137		460	LDI		
665	1140		34	CON	034	FCN CODE FOR MINUS SIGN
666	1141		1723	GOTO	ANUM44 (1133)	
667	1142	ANUM47	1046	C=C+1	X	C.X = ASCII " . "
668	1143		1546	? A#C	X	
669	1144		1177	GOC	ANUM70 (1063)	NOT " . ", UNRECOGNIZED ENTRY
670	1145		214	?S5=1		DECIMAL POINT FLAG SET ?
671	1146		1637	GOC	ANUM42 (1131)	LOAD FCN CODE OF D.P.
672	1147	ANUM48	114	?S4=1		DIGIT GROUPING FLAG SET?
673	1150		1133	GONC	ANUM70 (1063)	NO, DON'T RECOGNIZE SEPARATOR
674	1151		1523	GOTO	ANUM41 (1123)	OTHERWISE, IGNORE SEPARATOR

* AROT - ALPHA ROTATE *

* ROTATE THE CONTENTS OF THE ALPHA REGISTER X NUMBER OF CHARS. *

* ROTATE LEFT IF X IS POSITIVE, ROTATE RIGHT IF X IS NEGATIVE. *

* 662 ENTRY AROTAT *

684	1152	224	CON	0224	T
685	1153	17	CON	017	O
686	1154	22	CON	022	R
687	1155	1	CON	001	A
688	1156	1	GOSUB	GOSUB	COMPUTE ALPHA LENGTH
688	1157	0			
689	1160	0	XDEF	ALEN	
690	1161	1014	?S2=1		ALPHA EMPTY ?
691	1162	1540	RTN C		YES, NO NEED TO ROTATE
692	1163	160	N=C		SAVE ALPHA LENGTH IN N
693	1164	1	GOSUB	GOSUB	GET INT(X)
693	1165	0			
694	1166	0	XDEF	X<256	
695	1167	1346	? C#0	X	X=0 ?
696	1170	1640	RTN NC		YES, NO NEED TO ROTATE
697	1171	406	A=C	X	SAVE ROTATION COUNT IN A.X
698	1172	370	C=REGN	3	
699	1173	376	BC EX	S	SAVE THE SIGN OF X IN B.S
700	1174	260	C=N		GET ALPHA LENGTH BACK FROM N
701	1175	706	A=A-C	X	DO X MOD ALEN
702	1176	1773	GONC	AROT10 (1175)	
703	1177	506	A=A+C	X	A.X = ACTUAL ROTATION COUNT
704	1200	1336	? B#0	S	IS X NEGATIVE ?
705	1201	33	GONC	AROT20 (1204)	NO, X IS POSITIVE
706	1202	246	AC EX	X	A.X=ALEN C.X=ROTATION COUNT
707	1203	706	A=A-C	X	CONVERT ROTATE RIGHT TO LEFT
708	1204	246	AC EX	X	SAVE ROTATION COUNT IN N.X
709	1205	160	N=C		
710	1206	1	GOSUB	GOSUB	GET ADDR OF ALPHA HEAD
710	1207	0			
711	1210	0	XDEF	FAHED	
712	1211	252	AC EX	WPT	
713	1212	530	M=C		SAVE ALPHA HEAD ADDR IN M
714	1213	260	C=N		
715	1214	1146	C=C-1	X	DECREMENT ROTATION COUNT
716	1215	1540	RTN C		ALL DONE
717	1216	160	N=C		PUT ROTATION BACK TO N.X
718	1217	34	PT=	3	
719	1220	630	C=M		GET ALPHA HEAD ADDR BACK
720	1221	412	A=C	WPT	
721	1222	1	GOSUB	GTBYTA	GET LEFTMOST CHAR
721	1223	0			
722	1224	1634	PT=	0	
723	1225	130	G=C		
724	1226	106	C=0	X	
725	1227	34	PT=	3	
726	1230	1	GOSUB	PTBYTA	KICK OUT LEFTMOST CHAR
726	1231	0			
727	1232	1	GOSUB	APNDNW	SHIFT THE LEFTMOST CHAR IN
727	1233	0			
728	1234	1573	GOTO	AROT30 (1213)	

*
*

 * POSR - X POSITION ALPHA REGISTER *
 * X WILL BE TAKEN AS THE SUBSTRING. IF X HAS A NUMBER FROM 0-255, *
 * IT WILL BE TAKEN AS A SINGLE CHAR SUBSTRING. IF THE NUMBER IS *
 * GREATER THEN 255, THEN "DATA ERROR" WILL BE GENERATED. SEARCH *
 * THE ALPHA REGISTER FOR THE SUBSTRING GIVEN IN X AND PUT THE *
 * POSITION OF THE SUBSTRING IN X. IF THE SUBSTRING IS NOT FOUND, *

* A "-1" WILL BE LEFT IN THE X REGISTER. THE SUBSTRING WILL BE *
 * SAVED IN LAST X. THE POSITION OF THE CHAR IS COUNTED FROM LEFT *
 * TO RIGHT STARTING FROM ZERO. *

```

*
743          ENTRY  POSA
744          ENTRY  XPOAFN
745          ENTRY  XPOANF

*
747 1235      201 CON   0201      A
748 1236      23 CON   023        S
749 1237      17 CON   017        0
750 1240      20 CON   020        P
751 1241 POSA    370 C=REGN 3
752 1242      1176 C=C-1 S
753 1243      1376 ? C#0 S          X HAS A STRING ?
754 1244      137 GOC   XPOA20 (1257) NO, X HAS A NUMBER
755 1245      1534 PT=   12
756 1246      20 LC     0
757 1247      1356 ? C#0 W          A NULL STRING ?
758 1250      673 GONC  XPOANF (1337) YES, RETURN -1
759 1251      1434 PT=   1
760 1252 XPOA10 1574 RCR   12      LEFT JUSTIFY THE STRING
761 1253      1352 ? C#0 WPT      STILL A NULL ?
762 1254      1763 GONC  XPOA10 (1252) YES; SKIP IT
763 1255      1074 RCR   2          GET FIRST CHAR BACK TO LEFT END
764 1256      73 GOTO   XPOA30 (1265)
765 1257 XPOA20 1 GOSUB  GOSUB      GET INT(X)
766 1260      0
767 1261      0 XDEF   X<256
768 1262      1074 RCR   2          C[13:12] = SINGLE CHAR SUBSTRING
769 1263      634 PT=   11
770 1264      112 C=0   WPT          C = XX000000000000
771 1265 XPOA30 160 N=C          SAVE THE SUBSTRING IN N
772 1266      1 GOSUB  GOSUB      GET ALPHA HEAD ADDR
773 1267      0
774 1270      0 XDEF   FAHED
775 1271      1014 ?S2=1
776 1272      457 GOC   XPOANF (1337) YES, RETURN -1
777 1273 XPOA40 212 B=A   WPT      SAVE THE STARTING ADDR OF AN
778 1274      260 C=N          ITERATION IN B[3:0] AND COPY THE
779 1275      530 M=C          UNROTATED SUBSTRING FROM N TO M
780 1276 XPOA50 1 GOSUB  GTBYTA     GET NEXT CHAR FROM ALPHA REG
781 1277      0
782 1300      256 AC EX  W          A=CHAR   C=ADDR
783 1301      730 CM EX          C=ROTATED SUBSTR  M=ADDR  A=CHAR
784 1302      1574 RCR   12
785 1303      1434 PT=   1
786 1304      1604 S0=   0
787 1305      1552 ? A#C WPT      MATCH ?
788 1306      27 GOC   XPOA55 (1310) NO
789 1307      1610 S0=   1          REMEMBER CURRENT CHAR MATCH
790 1310 XPOA55 730 CM EX          M=ROTATED SUBSTR  C=CURRENT ADDR
791 1311      256 AC EX  W          A=CURRENT ADDR
792 1312      630 C=N
793 1313      1574 RCR   12          C[1:0]= NEXT CHAR IN SUBSTR
794 1314      1352 ? C#0 WPT      NEXT CHAR IS NULL ?
795 1315      37 GOC   XPOA60 (1320) NO
796 1316      1614 ?S0=1          CURRENT CHAR MATCH ?
797 1317      267 GOC   XPOA80 (1345) YES, WE FOUND IT

```

```

795 1320 XPOA60 116 C=0 W
796 1321 460 LDI
797 1322 5 CON 5
798 1323 34 PT= 3
799 1324 1552 ? A#C WPT REACHED END OF ALPHA ?
800 1325 123 GONC XPOANF (1337) YES, SUBSTR NOT FOUND
801 1326 1614 ?S0=1 CURRENT CHAR MATCH ?
802 1327 57 GOC XPOA65 (1334) YES
803 1330 152 AB EX WPT A[3:0]=START ADDR OF THIS ITERATION
804 1331 1 GOSUB INCADA POINT TO NEXT CHAR IN ALPHA REG
804 1332 0
805 1333 1403 GOTO XPOA40 (1273) START ANOTHER ITERATION
806 1334 XPOA65 1 GOSUB INCADA CONTINUE ON NEXT CHAR
806 1335 0
807 1336 1403 GOTO XPOA50 (1276)
808 1337 XPOANF 116 C=0 W
809 1340 1160 DADD=C ENABLE CHIP 0
810 1341 460 LDI SUBSTRING NOT FOUND
811 1342 221 CONZ 9 1
812 1343 1074 RCR 2 C = 91000000000000
813 1344 133 GOTO XPOA90 (1357) RETURN -1 TO X
814 1345 XPOA80 1 GOSUB GOSUB GET ALPHA HEAD ADDR
814 1346 0
815 1347 0 XDEF FAHED
816 1350 1 GOSUB GOSUB
816 1351 0
817 1352 0 XDEF CNTBY7 COUNT # OF BYTES FROM HEAD
818 1353 XPOAFN 1 GOSUB GOSUB CONVERT BINARY TO DECIMAL
818 1354 0
819 1355 0 XDEF BIN-D
820 1356 316 C=B W
821 1357 XPOA90 1 COLONG NFRX SAVE X TO LAST X, RESULT TO X
821 1360 2

```

*
*

* CNTBY7 - "COUNT BYTES FOR 7 BYTES PER REGISTER" COMPUTES # OF BYTES
* BETWEEN TWO ADDRESSES.

* INPUT : A[3:0] = STARTING ADDR (HIGHER ADDR)

* B[3:0] = ENDING ADDR (LOWER ADDR)

* OUTPUT : A[3:0] = NUMBER OF BYTES BETWEEN THE TWO ADDRESSES (NOTE: IF
* THE TWO ADDRESSES ARE THE SAME, # OF BYTES BETWEEN
* THEM IS ZERO. IF THEY ARE ONE BYTE DIFFERENT, # OF
* BYTES BETWEEN THEM IS ONE, ETC..)

* PT = 3

* USED A[3:0], B[3:0], C, S2,S3, +0 SUB LEVEL

* CNTBYE - SPECIAL ENTRY POINT FOR ADDRESSES IN THE EXTERNAL MEMORY

* OUTPUT : A[3:0] = # OF BYTES

* IF S2=1, MEMORY DISCONTINUITY DETECTED

* USED A[7:0], B[3:0], C, PT=3, S2,S3, +1 SUB LEVEL

*

840 ENTRY CNTBYE

841 ENTRY CNTBY7

*

843 1361 CNTBYE 306 C=B X

844 1362 1566 ? A#C XS TWO ADDRESSES IN THE SAME MODULE?

845 1363 323 GONC CNTBY7 (1415) YES

846 1364 1 GOSUB GOSUB1 ADVANCE TO NEXT MODULE

846 1365 0

847 1366 0 XDEF NXTMDL

848 1367	1014 ?S2=1		MEMORY DISCONTINUITY ?
849 1370	1540 RTN C		YES, ERROR EXIT
850 1371	306 C=B	X	GET END ADDR AGAIN
851 1372	1566 ? A#C	XS	ARE WE IN END MODULE YET ?
852 1373	43 GONC	CTBE30 (1377)	YES
853 1374	460 LDI		
854 1375	356 CON	238	ADD ONE MODULE TO START ADDR
855 1376	23 GOTO	CTBE40 (1400)	
856 1377 CTBE30	106 C=0	X	
857 1400 CTBE40	226 B=A	XS	FORCE TO THE SAME MODULE
858 1401	606 A=A-B	X	A.X= # OF REGS IN NEXT MODULE
859 1402	506 A=A+C	X	ADD 238 IF CROSSED 3 MODULES
860 1403	206 B=A	X	B.X= # OF REGS OVER STARTING MODULE
861 1404	256 C=A	W	
861 1405	416		
862 1406	174 RCR	4	CI[3:0]=STARTING ADDR
863 1407	412 A=C	WPT	
864 1410	1 GOSUB	GOSUB1	
864 1411	0		
865 1412	0 XDEF	ENRGAD	GET END REG ADDR OF THIS MODULE
866 1413	446 A=A+B	X	ADD NEXT MODULE REGS TO START ADDR
867 1414	346 BC EX	X	MOVE END ADDR TO MODULE END REG
868 1415 CNTBY7	34 PT=	3	
869 1416	116 C=0	W	
870 1417	602 A=A-B	PT	
871 1420	43 GONC	CBYT10 (1424)	
872 1421	646 A=A-1	X	
873 1422	642 A=A-1	PT	
874 1423 CBYT05	642 A=A-1	PT	
875 1424 CBYT10	642 A=A-1	PT	
876 1425	37 GOC	CBYT20 (1430)	
877 1426	1046 C=C+1	X	
878	LEGAL		
879 1427	1743 GOTO	CBYT05 (1423)	
880 1430 CBYT20	606 A=A-B	X	
881 1431	352 BC EX	WPT	
882 1432	2 A=0	PT	
883 1433	252 C=A	WPT	
883 1434	412		
884 1435	752 C=C+C	WPT	
885 1436	752 C=C+C	WPT	
886 1437	752 C=C+C	WPT	
887 1440	252 AC EX	WPT	
888 1441	712 A=A-C	WPT	
889 1442	452 A=A+B	WPT	
890 1443	1740 RTN		

* GETKEY - GET KEY *

* WAIT FOR KEY DOWN AND READ THE KEY. RETURN THE KEY CODE TO X AS *

* SHOWN IN THE "ASN" FUNCTION. TIME OUT AFTER 10 SECONDS AND RETURN* *

* ZERO TO X. *

895	ENTRY	GETKEY	
901 1444	231 CON	@231	Y
902 1445	5 CON	@05	E
903 1446	13 CON	@13	K
904 1447	24 CON	@24	T

905	1450	5	CON	005	E
906	1451	7	CON	007	G
907	1452	116	C=0	W	
908	1453	34	PT=	3	SET COUNTER FOR 10 SEC. TIME OUT
909	1454	420	LC	4	
910	1455	GTKE10	1714	CHK KB	
911	1456	57	GOC	GTKE20 (1463)	
912	1457	1156	C=C-1	W	
913	1460	1753	GONC	GTKE10 (1455)	NOT TIME OUT YET
914	1461	116	C=0	W	TIME OUT, RETURN ZERO TO X
915	1462	513	GOTO	GTKE80 (1533)	KEY CODE = 00
916	1463	GTKE20	1040	C=KEYS	READ THE PHYSICAL KEY CODE
917	1464	74	RCR	3	
918	1465	126	C=0	XS	
919	1466	406	A=C	X	A.X= PHYSICAL KEY CODE
920	1467	1634	PT=	0	
921	1470	742	C=C+C	PT	CHECK FOR "OFF" KEY
922	1471	33	GONC	GTKE30 (1474)	NOT "OFF" KEY
923	1472	116	C=0	W	"OFF" KEY CODE = 01
924	1473	123	GOTO	GTKE40 (1505)	
925	1474	GTKE30	460	LDI	CONVERT "USER", "PRGM", "ALPHA"
926	1475	304	CON2	12 4	TO 01,02,03
927	1476	1406	? A<C	X	IS THIS A TOP ROW KEY ?
928	1477	117	GOC	GTKE50 (1510)	NO
929	1500	706	A=A-C	X	NOW TOP ROW KEYS = 00,02,01,00
930	1501	116	C=0	W	
931	1502	320	LC	3	
932	1503	246	AC EX	X	
933	1504	1106	C=A-C	X	NOW TOP ROW KEYS = 00,01,02,03
934	1505	GTKE40	1046	C=C+1	NOW TOP ROW KEYS = 01,02,03,04
935			LEGAL		
936	1506	1074	RCR	2	C= 0X000000000000
937	1507	243	GOTO	GTKE80 (1533)	
938	1510	GTKE50	1746	ASL	SET UP A AND C FOR SUBTRACTING
939	1511	566	A=A+1	XS	3'S FROM THE ACTUAL COLUMN
940	1512	1034	PT=	2	NUMBER TO DETERMINE THE
941	1513	116	C=0	W	LOGICAL COLUMN NUMBER
942	1514	220	LC	2	
943	1515	1152	C=C-1	WPT	PUT '2FF' IN C ('300-1')
944	1516	GTKE60	706	A=A-C	LOOP TO SUBTRACT 3 FROM OLD COL.
945	1517	1773	GONC	GTKE60 (1516)	NO, AND INC. NEW COL. NO.
946	1520	460	LDI		
947	1521	61	CON2	3 1	
948	1522	1542	?A#C	PT	ROW 3?
949	1523	47	GOC	GTKE70 (1527)	NO, SKIP COL. NO. ADJUSTMENT
950	1524	1552	?A#C	WPT	'ENTER' KEY?
951	1525	23	GONC	GTKE70 (1527)	YES, SKIP ADJUSTMENT
952	1526	646	A=A-1	X	CORRECT ROW 3 COL. NUMBERS
953	1527	GTKE70	542	A=A+1	ROW 0 - 7 => ROW 1 - 8
954	1530	252	AC EX	WPT	MOVE KEYCODE TO C
955	1531	74	RCR	3	IN NORMALIZED FORM
956	1532	1046	C=C+1	X	
957	1533	GTKE80	356	BC EX	
958	1534	1	GOSUB	RSTKB	RESET KEY BOARD
959	1535	0			
959	1536	153	GOTO	ROLF10 (1553)	

*

 * RCLFLAG - RECALL FLAGS 0-43 TO THE X REGISTER *
 * THE RESULT IN THE X REGISTER IS THE BINARY BITS OF FLAGS 0-43, *

* IN CHARACTER DATA FORM, SO THE DISPLAY OF X IS MEANINGLESS. *

* STOF20 - RESTORE ALL OR A PORTION OF FLAGS 0-43 FROM THE X & Y *

* REGISTERS. IF X CONTAINS THE BINARY BITS OF THE FLAGS, ALL 43 *

* FLAGS WILL BE RESTORED. IF X CONTAINS THE INDEX OF THE FLAGS TO *

* RESTORE, THE BINARY BITS WILL BE TAKEN FROM THE Y REGISTER. THE *

* INDEX IN X SHOULD BE IN BB.EE FORM, WHERE BB IS THE BEGINNING *

* FLAG NUMBER AND EE IS THE ENDING FLAG NUMBER TO BE RESTORED. *

973		ENTRY	RCLFLG
974		ENTRY	STOFLG

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

976	1537	207	CON	0207
977	1540	1	CON	001
978	1541	14	CON	014
979	1542	6	CON	006
980	1543	14	CON	014
981	1544	3	CON	003
982	1545	22	CON	022
983	1546	RCLFLG	1670	C=REGN 14
984	1547	460	LDI	
985	1550	777	CON	0777
986	1551	74	ROR	3
987	1552	356	BC EX	W
988	1553	RCLF10	1	GOLONG RCL
988	1554		2	

G
A
L
F
L
C
R
GET STATUS REGISTER

C=1FFXXXXXXXXXXXX

990	1555	207	CON	0207
991	1556	1	CON	001
992	1557	14	CON	014
993	1560	6	CON	006
994	1561	17	CON	017
995	1562	24	CON	024
996	1563	23	CON	023

G
A
L
F
O
T
S

997	1564	STOFLG	460	LDI
998	1565		777	CON 0777
999	1566		406	A=C X

GET X

1000	1567		370	C=REGN 3
1001	1570		674	ROR 11
1002	1571	1546	? A#C	X
1003	1572		77	GOC STOF20 (1601)
1004	1573		416	A=C W
1005	1574		1670	C=REGN 14
1006	1575		246	AC EX X
1007	1576		256	AC EX W
1008	1577		1650	REGN=C 14
1009	1600		1740	RTN

DOES X HAVE THE BINARY BITS ?
NO, LET'S CHECK Y
RESTORE ALL FLAGS 0-43
GET STATUS REG

1010	1601	STOF20	270	C=REGN 2
1011	1602		674	ROR 11
1012	1603	1546	? A#C	X

GET Y REGISTER

1013	1604		1	GOLC ERRDE
1013	1605		3	

DOES Y HAVE THE BINARY BITS ?
NO, SAY "DATA ERROR"

1014	1606		204	S5= 0
1015	1607		1010	S2= 1
1016	1610		1	GOSUB GOSUB1
1016	1611		0	
1017	1612		0	XDEF GTIND2
1018	1613		460	LDI
1019	1614		53	CON 43
1020	1615		406	A=C X

DECODE X AS BB.EE

GET INDEX FROM X

1021	1616	260	C=N			C.X= ENDING FLAG NUMBER
1022	1617	1406	? A<C	X		FLAG # <= 43 ?
1023	1620	47	GOC	STOF25 (1624)		NO, SAY "NONEXISTENT"
1024	1621	74	ROR	3		C.X= STARTING FLAG NUMBER
1025	1622	1406	? A<C	X		FLAG # <= 43 ?
1026	1623	33	GONC	STOF30 (1626)		YES, INDEX O.K.
1027	1624	STOF25	1	GOLONG	ERRNE	
1027	1625		2			
1028	1626	STOF30	406	A=C	X	A.X = STARTING FLAG NUMBER
1029	1627		270	C=REGN	2	
1030	1630		674	ROR	11	CI[13:3] = FLAG 0-43
1031	1631	STOF35	646	A=A-1	X	SHIFT OUT THE UNAFFECTED FLAGS
1032	1632		47	GOC	STOF40 (1636)	
1033	1633		756	C=C+C		
1034	1634		0	HOP		
1035	1635		1743	GOTO	STOF35 (1631)	
1036	1636	STOF40	416	A=C	W	SAVE BINARY BITS IN A
1037	1637		260	C=N		
1038	1640		74	ROR	3	C.X= STARTING FLAG NUMBER
1039	1641		346	BC EX	X	SAVE FLAG # IN B.X
1040	1642		460	LDI		
1041	1643		250	CON2	10 8	FUNCTION CODE OF "SFnn"
1042	1644		256	AC EX	W	
1043	1645		756	C=C+C	W	TEST THE FLAG
1044	1646		27	GOC	STOF50 (1650)	THIS FLAG SET
1045	1647		546	A=A+1	X	A.X= FUNCTION CODE OF "CFnn"
1046	1650	STOF50	1150	REGN=C	9	SAVE THE REST OF THE BITS IN REG.9
1047	1651		246	AC EX	X	C.X= FUNCTION CODE
1048	1652		1634	PT=	0	
1049	1653		130	G=C		PUT THE FCN CODE IN G
1050	1654		1	GOSUB	ALLOK	SET OR RESET THE FLAG
1050	1655		0			
1051	1656		260	C=N		
1052	1657		406	A=C	X	A.X= ENDING FLAG NUMBER
1053	1660		74	ROR	3	C.X= LAST FLAG NUMBER
1054	1661		1046	C=C+1	X	C.X= NEXT FLAG NUMBER
1055	1662		1406	? A<C	X	PASSED ENDING FLAG # ?
1056	1663		1540	RTN C		YES, ALL DONE
1057	1664		674	ROR	11	
1058	1665		160	N=C		PUT FLAG NUMBERS BACK IN N
1059	1666		1170	C=REGN	9	GET REMAINING BINARY BITS
1060	1667		1473	GOTO	STOF40 (1636)	

*

*

* POLPS - PROGRAMMABLE CLEAR PROGRAMS *

* THE PROGRAM NAME IS TAKEN FROM THE ALPHA REGISTER. IF ALPHA *

* IS EMPTY, IT WILL DEFAULT TO THE CURRENT PROGRAM. IT ALSO *

* CLEARS ALL OF THE PROGRAMS FOLLOWING THE SPECIFIED PROGRAM. *

*

1070 ENTRY POLPS

*

1072 1670 223 CON 0223 S

1073 1671 20 CON 020 P

1074 1672 14 CON 014 L

1075 1673 3 CON 003 C

1076 1674 20 CON 020 P

1077 1675 POLPS 1 GOSUB GOSUB1

1077 1676 0

1078	1677	0	XDEF	GTPRAD	GET THE PROGRAM ADDR
1079	1700	216	B=A	W	A[3:0]=PR.HD ADDR, A[7:4]=PR.END ADDR
1080	1701	404	S8=	0	CLEAR CHECKSUM FLAG
1081	1702	1	GOSUB	GETPC	GET CURRENT PROG COUNTER ADDR
1081	1703	0			
1082	1704	1104	S9=	0	ASSUME CURRENT PROG WILL STAY
1083	1705	316	C=B	W	
1084	1706	1406	? A<C	X	PC BELOW PROG HEAD ?
1085	1707	67	GOC	LOSTPC (1715)	YES
1086	1710	1546	? A#C	X	ARE THEY AT SAME REG ?
1087	1711	57	GOC	KEEPPC (1716)	NO, PC ABOVE PROG HEAD
1088	1712	242	AC EX	PT	
1089	1713	1402	? A<C	PT	
1090	1714	27	GOC	KEEPPC (1716)	
1091	1715	LOSTPC	1110	S9=	1
1092	1716	KEEPPC	116	C=0	
1093	1717	312	C=B	WPT	GET PROG HEAD ADDR
1094	1720	1150	REGN=C	9	SAVE PROG HEAD ADDR IN REG.9
1095	1721	152	AB EX	WPT	PUT AN "END" TO THE BEGINNING OF THE
1096	1722	1	GOSUB	GOL2	
1096	1723	0			
1097	1724	0	XDEF	RP330	JUMP INTO "GETP" ROUTINE

*

 * X<>F - EXCHANGE THE X REGISTER WITH USER FLAG 0-7 *
 * 0 <= X <= 255 *

		ENTRY	X<>F	
1104				
1106	1725	206	CON	0206 F
1107	1726	76	CON	076 >
1108	1727	74	CON	074 <
1109	1730	30	CON	030 X
1110	1731	1	GOSUB	GOSUB
1110	1732	0		
1111	1733	0	XDEF	X<256
1112	1734	1	GOSUB	GOSUB
1112	1735	0		
1113	1736	0	XDEF	SWPBIT
1114	1737	1670	C=REGN	14
1115	1740	1574	ROR	12
1116	1741	252	AC EX	WPT
1117	1742	1074	ROR	2
1118	1743	1650	REGN=C	14
1119	1744	1	GOSUB	GOSUB
1119	1745	0		
1120	1746	0	XDEF	SWPBIT
1121	1747	1	GOSUB	GOSUB
1121	1750	0		
1122	1751	0	XDEF	BIN-D
1123	1752	316	C=B	
1124	1753	350	REGN=C	3
1125	1754	1	GOLONG	ANNOUT
1125	1755	2		

*
 *

 * SWPBIT - ROUTINE TO REVERSE THE 8 BITS IN A[1:0]
 * INPUT : A[1:0] = THE 8 BITS

```

*   OUTPUT: A[1:0] = SWAPPED 8 BITS
*           PT      = 1
*   USED A.X, C.X, PT  +0 SUBLEVEL
1134      ENTRY SWPBIT

```

```

*
1136 1756 SWPBIT 1134 PT= 9
1137 1757      1746 A SL X
1138 1760      246 AC EX X
1139 1761 STS30 26 A=0 XS
1140 1762      746 C=C+C X
1141 1763      23 GONC STS40 (1765)
1142 1764      566 A=A+1 XS
1143 1765 STS40 246 AC EX X
1144 1766      746 C=C+C X
1145 1767      746 C=C+C X
1146 1770      746 C=C+C X
1147 1771      1706 C SR X
1148 1772      246 AC EX X
1149 1773      1724 DEC PT
1150 1774      1424 ? PT= 1
1151 1775      1643 GONC STS30 (1761)
1152 1776      1740 RTN

```

```

A[2:1] = STATUS BYTE
C[2:1] = REMAINING STATUS BYTE
A[1:0] = SWITCHED STATUS BYTE

```

```

*
*
*
1156      UNLIST
1159      END

```

```

ERRORS : 0

```

SYMBOL TABLE

ADR500	753	-			
ALEN	516	-			
ALEN10	525	-	522		
ALEN20	535	-	524		
ALENG	512	-			
ANUM	774	-			
ANUM10	1002	-	1061		
ANUM20	1017	-	1062		
ANUM25	1042	-	1133	1117	
ANUM30	1046	-	1123	1066	
ANUM40	1120	-	1036		
ANUM41	1123	-	1151		
ANUM42	1131	-	1146		
ANUM44	1133	-	1141		
ANUM45	1134	-	1126		
ANUM47	1142	-	1136		
ANUM48	1147	-	1130		
ANUM60	1106	-	1032		
ANUM70	1063	-	1150	1144	1114 1111
ANUM75	1067	-	1054		
APHEAD	157	-			
AROT10	1175	-	1176		
AROT20	1204	-	1201		
AROT30	1213	-	1234		
AROTAT	1156	-			
AT0X	544	-			
AT0X10	553	-	550		
AT0X15	561	-	602		
AT0X20	562	-	621	617	515
AT0X30	565	-	552		
BIN-D	657	-			
BIN-D3	661	-			
BIND10	677	-	703		
BIND20	704	-	700		
BIND25	706	-			
BIND30	707	-	713		
BIND40	715	-	710		
CBYT05	1423	-	1427		
CBYT10	1424	-	1420		
CBYT20	1430	-	1425		
CLK10	177	-	213		
CLK20	214	-	206	200	
CLK30	217	-	242		
CLK40	241	-	222		
CLKEY5	165	-			
CNTBY7	1415	-	1363		
CNTBYE	1361	-			
CTBE30	1377	-	1373		
CTBE40	1400	-	1376		
FAHD10	724	-	737		
FAHD20	741	-	732		
FAHD30	743	-			
FAHED	726	-			
FLSIZE	575	-			
GETKEY	1452	-			
GTKE10	1455	-	1460		

