USER SURVIVAL GUIDE for TI-58/59 MASTER LIBRARY

INCLUDES: Program Listings, Register Assignments, Flowcharts, Interface Procedures and more!

[illegible]

FORWARD

This manual is designed to fill a very real gap in T.I.'s documentation of the Master Library manual, which is woefully inadequate for more than just a casual user. In addition, the appendices provide useful information for the 58/59. Specific procedures are given for efficiently interfacing most CROM programs with programs in user memory. Where practical, register assignments are given before and after executing particular functions.

While recognizing that programming is a multifaceted endeavor involving tradeoffs between execution time, program space, input-output ease and programming effort, any criticisms of T.I.'s programs expressed herein are made in the spirit of pointing out possible inefficiencies in the particular program structure chosen or comparing alternate techniques; not just as an exercise in picking nits. Text errors pointed out pertain to edition 1014984-21 (lower R.H. corner on back of M.L.M.). Other editions may have corrected these mistakes or added some new ones as a confusion factor.

Unlike computer programs in high level languages, calculator programs do not lend themselves well to flowcharting and no standard format exists which is both flexible and concise. As a result, the flowcharts herein contain varying mixtures of plain English, keyboard mnemonics, and fortran type assignment statements where the variable on the left-hand side of the equals symbol takes on the value specified by the right-hand side (is not an equation). In some cases where several operations are being performed concurrently, the order of completion may not be strictly adhered to in the flowchart. The emphasis is on understanding the program structure without getting bogged down to your eyeballs in the arithmetic details. Blocks which are dashed contain phantom variables or operations which exist only in the flowchart to enhance understanding.

Although every effort has been made to ensure technical accuracy, the author does not assume any responsibility for consequences resulting from use of any material herein. This manual is for informational purposes only and has been produced without any collaboration with Texas Instruments Inc.

For those 58/59 users who are interested in obtaining maximum performance from their machine, the author would like to recommend that they subscribe to 52 Notes, the newsletter for a club of T.I. programmable calculator users, which is independent of T.I. and an excellent source of information. Newsletters are published monthly at a nominal cost of $1. A six month membership is $6, which includes the newsletters. Back issues start June 1976 (get them all...much of the information for the SR-52 is applicable to the 58/59). The address is:

52-Notes
9459 Taylorsville Road
Dayton, OH 45424

Note the numbers which appear on the upper right-hand corner of each page of this manual pertaining to a CROM program. This makes locating specific material easier and faster.

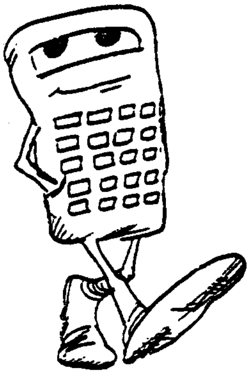
I welcome any comments or questions concerning this manual. In the meantime....HAPPY COMPUTING!

Fred Fish

Fred Fish

THIS MANUAL WAS REPRODUCED WITH THE PERMISSION
OF FRED FISH. HIS PRESENT ADDRESS IS:
2325 N. 87TH WAY, TEMPE, AZ 85251
HOME PHONE:602-894-6881
WORK PHONE:602-932-7391

FOR TECHNICAL INFORMATION, DIRECT YOUR INQUIRIES TO FRED FISH.
FOR COPIES OF THIS MANUAL, WRITE TO:
TI PPC NOTES, P.O.BOX 710, LANHAM MD 20801
OR CALL MAURICE SWINNEN, 301-459-5458.



ML-01

MASTER LIBRARY DIAGNOSTIC

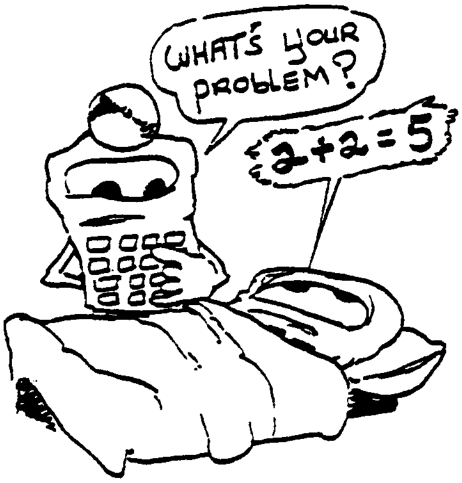
Not much can be said about ML-01 that wouldn't be just a duplication of the Master Library Manual. The coding is straightforward and information about each piece is contained with the program listing.

Let us simply note a few facts in passing:

(1) Running the "diagnostic" portion affects registers 1-7, 9, and the T register. Note the omission of the use of R07 in the M.L.M.

(2) PGM 01 SBR 012 with a value "MN" in the display can be used to clear registers 1-MN. For example, with 15 in the display, PGM 01 SBR 012 clears registers 1-15.

(3) The last step in the user instructions should indicate that the program in use <u>must not</u> be called.



ML-01 Program Listing

000	76	LBL	
001	24	CE	
002	00	0	Clear R09, put into degree mode, fix display format at 9 places (floating decimal).
003	42	STO	
004	09	09	
005	60	DEG	
006	58	FIX	
007	09	09	
008	76	LBL	
009	25	CLR	
010	29	CP	
011	06	6	
012	42	STO	Store display value in R01 as counter/pointer and clear registers one through display value.
013	01	01	
014	00	0	
015	72	ST*	
016	01	01	
017	97	DSZ	
018	01	01	
019	00	00	
020	15	15	
021	92	RTN	
022	76	LBL	
023	95	=	
024	71	SBR	
025	24	CE	
026	05	5	
027	32	X:T	
028	03	3	
029	00	0	
030	37	P/R	
031	78	Σ+	
032	22	INV	
033	37	P/R	
034	78	Σ+	
035	69	OP	Generate a test number.
036	12	12	
037	88	DMS	
038	78	Σ+	
039	69	OP	
040	11	11	
041	22	INV	
042	88	DMS	
043	22	INV	
044	78	Σ+	
045	69	OP	
046	14	14	
047	53	(	
048	24	CE	
049	75	-	
050	36	PGM	Finish generating the test number and truncate internal digits.
051	15	15	
052	71	SBR	
053	88	DMS	
054	54	)	
055	52	EE	
056	22	INV	
057	52	EE	
058	32	X:T	If test number is correct go to label WRT. If not, create an error state and fall through to label WRT.
059	03	3	
060	07	7	
061	07	7	
062	93	.	
063	02	2	
064	05	5	
065	08	8	
066	00	0	
067	09	9	
068	05	5	
069	04	4	
070	67	EQ	
071	96	WRT	
072	00	0	Error state producer.
073	35	1/X	
074	76	LBL	
075	96	WRT	
076	69	OP	
077	00	00	
078	01	1	
079	03	3	Print "MASTER" and a numeral 1.
080	03	3	
081	06	6	
082	03	3	
083	07	7	
084	01	1	
085	07	7	
086	03	3	
087	05	5	
088	69	OP	
089	04	04	
090	03	3	
091	00	0	
092	69	OP	
093	03	03	
094	69	OP	
095	05	05	
096	01	1	
097	99	PRT	
098	92	RTN	

ML-01 Program Listing (cont.)

```
099  76 LBL
100  11  A
101  98 ADV
102  99 PRT
103  62 PG*
104  00  00
105  11  A
106  99 PRT
107  92 RTN
108  76 LBL
109  12  B
110  98 ADV
111  99 PRT
112  62 PG*
113  00  00
114  12  B
115  99 PRT
116  92 RTN
117  76 LBL
118  13  C
119  98 ADV
120  99 PRT
121  62 PG*
122  00  00
123  13  C
124  99 PRT
125  92 RTN
126  76 LBL
127  14  D
128  98 ADV
129  99 PRT
130  62 PG*
131  00  00
132  14  D
133  99 PRT
134  92 RTN
135  76 LBL
136  15  E
137  98 ADV
138  99 PRT
139  62 PG*
140  00  00
141  15  E
142  99 PRT
143  92 RTN
```

```
144  76 LBL
145  16 A'
146  98 ADV
147  99 PRT
148  62 PG*
149  00  00
150  16 A'
151  99 PRT
152  92 RTN
153  76 LBL
154  17 B'
155  98 ADV
156  99 PRT
157  62 PG*
158  00  00
159  17 B'
160  99 PRT
161  92 RTN
162  76 LBL
163  18 C'
164  98 ADV
165  99 PRT
166  62 PG*
167  00  00
168  18 C'
169  99 PRT
170  92 RTN
171  76 LBL
172  19 D'
173  98 ADV
174  99 PRT
175  62 PG*
176  00  00
177  19 D'
178  99 PRT
179  92 RTN
180  76 LBL
181  10 E'
182  98 ADV
183  99 PRT
184  62 PG*
185  00  00
186  10 E'
187  99 PRT
188  92 RTN
```

```
001  24 CE
009  25 CLR
023  95  =
075  96 WRT
100  11  A
109  12  B
118  13  C
127  14  D
136  15  E
145  16 A'
154  17 B'
163  18 C'
172  19 D'
181  10 E'
```

Labels A-E' advance printer, print the display value (input), call the program whose number is in R00 and execute the label that matches the ML-01 label. Output results are then printed.

"It only prints lemons...... think it's trying to tell me something?"

ML-02

MATRIX INVERSION, DETERMINANTS AND SIMULTANEOUS EQUATIONS

ML-02 is not only the longest program in the Master Library, but also considerably more complex than any other due to the requirement that it handle various order systems. At this time the author's stack of note's for ML-02 is almost as thick as for all the other library programs combined. To adequately explain the detailed workings of the program would require delving deeply into numerical analysis of linear systems. On the premise that only a relatively small percentage of readers will be interested in the details, specific analysis is not included so as to prevent a large increase in the length of this manual (and hence price). Those readers who <u>are</u> interested may send the author a postcard with their name and address, and if sufficient demand exists, a complete dissection of ML-02 will be printed up and made available at cost plus postage at a later date. In the meantime, refer to pages 146-160 of <u>Numerical Methods</u>, Dahlquist & Bjork, Prentice Hall, 1974.

Interface procedure:

(1) Prestore system order (n) in R07.

(2) With desired starting column for entering Matrix A in display, execute PGM 02 B. Input each element by "STO*01 OP 21".

(3) To find the determinant execute PGM 02 C ...returns with value of determinant in the display and prints it. This step must be performed before finding the inverse or solving simultaneous equations since it does the LU decomposition.

(4) To solve Ax = b, prestore the desired starting row (i) for column vector b in R05. Input each element with PGM 02 SBR 355. After last input execute PGM 02 E. To output column vector x, with desired starting row in display, execute PGM 02 A', the "RCL*01 OP21 for each element to be output.

(5) To find the inverse matrix execute PGM 02 B'. Then with the desired output column in display, execute PGM 02 C' followed by PGM 02 SBR 860 for first element. Each element thereafter is output with PGM 02 SBR 869.

(6) To get the determinant and the inverse in one step, execute PGM 02 E' and see the determinant displayed and printed. Output each element of the inverse as in (5).

Special notes:

(1) During normal operation, with no pending operations, it is not necessary to hit CLR before executing labels E or B' contrary to M.L.M.

(2) Steps 339-349 (11 NOP's) apparently are a residual of converting from labels to absolute addressing. Granted, changing a lot of absolute addresses as the program gets shorter from label elimination is a pain in the posterior, but in this application there is no excuse for not doing so.

(3) The = at step 543 is redundant.

(4) Both RTN's at steps 051 and 069 are unnecessary...steps 034 and 056 could be changed to 14.

Special applications:

(1) PGM 02 D' evaluates I + (I-1)(R07) + 7 where I = display input.

(2) PGM 02 SBR 020 decrements R01, and makes R02 = R02-R07.

(3) PGM 02 SBR 327 evaluates and prints R06 = R06(RMN) where MN = display value. (uses R01)

ML-02 Program Listing p31

, 55

```
000  76  LBL
001  19  D'
002  85  +
003  53  (
004  24  CE
005  75  -
006  01  1
007  54  )
008  65  ×
009  43  RCL
010  07  07
011  85  +
012  07  7
013  95  =
014  92  RTN
015  76  LBL
016  18  C'
017  61  GTO
018  08  08
019  11  11
020  01  1
021  22  INV
022  44  SUM
023  01  01
024  43  RCL
025  07  07
026  22  INV
027  44  SUM
028  02  02
029  92  RTN
030  22  INV
031  97  DSZ
032  05  05
033  00  00
034  51  51
035  75  -
036  01  1
037  44  SUM
038  01  01
039  43  RCL
040  07  07
041  44  SUM
042  02  02
043  73  RC*
044  01  01
045  65  ×
046  73  RC*
047  02  02
048  61  GTO
049  00  00
050  30  30
051  92  RTN
052  22  INV
053  97  DSZ
054  05  05
055  00  00
056  69  69
057  75  -
058  71  SBR
059  00  00
060  20  20
061  73  RC*
062  01  01
063  65  ×
064  73  RC*
065  02  02
066  61  GTO
067  00  00
068  52  52
069  92  RTN
070  76  LBL
071  11  A
072  42  STO
073  07  07
074  99  PRT
075  98  ADV
076  92  RTN
077  76  LBL
078  12  B
079  75  -
080  32  X:T
081  01  1
082  95  =
083  65  ×
084  43  RCL
085  07  07
086  85  +
087  08  8
088  95  =
089  42  STO
090  01  01
091  32  X:T
092  92  RTN
093  99  PRT
094  72  ST*
095  01  01
096  92  RTN
097  32  X:T
098  01  1
099  44  SUM
100  01  01
101  32  X:T
102  61  GTO
103  00  00
104  93  93
105  76  LBL
106  13  C
107  43  RCL
108  07  07
109  42  STO
110  05  05
111  85  +
112  33  X²
113  85  +
114  07  7
115  95  =
116  42  STO
117  01  01
118  43  RCL
119  05  05
120  72  ST*
121  01  01
122  01  1
123  22  INV
124  44  SUM
125  01  01
126  97  DSZ
127  05  05
128  01  01
129  18  18
130  01  1
131  42  STO
132  04  04
133  42  STO
134  06  06
135  43  RCL
136  04  04
137  85  +
138  07  7
139  95  =
140  42  STO
141  03  03
142  75  -
143  43  RCL
144  07  07
145  75  -
146  08  8
147  95  =
148  42  STO
149  05  05
150  43  RCL
151  04  04
152  19  D'
153  42  STO
154  02  02
155  73  RC*
156  02  02
157  50  I×I
158  32  X:T
159  22  INV
160  97  DSZ
161  05  05
162  01  01
163  88  88
164  01  1
165  44  SUM
166  02  02
167  73  RC*
168  02  02
169  50  I×I
170  22  INV
171  77  GE
172  01  01
173  59  59
174  32  X:T
175  43  RCL
176  07  07
177  85  +
178  43  RCL
179  05  05
180  85  +
181  08  8
182  95  =
183  42  STO
184  03  03
185  61  GTO
186  01  01
187  59  59
188  43  RCL
189  03  03
190  32  X:T
191  43  RCL
192  04  04
193  85  +
194  07  7
195  95  =
196  67  EQ
197  02  02
198  28  28
199  42  STO
```

```
200  02  02
201  01  1
202  94  +/-
203  49  PRD
204  06  06
205  43  RCL
206  07  07
207  85  +
208  01  1
209  95  =
210  42  STO
211  05  05
212  73  RC*
213  02  02
214  63  EX*
215  03  03
216  72  ST*
217  02  02
218  43  RCL
219  07  07
220  44  SUM
221  02  02
222  44  SUM
223  03  03
224  97  DSZ
225  05  05
226  02  02
227  12  12
228  43  RCL
229  04  04
230  19  D'
231  42  STO
232  01  01
233  73  RC*
234  01  01
235  49  PRD
236  06  06
237  29  CP
238  67  EQ
239  03  03
240  31  31
241  43  RCL
242  07  07
243  42  STO
244  05  05
245  43  RCL
246  04  04
247  19  D'
248  42  STO
249  02  02
250  85  +
251  43  RCL
252  05  05
253  75  -
254  43  RCL
255  04  04
256  95  =
257  42  STO
258  03  03
259  73  RC*
260  03  03
261  55  ÷
262  73  RC*
263  02  02
264  95  =
265  42  STO
266  01  01
267  72  ST*
268  03  03
269  43  RCL
270  07  07
271  44  SUM
272  02  02
273  44  SUM
274  03  03
275  33  X²
276  85  +
277  08  8
278  95  =
279  32  X:T
280  43  RCL
281  03  03
282  77  GE
283  02  02
284  97  97
285  43  RCL
286  01  01
287  94  +/-
288  65  ×
289  73  RC*
290  02  02
291  95  =
292  74  SM*
293  03  03
294  61  GTO
295  02  02
296  69  69
297  01  1
298  22  INV
299  44  SUM
300  05  05
301  43  RCL
302  05  05
303  32  X:T
304  43  RCL
305  04  04
306  67  EQ
307  03  03
308  12  12
309  61  GTO
310  02  02
311  45  45
312  01  1
313  44  SUM
314  04  04
315  43  RCL
316  07  07
317  32  X:T
318  43  RCL
319  04  04
320  67  EQ
321  03  03
322  26  26
323  61  GTO
324  01  01
325  35  35
326  19  D'
327  42  STO
328  01  01
329  73  RC*
330  01  01
331  49  PRD
332  06  06
333  43  RCL
334  06  06
335  98  ADV
336  99  PRT
337  98  ADV
338  92  RTN
339  68  NOP
340  68  NOP
341  68  NOP
342  68  NOP
343  68  NOP
344  68  NOP
345  68  NOP
346  68  NOP
347  68  NOP
348  68  NOP
349  68  NOP
350  76  LBL
351  14  D
352  42  STO
353  05  05
354  92  RTN
355  42  STO
356  03  03
357  43  RCL
358  07  07
359  33  X²
360  85  +
361  07  7
362  95  =
363  42  STO
364  01  01
365  43  RCL
366  05  05
367  32  X:T
368  43  RCL
369  07  07
370  22  INV
371  77  GE
372  03  03
373  54  54
374  01  1
375  44  SUM
376  01  01
377  73  RC*
378  01  01
379  22  INV
380  67  EQ
381  03  03
382  74  74
383  43  RCL
384  07  07
385  44  SUM
386  01  01
387  01  1
388  44  SUM
389  05  05
390  43  RCL
391  03  03
392  72  ST*
393  01  01
394  99  PRT
395  92  RTN
396  61  GTO
397  03  03
398  55  55
399  76  LBL
```

```
400  15    E   | 450  42  STO | 500  77   GE  | 550  01   01
401  01    1   | 451  04   04 | 501  04   04  | 551  01    1
402  42  STO   | 452  07    7 | 502  52   52  | 552  44  SUM
403  04   04   | 453  85    + | 503  98  ADV  | 553  04   04
404  43  RCL   | 454  53    ( | 504  01    1  | 554  43  RCL
405  07   07   | 455  43  RCL | 505  92  RTN  | 555  04   04
406  85    +   | 456  07   07 | 506  76  LBL  | 556  32  X:T
407  33   X²   | 457  85    + | 507  16   A'  | 557  43  RCL
408  85    +   | 458  01    1 | 508  85    +  | 558  07   07
409  07    7   | 459  54    ) | 509  32  X:T  | 559  77   GE
410  95    =   | 460  33   X² | 510  07    7  | 560  05   05
411  42  STO   | 461  95    = | 511  85    +  | 561  40   40
412  01   01   | 462  42  STO | 512  43  RCL  | 562  01    1
413  43  RCL   | 463  01   01 | 513  07   07  | 563  94  +/-
414  04   04   | 464  75    - | 514  65    ×  | 564  44  SUM
415  42  STO   | 465  43  RCL | 515  53    (  | 565  04   04
416  05   05   | 466  07   07 | 516  24   CE  | 566  01    1
417  75    -   | 467  75    - | 517  85    +  | 567  42  STO
418  43  RCL   | 468  43  RCL | 518  01    1  | 568  03   03
419  07   07   | 469  04   04 | 519  54    )  | 569  43  RCL
420  85    +   | 470  42  STO | 520  95    =  | 570  04   04
421  07    7   | 471  05   05 | 521  42  STO  | 571  19   D'
422  95    =   | 472  95    = | 522  01   01  | 572  42  STO
423  42  STO   | 473  42  STO | 523  32  X:T  | 573  01   01
424  02   02   | 474  02   02 | 524  92  RTN  | 574  75    -
425  00    0   | 475  00    0 | 525  73  RC*  | 575  43  RCL
426  71  SBR   | 476  71  SBR | 526  01   01  | 576  03   03
427  00   00   | 477  00   00 | 527  99  PRT  | 577  42  STO
428  30   30   | 478  52   52 | 528  92  RTN  | 578  05   05
429  85    +   | 479  85    + | 529  01    1  | 579  95    =
430  01    1   | 480  71  SBR | 530  44  SUM  | 580  42  STO
431  44  SUM   | 481  00   00 | 531  01   01  | 581  02   02
432  01   01   | 482  20   20 | 532  61  GTO  | 582  73  RC*
433  73  RC*   | 483  73  RC* | 533  05   05  | 583  01   01
434  01   01   | 484  01   01 | 534  25   25  | 584  65    ×
435  95    =   | 485  95    = | 535  76  LBL  | 585  73  RC*
436  72  ST*   | 486  55    ÷ | 536  17   B'  | 586  02   02
437  01   01   | 487  73  RC* | 537  01    1  | 587  94  +/-
438  01    1   | 488  02   02 | 538  42  STO  | 588  71  SBR
439  44  SUM   | 489  95    = | 539  04   04  | 589  00   00
440  04   04   | 490  72  ST* | 540  43  RCL  | 590  52   52
441  43  RCL   | 491  01   01 | 541  04   04  | 591  95    =
442  04   04   | 492  01    1 | 542  19   D'  | 592  65    ×
443  32  X:T   | 493  44  SUM | 543  95    =  | 593  71  SBR
444  43  RCL   | 494  04   04 | 544  42  STO  | 594  00   00
445  07   07   | 495  43  RCL | 545  01   01  | 595  20   20
446  77   GE   | 496  04   04 | 546  73  RC*  | 596  73  RC*
447  04   04   | 497  32  X:T | 547  01   01  | 597  02   02
448  04   04   | 498  43  RCL | 548  35  1/X  | 598  95    =
449  01    1   | 499  07   07 | 549  72  ST*  | 599  72  ST*
```

```
600  01    01     650  95  =        700  02    02     750  85  +
601  01  1        651  72  ST*      701  43  RCL      751  01  1
602  44  SUM      652  01    01     702  07    07     752  95  =
603  03    03     653  01  1        703  75  -        753  42  STO
604  43  RCL      654  44  SUM      704  43  RCL      754  03    03
605  04    04     655  03    03     705  05    05     755  43  RCL
606  32  X:T      656  43  RCL      706  85  +        756  05    05
607  43  RCL      657  03    03     707  01  1        757  19  D'
608  03    03     658  32  X:T      708  95  =        758  42  STO
609  22  INV      659  43  RCL      709  48  EXC      759  02    02
610  67    EQ     660  07    07     710  05    05     760  43  RCL
611  05    05     661  75  -        711  32  X:T      761  03    03
612  69    69     662  43  RCL      712  73  RC*      762  42  STO
613  01  1        663  04    04     713  03    03     763  01    01
614  42  STO      664  95  =        714  94  +/-      764  43  RCL
615  03    03     665  77    GE     715  71  SBR      765  07    07
616  22  INV      666  06    06     716  00    00     766  75  -
617  44  SUM      667  27    27     717  30    30     767  43  RCL
618  04    04     668  01  1        718  95  =        768  05    05
619  43  RCL      669  44  SUM      719  94  +/-      769  85  +
620  04    04     670  04    04     720  72  ST*      770  01  1
621  32  X:T      671  42  STO      721  03    03     771  95  =
622  01  1        672  03    03     722  43  RCL      772  48  EXC
623  22  INV      673  43  RCL      723  07    07     773  05    05
624  67    EQ     674  07    07     724  44  SUM      774  32  X:T
625  05    05     675  32  X:T      725  03    03     775  73  RC*
626  69    69     676  43  RCL      726  32  X:T      776  01    01
627  43  RCL      677  04    04     727  85  +        777  65  X
628  04    04     678  22  INV      728  01  1        778  73  RC*
629  19  D'       679  67    EQ     729  95  =        779  02    02
630  42  STO      680  06    06     730  42  STO      780  94  +/-
631  01    01     681  27    27     731  05    05     781  71  SBR
632  85  +        682  01  1        732  32  X:T      782  00    00
633  43  RCL      683  42  STO      733  77    GE     783  30    30
634  03    03     684  04    04     734  06    06     784  95  =
635  42  STO      685  43  RCL      735  92    92     785  94  +/-
636  05    05     686  04    04     736  32  X:T      786  72  ST*
637  95  =        687  42  STO      737  43  RCL      787  03    03
638  42  STO      688  05    05     738  04    04     788  01  1
639  02    02     689  19  D'       739  67    EQ     789  44  SUM
640  73  RC*      690  42  STO      740  08    08     790  03    03
641  02    02     691  03    03     741  08    08     791  85  +
642  94  +/-      692  43  RCL      742  85  +        792  32  X:T
643  71  SBR      693  05    05     743  01  1        793  95  =
644  00    00     694  19  D'       744  95  =        794  42  STO
645  30    30     695  42  STO      745  42  STO      795  05    05
646  65  X        696  01    01     746  05    05     796  32  X:T
647  01  1        697  43  RCL      747  43  RCL      797  43  RCL
648  44  SUM      698  03    03     748  04    04     798  07    07
649  01    01     699  42  STO      749  19  D'       799  77    GE
```

```
800   07   07
801   55   55
802   01    1
803   44  SUM
804   04   04
805   61  GTO
806   06   06
807   85   85
808   98  ADV
809   01    1
810   92  RTN
811   42  STO
812   03   03
813   32  X:T
814   43  RCL
815   07   07
816   22  INV
817   77   GE
818   08   08
819   10   10
820   85    +
821   42  STO
822   05   05
823   33  X²
824   85    +
825   07    7
826   95    =
827   42  STO
828   01   01
829   00    0
830   42  STO
831   04   04
832   73  RC*
833   01   01
834   67   EQ
835   08   08
836   45   45
837   01    1
838   22  INV
839   44  SUM
840   01   01
841   97  DSZ
842   05   05
843   08   08
844   32   32
845   43  RCL
846   05   05
847   75    -
848   01    1
849   95    =
```

```
850   65    ×
851   43  RCL
852   07   07
853   85    +
854   07    7
855   95    =
856   42  STO
857   01   01
858   32  X:T
859   92  RTN
860   01    1
861   44  SUM
862   01   01
863   44  SUM
864   04   04
865   73  RC*
866   01   01
867   99  PRT
868   92  RTN
869   43  RCL
870   04   04
871   32  X:T
872   43  RCL
873   07   07
874   22  INV
875   67   EQ
876   08   08
877   60   60
878   01    1
879   85    +
880   43  RCL
881   03   03
882   95    =
883   18  C'
884   61  GTO
885   08   08
886   60   60
887   76  LBL
888   10  E'
889   13    C
890   29  CP
891   67   EQ
892   08   08
893   95   95
894   17  B'
895   43  RCL
896   06   06
897   92  RTN
```

```
001   19  D'
016   18  C'
071   11   A
078   12   B
106   13   C
351   14   D
400   15   E
507   16  A'
536   17  B'
888   10  E'
```

ML-03

MATRIX ADDITION AND MULTIPLICATION

ML-03 performs matrix addition and multiplication utilizing the formulas given in the Master Library Manual.