GTKE20	1463	-	1456						
GTKE30	1474	-	1471						
GTKE40	1505	-	1473						
GTKE50	1510	-	1477						
GTKE60	1516	-	1517						
GTKE70	1527	-	1525	1523					
GTKE80	1533	-	1507	1462					
INT10	633	-	626						
INT20	651	-	642						
INTER	644	-	655						
KEEPPC	1716	-	1714	1711					
LOSTPC	1715	-	1707						
NORM	312	-	306						
NORMEX	303	-							
PASN	400	-							
PASN10	425	-	407						
PASN12	450	-							
PASN15	455	-	453						
PASN20	467	-	465	463					
PASN30	503	-	477						
PASMER	410	-	457	447	443	436	431	427	
PCLPS	1675	-							
POSA	1241	-							
PSIZ10	327	-	302						
PSIZ20	343	-	373	366	364				
PSIZE	253	-							
RCLF10	1553	-	1536						
RCLFLG	1546	-							
SIZE?	610	-							
STOF20	1601	-	1572						
STOF25	1624	-	1620						
STOF30	1626	-	1623						
STOF35	1631	-	1635						
STOF40	1636	-	1667	1632					
STOF50	1650	-	1646						
STOFLG	1564	-							
STS30	1761	-	1775						
STS40	1765	-	1763						
SMP5IT	1756	-							
XX256	627	-							
XX999	622	-							
XXDF	1731	-							
XPOA10	1252	-	1254						
XPOA20	1257	-	1244						
XPOA30	1265	-	1256						
XPOA40	1273	-	1333						
XPOA50	1276	-	1336						
XPOA55	1310	-	1306						
XPOA60	1320	-	1315						
XPOA65	1334	-	1327						
XPOA80	1345	-	1317						
XPOA90	1357	-	1344						
XPOAFN	1353	-							
XPOANF	1337	-	1325	1272	1250				

ENTRY TABLE

ADRS00	753	-
ALEN	516	-
ALENG	512	-
ANUM	774	-
APHEAD	157	-
AROTAT	1156	-
ATOX	544	-
ATOX20	562	-
BIN-D	657	-
BIN-D3	661	-
BIND25	706	-
CLKEYS	165	-
CNTBY7	1415	-
CNTBYE	1361	-
FAHED	720	-
FLSIZE	575	-
GETKEY	1452	-
NORM	312	-
NORMEX	303	-
PASH	400	-
PCLPS	1675	-
POSR	1241	-
PSIZE	253	-
RCLFLG	1546	-
SIZE?	610	-
STOFLG	1564	-
SWPSIT	1756	-
X<256	627	-
X<999	622	-
X<F	1731	-
XPOAFN	1353	-
XPOANF	1337	-

EXTERNAL REFERENCES

ADRS08	722				
ALEN	514	1160			
ALENG	5				
ALENG	4				
ALLOK	1654				
ALLOK	1655				
ANNOUT	1754				
ANNOUT	1755				
ANUM	7				
ANUM	6				
APEREX	326				
APERMC	314	412			
APHEAD	3				
APHEAD	2				
APNDNW	1232				
APNDNW	1233				
APPCR	11				
APPCR	10				
APPREC	13				
APPREC	12				
ARCLRC	15				
ARCLRC	14				
AROTAT	17				
AROTAT	16				
ATOX	21				
ATOX	20				
BODSIN	651				
BODSIN	652				
BIN-D	564	1355	1751		
CHK#S	635				
CHK#S	636				
CLKEYS	25				
CLKEYS	24				
CLRFL	23				
CLRFL	22				
CNTBY7	533	1352			
CRFLAS	27				
CRFLAS	26				
CRFLD	31				
CRFLD	30				
DELCHR	33				
DELCHR	32				
DELREC	35				
DELREC	34				
DIGENT	1044				
DIGENT	1045				
DISEEP	424				
EMDIR	37				
EMDIR	36				
ENRGAD	1412				
ERRDE	646	1604			
ERRDE	647	1605			
ERRNE	1624				
ERRNE	1625				
FAHED	520	546	776	1210	1270 1347
FLSHAP	600				

[illegible]

PACKE	310		
PACKE	311		
PASH	67		
PASH	66		
PCLPS	71		
PCLPS	70		
PKIOAS	244		
PKIOAS	245		
POSA	73		
POSA	72		
POSFL	75		
POSFL	74		
PSIZE	77		
PSIZE	76		
PTBYTA	234	554	1230
PTBYTA	235	555	1231
PURFL	101		
PURFL	100		
RCL	565	1553	
RCL	566	1554	
RCLFLG	103		
RCLFLG	102		
RCLPT	105		
RCLPT	104		
RCLPTA	107		
RCLPTA	106		
REGNOV	111		
REGNOV	110		
REGSWP	113		
REGSWP	112		
RP330	1724		
RSTKB	1534		
RSTKB	1535		
R*SUB	1102		
R*SUB	1103		
SAVEAS	115		
SAVEAS	114		
SAVEP	117		
SAVEP	116		
SAVER	121		
SAVER	120		
SAVERX	123		
SAVERX	122		
SAVEX	125		
SAVEX	124		
SEKPT	127		
SEKPT	126		
SEKPTA	131		
SEKPTA	130		
SIZE?	133		
SIZE?	132		
SIZSUB	273		
SIZSUB	274		
STBT10	1010		
STBT10	1011		
STDFLG	135		
STDFLG	134		
SWPSIT	1736	1746	
UFLINK	217		
UFLINK	220		

X<256 1166 1261 1733
 X<999 255
 X<>F 137
 X<>F 136
 XASH 503
 XASH 504
 XTOA 141
 XTOA 140

NOMAS

NOT MANUFACTURER SUPPORTED
 recipient agrees NOT to contact manufacturer

End of VASM assembly

 VASM ROM ASSEMBLY REV. 6/81A

OPTIONS: L C S

2

FILE SCAP2B

+ 400H

*
*

* NXCHR - ROUTINE TO POINT TO THE NEXT BYTE IN THE EX.MEM
 * INPUT : A[3:0] = CURRENT ADDRESS
 * PT= 3
 * IF S3=1, WHEN IT MOVES TO THE NEXT MODULE, IT WILL UPDATE
 * THE BASE REG IN BOTH MODULE.
 * S3=0, WILL NOT UPDATE THE BASE REGISTERS.

* NXREG - SPECIAL ENTRY POINT TO POINT TO NEXT REG
 * OUTPUT : A[3:0]= NEXT ADDR
 * PT = 3
 * USED A[7:0], C, S2, +1 SUB LEVEL

		ENTRY	NXCHR	
		ENTRY	NXREG	
18				
19				
21	0 NXCHR	1 GOSUB	INCADA	POINT TO NEXT BYTE
21	1	0		
22	2	23 GOTO	NXCH10 (4)	
23	3 NXREG	646 A=A-1	X	POINT TO NEXT REG
24	4 NXCH10	460 LDI		
25	5	100 CON2	4 0	
26	6	1526 ? A#0	XS	IN BASE MODULE ?
27	7	53 GONC	NXCH20 (14)	YES
28	10	460 LDI		
29	11	1 CON2	0 1	
30	12	266 C=A	XS	
30	13	426		
31	14 NXCH20	1546 ? A#C	X	POINTING AT LAST REG OF THE MODULE ?
32	15	1540 RTN C		NO
33	16	1 GOSUB	GOSUB	POINT TO NEXT MODULE
33	17	0		
34	20	0 XDEF	NXTMDL	
35	21	14 ?S3=1		WANT TO UPDATE BASE REG ?
36	22	1640 RTN NC		NO
37	23	1434 PT=	1	

*
 * AT THIS POINT, A.X=C.X=NEF WHERE N=2 OR 3.

*

41	24	112 C=0	WPT	CHANGE C.X= N01 TO POINT TO THE LOWE
42	25	1052 C=C+1	WPT	REGISTER IN THIS MODULE
43	26	1160 DADD=C		

44	27	116 C=0	W	FIX UP NEXT MODULE BASE REG
45	30	534 PT=	6	
46	31	252 C=A	WPT	
46	32	412		
47	33	1574 RCR	12	
48	34	234 PT=	5	
49	35	112 C=0	WPT	
50	36	246 C=A	X	
50	37	406		
51	40	1360 DATA=C		
52	41	574 RCR	6	
53	42	1160 DADD=C		UPDATE CURRENT BASE REG
54	43	70 C=DATA		
55	44	74 RCR	3	
56	45	246 C=A	X	LOAD NEXT MODULE ADDR
56	46	406		
57	47	674 RCR	11	
58	50	1360 DATA=C		
59	51	773 GOTO	NXMDRT (150)	

*

* NXTMDL - GET NEXT MODULE ADDRESS

* INPUT : A.X = REG ADDR WITHIN CURRENT MODULE

* OUTPUT : A.X = HIGHEST REGISTER ADDRESS OF NEXT MODULE

* IF A.X = 0 THERE IS NO NEXT MODULE

* IF S2=1 NEXT MODULE IS NOT CONNECTED TO CURRENT MODULE

* A[7:4] = INPUT OF A[3:0]

* PT = 3

* USED A[7:0], C,S2,PT=3 +0 SUB LEVEL

*

71		ENTRY	NXTMDL	
73	52	NXTMDL	256 C=A	W SAVE A[3:0] TO A[7:4]
73	53		416	
74	54		174 RCR	4
75	55		34 PT=	3
76	56		252 AC EX	WPT
77	57		374 RCR	10
78	60		416 A=C	W
79	61		460 LDI	
80	62		100 CON2	4 0
81	63	1526 ? A#0	XS	CURRENTLY IN BASE MODULE ?
82	64	53 GONC	NXMD01 (71)	YES
83	65	460 LDI		
84	66	1 CON	01	
85	67	266 C=A	XS	
85	70	426		
86	71	NXMD01	1160 DADD=C	
87	72		460 LDI	
88	73	1357 CON2	46 15	C.X= HEX 2EF
89	74	406 A=C	X	A.X = HEX 2EF
90	75	70 C=DATA		GET LOWEST REG OF THE MODULE
91	76	74 RCR	3	C.X= NEXT MODULE ADDR
92	77	1546 ? A#C	X	NEXT MODULE ADDR = 2EF ?
93	100	43 GONC	NXMD05 (104)	YES
94	101	566 A=A+1	XS	A.X = HEX 3EF
95	102	1546 ? A#C	X	NEXT MODULE = 3EF ?
96	103	107 GOC	NXMD10 (113)	NO, NO NEXT MODULE CONNECTED
97	104	NXMD05	1434 PT=	1

*

* AT THIS POINT, A.X=C.X=NEF WHERE N=2 OR 3.

```
*
101 105      112 C=0      WPT      LOAD C.X= N01
102 106      1052 C=C+1  WPT
103 107      1160 DADD=C
104 110      70 C=DATA      GET LOWEST REG OF NEXT MODULE
105 111      1546 ? A#C  X      IS NEXT MODULE INITIALIZED ?
106 112      363 GONC      NXMDRT ( 150) YES, A.X=HIGHEST REG ADDR OF NEXT MOD
```

* NXMDRT SETS PT=3 AND RETURNS

```
*
110 113 HXMD10 1010 S2=      1      DISCONTINUE MEMORY FLAG
111 114      256 C=A
111 115      416
112 116      174 RCR      4      C.X=CURRENT REG ADDR
113 117      426 A=C      XS      A.X = 0EF/2EF/3EF
114 120      460 LDI
115 121      1001 CON      513      C.X = HEX 201
116 122      1526 ? A#0  XS      CURRENTLY IN BASE MODULE ?
117 123      277 GOC      NXMD30 ( 152) NO
118 124      406 A=C      X      A.X=HEX 201
119 125      1160 DADD=C
120 126      1360 DATA=C      WRITE TO NEXT MODULE
121 127      70 C=DATA
122 130      1546 ? A#C  X      REG 201 EXIST ?
123 131      133 GONC      NEWMDL ( 144) YES, PLUG IT IN
124 132      246 AC EX      X      HEX 201 BACK TO C
125 133 HXMD20 1066 C=C+1  XS      C.X = HEX 301
126 134 HXMD25 406 A=C      X      A.X = HEX 301
127 135      1160 DADD=C
128 136      1360 DATA=C
129 137      70 C=DATA
130 140      1546 ? A#C  X      REG 301 EXIST ?
131 141      33 GONC      NEWMDL ( 144) YES
132 142 NONXMD 6 A=0      X      NO NEXT MODULE
133 143      53 GOTO      NXMDRT ( 150)
```

*

*

```
176 144 NEWMDL 1434 PT=      1
```

*

* BOTH JUMPS TO NEWMDL ARE DONE WITH A.X=C.X=N01 WHERE N=2 OR 3,

* SO CHANGES THE 01 TO EF AS REQUIRED, AND C.XS ALREADY

* CONTAINS THE CORRECT VALUE FOR THE A=C X. (SEE ABOVE FOR FALLING IN)

*

```
142 145      1620 LC      14
143 146      1720 LC      15
144 147      406 A=C      X      A.X = 2EF/3EF
145 150 HXMDRT 34 PT=      3
146 151      1740 RTN
147 152 HXMD30 1566 ? A#C  XS      CURRENTLY AT REG 201 ?
148 153      27 GOC      NXMD40 ( 155) NO, CHECK REG 201
149 154      1066 C=C+1  XS      CURRENTLY AT REG 201,CHECK 301
150 155 HXMD40 1160 DADD=C      CHECK NEXT MODULE
151 156      70 C=DATA
152 157      74 RCR      3
153 160      1546 ? A#C  X      IS IT POINTING AT CURRENT ONE ?
154 161      1613 GONC      NONXMD ( 142) YES, NO NEXT MODULE
155 162      460 LDI
156 163      1001 CON      513
157 164      1566 ? A#C  XS      CURRENTLY AT REG 201 ?
```

```

158 165      1477 GOC      NXMD25 ( 134 ) NO, SEE IF REG 201 EXIST
159 166      1453 GOTO     NXMD20 ( 133 ) YES, SEE IF REG 301 EXIST

```

*

```

* ADVREC - ADVANCE ADDR TO POINT TO NEXT RECORD
* INPUT : A[3:0] = ADDR OF CURRENT RECORD LENGTH
*         PT= 3
* OUTPUT : A[3:0] = 1ST BYTE ADDR OF NEXT RECORD
* USED A[7:0], B[3:0], C, S2, PT, +1 SUB LEVEL
* ADVREB - SIMILAR TO "ADVREC" EXCEPT THE RECORD LENGTH IN B.X AS INPUT

```

*

```

169      ENTRY ADVREC
170      ENTRY ADVREB

```

*

```

172 167 ADVREC      1 GOSUB GTBYTA
172 170              0
173 171              126 C=0      XS
174 172              1046 C=C+1  X      C.X= RECORD LENGTH + 1
175 173              346 BC EX   X
176 174 ADVREB      146 AB EX   X      A.X= RECORD LENGTH
177 175              460 LDI
178 176              7 CON      7
179 177              132 C=0     M
180 200 ADVR10      706 A=A-C    X
181 201              37 GOC      ADVR20 ( 204 )
182 202              1072 C=C+1  M
183              LEGAL
184 203              1753 GOTO    ADVR10 ( 200 )
185 204 ADVR20      1006 C=A+C    X
186 205              746 C=C+C    X
187 206              674 RCR      11
188 207              342 BC EX    PT      B[3]= # OF REMAINING BYTES * 2
189 210              146 AB EX    X
190 211              574 RCR      6
191 212              346 BC EX    X      B.X= # OF REGISTERS
192 213              53 GOTO      ADVADR ( 220 )

```

*

```

* ADVADR - ADVANCE ADDRESS IN EXTERNAL MEMORY
* INPUT : A[3:0] = CURRENT ADDRESS
*         B[2:0] = NUMBER OF REGISTERS TO ADVANCE
*         B[3]   = PLUS HALF OF THIS # OF BYTES
* OUTPUT : A[3:0] = ADVANCED ADDRESS
*         PT= 3
*         IF S2 = 1, DISCONTINUITY DETECTED IN MEMORY
*         IF A.X = 0, MEMORY OVERFLOW
* USED A[7:0], B[3:0], C, S2, PT=3, +1 SUB LEVEL
* ADVAD1 - ADVANCE ONE REGISTER IN EXTERNAL MEMORY
* INPUT : A[3:0] = CURRENT ADDRESS
* OUTPUT : SAME AS "ADVADR"
* USED : SAME AS "ADVADR"

```

*

*

```

210      ENTRY ADVADR
211      ENTRY ADVAD1

```

*

```

213 214 ADVAD1      116 C=0      W
214 215              1056 C=C+1  W
215 216              34 PT=      3
216 217              352 BC EX    WPT

```

217	220	ADVADR	1	GOSUB	GOSUB	
217	221		0			
218	222		0	XDEF	ENRGAD	GET LOWEST REG ADDR OF CURRENT MODULE
219	223		1106	C=A-C	X	C.X=# OF REMAINING REGS IN THIS MODULE
220	224		146	AB EX	X	A=# OF REGS, B=CURRENT REG ADDR
221	225		1406	? A<C	X	ENOUGH ROOM IN CURRENT MODULE ?
222	226		113	GONC	ADVA20 (237)	NO, HAS TO GO TO NEXT MODULE
223	227		146	AB EX	X	B=# OF REGS, A=CURRENT REG ADDR
224	230		606	A=A-B	X	A.X=ADVANCED REG ADDR
225	231		34	PT=	3	SEE IF NEED TO ADVANCE BYTES ?
226	232		602	A=A-B	PT	ADVANCE REMAINING BYTES
227	233		1640	RTN NC		ENOUGH BYTES IN THE SAME REG
228	234		642	A=A-1	PT	
229	235		642	A=A-1	PT	POINT TO BYTE ADDR IN NEXT REG
230				LEGAL		
231	236		1563	GOTO	ADVAD1 (214)	ADVANCE ONE MORE REG
232	237	ADVA20	706	A=A-C	X	A.X=REMAINING REGS TO GO -1
233	240		146	AB EX	X	A=START ADDR, B=REGS TO GO
234	241		1	GOSUB	GOSUB	
234	242		0			
235	243		0	XDEF	NXTMDL	GET ADDR OF NEXT MODULE
236	244		1506	? A#0	X	IS THERE A NEXT MODULE ?
237	245		1640	RTN NC		NO
238	246		1523	GOTO	ADVADR (220)	

*

* GTINDX - GET INDEX FROM X IN FORM RRR.BBBEEE

* INPUT : IF S5=1, THE INDEX IS A REGISTER BLOCK SPECIFICATION

* IF S5=0, OTHERWISE

* IF S2=1, WILL DECODE X AS RRR.BBXXXXXXXXXX

* (IF S2=1, ENTRY MUST BE AT GTIND2)

* OUTPUT : IF S5=0, THEN NI(2:0)=EEE,

* NI(5:3)= REG ADDR OF BBB

* NI(8:6)= REG ADDR OF RRR

* C=N

* IF S5=1, THEN NI(2:0)= REG ADDR OF BBB

* NI(5:3)= REG ADDR OF RRR

* IF S2=1, THEN NI(5:3)= RRR

* NI(2:0)= BB

* USED A, B, C, N, S2,S3, PT +2 SUB LEVEL

256		ENTRY	GTINDX
257		ENTRY	GTIND2

*

259	247	GTINDX	1004	S2=	0	DECODE X AS RRR.BBBEEE
260	250	GTIND2	116	C=0		
261	251		1160	DADD=C		
262	252		370	C=REGN	3	
263	253		356	BC EX	W	
264	254		316	C=B	W	
265	255		1	GOSUB	BCDBIN	GET BINARY OF INT(X)
265	256		0			
266	257		160	N=C		
267	260		4	S3=	0	
268	261	GTIX10	1	GOSUB	GOSUB	
269	262		0			
269	263		0	XDEF	GTFRAB	GET 1ST 3 FRAC DIGIT OF X
270	264		406	A=C	X	
271	265		260	C=N		
272	266		674	ROR	11	

273	267	246	AC EX	X	
274	270	160	N=C		
275	271	14	?S3=1		GET 2ND 3 DIGIT OF FRAC(X) YET ?
276	272	37	GOC	GTIX20 (275)	YES
277	273	10	S3=	1	
278	274	1653	GOTO	GTIX10 (261)	
279	275	1014	?S2=1		FOR THE FCN "STOFLAG" ?
280	276	1540	RTN C		YES
281	277	1570	C=REGN	13	REGO TO A.X
282	300	74	RCR	3	
283	301	406	A=C	X	
284	302	260	C=N		CONVERT REGISTER INDICES TO
285	303	74	RCR	3	ABSOLUTE ADDRESSES AND
286	304	1006	C=A+C	X	CHECK THAT THEY EXIST
287	305	74	RCR	3	
288	306	1006	C=A+C	X	
289	307	214	?S5=1		FOR THE FCN "REGMOVE" ?
290	310	73	GONC	GTIX30 (317)	YES
291	311	1	GOSUB	CHKADR	
291	312	0			
292	313	674	RCR	11	
293	314	160	N=C		
294	315	1	GOLONG	CHKADR	
294	316	2			
295	317	474	RCR	8	
296	320	160	N=C		
297	321	1740	RTN		

*

* GTFRA - GET FIRST 3 FRACTION DIGITS OF A NUMBER
 * INPUT : A = THE NUMBER
 * OUTPUT : C.X = BINARY OF THE FRACTION DIGITS
 * B = FRACTION OF THE NUMBER TIMES 1000
 * USED A,B,C,S8 +1 SUB LEVEL
 * GTFRAB - SAME AS "GTFRA" EXCEPT THE INPUT NUMBER IS IN B

*

307		ENTRY	GTFRA
308		ENTRY	GTFRAB

*

310	322	GTFRAB	156	AB EX	W	
311	323	GTFRA	1240	SETDEC		
312	324		1526	? A#0	XS	THE NUMBER < 1 ?
313	325		47	GOC	GTFR20 (331)	YES
314	326	GTFR10	1772	A SL	M	
315	327		646	A=A-1	X	
316	330		1763	GONC	GTFR10 (326)	
317	331	GTFR20	460	LDI		
318	332		3	CON	3	
319	333		1014	?S2=1		DECODE X AS RRR.BB ?
320	334		33	GONC	GTFR30 (337)	NO
321	335		1146	C=C-1	X	
322	336		10	S3=	1	LOOP ONLY ONCE IN GTIND2
323	337	GTFR30	506	A=A+C	X	MULTIPLY BY 1000 (OR 100)
324	340		216	B=A	W	
325	341		256	AC EX	W	
326	342		1140	SETHX		
327	343		1	GOLONG	BODBIN	
327	344		2			

*

*

* ENRGAD - GET LOWEST REGISTER ADDRESS OF CURRENT MODULE
 * INPUT : A[3:0] = REG ADDR IN CURRENT MODULE
 * OUTPUT : C[3:0] = LOWEST REG ADDR OF CURRENT MODULE
 * USED C[3:0] +0 SUB LEVEL
 *

336 ENTRY ENRGAD

338 345 ENRGAD 460 LDI
 339 346 100 CON2 4 0
 340 347 1526 ? A#0 XS CURRENT AT BASE MODULE ?
 341 350 1640 RTN NC YES
 342 351 460 LDI
 343 352 1 CON 01
 344 353 266 C=A XS
 344 354 426
 345 355 1740 RTN

* GTRNA - GET THE PROGRAM FROM ALPHA REGISTER *
 * GTRAD - GET ADDR OF THE PROGRAM WHOSE NAME IS GIVEN IN ALPHA *
 * USED A,B[3:0],C,M,PT,S0-6,S2,S9 +3 SUB LEVEL *
 * REGISTER *
 * GRFLNA - GET THE FILE NAME FROM ALPHA REGISTER *

* THE FILE NAME WILL BE TERMINATED BY EITHER A COMMA OR END OF *
 * ALPHA REGISTER. A FILE NAME HAS TO BE EXACTLY 7 CHARS. IT WILL *
 * BE TRUNCATED OR FILLED WITH TRAILING BLANKS IF IT IS LONGER OR *
 * SHORTER THAN 7 CHARS. THE ADDR OF A DELIMITER WILL BE SAVED *
 * IN REG.8[13:10] EVERY TIME FOR LATER USE *

* USED A,B[3:0],C,M,PT,S6,S5,S3,S1 *
 * GTRNA, GTFLNA USE 1 SUB LEVELS; GTPRAD USES 3 SUB LEVELS *
 * ALNAM2 USES 1 LEVELS IF S1=0, 3 LEVELS IF S1=1 *
 * STATUS USED : *
 * S5 : SET TO "0" AT BEGINNING, WILL BE SET TO "1" FOR ANY *
 * NON-NULL CHAR IN ALPHA REGISTER *
 * S6 : SET TO "0" AT BEGINNING, WILL BE SET TO "1" IF A COMMA *
 * IS ENCOUNTERED *

* S3 : *
 * S3=1 THE FILE WILL BE SHIFTED INTO M FROM LEFT END *
 * S3=0 // RIGHT END *

* S1 : WHEN S3=1 : *
 * IF S1=1, GET THE PROGRAM ADDRESS *
 * S1=0, DON'T TRY TO GET PROGRAM ADDRESS *
 * OUTPUT : M = PROGRAM NAME OR FILE NAME *
 * A[3:0] = PROGRAM HEAD ADDR *
 * A[7:4] = PROGRAM END ADDR *
 * REG.8[13:10] = LAST CHAR ADDR *
 * ERROR EXIT : IF NAME NOT FOUND WILL EXIT TO "NAME ERROR" *

387 ENTRY GTRNA
 384 ENTRY GTPRAD
 385 ENTRY GTFLNA
 386 ENTRY ALNAM2
 387 356 GTRNA 1404 S1= 0
 388 357 23 GOTO GTPRN0 (361)

389	360	GTPRAD	1410	S1=	1	
390	361	GTPRNO	10	S3=	1	
391	362		33	GOTO	ALNAM1 (365)	
392	363	GTFLNA	4	S3=	0	
393	364		1404	S1=	0	
*						
395	365	ALNAM1	116	C=0		
396	366		1160	DADD=C		
397	367		1070	C=REGN	8	PUT ALPHA HEAD ADDR TO REG.8[13:10]
398	370		1334	PT=	13	
399	371		620	LC	6	
400	372		20	LC	0	
401	373		20	LC	0	
402	374		1020	LC	8	
403	375		1050	REGN=C	8	
404	376	ALNAM2	204	S5=	0	CLEAR NULL STRING FLAG
405	377		504	S6=	0	RESET COMMA FLAG
406	400		116	C=0		
407	401		530	M=C		
408	402		1160	DADD=C		
409	403		1070	C=REGN	8	
410	404		34	PT=	3	
411	405		374	RCR	10	
412	406		352	BC EX	WPT	PUT LAST ADDR IN "B"
413	407		1334	PT=	13	
414	410		720	LC	7	
415	411		276	AC EX	S	CHAR COUNTER IN A(13)
416	412		353	GOTO	ALNA20 (447)	
*						
418	413	ALNA10	1	GOSUB	INCADA	GET NEXT BYTE
419	414		0			
419	415		1	GOSUB	GTBYTA	
419	416		0			
420	417		212	B=A	WPT	ADDR TO B
421	420		1434	PT=	1	
422	421		1352	? C#0	WPT	IS IT A LEADING BLANK?
423	422		253	GONC	ALNA20 (447)	YES
424	423		412	A=C	WPT	CHECK FOR COMMA
425	424		460	LDI		
426	425		54	CON	Q54	COMMA
427	426		1552	? A#C	WPT	IS CHAR A COMMA?
428	427		37	GOC	ALNA12 (432)	NO
429	430		510	S6=	1	
430	431		263	GOTO	ALNA30 (457)	
431	432	ALNA12	210	S5=	1	SET NON-NULL STRING FLAG
432	433		1536	? A#0	S	7 CHARS YET ?
433	434		133	GONC	ALNA20 (447)	YES
434	435		676	A=A-1	S	DEC. CHAR COUNT
435	436		630	C=M		
436	437		14	?S3=1		SHIFT TO M FROM LEFT ?
437	440		43	GONC	ALNA16 (444)	NO, FROM RIGHT
438	441		252	AC EX	WPT	SHIFT CHAR TO M FROM LEFT
439	442		1074	RCR	2	
440	443		33	GOTO	ALNA17 (446)	
441	444	ALNA16	1574	RCR	12	SHIFT CHAR TO M FROM RIGHT
442	445		252	AC EX	WPT	
443	446	ALNA17	530	M=C		
444	447	ALNA20	34	PT=	3	
445	450		152	AB EX	WPT	ADDR BACK TO A.
446	451		460	LDI		

447	452	5	CON2	0	5	
448	453	102	C=0	PT		
449	454	1552	? A#C	WPT		END OF "A" REG.
450	455	1367	GOC	ALNA10 (413)		NOT YET
451	456	212	B=A	WPT		ADDR TO B
452	457	ALNA30	630	C=M		
453	460	1536	? A#0	S		7 CHARS YET ?
454	461	143	GONC	ALNA40 (475)		YES
455	462	14	?S3=1			SHIFT FROM LEFT ?
456	463	53	GONC	ALNA35 (470)		NO, FROM RIGHT
457	464	1074	RCR	2		
458	465	ALNA33	530	M=C		
459	466	ALNA34	676	A=A-1	S	
460				LEGAL		
461	467	1703	GOTO	ALNA30 (457)		
462	470	ALNA35	1574	RCR	12	SHIFT BLANK TO M FROM RIGHT
463	471	1434	PT=	1		
464	472	220	LC	2		
465	473	20	LC	0		
466	474	1713	GOTO	ALNA33 (465)		
467	475	ALNA40	1056	C=C+1	W	IS THE FILE NAME = FFFFFFFF ?
468	476	537	GOC	FILNER (551)		IF SO, SAY "NAME ERR"
469	477	34	PT=	3		
470	500	1070	C=REGN	8		
471	501	374	RCR	10		
472	502	312	C=B	WPT		
473	503	174	RCR	4		
474	504	1050	REGN=C	8		STO ADDR IN R(10-13)
475	505	214	?S5=1			ALPHA EMPTY ?
476	506	157	GOC	ALNA60 (523)		NO
477	507	14	?S3=1			LOOKING FOR FILE NAME ?
478	510	413	GONC	FILNER (551)		YES, SAY "NAME ERR"
479	511	1414	?S1=1			ONLY LOOKING FOR PROG NAME ?
480	512	1640	RTN NC			YES, ALPHA EMPTY IS O.K.
481	513	314	?S10=1			CURRENT PROG IN ROM ?
482	514	43	GONC	ALNA55 (520)		NO
483	515	ROMERR	1	GOSUB	ERROR	SAY "ROM"
484	516		0			
484	517		0	XDEF	MSGROM	
485	520	ALNA55	1	GOSUB	GETPC	GET CURRENT PROG COUNTER
485	521		0			
486	522		173	GOTO	ALNA70 (541)	
487	523	ALNA60	1414	?S1=1		WANT TO GET PROG ADDR ?
488	524		1640	RTN NC		NO
489	525		630	C=M		
490	526		1150	REGN=C	9	
491	527		1	GOSUB	ASRCH	SEARCH FOR THE PROG
491	530		0			
492	531		416	A=C	W	SAVE THE ADDR IN A
493	532		1356	? C#0	W	FOUND IT ?
494	533		163	GONC	FILNER (551)	NO, SAY "NAME ERR"
495	534		1114	?S9=1		MICRO CODE FUNCTION ?
496	535		147	GOC	FILNER (551)	YES, SAY "NAME ERR"
497	536		1014	?S2=1		PROGRAM IN ROM ?
498	537		1567	GOC	ROMERR (515)	YES, SAY "ROM"
499	540		34	PT=	3	
500	541	ALNA70	1	GOSUB	FLINKA	FIND PROG END ADDR
500	542		0			
501	543		174	RCR	4	
502	544		416	A=C	W	A[7:4] = PROG END ADDR

503	545	174	ROR	4
504	546	412	A=C	WPT
505	547	1	GOLONG	CPGM10
505	550	2		

A[3:0]= PROG END ADDR
GET PROGRAM HEAD ADDR

NOMAS

NOT MANUFACTURER SUPPORTED
recipient agrees NOT to contact manufacturer

507			ENTRY	FILNER
508	551	FILNER	1	GOSUB GOSUB3
508	552		0	
509	553		0	XDEF APERMG
510	554		16	CON @16
511	555		1	CON @01
512	556		15	CON @15
513	557	1005	CON	@1005
514	560		1	GOSUB GOL3
514	561		0	
515	562		0	XDEF DISERR

N
A
M
E

* GETREC - READ TO THE ALPHA REGISTER, THE CURRENT RECORD STARTING *
* FROM THE CURRENT CHAR IN THE CURRENT WORKING TEXT FILE. *
* THE ALPHA REGISTER WILL BE CLEARED BEFORE THE READ. IF THE *
* END OF THE CURRENT RECORD IN THE FILE IS REACHED, IT WILL *
* CLEAR USER FLAG 17. IF THE END OF THE RECORD IS NOT *
* REACHED, IT WILL READ 24 CHARS AND LEAVE THE CHAR POINTER *
* AT THE LAST CHAR READ AND WILL SET USER FLAG 17. IF IT *
* REACHED THE END OF THE RECORD, THEN THE NEXT READ WILL *
* START FROM THE BEGINNING OF THE NEXT RECORD. *
* ARCLREC - VERY SIMILAR TO "GETREC" EXCEPT IT WILL NOT CLEAR THE *
* ALPHA REGISTER BEFORE THE READ. *

532			ENTRY	GETREC
533			ENTRY	ARCLRC

535	563	203	CON	@203	C
536	564	5	CON	@05	E
537	565	22	CON	@22	R
538	566	24	CON	@24	T
539	567	5	CON	@05	E
540	570	7	CON	@07	G
541	571	GETREC 1210	S7=	1	FLAG FOR CLA
542	572	113	GOTO	GTRC05 (603)	

544	573	203	CON	@203	C
545	574	5	CON	@05	E
546	575	22	CON	@22	R
547	576	14	CON	@14	L
548	577	3	CON	@03	C
549	600	22	CON	@22	R
550	601	1	CON	@01	A
551	602	ARCLRC 1204	S7=	0	FLAG FOR NO CLA
552	603	GTRC05	1	GOSUB	GOSUB3
552	604		0		
553	605		0	XDEF	CURFLT
554	606	210	S5=	1	
555	607	404	S8=	0	FLAG FOR POINT TO NEXT REC WHEN AT T
556	610		1	GOSUB	GOSUB
556	611		0		
557	612		0	XDEF	CUREC#

POINT TO CURRENT POINTER

558	613	1614	?	S0=1		REACHED EOF ?
559	614	43	GONC	GTRC10 (620)	NO	
560	615	1	GOSUB	GOL3		
560	616	0				
561	617	0	XDEF	EOFL		
562	620	GTRC10	116	C=0	W	ENABLE CHIP 0
562	621	1160	DADD=C			
564	622	410	S8=	1		INDICATE NOT YET REACHED END OF REC
565	623	1214	?	S7=1		CLA ?
566	624	53	GONC	GTRC20 (631)	NOT FOR "ARCLREC"	
567	625	1	GOSUB	CLA	YES	
567	626	0				
568	627	6	A=0	X	SAY ALN = 0	
569	630	43	GOTO	GTRC25 (634)		
570	631	GTRC20	1	GOSUB	GOSUB0	GET CURRENT ALPHA LENGTH
570	632	0				
571	633	0	XDEF	ALN		
572	634	GTRC25	460	LDI		
573	635	30	CON	24	C.X = 24	
574	636	246	AC EX	X		
575	637	706	A=A-C	X	A.X = NO. OF CHARS TO FILL ALPHA	
576	640	260	C=N			
577	641	574	ROR	6	C.X= CURRENT CHAR POINTER	
578	642	356	BC EX	W	B.X= CURRENT CHAR POINTER	
579	643	1274	ROR	7	C.X= CURRENT RECORD LENGTH	
580	644	246	AC EX	X	A.X= CUR REC LEN; C.X= CHARS TO FILL	
581	645	606	A=A-B	X	A.X= # OF CHARS LEFT IN RECORD	
582	646	246	AC EX	X	A.X= CHAR TO FILL; C.X= CHARS LEFT	
583	647	1406	?	A<C	X	MORE THAN 24 CHARACTERS LEFT ?
584	650	37	GOC	GTRC30 (653)	YES, WILL NOT REACH EOR THIS TIME	
585	651	404	S8=	0	REACHED EOR	
586	652	246	AC EX	X	A.X= # OF CHARS LEFT IN RECORD	
587	653	GTRC30	146	AB EX	X	B.X= # OF CHARS TO READ
588	654	630	C=N			
589	655	1074	ROR	2	CI9:6J= NEXT CHAR ADDR IN FILE	
590	656	306	C=B	X	C.X= # OF CHARS TO READ	
591	657	530	M=C			
592	660	260	C=N			
593	661	74	ROR	3	C.X = CURRENT RECORD POINTER	
594	662	406	A=C	X		
595	663	1	GOSUB	GOSUB		
595	664	0				
596	665	0	XDEF	UPRCAB	UPDATE RECORD POINTER	
597	666	116	C=0			
598	667	1160	DADD=C			
599	670	1670	C=REGN	14	SET OR CLEAR USER FLAG 17	
600	671	1174	ROR	9		
601	672	1730	CST EX			
602	673	1004	S2=	0		
603	674	414	?	S8=1		
604	675	23	GONC	GTRC40 (677)		
605	676	1010	S2=	1		
606	677	GTRC40	1730	CST EX		
607	700	274	ROR	5		
608	701	1650	REGN=C	14		
* NOW M12:01 = # OF CHARS						
* M19:6J= NEXT CHAR ADDR IN FILE						
611	702	GTRC50	630	C=N		
612	703	1146	C=C-1	X	DONE YET ?	
613	704	1540	RTN C		YES	

614	705	574 RCR	6	
615	706	416 A=C	W	A[3:0]= NEXT CHAR ADDR IN FILE
616	707	1 GOSUB	GTBYTA	
616	710	0		
617	711	346 BC EX	X	SAVE THE BYTE IN B.X TEMP.
618	712	1 GOSUB	GOSUB	POINT TO THE NEXT BYTE IN THE FILE
618	713	0		
619	714	0 XDEF	NXCHR	DOES NOT CHANGE A[13:8]
620	715	256 AC EX	W	
621	716	474 RCR	8	
622	717	530 M=C		
623	720	306 C=B	X	
624	721	1634 PT=	0	
625	722	130 G=C		
626	723	116 C=0		
627	724	1160 DADD=C		
628	725	1 GOSUB	APNDNW	
628	726	0		
629	727	34 PT=	3	
630	730	1523 GOTO	GTRC50 (702)	

*
*

```

*  TXEND - FIND THE ADDR OF THE END OF A TEXT FILE
*  INPUT : N= FILE HEADER
*          IF S6 = 0 - FIND NEW RECORD #
*              1 - POINT TO CURRENT POINTER
*          S5 = 0 - GO ALL THE WAY TO END OF FILE
*              1 - STOP AT CURRENT RECORD #
*          S8 = 0 - GO TO NEXT RECORD WHEN ADVANCING
*                  CHARACTER ADDRESS TO END OF RECORD
*              1 - DON'T POINT TO NEXT RECORD WHEN AT
*                  END OF RECORD
*  OUTPUT : M[11:8]= CURRENT END OF FILE MARK ADDR
*           B[6:4] = LAST RECORD # + 1
*           B[13:10] = CURRENT OR LAST RECORD ADDR
*           IF S1=1 - FILE EMPTY
*           IF S2=1 - MEMORY DISCONTINUITY DETECTED
*  USED A[7:0], B, C, S0,S1,S2, PT=3, +2 SUB LEVEL
*  CUREC# - ROUTINE TO GET THE ADDR OF CURRENT RECORD
*  INPUT : N = FILE HEADER
*          S5 & S8 SAME AS "TXEND"
*  OUTPUT : B[13:10] = CURRENT RECORD ADDR
*           B[9:7]   = CURRENT RECORD LENGTH
*           M[11:8] = END OF FILE MARK ADDR
*           IF S1=1 - FILE EMPTY
*  TOREC# - GET THE ADDR OF A GIVEN RECORD #
*  INPUT : B[6:4] = RECORD #
*          N = FILE HEADER
*  OUTPUT : IF S0=1 - THE GIVEN RECORD # > LAST RECORD #
*           THEN B[13:10]= LAST RECORD ADDR
*              B[9:7]   = LAST RECORD LENGTH
*              M[11:8] = END OF FILE MARK ADDR
*           IF S0=0 - FOUND THE GIVEN RECORD
*           THEN B[13:10]= THE GIVEN RECORD ADDR
*              B[9:7]   = THE GIVEN RECORD LENGTH
*              M[11:8] = CURRENT CHAR ADDR
*  USED A[7:0], B, C, S0,S1,S2 ,PT=3 +2 SUB LEVEL

```

671		ENTRY	CUREC#
672		ENTRY	TOREC#
673		ENTRY	TXTE10

*

675	731	TXTE10	6	A=0	X	
676	732		514	?S6=1		FIND NEW REC # ?
677	733		53	GONC	REC#10 (740)	YES
678	734	CUREC#	510	S6=	1	STOP AT CURRENT RECORD
679	735		260	C=N		
680	736		74	ROR	3	
681	737		406	A=C	X	
682	740	REC#10	316	C=B	W	
683	741		174	ROR	4	
684	742		246	AC EX	X	
685	743		374	ROR	10	
686	744		356	BC EX	W	
687	745	TOREC#	260	C=N		
688	746		374	ROR	10	C.X.= HEADER REG ADDR
689	747		406	A=C	X	
690	750		34	PT=	3	
691	751		2	A=0	PT	
692	752		1	GOSUB	GOSUB	
692	753		0			
693	754		0	XDEF	NXCHR	POINT TO 1ST BYTE IN FILE
694	755		1604	S0=	0	
695	756		1410	S1=	1	
696	757		316	C=B	W	SAVE 1ST REC ADDR IN B[13:10]
697	760		374	ROR	10	
698	761		252	C=A	WPT	
698	762		412			
699	763		174	ROR	4	
700	764		356	BC EX	W	
701	765	TXTE10	1	GOSUB	GTBYTA	GET THE RECORD LENGTH
701	766		0			
702	767		126	C=0	XS	
703	770		346	B=C	X	SAVE THE RECORD LENGTH IN B.X
703	771		306			
704	772		1166	C=C-1	XS	C.XS = F
705	773		1046	C=C+1	X	IS THIS BYTE = "FF" ?
706	774		647	GOC	TXTE60 (1060)	YES, GOT THE END OF TEXT MARK
707	775		316	C=B	W	
708	776		174	ROR	4	
709	777		514	?S6=1		FIND NEW REC # ?
710	1000		37	GOC	TXTE20 (1003)	NO
711	1001		1046	C=C+1	X	
712				LEGAL		
713	1002		433	GOTO	TXTE40 (1045)	
714	1003	TXTE20	1146	C=C-1	X	REACHED CURRENT REC # ?
715	1004		413	GONC	TXTE40 (1045)	NOT YET
716	1005		1404	S1=	0	CURRENT RECORD EXISTS
717	1006		574	ROR	6	SAVE CURRENT REC ADDR IN B[13:10]
718	1007		252	C=A	WPT	
718	1010		412			
719	1011		674	ROR	11	SAVE CURRENT REC LEN IN B[9:7]
720	1012		306	C=B	X	
721	1013		1274	ROR	7	
722	1014		356	BC EX	W	
723	1015		214	?S5=1		STOP AT CURRENT RECORD ?
724	1016		313	GONC	TXTE50 (1047)	NO
725	1017		260	C=N		

726 1020	574 RCR	6	C.X= CURRENT CHAR #
727 1021	246 AC EX	X	
728 1022	1446 ? A<B	X	CURRENT CHAR # < REC LENGTH ?
729 1023	137 GOC	TXTE30 (1036)	YES
730 1024	414 ?S8=1		FOR DELREC OR DELCHRX ?
731 1025	117 GOC	TXTE30 (1036)	YES, DON'T POINT TO NEXT RECORD
732 1026	246 AC EX	X	
733 1027	106 C=0	X	POINT TO BEGINNING OF NEXT RECORD
734 1030	674 RCR	11	
735 1031	1046 C=C+1	X	
736 1032	674 RCR	11	
737 1033	160 N=C		
738 1034	72 B=0	M	FORCE IT TO FOLLOW THE SAME PATH AGAIN
739 1035	123 GOTO	TXTE50 (1047)	
740 1036 TXTE30	246 AC EX	X	
741 1037	1046 C=C+1	X	C.X= CURRENT CHAR # +1
742 1040	346 BC EX	X	
743 1041	1 GOSUB	GOSUB	
743 1042	0		
744 1043	0 XDEF	ADVREB	
745 1044	153 GOTO	TXTE70 (1061)	
746 1045 TXTE40	374 RCR	10	
747 1046	356 BC EX	W	
748 1047 TXTE50	346 CB EX	X	
749 1050	1046 C=C+1	X	PASS CURRENT RECORD
750 1051	346 BC EX	X	
751 1052	1 GOSUB	GOSUB	
751 1053	0		
752 1054	0 XDEF	ADVREB	POINT TO NEXT RECORD LENGTH
753 1055	1014 ?S2=1		MEMORY DISCONTINUITY ?
754 1056	1540 RTN C		YES (ARE THESE TWO STATES NEEDED?)
755 1057	1063 GOTO	TXTE10 (765)	
756 1060 TXTE60	1610 S0=	1	REACHED END OF FILE
757 1061 TXTE70	630 C=M		SAVE END OF FILE MARK ADDR IN MC11: 3
758 1062	474 RCR	8	
759 1063	252 AC EX	WPT	
760 1064	574 RCR	6	
761 1065	530 M=C		
762 1066	1740 RTN		

```

*
*****
* NWREC# - SET NEW RECORD #
*   INPUT : B[6:4]= NEW RECORD #
*           M.X = NEW RECORD LENGTH
*           N= FILE HEADER
* UPREC# - UPDATE CURRENT RECORD #
*   INPUT : B[6:4]= CURRENT RECORD #
*           M.X = ADDITIONAL STRING LENGTH
*           N= FILE HEADER
* UPRCAB - SPECIAL ENTRY POINT
*   INPUT A.X= CURRENT RECORD #
*           B.X = CHARACTER POSITION
*           N= FILE HEADER
*   USED A.X, C      +0 SUB LEVEL

```

```

779          ENTRY  NWREC#
780          ENTRY  UPRCAB
781          ENTRY  STRCAB
782          ENTRY  INREC#

```

*

784	1067	NWREC#	316	C=B	W	
785	1070		174	RCR	4	
786	1071		406	A=C	X	A.X= LAST RECORD # + 1
787	1072	INREC#	630	C=M		
788	1073		346	BC EX	X	B.X'= ALPHA LENGTH
789	1074	STRCAB	260	C=N		CLEAR CHAR POINTER
790	1075		574	RCR	6	C.X = CURRENT CHAR POINTER
791	1076		106	C=0	X	
792	1077		474	RCR	8	
793	1100		160	N=C		
794	1101	UPRCAB	260	C=N		
795	1102		374	RCR	10	C.X = FILE HEADER ADDR
796	1103		1160	DADD=C		
797	1104		1274	RCR	7	C.X = CURRENT RECORD POINTER
798	1105		246	AC EX	X	
799	1106		74	RCR	3	
800	1107		146	AB EX	X	
801	1110		1006	C=A+C	X	EXTEND CHAR POINTER
802	1111		474	RCR	8	
803	1112		1360	DATA=C		
804	1113		1740	RTN		

 * SAVEAS- SAVE ASCII FILE. WRITE A TEXT FILE TO A MASS MEMORY DEVICE *
 * THROUGH HPIL. THE USER HAS TO CREATE A DATA FILE IN THE *
 * MASS MEMORY DEVICE BY THE "CREATE" FUNCTION IN THE HPIL *
 * MODULE. THIS FUNCTION STARTS THE TRANSFER FROM THE TOP OF *
 * BOTH SOURCE AND DESTINATION FILE. IT STOPS UPON REACHING *
 * END OF TEXT OF SOURCE FILE OR REACHING END OF FILE OF DEST.*
 * FILE. THE SOURCE AND DESTINATION FILE NAMES ARE SPECIFIED *
 * IN THE ALPHA REGISTER, SEPARATED BY A COMMA. *

817	ENTRY	SAVEAS	
-----	-------	--------	--

819	1114	223	CON	@223	S	
820	1115	1	CON	@01	A	
821	1116	5	CON	@05	E	
822	1117	26	CON	@26	V	
823	1120	1	CON	@01	A	
824	1121	23	CON	@23	S	
825	1122	1	GOSUB	GOSUB		SEE IF CASSETTE DRIVE THERE ?
825	1123	0				
826	1124	0	XDEF	FNDPIL		
827	1125	1	GOSUB	GOSUB		
827	1126	0				
828	1127	0	XDEF	GTFLNA		GET SOURCE FILE NAME
829	1130	514	256=1			SOURCE NAME = DESTINATION NAME ?
830	1131	1	GOSUBNC	AOUTIN		IF SAME NAME, POINT TO ALPHA HEAD
830	1132	0				
831	1133	76	B=0	S		
832	1134	1	GOSUB	FLSCHX		SEARCH FOR THE DESTINATION FILE
832	1135	0				
833	1136	630	C=N			
834	1137	1356	? C#0	W		FOUND THE FILE IN MASS.MEM ?
835	1140	47	GOC	WRT110 (1144)	YES	
836	1141	1	GOSUB	GOL3		
836	1142	0				
837	1143	0	XDEF	FLNOFN		
838	1144	404	SS=	0		

839	1145	1334	PT=	13	
840	1146	1320	LC	11	
841	1147	436	A=C	S	
842	1150	260	C=N		C.S = FILE TYPE
843	1151	1176	C=C-1	S	TYPE 1 IS ASCII FILE
844	1152	1176	C=C-1	S	
845	1153	57	GOC	WRT130 (1160)	IT IS AN ASCII FILE
846	1154	410	S8=	1	
847	1155	1576	? A#C	S	TYPE 13 IS REGISTER FILE
848	1156	1	GOLC	FLTYER	NOT A REG FILE EITHER
848	1157	3			
849	1160	WRT130	136	C=0	S
850	1161	1076	C=C+1	S	
851	1162	160	N=C		
852	1163	1	GOSUB	CHKPCT	CHECK IF FILE SECURED ?
852	1164	0			
853	1165	414	?S8=1		HAVE TO CHANGE REG FILE TO ASCII FILE
854	1166	1	GOSUB	REWENT	YES
854	1167	1			
855	1170	1	GOSUB	SEKSUB	SEEK TO THE BEGINNING OF THE FILE
855	1171	0			
856	1172	260	C=N		COMPUTE FILE SIZE IN BYTES
857	1173	1716	C SR	W	N[7:4]= FILE SIZE IN RECORDS
858	1174	106	C=0	X	
859	1175	1716	C SR	W	
860	1176	1156	C=C-1	W	
861	1177	1156	C=C-1	W	
862	1200	1150	REGN=C	9	REG.9[3:0]= FILE SIZE IN BYTES
862	1201	1	GOSUB	GOSUB	
862	1202	0			
864	1203	0	XDEF	GTFLNA	GET FILE NAME
865	1204	1	GOSUB	GOSUB3	
865	1205	0			
866	1206	0	XDEF	FSCHT	
867	1207	116	C=0		
868	1210	1160	DAAD=C		
869	1211	1170	C=REGN	9	
870	1212	530	M=C		M[3:0]= DEST. FILE SIZE IN BYTES
871	1213	260	C=N		
872	1214	374	RCR	10	
873	1215	406	A=C	X	A.X= FILE HEADER ADDR
874	1216	2	A=0	PT	A[3:0] = LAST BYTE ADDR IN SOURCE FILE
875	1217	WRT200	1	GOSUB	GOSUB
875	1220	0			
876	1221	0	XDEF	NXCHR	POINT TO NEXT BYTE IN SOURCE
877	1222	1	GOSUB	GTEYTA	
877	1223	0			
878	1224	212	B=A	WPT	
879	1225	404	S8=	0	ASSUME IS EVEN RECORD LENGTH
880	1226	126	C=0	XS	
881	1227	160	N=C		
882	1230	134	PT=	4	
883	1231	12	A=0	WPT	
884	1232	406	A=C	X	SAVE RECORD LENGTH IN A.X TOO
885	1233	1166	C=C-1	XS	
886	1234	1046	C=C+1	X	IS IT THE END OF TEXT MARK ?
887	1235	537	GOC	WRT255 (1310)	YES, ALL DONE
888	1236	1146	C=C-1	X	
889	1237	1730	CST EX		
890	1240	1614	?S0=1		RECORD LENGTH ODD ?

891	1241	33	GONC	WRT210 (1244)	NO
892	1242	410	S8=	1	REMEMBER RECORD LENGTH ODD
893	1243	552	A=A+1	WPT	HAVE TO SENT ONE MORE BYTES
894	1244	WRT210 1730	CST EX		RESTORE STATUS BITS 0-7
895	1245	552	A=A+1	WPT	ADD TWO BYTE FOR RECORD LENGTH
896	1246	552	A=A+1	WPT	
897	1247	630	C=M		C[3:0]= REMAINING FILE SIZE OF DESTIN
898	1250	252	AC EX	WPT	
899	1251	712	A=A-C	WPT	
900	1252	357	GOC	WRT250 (1307)	NOT ENOUGH ROOM TO STORE THIS RECORD
901	1253	252	AC EX	WPT	
902	1254	530	M=C		
903	1255	34	PT=	3	
904	1256	106	C=0	X	
905	1257	1	GOSUB	SDATA0	SEND 1ST HALF OF RECORD LENGTH
905	1260	0			
906	1261	260	C=N		
907	1262	1	GOSUB	SDATA	SEND 2ND HALF OF RECORD LENGTH
907	1263	0			
908	1264	152	AB EX	WPT	
909	1265	WRT220 260	C=N		C.X= REMAINING BYTES TO SEND
910	1266	1146	C=C-1	X	DONE WITH ONE RECORD ?
911	1267	127	GOC	WRT230 (1301)	YES
912	1270	160	N=C		
913	1271	1	GOSUB	GOSUB	
913	1272	0			
914	1273	0	XDEF	NXCHR	
915	1274	1	GOSUB	GTBYTA	
915	1275	0			
916	1276	1	GOSUB	SDATA	
916	1277	0			
917	1300	1653	GOTO	WRT220 (1265)	
918	1301	WRT230 414	?S8=1		RECORD LENGTH ODD ?
919	1302	1153	GONC	WRT200 (1217)	NO
920	1303	106	C=0	X	YES, SEND AN ADDITIONAL BYTE
921	1304	1	GOSUB	SDATA	
921	1305	0			
922	1306	1113	GOTO	WRT200 (1217)	
923	1307	WRT250 410	S8=	1	REACHED END OF DESTINATION FILE
924	1310	WRT255 460	LDI		
925	1311	377	CON	255	
926	1312	1	GOSUB	SDATA0	SEND END OF TEXT MARK
926	1313	0			
927	1314	144	HPL=CH	1	
928	1315	405	CH=	@101	SEND THE LAST FRAME AS END FRAME
929	1316	460	LDI		
930	1317	377	CON	255	
931	1320	1	GOSUB	SDATA	
931	1321	0			
932	1322	1	GOSUB	WAITS	CHECK ERROR
932	1323	0			
*	934		ENTRY	WRT260	
*	936	1324	WRT260 1	GOSUB	UNTCHK
	936	1325	0		
	937	1326	414	?S8=1	REACHED END OF DESTINATION FILE ?
	938	1327	1	GOLNC	NFRPU
	938	1330	2		NO
	939	1331	1	GOSUB	GOL3
					SAY "END OF FL"

```

939 1332      0
940 1333      0 XDEF  EOFI

```

```

*
*****

```

```

*****
* GETAS- READ A ASCII FILE FROM MASS MEM. DEVICE TO EXT. MEMORY      *
* SOURCE AND DEST. FILE NAMES ARE TAKEN FROM THE ALPHA              *
* REGISTER. IT WILL STOP UPON EITHER REACHING END OF TEXT OF        *
* SOURCE OR REACHING END OF FILE OF DESTINATION.                     *
* THE DESTINATION FILE MUST EXIST ALREADY.                           *
*****

```

```

*
952          ENTRY  GETAS
*
954 1334      223 CON  @223      S
955 1335        1 CON  @01      A
956 1336      24 CON  @24      T
957 1337        5 CON  @05      E
958 1340        7 CON  @07      G
959 1341 GETAS    1 GOSUB  GOSUB
960 1342          0
961 1343          0 XDEF  GTFLHA  GET SOURCE FILE NAME
962 1344          1 GOSUB  GOSUB  SEE IF CASSETTE DRIVE THERE ?
963 1345          0
964 1346          0 XDEF  FNDPIL
965 1347      136 C=0  S
966 1350     1076 C=C+1  S
967 1351          LEGAL
968 1352          1 GOSUB  FLSCN
969 1353          0
970 1354          1 GOSUB  SEEKRN  SEEK TO BEGINNING OF THE FILE AND RE
971 1355          0
972 1356          1 GOSUB  UNT     UNTALK IT FOR NOW
973 1357          0
974 1358          1 GOSUB  GOSUB
975 1360          0
976 1361          0 XDEF  GTFLNA
977 1362      514 ?S6=1  DEST. FILE NAME SAME AS SOURCE ?
978 1363      43 GONC  RDT120 (1367) YES
979 1364          1 GOSUB  GOSUB  GET DESTINATION FILE NAME
980 1365          0
981 1366          0 XDEF  ALNAM2
982 1367 RDT120    1 GOSUB  GOSUB3
983 1370          0
984 1371          0 XDEF  FSCN    SEARCH FOR THE TEXT FILE
985 1372      260 C=N
986 1373      374 RCR    10
987 1374     1160 DADD=C  ENABLE FILE HEADER REGISTER
988 1375      406 A=C    X      A.X = FILE HEADER ADDR
989 1376        2 A=0    PT
990 1377        70 C=DATA  ZERO RECORD & CHAR POINTER IN FILE HE
991 1400      132 C=0    M
992 1401     1360 DATA=C
993 1402          1 GOSUB  GOSUB3  COMPUTE DEST. FILE SIZE IN BYTES
994 1403          0
995 1404          0 XDEF  BYTLFT
996 1405      652 A=A-1  WPT
997 1406      256 AC EX  W
998 1407      474 RCR    8

```

990	1410	530	M=C		M[9:6]=FILE SIZE, M[3:0]=LAST CHAR AC
991	1411	1	GOSUB	TALKER	MAKE THE DRIVE A TALKER
991	1412	0			
992	1413	1	GOSUB	SNDDATA	START TO SEND DATA
992	1414	0			
993	1415	630	C=M		
994	1416	412	A=C	WPT	A[3:0]= LAST DEST. BYTE ADDR
995	1417	1	GOSUB	GOSUB	
995	1420	0			
996	1421	0	XDEF	NXCHR	POINT TO NEXT BYTE
997	1422	630	C=M		
998	1423	252	C=A	WPT	
998	1424	412			
999	1425	530	M=C		
1000	1426	1	GOSUB	RDDFRM	READ A BYTE FROM LOOP
1000	1427	0			
1001	1430	1200	HPIL=C	2	ECHO THE DATA FRAME
1002	1431	1346	? C#0	X	END OF TEXT ?
1003	1432	617	GOC	RDT210 (1513)	YES
1004	1433	1	GOSUB	RDDFRM	READ 2ND BYTE OF RECORD LENGTH
1004	1434	0			
1005	1435	1114	?S9=1		ANY ERROR ?
1006	1436	537	GOC	RDT200 (1511)	YES, LET'S QUIT
1007	1437	1200	HPIL=C	2	ECHO THE DATA FRAME
1008	1440	1346	? C#0	X	RECORD LENGTH = 0 ?
1009	1441	1653	GONC	RDT135 (1426)	YES, IGNORE THIS RECORD
1010	1442	160	N=C		SAVE RECORD LENGTH IN N.X
1011	1443	1	GOSUB	PTBYTA	
1011	1444	0			
1012	1445	630	C=M		
1013	1446	574	RCR	6	C[3:0]= DEST. FILE SIZE REMAINING
1014	1447	416	A=C	W	
1015	1450	260	C=N		
1016	1451	404	S8=	0	ASSUME RECORD LENGTH EVEN
1017	1452	1730	CST EX		
1018	1453	1614	?S0=1		
1019	1454	23	GONC	RDT140 (1456)	RECORD LENGTH EVEN
1020	1455	410	S8=	1	REMEMBER RECORD LENGTH ODD
1021	1456	1730	CST EX		
1022	1457	1046	C=C+1	X	ADD ONE BYTE FOR RECORD LENGTH
1023	1460	712	A=A-C	WPT	
1024	1461	307	GOC	RDT200 (1511)	NOT ENOUGH ROOM FOR THIS RECORD
1025	1462	256	AC EX	W	
1026	1463	474	RCR	8	
1027	1464	530	M=C		
1028	1465	412	A=C	WPT	
1029	1466	260	C=N		
1030	1467	1146	C=C-1	X	DONE WITH ONE RECORD ?
1031	1470	137	GOC	RDT160 (1503)	YES
1032	1471	160	N=C		
1033	1472	1	GOSUB	GOSUB	
1033	1473	0			
1034	1474	0	XDEF	NXCHR	POINT TO NEXT BYTE IN DES. FILE
1035	1475	1	GOSUB	RDDFRM	READ A BYTE FROM LOOP
1035	1476	0			
1036	1477	1200	HPIL=C	2	ECHO THE DATA FRAME
1037	1500	1	GOSUB	PTBYTA	
1037	1501	0			
1038	1502	1643	GOTO	RDT150 (1466)	
1039	1503	414	?S8=1		RECORD LENGTH ODD ?