MATRIX ADDITION:

Register assignments are:

R01: Pointer p	R03: m	R05: $\lambda 1$	R07: Pointer or counter
R02: Pointer q	R04: n	R06: $\lambda 2$	

a_{11}	a_{21}	a_{31}	.....	a_{m1}	a_{12}	a_{22}	a_{32}	.....	a_{mn}

R08 R(mn+7)

b_{11}	b_{21}	b_{31}	.....	b_{m1}	b_{12}	b_{22}	b_{32}	.....	b_{mn}

R(mn+8) R(2mn+7)

For final assignments, c_{ij} replaces a_{ij} and b_{ij} is not affected.

Interface procedure:

(1) Prestore $\lambda 1$, $\lambda 2$, m, and n in the assigned registers.

(2) With number of desired starting column of matrix A in the display, execute PGM 03 B.

(3) With element of matrix A in display, execute STO*07 OP 27 for each element to be input. Insert a print command if desired.

(4) Repeat steps (2) and (3) for matrix B using PGM 03 C in (2).

(5) Execute PGM 03 E. To output matrix C, enter the desired starting column in display and execute PGM 03 A'. To output each element execute RCL*02 OP 27. Insert a print command if desired.

Interface data:

Flags used: none
Parentheses levels: 1 (note that equals is used)
Subroutine levels: none

Special notes:

Contrary to M.L.M. user instructions, it is not necessary to hit CLR before executing label E under normal use.

MATRIX MULTIPLICATION:

Register assignments are:

R01: Pointer A R03: m R05: not used R07: Pointer or
R02: Pointer B R04: n R06: not used counter

a_{11}	a_{21}	a_{31}	.....	a_{m1}	a_{12}	a_{22}	a_{32}	.....	a_{mn}

R08 R(mn+7)

c_{1j}	c_{2j}	c_{3j}	.....	c_{mj}	0	b_{1j}	b_{2j}	b_{3j}	.....	b_{nj}

R(mn+8) R(mn+m+8) R(mn+m+n+8)

Interface procedure:

(1) Execute steps (1)-(3) of the matrix addition interface procedure.

(2) With number of desired starting row in column vector x of matrix B in display execute PGM 03 B'.

(3) Execute STO*07 OP 27 for each element of column vector to be input, with the element in the display. Insert a print command if desired.

(4) Execute PGM 03 C'. With the number of the desired starting row of column vector y of matrix C in the display, execute PGM 03 D'. For each element to be output execute RCL*07 OP 27. Insert a print command if desired.

Interface data:

Flags used: none
Parentheses levels: 1 (note that the equals function is used)
Subroutine levels: 1

Special notes:

(1) Label DEG is an exact duplicate of label |x| and could be eliminated by replacing steps 154-166 with GTO |x| thus saving a net of 11 steps.

(2) Steps 048-062 could be replaced with GTO 015, saving a net 12 steps.

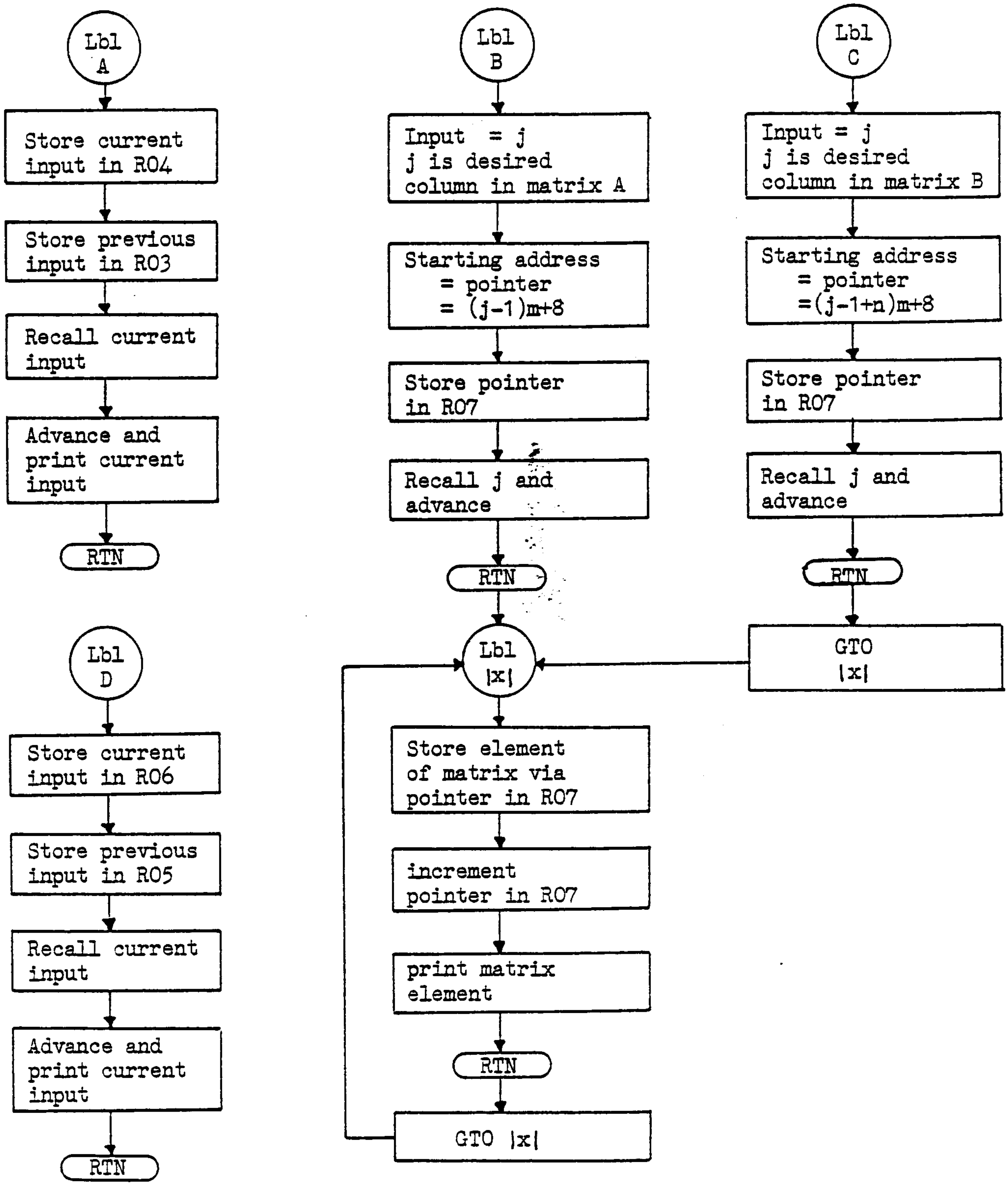
(3) Steps 263-273 are identical in function with steps 122-132. A GTO y^x at step 263 would save a net 9 steps.

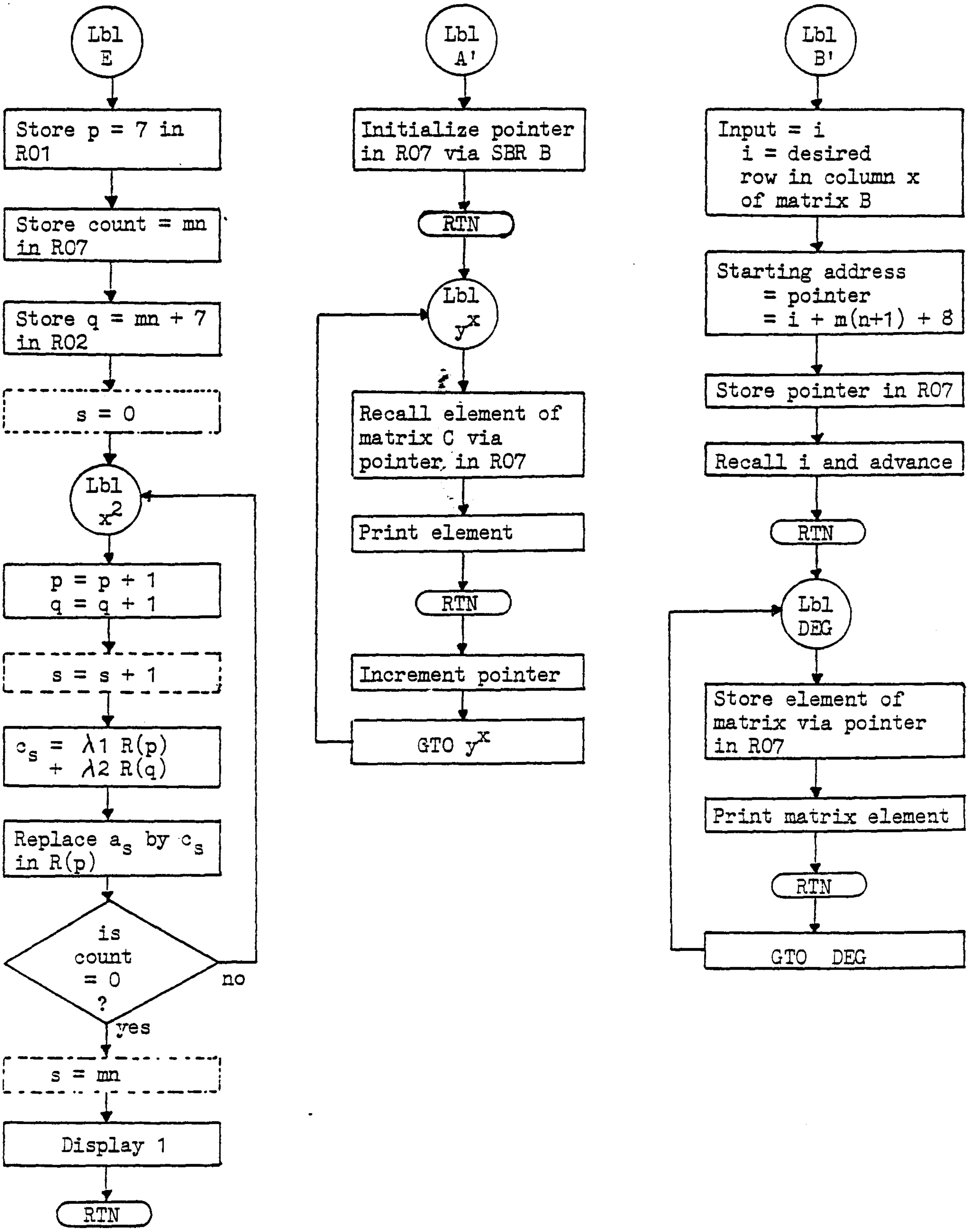
(4) By inserting a LBL CLR between steps 115 and 116, steps 242-245 could be eliminated, saving a net 2 steps.

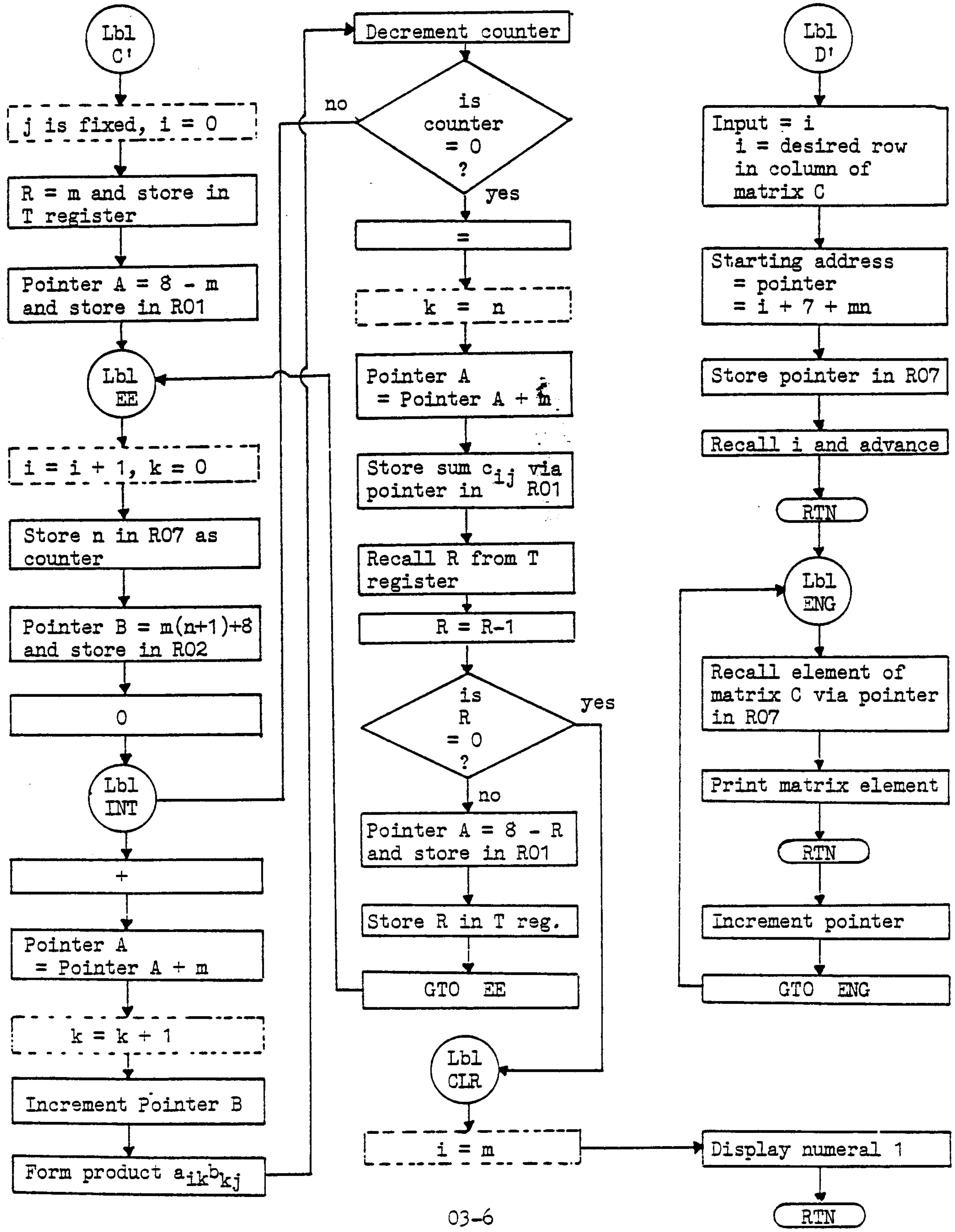
(5) Due to a programming error there is always a wasted register at R(mn+m+8) during matrix multiplication. This could be eliminated by changing the 8's at steps 147 and 192 to 7's.

(6) Contrary to M.L.M., for matrix multiplication, the highest register used is R(mn+m+n+8) not R(mn+2n+7).

(7) Though it might appear at first glance that the Lbl A' B RTN sequence could be eliminated by Lbl A' Lbl B'...., this is not so since label A' execution is normally followed by R/S.


Lbl A
Store current input in R04
Store previous input in R03
Recall current input
Advance and print current input
RTN
Lbl D
Store current input in R06
Store previous input in R05
Recall current input
Advance and print current input
RTN
Lbl B
Input = j
j is desired column in matrix A
Starting address = pointer = (j-1)m+8
Store pointer in R07
Recall j and advance
RTN
Lbl |x|
Store element of matrix via pointer in R07
increment pointer in R07
print matrix element
RTN
GTO |x|
Lbl C
Input = j
j is desired column in matrix B
Starting address = pointer =(j-1+n)m+8
Store pointer in R07
Recall j and advance
RTN
GTO |x|


Lbl E
Store p = 7 in R01
Store count = mn in R07
Store q = mn + 7 in R02
s = 0
Lbl x^2
p = p + 1
q = q + 1
s = s + 1
c_s = λ1 R(p) + λ2 R(q)
Replace a_s by c_s in R(p)
is count = 0 ?
no
yes
s = mn
Display 1
RTN
Lbl A'
Initialize pointer in R07 via SBR B
RTN
Lbl y^x
Recall element of matrix C via pointer in R07
Print element
RTN
Increment pointer
GTO y^x
Lbl B'
Input = i
i = desired row in column x of matrix B
Starting address = pointer = i + m(n+1) + 8
Store pointer in R07
Recall i and advance
RTN
Lbl DEG
Store element of matrix via pointer in R07
Print matrix element
RTN
GTO DEG


Lbl C'
j is fixed, i = 0
R = m and store in T register
Pointer A = 8 - m and store in R01
Lbl EE
i = i + 1, k = 0
Store n in R07 as counter
Pointer B = m(n+1)+8 and store in R02
0
Lbl INT
+
Pointer A = Pointer A + m
k = k + 1
Increment Pointer B
Form product $a_{ik}b_{kj}$
Decrement counter
no
is counter = 0 ?
yes
=
k = n
Pointer A = Pointer A + m
Store sum c_{ij} via pointer in R01
Recall R from T register
R = R-1
is R = 0 ?
yes
no
Pointer A = 8 - R and store in R01
Store R in T reg.
GTO EE
Lbl CLR
i = m
Lbl D'
Input = i
i = desired row in column of matrix C
Starting address = pointer = i + 7 + mn
Store pointer in R07
Recall i and advance
RTN
Lbl ENG
Recall element of matrix C via pointer in R07
Print matrix element
RTN
Increment pointer
GTO ENG
Display numeral 1
RTN

ML-03 Program Listing

```
000   76  LBL
001   11   A
002   48  EXC
003   04   04
004   48  EXC
005   03   03
006   43  RCL
007   04   04
008   98  ADV
009   99  PRT
010   92  RTN
011   76  LBL
012   12   B
013   75   -
014   32  X:T
015   01   1
016   95   =
017   65   ×
018   43  RCL
019   03   03
020   85   +
021   08   8
022   95   =
023   42  STO
024   07   07
025   32  X:T
026   98  ADV
027   92  RTN
028   76  LBL
029   50  I×I
030   72  ST÷
031   07   07
032   32  X:T
033   01   1
034   44  SUM
035   07   07
036   32  X:T
037   99  PRT
038   92  RTN
039   61  GTO
040   50  I×I
041   76  LBL
042   13   C
043   85   +
044   32  X:T
045   43  RCL
046   04   04
047   75   -
048   01   1
049   95   =
050   65   ×
051   43  RCL
052   03   03
053   85   +
054   08   8
055   95   =
056   42  STO
057   07   07
058   32  X:T
059   98  ADV
060   92  RTN
061   61  GTO
062   50  I×I
063   76  LBL
064   14   D
065   48  EXC
066   06   06
067   48  EXC
068   05   05
069   43  RCL
070   06   06
071   98  ADV
072   99  PRT
073   92  RTN
074   76  LBL
075   15   E
076   07   7
077   42  STO
078   01   01
079   85   +
080   53   (
081   43  RCL
082   03   03
083   65   ×
084   43  RCL
085   04   04
086   54   )
087   42  STO
088   07   07
089   95   =
090   42  STO
091   02   02
092   76  LBL
093   33  X²
094   01   1
095   44  SUM
096   01   01
097   44  SUM
098   02   02
099   43  RCL
100   05   05
101   65   ×
102   73  RC÷
103   01   01
104   85   +
105   43  RCL
106   06   06
107   65   ×
108   73  RC÷
109   02   02
110   95   =
111   72  ST÷
112   01   01
113   97  DSZ
114   07   07
115   33  X²
116   01   1
117   92  RTN
118   76  LBL
119   16  A'
120   12   B
121   92  RTN
122   76  LBL
123   45  YX
124   73  RC÷
125   07   07
126   99  PRT
127   92  RTN
128   01   1
129   44  SUM
130   07   07
131   61  GTO
132   45  YX
133   76  LBL
134   17  B'
135   85   +
136   32  X:T
137   43  RCL
138   03   03
139   65   ×
140   53   (
141   43  RCL
142   04   04
143   85   +
144   01   1
145   54   )
146   85   +
147   08   8
148   95   =
149   42  STO
150   07   07
151   32  X:T
152   98  ADV
153   92  RTN
154   76  LBL
155   60  DEG
156   72  ST÷
157   07   07
158   32  X:T
159   01   1
160   44  SUM
161   07   07
162   32  X:T
163   99  PRT
164   92  RTN
165   61  GTO
166   60  DEG
167   76  LBL
168   18  C'
169   43  RCL
170   03   03
171   75   -
172   32  X:T
173   08   8
174   95   =
175   94  +/-
176   42  STO
177   01   01
178   76  LBL
179   52  EE
180   43  RCL
181   03   03
182   65   ×
183   53   (
184   43  RCL
185   04   04
186   42  STO
187   07   07
188   85   +
189   01   1
190   54   )
191   85   -
192   08   8
193   95   =
194   42  STO
195   02   02
196   00   0
197   76  LBL
198   59  INT
199   85   +
```

ML-03 Program Listing (cont.)

```
200   43  RCL
201   03   03
202   44  SUM
203   01   01
204   01   1
205   44  SUM
206   02   02
207   73  RC*
208   01   01
209   65   x
210   73  RC*
211   02   02
212   97  DSZ
213   07   07
214   59  INT
215   95   =
216   48  EXC
217   03   03
218   44  SUM
219   01   01
220   48  EXC
221   03   03
222   72  ST*
223   01   01
224   32  X:T
225   75   -
226   01   1
227   95   =
228   42  STO
229   01   01
230   29  CP
231   67   EQ
232   25  CLR
233   94  +/-
234   85   +
235   08   8
236   95   =
237   48  EXC
238   01   01
239   32  X:T
240   61  GTO
241   52  EE
242   76  LBL
243   25  CLR
244   01   1
245   92  RTN
246   76  LBL
247   19  D'
248   85   +
249   32  X:T
```

```
250   07   7
251   85   +
252   43  RCL
253   03   03
254   65   x
255   43  RCL
256   04   04
257   95   =
258   42  STO
259   07   07
260   32  X:T
261   98  ADV
262   92  RTN
263   76  LBL
264   57  ENG
265   73  RC*
266   07   07
267   99  PRT
268   92  RTN
269   01   1
270   44  SUM
271   07   07
272   61  GTO
273   57  ENG
```

```
001   11    A
012   12    B
029   50  IxI
042   13    C
064   14    D
075   15    E
093   33  X²
119   16  A'
123   45  Yx
134   17  B'
155   60  DEG
168   18  C'
179   52  EE
198   59  INT
243   25  CLR
247   19  D'
264   57  ENG
```

ML-04

COMPLEX ARITHMETIC

ML-04 is based upon the formulas given in the Master Library Manual and is a straightforward execution of said formulas. Inputs and outputs are designed to be directly compatible with ML-05 and ML-06 so that the user can switch back and forth at will.

X + Y :

Interface procedure:

(1) Prestore a,b,c, &d according to the table.

(2) Execute PGM 04 B, returns with real part in display and imaginary part in T register.

X - Y :

Interface procedure:

Same as for X + Y except use PGM 04 B'.

Special note:

A subroutine call and two steps could be eliminated by starting the label B sequence where the B is at step 062.

X x Y :

Interface procedure:

Same as for X + Y except use PGM 04 C.

X ÷ Y :

Interface procedure:

Same as for X + Y except use PGM 04 C'.

Special note:

A subroutine call and two steps could be eliminated by starting the label C sequence where the C is at step 119.

Y^X :

Interface procedure:

Same as for X + Y except use PGM 04 D.

Special note:

If the magnitude of Y is zero (c and d are both zero), the correct results appear in the display and T register but not in registers R01 and R02. Thus for chained operations, if Y = 0, you must swap X and Y via label E' after executing D to get the correct final answer. Reentering X as zero will also work.

$\sqrt[X]{Y}$:

Interface procedure:

Same as for X + Y except use PGM 04 E.

Special notes:

(1) If the magnitude of X is zero then this quantity is indeterminate. The program defines it to be 1 + 0i and sets an error state.

(2) If the magnitude of Y is zero the program defines this quantity to be 0 + 0i but does not leave this value in R01 and R02 as the new X. Thus for chained operations you must exchange X and Y via label E' or reenter X as zero.

$\log_Y X$:

Interface procedure:

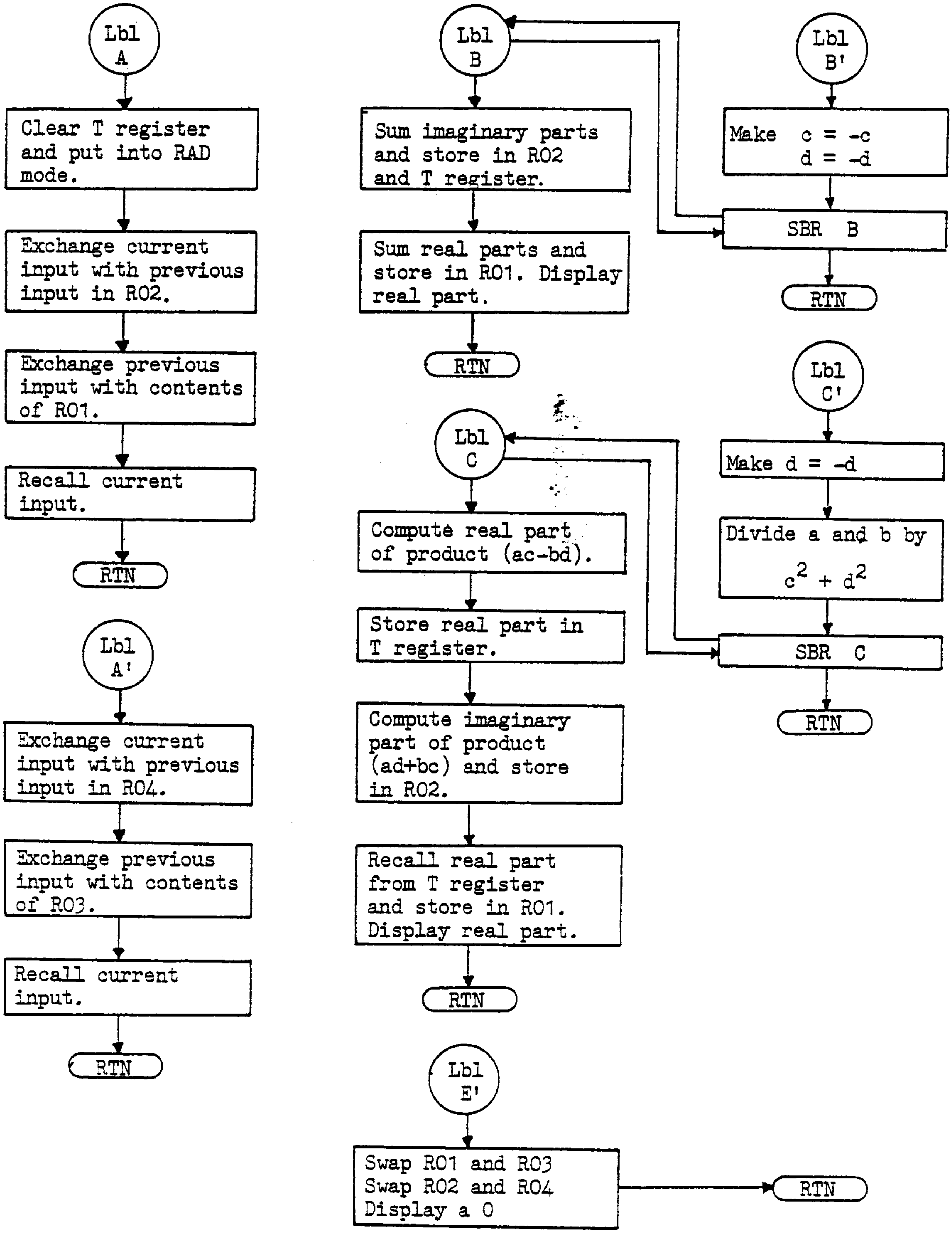
Same as for X + Y except use PGM 04 D'.

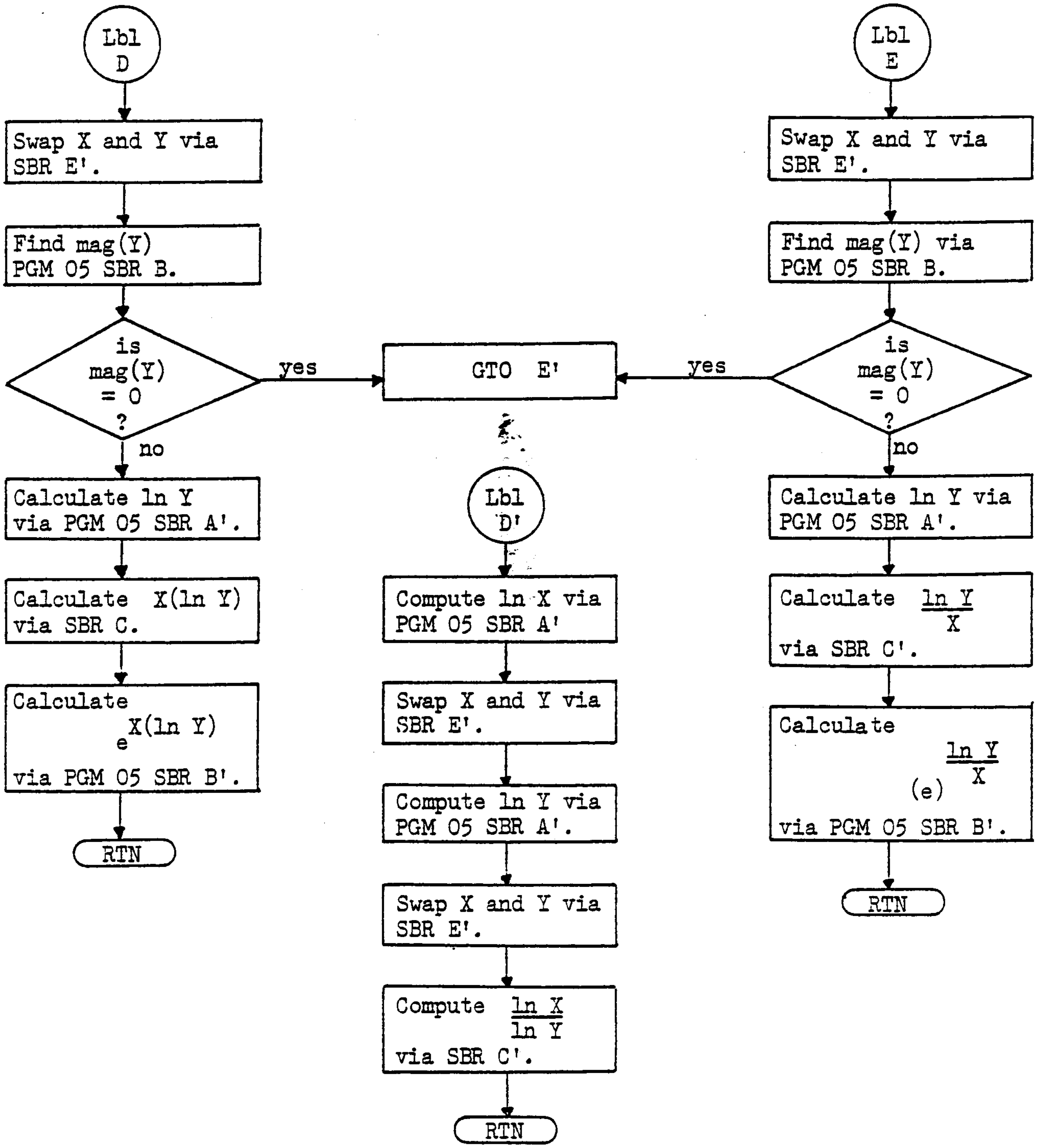
Register assignments are:

OPERATION		R01	R02	R03	R04	T reg	Dis reg	( ) level	SBR level
X + Y	INITIAL	a	b	c	d	*	---	0	0
	FINAL	a + c	b + d	c	d	b + d	a + c		
X - Y	INITIAL	a	b	c	d	*	---	0	1
	FINAL	a - c	b - d	-c	-d	b - d	a - c		
X x Y	INITIAL	a	b	c	d	*	---	1	0
	FINAL	ac-bd	ad+bc	c	d	ad+bc	ac-bd		
X ÷ Y	INITIAL	a	b	c	d	*	---	1	1
	FINAL	Re(Z)	Im(Z)	c	-d	Im(Z)	Re(Z)		
Y^x	INITIAL	a	b	c	d	*	---	1	2
	FINAL	Re(Z)	Im(Z)	a	b	Im(Z)	Re(Z)		
$\sqrt[x]{Y}$	INITIAL	a	b	c	d	*	---	1	2
	FINAL	Re(Z)	Im(Z)	c	-d	Im(Z)	Re(Z)		
$\log_Y X$	INITIAL	a	b	c	d	*	---	1	2
	FINAL	Re(Z)	Im(Z)	e	f	Im(Z)	Re(Z)		
X ⇄ Y	INITIAL	a	b	c	d	not used	---	0	0
	FINAL	c	d	a	b		0		

*Normally zero after executing label A but doesn't matter since it is overwritten.

e = Re(ln Y)
f = -Im(ln Y)


Lbl A
Clear T register and put into RAD mode.
Exchange current input with previous input in R02.
Exchange previous input with contents of R01.
Recall current input.
RTN
Lbl A'
Exchange current input with previous input in R04.
Exchange previous input with contents of R03.
Recall current input.
RTN
Lbl B
Sum imaginary parts and store in R02 and T register.
Sum real parts and store in R01. Display real part.
RTN
Lbl B'
Make c = -c d = -d
SBR B
RTN
Lbl C
Compute real part of product (ac-bd).
Store real part in T register.
Compute imaginary part of product (ad+bc) and store in R02.
Recall real part from T register and store in R01. Display real part.
RTN
Lbl C'
Make d = -d
Divide a and b by c2 + d2
SBR C
RTN
Lbl E'
Swap R01 and R03 Swap R02 and R04 Display a 0
RTN


Lbl
D
Swap X and Y via
SBR E'.
Find mag(Y)
PGM 05 SBR B.
is
mag(Y)
= 0
?
yes
no
Calculate ln Y
via PGM 05 SBR A'.
Calculate X(ln Y)
via SBR C.
Calculate
e^X(ln Y)
via PGM 05 SBR B'.
RTN
GTO E'
Lbl
D'
Compute ln X via
PGM 05 SBR A'
Swap X and Y via
SBR E'.
Compute ln Y via
PGM 05 SBR A'.
Swap X and Y via
SBR E'.
Compute ln X / ln Y
via SBR C'.
RTN
Lbl
E
Swap X and Y via
SBR E'.
Find mag(Y) via
PGM 05 SBR B.
is
mag(Y)
= 0
?
yes
no
Calculate ln Y via
PGM 05 SBR A'.
Calculate ln Y / X
via SBR C'.
Calculate
(e)^(ln Y / X)
via PGM 05 SBR B'.
RTN

ML-04 Program Listing

```
000   76 LBL
001   10 E'
002   43 RCL
003   01  01
004   48 EXC
005   03  03
006   42 STO
007   01  01
008   43 RCL
009   02  02
010   48 EXC
011   04  04
012   42 STO
013   02  02
014   00  0
015   92 RTN
016   76 LBL
017   11  A
018   29 CP
019   70 RAD
020   48 EXC
021   02  02
022   48 EXC
023   01  01
024   43 RCL
025   02  02
026   24 CE
027   92 RTN
028   76 LBL
029   16 A'
030   48 EXC
031   04  04
032   48 EXC
033   03  03
034   43 RCL
035   04  04
036   24 CE
037   92 RTN
038   76 LBL
039   12  B
040   43 RCL
041   04  04
042   44 SUM
043   02  02
044   43 RCL
045   02  02
046   32 X:T
047   43 RCL
048   03  03
049   44 SUM
050   01  01
051   43 RCL
052   01  01
053   92 RTN
054   76 LBL
055   17 B'
056   01  1
057   94 +/-
058   49 PRD
059   03  03
060   49 PRD
061   04  04
062   12  B
063   92 RTN
064   76 LBL
065   13  C
066   53  (
067   43 RCL
068   01  01
069   65  ×
070   43 RCL
071   03  03
072   75  -
073   43 RCL
074   02  02
075   65  ×
076   43 RCL
077   04  04
078   54  )
079   32 X:T
080   53  (
081   43 RCL
082   01  01
083   65  ×
084   43 RCL
085   04  04
086   85  +
087   43 RCL
088   02  02
089   65  ×
090   43 RCL
091   03  03
092   54  )
093   42 STO
094   02  02
095   32 X:T
096   42 STO
097   01  01
098   92 RTN
099   76 LBL
100   18 C'
101   01  1
102   94 +/-
103   49 PRD
104   04  04
105   53  (
106   43 RCL
107   03  03
108   33 X²
109   85  +
110   43 RCL
111   04  04
112   33 X²
113   54  )
114   35 1/X
115   49 PRD
116   01  01
117   49 PRD
118   02  02
119   13  C
120   92 RTN
121   76 LBL
122   14  D
123   10 E'
124   36 PGM
125   05  05
126   12  B
127   29 CP
128   67  EQ
129   10 E'
130   36 PGM
131   05  05
132   16 A'
133   13  C
134   36 PGM
135   05  05
136   17 B'
137   92 RTN
138   76 LBL
139   19 D'
140   36 PGM
141   05  05
142   16 A'
143   10 E'
144   36 PGM
145   05  05
146   16 A'
147   10 E'
148   18 C'
149   92 RTN
150   76 LBL
151   15  E
152   10 E'
153   36 PGM
154   05  05
155   12  B
156   29 CP
157   67  EQ
158   10 E'
159   36 PGM
160   05  05
161   16 A'
162   18 C'
163   36 PGM
164   05  05
165   17 B'
166   92 RTN

001   10 E'
017   11  A
029   16 A'
039   12  B
055   17 B'
065   13  C
100   18 C'
122   14  D
139   19 D'
151   15  E
```

ML-05

COMPLEX FUNCTIONS

ML-05 mechanizes the formulas given in the Master Library Manual. Note that the angle of the number in the complex plane is determined from the P/R conversion to insure that it is in the proper quadrant, rather than from the ARCTAN function as implied.

r, θ:

Interface procedure:

If a and b are already prestored in R01 and R02 respectively, PGM 05 B is a handy routine to display r and leave θ in the T register.

X^2

Interface procedure:

(1) Prestore a and b in R01 and R02 respectively.
(2) Execute PGM 05 C ....returns with real part in R01 and display, imaginary part in R02 and T register.

$\sqrt{X}$, $1/X$, $\ln X$, e^X:

Interface procedure:

Same as for X^2 but use the appropriate user defined key.

Special notes and applications:

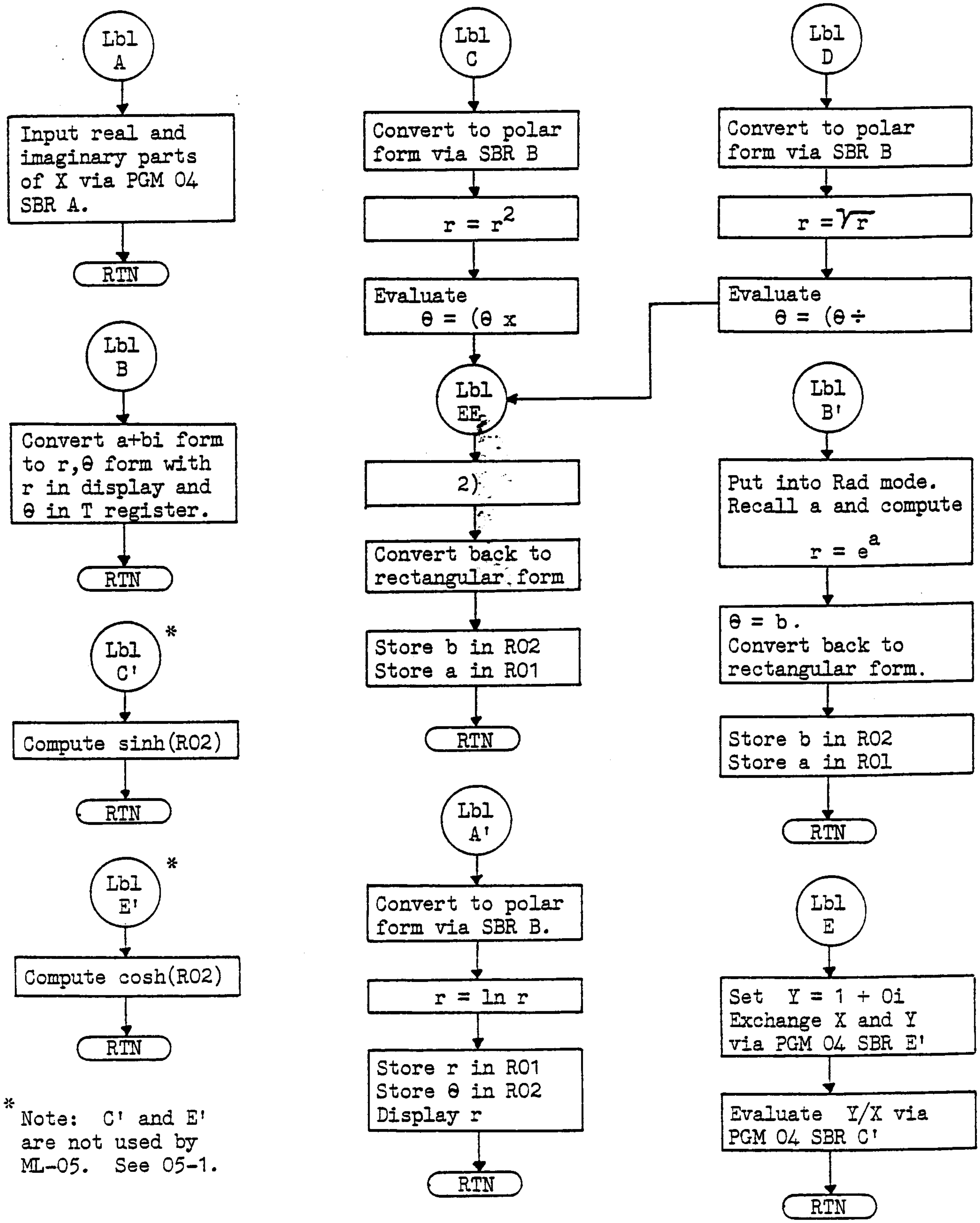
(1) Routines C' and E' can be used to calculate sinh x and cosh x respectively, with x in R02. (x must be in radians) Note that these routines are not used by ML-05 at all but but by ML-06. It would have been better to label them something else and put them in ML-06 since they could then be accessed by ML-06 with SBR ___ instead of PGM 05 ___ which is obviously longer (and probably slower).
(2) PGM 05 SBR 052 will change a polar form, with r in display and θ in R02, to rectangular form, with x in display and R01, and y in T register and R02.
(3) Note that steps 038-044 could be eliminated by X⇄T GTO 056, saving a net 3 steps. Steps 113-118 could be eliminated by GTO 009, saving a net 3 steps. The RAD at step 014 appears to have no significance since the machine is already in RAD mode when routines E' and C' are called by ML-06. (The given addresses would have to be changed after eliminating steps.)

Special notes and applications (cont):

(4) Steps 073-079 could be eliminated by GTO 055, saving a net 4 steps.

Register assignments are:

OPERATION		R01	R02	R03	R04	T reg	Dis reg	( ) level	SBR level
r,θ	INITIAL	a	b			---	---	0	0
	FINAL	a	b			θ	r		
X^2	INITIAL	a	b			---	---	1	1
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
$\sqrt{X}$	INITIAL	a	b			---	---	1	1
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
1/X	INITIAL	a	b	---	---	---	---	1	2
	FINAL	Re(Z)	Im(Z)	-a	-b	Im(Z)	Re(Z)		
ln X	INITIAL	a	b			---	---	0	1
	FINAL	ln a	θ			θ	ln a		
e^X	INITIAL	a	b			---	---	0	0
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
sinh X	INITIAL		x				---	2	0
	FINAL		x				sinhx		
cosh X	INITIAL		x				---	2	0
	FINAL		x				coshx		


Lbl A
Input real and imaginary parts of X via PGM 04 SBR A.
RTN
Lbl B
Convert a+bi form to r,θ form with r in display and θ in T register.
RTN
Lbl C' *
Compute sinh(R02)
RTN
Lbl E' *
Compute cosh(R02)
RTN
Lbl C
Convert to polar form via SBR B
r = r²
Evaluate θ = (θ x
Lbl EE
2)
Convert back to rectangular form
Store b in R02
Store a in R01
RTN
Lbl A'
Convert to polar form via SBR B.
r = ln r
Store r in R01
Store θ in R02
Display r
RTN
Lbl D
Convert to polar form via SBR B
r = √r
Evaluate θ = (θ ÷
Lbl B'
Put into Rad mode. Recall a and compute r = eª
θ = b. Convert back to rectangular form.
Store b in R02
Store a in R01
RTN
Lbl E
Set Y = 1 + 0i
Exchange X and Y via PGM 04 SBR E'
Evaluate Y/X via PGM 04 SBR C'
RTN

* Note: C' and E' are not used by ML-05. See 05-1.

ML-05 Program Listing

```
000  76  LBL
001  10  E'
002  53   (
003  53   (
004  43  RCL
005  02   02
006  22  INV
007  23  LNX
008  85   +
009  35  1/X
010  54   )
011  55   ÷
012  02   2
013  54   )
014  70  RAD
015  92  RTN
016  76  LBL
017  11   A
018  36  PGM
019  04   04
020  11   A
021  92  RTN
022  76  LBL
023  12   B
024  70  RAD
025  43  RCL
026  01   01
027  32  X:T
028  43  RCL
029  02   02
030  22  INV
031  37  P/R
032  32  X:T
033  92  RTN
034  76  LBL
035  16  A'
036  12   B
037  23  LNX
038  42  STO
039  01   01
040  32  X:T
041  42  STO
042  02   02
043  32  X:T
044  92  RTN
045  76  LBL
046  17  B'
047  70  RAD
048  43  RCL
049  01   01
```

```
050  22  INV
051  23  LNX
052  32  X:T
053  43  RCL
054  02   02
055  37  P/R
056  42  STO
057  02   02
058  32  X:T
059  42  STO
060  01   01
061  92  RTN
062  76  LBL
063  13   C
064  12   B
065  33  X²
066  53   (
067  32  X:T
068  65   ×
069  76  LBL
070  52  EE
071  02   2
072  54   )
073  37  P/R
074  42  STO
075  02   02
076  32  X:T
077  42  STO
078  01   01
079  92  RTN
080  76  LBL
081  14   D
082  12   B
083  34  √X
084  53   (
085  32  X:T
086  55   ÷
087  61  GTO
088  52  EE
089  76  LBL
090  15   E
091  01   1
092  42  STO
093  03   03
094  00   0
095  42  STO
096  04   04
097  36  PGM
098  04   04
099  10  E'
```

```
100  36  PGM
101  04   04
102  18  C'
103  92  RTN
104  76  LBL
105  18  C'
106  53   (
107  53   (
108  43  RCL
109  02   02
110  22  INV
111  23  LNX
112  75   -
113  35  1/X
114  54   )
115  55   ÷
116  02   2
117  54   )
118  92  RTN
```

```
001  10  E'
017  11   A
023  12   B
035  16  A'
046  17  B'
063  13   C
070  52  EE
081  14   D
090  15   E
105  18  C'
```

ML-06

COMPLEX TRIGONOMETRIC FUNCTIONS

ML-06 is a continuation of the tradition of ML-04 and ML-05. The user should be aware however that the inverse trigonometric functions for complex arguments are actually multivalued functions (true for real numbers also, since they are simply a special case). T.I.'s source for the formulas given in the M.L.M. was Handbook of Mathematical Functions, U.S. Dept. of Commerce, National Bureau of Standards, Applied Math Series #55, Topic 4.4.37-4.4.39. The complete formulas are:

arcsin $Z = k\pi + (-1)^k$ (formula in M.L.M.)

arccos $Z = 2k\pi \pm$ (formula in M.L.M.)

arctan $Z = k\pi +$ (formula in M.L.M.) if $Z^2 \neq -1$

where k is an integer or zero.

Sin X: A slightly different formula than that given in M.L.M. is actually used:

$$\sin X = \sin a \cosh b + i \cos a \sinh b$$

Interface procedure:

(1) Prestore a and b in R01 and R02 respectively.
(2) Execute PGM 06 B ....returns with real part in R01 and display, imaginary part in R02 and T register.

Cos X: Again, a slightly different formula is used:

$$\cos X = \cos a \cosh b - i \sin a \sinh b$$

Interface procedure:

Same as for Sin X except use PGM 06 C.

Tan X, arcsin X, arccos X, and arctan X:

Execution is a straightforward mechanization of the formulas given in the M.L.M.

Interface procedure:

Same as for Sin X except use appropriate user defined key.

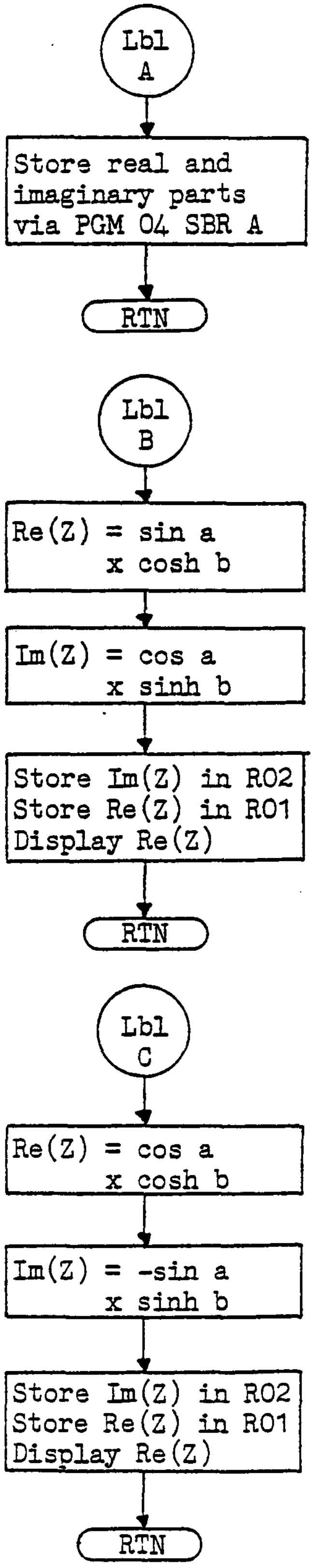
Special notes and applications:

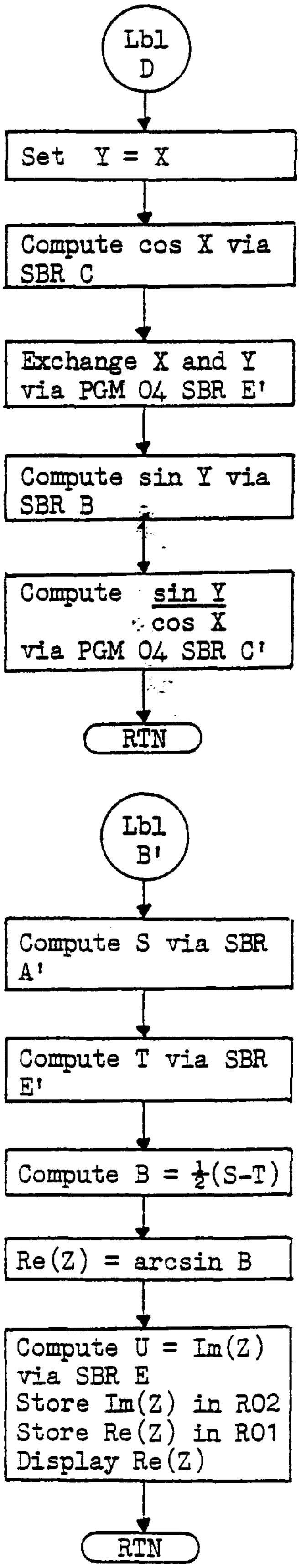
See last page of program listing.