1040	1504	43	GONC	RDT170 (1510)	
1041	1505	1	GOSUB	RDDFRM	IF YES, READ ANOTHER BYTE AND DROP
1041	1506	0			
1042	1507	1200	HPIL=C	2	ECHO THE DATA FRAME
1043	1510	RDT170	1073	GOTO	RDT130 (1417)
1044	1511	RDT200	410	S8=	1
1045	1512	23	GOTO	RDT220 (1514)	SAY REACHED END OF DESTINATION FILE
1046	1513	RDT210	404	S8=	0
1047	1514	RDT220	630	C=M	SAY NOT REACHED END OF DESTINATION FILE
1048	1515	412	A=C	WPT	
1049	1516	460	LDI		
1050	1517	377	CON	255	
1051	1520	1	GOSUB	PTBYTA	WRITE EOR MARK IN DES. FILE
1051	1521	0			
1052	1522	1	GOSUB	GOLONG	UNTALK AND CHECK ERROR
1052	1523	0			
1053	1524	0	XDEF	WRT260	

NOMAS

NOT MANUFACTURER SUPPORTED
recipient agrees NOT to contact manufacturer

* FNDPIL - TEST IF THERE IS A HPIL MODULE PLUG-IN

1058		ENTRY	FNDPIL	
1060	1525	FNDPIL	116	C=0 W
1061	1526	1	GOSUB	CHKCST
1061	1527	0		
1062	1530	1356	? C#0	W
1063	1531	1540	RTN C	

TRY TO CALL A ROUTINE IN HPIL MODULE

IF C=0 ON RETURN, WE KNOW THE MODULE
THE MODULE IS NOT THERE

1065	1532	1	GOSUB	GOSUB3	
1065	1533	0			
1066	1534	0	XDEF	APERMG	
1067	1535	16	CON	016	N
1068	1536	17	CON	017	G
1069	1537	40	CON	040	
1070	1540	4	CON	004	D
1071	1541	22	CON	022	R
1072	1542	11	CON	011	I
1073	1543	26	CON	026	V
1074	1544	1005	CON	01005	E
1075	1545	1	GOSUB	GOL3	
1075	1546	0			
1076	1547	0	XDEF	APEREX	

* POSFL - SEARCH THE CURRENT TEXT FILE STARTING FROM CURRENT *
* POINTER FOR THE SUBSTRING GIVEN IN THE ALPHA REGISTER. *
* RETURN THE RECORD AND CHARACTER POINTER TO X IF THE SUB- *
* STRING IS FOUND AND LEAVE THE POINTER THERE. OTHERWISE, *
* RETURN "-1" TO X AND DON'T CHANGE THE CURRENT POINTER. *

1086		ENTRY	POSFL	
1087	1550	214	CON	0214
1088	1551	6	CON	006
1089	1552	23	CON	023
1090	1553	17	CON	017
1091	1554	20	CON	020
1092	1555	POSFL	1	GOSUB
1092	1556	0		

1093	1557	0	XDEF	CURFLT	SEARCH FOR THE CURRENT TEXT FILE
1094	1560	1	GOSUB	GOSUB0	
1094	1561	0			
1095	1562	0	XDEF	ALEN	GET ALPHA LENGTH
1096	1563	1346	? C#0	X	ALPHA EMPTY ?
1097	1564	47	GOC	APOS10 (1570)	NO
1098	1565	APOSNF	1	GOSUB	GOL0
					RETURN "-1" TO X
1098	1566	0			
1099	1567	0	XDEF	XPOANF	
1100	1570	APOS10	1070	C=REGN	8
1101	1571		1274	RCR	7
					SAVE ALPHA LENGTH IN REGN.8[9:7]
1102	1572		246	AC EX	X
1103	1573		1274	RCR	7
					REG.8[13:10]= ALPHA HEAD ADDR
1104	1574		1050	REGN=C	8
					REG.8[9:7] = ALPHA LENGTH
1105	1575		210	S5=	1
1106	1576		404	S8=	0
					FOR POINT TO NEXT REC WHEN AT END
1107	1577		1	GOSUB	GOSUB
1107	1600		0		
1108	1601		0	XDEF	CUREC#
					GET CURRENT POINTER ADDR
1109	1602	APOS15	1614	?S0=1	
					REACHED END OF FILE ?
1110	1603		1627	GOC	APOSNF (1565)
					YES, RETURN "-1" TO X
1111	1604		316	C=B	W
					CI[9:7] = CURRENT RECORD LENGTH
1112	1605		1274	RCR	7
1113	1606		406	A=C	X
					A.X= CURRENT RECORD LENGTH
1114	1607		260	C=N	
					N= FILE HEADER
1115	1610		574	RCR	6
					C.X= CURRENT CHAR POINTER
1116	1611		706	A=A-C	X
					A.X=# OF CHARS REMAINING IN THIS RECC
1117	1612	APOS16	630	C=M	
					M[11:8]= CURRENT CHAR ADDR
1118	1613		274	RCR	5
					CI[6:3]= CURRENT CHAR ADDR
1119	1614		432	A=C	M
					AI[6:3]=CURR.CHAR.ADDR;A.X=REMAIN CHAF
1120	1615		1274	RCR	7
					CI[13:10]=START ADDR FOR AN ITERATION
1121	1616		356	BC EX	W
					BI[13:10]=START ADDR OF CURRENT ITERAT
*	NOW REG.8[13:10] = ALPHA HEAD ADDR				
*	REG.8[9:7] = ALPHA REGISTER LENGTH				
*	AI[6:3] = CURRENT CHAR ADDR IN CURRENT RECORD				
*	AI[2:0] = # OF CHARS REMAINING IN THIS RECORD				
*	BI[13:10] = STARTING ADDR FOR THIS ITERATION				
1127	1617	APOS20	116	C=0	
1128	1620		1160	DADD=C	
1129	1621		1070	C=REGN	8
1130	1622		534	PT=	6
1131	1623		252	AC EX	WPT
1132	1624		530	M=C	
1133	1625		34	PT=	3
*	NOW M[13:10] = ALPHA HEAD ADDR				
*	M[9:7] = ALPHA LENGTH				
*	M[6:3] = CURRENT CHAR ADDR IN CURRENT RECORD				
*	M[2:0] = # OF CHARS REMAINING IN THIS RECORD				
*					
1139	1626		406	A=C	X
					A.X = # OF REMAINING CHARS
1140	1627		1274	RCR	7
					C.X = ALPHA LENGTH
1141	1630		1406	? A<C	X
					REMAINING CHARS < ALPHA LENGTH ?
1142	1631		537	GOC	APOS70 (1704)
					YES, SKIP TO NEXT RECORD
1143	1632		74	RCR	3
					CI[3:0]= CURRENT ALPHA CHAR ADDR
1144	1633	APOS30	416	A=C	W
1145	1634		1	GOSUB	GTBYTA
					GET NEXT ALPHA CHAR
1145	1635		0		
1146	1636		1	GOSUB	INCADA
					POINT TO NEXT ALPHA CHAR
1146	1637		0		
1147	1640		346	BC EX	X
					SAVE THE ALPHA CHAR IN B.X

1148	1641	256	AC EX	W	
1149	1642	1274	RCR	7	CI3:0J= CURRENT CHAR ADDR IN RECORD
1150	1643	416	A=C	W	
1151	1644	1	GOSUB	GTBYTA	GET NEXT CHAR IN RECORD
1151	1645	0			
1152	1646	256	AC EX	W	
1153	1647	530	M=C		M HAS BEEN ROTATED RIGHT 3 DIGITS
1154	1650	1604	S0=	0	ASSUME NO MATCH THIS TIME
1155	1651	306	C=B	X	
1156	1652	1434	PT=	1	
1157	1653	1552	? A#C	WPT	
1158	1654	27	GOC	APOS40 (1656)	NO MATCH
1159	1655	1610	S0=	1	THIS CHAR MATCH
1160	1656	APOS40 34	PT=	3	
1161	1657	630	C=M		M HAS BEEN ROTATED RIGHT 3 DIGITS
1162	1660	174	RCR	4	C.X= REMAINING ALPHA LENGTH
1163	1661	1146	C=C-1	X	
1164	1662	1346	? C#0	X	IS THIS LAST ALPHA CHAR ?
1165	1663	37	GOC	APOS50 (1666)	NO
1166	1664	1614	?S0=1		LAST CHAR MATCH ?
1167	1665	707	GOC	APOSFD (1755)	YES, WE FOUND IT
1168	1666	APOS50 1614	?S0=1		LAST CHAR MATCH ?
1169	1667	413	GONC	APOS60 (1730)	NO
1170	1670	374	RCR	10	
1171	1671	530	M=C		M HAS BEEN ROTATED RIGHT 3 DIGITS
1172	1672	412	A=C	WPT	
1173	1673	1	GOSUB	GOSUB	
1173	1674	0			
1174	1675	0	XDEF	NXCHR	POINT TO NEXT CHAR IN RECORD
1175	1676	630	C=M		
1176	1677	252	AC EX	WPT	
1177	1700	1274	RCR	7	M HAS BEEN RCR 10
1178	1701	1323	GOTO	APOS30 (1633)	
1179	1702	APOS54 1003	GOTO	APOS15 (1602)	
1180	1703	APOS56 1073	GOTO	APOS16 (1612)	
1181	1704	APOS70 260	C=N		SKIP TO NEXT RECORD & CONTINUE SEARCH
1182	1705	74	RCR	3	
1183	1706	1046	C=C+1	X	INCREMENT RECORD POINTER
1184	1707	74	RCR	3	
1185	1710	106	C=0	X	RESET CHAR POINTER TO ZERO
1186	1711	474	RCR	8	
1187	1712	160	N=C		
1188	1713	630	C=M		
1189	1714	346	BC EX	X	B.X= REMAINING RECORD LENGTH
1190	1715	74	RCR	3	
1191	1716	412	A=C	WPT	CURRENT CHAR ADDR
1192	1717	1	GOSUB	GOSUB	POINT TO NEXT RECORD
1192	1720	0			
1193	1721	0	XDEF	ADVREB	
1194	1722	56	B=0	W	
1195	1723	1604	S0=	0	
1196	1724	1	GOSUB	GOSUB	
1196	1725	0			
1197	1726	0	XDEF	TXTE10	GET NEXT RECORD ADDR & LENGTH
1198	1727	1533	GOTO	APOS54 (1702)	
1199	1730	APOS60 1274	RCR	7	M BACK TO NORMAL
1200	1731	1146	C=C-1	X	DECREMENT REMAINING RECORD LENGTH
1201	1732	530	M=C		
1202	1733	316	C=B		
1203	1734	374	RCR	10	CI3:0J= START ADDR FOR THIS ITERATION

1204 1735	412 A=C	WPT	
1205 1736	1 GOSUB	GOSUB	MOVE ONE CHAR TO START NEXT ITERATION
1205 1737	0		
1206 1740	0 XDEF	NXCHR	
1207 1741	260 C=N		
1206 1742	574 RCR	6	C.X= CURRENT CHAR POINTER
1209 1743	1046 C=C+1	X	
1210 1744	474 RCR	8	
1211 1745	160 N=C		
1212 1746	630 C=M		
1213 1747	474 RCR	8	
1214 1750	252 AC EX	WPT	SAVE NEXT CHAR ADDR IN M[11:8]
1215 1751	574 RCR	6	
1216 1752	530 M=C		
1217 1753	406 A=C	X	A.X= # OF REMAINING CHARS IN RECORD
1218 1754	1273 GOTO	APOS56 (1703)	
1219 1755 APOSFD	260 C=N		SAVE THE CURRENT POINTER TO FILE
1220 1756	374 RCR	10	
1221 1757	1160 DADD=C		
1222 1760	174 RCR	4	
1223 1761	1360 DATA=C		
1224 1762	1 GOSUB	GOL3	
1224 1763	0		
1225 1764	0 XDEF	RCLP30	

*
*
*

1229 UNLIST

ERRORS : 0

SYMBOL TABLE

ADVA20	237	-	226				
ADVAD1	214	-	236				
ADVADR	220	-	246	213			
ADVR10	200	-	203				
ADVR20	204	-	201				
ADVRES	174	-					
ADVREC	167	-					
ALNA10	413	-	455				
ALNA12	432	-	427				
ALNA16	444	-	440				
ALNA17	446	-	443				
ALNA20	447	-	434	422	412		
ALNA30	457	-	467	431			
ALNA33	465	-	474				
ALNA34	466	-					
ALNA35	470	-	463				
ALNA40	475	-	461				
ALNA55	520	-	514				
ALNA60	523	-	506				
ALNA70	541	-	522				
ALNAM1	365	-	362				
ALNAM2	376	-					
APOS10	1570	-	1564				
APOS15	1602	-	1702				
APOS16	1612	-	1703				
APOS20	1617	-					
APOS30	1633	-	1701				
APOS40	1656	-	1654				
APOS50	1666	-	1663				
APOS54	1702	-	1727				
APOS56	1703	-	1754				
APOS60	1730	-	1667				
APOS70	1704	-	1631				
APOSFD	1755	-	1665				
APOSNE	1565	-	1603				
ARCLRC	602	-					
CUREC*	734	-					
ENRGAD	345	-					
FILNER	551	-	535	533	510	476	
FNDPIL	1525	-					
GETAS	1341	-					
GETREC	571	-					
GTFLNA	367	-					
GTER10	326	-	330				
GTER20	331	-	325				
GTER30	337	-	334				
GTERA	323	-					
GTERAB	322	-					
GTIND2	250	-					
GTINDX	247	-					
GTIX10	261	-	274				
GTIX20	275	-	272				
GTIX30	317	-	310				
GTERAD	360	-					
GTERNG	361	-	357				
GTERNA	356	-					

GTRC05	603	-	572		
GTRC10	620	-	614		
GTRC20	631	-	624		
GTRC25	634	-	630		
GTRC30	653	-	650		
GTRC40	677	-	675		
GTRC50	702	-	730		
INREC#	1072	-			
NEWMDL	144	-	141	131	
NONXMD	142	-	161		
NWREC#	1067	-			
NXCH10	4	-	2		
NXCH20	14	-	7		
NXCHR	0	-			
NXMD01	71	-	64		
NXMD05	104	-	100		
NXMD10	113	-	103		
NXMD20	133	-	166		
NXMD25	134	-	165		
NXMD30	152	-	123		
NXMD40	155	-	153		
NXMDRY	150	-	143	112	51
NXREG	3	-			
NXTMDL	52	-			
POSFL	1555	-			
RDT120	1367	-	1363		
RDT130	1417	-	1510		
RDT135	1426	-	1441		
RDT140	1456	-	1454		
RDT150	1466	-	1502		
RDT160	1503	-	1470		
RDT170	1510	-	1504		
RDT200	1511	-	1461	1436	
RDT210	1513	-	1432		
RDT220	1514	-	1512		
REC#10	740	-	733		
ROMERR	515	-	537		
SAVEAS	1122	-			
STRCAB	1074	-			
TORREC#	745	-			
TXTE10	765	-	1057		
TXTE20	1002	-	1000		
TXTE30	1036	-	1025	1023	
TXTE40	1045	-	1004	1002	
TXTE50	1047	-	1035	1016	
TXTE60	1060	-	774		
TXTE70	1061	-	1044		
TXTEN0	731	-			
UPRCAB	1101	-			
WRT110	1144	-	1140		
WRT130	1160	-	1153		
WRT200	1217	-	1306	1302	
WRT210	1244	-	1241		
WRT220	1265	-	1300		
WRT230	1301	-	1267		
WRT250	1307	-	1252		
WRT255	1310	-	1235		
WRT260	1324	-			

ENTRY TABLE

ADVAD1	214	-
ADVADR	220	-
ADVREB	174	-
ADVREC	167	-
ALNAM2	376	-
ARCLRC	602	-
CUREC#	734	-
ENRGAD	345	-
FILNER	551	-
FNDPIL	1525	-
GETAS	1341	-
GETREC	571	-
GTFLNA	363	-
GTFRA	323	-
GTFRAS	322	-
GTIND2	250	-
GTINDX	247	-
GTPRAD	360	-
GTPRNA	356	-
INREC#	1072	-
NWREC#	1067	-
NXCHR	0	-
NXREG	3	-
NXTMDL	52	-
POSFL	1555	-
SAVEAS	1122	-
STRCA8	1074	-
TOREC#	745	-
TXTE10	765	-
TXTEND	731	-
UPRCAS	1101	-
WRT260	1324	-

ADVREB	1043	1054	1721
ALEN	632	1562	

[illegible]

GOSUB0	631	1560						
GOSUB0	632	1561						
GOSUB3	551	603	1204	1367	1402	1532	1555	
GOSUB3	552	604	1205	1370	1403	1533	1556	
GTBYTA	167	415	707	765	1222	1274	1634	1644
GTBYTA	170	416	710	766	1223	1275	1635	1645
GTFLNA	1127	1203	1343	1361				
GTERAB	263							
INCADA	0	413	1636					
INCAD4	1	414	1637					
MSGROM	517							
HFRPU	1327							
HFRPU	1330							
HXCHR	714	754	1221	1273	1421	1474	1675	1740
NXTMDL	20	243						
PTBYTA	1443	1500	1520					
PTBYTA	1444	1501	1521					
RCLP20	1764							
RDDFRM	1426	1433	1475	1505				
RDDFRM	1427	1434	1476	1506				
REWENT	1166							
REWENT	1167							
SDATA	1262	1276	1304	1320				
SDATA	1263	1277	1305	1321				
SDATA0	1257	1312						
SDATA0	1260	1313						
SEEKRN	1353							
SEEKRN	1354							
SEKSUB	1170							
SEKSUB	1171							
SNDATA	1413							
SNDATA	1414							
TALKER	1411							
TALKER	1412							
TXTE10	1726							
UNT	1355							
UNT	1356							
UNTCHK	1324							
UNTCHK	1325							
UPRCAB	665							
WAITS	1322							
WAITS	1323							
WRT260	1524							
XPOANF	1567							

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

End of VASM assembly

VASM ROM ASSEMBLY REV. 6/81A

OPTIONS: L C S

2

FILE SCAP3B

*
*

* SAVEX - SAVE THE X REGISTER TO THE CURRENT WORKING REGISTER FILE *
* AT CURRENT REG POINTER *
* GETX - READ A REGISTER FROM CURRENT WORKING REGISTER FILE AT *
* CURRENT REG POINTER TO THE X REGISTER *

```

*
12          ENTRY  GETXX
13          ENTRY  SAVEX

*
15    0      230 CON   @230      X
16    1        5 CON   @05      E
17    2      26 CON   @26      V
18    3        1 CON   @01      A
19    4      23 CON   @23      S
20    5  SAVEX 1210 S7=    1
21    6        63 GOTO GETX10 ( 14 )

*
23    7      230 CON   @230      X
24   10      24 CON   @24      T
25   11        5 CON   @05      E
26   12        7 CON   @07      G
27   13  GETXX 1204 S7=    0
28   14  GETX10    1 GOSUB GOSUB3
29   15        0
30   16        0 XDEF   CURFLR
31   17      260 C=N
32   20      406 A=C    X      A.X= FILE SIZE
33   21      74 RCR     3
34   22     1046 C=C+1  X      C.X= CURRENT REG # + 1
35   23     1406 ? A<C  X      AT EOF ?
36   24        43 GONC  GETX20 ( 30 ) NOT YET
37   25        1 GOSUB GOL3
38   26        0
39   27        0 XDEF   EOFL
40   30  GETX20 346 B=C    X
41   31      306
42   32      42 B=0     PT
43   33     1274 RCR     7      C.X= FILE HEADER ADDR
44   34     1160 DADD=C
45   35      406 A=C    X
46   36      174 RCR     4
47   37     1360 DATA=C
48   40        1 GOSUB GOSUB1
49   41        0
50   42        0 XDEF   ADVADR
51   43      116 C=0
52   44     1160 DADD=C
53   45      370 C=REGN 3
54   46      256 AC EX   W
55   47     1160 DADD=C
56   50     1214 ?S7=1
57   51      57 GOC     SAVX10 ( 56 )
58   52      70 C=DATA
59   53     356 BC EX   W
60   54        1 GOLONG RCL
61   55        2
62   56  SAVX10 256 AC EX   W
63   57     1360 DATA=C
64   60     1740 RTN

```

```

*
*****
* _SAFHSE - SAVE ALPHA SUB
*   INPUT : CHIP 0 ENABLE
*           CURRENT WORKING FILE IS A TEXT FILE
*           IF S0=0 LEAVE BYTE FOR RECORD LENGTH

```

```

*      S8=1 ADDING CHARS ONLY; NO RECORD LENGTH NEEDED
*      OUTPUT : M[10:8] = CURRENT EOF MARK ADDR OR CURRENT POINTER ADDR
*      B[13:10] = CURRENT OR LAST RECORD ADDR
*      B[9:7] = CURRENT RECORD LENGTH
*      IF ALPHA IS EMPTY, IT WILL EXIT TO "NFRPU"
* SAPHSS - SPECIAL ENTRY POINT WILL NOT SET S6=1
*      INPUT : SAME AS SAPHSB
*      OUTPUT : M[11:8] = CURRENT EOF MARK ADDRESS
*      B[13:10] = FIRST RECORD ADDRESS
*      B[6:4] = NEW LAST RECORD NUMBER
*      BOTH USE A, B, C, M, N, PT, S0-6 +3 SUB LEVEL

```

			ENTRY	SAPHSB	
78			ENTRY	SAPHSS	
79					
81	61	SAPHSB	510	S6=	1
82	62	SAPHSS	1	GOSUB	GOSUB0
82	63		0		
83	64		0	XDEF	ALEN
84	65		1506	? A#0	X
85	66		1	GOLNC	NFRPU
85	67		2		
86	70		1070	C=REGN	8
87	71		106	C=0	X
88	72		574	ROR	6
89	73		246	AC EX	X
90	74		530	M=C	
91	75		1	GOSUB	GOSUB3
91	76		0		
92	77		0	XDEF	CURFLT
93	100		1	GOSUB	GOSUB1
93	101		0		
94	102		0	XDEF	TXTEAD
95	103		260	C=M	
96	104		346	BC EX	X
97	105		42	B=0	PT
98	106		374	ROR	10
99	107		406	A=C	X
100	110		2	A=0	PT
101	111		1	GOSUB	GOSUB1
101	112		0		
102	113		0	XDEF	ADVADR
103	114		212	B=A	WPT
104	115		630	C=M	
105	116		474	ROR	8
106	117		412	A=C	WPT
107	120		1	GOSUB	GOSUB0
107	121		0		
108	122		0	XDEF	CNTBYE
109	123		630	C=M	
110	124		102	C=0	PT
111	125		414	?S8=1	
112	126		27	GOC	SAPH10 (130)
113	127		1056	C=C+1	
114	130	SAPH10	1412	? ACC	WPT
115	131		1640	RTN NC	
116	132		1	GOSUB	GOL3
116	133		0		
117	134		0	XDEF	EOFL

```

ALPHA EMPTY ?
YES, DON'T DO ANYTHING

SAVE ALPHA HEAD ADDR IN M[7:4]
MAKE SURE NO NAME = M

SAVE ALPHA LENGTH IN M[2:0]

SEARCH FOR THE CURRENT FILE

B.X= FILE SIZE IN REGS

A.X= FILE HEADER ADDR

[3:0]= END OF FILE MARK ADDR

COMPUTE # OF BYTES BETWEEN
GET ALPHA LENGTH

FOR CHAR INSERT OR APPEND ?
YES
RESERVE BYTE FOR RECORD LENGTH
ENOUGH ROOM
YES

SAY "END OF FL"

```

```

*
*
*****
* SAVEP - SAVE PROGRAM TO EXTERNAL MEMORY
* THE PROGRAM/FILE NAME IS TAKEN FROM THE ALPHA REGISTER
* THE FILE NAME CAN BE ANY ALPHA-NUMERICAL LETTERS UP TO 7 CHARS.
* THE ALPHA REGISTER CAN BE IN ANY OF THE FOLLOWING FORMS :
* "ABCDEFGH" - SAVE PROGRAM "ABCDEFGH" TO FILE "ABCDEFGH"
* ",ABCDEFGH" - SAVE CURRENT PROGRAM TO FILE "ABCDEFGH"
* "ABCDEFGH,HIJKLMN" - SAVE PROGRAM "ABCDEFGH" TO FILE "HIJKLMN"
* IF THE FILE ALREADY EXISTS, THE NEW PROGRAM WILL REPLACE THE OLD
* ONE. IF THE FILE DOES NOT EXIST YET, IT WILL CREATE A NEW FILE
* AND SAVE THE PROGRAM IN IT.
*****

```

```

*
134 ENTRY SAVEP
*
136 135 220 CON 0220 P
137 136 5 CON 005 E
138 137 26 CON 026 V
139 140 1 CON 001 A
140 141 23 CON 023 S
141 142 SAVEP 1 GOSUB GOSUB1
141 143 0
142 144 0 XDEF GTPRAD GET PROGRAM ADDRESS
* NOW A[3:0] = PROG HEAD ADDR
* A[7:4] = PROG END ADDR
145 145 256 C=A W
145 146 416
146 147 474 RCR 8
147 150 356 B=C W SAVE PROG END & HEAD ADDR IN B[13:6]
147 151 316
148 152 374 RCR 10
149 153 352 BC EX WPT B[3:0] = PROG END ADDR
150 154 1 GOSUB GOSUB0
150 155 0
151 156 0 XDEF CNTBY7 COMPUTE PROG LENGTH IN BYTES
152 157 546 A=A+1 X
153 160 546 A=A+1 X A.X = PROG LENGTH
154 161 206 B=A X SAVE THE PROG LENGTH IN B.X
155 162 546 A=A+1 X ONE BYTE FOR CHECKSUM
156 163 116 C=0 W
157 164 460 LDI
158 165 7 CON 7
159 166 546 A=A-1 X COMPUTE # OF REGS REQUIRED
160 167 SVPR10 1072 C=C+1 M
161 170 706 A=A-C X
162 171 1763 GONC SVPR10 ( 167 )
163 172 106 C=0 X
164 173 1160 DADD=C
165 174 74 RCR 3 C.X = PROG SIZE IN REGS
166 175 356 BC EX W SAVE PROG SIZE IN B.X
167 176 674 RCR 11 C[5:3] = PROG LENGTH IN BYTES
168 177 306 C=B X
169 200 1150 REGN=C 9
* REG.9 [12:9] = PROG HEAD ADDR
* REG.9 [5:3] = PROG LENGTH IN BYTES
* REG.9 [2:0] = PROG SIZE IN REGISTERS
*
174 201 1404 S1= 0

```

```

175 202          4 S3=      0
176 203          514 ?S6=1
177 204          47 GOC      SVPR25 ( 210 )
178 205          1 GOSUB     GOSUB0
178 206          0
179 207          0 XDEF      ADR608
180 210 SVPR25    1 GOSUB     GOSUB1
180 211          0
181 212          0 XDEF      ALNAM2
182 213          10 S3=      1
183 214          1 GOSUB     GOSUB3
183 215          0
184 216          0 XDEF      EFLSCH
185 217          1614 ?S0=1
186 220          253 GONC     SVPR40 ( 245 )
187 221          260 C=N
188 222          1176 C=C-1   S
189 223          1376 ? C#0   S
190 224          57 GOC       DUFFER ( 231 )
191 225          1 GOSUB     GOSUB3
191 226          0
192 227          0 XDEF      PUFLSB
193 230          1123 GOTO     SAVEP ( 142 )

*
195              ENTRY      DUFFER

*
197 231 DUFFER    1 GOSUB     GOSUB3
197 232          0
198 233          0 XDEF      APERMG
199 234          4 CON        004
200 235          25 CON        025
201 236          20 CON        020
202 237          40 CON        040
203 240          6 CON        006
204 241          1014 CON      01014
205 242          1 GOSUB     GOL3
205 243          0
206 244          0 XDEF      APEREX

* NOW B.X = # OF REGISTERS STILL AVAILABLE
* REG.9[2:0] = REQUIRED FILE SIZE IN REGISTERS
205 245 SVPR40    116 C=0
210 246          1160 DADD=C
211 247          1170 C=REGN  9
212 250          406 A=C      X
213 251          546 A=A+1    X
214 252          1446 ? A<B   X
215 253          47 GOC       SVPR42 ( 257 )
216 254          1 GOSUB     GOL0
216 255          0
217 256          0 XDEF      NORM
218 257 SVPR42    356 BC EX    W
219 260          116 C=0      W
220 261          234 PT=      5
221 262          312 C=B      WPT
222 263          34 PT=       3
223 264          1076 C=C+1    S
224 265          160 N=C
225 266          256 AC EX     W
226 267          474 RCR      S
227 270          406 A=C      X

IS THERE A COMMA IN ALPHA REG ?
YES, FILE NAME IS THE NAME 2
NO, USE THE PROG NAME AS FILE NAME

SEARCH FOR THE FILE
FILE FOUND ?
NO

IS THIS FILE A PROG FILE ?
NO, SAY "DUP FL"
PURGE THE FILE OR SAY "NO ROOM"

D
U
P
F
L

A.X = FILE SIZE + 1
ENOUGH ROOM FOR A NEW FILE ?
YES
SAY "NO ROOM"

N= NEW FILE HEADER
C[10:8]= 1ST AVAILABLE REG ADDR

```

228	271	2	A=0	PT	
229	272	1160	DADD=C		ENABLE FILE NAME REG
230	273	630	C=N		GET FILE NAME
231	274	1360	DATA=C		
232	275	1	GOSUB	GOSUB1	
232	276	0			
233	277	0	XDEF	NXREG	POINT TO NEXT REG
234	300	246	C=A	X	
234	301	406			
235	302	1160	DADD=C		ENABLE FILE HEADER REG
236	303	260	C=N		CI[5:3]= PROG LENGTH IN BYTES
237	304	1360	DATA=C		
238	305	1172	C=C-1	M	CI[5:3]= # OF BYTES -1
239	306	106	C=0	X	INIT. CHECKSUM
240	307	160	N=C		N.M=BYTE COUNT, N.X=RUNNING CHKSUM
241	310	316	C=B	W	CI[12:9]= PROG HEAD ADDR
242	311	252	AC EX	WPT	CI[3:0]= LAST STORE ADDR
* NOW N.M= BYTE COUNT, N.X = RUNNING CHECKSUM					
* CI[12:9]= LAST SOURCE ADDR, CI[3:0]= LAST DEST ADDR					
*					
246	312	SVPR45	1174	ROR	9
247	313	416	A=C	W	
248	314	1	GOSUB	NXBYTA	GET NEXT BYTE FROM PROG
248	315	0			
249	316	346	BC EX	X	SAVE IT IN B.X
250	317	256	AC EX	W	
251	320	274	ROR	5	
252	321	416	A=C	W	A[3:0]= LAST STORE ADDR
253	322	1	GOSUB	GOSUB1	
253	323	0			
254	324	0	XDEF	NXCHR	POINT TO NEXT CHAR IN EX.MEM
255	325	306	C=B	X	
256	326	1	GOSUB	PTBYTA	LEAVES CI[1:0] IN BI[1:0]
256	327	0			
257	330	260	C=N		
258	331	246	AC EX	X	UPDATE CHECKSUM
259	332	446	A=A+B	X	
260	333	246	AC EX	X	
261	334	1172	C=C-1	M	ALL DONE ?
262	335	47	GOC	SVPR60 (341)	YES, GO TO WRITE EOF MARK
263	336	160	N=C		
264	337	256	AC EX	W	
265	340	1523	GOTO	SVPR45 (312)	
266	341	SVPR60	160	N=C	
267	342	1	GOSUB	GOSUB1	WRITE THE CHECKSUM TO FILE
267	343	0			
268	344	0	XDEF	NXCHR	
269	345	260	C=N		
270	346	1	GOSUB	PTBYTA	
270	347	0			
271	350	1	GOSUB	GOSUB1	
271	351	0			
272	352	0	XDEF	NXREG	POINT TO END OF FILE MARK
273	353	246	AC EX	X	
274	354	1160	DADD=C		
275	355	116	C=0	W	
276	356	1156	C=C-1	W	
277	357	1360	DATA=C		
278	360	1	GOLONG	NFRPU	
278	361	2			

*

```

*****
* GETP  -READ PROGRAM FROM EXTERNAL MEMORY AND REPLACE THE LAST *
*        PROGRAM IN MAIN MEMORY                                *
* GETSUB - GET SUBROUTINE. SAME AS "GETP" EXCEPT IT APPENDS THE *
*        PROGRAM TO THE END OF MAIN MEMORY.                    *
*
* INPUT : ALPHA REG = FILE NAME                                *
* THE KEY ASSIGNMENT WILL BE HONORED IF THE FUNCTION IS EXECUTED *
* IN USER MODE                                                *
*****

```

*

```

291          ENTRY GETP
292          ENTRY GETSUB

```

*

```

294 362          202 CON    @202          B
295 363          25 CON    @25           U
296 364          23 CON    @23           S
297 365          24 CON    @24           T
298 366          5 CON     @05           E
299 367          7 CON     @07           G
300 370 GETSUB 1104 S9=    0             REMEMBER THAT IT IS GETSUB
301 371          63 GOTO   RP100 ( 377)

```

*

```

303 372          220 CON    @220          P
304 373          24 CON    @24           T
305 374          5 CON     @05           E
306 375          7 CON     @07           G
307 376 GETP    1110 S9=    1
308 377 RP100    1 GOSUB   GOSUB1        GET FILE NAME FROM THE ALPHA REGIST !
309 400          0
309 401          0 XDEF    GTFLNA
310 402          1 GOSUB   GOSUB3
310 403          0
311 404          0 XDEF    FSCHP          SEARCH FOR THE PROG IN EX.MEM
312 405          1114 ?S9=1              GETSUB ?
313 406          57 GOC     RP150 ( 413) NO, IT IS GETP
314 407          1 GOSUB   GTFEND        GET FINAL END ADDR
314 410          0
315 411          2 A=0      PT
316 412          333 GOTO   RP245 ( 445)
317 413 RP150    1314 ?S13=1              RUNNING ?
318 414          77 GOC     RP200 ( 423) YES, SEE IF IN LAST PRGM
319 415          116 C=0
320 416          1160 DADD=C
321 417          1670 C=REGN 14
322 420          1530 ST=C
323 421          114 ?S4=1              SINGLE STEPPING ?
324 422          173 GONC    RP240 ( 441) NO, CHANGE PC
325 423 RP200    314 ?S10=1              ARE WE IN ROM ?
326 424          147 GOC     RP220 ( 440) YES, DON'T CHANGE PC
327 425          1 GOSUB   FLINKP        FIND PRGM END
327 426          0
328 427          474 RCR     8
329 430          412 A=C      WPT          A[3:0]=CURRENT PRGM END
330 431          1 GOSUB   INCADA
330 432          0
331 433          1 GOSUB   NXBYTA        GET 3RD BYTE OF "END"
331 434          0
332 435          1730 CST EX

```

333	436	214	?S5=1		IS THIS FINAL END ?
334	437	27	GOC	RP240 (441)	YES, CHANGE PC
335	440	1104	S9=	0	REMEMBER DON'T CHANGE PC
336	441	1	GOSUB	GTFEND	GET FINAL END ADDR
336	442	0			
337	443	1	GOSUB	CPCM10	GET CURRENT PRGM HEAD
337	444	0			
338	445	212	B=A	WPT	B[3:0]=STARTING ADDR
339	446	1	GOSUB.	MENLFT	COMPUTE AVAILABLE REGS
339	447	0			
340	450	406	A=C	X	A.X=# OF UNUSED MEM REGS
341	451	116	C=0	W	
342	452	1160	DADD=C		
343	453	312	C=B	WPT	
344	454	1150	REGN=C	9	
345	455	530	M=C		M[3:0]= PROG HEAD ADDR
346	456	1570	C=REGN	13	
347	457	246	AC EX	X	
348	460	706	A=A-C	X	
349	461	2	A=0	PT	
350	462	152	AB EX	WPT	
351	463	1	GOSUB	GOSUB0	COMPUTE TOTAL AVAILABLE BYTES
351	464	0			
352	465	0	XDEF	CNTBY7	
353	466	260	C=N		
354	467	136	C=0	S	
355	470	74	ROR	3	C.X = PROG SIZE IN BYTES
356	471	1406	? A<C	X	ENOUGH ROOM ?
357	472	43	GONC	RP250 (476)	

*
*

360	473	1	GOSUB	GOL0	
360	474	0			
361	475	0	XDEF	NORMEX	SAY "NO ROOM" OR "PACK, TRY AGAIN"
362	476	1732	C SR	M	C[9:6]= LAST BYTE ADDR IN EX.MEM
363	477	160	N=C		N[9:6]= LAST BYTE ADDR IN EX.MEM
364	500	4	S3=	0	DON'T UPDATE BASE REG IN NXCHR

* NOW M[3:0] = LAST BYTE ADDR IN MAIN MEMORY
 * M[6:4] = RUNNING CHECKSUM
 * N[9:6] = LAST BYTE ADDR IN EX.MEM
 * N.X = BYTE COUNT
 *

370	501	260	C=N		
371	502	574	ROR	6	
372	503	416	A=C	W	A[3:0]= LAST BYTE ADDR IN EX.MEM
373	504	1	GOSUB	GOSUB1	POINT TO NEXT BYTE
373	505	0			
374	506	0	XDEF	NXCHR	
375	507	1	GOSUB	GTBYTA	
375	510	0			
376	511	256	AC EX	W	
377	512	474	ROR	8	
378	513	1146	C=C-1	X	IS THIS THE CHECKSUM BYTE ?
379	514	177	GOC	RP310 (533)	YES
380	515	160	N=C		
381	516	630	C=M		
382	517	256	AC EX	W	
383	520	1	GOSUB	INCADA	
383	521	0			
384	522	1	GOSUB	PTBYTA	LEAVES C[1:0] IN B[1:0]

384	523	0			
385	524	256	AC EX	W	
386	525	174	RCR	4	UPDATE CHECKSUM
387	526	146	AB EX	X	
388	527	1006	C=A+C	X	
389	530	374	RCR	10	
390	531	530	M=C		
391	532	1473	GOTO	RP300 (501)	
392	533	404	S8=	0	ASSUME CHECKSUM WILL MATCH
393	534	630	C=M		
394	535	174	RCR	4	C.X= ACCUMULATED CHECKSUM
395	536	426	A=C	XS	
396	537	1546	? A#C	X	CHECKSUM MATCH ?
397	540	303	GONC	CLEANM (570)	YES
398	541	410	S8=	1	SET CHECKSUM ERROR FLAG
399	542	116	C=0		
400	543	1160	DADD=C		
401	544	630	C=M		
402	545	416	A=C		AI[3:0]= LAST BYTE ADDR IN MAIN MEM
403	546	1570	C=REGN	13	
404	547	1406	? A<C	X	
405	550	33	GONC	RP320 (553)	
406	551	246	AC EX	X	
407	552	1550	REGN=C	13	
408	553	1170	C=REGN	9	
409	554	416	A=C	W	AI[3:0]= BEGINNING OF THIS PROGRAM
*					
411			ENTRY	RP330	
*					
413	555	1	GOSUB	INCADA	POINT TO 1ST BYTE OF THE "END"
413	556	0			
414	557	460	LDI		PUT AN "END" TO THE BEGINNING OF THE
415	560	300	CON2	12 0	
416	561	1	GOSUB	PTBYTA	WRITE HEX "C0" TO THE 1ST BYTE
416	562	0			
417	563	1	GOSUB	INCAD2	POINT TO 3RD BYTE OF THE "END"
417	564	0			
418	565	106	C=0	X	
419	566	1	GOSUB	PTBYTA	PUT "00" TO THE 3DR BYTE OF THE "END"
419	567	0			
*					
421	570	CLEANM	116	C=0	
422	571		530	M=C	
423	572		1160	DADD=C	
424	573		1170	C=REGN	9
425	574		416	A=C	GET STARTING ADDR
426	575		1114	?S9=1	RUNNING OR SINGLE STEPPING
427	576		43	GONC	RP405 (602)
428	577		304	S10=	0
429	600		1	GOSUB	PUTPCX
429	601		0		
430	602	RP405	1570	C=REGN	13
431	603		74	RCR	3
432	604		1546	? A#C	X
433	605		173	GONC	RP410 (624)
434	606		1	GOSUB	GTBYTA
434	607		0		
435	610		1730	CST EX	TO LOCAL END
436	611		204	S5=	0
437	612		1730	CST EX	

```

438 613          1 GOSUB PTBYTA
438 614          0
439 615          1 GOSUB DECADA      POINT TO 1ST OF PREV. END
439 616          0
440 617          1 GOSUB DECADA
440 620          0
441 621        256 C=A
441 622        416
442 623        530 M=C      SAVE PREV. LINK ADDR IN M
443 624 RP410    116 C=0
444 625        1160 DADD=C
445 626        1170 C=REGN 9      RESTORE PGMHD TO A
446 627        416 A=C      W
447 630        1670 C=REGN 14
448 631        1274 RCR      7      GET USER MODE FLAG
449 632        1530 ST=C
450 633        1204 S7=      0
451 634        340 SEL Q
452 635        1334 PT=      13
453 636        240 SEL P
454 637        1334 PT=      13
455 640        1420 LC      12
456 641        1520 LC      13
457 642        422 A=C      PQ
458 643 RP425    34 PT=      3
* START SET LINK HERE - LOOK AT NEXT BYTE SEE IF AN ALBL
*
461 644 RP430    212 B=A      WPT
462 645          1 GOSUB NXBYTA
462 646          0
463 647        1074 RCR      2
464 650        1534 PT=      12
465 651        1362 ? C#0    PQ      NULL ?
466 652        1713 GONC     RP425 ( 643 ) YES
467 653        1576 ? A#C    S      AN END OR ALBL ?
468 654        417 GOC       RP450 ( 715 ) NO
469 655        1402 ? A<C    PT      X<>N OR LBL.NN ?
470 656        377 GOC       RP450 ( 715 ) YES
471 657          34 PT=      3
* SEE IF IT IS AN END
*
474 660          212 B=A      WPT
475 661          1 GOSUB INCAD2
475 662          0
476 663          1 GOSUB GTBYTA
476 664          0
477 665        1574 RCR      12
478 666        152 AB EX     WPT
479 667        1042 C=C+1    PT
480 670        323 GONC     RP470 ( 722 ) IT IS AN END
* RECOMPUTE THE LINK FOR AN ALBL
*
482 671          630 C=N
484 672        252 AC EX     WPT
485 673          1 GOSUB GENLNK
485 674          0
486 675        412 A=C      WPT
487 676        530 M=C
488 677        1614 S50=1      ARE WE IN USER MODE ?
489 700        127 GOC       RP440 ( 712 ) YES, DON'T CLEAR THE KEY CODE

```

```

490 701          1 GOSUB  INCAD2
490 702          0
491 703          1 GOSUB  INCADA          POINT TO KEY CODE
491 704          0
492 705          106 C=0    X
493 706          1 GOSUB  PTBYTA          CLEAR THE KEY CODE
493 707          0
494 710          630 C=M
495 711          412 A=C    WPT
496 712 RP440    1 GOSUB  DECADA          POINT TO 1 BYTE BEFORE ALBL
496 713          0
497 714          33 GOTO   RP460  ( 717 )
498 715 RP450    34 PT=    3
499 716          152 AB EX  WPT
500 717 RP460    1 GOSUB  NXLDEL          SKPLIN ENTRY NOT CHK ROMFLAG
500 720          0
501 721          1233 GOTO  RP430  ( 644 )
* SET LINK FOR FINAL END AND PUT IT TO PROPER PLACE.
* (FINAL END HAS TO BE RIGHT JUSTIFY IN A REG)
*
```

```

505 722 RP470    1142 C=C-1  PT          RESTORE END BYTE
506 723          1074 RCR    2          MOVE TO C[1:0]
507 724          1530 ST=C          SAVE BYTE IN STATUS
508 725          210 S5=    1          SET BIT FOR FINAL END
509 726          202 B=A    PT          SAVE BYTE PTR IN B[3]
510 727          33 GOTO   RP477  ( 732 )
511 730 RP475    1 GOSUB  INCADA
511 731          0
512 732 RP477    106 C=0    X
513 733          1 GOSUB  PTBYTA
513 734          0
514 735          1502 ? A#0  PT
515 736          1727 GOC    RP475  ( 730 )
516 737          142 AB EX  PT          RESTORE BYTE PTR
517 740          420 LC    4
518 741          34 PT=    3
519 742          1402 ? A<C  PT          CAN .END. FIT IN THIS REG ?
520 743          37 GOC    RP480  ( 746 ) NO
521 744          402 A=C    PT
522 745          73 GOTO   RP490  ( 754 )
523 746 RP480    246 AC EX  X
524 747          1146 C=C-1  X          PUT .END. TO NEXT REG
525 750          1160 DADD=C
526 751          412 A=C    WPT
527 752          116 C=0    W
528 753          1360 DATA=C
```

* GENERATE LINK FOR FINAL END

```

*
531 754 RP490    630 C=M
532 755          252 AC EX  WPT
533 756          1 GOSUB  GENLNK          LEAVES C IN M
533 757          0
```

* UPDATE CHAIN HEAD

```

*
536 760          1160 DADD=C
537 761          70 C=DATA          GIVE LAST BYTE TO FINAL END
538 762          1630 C=ST          GET END BYTE BACK FROM STATUS
539 763          1634 PT=    0
540 764          1520 LC    13
* CLEAR MEM BETWEEN OLD FINAL END AND NEW FINAL END
```

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

```

*
543 765      1360 DATA=C
544 766      630 C=M          GET NEW .END. ADDR
545 767      406 A=C        X
546 770      106 C=0        X
547 771      1160 DADD=C
548 772      1570 C=REGN 13   GET OLD FINAL END
549 773      246 AC EX      X
550 774      1550 REGN=C 13   UPDATE NEW FINAL END ADDR
551 775      56 B=0         W
552 776 CLM15 1406 ? A<C     X   WE DONE ?
553 777      73 GONC      RSTKCA (1006) YES, GOTO REBUILD THE KEY ASSIGNMENT
554 1000      1146 C=C-1     X
555 1001      1160 DADD=C
556 1002      356 BC EX      W
557 1003      1360 DATA=C
558 1004      356 BC EX      W
559 1005      1713 GOTO     CLM15 ( 776 )

```

```

*
* RSTKCA - REBUILD THE KEY REASSIGNMENT BIT MAPS
* AFTER READING IN THE STATUS TRACKS, IT DESTROYED THE OLD BITMAPS,
* SO THEY HAVE TO BE REBUILT. IN THIS CASE, THE KEY REASSIGNMENT
* JUST READ IN WILL TAKE PRECEDENCE. READING IN A PROGRAM WILL DESTROY
* THE KEY CODE USED IN THE LAST PROGRAM. IF A PROGRAM IS READ IN IN
* USER MODE, THE KEY ASSIGNED TO ANY ALPHA LABEL IN THE PROGRAM
* JUST READ IN SHOULD TAKE PRECEDENCE. THEREFORE, AFTER READING
* IN A PROGRAM, THE BIT MAP HAS TO BE REBUILT TOO.
* THE PROCEDURE IS AS FOLLOWING:
* 1. CLEAR THE BIT MAP
* 2. RESTORE THE KEY REASSIGNMENTS OF MAINFRAME FUNCTIONS & XROM FUNCTIONS
* 3. RESTORE KEY REASSIGNMENTS IN ALPHA LABELS; IF THE KEY HAS ALREADY BEEN
*    ASSIGNED TO ANOTHER FUNCTION :
*   A. IF IT IS AFTER READING IN A STATUS TRACK, CLEAR THE KEY CODE
*      IN THE ALPHA LABEL.
*   B. IF IT IS AFTER READING IN A PROGRAM, FIND THE KEY CODE SOMEWHERE
*      ELSE AND CLEAR IT THERE.

```

```

581 1006 RSTKCA      1 GOSUB ENCP00
581 1007              0
582 1010      1770 C=REGN 15
587 1011      132 C=0      M   CLEAR BIT MAP
584 1012      136 C=0      S
585 1013      1750 REGN=C 15
586 1014      460 LDI
587 1015      277 CON2     11    15

```

```

*
* GET THE KEY CODE FROM KEY REASSIGNMENT RECORD AND SET ITS BIT
* IN BIT MAP.

```

```

592 1016 RTKC10 1046 C=C+1   X
593 1017      1204 S7=       0
594 1020      1250 REGN=C 10
595 1021      416 A=C        W
596 1022      1570 C=REGN 13
597 1023      256 AC EX      W
598 1024      1546 ? A#C     X   REACH CHAIN HEAD ?
599 1025      313 GONC      RTKC30 (1056) YES, DONE WITH ALL THOSE REGISTERS
600 1026 RTKC15 1160 DADD=C

```

601	1027	70	C=DATA		
602	1030	416	A=C	SAVE C IN A TEMP	
603	1031	106	C=0	X	
604	1032	1160	DADD=C		
605	1033	256	AC EX	RESTORE C	
606	1034	1076	C=C+1	S	STILL A KEY REASSIGNMENT REG ?
607	1035	213	GONC	RTKC30 (1056)	NO, DONE WITH ALL THOSE REGISTERS
608	1036	1434	PT=	1	
609	1037	1214	?S7=1		FIRST KEY CODE ?
610	1040	23	GONC	*+2 (1042)	YES
611	1041	574	RCR	6	
612	1042	1352	? C#0	WPT	IS THERE A KEY CODE ?
613	1043	63	GONC	RTKC20 (1051)	NO
614	1044	416	A=C		
615	1045	1	GOSUB	TBITMA	
615	1046	0			
616	1047	1	GOSUB	SRBMAP	SET THE BIT IN BIT MAP
616	1050	0			
617	1051	RTKC20	1270 C=REGN	10	
618	1052	1214	?S7=1		DONE WITH 2ND KEY CODE ?
619	1053	1437	GOC	RTKC10 (1016)	YES
620	1054	1210	S7=	1	
621	1055	1513	GOTO	RTKC15 (1026)	
622	1056	RTKC30	1170 C=REGN	9	GET STARTING LOAD ADDR
623	1057	34	PT=	3	
624	1060	412	A=C	WPT	
625	1061	1	GOSUB	DECADA	POINT TO PREV. END ADDR
625	1062	0			
626	1063	1	GOSUB	DECADA	
626	1064	0			
627	1065	1270	C=REGN	10	
628	1066	252	AC EX	WPT	
629	1067	1250	REGN=C	10	SAVE THE ADDR IN REG.10
630	1070	RTKC40	1	GOSUB	GTFEND
630	1071	0			
631	1072	RTKC45	34	PT=	3
632	1073	1	GOSUB	GTLINK	
632	1074	0			
633	1075	1346	? C#0	X	CHAIN END ?
634	1076	207	GOC	RTKC50 (1116)	NOT YET
635	1077	414	?S8=1		CHECKSUM ERROR ?
636	1100	1	GOLNC	NFRPU	NO, ALL DONE. RTN
636	1101	2			
637	1102	1	GOSUB	GOSUB3	SAY "CHKSUM ERR"
637	1103	0			
638	1104	0	XDEF	APERMG	
639	1105	3	CON	003	C
640	1106	10	CON	010	H
641	1107	13	CON	013	K
642	1110	23	CON	023	S
643	1111	25	CON	025	U
644	1112	1015	CON	01015	M
645	1113	1	GOSUB	GOL3	
645	1114	0			
646	1115	0	XDEF	DISERR	
647	1116	RTKC50	1	GOSUB	UPLINK
647	1117	0			
648	1120	1076	C=C+1	S	IS IT AN END ?
649	1121	RTKC55	1513 GONC	RTKC45 (1072)	YES
650	1122	106	C=0	X	

```

651 1123      1160 DADD=C
652 1124      252 AC EX WPT
653 1125      1150 REGN=C 9          SAVE LINK & ADDR IN REG.9
654 1126      412 A=C WPT
655 1127      1 GOSUB INCAD2        POINT TO KEY CODE OF ALBL
656 1130      0
656 1131      1 GOSUB NXBYTA        GET KEY CODE
656 1132      0
657 1133      252 AC EX WPT
658 1134      160 N=C
659 1135      106 C=0 X
660 1136      1160 DADD=C
661 1137      1570 C=REGN 13
662 1140      530 M=C
663 1141      26 A=0 XS
664 1142      1506 ? A#0 X          IS THERE A KEY CODE ?
665 1143      203 GONC RUSR40 (1163) NO
666 1144      206 B=A X
667 1145      1 GOSUB TBITMA        TEST THE BIT MAP
667 1146      0
668 1147      1356 ? C#0           IS THIS BIT SET ?
669 1150      113 GONC RUSR30 (1161) NO, JUST SET IT
670 1151      146 AB EX X          CLEAR KEY CODE SOMEWHERE ELSE
671 1152      1270 C=REGN 10
672 1153      374 RCR 10
673 1154      356 BC EX
674 1155      1410 S1= 1
675 1156      1 GOSUB GCPKC0
675 1157      0
676 1160      33 GOTO RUSR40 (1163)
677 1161 RUSR30 1 GOSUB SRBMAP      SET BIT MAP
677 1162      0
678 1163 RUSR40 1170 C=REGN 9
679 1164      416 A=C
680 1165      1343 GOTO RTKC55 (1121)

```

```

*
*****
* CRFLD - CREATE A DATA FILE
* CRFLAS- CREATE AN ASCII FILE
* THE FILE NAME IS TAKEN FROM THE ALPHA REGISTER AND THE FILE*
* SIZE IN NUMBER OF REGISTERS IS TAKEN FROM THE X REGISTER. *
* IF THE SAME FILE NAME ALREADY EXIST, IT WILL GENERATE THE *
* "DUP FL" ERROR MESSAGE.
*****

```

```

*
691          ENTRY CRFLAS
692          ENTRY CRFLD
*
694 1166      204 CON 0204 D
695 1167      14 CON 014 L
696 1170      6 CON 006 F
697 1171      22 CON 022 R
698 1172      3 CON 003 C
699 1173 CRFLD 1204 S7= 0
700 1174      103 GOTO CRFL10 (1204)
*
702 1175      223 CON 0223 S
703 1176      1 CON 001 A
704 1177      14 CON 014 L
705 1200      6 CON 006 F

```

706	1201	22	CON	022	R
707	1202	3	CON	003	C
708	1203	CRFLAS	1210	S7=	1
709	1204	CRFL10	1	GOSUB	GOSUB0
709	1205		0		
710	1206		0	XDEF	X<999
711	1207		1346	? C#0	X
712	1210		1	GOLNC	ERRDE
712	1211		2		
713	1212		1334	PT=	13
714	1213		220	LC	2
715	1214		1214	?S7=1	
716	1215		23	GONC	CRFL20 (1217)
717	1216		1076	C=C+1	S
718	1217	CRFL20	1150	REGN=C	9
719	1220		1	GOSUB	GOSUB1
719	1221		0		
720	1222		0	XDEF	GTFLNA
721	1223		10	S3=	1
722	1224		1	GOSUB	GOSUB3
722	1225		0		
723	1226		0	XDEF	EFLSCH
724	1227		1614	?S0=1	
725	1230		43	GONC	CRF140 (1234)
726	1231		1	GOSUB	GOLONG
726	1232		0		
727	1233		0	XDEF	DUPFER
728	1234	CRF140	116	C=0	
729	1235		1160	DADD=C	
730	1236		1170	C=REGN	9
731	1237		406	A=C	X
732	1240		546	A=A+1	X
733	1241		1446	? A<B	X
734	1242		47	GOC	CRF200 (1246)
735	1243		1	GOSUB	GOL0
735	1244		0		
736	1245		0	XDEF	NORM
737	1246	CRF200	132	C=0	M
738	1247		356	BC EX	W
739	1250		256	AC EX	W
740	1251		474	ROR	S
741	1252		1160	DADD=C	
742	1253		406	A=C	X
743	1254		630	C=M	
744	1255		1360	DATA=C	
745	1256		1	GOSUB	GOSUB1
745	1257		0		
746	1260		0	XDEF	NXREG
747	1261		246	C=A	X
747	1262		406		
748	1263		530	M=C	
749	1264		1160	DADD=C	
750	1265		316	C=B	W
751	1266		1360	DATA=C	
752	1267		160	N=C	
753	1270		410	S8=	1
754	1271	CRF220	1214	?S7=1	
755	1272		153	GONC	CRF250 (1307)
756	1273		1146	C=C-1	X
757	1274		160	N=C	

GET THE REQUESTED SIZE

AND CHECK IF IT IS <=999
FILE SIZE = 0 ?
YES, SAY "DATA ERROR"

PUT THE FILE TYPE TO C.S
DATA FILE TYPE = 2
CREATE AN ASCII FILE ?
NO
ASCII FILE TYPE = 3
SAVE FILE SIZE IN REG.9(X)

GET FILE NAME

FILE ALREADY EXISTS ?
NO

A.X= REQUESTED FILE SIZE

ENOUGH ROOM ?
YES

SAVE THE HEADER IN B
C[10:8]= ADDR OF 1ST AVAILABLE REG
C.X= FILE NAME ADDR

STORE FILE NAME

POINT TO FILE HEADER REG

SAVE FILE HEADER ADDR IN M.X

STORE FILE HEADER
SAVE FILE SIZE IN M.X
SET FLAG FOR WRITING EOF MARK
CREATE A TEXT FILE ?
NO, IT IS A REGISTER FILE

```

758 1275      2 A=0      PT
759 1276      1 GOSUB   GOSUB1
759 1277      0
760 1300      0 XDEF     NXCHR
761 1301     460 LDI
762 1302     377 CON      255
763 1303      1 GOSUB   PTBYTA      STORE END OF TEXT MARK
763 1304      0
764 1305     246 AC EX   X          C.X= 1ST REG ADDR
765 1306     530 M=C
* NOW N.X= REGS COUNT,   M.X= FIRST REG ADDR
767 1307 CRF250 630 C=M
768 1310     416 A=C      W
769 1311      1 GOSUB   GOSUB1
769 1312      0
770 1313      0 XDEF     NXREG
771 1314     246 AC EX   X
772 1315     530 M=C
773 1316     1160 DADD=C
774 1317     260 C=N
775 1320     1146 C=C-1   X          ALL DONE ?
776 1321      57 GOC      CRF270 (1326) YES
777 1322     160 N=C
778 1323     116 C=0      W
779 1324     1360 DATA=C
780 1325     1623 GOTO     CRF250 (1307)
781 1326 CRF270 414 ?S8=1          FOR CLEAR FILE ?
782 1327      43 GONC      CRFLRT (1333) YES, DON'T WRITE EOF MARK
783 1330     116 C=0      W
784 1331     1156 C=C-1   W
785 1332     1360 DATA=C
786 1333 CRFLRT  1 GOLONG NFRPU
786 1334      2

```

*

* CLRFL - CLEAR FILE *

* FILE NAME IS TAKEN FROM THE ALPHA REGISTER. IF ALPHA IS EMPTY, *

* WILL GENERATE "NAME ERR" ERROR MESSAGE.