Register assignments are:

OPERATION		R01	R02	R03	R04	T reg	Dis reg	( ) level	SBR level
sin X	INITIAL	a	b			---	---	3	1
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
cos X	INITIAL	a	b			---	---	3	1
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
tan X	INITIAL	a	b	---	---	---	---	3	2
	FINAL	Re(Z)	Im(Z)	e*	f*	Im(Z)	Re(Z)		
arcsin X	INITIAL	a	b			---	---	4	2
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
arccos X	INITIAL	a	b			---	---	4	2
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		
arctan X	INITIAL	a	b			---	---	3	0
	FINAL	Re(Z)	Im(Z)			Im(Z)	Re(Z)		

* e = Re(cos Z)
f = -Im(cos Z)





$$S = ((a+1)^2 + b^2)^{\frac{1}{2}}$$

$$T = ((a-1)^2 + b^2)^{\frac{1}{2}}$$

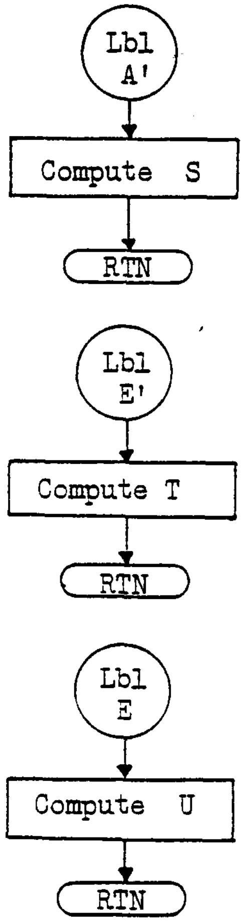
$$A = \tfrac{1}{2}(S+T)$$

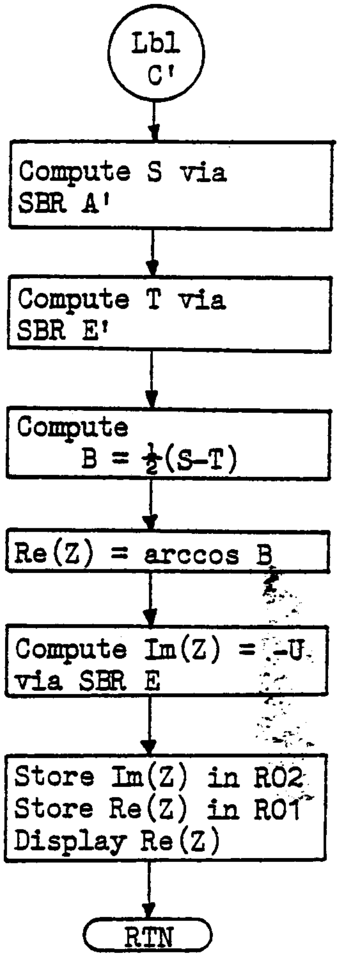
$$B = \tfrac{1}{2}(S-T)$$

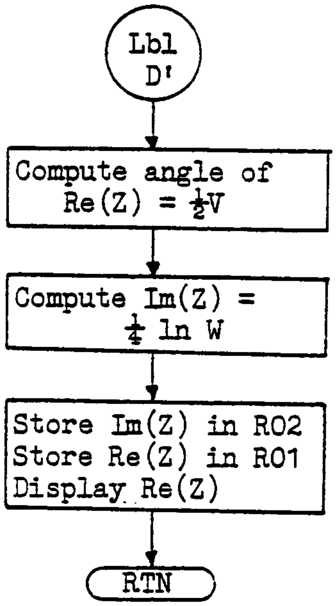
$$U = \ln(A + (A^2 - 1)^{\frac{1}{2}})$$

$$V = (1 - a^2 - b^2) + i(2a)$$

$$W = \frac{a^2 + (b+1)^2}{a^2 + (b-1)^2}$$




Lbl C'
Compute S via SBR A'
Compute T via SBR E'
Compute B = ½(S-T)
Re(Z) = arccos B
Compute Im(Z) = -U via SBR E
Store Im(Z) in R02
Store Re(Z) in R01
Display Re(Z)
RTN


Lbl D'
Compute angle of Re(Z) = ½V
Compute Im(Z) = ½ ln W
Store Im(Z) in R02
Store Re(Z) in R01
Display Re(Z)
RTN

ML-06 Program Listing

```
000  76  LBL
001  16  A'
002  70  RAD
003  53   (
004  53   (
005  43  RCL
006  01   01
007  85   +
008  01   1
009  54   )
010  33  X²
011  85   +
012  43  RCL
013  02   02
014  33  X²
015  54   )
016  34  √X
017  92  RTN
018  76  LBL
019  10  E'
020  53   (
021  53   (
022  43  RCL
023  01   01
024  75   -
025  01   1
026  54   )
027  33  X²
028  85   +
029  43  RCL
030  02   02
031  33  X²
032  54   )
033  34  √X
034  92  RTN
035  76  LBL
036  15   E
037  53   (
038  53   (
039  16  A'
040  85   +
041  10  E'
042  54   )
043  55   ÷
044  02   2
045  85   +
046  53   (
047  52  EE
048  22  INV
049  52  EE
050  33  X²
051  75   -
052  01   1
053  54   )
054  34  √X
055  54   )
056  23  LNX
057  92  RTN
058  76  LBL
059  11   A
060  36  PGM
061  04   04
062  11   A
063  92  RTN
064  76  LBL
065  12   B
066  70  RAD
067  53   (
068  43  RCL
069  01   01
070  38  SIN
071  65   ×
072  36  PGM
073  05   05
074  10  E'
075  54   )
076  32  X:T
077  53   (
078  43  RCL
079  01   01
080  39  COS
081  65   ×
082  36  PGM
083  05   05
084  18  C'
085  54   )
086  42  STO
087  02   02
088  32  X:T
089  42  STO
090  01   01
091  92  RTN
092  76  LBL
093  13   C
094  70  RAD
095  53   (
096  43  RCL
097  01   01
098  39  COS
099  65   ×
100  36  PGM
101  05   05
102  10  E'
103  54   )
104  32  X:T
105  53   (
106  43  RCL
107  01   01
108  38  SIN
109  94  +/-
110  65   ×
111  36  PGM
112  05   05
113  18  C'
114  54   )
115  42  STO
116  02   02
117  32  X:T
118  42  STO
119  01   01
120  92  RTN
121  76  LBL
122  14   D
123  43  RCL
124  01   01
125  42  STO
126  03   03
127  43  RCL
128  02   02
129  42  STO
130  04   04
131  13   C
132  36  PGM
133  04   04
134  10  E'
135  12   B
136  36  PGM
137  04   04
138  18  C'
139  92  RTN
140  76  LBL
141  17  B'
142  53   (
143  53   (
144  16  A'
145  75   -
146  10  E'
147  54   )
148  55   ÷
149  02   2
150  54   )
151  22  INV
152  38  SIN
153  32  X:T
154  15   E
155  42  STO
156  02   02
157  32  X:T
158  42  STO
159  01   01
160  92  RTN
161  76  LBL
162  18  C'
163  53   (
164  53   (
165  16  A'
166  75   -
167  10  E'
168  54   )
169  55   ÷
170  02   2
171  54   )
172  22  INV
173  39  COS
174  32  X:T
175  15   E
176  94  +/-
177  42  STO
178  02   02
179  32  X:T
180  42  STO
181  01   01
182  92  RTN
183  76  LBL
184  19  D'
185  53   (
186  53   (
187  01   1
188  75   -
189  43  RCL
190  01   01
191  33  X²
192  75   -
193  43  RCL
194  02   02
195  33  X²
196  54   )
197  32  X:T
198  53   (
199  02   2
```

ML-06 Program Listing (cont.)

```
200  65   ×
201  43  RCL
202  01   01
203  54   )
204  22  INV
205  37  P/R
206  55   ÷
207  02   2
208  54   )
209  32  X:T
210  53   (
211  53   (
212  53   (
213  43  RCL
214  01   01
215  33  X²
216  85   +
217  53   (
218  43  RCL
219  02   02
220  85   +
221  01   1
222  54   )
223  33  X²
224  54   )
225  55   ÷
226  53   (
227  43  RCL
228  01   01
229  33  X²
230  85   +
231  53   (
232  43  RCL
233  02   02
234  75   -
235  01   1
236  54   )
237  33  X²
238  54   )
239  54   )
240  23  LNX
241  55   ÷
242  04   4
243  54   )
244  42  STO
245  02   02
246  32  X:T
247  42  STO
248  01   01
249  92  RTN
```

```
001  16  A'
019  10  E'
036  15   E
059  11   A
065  12   B
093  13   C
122  14   D
141  17  B'
162  18  C'
184  19  D'
```

Special notes:

(1) Steps 025-034 could be eliminated with a GTO 008, saving a net 7 steps.

(2) Steps 110-120 could be eliminated with a GTO 081, saving a net 7 steps

(3) Steps 155-160, 177-182, and 244-249 could be eliminated with a GTO 086, saving a net 9 steps.

(4) These are only some of the more obvious faults. Changing the entire program structure could result in considerably more savings, particularly if use is made of the ML-04 and ML-05 functions.

ML-07

POLYNOMIAL EVALUATION

ML-07 evaluates a polynomial with real coefficients for any real value of x by the method of synthetic substitution. Consider a general third order polynomial:

$$P(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

This can also be rearranged and written as:

$$P(x) = a_0 + x(a_1 + x(\underbrace{a_2 + x(a_3)}_{Q_2}))$$

$$P(x) = a_0 + x(\underbrace{a_1 + x(Q_2)}_{Q_1})$$

$$P(x) = \underbrace{a_0 + x(Q_1)}_{}$$

$$P(x) = Q_0$$

How to squeeze an answer out of your calculator...

Thus it is obvious that this procedure could be followed for any order polynomial by a simple loop which calculates:

$$Q_{n-1} = a_{n-1} + (Q_n)(x)$$

and then plugs the new "Q value" back in on the next loop.

Synthetic substitution has many advantages over the direct use of the Y^x function; primarily exact integer outputs for integer inputs and coefficients (of reasonable size) and the ability to handle negative inputs.

Register assignments:

R01:	pointer	R03:	x (input)	R05-R(n+5):	coefficients; a_0,
R02:	counter (i)	R04:	n (order)		$a_1, a_2, \ldots a_n$

Interface procedures:

Since program execution is quite simple, little improvement can be made in program length or number of data registers (as far as interfacing is concerned. You may however wish to bypass the print commands in routines A and B by prestoring the order and coefficients in the indicated registers and then executing PGM 07 C.

Interface procedures (cont,):

If you absolutely must suppress any printing then the following routine will perform the same function as ML-07:

```
Lbl A RCL*11 = x RCL 12 + Dsz 11 A RCL 00 = INVSBR
```

See Appendix C concerning synthesizing Dsz 11. Prestore coefficients starting with a_0 in R00 up to a_{10} in R10. R11 holds the order (n) which must be prestored each time the routine is called. R12 holds the value to be input (x). The maximum order n can be changed by changing the register assignments for x and n.

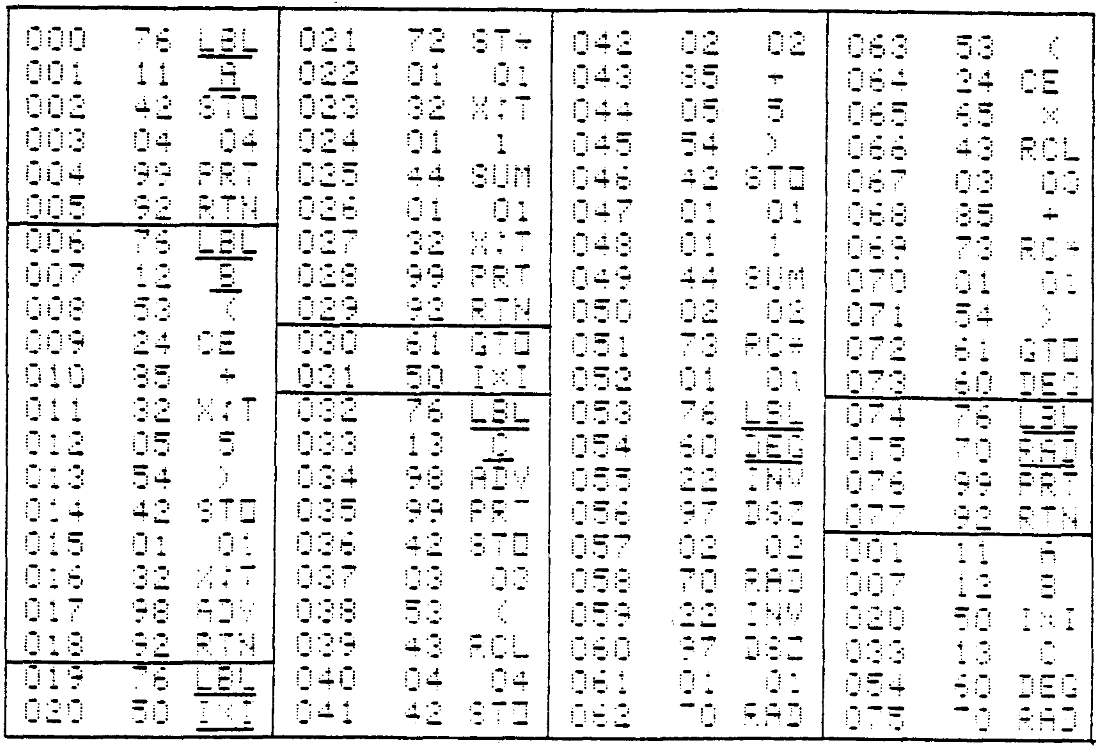
ML-07 normal use data:

Flags used: none
Parentheses levels: 1
Subroutine levels: none

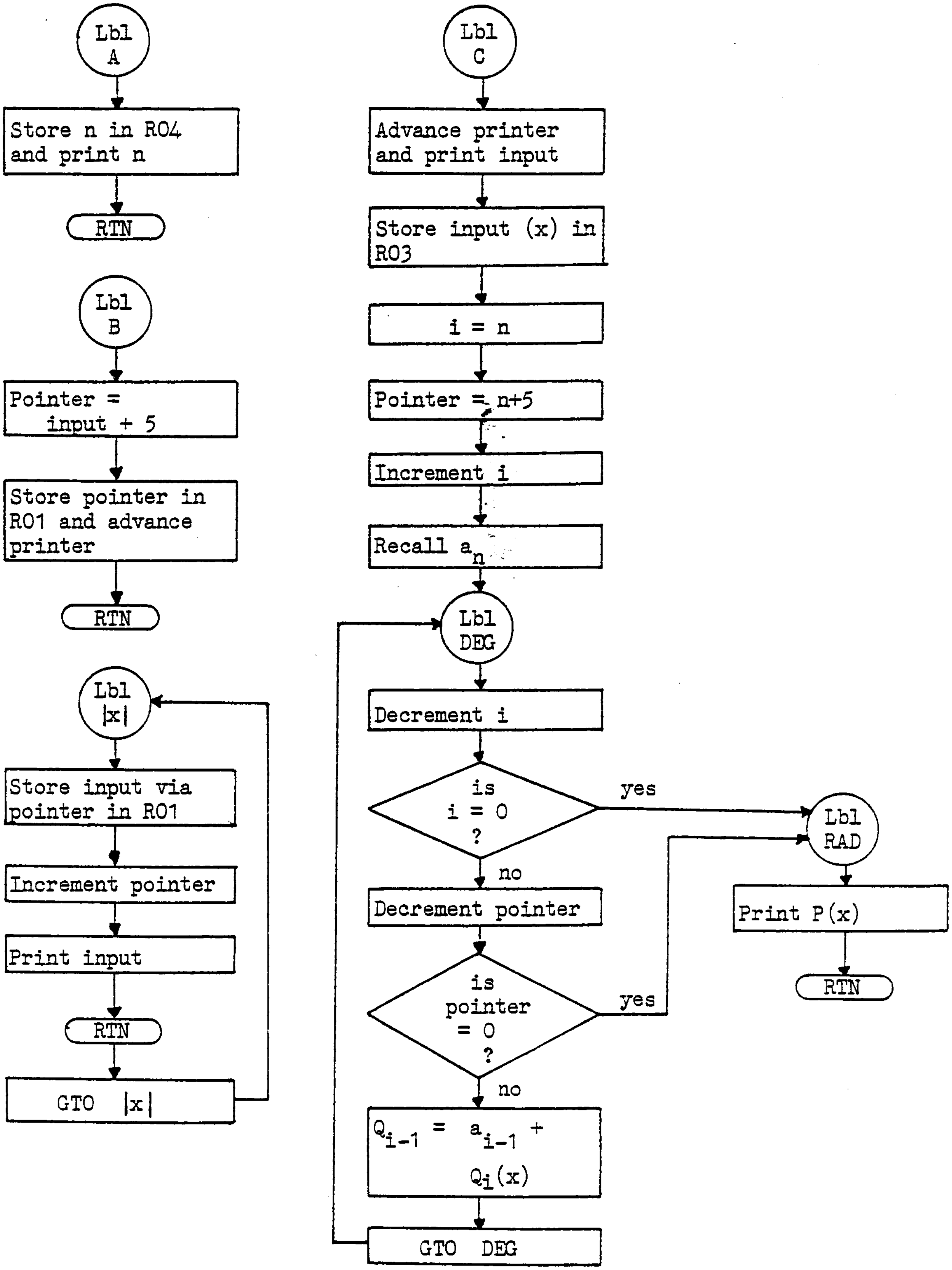
Special applications:

PGM 07 SBR |x| might be a useful input routine. A pointer stored in R01 determines the location of the input data and is incremented during each execution.

ML-07 Program Listing



```
000  76  LBL
001  11   A
002  42  STO
003  04   04
004  99  PRT
005  92  RTN
006  76  LBL
007  12   B
008  53   (
009  24  CE
010  85   +
011  32  X:T
012  05   5
013  54   )
014  42  STO
015  01   01
016  32  X:T
017  98  ADV
018  92  RTN
019  76  LBL
020  50  IXI

021  72  ST*
022  01   01
023  32  X:T
024  01   1
025  44  SUM
026  01   01
027  32  X:T
028  99  PRT
029  92  RTN
030  61  GTO
031  50  IXI
032  76  LBL
033  13   C
034  98  ADV
035  99  PRT
036  42  STO
037  03   03
038  53   (
039  43  RCL
040  04   04
041  42  STO

042  02   02
043  85   +
044  05   5
045  54   )
046  42  STO
047  01   01
048  01   1
049  44  SUM
050  02   02
051  73  RC*
052  01   01
053  76  LBL
054  60  DEG
055  22  INV
056  97  DSZ
057  02   02
058  70  RAD
059  22  INV
060  97  DSZ
061  01   01
062  70  RAD

063  53   (
064  24  CE
065  65   x
066  43  RCL
067  03   03
068  85   +
069  73  RC*
070  01   01
071  54   )
072  61  GTO
073  60  DEG
074  76  LBL
075  70  RAD
076  99  PRT
077  92  RTN
001  11   A
007  12   B
020  50  IXI
033  13   C
054  60  DEG
075  70  RAD
```

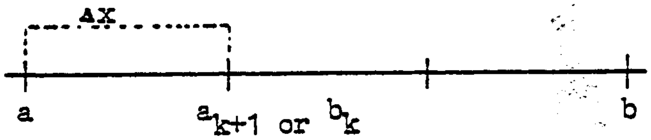

Lbl A
Store n in R04 and print n
RTN
Lbl B
Pointer = input + 5
Store pointer in R01 and advance printer
RTN
Lbl |x|
Store input via pointer in R01
Increment pointer
Print input
RTN
GTO |x|
Lbl C
Advance printer and print input
Store input (x) in R03
i = n
Pointer = n+5
Increment i
Recall a_n
Lbl DEG
Decrement i
is i = 0 ?
yes
no
Decrement pointer
is pointer = 0 ?
yes
no
$Q_{i-1} = a_{i-1} + Q_i(x)$
GTO DEG
Lbl RAD
Print P(x)
RTN

ML-08

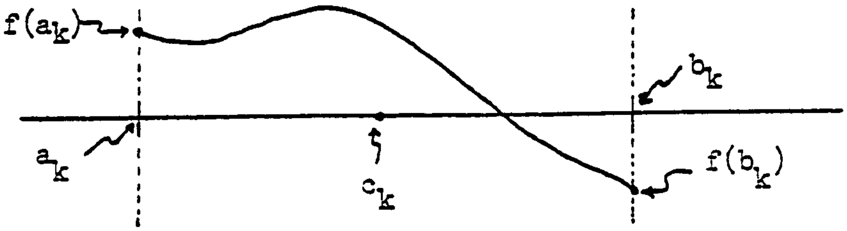
ZEROS OF FUNCTIONS

ML-08 uses the bisection method to find one root in a sampling interval, delta x, where the function changes sign an odd number of times. The program will not find a root on any interval where the function changes sign an even number of times. Roots which are maximum or minimum points are ignored unless by chance they are an interval endpoint.

Execution starts at the lower limit with an interval of delta x. If no sign change is detected between endpoints, the next interval is checked and so on. This is the purpose of the first loop in the flowchart label E.



If a sign change is detected then the interval is progressively halved until the root is found to some arbitrary input error limit. This is the purpose of the second loop in the flowchart label E.



Register assignments are:

	R01	R02	R03	R04	R05	R06	R07	R08
INITIAL	a	b	Δx	---	---	---	---	E
FINAL*	a_{k+1}	b	Δx	a_{k+n}	b_{k+n}	root	$f(a_k)$	E

*After each root is found.

$$\text{root} = c_{k+n} = \frac{a_{k+n} + b_{k+n}}{2}$$

Interface procedure:

(1) Prestore values for a, b, Δx, and E in the indicated registers.

(2) Enter f(x) in main program according to user instructions in M.L.M.

(3) Execute PGM 08 E for each root...returns with root in display and in R06.

(4) Use flag 7 to detect the error states indicating no more roots or an undefined point.

ML-08 normal use data:

Flags used: none
Parentheses levels: 2 (contrary to M.L.M.)
Subroutine levels: 1

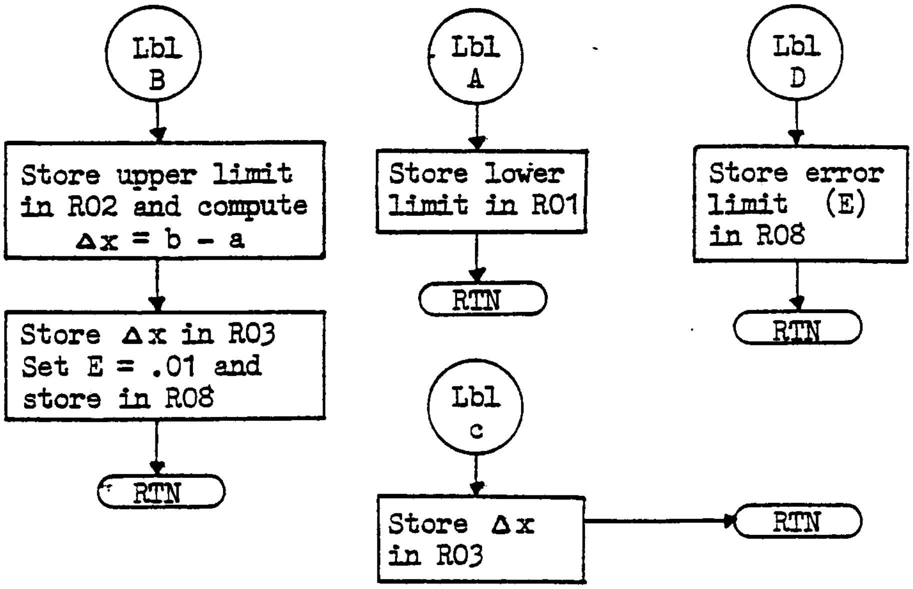
Special notes:

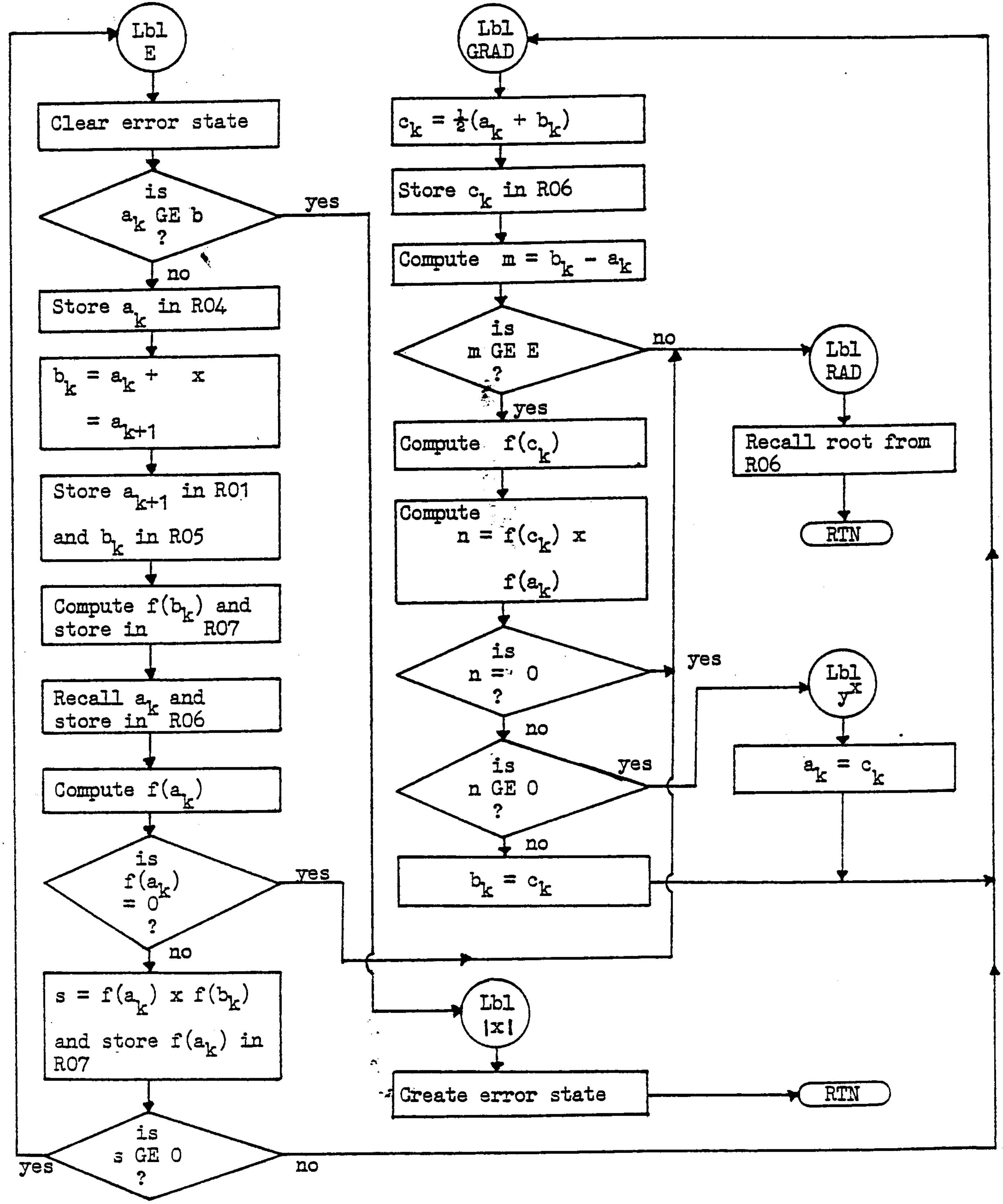
(1) Contrary to M.L.M., register 09 is not used. (pg 28)

(2) Contrary to M.L.M., the use of = in f(x) is permissible if there are no pending operations you need to protect while using ML-08.