*

794 ENTRY CLRFL

*

```

796 1335 CLRF30 1204 S7= 0          ASSUME IT IS A REGISTER FILE
797 1336     1176 C=C-1   S
798 1337      27 GOC      CLRF40 (1341) IT IS A REGISTER FILE
799 1340     1210 S7= 1
800 1341 CLRF40  260 C=N
801 1342      374 RCR      10
802 1343      530 M=C
803 1344      406 A=C      X
804 1345     1160 DADD=C
805 1346     1274 RCR      7
806 1347      234 PT= 5
807 1350      112 C=0      WPT      SET POINTER TO ZERO
808 1351      34 PT= 3
809 1352      674 RCR      11
810 1353     1360 DATA=C
811 1354      404 S8= 0
812 1355     1143 GOTO     CRF220 (1271)

```

*

814	1356	214	CON	0214	L
815	1357	6	CON	006	F
816	1360	14	CON	014	L
817	1361	3	CON	003	C
818	1362	76	B=0	S	
819	1363	1	GOSUB	GOSUB	
819	1364	0			
820	1365	0	XDEF	FLSHAC	
821	1366	260	C=N		SEE WHAT TYPE IT IS
822	1367	1176	C=C-1	S	
823	1370	1176	C=C-1	S	
824	1371	1443	GONC	CLRF30 (1335)	NOT A PROG FILE
825	1372	1	GOSUB	GOL3	
825	1373	0			
826	1374	0	XDEF	FLTPER	

*

* FLSHAP - GET FILE ENTRY

* INPUT : S0=0 - GET CURRENT FILE ENTRY

* S0=1 - GET FILE ENTRY BY ITS NAME IN THE ALPHA REGISTER

* IF ALPHA IS EMPTY, IT WILL DEFAULT TO THE CURRENT FILE

* FLSHAB - SPECIAL ENTRY POINT

* FLSHAC - SPECIAL ENTRY POINT

* OUTPUT : N = FILE HEADER

* USED A,B,C,M,N,PT,S0-7 +2 SUB LEVEL

*

838		ENTRY	FLSHAP
839		ENTRY	FLSHAB
840		ENTRY	FLSHAC

*

842	1375	FLSHAP	76	B=0	S	
843	1376		1614	?S0=1		WANT CURRENT FILE ?
844	1377		123	GONC	FSA10 (1411)	YES
845	1400	FLSHAB	570	C=REGN	5	
846	1401		1356	? C#0	W	ALPHA EMPTY ?
847	1402		73	GONC	FSA10 (1411)	YES, DEFAULT TO CURRENT FILE
848	1403	FLSHAC	1	GOSUB	GOSUB1	
848	1404		0			
849	1405		0	XDEF	GTFLNA	GET FILE NAME
850	1406		1	GOSUB	GOL3	
850	1407		0			
851	1410		0	XDEF	RFLSCH	SEARCH FOR THE FILE
852	1411	FSA10	1	GOSUB	GOL3	
852	1412		0			
853	1413		0	XDEF	CURFL	GET CURRENT FILE

*

855	1414	RECLNG	1	GOSUB	GOSUB3	SAY RECORD TOO LONG
855	1415		0			
856	1416		0	XDEF	APERMG	
857	1417		22	CON	022	R
858	1420		5	CON	005	E
859	1421		3	CON	003	C
860	1422		40	CON	040	
861	1423		24	CON	024	T
862	1424		17	CON	017	0
863	1425		17	CON	017	0
864	1426		40	CON	040	
865	1427		14	CON	014	L
866	1430		17	CON	017	0
867	1431		16	CON	016	N

869 1432	1007 CON	@1007	G
869 1433	1 GOSUB	GOL3	
869 1434	0		
870 1435	0 XDEF	APEREX	

*

 * APPCHR - APPEND ALPHA REGISTER TO CURRENT RECORD IN THE WORKING *
 * TEXT FILE AND EXTEND THE CURRENT RECORD LENGTH. *
 * INSCR - INSERT ALPHA REGISTER IN FRONT OF CURRENT RECORD AND *
 * CHARACTER POINTER AND EXTEND THE CURRENT RECORD LENGTH. *

879		ENTRY	APPCHR	
881 1436	222 CON	@222	R	
882 1437	10 CON	@10	H	
883 1440	3 CON	@03	C	
884 1441	20 CON	@20	P	
885 1442	20 CON	@20	P	
886 1443	1 CON	@01	A	
887 1444	APPCHR 1204 S7=	0		
888 1445	APCH05 410 S8=	1		
889 1446	1 GOSUB	GOSUB		
889 1447	0			
890 1450	0 XDEF	SAPH5B		
891 1451	1414 ?S1=1			FILE EMPTY ?
892 1452	677 GOC	INRC10 (1541)		YES, GENERATE A NEW RECORD
893 1453	316 C=B	W		
894 1454	1274 RCR	7		C.X= CURRENT RECORD LENGTH.
895 1455	406 A=C	X		SAVE IN A
896 1456	260 C=N			
897 1457	574 RCR	6		C.X = CURRENT CHAR POSITION
898 1460	1214 ?S7=1			"INSCR" ?
899 1461	57 GOC	APCH20 (1466)		YES
900 1462	246 AC EX	X		C.X = RECORD LENGTH
901 1463	474 RCR	8		UPDATE CHARACTER POINTER
902 1464	160 N=C			TO END OF RECORD
903 1465	574 RCR	6		MOVE N BACK INTO PLACE
904 1466	APCH20 1046 C=C+1	X		
905 1467	346 BC EX	X		
906 1470	316 C=B			
907 1471	374 RCR	10		C[3:0]= CURRENT RECORD ADDR
908 1472	412 A=C	WPT		
909 1473	1 GOSUB	GOSUB1		
909 1474	0			
910 1475	0 XDEF	ADVREB		POINT TO STARTING CHAR
911 1476	256 AC EX	W		SAVE START CHAR ADDR IN A[11:8]
912 1477	574 RCR	6		
913 1500	416 A=C	W		A[11:8]= STARTING CHAR ADDR
914 1501	630 C=M			
915 1502	406 A=C	X		A.X= ALPHA REGISTER LENGTH
916 1503	316 C=B			
917 1504	1274 RCR	7		C.X= CURRENT RECORD LENGTH
918 1505	506 A=A+C	X		A.X = EXTENDED RECORD LENGTH
919 1506	460 LDI			
920 1507	377 CON	255		
921 1510	1406 ? ACC	X		RECORD LENGTH > 254 ?
922 1511	1033 GONC	RECLNG (1414)		YES
923 1512	74 RCR	3		
924 1513	252 AC EX	WPT		A[3:0]= CURRENT RECORD ADDR

```

925 1514      1 GOSUB PTBYTA      UPDATE CURRENT RECORD LENGTH
925 1515      0
926 1516     260 C=N
927 1517      74 RCR      3      C.X= CURRENT RECORD POINTER
928 1520     406 A=C      X
929 1521     630 C=M
930 1522     346 BC EX      X      B.X = ALPHA LENGTH
931 1523      1 GOSUB GOSUB1
931 1524      0
932 1525      0 XDEF      UPRCAB      UPDATE CURRENT RECORD & CHAR POINTE
933 1526     1204 S7=      0
934 1527     630 C=M
935 1530     443 GOTO      INRC30 (1574)
*
937          ENTRY      INSCHR
*
939 1531     222 CON      0222      R
940 1532      10 CON      010      H
941 1533      3 CON      003      C
942 1534     23 CON      023      S
943 1535     16 CON      016      N
944 1536     11 CON      011      I
945 1537 INSCHR 1210 S7=      1
946 1540     1053 GOTO      APCH05 (1445)
*
948 1541 INRC10      1 GOSUB GOLONG
948 1542      0
949 1543      0 XDEF      APPREC
*
*****
* INSREC - INSERT ALPHA REGISTER IN FRONT OF CURRENT RECORD AS      *
* A NEW RECORD.      *
*****
*
956          ENTRY      INSREC
*
958 1544     203 CON      0203      C
959 1545      5 CON      005      E
960 1546     22 CON      022      R
961 1547     23 CON      023      S
962 1550     16 CON      016      N
963 1551     11 CON      011      I
964 1552 INSREC 404 S8=      0
965 1553      1 GOSUB GOSUB
965 1554      0
966 1555      0 XDEF      SAPH5B
967 1556     1414 ?S1=1      FILE EMPTY ?
968 1557     1627 GOC      INRC10 (1541) YES
969 1560     260 C=N
970 1561      74 RCR      3      UPDATE RECORD & CHAR POINTER
971 1562     406 A=C      X
972 1563      1 GOSUB GOSUB1
972 1564      0
973 1565      0 XDEF      INREC#
974 1566     316 C=B
975 1567     1074 RCR      2
976 1570     416 A=C      W      A[11:8]= CURRENT RECORD ADDR
977 1571     1210 S7=      1
978 1572     630 C=M
979 1573     1046 C=C+1      X      ADD 1 FOR RECORD LENGTH

```

986	1574	INRC30	530	M=C		
981	1575		174	ROR	4	C[7:4]= CURRENT EOF MARK ADDR
982	1576		1234	PT=	7	
983	1577		412	A=C	WPT	
984	1600		174	ROR	4	
985	1601		34	PT=	3	
986	1602		412	A=C	WPT	A[3:0]= CURRENT EOF MARK ADDR
987	1603		256	AC EX	W	SAVE A IN N
988	1604		160	N=C		
989	1605		630	C=M		
990	1606		346	BC EX	X	B.X= ALPHA LENGTH
991	1607		1	GOSUB	GOSUB1	
991	1610		0			
992	1611		0	XDEF	ADVREB	POINT TO NEW EOF MARK
993	1612		260	C=N		GET ADDRESSES BACK FROM N
994	1613		252	AC EX	WPT	
* MAKE ROOM FOR INSERTING THE STRING						
* C[11:8]= CURRENT RECORD ADDR						
* C[3:0]= NEXT STORE ADDR(EOF MARK + # OF BYTES)						
* C[7:4]= NEXT PICK ADDR(CURRENT EOF MARK)						
*						
1000	1614	MKRM	174	ROR	4	
1001	1615		416	A=C	W	A[3:0]= NEXT PICK UP ADDR
1002	1616		1	GOSUB	GTBYTA	PICK UP NEXT BYTE
1002	1617		0			
1003	1620		256	AC EX	W	SAVE THE BYTE IN A.X
1004	1621		374	ROR	10	
1005	1622		256	AC EX	W	A[3:0]= NEXT STORE ADDR
1006	1623		1	GOSUB	PTBYTA	
1006	1624		0			
1007	1625		256	AC EX	W	IF THE CURRENT PICK UP ADDR REACHED
1008	1626		174	ROR	4	ORIGINAL STARTING ADDR, WE ARE DONE
1009	1627		416	A=C	W	A[3:0]= CURRENT PICK UP ADDR
1010	1630		174	ROR	4	C[3:0] = ORIGINAL START ADDR
1011	1631		1552	? A#C	WPT	
1012	1632		433	GONC.	INRC40 (1675)	ALL DONE
1013	1633		1604	S0=	0	DECREASE BOTH ADDRESS
1014	1634	MKRM30	542	A=A+1	PT	
1015	1635		542	A=A+1	PT	
1016	1636		1620	LC	14	
1017	1637		34	PT=	3	
1018	1640		1542	? A#C	PT	MOVE OVER THE EDGE OF THE REG ?
1019	1641		257	GOC	MKRM50 (1666)	NO
1020	1642		460	LDI		
1021	1643		357	CON2	14 15	C.X = HEX 0EF
1022	1644		266	C=A	XS	
1022	1645		426			
1023	1646		1546	? A#C	X	HAVE WE REACHED TOP OF MODULE ?
1024	1647		157	GOC	MKRM40 (1664)	NOT YET
1025	1650		460	LDI		
1026	1651		1	CON	01	
1027	1652		266	AC EX	XS	C.X = 201/301
1028	1653		1160	DADD=C		
1029	1654		70	C=DATA		
1030	1655		574	ROR	6	C.X= PREVIOUS MODULE ADDR
1031	1656		406	A=C	X	
1032	1657		1526	? A#C	XS	PREVIOUS MODULE THE BASE MODULE ?
1033	1660		43	GONC	MKRM40 (1664)	YES, THE PRECEEDING REG IS REG.41
1034	1661		6	A=0	X	
1035	1662		266	AC EX	XS	A.X = 200/300

NOMAS

Not Manufacturer Supported
recipient agrees NOT to contact manufacturer

```

1036 1663      546 A=A+1  X      A.X = 201/301
1037 1664 MKRM40 546 A=A+1  X
1038 1665      2 A=0    PT
1039 1666 MKRM50 256 AC EX  W      ADDRESSES INTO C
1040 1667      1614 ?S0=1      DONE WITH BOTH ADDR ?
1041 1670      1247 GOC   MKRM  (1614) YES
1042 1671      374 RCR    10      PUT SECOND ADDR IN PLACE
1043 1672      416 A=C    W
1044 1673      1610 S0=    1
1045 1674      1403 GOTO  MKRM30 (1634)

```

```

*
1047 1675 INRC40 630 C=M
1048 1676      174 RCR    4
1049 1677      412 A=C    WPT      MOVE M[7:4] TO A[3:0]
1050 1700      374 RCR    10
1051 1701      1146 C=C-1  X
1052 1702      530 M=C
1053 1703      1214 ?S7=1      FOR "INSREC" ?
1054 1704      253 GONC   PUTAPH (1731) NO
1055 1705      303 GOTO   PUTA20 (1735)

```

```

*
*****
* APPREC - APPEND ALPHA REGISTER TO END OF CURRENT WORKING TEXT FILE*
* AND GENERATE A NEW RECORD. IF ALPHA EMPTY, THEN NO ACTION*
* WILL BE TAKEN.
*
*****

```

```

*
1063      ENTRY  APPREC
*
1065 1706      203 CON    0203      C
1066 1707      5 CON     005        E
1067 1710      22 CON    022        R
1068 1711      20 CON    020        P
1069 1712      20 CON    020        P
1070 1713      1 CON     001        A
1071 1714 APPREC 504 S6=    0
1072 1715      404 S8=    0
1073 1716      1 GOSUB   GOSUB
1073 1717      0
1074 1720      0 XDEF    SAPH35

```

```

* NOW N = FILE HEADER
* M[2:0] = ALPHA LENGTH - 1
* M[7:4] = ALPHA HEAD ADDR
* M[11:8] = CURRENT END OF FILE MARK ADDR
* R[6:4] = LAST RECORD # + 1
*

```

```

1081 1721      1 GOSUB   GOSUB1
1081 1722      0
1082 1723      0 XDEF    NWREC#      SET CURRENT REC# = LAST REC #
1083 1724      630 C=M      MOVE M[11:4] TO A[7:0]
1084 1725      174 RCR    4
1085 1726      416 A=C    W
1086 1727      374 RCR    10      C.X = RECORD LENGTH
1087 1730      53 GOTO   PUTA20 (1735)

```

```

*
*****
* PUTAPH - EXTEND CURRENT RECORD
* NO SPECIAL INPUTS
* PUTA20 - GENERATE NEW RECORD
* INPUT : C.X = NEW RECORD LENGTH

```

```

* BOTH -
* INPUT : M[2:0] = # OF BYTES ADDED - 1
*          A[3:0] = STARTING SOURCE CHAR ADDR
*          A[7:4] = STARTING DESTINATION CHAR ADDR
*          S6 = 0 - WILL WRITE END OF TEXT MARK AT END OF RECORD
*              1 - WILL NOT      //
*

```

```

*
1101          ENTRY  PUTAPH
*
1103 1731 PUTAPH    1 GOSUB  GTBYTA          GET A BYTE FROM SOURCE
1103 1732          0
1104 1733          1 GOSUB  INCADA          UPDATE SOURCE CHAR ADDR
1104 1734          0
1105 1735 PUTA20    256 AC EX  W            MOVE DEST ADDR INTO PLACE
1106 1736          174 RCR    4
1107 1737          256 AC EX  W
1108 1740          1 GOSUB  PTBYTA
1108 1741          0
1109 1742          1 GOSUB  GOSUB1
1109 1743          0
1110 1744          0 XDEF    NXCHR          POINT TO NEXT CHAR ADDR IN FILE
1111 1745          630 C=M
1112 1746          1146 C=C-1  X            ALL DONE ?
1113 1747          67 GOC     PUTA30 (1755) YES
1114 1750          530 M=C
1115 1751          256 AC EX  W            MOVE SOURCE ADDR BACK INTO PLACE
1116 1752          374 RCR    10
1117 1753          256 AC EX  W
1118 1754          1553 GOTO   PUTAPH (1731)
1119 1755 PUTA30    514 ?S6=1              WRITE END OF TEXT MARK
1120 1756          57 GOC     PUTA40 (1763) NO
1121 1757          460 LDI
1122 1760          377 CON     255
1123 1761          1 GOSUB  PTBYTA
1123 1762          0
1124 1763 PUTA40    1 GOLONG  NFRPU
1124 1764          2
*
*
*****
* CUR#=0 - ROUTINE TO SET CURRENT FILE NUMBER TO ZERO
*
1130          ENTRY  CUR#=0
*
1132 1765 CUR#=0    460 LDI
1133 1766          100 CON2    4          0
1134 1767          1160 DADD=C
1135 1770          70 C=DATA
1136 1771          1174 RCR     9
1137 1772          106 C=0     X
1138 1773          274 RCR     5
1139 1774          1360 DATA=C
1140 1775          1740 RTN
*
*
1144          UNLIST
1147          END

```

ERRORS : 0

SYMBOL TABLE

APCH05	1445	-	1540
APCH20	1466	-	1461
APCH8	1444	-	
APREC	1714	-	
CLEANM	570	-	540
CLM15	776	-	1005
CLRF30	1335	-	1371
CLRF40	1341	-	1337
CLRFL	1362	-	
CRF140	1234	-	1230
CRF200	1246	-	1242
CRF220	1271	-	1355
CRF250	1307	-	1325 1272
CRF270	1326	-	1321
CRFL10	1204	-	1174
CRFL20	1217	-	1215
CRFLAS	1203	-	
CRFLD	1173	-	
CRFLRT	1333	-	1327
CUR#=0	1765	-	
DUPFER	231	-	224
FLSHA2	1400	-	
FLSHA0	1403	-	
FLSHA2	1375	-	
FSA10	1411	-	1402 1377
GETP	376	-	
GETSUB	370	-	
GETX10	14	-	6
GETX20	30	-	24
GETXX	13	-	
INRC10	1541	-	1557 1452
INRC30	1574	-	1530
INRC40	1675	-	1632
INSCHR	1537	-	
INSREC	1552	-	
MCRM	1614	-	1670
MCRM30	1634	-	1674
MCRM40	1664	-	1660 1647
MCRM50	1666	-	1641
PUTA20	1735	-	1730 1705
PUTA30	1755	-	1747
PUTA40	1763	-	1756
PUTAPH	1731	-	1754 1704
RECLNG	1414	-	1511
RP100	377	-	371
RP150	413	-	406
RP200	423	-	414
RP220	440	-	424
RP240	441	-	437 422
RP245	445	-	412
RP250	476	-	472
RP300	501	-	532
RP310	533	-	514
RP320	553	-	550
RP330	553	-	
RP405	602	-	576

RP410	624	-	605	
RP425	643	-	652	
RP430	644	-	721	
RP440	712	-	700	
RP450	715	-	656	654
RP460	717	-	714	
RP470	722	-	670	
RP475	730	-	736	
RP477	732	-	727	
RP480	743	-	743	
RP490	754	-	745	
RSTKCA	1006	-	777	
RTKC10	1016	-	1053	
RTKC15	1026	-	1055	
RTKC20	1051	-	1043	
RTKC30	1056	-	1035	1025
RTKC40	1070	-		
RTKC45	1072	-	1121	
RTKC50	1116	-	1076	
RTKC55	1121	-	1165	
RUSR30	1161	-	1150	
RUSR40	1163	-	1160	1143
SAPH10	130	-	126	
SAPH55	62	-		
SAPH5E	61	-		
SAVEP	142	-	230	
SAVEX	5	-		
SAVX10	56	-	51	
SVPR10	167	-	171	
SVPR25	210	-	204	
SVPR40	245	-	220	
SVPR42	257	-	253	
SVPR45	312	-	340	
SVPR60	341	-	335	

ENTRY TABLE

APPCR	1444	-
APPREC	1714	-
CLRFL	1362	-
CRFLAS	1203	-
CRFLD	1173	-
CUR#=0	1765	-
DUPFER	231	-
FLSHAS	1400	-
FLSHAC	1403	-
FLSHAP	1375	-
GETP	376	-
GETSUR	370	-
GETXX	13	-
INSCNR	1537	-
INSREC	1552	-
PUTAPH	1731	-
RP330	555	-
SAPHSS	62	-
SAPHSB	61	-
SAVEP	142	-
SAVEX	5	-

EXTERNAL REFERENCES

[illegible]

GTLINK	1074																			
GTPRAD	144																			
INCAD2	562	661	701	1127																
INCAD2	564	662	702	1130																
INCADA	431	520	555	703	730	1733														
INCADA	432	521	556	704	731	1734														
INREC*	1565																			
MEMLFT	446																			
MEMLFT	447																			
NFRPU	66	360	1100	1333	1763															
NFRPU	67	361	1101	1334	1764															
NORM	256	1245																		
NORNEX	475																			
NWREC*	1723																			
NXBYTA	314	433	645	1131																
NXBYTA	315	434	646	1132																
NXCHR	324	344	506	1300	1744															
NXDEL	717																			
NXDEL	720																			
NXREG	277	352	1260	1313																
PTBYTA	326	346	522	561	566	613	706	733	1303	1514	1623	1740								
PTBYTA	1761																			
PTBYTA	327	347	523	562	567	614	707	734	1304	1515	1624	1741								
PTBYTA	1762																			
PUFLSB	227																			
PUTPCX	600																			
PUTPCX	601																			
RCL	54																			
RCL	55																			
RFLSCH	1410																			
SAPHSS	1720																			
SAPHSE	1450	1555																		
GREMAP	1047	1161																		
GREMAP	1050	1162																		
TBITMA	1045	1145																		
TBITMA	1046	1146																		
TXEND	102																			
UPLINK	1116																			
UPLINK	1117																			
UPRCAS	1525																			
XK999	1206																			

End of VASM assembly

 VASM ROM ASSEMBLY REV. 6/81A

OPTIONS: L C S

2 FILE SCAP4B

*

 *
 * EXTENDED MEMORY FILE STRUCTURE *
 *

*
 * THE EXTENDED MEMORY CAN BE COMPOSED BY UP TO THREE MODULES. THE FIRST
 * MODULE WE CALL IT THE BASE MODULE IS THE SAME MODULE CONTAINING THE
 * EXTENDED FUNCTION ROM. THIS MODULE HAS TO BE PRESENT TO EXECUTE ANY
 * FUNCTION RELATED TO THE EXTENDED MEMORY. THIS MODULE CONTAINS 128

* REGISTERS OF MEMORY AND IS LOCATED AT MEMORY ADDRESS 040-0BF(HEX).
 * THE FIRST FILE ALWAYS START FROM THE BASE MODULE AND THIS FILE OR THE
 * SEQUENT FILES MAY EXPEND INTO THE EXTENDED MEMORY MODULE.
 * THE SECOND KIND OF MODULE WE CALL IT THE EXTENDED MEMORY MODULE. IT
 * CONTAINS 239 REGISTERS AND IS LOCATED AT MEMORY ADDRESS 201-2EF(HEX)
 * IF IT IS PLUG-IN TO PORT 1 OR 3, OR WILL BE AT ADDRESS 301-3EF(HEX)
 * IF IT IS PLUG-IN TO PORT 2 OR 4. UP TO 2 EX.MEM MODULES CAN BE PLUG-IN
 * AT THE SAME TIME TO GET THE FULL CAPACITY OF THE EXTENDED MEMORY. BUT
 * 2 EX.MEM MODULES HAVE TO BE PLUG-IN HORIZONTALLY OPPOSITE PORT.
 * THE LINKAGE AMONG THESE THREE DISCONTINUE MEMORY IS AS FOLLOWS: THE
 * BASE MODULE ALWAYS IS THE BEGINNING OF THE EXTENDED MEMORY. THEN IT
 * WILL LINK TO 2EF OR 3EF DEPENDING ON WHICH ONE IS PLUG-IN FIRST. IF
 * THE TWO EX.MEM MODULE ARE PLUG-IN AT THE SAME TIME, THE BASE MODULE
 * WILL BE LINKED TO ADDRESS 2EF FIRST(THE EX.MEM MODULE IN PORT 1 OR 3)
 * THE LOWEST REGISTER IN EVERY MODULE(040 FOR BASE MODULE, 201/301 FOR
 * THE EX.MEM MODULE) IS USED FOR THE LINKAGE BETWEEN MODULES. IT
 * CONTAINS THE FOLLOWING INFORMATIONS:

* 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 * -----
 * | | | | | | | | | | | | | |
 * -----
 * D C B A

* A - MODULE ID
 * 0BF FOR BASE MODULE
 * 2EF FOR EX.MEM MODULE IN PORT 1 OR 3
 * 3EF FOR EX.MEM MODULE IN PORT 2 OR 4
 * B - NEXT MODULE ID = 2EF OR 3EF (= 000 IF NO NEXT MODULE CONNECTED)
 * C - PREVIOUS MODULE ID = 040 OR 2EF OR 3EF
 * D - IN BASE MODULE THIS IS THE CURRENT FILE NUMBER
 * DON'T CARE IN EX.MEM MODULE

* FILE HEADER INFORMATION:
 * EVERY FILE START WITH A REGISTER CONTAINS THE FILE NAME. THE FILE NAME
 * CAN BE UP TO 7 CHARACTERS WITH TAILING BLANKS. THE NEXT REGISTER CALLED
 * THE FILE HEADER, IT CONTAINS :

* 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 * -----
 * | | | | | | | | | | | | | |
 * -----
 * D C B A

* A - FILE SIZE IN NUMBER OF REGISTERS (NOT INCLUDING THE FILE NAME &
 * HEADER REGISTERS)
 * B - FOR PROGRAM FILE = PROGRAM LENGTH IN NUMBER OF BYTES
 * FOR DATA FILE = CURRENT REGISTER POINTER
 * FOR ASCII FILE = CURRENT RECORD POINTER
 * C - FOR PROGRAM FILE = DON'T CARE
 * FOR DATA FILE = DON'T CARE
 * FOR ASCII FILE = CURRENT CHARACTER POINTER
 * D - FILE TYPE = 1 - PROGRAM FILE
 * = 2 - DATA FILE
 * = 3 - ASCII FILE

* THE ASCII FILE IS A FILE CONTAINS RECORDS OF ASCII STRING. EVERY RECORD
 * COMPOSE 1 BYTE OF RECORD LENGTH AT THE BEGINNING AND FOLLOWED BY THE
 * THE ASCII STRING. THE RECORD LENGTH CAN ONLY AS LONG AS 254. AFTER THE
 * LAST RECORD IN THE ASCII FILE, THERE ALWAYS BE AN "FF" INDICATING THE
 * END OF FILE.

* END OF MEMORY MARK :
 * AT THE END OF THE LAST FILE IN MEMORY, THERE ALWAYS BE A REGISTER
 * CONTAINING "FFFFFFFFFFFFFF" TO INDICATE THE END OF MEMORY.
 * AFTER FILES HAVE BEEN CREATED, PULL OUT ANY EXTENDED MEMORY MODULE
 * WILL CAUSE ALL OR SOME FILES LOST.
 * 1. IF PULL OUT THE BASE MODULE, ALL FILES ARE LOST.
 * 2. IF PULL OUT ANY EX.MEM MODULE, ANY PARTIAL FILE LEFT WILL BE LOST.
 * 3. IF THE END OF MEMORY MARK IS MISSING, LAST FILE WILL BE LOST

 * XTOA - X TO ALPHA REGISTER *
 * IF X IS A NUMBER, IT HAS TO BE IN THE RANGE 0 <= X < 256 *

	88		ENTRY	XTOA	
	90	0	201	CON @201	A
	91	1	17	CON @17	0
	92	2	24	CON @24	T
	93	3	30	CON @30	X
	94	4 XTOA	370	C=REGN 3	
	95	5	356	B=C W	SAVE X IN B
	95	6	316		
	96	7	1176	C=C-1 S	
	97	10	1376	? C#0 S	DOES X HAVE A STRING ?
	98	11	1	GOLNC XARCL	YES, JUST DO A REGULAR ARCL
	98	12	2		
	99	13	1	GOSUB GOSUB0	GET INT(X)
	99	14	0		
	100	15	0	XDEF X<256	
	101	16	1634	PT= 0	
	102	17	130	G=C	
	103	20	1	GOLONG APPEND	SHIFT THE BYTE INTO THE ALPHA REGIS
	103	21	2		

NOMAS

NOT Manufacturer Supported
 recipient agrees NOT to contact manufacturer

 * ENDIR - EXTERNAL MEMORY DIRECTORY *
 * WHILE THE DIRECTORY IS RUNNING, "R/S" AND "OFF" KEYS WILL *
 * ABORT IT. ANY OTHER KEY DOWN WILL ONLY FREEZE THE DISPLAY *
 * UNTIL THE KEY IS UP AGAIN. *
 * THIS FUNCTION WILL RETURN THE # OF REGISTERS LEFT UNUSED IN *
 * THE EXTERNAL MEMORY TO THE X REGISTER. *

	115		ENTRY	ENDIR	
	117	22	222	CON @222	R
	118	23	11	CON @11	I
	119	24	4	CON @04	D
	120	25	15	CON @15	M
	121	26	5	CON @05	E
	122	27 ENDIR	1	GOSUB GOSUB2	
	122	30	0		
	123	31	0	XDEF CUR#=0	SET CURRENT FILE # TO 0
	124	32 ENDR10	460	LDI	
	125	33	100	CON2 4 0	
	126	34	1160	DADD=C	
	127	35	70	C=DATA	
	128	36	1174	ROR 9	INCREMENT CURRENT FILE #

129	37	1046	C=C+1	X	
130	40	274	ROR	5	
131	41	1360	DATA=C		
132	42	210	S5=	1	
133	43	76	B=0	S	
134	44	116	C=0	W	
135	45	530	M=C		GUARANTEE WON'T FIND ANY FILE NAME
136	46	1	GOSUB	GOSUB	
136	47	0			
137	50	0	XDEF	CURFLD	GET NEXT FILE
138	51	214	?S5=1		REACHED END OF DIR ?
139	52	733	GONC	EMDR60 (145)	YES
140	53	1	GOSUB	CLLCDE	
140	54	0			
141	55	34	PT=	3	COUNT CHARACTERS IN A.M
142	56	116	C=0	W	
143	57	620	LC	6	
144	60	432	A=C	M	A.M= 6
145	61	EMDR20	630	C=M	SEND THE FILE NAME TO DISPLAY
146	62	1574	ROR	12	
147	63	530	M=C		
148	64	1	GOSUB	ASCLCD	SEND THE ASCII TO DISPLAY
148	65	0			
149	66	672	A=A-1	M	DONE WITH 7 CHARS ?
150	67	1723	GONC	EMDR20 (61)	NOT YET
151	70	460	LDI		
152	71	40	CON	@40	
153	72	1750	SLSABC		SEND A BLANK
154	73	260	C=N		
155	74	460	LDI		START AT @400 SO THAT DIVIDING BY 4
156	75	400	CON	@400	TWICE WILL GET IT TO @20 FOR ASCII P
157	76	EMDR30	746	C=C+C	X
158	77		746	C=C+C	X
159	100	1706	C SR	X	DIVIDE BY 4 TO GET NEXT
160	101	1176	C=C-1	S	FILE TYPE CHARACTER
161	102	1743	GONC	EMDR30 (76)	C.X= @04(D) OR @01(A)
162	103	DSFLTP	1750	SLSABC	FILE TYPE REACHED ZERO YET ?
163	104		260	C=N	NO, DIVIDE AND DECREMENT AGAIN
164	105	1334	PT=	13	SEND FILE TYPE CHARACTER TO DISPLAY
165	106	320	LC	3	SEND FILE SIZE
166	107	416	A=C	W	
167	110	1	GOSUB	GENNUM	3 DIGITS FOR FILE SIZE
167	111	0			
168	112	410	S8=	1	
169	113	1	GOSUB	MSG105	PRINT IT
169	114	0			
170	115	1314	?S13=1		RUNNING ?
171	116	1	GOSUBNC	RSTMS0	NO, CLEAR MESSAGE FLAG
171	117	0			
172	120	106	C=0	X	
173	121	EMDR45	1046	C=C+1	X
174	122		1773	GONC	EMDR45 (121)
175	123		1714	CHK KB	ANY KEY HIT ?
176	124	EMDR50	1063	GONC	EMDR10 (32)
177	125		1040	C=KEYS	NO, GO FOR THE NEXT FILE
178	126		74	ROR	SEE WHAT KEY IT IS
179	127		406	A=C	
180	130		1634	PT=	0
181	131		742	C=C+C	PT
182	132		1	GOLC	OFF
					OFF KEY ?
					YES