Special applications:

If R05-R04 is less than R08 then PGM 08 SBR GRAD will calculate the average of R05 and R04 and store it in R06. If the inequality is not satisfied and A' is not defined, the results are the same but an error state is created and program execution halts.


Lbl B
Store upper limit in R02 and compute Δx = b - a
Store Δx in R03 Set E = .01 and store in R08
RTN
Lbl A
Store lower limit in R01
RTN
Lbl c
Store Δx in R03
RTN
Lbl D
Store error limit (E) in R08
RTN


Lbl E
Clear error state
is a_k GE b ?
yes
no
Store a_k in R04
$b_k = a_k +$ x $= a_{k+1}$
Store a_{k+1} in R01 and b_k in R05
Compute $f(b_k)$ and store in R07
Recall a_k and store in R06
Compute $f(a_k)$
is $f(a_k) = 0$?
yes
no
$s = f(a_k)$ x $f(b_k)$ and store $f(a_k)$ in R07
is s GE 0 ?
yes
no
Lbl GRAD
$c_k = \frac{1}{2}(a_k + b_k)$
Store c_k in R06
Compute $m = b_k - a_k$
is m GE E ?
no
yes
Compute $f(c_k)$
Compute $n = f(c_k)$ x $f(a_k)$
is n = 0 ?
yes
no
is n GE 0 ?
yes
no
$b_k = c_k$
Lbl |x|
Create error state
RTN
Lbl RAD
Recall root from R06
RTN
Lbl y^x
$a_k = c_k$

ML-08 Program Listing

```
000  76  LBL
001  11   A
002  42  STO
003  01   01
004  92  RTN
005  76  LBL
006  12   B
007  53   (
008  42  STO
009  02   02
010  75   -
011  32  X:T
012  43  RCL
013  01   01
014  54   )
015  42  STO
016  03   03
017  93   .
018  00   0
019  01   1
020  42  STO
021  08   08
022  32  X:T
023  92  RTN
024  76  LBL
025  13   C
026  42  STO
027  03   03
028  92  RTN
029  76  LBL
030  14   D
031  42  STO
032  08   08
033  92  RTN
034  76  LBL
035  15   E
036  53   (
037  24  CE
038  43  RCL
039  02   02
040  32  X:T
041  43  RCL
042  01   01
043  77   GE
044  50  I×I
045  42  STO
046  04   04
047  85   +
048  43  RCL
049  03   03
```

```
050  54   )
051  42  STO
052  01   01
053  42  STO
054  05   05
055  36  PGM
056  00   00
057  16  A'
058  42  STO
059  07   07
060  53   (
061  43  RCL
062  04   04
063  42  STO
064  06   06
065  36  PGM
066  00   00
067  16  A'
068  29  CP
069  67   EQ
070  70  RAD
071  65   ×
072  48  EXC
073  07   07
074  54   )
075  77   GE
076  15   E
077  76  LBL
078  80  GRD
079  53   (
080  53   (
081  43  RCL
082  04   04
083  85   +
084  43  RCL
085  05   05
086  54   )
087  55   ÷
088  02   2
089  54   )
090  42  STO
091  06   06
092  43  RCL
093  08   08
094  32  X:T
095  53   (
096  43  RCL
097  05   05
098  75   -
099  43  RCL
```

```
100  04   04
101  54   )
102  22  INV
103  77   GE
104  70  RAD
105  53   (
106  43  RCL
107  06   06
108  36  PGM
109  00   00
110  16  A'
111  65   ×
112  43  RCL
113  07   07
114  54   )
115  29  CP
116  67   EQ
117  70  RAD
118  77   GE
119  45  YX
120  43  RCL
121  06   06
122  42  STO
123  05   05
124  61  GTO
125  80  GRD
126  76  LBL
127  45  YX
128  43  RCL
129  06   06
130  42  STO
131  04   04
132  61  GTO
133  80  GRD
134  76  LBL
135  70  RAD
136  43  RCL
137  06   06
138  92  RTN
139  76  LBL
140  50  I×I
141  00   0
142  35  1/X
143  92  RTN
```

```
001  11   A
006  12   B
025  13   C
030  14   D
035  15   E
078  80  GRD
127  45  YX
135  70  RAD
140  50  I×I
```

ML-09

SIMPSON'S APPROXIMATION (CONTINUOUS)

ML-09 evaluates the definite integral of a user defined function over a specified interval using Simpson's Rule. An interesting feature of Simpson's Rule is that it yields the exact answer for polynomials of third degree or less (subject to display rounding).

Register assignments are:

	R01	R02	R03	R04	R05
INITIAL	x_o	x_n	h	---	n (i)
FINAL	x_o	x_n	h	I	0

Interface procedure:

(1) Enter f(x) as specified by the M.L.M.

(2) Store x_o and x_n in the indicated registers.

(3) Enter n (must be even) and execute PGM 09 C.

(4) Execute PGM 09 D ....returns with I in display and in R04

ML-09 normal use data:

Flags used: none
Parentheses levels: 2
Subroutine levels: 1

Special note:

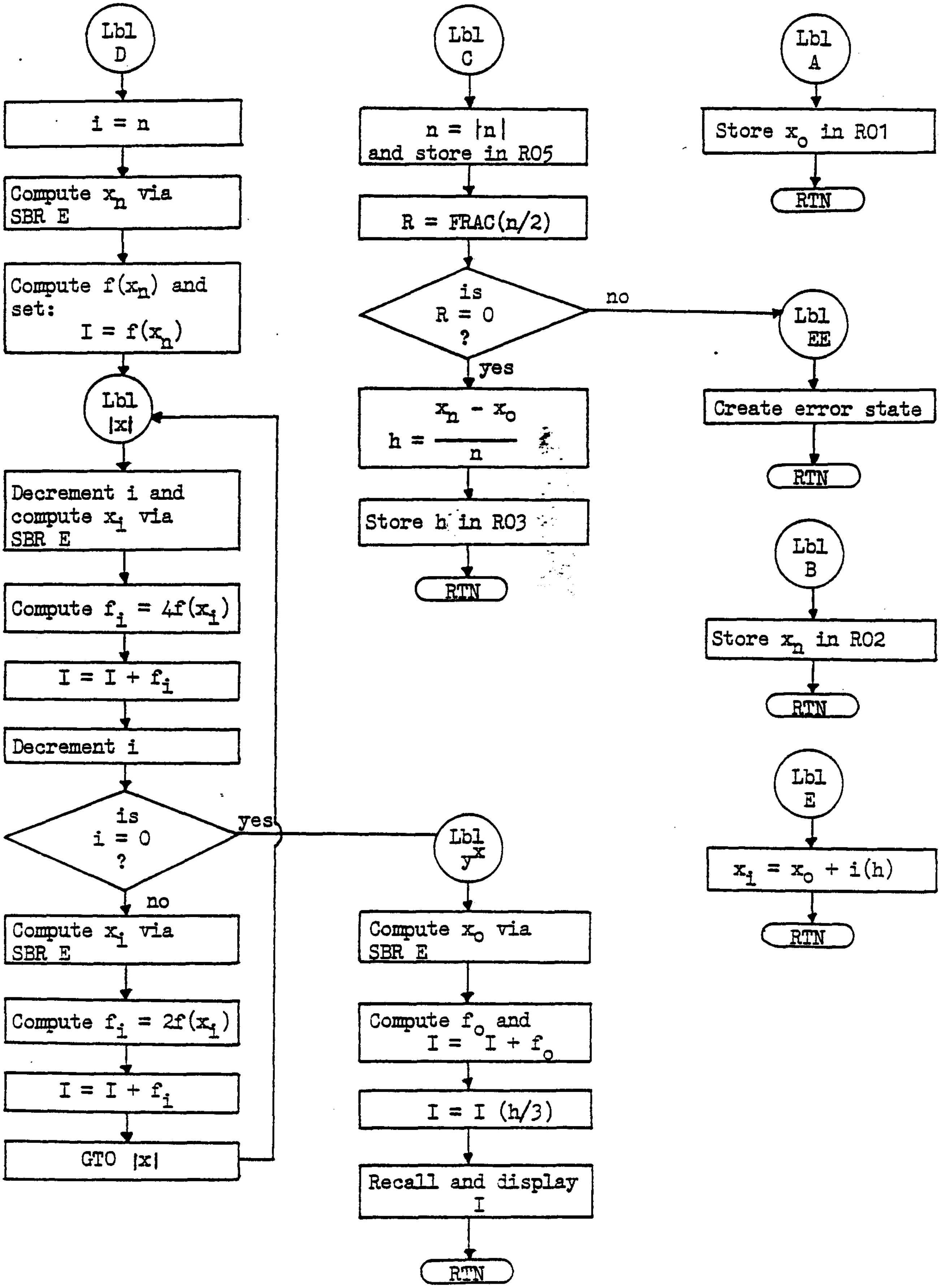
Contrary to M.L.M., the use of = in the f(x) subroutine is permissible as long as no pending operations have to be preserved while using ML-09.

Special applications:

(1) PGM 09 E evaluates R01 ÷ (R03)(R05)

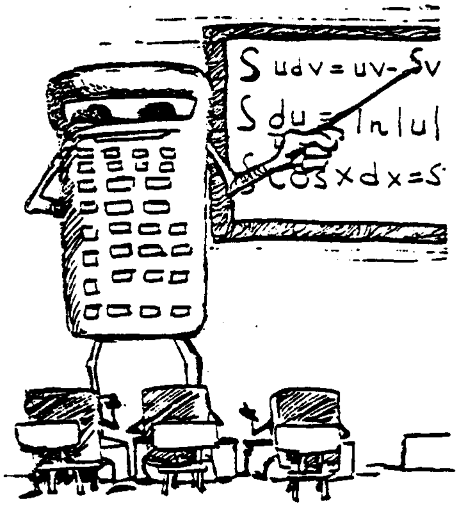
(2) PGM 09 SBR 038 evaluates R03 = (R02-R01)/R05

(3) PGM 09 SBR 107 evaluates R04 = (R04)(R03)/3


Lbl D
i = n
Compute x_n via SBR E
Compute $f(x_n)$ and set: $I = f(x_n)$
Lbl |x|
Decrement i and compute x_i via SBR E
Compute $f_i = 4f(x_i)$
$I = I + f_i$
Decrement i
is i = 0 ?
yes
no
Compute x_i via SBR E
Compute $f_i = 2f(x_i)$
$I = I + f_i$
GTO |x|
Lbl C
n = |n| and store in R05
R = FRAC(n/2)
is R = 0 ?
no
yes
$h = \frac{x_n - x_o}{n}$
Store h in R03
RTN
Lbl y^x
Compute x_o via SBR E
Compute f_o and $I = I + f_o$
I = I (h/3)
Recall and display I
RTN
Lbl A
Store x_o in R01
RTN
Lbl EE
Create error state
RTN
Lbl B
Store x_n in R02
RTN
Lbl E
$x_i = x_o + i(h)$
RTN

ML-09 Program Listing

```
000   76  LBL
001   15   E
002   53   (
003   43  RCL
004   01   01
005   85   +
006   43  RCL
007   05   05
008   65   ×
009   43  RCL
010   03   03
011   54   )
012   92  RTN
013   76  LBL
014   11   A
015   42  STO
016   01   01
017   92  RTN
018   76  LBL
019   12   B
020   42  STO
021   02   02
022   92  RTN
023   76  LBL
024   13   C
025   53   (
026   50  I×I
027   42  STO
028   05   05
029   55   ÷
030   02   2
031   54   )
032   22  INV
033   59  INT
034   29  CP
035   22  INV
036   67   EQ
037   52  EE
038   53   (
039   43  RCL
040   05   05
041   35  1/X
042   65   ×
043   53   (
044   43  RCL
045   02   02
046   75   -
047   43  RCL
048   01   01
049   54   )

050   54   )
051   42  STO
052   03   03
053   92  RTN
054   76  LBL
055   52  EE
056   00   0
057   35  1/X
058   92  RTN
059   76  LBL
060   14   D
061   15   E
062   36  PGM
063   00   00
064   16  A'
065   42  STO
066   04   04
067   76  LBL
068   50  I×I
069   01   1
070   22  INV
071   44  SUM
072   05   05
073   53   (
074   15   E
075   36  PGM
076   00   00
077   16  A'
078   65   ×
079   04   4
080   54   )
081   44  SUM
082   04   04
083   22  INV
084   97  DSZ
085   05   05
086   45  Y×
087   53   (
088   15   E
089   36  PGM
090   00   00
091   16  A'
092   65   ×
093   02   2
094   54   )
095   44  SUM
096   04   04
097   61  GTO
098   50  I×I
099   76  LBL

100   45  Y×
101   15   E
102   36  PGM
103   00   00
104   16  A'
105   44  SUM
106   04   04
107   53   (
108   43  RCL
109   03   03
110   55   ÷
111   03   3
112   54   )
113   49  PRD
114   04   04
115   43  RCL
116   04   04
117   92  RTN

001   15   E
014   11   A
019   12   B
024   13   C
055   52   EE
060   14   D
068   50   I×I
100   45   Y×
```



"Yes, I know Simpson's rule is easier but ....."

ML-10

SIMPSON'S APPROXIMATION (DISCRETE)

ML-10 is a good example of a poor programming approach. Essentially it stores the values of f(x) at discrete points and then after all data is entered, plugs them into a long summation. It's a lot like killing fleas with a sledge hammer...a big waste of effort to accomplish a small task. A much better approach would have been to create each term as that value of f(x) was input and sum into a single register.

Register assignments are:

	R01	R02	R03	R04	R05
INITIAL	i	count	h	----	n
FINAL	6	0	h	I	n

Interface procedure:

If you insist on using ML-10 then simply follow the user instructions and precede each user defined key with PGM 10. In place of the R/S's use PGM 10 SBR |x|.

As an alternative I offer the following routine to be entered somewhere in your program. See Appendix B for an explanation of HIR code 82.

```
000  76  LBL  | 010  04    4  | 020  39  COS  | 030  91  R/S
001  11    A  | 011  54    )  | 021  53    (  | 031  61  GTO
002  42  STO  | 012  44  SUM  | 022  24   CE  | 032  38  SIN
003  01   01  | 013  01   01  | 023  65    ×  | 033  76  LBL
004  91  R/S  | 014  82  HIR  | 024  02    2  | 034  39  COS
005  76  LBL  | 015  11   11  | 025  54    )  | 035  44  SUM
006  38  SIN  | 016  91  R/S  | 026  44  SUM  | 036  01   01
007  53    (  | 017  22  INV  | 027  01   01  | 037  43  RCL
008  24   CE  | 018  97  DSZ  | 028  82  HIR  | 038  01   01
009  65    ×  | 019  02   02  | 029  11   11  | 039  92  RTN
```

To use, first calculate $\frac{1}{2}(n-1)$ in the main program and store in R02. Enter f_o and call routine A. While in routine A, enter f_i and key R/S after each entry. After f_n is entered and you key R/S, the sum $(f_o+4f_1+2f_2+4f_3+....4f_{n-1}+f_n)$ will be recalled from R01 and control will return to main program. To get the integral, multiply this sum by h and divide by 3. Note that only two registers are used vs. n+6 for ML-10.

ML-10 normal use data:

Flags used: none
Parentheses levels: 1
Subroutine levels: 0

Special notes:

n, which must be even, is the number of intervals thus there are n+1 data points, which must be equally spaced of interval length h.

Special applications:

(1) PGM 10 SBR 110 evaluates R04 = (R03)(R04)/3 and prints it.

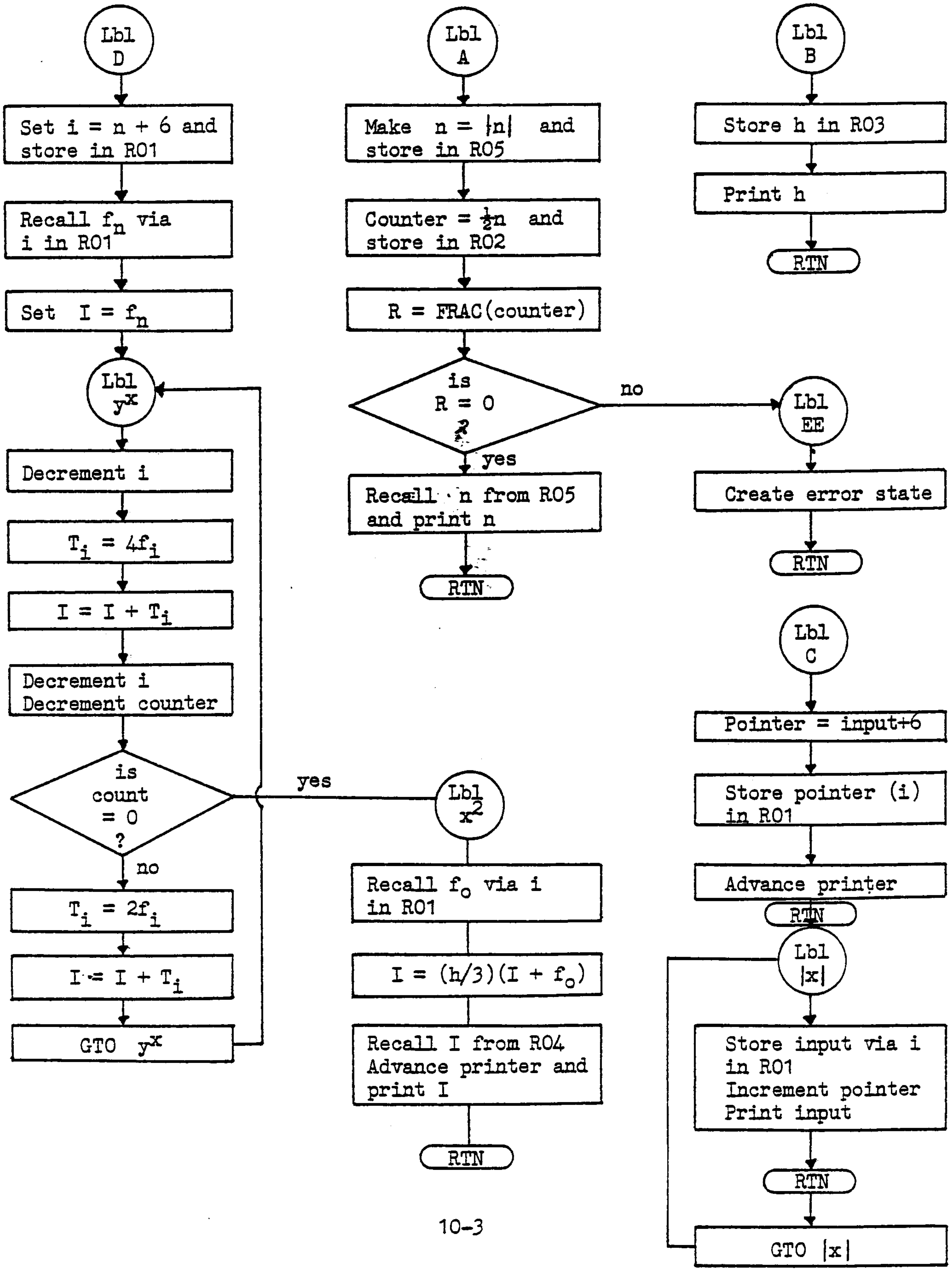
(2) PGM 10 SBR 011 prints the contents of R05 if the display input is an integer or zero; creates an error state for non-integer inputs.

(3) PGM 10 SBR 012 reverses the results of (2).

(4) PGM 10 SBR 013 prints the contents of R05 for an input of zero and creates an error state for all other inputs.

(5) If the T register contains a number Q, PGM 10 SBR 014 prints the contents of R05 for an input equal to Q and creates an error state for all other inputs.

(6) PGM 10 SBR 015 reverses the results of (5).


Lbl D
Set i = n + 6 and store in R01
Recall f_n via i in R01
Set I = f_n
Lbl y^x
Decrement i
$T_i = 4f_i$
$I = I + T_i$
Decrement i
Decrement counter
is count = 0 ?
yes
no
$T_i = 2f_i$
$I = I + T_i$
GTO y^x
Lbl A
Make n = |n| and store in R05
Counter = ½n and store in R02
R = FRAC(counter)
is R = 0 ?
no
yes
Recall n from R05 and print n
RTN
Lbl x^2
Recall f_o via i in R01
$I = (h/3)(I + f_o)$
Recall I from R04
Advance printer and print I
RTN
Lbl B
Store h in R03
Print h
RTN
Lbl EE
Create error state
RTN
Lbl C
Pointer = input+6
Store pointer (i) in R01
Advance printer
RTN
Lbl |x|
Store input via i in R01
Increment pointer
Print input
RTN
GTO |x|

ML-10 Program Listing

```
000  76  LBL
001  11  A
002  53  (
003  50  I×I
004  42  STO
005  05  05
006  55  ÷
007  02  2
008  54  )
009  42  STO
010  02  02
011  22  INV
012  59  INT
013  29  CP
014  22  INV
015  67  EQ
016  52  EE
017  43  RCL
018  05  05
019  99  PRT
020  92  RTN
021  76  LBL
022  52  EE
023  00  0
024  35  1/X
025  92  RTN
026  76  LBL
027  12  B
028  42  STO
029  03  03
030  99  PRT
031  92  RTN
032  76  LBL
033  13  C
034  53  (
035  24  CE
036  85  +
037  32  X:T
038  06  6
039  54  )
040  42  STO
041  01  01
042  32  X:T
043  98  ADV
044  92  RTN
045  76  LBL
046  50  I×I
047  72  ST*
048  01  01
049  32  X:T
050  01  1
051  44  SUM
052  01  01
053  32  X:T
054  99  PRT
055  92  RTN
056  61  GTO
057  50  I×I
058  76  LBL
059  14  D
060  53  (
061  43  RCL
062  05  05
063  85  +
064  06  6
065  54  )
066  42  STO
067  01  01
068  73  RC*
069  01  01
070  42  STO
071  04  04
072  76  LBL
073  45  Y×
074  01  1
075  22  INV
076  44  SUM
077  01  01
078  53  (
079  73  RC*
080  01  01
081  65  ×
082  04  4
083  54  )
084  44  SUM
085  04  04
086  01  1
087  22  INV
088  44  SUM
089  01  01
090  22  INV
091  97  DSZ
092  02  02
093  33  X²
094  53  (
095  73  RC*
096  01  01
097  65  ×
098  02  2
099  54  )
100  44  SUM
101  04  04
102  61  GTO
103  45  Y×
104  76  LBL
105  33  X²
106  73  RC*
107  01  01
108  44  SUM
109  04  04
110  53  (
111  43  RCL
112  03  03
113  55  ÷
114  03  3
115  54  )
116  49  PRD
117  04  04
118  43  RCL
119  04  04
120  98  ADV
121  99  PRT
122  92  RTN

001  11  A
022  52  EE
027  12  B
033  13  C
046  50  I×I
059  14  D
073  45  Y×
105  33  X²
```

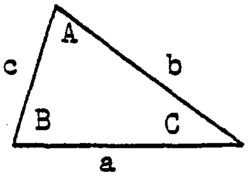
ML-11

TRIANGLE SOLUTION (1)

ML-11 is designed to handle three "types" of triangle solutions; SSS, SSA, and SAS. Each combination will be analyzed separately.

S.S.S.:

This is the simplest case. To evaluate A, B, and C, the law of cosines is applied three times with sides a, b, and c "rotated" through registers 01, 02, and 06. This allows a single routine with fixed register manipulations to be used for each angle.



Register assignments are:

	R01	R02	R03	R04	R05	R06
INITIAL	side b	side c	---	---	---	side a
FINAL	side c	side a	ang. A	ang. B	ang. C	side b

Interface procedure:

(1) Prestore a, b, and c in assigned registers.
(2) Ensure that flags 0 and 1 are not set.
(3) Execute PGM 11 A' ...returns with value of angle A in display.
(4) Recall values directly as needed.

Normal use data:

Flags affected: 0, 1, 2, & 3
Flags used: 0 & 1
Parentheses levels: 2
Subroutine levels: 0

Special notes:

(1) Scientific notation mode is not affected by this portion.
(2) Be sure to select proper angular mode for input data.
(3) Labels B' and C' simply recall previously calculated results.

S.A.S.:

This solution incorporates the SSS solution by reducing the problem to SSS. The law of cosines is used to calculate the remaining side then control is turned over to the SSS solution routine.

Register assignments are:

	R01	R02	R03	R04	R05	R06
INITIAL	side b	ang. c	---	---	---	side a
FINAL	side c	side a	ang. A	ang. B	ang. A	side b

Interface procedure:

(1) Prestore a, b, and angle C in assigned registers.
(2) Ensure that flags 1 and 3 are not set.
(3) Execute PGM 11 E ...returns with side c in display.
(4) Recall or use values directly as needed.

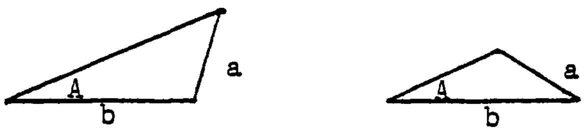
Special notes:

(1) See notes for SSS.
(2) Note that the value of angle C is destroyed during processing.

S.S.A.:

This case incorporates part of the SAS solution. The SSA problem is reduced to a SAS problem using the law of sines and the fact that the sum of all angles equals 180 degrees (or pi radians or 200 grad.). Then side c is calculated from part of the SAS solution routine.

IMPORTANT: This solution neglects the fact that given side b greater than side a, and angle A such that sin A is less than a/b, two solutions exist. It calculates the triangle with the largest area.



Register assignments are:

	R01	R02	R03	R04	R05	R06
INITIAL	side b	ang. A	---	---	---	side a
FINAL	side b	side c	---	ang. B	ang. C	side a

Interface procedure:

(1) Prestore a, b, and angle A in the assigned registers.
(2) Execute PGM 11 D ...returns with value of side c in display.
(3) Recall or use other values as needed.

Normal use data:

Flags affected: 0, 1, 2, & 3
Flags used: 3
Parentheses levels: 2

Special notes:

(1) Flag status is immaterial.
(2) Scientific notation is affected.
(3) The value of angle A is destroyed during processing.