182	133	3			
183	134	1	GOSUB	RSTKB	WAIT UNTIL KEY UP
183	135	0			
184	136	26	A=0	XS	
185	137	460	LDI		
186	140	207	CON2	8	7
187	141	1546	? A#C	X	IS IT THE "R/S" ?
188	142	1627	GOC	EMDR50 (124)	NO, IGNORE ANY OTHER KEYS
189	143	1	GOLONG	RSTSEQ	CLEAR FLAGS FOR STOPPING PROGRAM
189	144	2			
190	145	EMDR60	146	AB EX	X
191	146	460	LDI		A.X= # OF REG STILL AVAILABLE
192	147	2	CON	2	
193	150	706	A=A-C	X	RESERVE 2 REGS FOR FILE NAME & HEADER
194	151	23	GONC	EMDR70 (153)	
195	152	6	A=0	X	
196	153	EMDR70	1	GOSUB	GOSUB0
196	154	0			CONVERT INTO DECIMAL
197	155	0	XDEF	BIN-D	
198	156	116	C=0		
199	157	1160	DADD=C		
200	160	614	?S11=1		PUSH FLAG SET ?
201	161	1	GSUBC	R*SUB	PUSH STACK IF YES
201	162	1			
202	163	356	BC EX	W	
203	164	350	REGN=C	3	PUT THE AVAILABLE REGS # TO X
204	165	460	LDI		
205	166	100	CON2	4	0
206	167	1160	DADD=C		
207	170	70	C=DATA		
208	171	1174	ROR	9	
209	172	1146	C=C-1	X	
210	173	406	A=C	X	
211	174	274	ROR	5	
212	175	1760	DATA=C		MAKE THE LAST FILE BECOME WORKING FIL
213	176	646	A=A-1	X	IS CURRENT FILE # = 0 ?
214	177	EMDR80	1	GOLNC	NFRPR
214	200	2			NO
215	201	1	GOSUB	CLLCODE	
215	202	0			
216	203	1	GOSUB	MESSL	
216	204	0			
217	205	4	CON	@04	D
218	206	11	CON	@11	I
219	207	22	CON	@22	R
220	210	40	CON	@40	
221	211	5	CON	@05	E
221	212	15	CON	@15	M
223	213	20	CON	@20	P
224	214	24	CON	@24	T
225	215	1031	CON	@1031	Y
226	216	1	GOSUB	LEFTJ	
226	217	0			
227	220	410	S8=	1	
228	221	1	GOSUB	MSG105	
228	222	0			
229	223	1543	GOTO	EMDR80 (177)	

*

* EFLSCH - EXTERNAL MEMORY FILE SEARCH

```

* INPUT : M = LEFT JUSTIFIED TARGET FILE NAME
*         IF S3=1, WHEN THE 1ST AVAILABLE REG ADDR IS "0BF", IT WILL
*         WRITE "000000000000BF" TO REG.40
*         IF S3=0, WON'T DO ANYTHING TO REG.40
* OUTPUT : IF S0=1, FILE IS FOUND. THEN :
*         A[10:8] = 1ST REG ADDR OF THE FILE(FILE NAME REG)
*         N = FILE HEADER (2ND REG OF THE FILE)
*         N[12:10] = ADDR OF FILE HEADER
*         IF S0=0, FILE IS NOT FOUND. THEN
*         A[10:8] = 1ST AVAILABLE REGISTER ADDR
*         B.X = # OF REGISTER STILL AVAILABLE
*         PT = 3
* USED A, B.S, B[6:0],C,N,S0,S1,S2,PT +2 SUB LEVEL
*
247          ENTRY EFLSCH          GENERAL FILE SEARCH
248          ENTRY FSCHT           SEARCH A TEXT FILE
249          ENTRY FSCHP           SEARCH A PROG FILE
250          ENTRY RFLSCH          SEARCH A FILE TO READ
251          ENTRY CURFL           GET CURRENT FILE
252          ENTRY CURFLD          GET CURRENT FILE FOR DIRECTORY
253          ENTRY CURFLT          GET CURRENT TEXT FILE
254          ENTRY CURFLR          GET CURRENT REG FILE
*
256 224 CURFLR 1334 PT= 13
257 225          220 LC 2
258 226          33 GOTO CURFL0 ( 231 )
259 227 CURFLT 1334 PT= 13
260 230          320 LC 3
261 231 CURFL0 376 BC EX S
262 232 CURFL 204 S5= 0
263 233 CURFLD 110 S4= 1
264 234          4 S3= 0
265 235          123 GOTO EFLS05 ( 247 )
266 236 FSCHP 1334 PT= 13
267 237          120 LC 1
268 240          33 GOTO FSCH10 ( 243 )
269 241 FSCHT 1334 PT= 13
270 242          320 LC 3
271 243 FSCH10 376 BC EX S          SAVE THE FILE TYPE IN B.S
272 244 RFLSCH 4 S3= 0
273 245 EFLSCH 104 S4= 0
274
275          ENTRY EFLS02          USED IN 41CX
276 246 EFLS02 204 S5= 0          NOT FOR DIRECTORY
277 247 EFLS05 1604 S0= 0
278 250          1404 S1= 0
279 251          1004 S2= 0
280 252          16 A=0 W
281 253          460 LDI
282 254          100 CON2 4 0
283 255          1160 DADD=C
284 256          460 LDI
285 257          277 CON2 11 15
286 260          406 A=C X          A.X = HEX 0BF
287 261          70 C=DATA          LOAD REG 40
288 262          1546 ? A#C X          HAS REG 40 BEEN INITIALIZED ?
289 263          33 GOMC EFLS08 ( 266 ) YES, AT LEAST 1 FILE EXISTS
290 264 EFLS07 1410 S1= 1          SAY END OF DIR REACHED
291 265          133 GOTO EFLS10 ( 300 )
292 266 EFLS08 114 S4=1          FOR CURRENT FILE ?

```

293	267	113	GONC	EFLS10 (300)	NO
294	270	1174	RCR	9	C.X= CURRENT FILE NUMBER
295	271	1146	C=C-1	X	
296	272	777	GOC	FL# = 0 (371)	CURRENT FILE # = 0
297	273	356	BC EX	W	
298	274	174	RCR	4	SAVE CURRENT FILE # -1 IN B[6:4]
299	275	346	BC EX	X	
300	276	374	RCR	10	
301	277	356	BC EX	W	
302	300	EFLS10	256	C=A	W
302	301		416		SAVE LAST FILE ADDR IN A[10:8]
303	302		474	RCR	8
304	303		246	AC EX	X
305	304		74	RCR	3
306	305		1046	C=C+1	X
307	306		74	RCR	3
308	307		416	A=C	W
309	310		1414	?S1=1	REACHED END OF DIR ?
310	311		617	GOC	EFLS70 (372) YES
311	312		1160	DADD=C	ENABLE FILE NAME REG
312	313		70	C=DATA	
313	314		256	AC EX	W
314	315		730	CM EX	A= FILE NAME, C= ADDR
315	316		1556	? A#C	C=TARGET NAME, M=ADDR
316	317		27	GOC	IS THIS THE FILE ?
317	320		1610	S0=	EFLS20 (321) NO
318	321	EFLS20	730	CM EX	REMEMBER FOUND IT
319	322		256	AC EX	C=ADDR, M=TARGET NAME
320	323		1056	C=C+1	A=ADDR, C=FILE NAME
321	324		577	GOC	REACHED END OF DIR ?
322	325		214	?S5=1	YES, COMPUTE AVAILABLE REGS
323	326		33	GONC	FOR DIRECTORY ?
324	327		1156	C=C-1	EFLS30 (331) NO
325	330		530	M=C	
326	331	EFLS30	1	GOSUB	SAVE THE FILE NAME IN M
326	332		0		GOSUB1
327	333		0	XDEF	POINT TO NEXT REGISTER
328	334		1014	?S2=1	MEMORY DISCONTINUE ?
329	335		357	GOC	EFLS70 (372) YES
330	336		246	C=A	
331	337		406		
331	340		1160	DADD=C	
332	341		70	C=DATA	LOAD FILE HEADER REG
333	342		374	RCR	10
334	343		246	C=A	X
334	344		406		SAVE HEADER ADDR IN N[12:10]
335	345		174	RCR	4
336	346		160	N=C	SAVE FILE HEADER IN N
337	347		1046	C=C+1	X
338	350		346	BC EX	X
339	351		1	GOSUB	C.X= FILE LENGTH + 1
339	352		0		B.X= FILE LENGTH + 1 IN REGISTERS
340	353		0	XDEF	POINT TO NEXT FILE NAME REG
341	354		1014	?S2=1	MEMORY DISCONTINUE ?
342	355		157	GOC	EFLS70 (372) YES
343	356		1614	?S0=1	FOUND THE FILE YET ?
344	357		577	GOC	EFLS85 (436) YES
345	360		114	?S4=1	LOOKING FOR CURRENT FILE ?
346	361		1173	GONC	EFLS10 (300) NO
347	362		316	C=B	W

348	363	174	RCR	4	
349	364	1146	C=C-1	X	
350	365	517	GOC	EFLS85 (436)	REACHED CURRENT FILE
351	366	374	RCR	10	
352	367	356	BC EX	W	
353	370	1103	GOTO	EFLS10 (300)	
354	371	FL#=0	753	GOTO	FLNOFN (466) JUST A JUMP
355	372	EFLS70	1604	S0=	0 SAY FILE NOT FOUND
356	373		256	C=A	W FIND OUT AVAILABLE REGS
356	374		416		
357	375		474	RCR	8
358	376		406	A=C	X A.X=1ST AVAILABLE REG ADDR
359	377		1160	DADD=C	WRITE END OF MEM MARK
360	400		116	C=0	W
361	401		1156	C=C-1	W
362	402		1360	DATA=C	
362	403	EFLS75	14	?S3=1	FOR READ ?
364	404		47	GOC	EFLS76 (410) NO, FOR WRITE
365	405		214	?S5=1	FOR DIRECTORY ?
366	406		603	GONC	FLNOFN (466) NO, FILE NOT FOUND
367	407		204	S5=	0
368	410	EFLS76	1	GOSUB	GOSUB1
369	411		0		
369	412		0	XDEF	ENRGAD GET LAST REG ADDR IN THIS MODULE
370	413		706	A=A-C	X A.X=AVAILABLE REGS IN THIS MODULE
371	414		646	A=A-1	X RESERVE ONE REG FOR END OF MEM MARK
372	415		206	B=A	X B <- REGISTERS LEFT IN PRESENT MODULE
373	416		246	AC EX	X
374	417	EFLS80	1	GOSUB	GOSUB1 LOOK FOR NEXT MODULE
374	420		0		
375	421		0	XDEF	NXTMDL
376	422		1506	? A#0	X IS THERE A NEXT MODULE ?
377	423		653	GONC	EFLS90 (510) NO
378	424		460	LDI	
379	425		356	CON	238 ADD 238 TO B.X
380	426		246	AC EX	X
381	427		446	A=A+B	X
382	430		206	B=A	X
383	431		406	A=C	X
384	432		534	PT=	6
385	433		1502	? A#0	PT ARE WE COMING FROM 201 OR 301 ?
386	434		547	GOC	EFLS90 (510) YES, DON'T LOOK FOR ANOTHER MODULE
387	435		1623	GOTO	EFLS80 (417)
388	436	EFLS85	14	?S3=1	FOR WRITE ?
389	437		517	GOC	EFLS90 (510) YES, DON'T CARE ABOUT FILE TYPE
390	440		260	C=N	IT IS FOR READ, CHECK FILE TYPE
391	441		674	RCR	11 C.XS = FILE TYPE
392	442		426	A=C	XS A.XS= FILE TYPE IN EX.MEM
393	443		336	C=B	S C.S= FILE TYPE LOOKED FOR
394	444		674	RCR	11 C.XS = FILE TYPE LOOKED FOR
395	445		1336	? B#0	S NEED TO CHECK FILE TYPE ?
396	446		423	GONC	EFLS90 (510) NO
397	447		1566	? A#C	XS SAME FILE TYPE ?
398	450		403	GONC	EFLS90 (510) YES,

*

400		ENTRY	FLTPER	
401	451	FLTPER	1	GOSUB
401	452		0	
402	453		0	XDEF
403	454		6	CON

F

404	455	14	CON	014	L
405	456	40	CON	040	
406	457	24	CON	024	T
407	460	31	CON	031	Y
408	461	20	CON	020	P
409	462	1005	CON	01005	E
410	463	1	GOSUB	GOLONG	
410	464	0			
411	465	0	XDEF	DISERR	

*

413			ENTRY	FLNOFN	
414	466	FLNOFN	1	GOSUB	GOSUB
414	467		0		
415	470		0	XDEF	APERMG
416	471		6	CON	006
417	472		14	CON	014
418	473		40	CON	040
419	474		16	CON	016
420	475		17	CON	017
421	476		24	CON	024
422	477		40	CON	040
423	500		6	CON	006
424	501		17	CON	017
425	502		25	CON	025
426	503		16	CON	016
427	504		1004	CON	01004
428	505		1	GOSUB	GOLONG
428	506		0		
429	507		0	XDEF	APEREX
430	510	EFLS90	34	PT=	3
431	511		460	LDI	
432	512		100	CON2	4 0
433	513		1160	DADD=C	
434	514		114	?S4=1	
435	515		127	GOC	EFLS92 (527)
436	516		256	C=A	W
436	517		416		
437	520		674	ROR	11
438	521		406	A=C	X
439	522		70	C=DATA	
440	523		1174	ROR	9
441	524		246	AC EX	X
442	525		274	ROR	5
443	526		1360	DATA=C	

LOOKING FOR CURRENT FILE ?

YES

C.X = FILE #

SAVE CURRENT # IN REG.40(11:9)

*
*
*
*

IF THE NEW FILE IS STARTING AT REG.BF, IT WILL WRITE
000010000000BF TO REG.40

BASE REG. INITIALIZED ?
YES

WRITE 000010000000BF TO REG.40

449	527	EFLS92	1414	?S1=1	
449	530		1640	RTN NC	
450	531		116	C=0	W
451	532		1056	C=C+1	W
452	533		274	ROR	5
453	534		460	LDI	
454	535		277	CON2	11 15
455	536		1360	DATA=C	
456	537		1740	RTN	

*

* DELREC - DELETE CURRENT RECORD FROM CURRENT TEXT FILE *

* DELCHR - DELETE X NUMBER OF CHARACTERS FROM CURRENT RECORD *
 * STARTING FROM CURRENT POINTER *

```

*
464          ENTRY  DELREC
465          ENTRY  DELCHR

*
467 540      222 CON   @222      R
468 541      10 CON   @10       H
469 542      3  CON   @03       C
470 543      14 CON   @14       L
471 544      5  CON   @05       E
472 545      4  CON   @04       D
473 546 DELCHR 1210 S7=      1
474 547      103 GOTO DLRC10 ( 557)

*
476 550      203 CON   @203      C
477 551      5  CON   @05       E
478 552      22 CON   @22       R
479 553      14 CON   @14       L
480 554      5  CON   @05       E
481 555      4  CON   @04       D
482 556 DELREC 1204 S7=      0
483 557 DLRC10      1 GOSUB  GOSUB
483 560          0
484 561          0 XDEF  CURFLT      GET CURRENT FILE
485 562      210 S5=      1
486 563      410 S8=      1      DON'T GO TO NEXT REC WHEN AT END OF F
487 564          1 GOSUB  GOSUB1
487 565          0
488 566          0 XDEF  CUREC#
489 567      1614 ?S0=1      REACHED END OF FILE ?
490 570      43  GONC  DLRC20 ( 574) NO
491 571          1 GOSUB  GOLONG
491 572          0
492 573          0 XDEF  EOFL      SAY "END OF FL"
493 574 DLRC20 1214 ?S7=1      FOR "DELCHR" ?
494 575      613 GONC  DLRC50 ( 656) NO, IS FOR "DELREC"
495 576      116 C=0
496 577      1160 DADD=C
497 600          1 GOSUB  GOSUB0
497 601          0
498 602          0 XDEF  X<999      GET INTEGER OF X
499 603      630 C=M      SAVE INT(X) IN M.X
500 604      246 AC EX  X
501 605      530 M=C
502 606      316 C=B
503 607      1274 RCR      7
504 610      406 A=C      X      A.X= CURRENT RECORD LENGTH
505 611      260 C=N
506 612      574 RCR      6      C.X= CURRENT CHAR POINTER
507 613      706 A=A-C  X      A.X= # OF REMAINING CHARS IN REC
508 614      346 BC EX  X      B.X= CURRENT CHAR POINTER
509 615      630 C=N      C.X= # OF CHARS TO DELETE
510 616      246 AC EX  X      A.X= # TO DEL  C.X= # REMAINING
511 617      1406 ? A<C  X      DEL TO END OF REC ?
512 620      47  GOC   DLRC30 ( 624) NO
513 621      1706 ? B#0  X      STARTING FROM CHAR ZERO ?
514 622      343 GONC  DLRC50 ( 656) YES, DELETE ENTIRE RECORD
515 623      406 A=C      X      ONLY DELETE REMAINING CHARS

```

516	624	DLRC30	206	B=A	X	B.X= # OF CHARS TO DELETE
517	625		246	AC EX	X	A.X= # TO DEL; C.X= # LEFT
518	626		1106	C=A-C	X	C.X= CHARS TO END OF REC
519	627		160	N=C		SAVE IN N.X
520	630		630	C=M		SAVE # TO DEL IN M.X
521	631		306	C=B	X	
522	632		406	A=C	X	# TO DEL BACK TO A.X
523	633		530	M=C		
524	634		316	C=B	W	
525	635		1274	ROR	7	C.X= CURRENT RECORD LENGTH
526	636		246	AC EX	X	
527	637		706	A=A-C	X	A.X= # OF CHAR LEFT IN REC
528	640		74	ROR	3	C[3:0]= CURRENT RECORD ADDR
529	641		252	AC EX	WPT	
530	642		11	GOSUB	PTBYTA	UPDATE RECORD LENGTH
530	643		0			
531	644		630	C=M		C.X= # OF CHARS TO DELETE
532	645		346	BC EX	X	
533	646		474	ROR	8	
534	647		412	A=C	WPT	A[3:0]= CURRENT POINTER ADDR
535	650		1	GOSUB	GOSUB1	
535	651		0			
536	652		0	XDEF	ADVREB	SKIP OVER THE DELETED STRING
537	653		630	C=M		
538	654		174	ROR	4	C[7:4]= CURRENT POINTER ADDR
539	655		213	GOTO	DLRC60 (676)	
540	656	DLRC50	260	C=N		
541	657		574	ROR	6	SET CURRENT CHAR POINTER TO ZERO
542	660		106	C=0	X	
543	661		160	N=C		SET N.X = 0 FOR # CHARS TO END OF F I
544	662		174	ROR	4	
545	663		1160	DADD=C		ADDRESS HEADER REGISTER
546	664		174	ROR	4	
547	665		1360	DATA=C		UPDATE FILE POINTER
548	666		316	C=B	W	
549	667		374	ROR	10	C[3:0] = CURRENT RECORD ADDR
550	670		412	A=C	WPT	
551	671		1	GOSUB	GOSUB1	
551	672		0			
552	673		0	XDEF	ADVREC	POINT TO NEXT REC
553	674		316	C=B	W	
554	675		574	ROR	6	C[7:4]= CURRENT POINTER ADDR
555	676	DLRC60	252	AC EX	WPT	
556	677		416	A=C	W	
* NOW A[3:0] = BYTE ADDR OF NEXT PICK UP BYTE						
* A[7:4] = BYTE ADDR OF NEXT STORING BYTE						
* N.X = # OF CHARS TO END OF RECORD						
*						
561	700	DELB10	1604	S0=	0	NOT END OF FILE YET
562	701		1	GOSUB	GTBYTA	GET NEXT RECORD LENGTH
562	702		0			
563	703		360	CN EX		GET # OF CHARS LEFT IN REC
564	704		1146	C=C-1	X	DONE WITH RECORD YET ?
565	705		113	GONC	DELB15 (716)	NO, CONTINUE
566	706		260	C=N		
567	707		126	C=0	XS	C.X= # OF CHARS IN NEXT REC
568	710		1434	PT=	1	
569	711		1052	C=C+1	WPT	END OF FILE ?
570	712		23	GONC	DELB14 (714)	NO
571	713		1610	S0=	1	REMEMBER END OF FILE

```

572 714 DEL814 1152 C=C-1 WPT RESTORE RECORD LENGTH
573 715 34 PT= 3
574 716 DEL815 360 CN EX CHRS LEFT TO N; NEXT BYTE TO C
575 717 256 AC EX W
576 720 174 RCR 4
577 721 256 AC EX W A[3:0] = NEXT STORING ADDR
578 722 1 GOSUB PTBYTA STORE THE BYTE
578 723 0
579 724 1614 ?S0=1 REACHED END OF FILE MARK ?
580 725 1540 RTN C YES, ALL DONE
* NOW TRY TO ADVANCE BOTH NEXT PICK UP AND NEXT STORE ADDR
582 726 DEL830 642 A=A-1 PT
587 727 223 GONC DELB60 ( 751 ) STILL IN THE SAME REG
584 730 646 A=A-1 X
585 731 460 LDI
586 732 100 CON2 4 0
587 733 1526 ? A#0 XS ARE WE IN BASE MODULE ?
587 734 53 GONC DELB40 ( 741 ) YES
587 735 460 LDI
590 736 1 CON 01
591 737 266 C=A XS
591 740 426
592 741 DELB40 1546 ? A#C X ABOUT TO TURN OVER TO NEXT MODULE?
593 742 57 GOC DELB50 ( 747 ) NOT YET
594 743 1160 DADD=C
595 744 70 C=DATA
596 745 74 RCR 3 C.X = NEXT MODULE ADDR
597 746 406 A=C X
598 747 DEL850 642 A=A-1 PT FINISH CHANGING BYTE NUMBER
599 750 642 A=A-1 PT
600 751 DELB60 642 A=A-1 PT
601 752 1614 ?S0=1 DONE WITH BOTH ADDR ?
602 753 1257 GOC DELB10 ( 700 ) YES, MOVE ANOTHER BYTE
607 754 256 AC EX W GET READY TO ADJUST FIRST ADDRESS
604 755 374 RCR 10
605 756 256 AC EX W
606 757 1610 S0= 1 FLAG FOR SECOND ADJUSTMENT
607 760 1463 GOTO DELB30 ( 726 )

```

*

* PURFL - PURGE AN EXTERNAL MEMORY FILE *

* THE FILE NAME IS TAKEN FROM ALPHA REGISTER *

*

```

614 ENTRY PURFL
615 ENTRY PUFLSB
616 761 214 CON @214 L
617 762 6 CON @06 F
618 763 22 CON @22 R
619 764 25 CON @25 U
620 765 20 CON @20 P
621 766 PURFL 640 CLRABC GUARANTEE WILL NOT SAY "NO ROOM" AND
622 767 1150 REGN=C 9 DON'T CHECK FILE TYPE
623 770 1 GOSUB GOSUB2
623 771 0
624 772 0 XDEF FLSHAC

```

*

* PUFLSB - PURGE FILE SUBROUTINE

* PURGING A FILE WITHOUT AN END OF MEMORY MARK(FFFFFFFF...) IN THE

```

* MEMORY WILL CAUSE PART OF THE LAST FILE TO BE FILLED WITH ZEROS.
* IN ORDER TO PREVENT THIS, WE ALWAYS WRITE THE END OF MEMORY MARK
* TO THE 1ST UNUSED REG BEFORE PURGING A FILE. THIS ROUTINE IS
* ALSO USED BY THE FUNCTION "SAVEP". WHEN IT IS CALLED BY "SAVEP",
* IT WILL EITHER PURGE THE OLD FILE, OR IT WILL SAY "NO ROOM" IF
* AFTER PURGING THE OLD FILE THERE STILL WON'T BE ENOUGH ROOM TO
* PUT IN THE NEW FILE. TO MAKE SURE THAT THE "PURFL" FUNCTION WILL
* NEVER SAY "NO ROOM", REG.9 IS CLEARED AT THE BEGINNING OF "PURFL".
* INPUT : A[10:8] = FILE NAME ADDR OF THE FILE TO BE PURGED
*         N       = FILE HEADER
*         REG.9[2:0] = # OF REGISTERS NEEDED FOR A NEW FILE
*
641 773 PUFLSB 216 B=A      W      SAVE START REG ADDR OF THE FL IN B
642 774      260 C=N
643 775      132 C=0      M      MAKE SURE THAT THE NAME WON'T BE FOUR
644 776      530 M=C      SAVE FILE HEADER IN M
645 777      10 S3=      1
646 1000      1 GOSUB    GOSUB
647 1001      0
648 1002      0 XDEF     EFLSCH  FIND OUT AVAILABLE REGS
649 1003      630 C=M      C.X= OLD FILE SIZE
650 1004      156 AB EX   W      A.X= # OF REGS UNUSED
651 1005      506 A=A+C   X      A.X= TOTAL # OF REGS AVAILABLE
652 1006      1046 C=C+1  X
653 1007      1046 C=C+1  X      C.X = FILE SIZE + 2
654 1010      102 C=0     PT
655 1011      352 BC EX   WPT    B[3:0] = FILE SIZE + 2
656 1012      116 C=0
657 1013      1160 DADD=C
658 1014      1170 C=REGN 9      C.X= NEW PROG SIZE
659 1015      1406 ? A<C   X      ENOUGH ROOM TO PUT IN THE NEW ONE ?
660 1016      43 GONC     PUFL10 (1022) YES
661 1017      1 GOSUB     GOL0    SAY "NO ROOM"
662 1020      0
663 1021      0 XDEF     NORM
* NOW A[10:8] = ADDR OF FILE NAME OF THE FILE
* B[10:8] = ADDR OF THE LAST REG OF EX.MEM
* B[3:0] = FILE SIZE + 2
* M       = FILE HEADER
*
664 1022 PUFL10 1 GOSUB    GOSUB2
665 1023      0
666 1024      0 XDEF     CUR#=0  SET CURRENT FILE # TO ZERO
667 1025      256 C=A      W
668 1026      416
669 1027      474 RCR      8
670 1030      412 A=C      WPT    A[3:0]= ADDR OF FILE NAME(1ST REG OF
671 1031      1 GOSUB     GOSUB1  PASS OVER THE CURRENT FILE
672 1032      0
673 1033      0 XDEF     ADVADR  POINT TO NEXT FILE NAME
674 1034      272 AC EX   M      MOVE A[10:8] TO B.X
675 1035      474 RCR      8
676 1036      346 BC EX   X
* NOW B.X = NEXT STORING REG ADDR
* A.X = NEXT PULL UP REG ADDR
* B[10:8] = ADDRESS OF END OF MEMORY MARK
*
682 1037 PAKEXM 246 AC EX   X      C.X = NEXT PULL UP ADDR
683 1040      1160 DADD=C

```

NOMAS

Not Manufacturer Supported
recipient agrees NOT to contact manufacturer

684	1041	246	AC EX	X		A.X = NEXT PULL UP ADDR
685	1042	70	C=DATA			
686	1043	356	BC EX	W		SAVE NEXT PULL UP REG & GET STORING A
687	1044	1160	DADD=C			
688	1045	356	BC EX	W		
689	1046	1360	DATA=C			PULL UP ONE REG
690	1047	316	C=B	W		
691	1050	474	ROR	8		C.X = END OF MEMORY MARK ADDRESS
692	1051	1546	?A#C	X		IS THIS END OF DIR MARK ?
693	1052	1640	RTN NC			IF SO, ALL DONE
694	1053	1604	S0=	0		
695	1054	PKRG20	646	A=A-1	X	
696	1055		460	LDI		
697	1056		100	CON2	4	0
698	1057	1526	? A#0	XS		ARE WE IN BASE MODULE ?
699	1060		53	GONC	PKRG30	(1065) YES
700	1061		460	LDI		
701	1062		1	CON2	0	1
702	1063		266	C=A	XS	
702	1064		426			
703	1065	PKRG30	1546	? A#C	X	REACHED LAST REG IN THE MODULE ?
704	1066		47	GOC	PKRG40	(1072) NO
705	1067		1	GOSUB	GOSUB1	
705	1070		0			
706	1071		0	XDEF	NXTMDL	GO OVER TO NEXT MODULE
707	1072	PKRG40	146	AB EX	X	EXCHANGE B.X & A.X
708	1073		1614	?S0=1		UPDATE BOTH ADDR YET ?
709	1074		1437	GOC	PAKEXM	(1037) YES
710	1075		1610	S0=	1	
711	1076		1563	GOTO	PKRG20	(1054) UPDATE STORING ADDR

* GETRX - DOWN LOAD A BLOCK OF REGS FROM CURRENT WORKING REGISTER *
 * FILE TO RESIDENT MEMORY. THE STARTING POINT IN THE WORKING *
 * FILE IS THE CURRENT REG POINTER. THE BLOCK OF THE RESIDENT *
 * REGS IS SPECIFIED BY THE X REGISTER IN BBB.EEE FORM. *
 * SAVERX - SIMILAR TO "GETRX" EXCEPT THE DIRECTION OF THE FLOW IS *
 * REVERSED. *

722		ENTRY	SAVERX	
723		ENTRY	GETRX	
725	1077	230	CON	@230 X
726	1100	22	CON	@22 R
727	1101	5	CON	@05 E
728	1102	26	CON	@26 V
729	1103	1	CON	@01 A
730	1104	23	CON	@23 S
731	1105	SAVERX	1210	S7= 1
732	1106		73	GOTO GETR10 (1115)
734	1107	230	CON	@230 X
735	1110	22	CON	@22 R
736	1111	24	CON	@24 T
737	1112	5	CON	@05 E
738	1113	7	CON	@07 G
739	1114	GETRX	1204	S7= 0
740	1115	GETR10	1	GOSUB GOSUB
740	1116		0	

```

741 1117      0 XDEF   CURFLR      GET CURRENT REGISTER FILE
742 1120      260 C=N
743 1121      530 M=C              SAVE FILE HEADER IN M TEMP.
744 1122      210 S5=      1
745 1123      1 GOSUB   GOSUB1
745 1124      0
746 1125      0 XDEF   GTINDX      GET BBB.EEE FROM X
747 1126      260 C=N
748 1127      406 A=C      X        END ADDRESS TO A.X
749 1130      74 RCR      3        START ADDRESS TO C.X
750 1131      706 A=A-C    X        A.X <- # OF REGS -1
751 1132      23 GONC     GETR15 (1134) IF END > START, THEN SKIP
752 1133      6 A=0      X        SETTING # REGS = 1
753 1134 GETR15 674 RCR      11      PUT BLOCK SIZE IN C
754 1135      246 AC EX   X
755 1136      406 A=C      X
756 1137      674 RCR      11      ROTATE C BACK
757 1140      730 CM EX           SAVE IN M
758 1141      160 N=C           PREVIOUS M -> N
* NOW M[5:3]= # OF REGS -1, M[8:6]= R.M. STARTING REG ADDR
*   N[2:0]= FILE SIZE IN REGS, N[5:3]= CURRENT REG # IN FILE
*   N[12:10]= FILE HEADER ADDR, A.X= # OF REGS -1
762 1142      346 BC EX   X        B.X = FILE SIZE
763 1143      74 RCR      3        C.X= CURRENT REG #
764 1144      506 A=A+C    X        A.X= LAST REG #
765 1145      1446 ? A<B   X        ENOUGH ROOM ?
766 1146      623 GONC     SAVRER (1230) NO, SAY "END OF FL"
767 1147      1046 C=C+1   X        C.X= CURPENT REG # + 1
768 1150      346 BC EX   X        B.X= CURRENT REG # + 1
769 1151      630 C=M
770 1152      246 AC EX   X
771 1153      530 M=C        M.X= CURRENT LAST REG #
772 1154      260 C=N
773 1155      374 RCR      10
774 1156      246 AC EX   X        A.X=FL.HED.ADDR
775 1157      34 PT=      3
776 1160      42 B=0      PT
777 1161      1 GOSUB   GOSUB1
777 1162      0
778 1163      0 XDEF   ADVADR      POINT TO START REG ADDR IN E.M.
779 1164      630 C=M
780 1165      356 BC EX   W
* NOW R[8:6]= STARTING REG ADDR IN RESIDENT MEMORY
*   R[5:3]= # OF REGS -1
*   R[2:0]= CURRENT LAST REG #
784 1166      633 GOTO    SAVR20 (1251)
785
*
*****
* SAVER - SAVE ALL RESIDENT REGISTERS TO A GIVEN REGISTER FILE *
* GETR - LOAD UP THE RESIDENT REGISTERS FROM A GIVEN REGISTER FILE. *
*       THE FILE NAME IS TAKEN FROM THE ALPHA REGISTER. *
*       IF THE FILE NAME IS EMPTY, IT WILL DEFAULT TO CURRENT FILE.*
*****
*
794          ENTRY   SAVER
795          ENTRY   GETR
*
797 1167      222 CON      @222      R
798 1170      24 CON      @24        T