Special applications:

(1) PGM 11 A' can be used to calculate the single angle C given sides a, b, and c in R06, R01, and R02 respectively. The only register affected is R03 where the answer is stored. Set flags 0 and 1, ensure that flag 2 is reset.
(2) PGM 11 E' resets flags 0-3.
(3) PGM 11 SBR 063 or SBR 157 causes a total wipeout.[1] (clears all program memory and data registers)

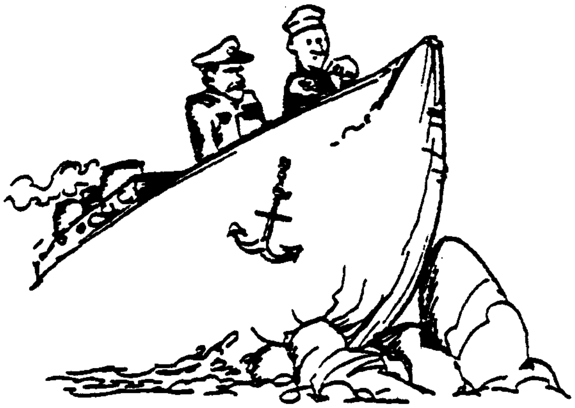
Comment:

Contrary to implication in M.L.M., it is not necessary to input data in a specific order.

Addendum:

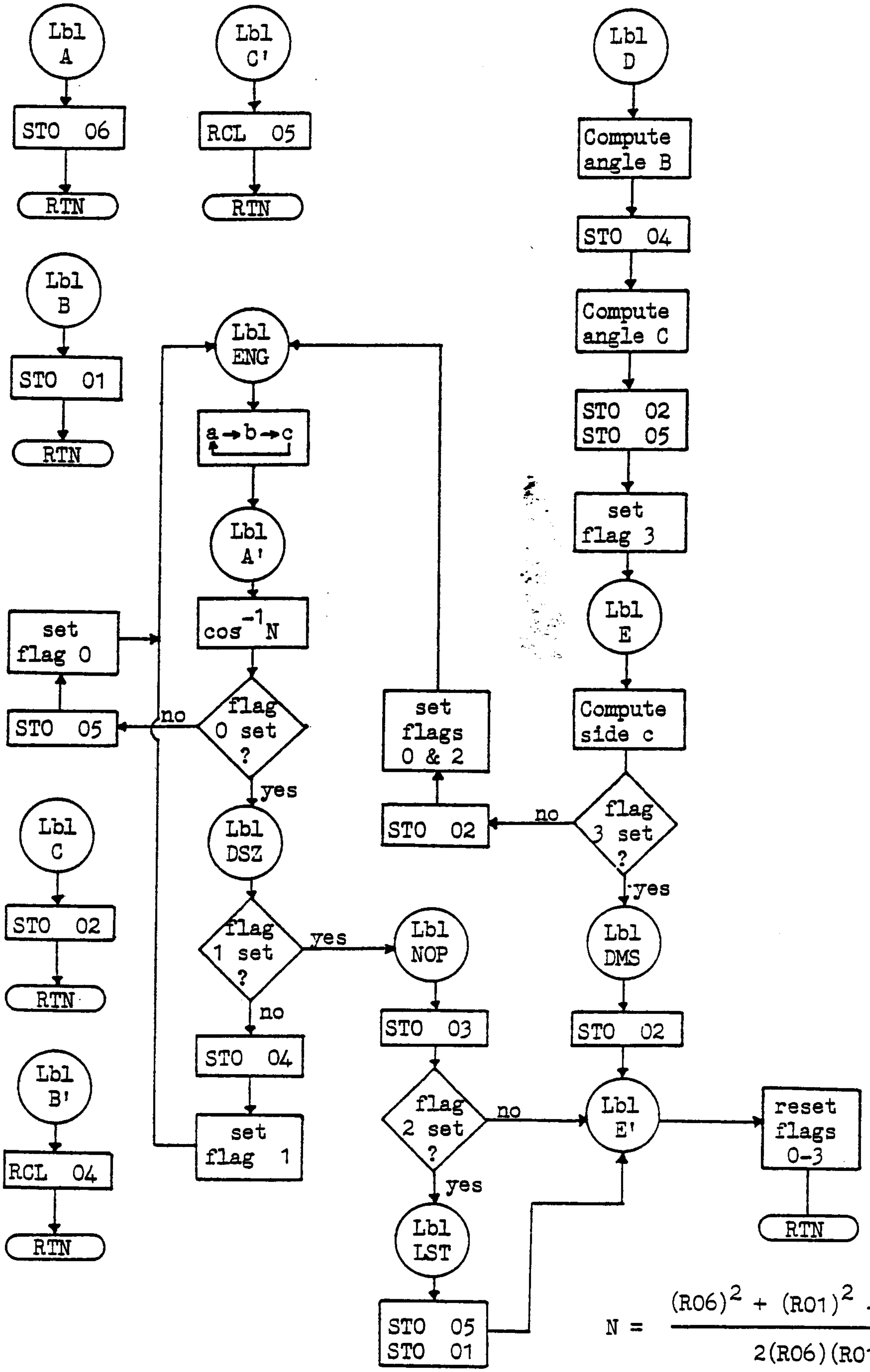
Normal use data for SAS is:

Flags affected: 0, 1, 2, and 3
Flags used: 0, 1, 2, and 3
Parentheses levels: 2



"What do you mean...you used your calculator to 'triangulate' our position......"

[1] SR-52 Notes V3n1P5
(only on printer)



$$N = \frac{(R06)^2 + (R01)^2 - (R02)^2}{2(R06)(R01)}$$

ML-11 Program Listing

```
000  76  LBL
001  16  A'
002  53  (
003  53  (
004  43  RCL
005  06   06
006  33  X²
007  85   +
008  43  RCL
009  01   01
010  33  X²
011  75   -
012  43  RCL
013  02   02
014  33  X²
015  54   )
016  55   ÷
017  02   2
018  55   ÷
019  43  RCL
020  06   06
021  55   ÷
022  43  RCL
023  01   01
024  54   )
025  22  INV
026  39  COS
027  87  IFF
028  00   00
029  97  DSZ
030  42  STO
031  05   05
032  86  STF
033  00   00
034  76  LBL
035  57  ENG
036  43  RCL
037  06   06
038  48  EXC
039  01   01
040  48  EXC
041  02   02
042  42  STO
043  06   06
044  61  GTO
045  16  A'
046  76  LBL
047  97  DSZ
048  87  IFF
049  01   01
050  68  NOP
051  42  STO
052  04   04
053  86  STF
054  01   01
055  61  GTO
056  57  ENG
057  76  LBL
058  68  NOP
059  42  STO
060  03   03
061  87  IFF
062  02   02
063  90  LST
064  76  LBL
065  10  E'
066  22  INV
067  86  STF
068  00   00
069  22  INV
070  86  STF
071  01   01
072  22  INV
073  86  STF
074  02   02
075  22  INV
076  86  STF
077  03   03
078  92  RTN
079  76  LBL
080  14   D
081  53   (
082  53   (
083  43  RCL
084  02   02
085  38  SIN
086  65   ×
087  43  RCL
088  01   01
089  55   ÷
090  43  RCL
091  06   06
092  54   )
093  22  INV
094  52  EE
095  52  EE
096  22  INV
097  52  EE
098  22  INV
099  38  SIN
100  52  EE
101  22  INV
102  52  EE
103  42  STO
104  04   04
105  94  +/-
106  85   +
107  01   1
108  94  +/-
109  22  INV
110  39  COS
111  75   -
112  43  RCL
113  02   02
114  54   )
115  42  STO
116  02   02
117  42  STO
118  05   05
119  86  STF
120  03   03
121  76  LBL
122  15   E
123  53   (
124  43  RCL
125  06   06
126  33  X²
127  85   +
128  43  RCL
129  01   01
130  33  X²
131  75   -
132  02   2
133  65   ×
134  43  RCL
135  06   06
136  65   ×
137  43  RCL
138  01   01
139  65   ×
140  43  RCL
141  02   02
142  39  COS
143  54   )
144  34  √X
145  87  IFF
146  03   03
147  88  DMS
148  42  STO
149  02   02
150  86  STF
151  00   00
152  86  STF
153  02   02
154  61  GTO
155  57  ENG
156  76  LBL
157  90  LST
158  42  STO
159  05   05
160  43  RCL
161  01   01
162  61  GTO
163  10  E'
164  76  LBL
165  88  DMS
166  42  STO
167  02   02
168  61  GTO
169  10  E'
170  76  LBL
171  17  B'
172  43  RCL
173  04   04
174  92  RTN
175  76  LBL
176  18  C'
177  43  RCL
178  05   05
179  92  RTN
180  76  LBL
181  11   A
182  42  STO
183  06   06
184  92  RTN
185  76  LBL
186  12   B
187  42  STO
188  01   01
189  92  RTN
190  76  LBL
191  13   C
192  42  STO
193  02   02
194  92  RTN
```

ML-12

TRIANGLE SOLUTION (2)

ML-12 compliments ML-11 by solving the remaining two types of triangles, ASA and SAA. In addition, it calculates the area.

A.S.A.:

Given two angles, the third is calculated from the fact that the sum of all angles equals 180 degrees (or pi radians, or 200 grad.). Then the two remaining sides are calculated from the law of sines.

Register assignments are:

	R01	R02	R03	R04	R05	R06	R07
INITIAL	---	---	---	ang. B	ang. C	---	side a
FINAL	side b	side c	ang. A	ang. B	ang. C	---	side a

Interface procedure:

(1) Prestore B, C, and a in the assigned registers.
(2) Ensure that flag 0 is not set.
(3) Execute PGM 12 A' ...returns with value of angle A in display.
(4) Recall or use other data directly as needed.

Normal use data:

Flags affected: none
Parentheses levels: 1
Subroutine levels: 0

Special notes:

(1) No data is destroyed or moved during execution.
(2) Be sure to select proper angular mode to fit input data.
(3) If flag 0 is set, the only effect is that A' returns with angle B in display and resets flag 0. Nothing else changes.

S.A.A.:

The data is rearranged to look like the ASA problem then "unscrambled" after using the ASA solution to find the unknown quantities.

Register assignments are:

	R01	R02	R03	R04	R05	R06	R07
INITIAL	---	---	---	ang. A	ang. C	---	side a
FINAL	side b	side c	ang. A	ang. B	ang. C	---	side a

Interface procedure:

(1) Prestore A, C, and a in the assigned registers.
(2) Execute PGM 12 B' ...returns with value of angle B in display.
(3) Recall or use other data directly as needed.

Normal use data:

Flags affected: flag 0
Parentheses levels: 1
Subroutine levels: 0

Special notes:

(1) Be sure to select proper angular mode to fit input data.
(2) The location of angle A is changed during execution.

AREA:

The area is calculated directly via the formulas in the M.L.M.

Register assignments are:

	R01	R02	R03	R04	R05	R06	R07
INITIAL	side b	side c	---	---	---	---	side a
FINAL	side b	side c	---	---	---	s	side a

Interface procedure:

(1) Prestore values of sides a, b, and c in the assigned registers.
(2) Execute PGM 12 C' ...returns with value of area in display and R06.

Normal use data:

Flags affected: none
Parentheses levels: 2
Subroutine levels: 0

Special notes:

(1) Sides a, b, and c are interchangeable.
(2) This is the only part of the program that uses R06.

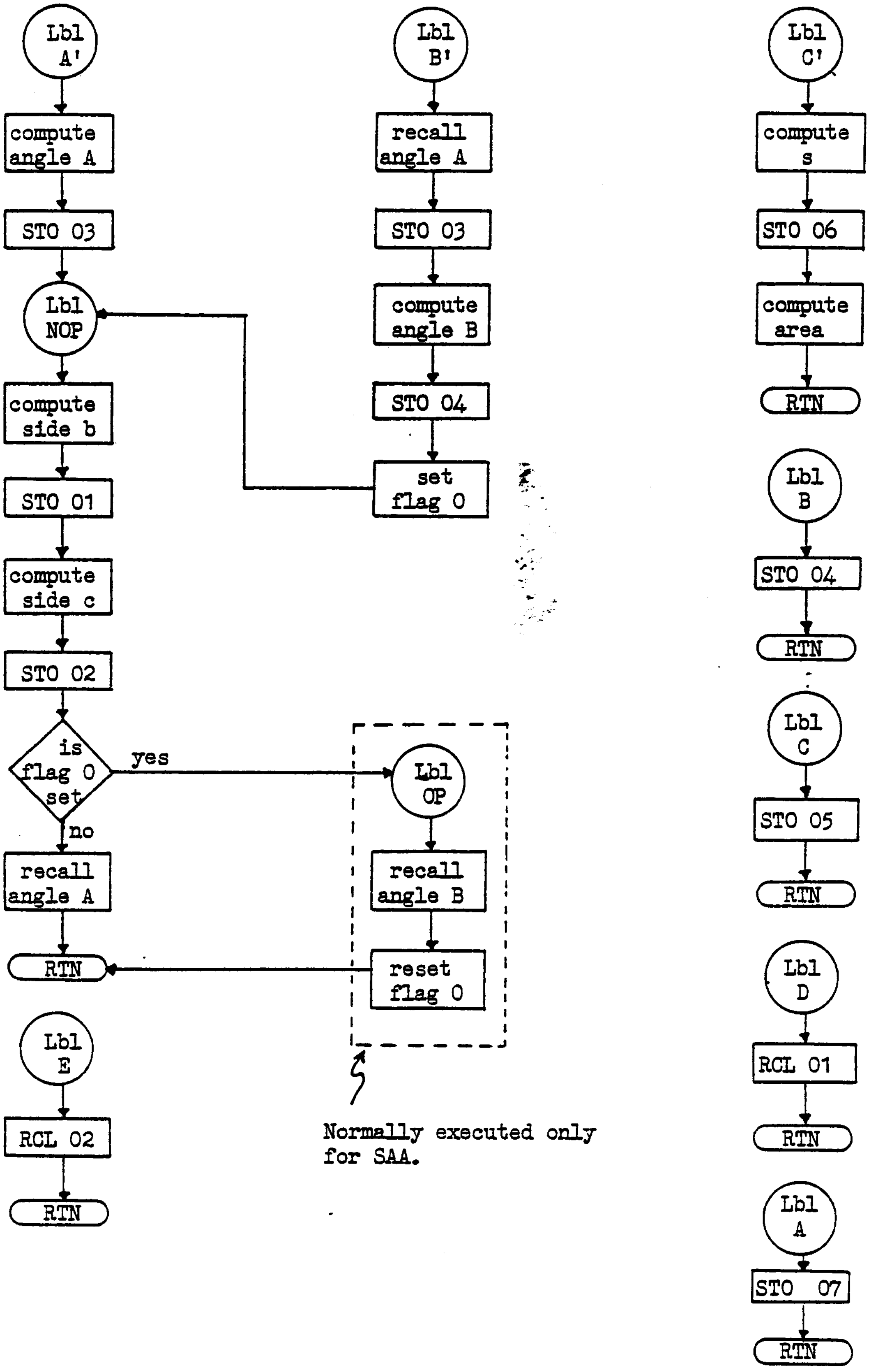
For ML-12:

Special applications:

(1) PGM 12 SBR 083 will recall register 03 if flag 0 is not set or recall register 04 if it is set, then reset the flag.
(2) PGM 12 SBR OP recalls register 04 and resets flag 0. (saves a step)
(3) If the roots of a fourth order polynomial are 0, a, b, and c; then PGM 12 SBR 123 x^2 can be used to evaluate the polynomial for any value in the display. Store the three non-zero roots in R01, R02, and R07. Note that R06 is used to store the input value. If the input is already in R06, then use SBR 125.

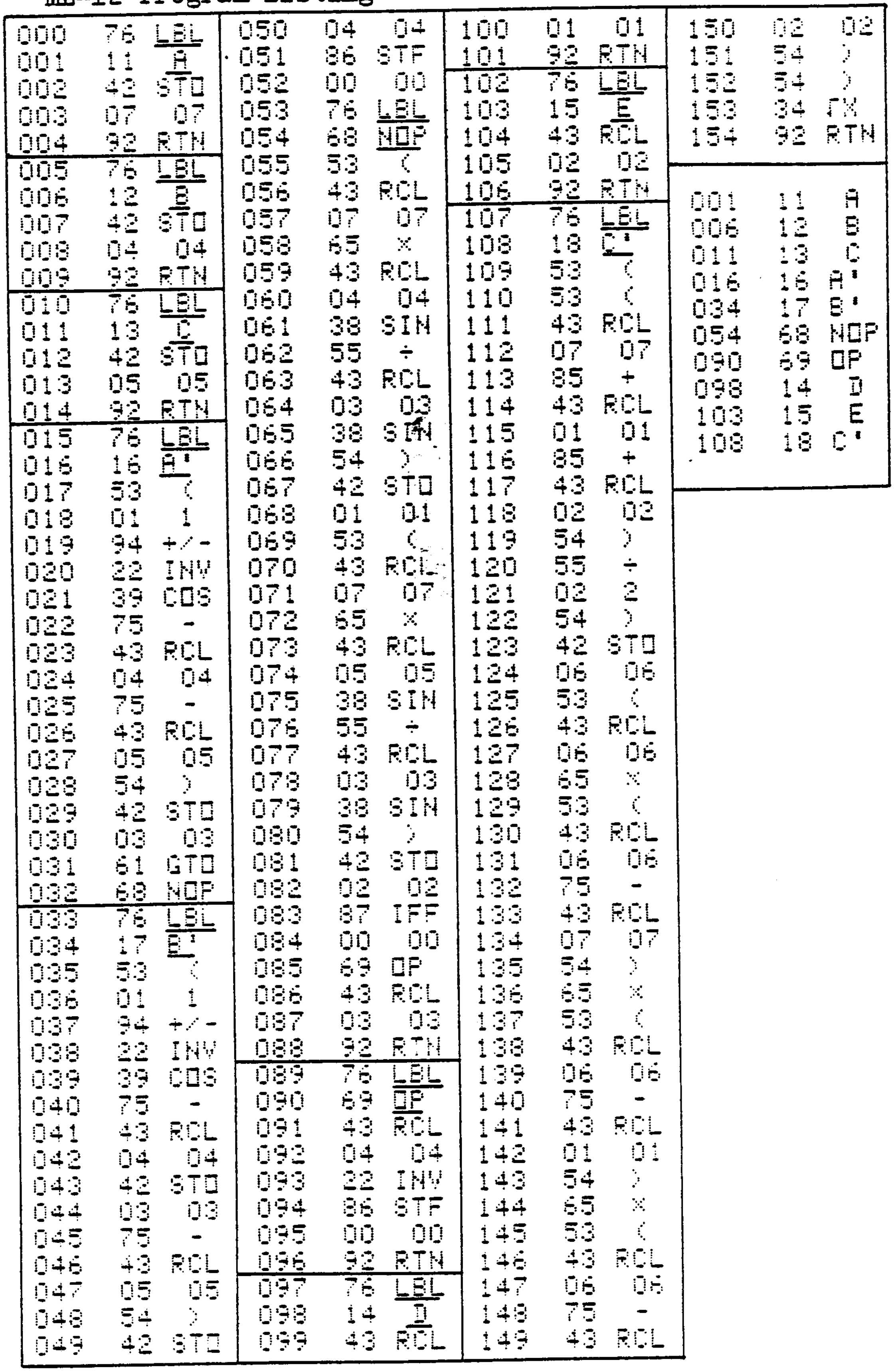
Comments:

(1) Contrary to M.L.M., the law of cosines is not used in this program. Input data need only be reentered for SAA calculations.
(2) Inputs may be entered in any order.



Normally executed only for SAA.

ML-12 Program Listing

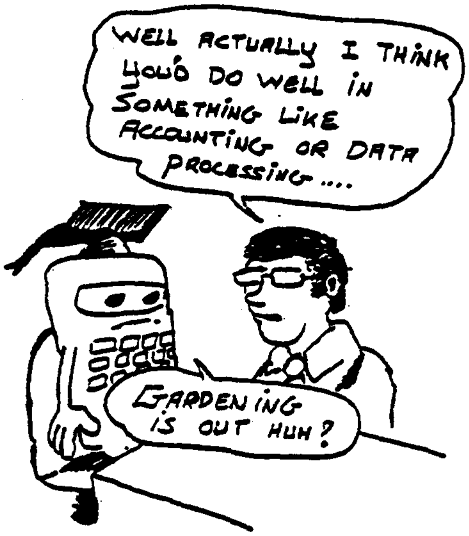


```
000  76  LBL
001  11  A
002  42  STO
003  07  07
004  92  RTN
005  76  LBL
006  12  B
007  42  STO
008  04  04
009  92  RTN
010  76  LBL
011  13  C
012  42  STO
013  05  05
014  92  RTN
015  76  LBL
016  16  A'
017  53  (
018  01  1
019  94  +/-
020  22  INV
021  39  COS
022  75  -
023  43  RCL
024  04  04
025  75  -
026  43  RCL
027  05  05
028  54  )
029  42  STO
030  03  03
031  61  GTO
032  68  NOP
033  76  LBL
034  17  B'
035  53  (
036  01  1
037  94  +/-
038  22  INV
039  39  COS
040  75  -
041  43  RCL
042  04  04
043  42  STO
044  03  03
045  75  -
046  43  RCL
047  05  05
048  54  )
049  42  STO
050  04  04
051  86  STF
052  00  00
053  76  LBL
054  68  NOP
055  53  (
056  43  RCL
057  07  07
058  65  ×
059  43  RCL
060  04  04
061  38  SIN
062  55  ÷
063  43  RCL
064  03  03
065  38  SIN
066  54  )
067  42  STO
068  01  01
069  53  (
070  43  RCL
071  07  07
072  65  ×
073  43  RCL
074  05  05
075  38  SIN
076  55  ÷
077  43  RCL
078  03  03
079  38  SIN
080  54  )
081  42  STO
082  02  02
083  87  IFF
084  00  00
085  69  OP
086  43  RCL
087  03  03
088  92  RTN
089  76  LBL
090  69  OP
091  43  RCL
092  04  04
093  22  INV
094  86  STF
095  00  00
096  92  RTN
097  76  LBL
098  14  D
099  43  RCL
100  01  01
101  92  RTN
102  76  LBL
103  15  E
104  43  RCL
105  02  02
106  92  RTN
107  76  LBL
108  18  C'
109  53  (
110  53  (
111  43  RCL
112  07  07
113  85  +
114  43  RCL
115  01  01
116  85  +
117  43  RCL
118  02  02
119  54  )
120  55  ÷
121  02  2
122  54  )
123  42  STO
124  06  06
125  53  (
126  43  RCL
127  06  06
128  65  ×
129  53  (
130  43  RCL
131  06  06
132  75  -
133  43  RCL
134  07  07
135  54  )
136  65  ×
137  53  (
138  43  RCL
139  06  06
140  75  -
141  43  RCL
142  01  01
143  54  )
144  65  ×
145  53  (
146  43  RCL
147  06  06
148  75  -
149  43  RCL
150  02  02
151  54  )
152  54  )
153  34  √X
154  92  RTN

001  11  A
006  12  B
011  13  C
016  16  A'
034  17  B'
054  68  NOP
090  69  OP
098  14  D
103  15  E
108  18  C'
```

ML-13

CURVE SOLUTION

Although ML-13 is a relatively straightforward execution of the formulas given in the Master Library Manual, it has a few minor programming flaws. Label D' has an extra right hand parenthesis and if called as a subroutine will complete pending operations in the calling routine. Also, flags 0 and 1 are set and reset simultaneously; a single flag would have been just as effective and saved 8 steps (who said two flags were better than one?). Input-output ease leaves a lot to be desired.


WELL ACTUALLY I THINK YOU'D DO WELL IN SOMETHING LIKE ACCOUNTING OR DATA PROCESSING....
GARDENING IS OUT HUH?

Register assignments are:

R01: θ (central angle in radians)
R02: r (radius of circle)
R03: s (arc length)
R04: c (cord length)

Normal use data:

Flags used: flags 0 and 1
Subroutine levels: 1
Parentheses levels: 3

Interface procedure:

θ,r input:

(1) Prestore θ and r in R01 and R02 respectively. ($0 \leq \theta < \pi$, $r \geq 0$)
(2) Execute PGM 13 Z where Z is C', D', E', or E for the desired quantity. Returns with quantity in display only, does not disturb any registers.

θ,s input:

(1) Prestore θ and s in R01 and R03 respectively. ($0 \leq \theta < \pi$)
(2) Ensure that flag 1 is not set.
(3) Execute PGM 13 B' to get r stored in R02.
(4) Go to step (2) of θ,r input solution.

θ,c input:

(1) Prestore θ in R01 and with c in display, execute PGM 13 D. ($0 \leq \theta < \pi$) Note that this sets flags 0 and 1, and stores c in R04.
(2) Go to step (3) of θ,s input solution.

Interface procedure (cont.):

r,s input:

(1) Prestore r and s in R02 and R03 respectively. ($r \geq 0$)
(2) Ensure that flag 0 is not set.
(3) Execute PGM 13 A' to store θ in R01.
(4) Go to step (2) of θ,r input solution.

r,c input:

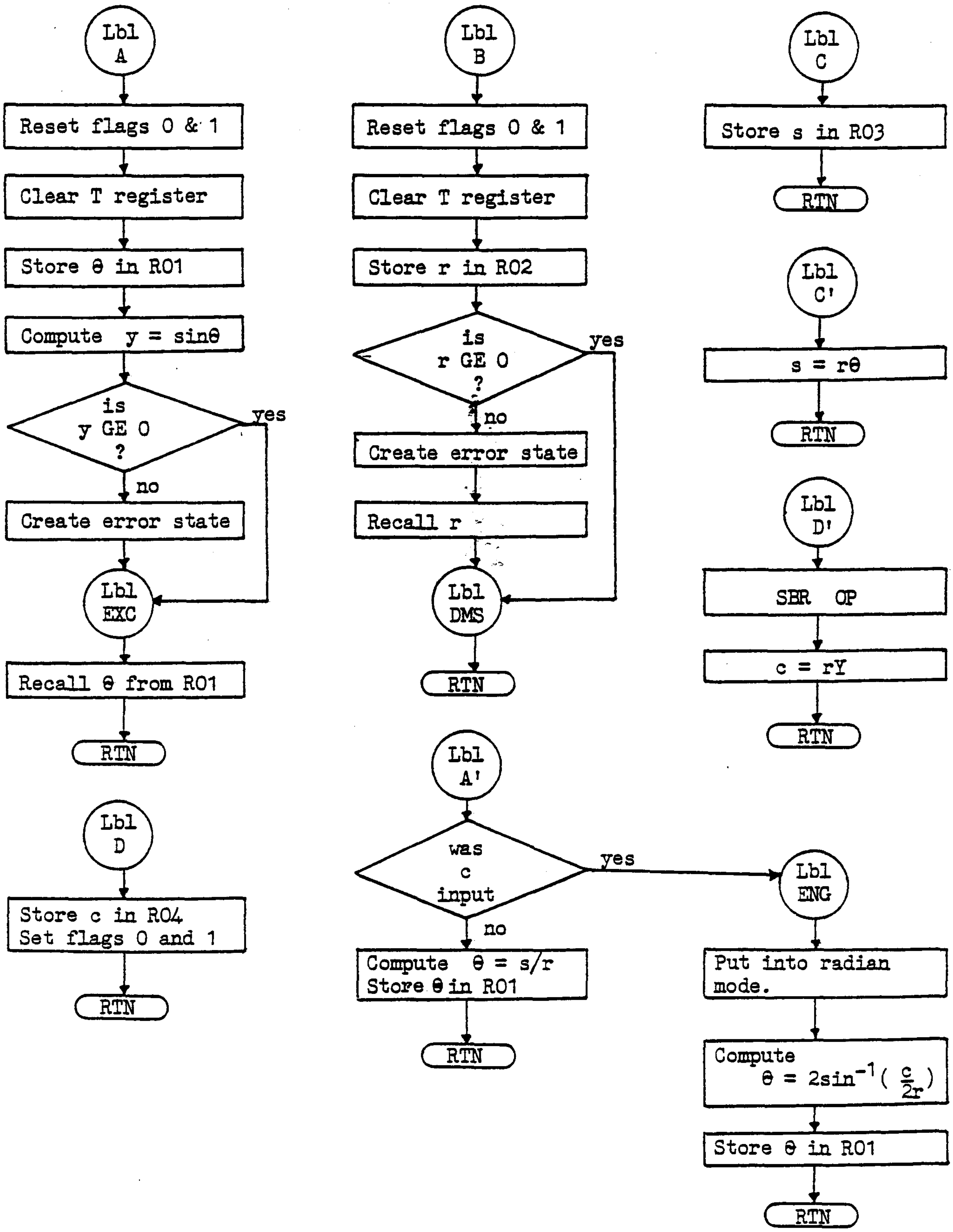
(1) Prestore r in R02 and with c in display, execute PGM 13 D. ($r \geq 0$)
(2) Go to step (2) of r,s input solution.

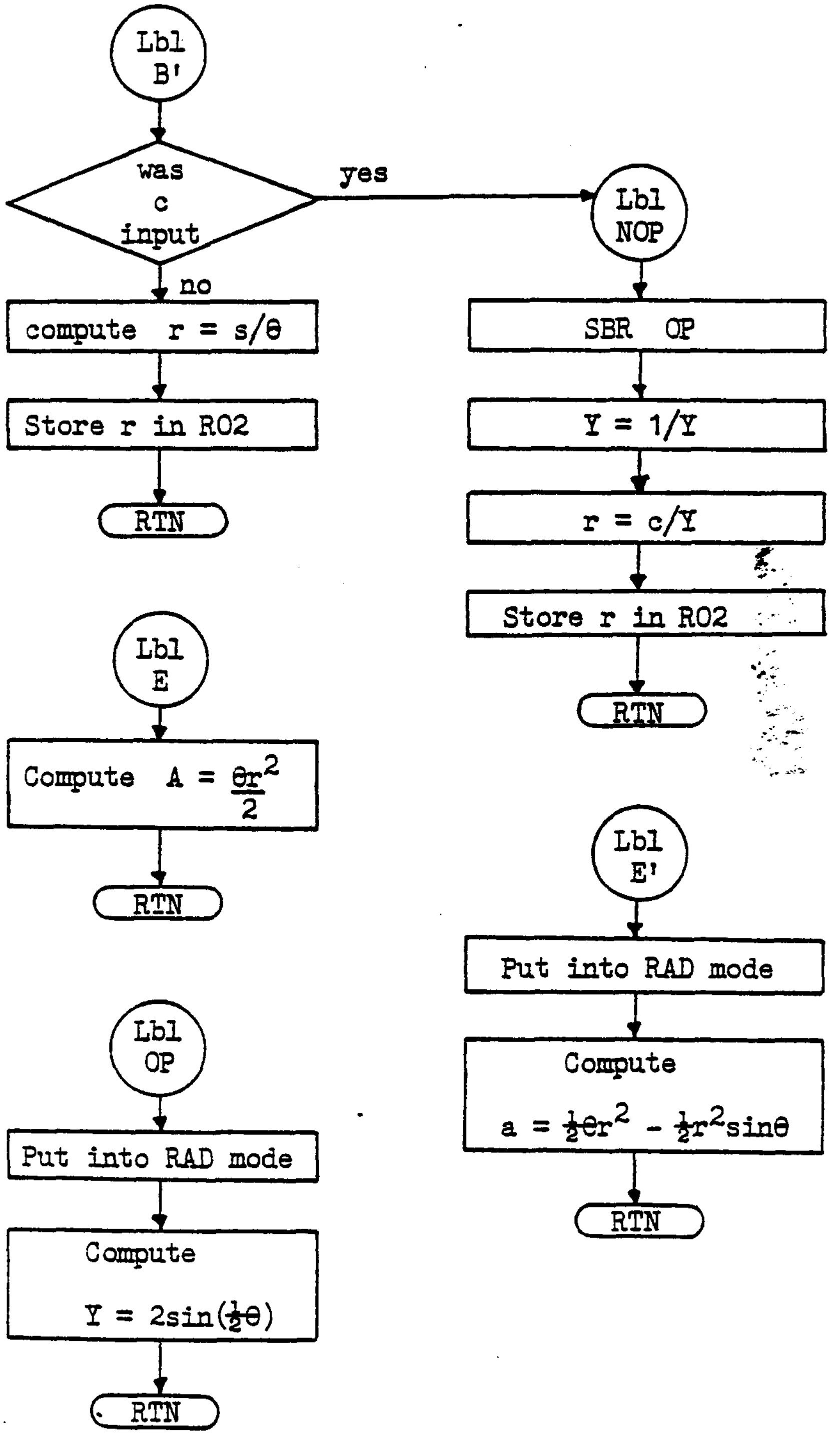
Special notes:

(1) Contrary to M.L.M., in user instructions only A' or B' must be executed first (if not input). The order of C', D', E', and E is immaterial.
(2) Steps 168-175 are essentially a repeat of label E and could be replaced with E, saving a net 7 steps.

Special applications:

(1) PGM 13 C' evaluates (R01)x(R02).
(2) PGM 13 SBR 012 will recall the contents of R01 and if the display was less than zero, will flash said contents. (T register must be zero)
(3) PGM 13 SBR 031 will return with the same value input if the input is greater than or equal to zero, or will flash the contents of R02 if the input is less than zero. (T register must be zero)
(4) PGM 13 SBR 028 will return with the value input if the input is greater than or equal to zero, or flash the value input if it is negative. (note that R02 is used)
PGM 13 B will perform the same function in addition to reseting flags 0 and 1.
(5) PGM 13 E will evaluate $\frac{1}{2}(R01)(R02)^2$ a form which appears quite often in formulas involving the physical sciences.


Lbl A
Reset flags 0 & 1
Clear T register
Store θ in R01
Compute y = sinθ
is y GE 0 ?
yes
no
Create error state
Lbl EXC
Recall θ from R01
RTN
Lbl D
Store c in R04
Set flags 0 and 1
RTN
Lbl B
Reset flags 0 & 1
Clear T register
Store r in R02
is r GE 0 ?
yes
no
Create error state
Recall r
Lbl DMS
RTN
Lbl A'
was c input
yes
no
Compute θ = s/r
Store θ in R01
RTN
Lbl C
Store s in R03
RTN
Lbl C'
s = rθ
RTN
Lbl D'
SBR OP
c = rY
RTN
Lbl ENG
Put into radian mode.
Compute θ = 2sin⁻¹(c/2r)
Store θ in R01
RTN


Lbl B'
was c input
yes
no
compute r = s/θ
Store r in R02
RTN
Lbl NOP
SBR OP
Y = 1/Y
r = c/Y
Store r in R02
RTN
Lbl E
Compute A = θr²/2
RTN
Lbl E'
Put into RAD mode
Compute a = ½θr² - ½r²sinθ
RTN
Lbl OP
Put into RAD mode
Compute Y = 2sin(½θ)
RTN

ML-13 Program Listing

```
000  76  LBL
001  11   A
002  22  INV
003  86  STF
004  00   00
005  22  INV
006  86  STF
007  01   01
008  29  CP
009  42  STO
010  01   01
011  38  SIN
012  77   GE
013  48  EXC
014  23  LNX
015  76  LBL
016  48  EXC
017  43  RCL
018  01   01
019  92  RTN
020  76  LBL
021  12   B
022  22  INV
023  86  STF
024  00   00
025  22  INV
026  86  STF
027  01   01
028  29  CP
029  42  STO
030  02   02
031  77   GE
032  88  DMS
033  23  LNX
034  43  RCL
035  02   02
036  76  LBL
037  88  DMS
038  92  RTN
039  76  LBL
040  13   C
041  42  STO
042  03   03
043  92  RTN
044  76  LBL
045  14   D
046  42  STO
047  04   04
048  86  STF
049  00   00
050  86  STF
051  01   01
052  92  RTN
053  76  LBL
054  16  A'
055  87  IFF
056  00   00
057  57  ENG
058  53   (
059  43  RCL
060  03   03
061  55   ÷
062  43  RCL
063  02   02
064  54   )
065  42  STO
066  01   01
067  92  RTN
068  76  LBL
069  57  ENG
070  70  RAD
071  53   (
072  53   (
073  43  RCL
074  04   04
075  55   ÷
076  02   2
077  55   ÷
078  43  RCL
079  02   02
080  54   )
081  22  INV
082  38  SIN
083  65   ×
084  02   2
085  54   )
086  42  STO
087  01   01
088  92  RTN
089  76  LBL
090  17  B'
091  87  IFF
092  01   01
093  68  NOP
094  53   (
095  43  RCL
096  03   03
097  55   ÷
098  43  RCL
099  01   01
100  54   )
101  42  STO
102  02   02
103  92  RTN
104  76  LBL
105  68  NOP
106  53   (
107  71  SBR
108  69  OP
109  35  1/X
110  65   ×
111  43  RCL
112  04   04
113  54   )
114  42  STO
115  02   02
116  92  RTN
117  76  LBL
118  69  OP
119  70  RAD
120  53   (
121  53   (
122  43  RCL
123  01   01
124  55   ÷
125  02   2
126  54   )
127  38  SIN
128  65   ×
129  02   2
130  54   )
131  92  RTN
132  76  LBL
133  18  C'
134  53   (
135  43  RCL
136  01   01
137  65   ×
138  43  RCL
139  02   02
140  54   )
141  92  RTN
142  76  LBL
143  19  D'
144  71  SBR
145  69  OP
146  65   ×
147  43  RCL
148  02   02
149  54   )
150  92  RTN
151  76  LBL
152  15   E
153  53   (
154  43  RCL
155  02   02
156  33  X²
157  65   ×
158  43  RCL
159  01   01
160  55   ÷
161  02   2
162  54   )
163  92  RTN
164  76  LBL
165  10  E'
166  70  RAD
167  53   (
168  43  RCL
169  01   01
170  65   ×
171  43  RCL
172  02   02
173  33  X²
174  55   ÷
175  02   2
176  75   -
177  43  RCL
178  02   02
179  33  X²
180  55   ÷
181  02   2
182  65   ×
183  43  RCL
184  01   01
185  38  SIN
186  54   )
187  92  RTN
```

ML-14

NORMAL DISTRIBUTION

ML-14 is a rather straightforward execution of the formulas presented in the Master Library Manual. Note however, that the error term $\epsilon(x)$ is neglected in calculating Q(x). Since in most applications the magnitude of x is less than 5, the error should not be significant.

Register assignments are:

	R01	R02	R03
INITIAL	Z(x)	---	x
FINAL*	Z(x)	t	1/t

*After computing Q(x).

Normal use data:

Flags used: flag 1
Parentheses levels: 2
Subroutine levels: 0

Interface procedure:

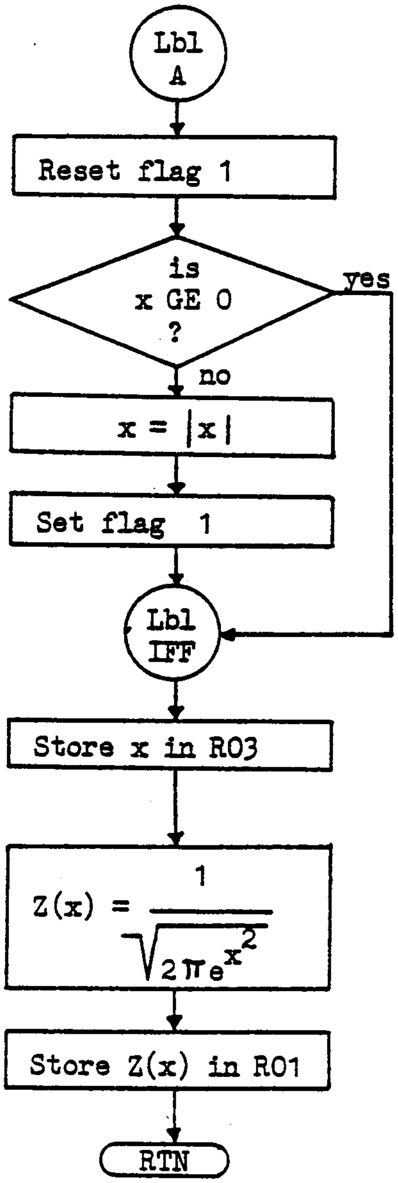
Simply follow the user instructions...the program is too simple for any fancy shortcuts.

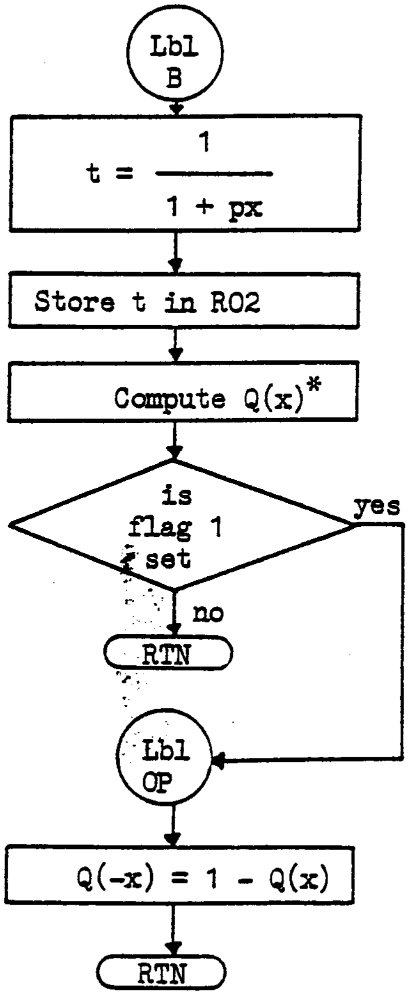
Special applications:

In first order systems analysis, the system response to a unit step input frequently has the form:

$$e^{-t/T} \quad \text{or} \quad 1 - e^{-t/T}$$

where t is some input time and T is the time constant of the system. PGM 14 SBR 131 with the numerical value of the first form as an input will return with the value of the second form if flag 1 is set, or the same input if flag 1 is not set.





*For Q(x):

$$Q(x) = (t^4b_5 - t^3b_4 + t^2b_3 - tb_2 + b_1)(t)(Z(x))$$

b_5 = 1.330274429

b_4 = 1.821255978

b_3 = 1.781477937

b_2 = .356563782

b_1 = .319381530

ML-14 Program Listing

```
000  76  LBL
001  11   A
002  22  INV
003  86  STF
004  01   01
005  29  CP
006  77   GE
007  87  IFF
008  94  +/-
009  86  STF
010  01   01
011  76  LBL
012  87  IFF
013  53   (
014  42  STO
015  03   03
016  33  X²
017  22  INV
018  23  LNX
019  65   ×
020  02   2
021  65   ×
022  89   π
023  54   )
024  34  √X
025  35  1/X
026  42  STO
027  01   01
028  92  RTN
029  76  LBL
030  12   B
031  93   .
032  02   2
033  03   3
034  01   1
035  06   6
036  04   4
037  01   1
038  09   9
039  49  PRD
040  03   03
041  01   1
042  44  SUM
043  03   03
044  43  RCL
045  03   03
046  35  1/X
047  53   (
048  53   (
049  42  STO
050  02   02
051  45  YX
052  04   4
053  65   ×
054  01   1
055  93   .
056  03   3
057  03   3
058  00   0
059  02   2
060  07   7
061  04   4
062  04   4
063  02   2
064  09   9
065  75   -
066  43  RCL
067  02   02
068  45  YX
069  03   3
070  65   ×
071  01   1
072  93   .
073  08   8
074  02   2
075  01   1
076  02   2
077  05   5
078  05   5
079  09   9
080  07   7
081  08   8
082  85   +
083  43  RCL
084  02   02
085  45  YX
086  02   2
087  65   ×
088  01   1
089  93   .
090  07   7
091  08   8
092  01   1
093  04   4
094  07   7
095  07   7
096  09   9
097  03   3
098  07   7
099  75   -
100  43  RCL
101  02   02
102  65   ×
103  93   .
104  03   3
105  05   5
106  06   6
107  05   5
108  06   6
109  03   3
110  07   7
111  08   8
112  02   2
113  85   +
114  93   .
115  03   3
116  01   1
117  09   9
118  03   3
119  08   8
120  01   1
121  05   5
122  03   3
123  54   )
124  65   ×
125  43  RCL
126  02   02
127  65   ×
128  43  RCL
129  01   01
130  54   )
131  87  IFF
132  01   01
133  69  OP
134  92  RTN
135  76  LBL
136  69  OP
137  53   (
138  94  +/-
139  85   +
140  01   1
141  54   )
142  92  RTN

001  11   A
012  87  IFF
030  12   B
136  69  OP
```

ML-15

RANDOM NUMBER GENERATOR

ML-15 can be analyzed as three separate routines; one to calculate uniformly distributed numbers on the range 0 to 1, one for uniform distribution on a user defined range, and one for normal distribution with standard deviation and mean as user inputs.

UNIFORM DISTRIBUTION: (0-1)

Normal use data:

Flags affected: none
Parentheses levels: 3
Subroutine levels: none

Register assignments are:

	R01-R06	R07	R09
INITIAL	—	—	x_n/seed
FINAL	—	199017	x_{n+1}

Note: x_n or x_{n+1} are the new seeds, not the random number.

Interface procedure:

(1) Store a seed in R09. ($0 \leq$ seed ≤ 199017)
(2) To generate numbers use PGM 15 SBR DMS each time a number is needed...returns with random number in display.

UNIFORM DISTRIBUTION: ($A \leq N \leq B$)

Register assignments are:

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10	R11
INITIAL	0	0	0	0	0	0	—	—	x_n	A	B
FINAL	ΣY	ΣY^2	n	ΣX	ΣX^2	ΣXY	N	—	x_{n+1}	A	B

Normal use data:

Flags affected: none
Parentheses levels: 3
Subroutine levels: 1

Interface procedure:

If the statistical functions are needed then:

(1) Prestore seed in R09, A in R10 and B in R11.
(2) Execute PGM 01 SBR CLR to initialize R01-R06. (or PGM 15 E')
(3) Execute PGM 15 C for each number to be generated. Follow user instructions for statistical information.

If the statistical functions are not needed, a considerable savings in registers (and hence program steps) can be realized.

(1) Prestore seed in R09, A in R08, and B in R10.
(2) Execute (RCL 08 - (CE - RCL 10) x PGM 15 SBR DMS) for each number to be generated. Note that this uses 16 program steps and four consecutive registers for 16+(4)(8)=48 equivalent program steps. By contrast, the sequence PGM 15 C uses 3 program steps and 11 registers for an equivalent 91 program steps.
(3) If A and B are not variable, step (2) can be reduced to (PGM 15 SBR DMS x Q + P) where Q=A-B and P=A. This only uses 2 registers and 10 program steps for an equivalent 26 program steps.

A= UPPER LIMIT
B= LOWER LIMIT

Special applications:

(1) PGM 15 SBR 048 evaluates (INT(input)/100,000).
(2) PGM 15 SBR 043 truncates the displayed number at 5 decimal places.
(3) PGM 15 SBR 026 displays the fractional part of an input, truncated to 5 decimal places.

NORMAL DISTRIBUTION:

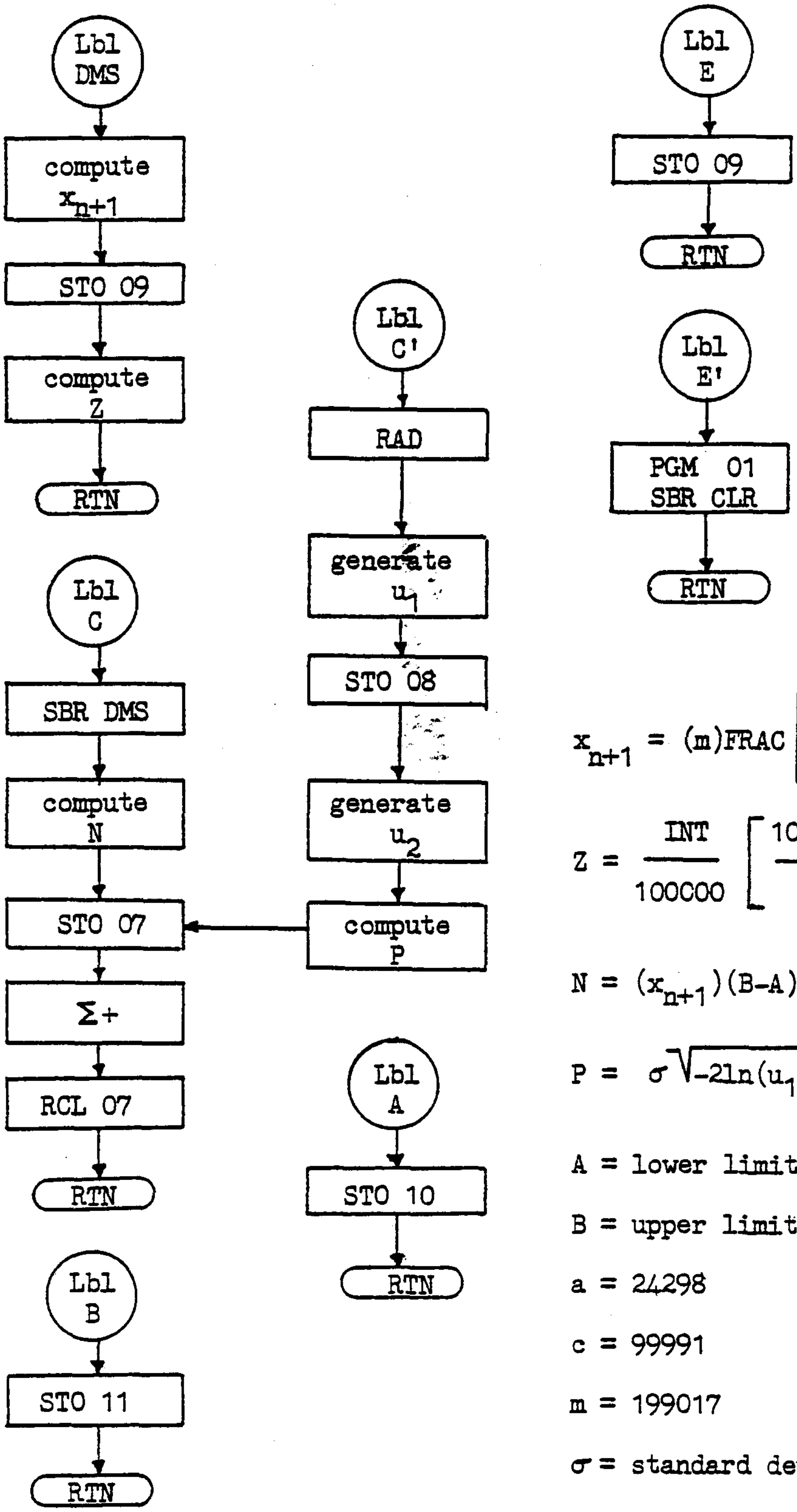
This case is entirely similar to the uniform distribution case. Replace references to A with $\bar{x}$ and tc B with σ. For the statistical version execute PGM 15 C' for each number to be generated. For the shortened version use either:

#1 ((RAD PGM 15 SBR DMS lnx x 2 +/-)$\sqrt{x}$ x RCL 10 x (2 x π x PGM 15 SBR DMS) cos + RCL 08)

with σ in R10 and $\bar{x}$ in R08.

#2 If σ and $\bar{x}$ are fixed, in #1 replace RCL 10 with σ and RCL 08 with $\bar{x}$ to save two more registers.

Note that the "shorter versions" are only shorter if the freed registers can be used to store other data.



$$x_{n+1} = (m)\text{FRAC}\left[\frac{a(x_n) + c}{m}\right]$$

$$Z = \frac{\text{INT}}{100000}\left[\frac{100000(x_{n+1})}{m}\right]$$

$$N = (x_{n+1})(B-A) + A$$

$$P = \sigma\sqrt{-2\ln(u_1)}\ \cos(2\pi u_2) + \bar{x}$$

A = lower limit

B = upper limit

a = 24298

c = 99991

m = 199017

σ = standard deviation

$\bar{x}$ = mean

u_1 & u_2: normally distributed random numbers.