```

799	1171		5 CON	005	E
800	1172		7 CON	007	G
801	1173	GETR	1204 S7=	0	
802	1174		73 GOTO	SAVR05 (1203)	
*					
804	1175		222 CON	0222	R
805	1176		5 CON	005	E
806	1177		26 CON	026	V
807	1200		1 CON	001	A
808	1201		23 CON	023	S
809	1202	SAVER	1210 S7=	1	
810	1203	SAVR05	1334 PT=	13	
811	1204		220 LC	2	TYPE OF DATA FILE = 2
812	1205		376 BC EX	S	
813	1206		1 GOSUB	GOSUB2	SEARCH CURRENT FILE OR THE GIVEN FILE
813	1207		0		
814	1210		0 XDEF	FLSHAB	AND CHECK ITS TYPE(DATA FILE)
815	1211		1 GOSUB	FNDEND	FIND 1ST NONEXIST REG ADDR
815	1212		0		
816	1213		116 C=0	W	
817	1214		1160 DADD=C		
818	1215		1570 C=REGN	13	
819	1216		74 RCR	3	C.X = REG 0 ADDR
820	1217		706 A=A-C	X	A.X=# OF RESIDENT REGS
821	1220		646 A=A-1	X	
822	1221		1540 RTN C		SIZE = 0, DON'T DO ANYTHING
823	1222		346 BC EX	X	SAVE REG0 ADDR IN B.X
824	1223		260 C=N		C.X = FILE SIZE
825	1224		1406 ? ACC	X	WILL IT FIT ?
826	1225		107 GOC	SAVR10 (1235)	YES
827	1226		1214 ?S7=1		FOR "GETR"
828	1227		43 GONC	SAVR07 (1233)	YES
829	1230	SAVRER	1 GOSUB	GOLONG	
829	1231		0		
830	1232		0 XDEF	EOFL	
831	1233	SAVR07	1146 C=C-1	X	JUST READ TO END OF FILE
832	1234		246 AC EX	X	A.X= LAST REG #
833	1235	SAVR10	306 C=B	X	C.X= REG 0 ADDR
834	1236		206 B=A	X	B.X= LAST REG NUMBER
835	1237		674 RCR	11	
836	1240		246 AC EX	X	C.X = # OF REGS -1
837	1241		674 RCR	11	
838	1242		372 BC EX	M	B[8:6]=REG0 B[5:3]=# OF REGS -1
839	1243		260 C=N		
840	1244		374 RCR	10	
841	1245		406 A=C	X	A.X = FILE HEADER ADDR
842	1246		1 GOSUB	GOSUB1	POINT TO START REG ADDR
842	1247		0		
843	1250		0 XDEF	NXREG	
844	1251	SAVR20	172 AB EX	M	
845	1252		260 C=N		
846	1253		374 RCR	10	
847	1254		1160 DADD=C		
848	1255		1274 RCR	7	C.X= CURRENT REG #
849	1256		306 C=B	X	C.X= LAST REG #
850	1257		1046 C=C+1	X	UPDATE CURRENT REG #
851	1260		674 RCR	11	
852	1261		1360 DATA=C		
853	1262		256 AC EX	W	
854	1263		160 N=C		

```

855 1264      204 S5=      0
856 1265      104 S4=      0
857 1266      1214 ?S7=1      FOR SAVE ?
858 1267      127 GOC      SAVR50 (1301) YES,
859 1270      246 AC EX      X      EXCHANGE THE SOURCE & TARGET ADDR
860 1271      574 RCR      6
861 1272      246 AC EX      X
862 1273      474 RCR      8
863 1274      246 AC EX      X
864 1275      160 N=C
865 1276      1404 S1=      0
866 1277      1610 S0=      1
867 1300      673 GOTO      RGMV      (1367)
868 1301 SAVR50 1410 S1=      1
869 1302      1604 S0=      0
870 1303      643 GOTO      RGMV      (1367)

```

```

*
*****
* REGMOV - MOVE A BLOCK OF REGISTERS TO ANOTHER DATA REGISTER SPACE *
* THE REGISTER INDEX IS TAKEN FROM X IN SSS.DDDNNN FORM, WHERE SSS *
* IS THE SOURCE REG #, DDD IS THE DESTINATION REG #, NNN IS THE *
* NUMBER OF REGS TO MOVE. *
* REGSWAP-EXCHANGE TWO REGISTER BLOCKS. THIS FUNCTION TAKES THE *
* SAME ARGUMENT FROM X REGISTER AND PERFORMS A SIMILAR OPERATION; *
* IT WILL EXCHANGE THE TWO BLOCKS RATHER THAN JUST COPY FROM ONE TO *
* ANOTHER. *
*****

```

```

*
883      ENTRY  REGMOV
884      ENTRY  REGSWP
*
886 1304      220 CON      @220      P
887 1305      1 CON      @01      A
888 1306      27 CON      @27      W
889 1307      23 CON      @23      S
890 1310      7 CON      @07      G
891 1311      5 CON      @05      E
892 1312      22 CON      @22      R
893 1313 REGSWP 110 S4=      1      SET EXCHANGE FLAG
894 1314      113 GOTO      REGM10 (1325)
*
896 1315      205 CON      @205      E
897 1316      26 CON      @26      V
898 1317      17 CON      @17      O
899 1320      15 CON      @15      M
900 1321      7 CON      @07      G
901 1322      5 CON      @05      E
902 1323      22 CON      @22      R
903 1324 REGMOV 104 S4=      0      CLEAR EXCHANGE FLAG
*
905 1325 REGM10 204 S5=      0      SET REG INDEX FLAG
906 1326      1 GOSUB      GOSUB1
906 1327      0
907 1330      0 XDEF      GTINDX
* NOW N[2:0] = # OF REGS TO MOVE
* N[5:3] = DESTINATION REG ADDRESS
* N[8:6] = SOURCE REG ADDRESS
* A.X = REG 0 ADDR
* B.X = 1ST NONEXIST REG ADDR
* C=N

```

914	1331	1346 ? C#0	X	NNN = 0 ?
915	1332	23 GONC	REGM20 (1334)	YES, DEFAULT TO 1
916	1333	1146 C=C-1	X	C.X= # OF REGS -1
917	1334	160 N=C		N.X= # OF REGS -1
918	1335	406 A=C	X	
919	1336	74 RCR	3	C.X= 1ST DESTINATION REG ADDR
920	1337	1006 C=A+C	X	C.X= LAST DESTINATION REG ADDR
921	1340	74 RCR	3	C.X= 1ST SOURCE REG ADDR
922	1341	1006 C=A+C	X	C.X= LAST DESTINATION REG ADDR
923		LEGAL		
924	1342	1 GOSUB	CHKADR	SEE IF LAST REG EXIST ?
924	1343	0		
925	1344	674 RCR	11	C.X= LAST DESTINATION REG ADDR
926	1345	1 GOSUB	CHKADR	SEE IF THAT REG EXIST
926	1346	0		
927	1347	406 A=C	X	A.X=LAST DESTINATION REG ADDR
928	1350	74 RCR	3	C.X=LAST SOURCE REG ADDR
929	1351	1406 ? ACC	X	
930	1352	47 GOC	REGM30 (1356)	START FROM BOTTOM & INCREMENT ADDR
931	1353	210 S5=	1	
932	1354	474 RCR	8	
933	1355	160 N=C		START FROM TOP & DECREMENT ADDR
934	1356	260 C=N		
935	1357	416 A=C	W	SWITCH N[5:3] WITH N[2:0]
936	1360	74 RCR	3	
937	1361	246 AC EX	X	
938	1362	674 RCR	11	
939	1363	246 AC EX	X	
940	1364	160 N=C		
941	1365	1604 S0=	0	
942	1366	1404 S1=	0	SAY SOURCE & DEST. ALL IN MAIN MEMORY
* FALL INTO "RGMV" ROUTINE FROM HERE				

* RGMV - MOVES A BLOCK OF REGISTERS TO ANOTHER LOCATION				
* INPUT : N[2:0] = STARTING REG ADDR OF DESTINATION				
* N[5:3] = # OF REGS TO MOVE -1				
* N[8:6] = STARTING REG ADDR OF SOURCE				
* IF S0 = 0 - SOURCE IS IN MAIN MEMORY				
* = 1 - SOURCE IS IN EXTERNAL MEMORY				
* S1 = 0 - DESTINATION IS IN MAIN MEMORY				
* = 1 - DESTINATION IS IN EXTERNAL MEMORY				
* S4 = 0 - ONLY MOVE SOURCE TO DESTINATION				
* = 1 - EXCHANGE SOURCE AND DESTINATION				
* S5 = 0 - START FROM LOWEST REG ADDR AND INCREMENT				
* = 1 - START FROM HIGHEST REG ADDR AND DECREMENT				
* (USED ONLY IN REGMOVE AND REGSWAP)				
* OUTPUT : IF S2 = 1 - DISCONTINUE MEMORY DETECTED				
* USED A,B,C,M,PT,S2 +1 SUB LEVEL				
* 962 ENTRY RGMV				
* 259 964 1367 RGMV 1004 S2= 0				
965 1370 260 C=N				
966 1371 574 RCR 6 C.X= NEXT SOURCE REG ADDR				
967 1372 1160 DADD=C				
968 1373 70 C=DATA LOAD NEXT SOURCE REG				
969 1374 356 BC EX W SAVE IN B				
970 1375 260 C=N C.X= NEXT DESTINATION REG ADDR				
971 1376 1160 DADD=C				

972	1377	70	C=DATA		LOAD NEXT DEST. REG	
973	1400	356	CB EX	W		
974	1401	1360	DATA=C		PUT SOURCE TO DESTINATION	
975	1402	114	?S4=1		EXCHANGE ?	
976	1403	63	GONC	RGMV10 (1411)	NO	
977	1404	260	C=N			
978	1405	574	RCR	6		
979	1406	1160	DADD=C		SELECT SOURCE AGAIN	
980	1407	316	C=B	W	GET DEST. REG FROM B	
981	1410	1360	DATA=C			
982	1411	RGMV10	260	C=N		
983	1412	74	RCR	3	C.X= REGS COUNT	
984	1413	1146	C=C-1	X	DECREMENT REGS COUNT	
985	1414	1540	RTN C		ALL DONE	
986	1415	74	RCR	3	C.X= CURRENT SOURCE ADDR	
987	1416	4	S3=	0	FLAG FOR SOURCE ADDRESS UPDATE	
988	1417	1614	?S0=1		SOURCE IN MAIN MEMORY ?	
989	1420	177	GOC	RGMV20 (1437)	NO, IN EXTERNAL MEMORY	
990	1421	RGMV12	214	?S5=1	START FROM TOP TO BOTTOM ?	
991	1422	33	GONC	RGMV13 (1425)	NO, BOTTOM TO TOP	
992	1423	1146	C=C-1	X	DECREMENT REGISTER ADDRESS	
993			LEGAL			
994	1424	23	GOTO	RGMV15 (1426)		
995	1425	RGMV13	1046	C=C+1	X	INCREMENT REGISTER ADDRESS
996	1426	RGMV15	160	N=C		
997	1427	1014	?S2=1		MEMORY DISCONTINUITY ?	
998	1430	1540	RTN C		YES	
999	1431	14	?S3=1		DEST. ADDRESS UPDATED YET ?	
1000	1432	1357	GOC	RGMV (1367)	YES	
1001	1433	474	RCR	8	C.X= CURRENT DEST. ADDRESS	
1002	1434	10	S3=	1	FLAG FOR DEST. ADDRESS UPDATE	
1003	1435	1414	?S1=1		DEST. REG IN EXT. MEMORY ?	
1004	1436	1633	GONC	RGMV12 (1421)	NO	
1005	1437	RGMV20	160	N=C		
1006	1440	416	A=C	W		
1007	1441	1	GOSUB	GOSUB1		
1007	1442	0				
1008	1443	0	XDEF	ADVAD1	ADVANCE ONE REG IN EXT. MEMORY	
1009	1444	260	C=N		GET ADDR BACK FROM N	
1010	1445	246	AC EX	X		
1011	1446	1603	GOTO	RGMV15 (1426)		

*

* BYTLFT - COMPUTE # OF BYTES LEFT AT THE END UNUSED

* INPUT : N= FILE HEADER

* A[3:0]= ADDR OF END OF RECORD MARK

* OUTPUT : A[3:0]= # OF BYTE AVAILABLE

* A[11:8]= ADDR OF CURRENT END OF RECORD MARK

* USED A[11:0], B[3:0], C +2 SUB LEVEL

*

		ENTRY	BYTLFT	
1021				
1022	1447	BYTLFT	256	AC EX W
1023	1450		574	RCR 6
1024	1451		416	A=C W
1025	1452		260	C=N
1026	1453		346	BC EX X
1027	1454		42	B=0 PT
1028	1455		374	RCR 10
1029	1456		406	A=C X
1030	1457		2	A=0 PT

SAVE END ADDR IN A[12:8]

B.X= FILE SIZE IN REGS

A.X= FILE HEADER ADDR

```

1031 1460          1 GOSUB  GOSUB1
1031 1461          0
1032 1462          0 XDEF   ADVADR      GET LAST REG ADDR
1033 1463        212 B=A     WPT
1034 1464        256 C=A     W
1034 1465        416
1035 1466        474 RCR     8
1036 1467        412 A=C     WPT      A[3:0]= END OF RECORD MARK ADDR
1037 1470          1 GOSUB  GOL0
1037 1471          0
1038 1472          0 XDEF   CNTBYE      COMPUTE # OF BYTES BETWEEN

```

*
*

```

*****
* SEEKPTA - SET THE POINTER OF A DATA FILE TO A GIVEN POINT *
* THE FILE NAME IS TAKEN FROM ALPHA REGISTER AND THE POINTER IS *
* FROM X REGISTER. IF ALPHA IS EMPTY, IT WILL DEFAULT TO CURRENT *
* WORKING DATA FILE. *
* SEEKPT - SAME AS "SEEKPTA" EXCEPT IT WILL SET THE POINTER IN THE *
* CURRENT WORKING DATA FILE. *
*****

```

*

```

1050          ENTRY  SEKPTA
1051          ENTRY  SEKPT
*
1053 1473        224 CON    @224      T
1054 1474        20 CON    @20       P
1055 1475        13 CON    @13       K
1056 1476         5 CON    @05       E
1057 1477         5 CON    @05       E
1058 1500        23 CON    @23       S
1059 1501 SEKPT  1604 S0=    0
1060 1502        113 GOTO   SKPT10 (1513)

```

*

```

1062 1503        201 CON    @201     A
1063 1504        24 CON    @24       T
1064 1505        20 CON    @20       P
1065 1506        13 CON    @13       K
1066 1507         5 CON    @05       E
1067 1510         5 CON    @05       E
1068 1511        23 CON    @23       S
1069 1512 SEKPTA 1610 S0=    1
1070 1513 SKPT10  1 GOSUB  GOSUB2
1070 1514         0
1071 1515         0 XDEF   FLSHAP
1072 1516        116 C=0
1073 1517        1160 DADD=C
1074 1520         1 GOSUB  GOSUB0
1074 1521         0
1075 1522         0 XDEF   X<999      GET INTEGER OF X IN BINARY
1076 1523        374 RCR     10
1077 1524        372 BC EX   M      B[6:4]= GIVEN POINTER
1078 1525         36 A=0     S
1079 1526        576 A=A+1   S
1080 1527        576 A=A+1   S
1081 1530        260 C=N
1082 1531        374 RCP     10      C,X= FILE HEADER ADDR
1083 1532        1160 DADD=C      ENABLE FILE HEADER REG
1084 1533        1274 RCR     7      SAVE GIVEN REC PTR IN N[5:3]
1085 1534        246 C=A     X

```

1085	1535	406		
1086	1536	74 RCR	3	
1087	1537	106 C=0	X	SET CHAR POINTER TO ZERO
1088	1540	474 RCR	8	
1089	1541	160 N=C		
1090	1542	1576 ? A#C	S	IS THIS A REGISTER FILE ?
1091	1543	247 GOC	SKPT50 (1567)	NO
1092	1544	1406 ? AKC	X	POINTER < FILE SIZE ?
1093	1545	207 GOC	SKPT20 (1565)	YES

*

1095		ENTRY	EOFL	
1096	1546	EOFL	1 GOSUB	GOSUB
1096	1547	0		
1097	1550	0 XDEF	APERMG	SAY " END OF FILE"
1098	1551	5 CON	Q05	E
1099	1552	16 CON	Q16	N
1100	1553	4 CON	Q04	D
1101	1554	40 CON	Q40	
1102	1555	17 CON	Q17	O
1103	1556	6 CON	Q06	F
1104	1557	40 CON	Q40	
1105	1560	6 CON	Q06	F
1106	1561	1014 CON	Q1014	L
1107	1562	1 GOSUB	GOLONG	
1107	1563	0		
1108	1564	0 XDEF	APEREX	
1109	1565	SKPT20 1360	DATA=C	
1110	1566	663 GOTO	SKPT80 (1654)	

*

1112	1567	SKPT50	576 A=A+1	S	A.S=3
1113	1570		1576 ? A#C	S	IS THIS A TEXT FILE ?
1114	1571		43 GONC	SKPT60 (1575)	YES
1115	1572		1 GOSUB	GOLONG	
1115	1573		0		
1116	1574		0 XDEF	FLTPER	SAY "FL TYPE ERR"
1117	1575	SKPT60	210 S5=	1	
1118	1576		510 S6=	1	
1119	1577		1 GOSUB	GOSUB1	
1119	1600		0		
1120	1601		0 XDEF	TOREC#	POINT TO THE GIVEN RECORD
1121	1602		316 C=B		
1122	1603		530 M=C		MI(9:7)= RECORD LENGTH
1123	1604		116 C=0		
1124	1605		1160 DADD=C		
1125	1606		370 C=REGN	3	
1126	1607		416 A=C	W	
1127	1610		1614 ?S0=1		REACHED END OF FILE YET ?
1128	1611		43 GONC	SKPT65 (1615)	NO
1129	1612		1356 ? C#0	W	SEEK TO POINTER 0.0 ?
1130	1613		1337 GOC	EOFL (1546)	NO, SAY "END OF FL"
1131	1614		313 GOTO	SKPT70 (1645)	ALWAYS ALLOW SEEK TO 0.0
1132	1615	SKPT65	1 GOSUB	GOSUB1	
1132	1616		0		
1133	1617		0 XDEF	GTFRA	GET FRACTION PART OF X
1134	1620		406 A=C	X	A.X = GIVEN CHAR POINTER
1135	1621		630 C=M		
1136	1622		1274 RCR	7	C.X = RECORD LENGTH
1137	1623		1406 ? AKC	X	CHAR PTR PAST END OF RECORD ?
1138	1624		217 GOC	SKPT70 (1645)	NO, SET IT
1139	1625		1 GOSUB	GOSUB	

```

1139 1626      0
1140 1627      0 XDEF  APERMG
1141 1630      5 CON   @05
1142 1631     16 CON   @16
1143 1632      4 CON   @04
1144 1633     40 CON   @40
1145 1634     17 CON   @17
1146 1635      6 CON   @06
1147 1636     40 CON   @40
1148 1637     22 CON   @22
1149 1640      5 CON   @05
1150 1641    1003 CON   @1003
1151 1642      1 GOSUB GOLONG
1151 1643      0
1152 1644      0 XDEF  APEREX
1153 1645 SKPT70 146 AB EX X
1154 1646     260 C=N
1155 1647      74 RCR   3
1156 1650     406 A=C   X
1157 1651      1 GOSUB GOSUB1
1157 1652      0
1158 1653      0 XDEF  STRCAB
1159 1654 SKPT80  1 GOLONG NFRPU
1159 1655      2

```

E
N
D

O
F

R
E
C

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

B.X= GIVEN CHAR POINTER

C.X= GIVEN RECORD POINTER

*
*

```

*****
* RCLPTA - RECALL POINTER OF THE FILE SPECIFIED BY THE ALPHA REGISTER*
* THE FILE NAME IS TAKEN FROM THE ALPHA REGISTER. IF ALPHA IS      *
* EMPTY, IT WILL DEFAULT TO THE CURRENT FILE.                      *
* RCLPT - "RECALL POINTER"; SIMILAR TO "RCLPTA" EXCEPT ALWAYS    *
* DEFAULT TO CURRENT FILE.                                         *
*****

```

*

```

1170      ENTRY  RCLPTA
1171      ENTRY  RCLPT
1172      ENTRY  RCLP30

```

*

```

1174 1656     224 CON   @224      T
1175 1657     20 CON   @20       P
1176 1660     14 CON   @14       L
1177 1661      3 CON   @03       C
1178 1662     22 CON   @22       R
1179 1663 RCLPT 1604 S0=  0
1180 1664     103 GOTO  RCLP10 (1674)

```

*

```

1182 1665     201 CON   @201      A
1183 1666     24 CON   @24       T
1184 1667     20 CON   @20       P
1185 1670     14 CON   @14       L
1186 1671      3 CON   @03       C
1187 1672     22 CON   @22       R
1188 1673 RCLPTA 1610 S0=  1
1189 1674 RCLP10  1 GOSUB GOSUB2
1189 1675      0
1190 1676      0 XDEF  FLSHAP
1191 1677 RCLP30 260 C=N
1192 1700      74 RCR   3
1193 1701     246 AC EX  X
1194 1702      1 GOSUB GOSUB0

```

C.X = CURRENT FILE POINTER

1194	1703	0			
1195	1704	0	XDEF	BIN-D	CONVERT TO DECIMAL
1196	1705	316	C=B		
1197	1706	360	CN EX		SAVE RECORD POINTER IN N
1198	1707	574	RCR	6	C.X= CHAR POINTER
1199	1710	406	A=C	X	
1200	1711	1	GOSUB	GOSUB0	
1200	1712	0			
1201	1713	0	XDEF	BIN-D	CONVERT TO DECIMAL
1202	1714	156	AB EX	W	
1203	1715	1240	SETDEC		
1204	1716	1532	? A#0	M	CHAR POINTER = 0 ?
1205	1717	43	GONC	RCLP40 (1723)	YES
1206	1720	460	LDI		
1207	1721	3	CON	3	
1208	1722	706	A=A-C	X	DIVIDE CHAR PTR BY 1000
1209	1723	260	C=N		
1210	1724	1	GOSUB	AD2-10	ADD REC AND CHAR POINTER TOGETHER
1210	1725	0			
1211	1726	356	BC EX	W	
1212	1727	1	GOLONG	RCL	
1212	1730	2			

*

* APERMG - ENTRY TO PRINT ERROR MESSAGE
 * APEREX - ERROR EXIT ENTRY POINT
 * DISERR - DISPLAY 'ERR' AND FALL INTO APEREX

*

		ENTRY	DISERR	
1219				
1220	1731	DISERR	1 GOSUB	MESSL
1220	1732		0	
1221	1733		40 CON	@40
1222	1734		5 CON	@5
1223	1735		22 CON	@22
1224	1736		1022 CON	@1022

E
R
R

*

		ENTRY	APEREX
1226			
1227	1737	APEREX	1 GOSUB
1227	1740		0
1228	1741		410 S8=
1229	1742		1 GOSUB
1229	1743		0
1230	1744		1 GOLONG
1230	1745		2

*

		ENTRY	APERMG
1232			
1233	1746	APERMG	1 GOSUB
1233	1747		0
1234	1750		1 GOSUB
1234	1751		0
1235	1752		1 GOLONG
1235	1753		2

*

*

*

*

1240		FILLTO @1763
1754	0000	NOP
1755	0000	NOP
1756	0000	NOP

1757	0000	NOP		
1760	0000	NOP		
1761	0000	NOP		
1762	0000	NOP		
1763	0000	NOP		
1241		LIST		
1242	1764	PPAUSE	0	NOP
1243	1765	PRUN	0	NOP
1244	1766	WAKEP	0	NOP
1245	1767	POWOFB	0	NOP
1246	1770	I/OSVP	0	NOP
1247	1771	DEEPSB	0	NOP
1248	1772	COLDSP	0	NOP
1249	1773	PRTID	3	CON @03
1250	1774		61	CON @61
1251	1775		20	CON @20
1252	1776		1	CON @01
1253	1777	CKSUMF	0	NOP
* 1255 END				
ERRORS : 0				

ENTRY FROM PAUSE LOOP
 RUNNING
 WAKE UP FROM DEEP SLEEP W/O KEY

WAKE-UP FROM DEEP SLEEP
 COLD START ENTRY POINT
 C
 I
 P
 A
 AP ROM CHECKSUM

SYMBOL TABLE

APEREX	1737	-							
APERMG	1746	-							
BYTLFT	1447	-							
CKSUMP	1777	-							
COLDSP	1772	-							
CURFL	232	-							
CURFLO	231	-	226						
CURFLD	233	-							
CURFLR	224	-							
CURFLT	227	-							
DEEPSP	1771	-							
DEL810	700	-	753						
DEL814	714	-	712						
DEL815	716	-	705						
DEL830	726	-	760						
DEL840	741	-	734						
DEL850	747	-	742						
DEL860	751	-	727						
DELCHR	546	-							
DELREC	556	-							
DISEFR	1731	-							
DLR010	557	-	547						
DLR020	574	-	570						
DLR030	624	-	620						
DLR050	656	-	622	575					
DLR060	676	-	655						
DSFLTP	103	-							
EFLS02	246	-							
EFLS05	247	-	235						
EFLS07	264	-							
EFLS08	266	-	263						
EFLS10	300	-	370	361	267	265			
EFLS20	321	-	317						
EFLS30	331	-	326						
EFLS70	372	-	355	335	311				
EFLS75	403	-	324						
EFLS76	410	-	404						
EFLS80	417	-	435						
EFLS85	436	-	365	357					
EFLS90	510	-	450	446	437	434	423		
EFLS92	527	-	515						
EFLSCH	245	-							
EMDIR	27	-							
EMDR10	32	-	124						
EMDR20	61	-	67						
EMDR30	76	-	102						
EMDR45	121	-	122						
EMDR50	124	-	142						
EMDR60	145	-	52						
EMDR70	153	-	151						
EMDR80	177	-	223						
EOFL	1546	-	1613						
FL#0	371	-	272						
FLNCPN	466	-	406	371					
FLTPER	451	-							
FSCH10	243	-	240						

FSCHP	236	-			
FSCHT	241	-			
GETR	1173	-			
GETR10	1115	-	1106		
GETR15	1134	-	1132		
GETRX	1114	-			
I/O SVP	1770	-			
PAKEXM	1037	-	1074		
PKRG20	1054	-	1076		
PKRG30	1065	-	1060		
PKRG40	1072	-	1066		
POWCFP	1767	-			
PFAUSE	1764	-			
PRT10	1773	-			
PRUN	1765	-			
PUFL10	1022	-	1016		
PUFLSB	773	-			
PURFL	766	-			
RCLP10	1674	-	1664		
RCLP30	1677	-			
RCLP40	1723	-	1717		
RCLPT	1667	-			
RCLPTA	1673	-			
REGM10	1325	-	1314		
REGM20	1334	-	1332		
REGM30	1356	-	1352		
REGMOV	1324	-			
REGSWP	1313	-			
RFLSCH	244	-			
RGMV	1367	-	1432	1303	1300
RGMV10	1411	-	1403		
RGMV12	1421	-	1436		
RGMV13	1425	-	1422		
RGMV15	1426	-	1446	1424	
RGMV20	1437	-	1420		
SAVER	1202	-			
SAVERX	1105	-			
SAVR05	1203	-	1174		
SAVR07	1233	-	1227		
SAVR10	1235	-	1225		
SAVR20	1251	-	1166		
SAVR50	1301	-	1267		
SAVRER	1230	-	1146		
SEKPT	1501	-			
SEKPTA	1512	-			
SKPT10	1513	-	1502		
SKPT20	1565	-	1545		
SKPT50	1567	-	1543		
SKPT60	1575	-	1571		
SKPT65	1615	-	1611		
SKPT70	1645	-	1624	1614	
SKPT80	1654	-	1566		
WAKEP	1766	-			
XTOR	4	-			

ENTRY TABLE

APEREX	1737	-
APERMG	1746	-
BYTLFT	1447	-
CURFL	232	-
CURFLD	233	-
CURFLR	224	-
CURFLT	227	-
DELCHR	546	-
DELREC	556	-
DISERR	1731	-
EFLS02	246	-
EFLSCH	245	-
EMDIR	27	-
EOFL	1546	-
FLNGFN	466	-
FLTPER	451	-
FSCHP	236	-
FSCHT	241	-
GETR	1173	-
GETRX	1114	-
PUFLSB	773	-
PURFL	766	-
RCLP30	1677	-
RCLPT	1663	-
RCLPTA	1673	-
REGMOV	1324	-
REGSWP	1313	-
RFLSCH	244	-
RGMV	1367	-
SAVER	1202	-
SAVERX	1105	-
SEKPT	1501	-
SEKPTA	1512	-
XTOR	4	-

NOMAS

NOT Manufacturer Supported
recipient agrees NOT to contact manufacturer

EXTERNAL REFERENCES

[illegible]

GTINDEX	1125	1330	
LEFTJ	216	1737	
LEFTJ	217	1740	
MESSL	203	1731	1752
MESSL	204	1732	1753
MSG105	113	221	1742
MSG105	114	222	1743
NFRPR	177		
NFRPR	200		
NFRPU	1654		
NFRPU	1655		
NORM	1021		
NXREG	1250		
NXTNDL	421	1071	
OFF	132		
OFF	133		
PTBYTA	642	722	
PTBYTA	643	723	
RCL	1727		
RCL	1730		
RSTKB	134		
RSTKB	135		
RSTMS0	116		
RSTMS0	117		
RSTSE0	143		
RSTSE0	144		
RASUB	161		
RASUB	162		
STRCAF	1653		
TOREC#	1601		
XX256	15		
XX999	602	1522	
XARCL	11		
YARCL	12		

End of VASM assembly