ML-15 Program Listing

```
000  76  LBL
001  88  DMS
002  53  (
003  53  (
004  02  2
005  04  4
006  02  2
007  09  9
008  08  8
009  65  ×
010  43  RCL
011  09   09
012  85  +
013  09  9
014  09  9
015  09  9
016  09  9
017  01  1
018  54  )
019  55  ÷
020  01  1
021  09  9
022  09  9
023  00  0
024  01  1
025  07  7
026  42  STO
027  07   07
028  54  )
029  53  (
030  53  (
031  53  (
032  22  INV
033  59  INT
034  65  ×
035  43  RCL
036  07   07
037  54  )
038  42  STO
039  09   09
040  55  ÷
041  43  RCL
042  07   07
043  65  ×
044  05  5
045  22  INV
046  28  LOG
047  54  )
048  59  INT
049  55  ÷
```

```
050  05  5
051  22  INV
052  28  LOG
053  54  )
054  92  RTN
055  76  LBL
056  13  C
057  71  SBR
058  88  DMS
059  53  (
060  24  CE
061  65  ×
062  53  (
063  43  RCL
064  11   11
065  75  -
066  43  RCL
067  10   10
068  54  )
069  76  LBL
070  37  P/R
071  85  +
072  43  RCL
073  10   10
074  54  )
075  42  STO
076  07   07
077  78  Σ+
078  43  RCL
079  07   07
080  92  RTN
081  76  LBL
082  18  C'
083  70  RAD
084  71  SBR
085  88  DMS
086  42  STO
087  08   08
088  71  SBR
089  88  DMS
090  53  (
091  53  (
092  24  CE
093  65  ×
094  02  2
095  65  ×
096  89  π
097  54  )
098  39  COS
099  65  ×
```

```
100  53  (
101  43  RCL
102  08   08
103  23  LNX
104  65  ×
105  02  2
106  94  +/-
107  54  )
108  34  √X
109  65  ×
110  43  RCL
111  11   11
112  61  GTO
113  37  P/R
114  76  LBL
115  10  E'
116  36  PGM
117  01   01
118  71  SBR
119  25  CLR
120  92  RTN
121  76  LBL
122  15  E
123  42  STO
124  09   09
125  92  RTN
126  76  LBL
127  11  A
128  42  STO
129  10   10
130  92  RTN
131  76  LBL
132  12  B
133  42  STO
134  11   11
135  92  RTN
```

```
001  88  DMS
056  13  C
070  37  P/R
082  18  C'
115  10  E'
122  15  E
127  11  A
132  12  B
```

ML-16

COMBINATIONS, PERMUTATIONS AND FACTORIALS

ML-16 computes combinations, permutations and factorials. Consult the Master Library Manual for detailed explanation and applicable formulas.

FACTORIALS:

Register assignments are:

	R01	R02	R03	R04
INITIAL	|INT(n)|	---	1	1
FINAL	0	---	1	n!

Interface procedure:

With a known valid integer input, a gain of about one second in reduced execution time can be made at the expense of length by prestoring n in R01 and 1 in R04 then executing PGM 16 C. This eliminates execution of label A. (R02 and R03 are not used)

Normal use data:

Flags used: flag 1
Parentheses levels: none
Subroutine levels none

Special notes:

(1) With the interface procedure given, the display must be non-zero when PGM 16 C is executed, but does not have to contain n.
(2) If Q is prestored in R04 then the above interface gives the output (Q)(n!).

PERMUTATIONS:

Register assignments are:

	R01	R02	R03	R04
INITIAL	|INT(n)|	|INT(r)|	2	1
FINAL	s*	0	2	PER.

*s = |INT(n)| - |INT(r)|

PERMUTATIONS (CONT.)

Interface procedure:

With known valid integer inputs of n and r, execution time can be cut approximately in half by prestoring n in R01, r in R02, and 1 in R04, then executing PGM 16 D ...returns with value of permutations in display and R04. Note that R03 is not used for this.

Normal use data:

Flags used: flag 1
Parentheses levels: none
Subroutine levels: 1

Special notes:

See notes for factorials.

COMBINATIONS:

Register assignments are:

	R01	R02	R03	R04
INITIAL	$\|INT(n)\|$	$\|INT(r)\|$	2	1
FINAL	s*	0	2	COM.

$*s = |INT(n)| - |INT(r)|$

Interface procedure:

With known valid integer inputs of n and r, execution time can be cut approximately in half by prestoring n in R01, r in R02 and 1 in R04 then executing PGM 16 E ...returns with number of combinations in display and R04. Note that R03 is not used for this.

Normal use data:

Flags affected: flag 1
Parentheses levels: none
Subroutine levels: 1

Special notes:

See notes for factorials.

Special applications:

(1) PGM 16 SBR 119 will recall contents of R02 if the input in display is less than or equal to a prestored limit in R01, otherwise will give an error indication

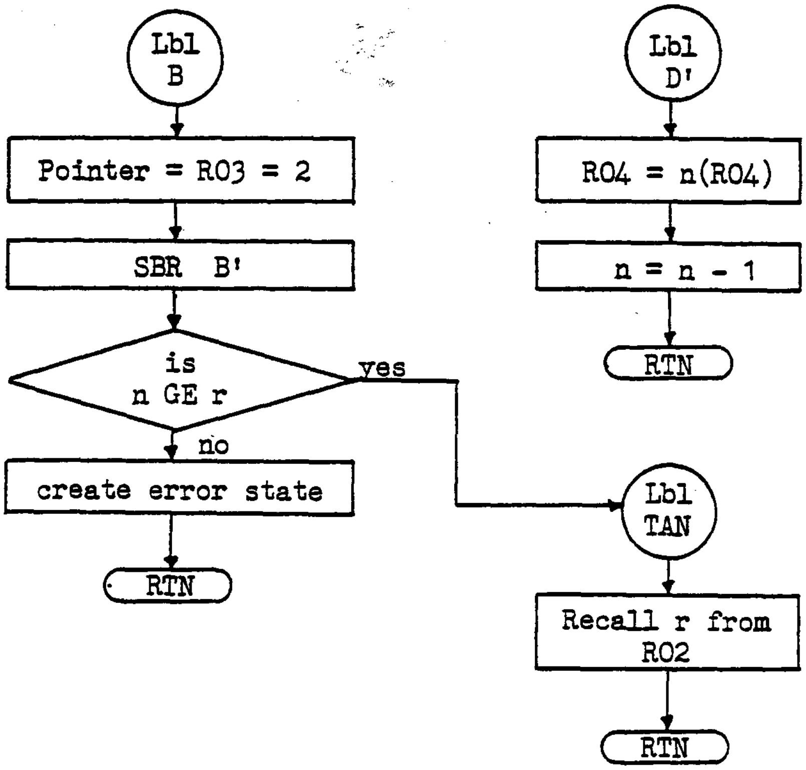
(2) PGM 16 SBR SIN will recall the contents of R04 and if flag 1 is set, flash the value.

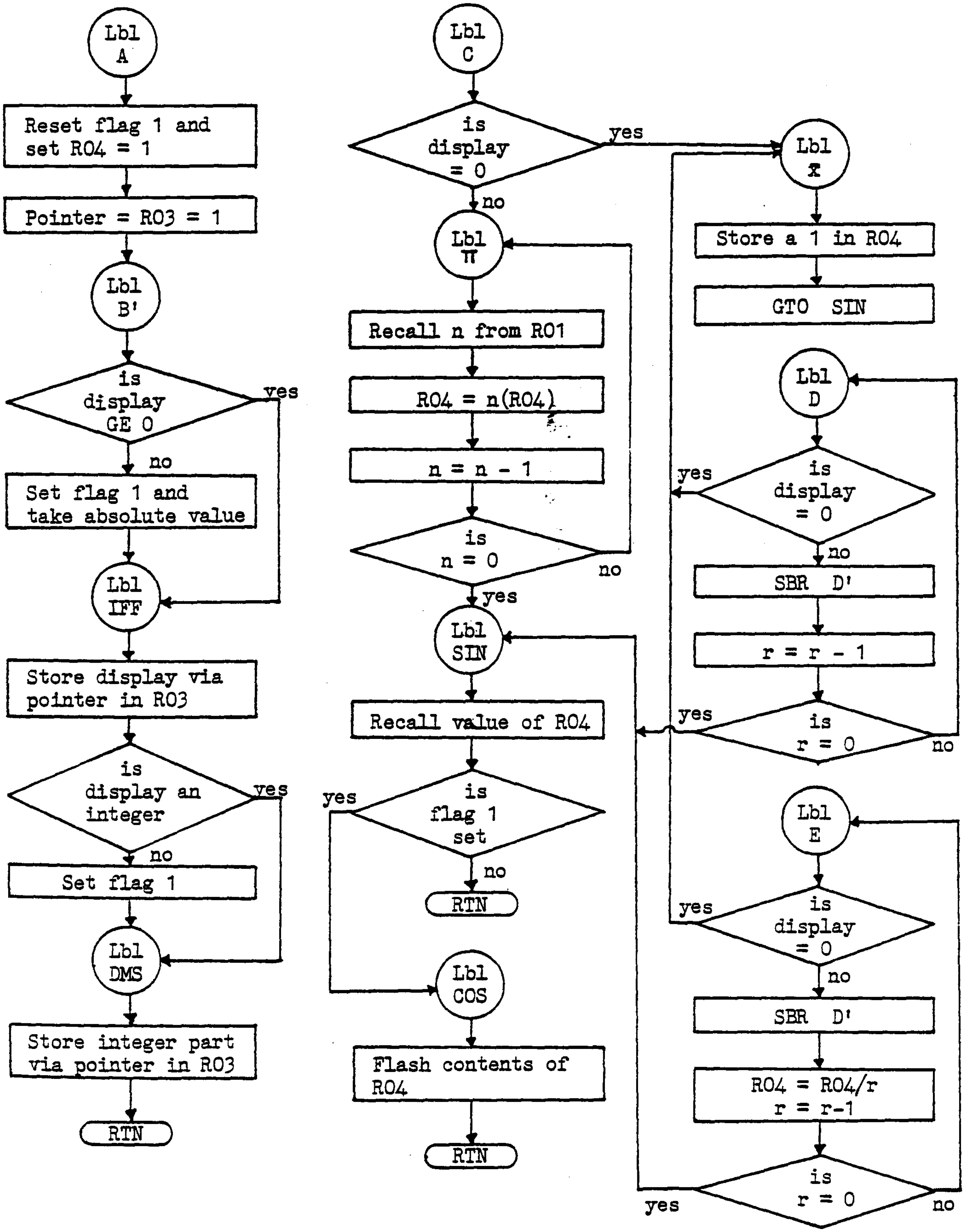
(3) PGM 16 SBR $\bar{x}$ will display a 1 if flag 1 is not set or a flashing 1 if it is set. (uses R04)

(4) PGM 16 SBR 018 will return with the input value in display if flag 1 is not set, or with the flashing contents of R04 if flag 1 is set.

(5) PGM 16 B' will check an input to see if it is an integer value greater than or equal to zero. In any case, the absolute value of the integer portion is stored in the register whose address is in R03. If the input was not an integer or was less than zero, flag 1 is set.

(6) PGM 16 D' evaluates R04 = (R04)(R01), & R01 = R01-1




Lbl A
Reset flag 1 and set R04 = 1
Pointer = R03 = 1
Lbl B'
is display GE 0
yes
no
Set flag 1 and take absolute value
Lbl IFF
Store display via pointer in R03
is display an integer
yes
no
Set flag 1
Lbl DMS
Store integer part via pointer in R03
RTN
Lbl C
is display = 0
yes
no
Lbl π
Recall n from R01
R04 = n(R04)
n = n - 1
is n = 0
no
yes
Lbl SIN
Recall value of R04
is flag 1 set
yes
no
RTN
Lbl COS
Flash contents of R04
RTN
Lbl x̄
Store a 1 in R04
GTO SIN
Lbl D
is display = 0
yes
no
SBR D'
r = r - 1
is r = 0
yes
no
Lbl E
is display = 0
yes
no
SBR D'
R04 = R04/r
r = r-1
is r = 0
yes
no

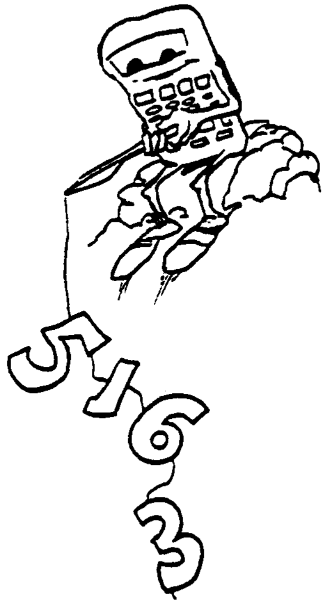
ML-16 Program Listing

```
000  76  LBL
001  13  C
002  29  CP
003  67   EQ
004  79   x̄
005  76  LBL
006  89   π
007  43  RCL
008  01   01
009  49  PRD
010  04   04
011  97  DSZ
012  01   01
013  89   π
014  76  LBL
015  38  SIN
016  43  RCL
017  04   04
018  87  IFF
019  01   01
020  39  COS
021  92  RTN
022  76  LBL
023  14   D
024  29  CP
025  67   EQ
026  79   x̄
027  19  D'
028  97  DSZ
029  02   02
030  14   D
031  61  GTO
032  38  SIN
033  76  LBL
034  15   E
035  29  CP
036  67   EQ
037  79   x̄
038  19  D'
039  43  RCL
040  02   02
041  22  INV
042  49  PRD
043  04   04
044  97  DSZ
045  02   02
046  15   E
047  61  GTO
048  38  SIN
049  76  LBL
```

```
050  79   x̄
051  01   1
052  42  STO
053  04   04
054  61  GTO
055  38  SIN
056  76  LBL
057  11   A
058  32  X:T
059  22  INV
060  86  STF
061  01   01
062  01   1
063  42  STO
064  04   04
065  42  STO
066  03   03
067  32  X:T
068  76  LBL
069  17  B'
070  29  CP
071  77   GE
072  87  IFF
073  86  STF
074  01   01
075  50  I×I
076  76  LBL
077  87  IFF
078  72  ST*
079  03   03
080  32  X:T
081  73  RC*
082  03   03
083  59  INT
084  67   EQ
085  88  DMS
086  86  STF
087  01   01
088  76  LBL
089  88  DMS
090  72  ST*
091  03   03
092  92  RTN
093  76  LBL
094  39  COS
095  00   0
096  35  1/X
097  43  RCL
098  04   04
099  92  RTN
```

```
100  76  LBL
101  19  D'
102  43  RCL
103  01   01
104  49  PRD
105  04   04
106  01   1
107  22  INV
108  44  SUM
109  01   01
110  92  RTN
111  76  LBL
112  12   B
113  32  X:T
114  02   2
115  42  STO
116  03   03
117  32  X:T
118  17  B'
119  32  X:T
120  43  RCL
121  01   01
122  77   GE
123  30  TAN
124  00   0
125  35  1/X
126  92  RTN
127  76  LBL
128  30  TAN
129  43  RCL
130  02   02
131  92  RTN
```

```
001  13   C
006  89   π
015  38  SIN
023  14   D
034  15   E
050  79   x̄
057  11   A
069  17  B'
077  87  IFF
089  88  DMS
094  39  COS
101  19  D'
112  12   B
128  30  TAN
```



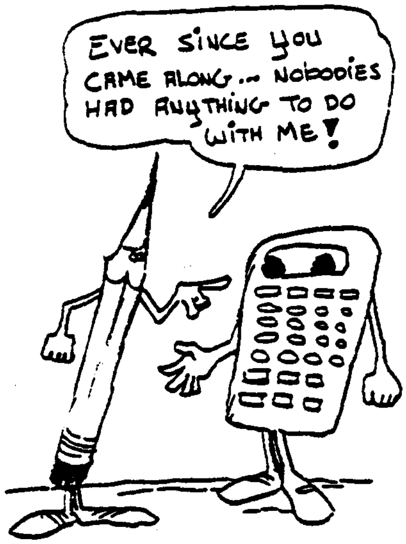
ML-17

MOVING AVERAGES

ML-17 calculates moving averages. See the Master Library Manual for a discussion of moving averages and the applicable formulas. Refer to the block diagram during the following discussion of program operation:

Inputs are stored in n registers starting with R06. On the first pass through the data registers, the path labeled IFF is followed for each input (flag 1 is not set yet).

At the n+1 input, the pointer is reset to 6 and flag 1 is set. The path labeled π is then executed for this and subsequent inputs. Notice that the location of the latest input moves down through the array with each pass.


EVER SINCE YOU CAME ALONG... NOBODIES HAD ANYTHING TO DO WITH ME!

Register assignments are:

R01: pointer for storing current input
R02: n
R03: number of inputs up to n+1
R04: sum of all inputs
R05: current input
R06-R(n+5): last n inputs

Interface procedures:

(1) With a known valid input for n (integer greater than zero), prestore n in R02.
(2) Execute PGM 17 E' to initialize R01, R03, and R04 (also resets flag 1)
(3) Execute PGM 17 B for each input...returns with average in display.

Normal use data:

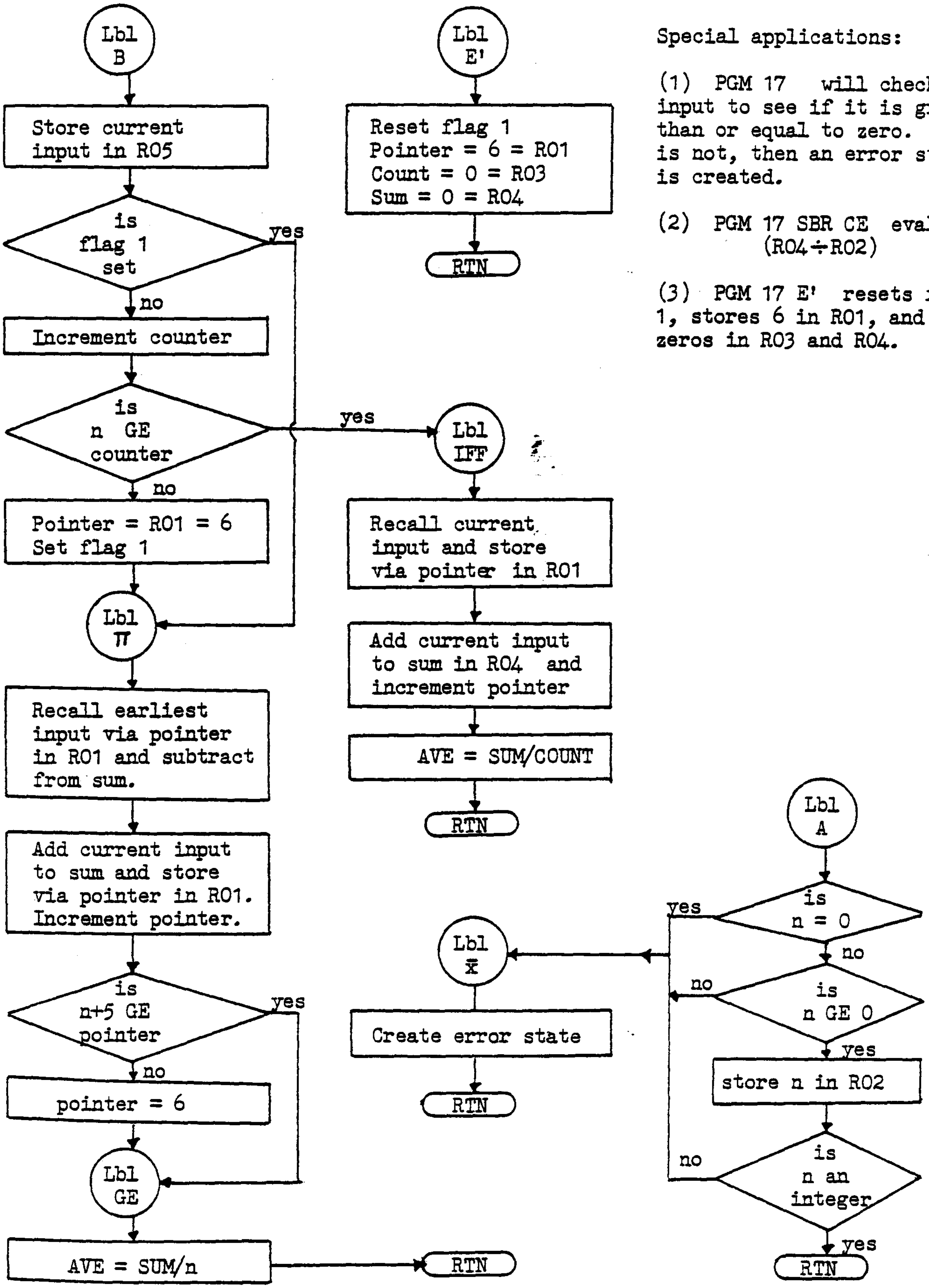
Flags used: flag 1
Parentheses levels: 1
Subroutine levels: 0

Special applications:

(1) PGM 17 will check an input to see if it is greater than or equal to zero. If it is not, then an error state is created.

(2) PGM 17 SBR CE evaluates (R04÷R02)

(3) PGM 17 E' resets flag 1, stores 6 in R01, and puts zeros in R03 and R04.



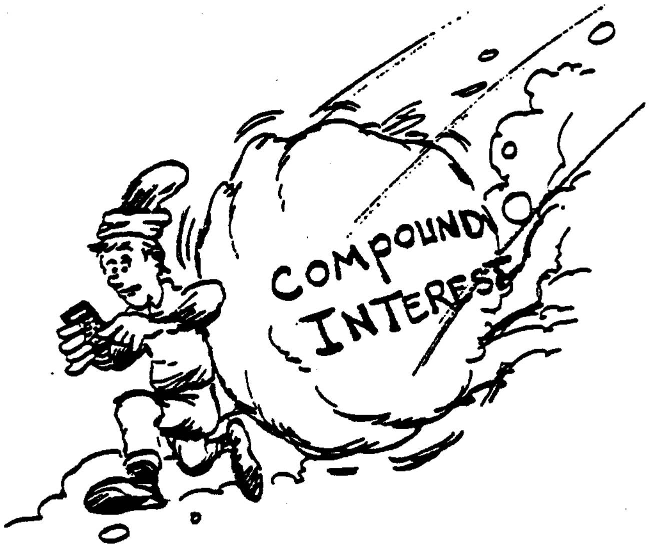
ML-17 Program Listing

```
000   76  LBL
001   87  IFF
002   43  RCL
003   05   05
004   72  ST*
005   01   01
006   44  SUM
007   04   04
008   01   1
009   44  SUM
010   01   01
011   53   (
012   43  RCL
013   04   04
014   55   ÷
015   32  X:T
016   54   )
017   92  RTN
018   76  LBL
019   12   B
020   42  STO
021   05   05
022   87  IFF
023   01   01
024   89   π
025   01   1
026   44  SUM
027   03   03
028   43  RCL
029   03   03
030   32  X:T
031   43  RCL
032   02   02
033   77   GE
034   87  IFF
035   06   6
036   42  STO
037   01   01
038   86  STF
039   01   01
040   76  LBL
041   89   π
042   73  RC*
043   01   01
044   22  INV
045   44  SUM
046   04   04
047   43  RCL
048   05   05
049   44  SUM
050   04   04
051   72  ST*
052   01   01
053   01   1
054   44  SUM
055   01   01
056   43  RCL
057   01   01
058   32  X:T
059   53   (
060   43  RCL
061   02   02
062   85   +
063   05   5
064   54   )
065   77   GE
066   77   GE
067   06   6
068   42  STO
069   01   01
070   76  LBL
071   77   GE
072   53   (
073   43  RCL
074   04   04
075   55   ÷
076   43  RCL
077   02   02
078   54   )
079   92  RTN
080   76  LBL
081   79   x̄
082   00   0
083   35  1/X
084   92  RTN
085   76  LBL
086   10   E'
087   22  INV
088   86  STF
089   01   01
090   06   6
091   42  STO
092   01   01
093   00   0
094   42  STO
095   03   03
096   42  STO
097   04   04
098   92  RTN
099   76  LBL
100   11   A
101   29  CP
102   67   EQ
103   79   x̄
104   22  INV
105   77   GE
106   79   x̄
107   42  STO
108   02   02
109   32  X:T
110   43  RCL
111   02   02
112   59  INT
113   22  INV
114   67   EQ
115   79   x̄
116   92  RTN

001   87  IFF
019   12   B
041   89   π
071   77   GE
081   79   x̄
086   10  E'
100   11   A
```

ML-18

COMPOUND INTEREST

Despite a couple of minor faults, ML-18 is probably one of the most well written and documented programs in the master library. It's good points are:



(1) Inputs are not ordered and may be entered in any sequence.
(2) The program is restartable; that is, only those inputs which change need to be reentered to run the program again. Present data is not affected by program execution.
(3) The same user defined key is used for inputs and outputs.

As for faults:

(1) Note that nothing is gained by making (RCL 04 ÷ RCL 03) into a subroutine which is only called twice by the program. To use as a subroutine requires seven "control instructions"; SBR SBR, SBR SBR, Lbl SBR, RTN. Adding this to the seven steps to evaluate the function itself gives a total of 14 steps, which is exactly what would be required to compute it directly each time it is needed.
(2) There are "gaps" in the register sequence used. In general it is a good idea to keep register assignments in sequence.

Interface procedures:

Prestore the appropriate data for the unknown quantity according to the register assignment table. Execute PGM 18 followed by the user defined key for that unknown. Note that R02, R08, and R09 are easiest filled by PGM 18 E with %I in the display, although sometimes not all three quantities are needed for a particular unknown.

Special notes:

(1) %I = percent interest

(2) i = interest in decimal form (%I/100)

Register assignments are:

UNKNOWN		R01	R02	R03	R04	R08	R09	R12	T reg	( ) level	SBR level
N	INITIAL	---	I	PV	FV	i*	1+i	---	0	2	1
	FINAL	N	I	PV	FV	i	1+i	---	0		
I	INITIAL	N	---	PV	FV	---	---	---	0	2	1
	FINAL	N	I	PV	FV	i	1+i	---	0		
PV	INITIAL	N	I*	---	FV	i*	1+i	---	0	1	0
	FINAL	N	I	PV	FV	i	1+i	---	0		
FV	INITIAL	N	I*	PV	---	i*	1+i	---	0	1	0
	FINAL	N	I	PV	FV	i	1+i	---	0		
$s_{\overline{n}\vert i}$	INITIAL	N	---	---	---	i	1+i	---	---	2	0
	FINAL	N	---	---	---	i	1+i	$(1+i)^N$	---		
$(1+i)s_{\overline{n}\vert i}$	INITIAL	N	---	---	---	i	1+i	---	---	3	1
	FINAL	N	---	---	---	i	1+i	$(1+i)^N$	---		
$a_{\overline{n}\vert i}$	INITIAL	N	---	---	---	i	1+i	---	---	3	1
	FINAL	N	---	---	---	i	1+i	$(1+i)^N$	---		
$(1+i)a_{\overline{n}\vert i}$	INITIAL	N	---	---	---	i	1+i	---	---	3	2
	FINAL	N	---	---	---	i	1+i	$(1+i)^N$	---		

* Inputs which are stored or calculated while using ML-18 according to the user instructions but are not needed to find the particular unknown and may be omitted while using the interface procedure.

--- Not used or doesn't matter.

Special applications:

(1) PGM 18 A' evaluates $\frac{(R09)^{R01} - 1}{R08}$ and stores $(R09)^{R01}$ in R12.

(2) PGM 18 B' evaluates $\left[\frac{R09}{R08}\right]\left[(R09)^{R01} - 1\right]$ and stores $(R09)^{R01}$ in R12.

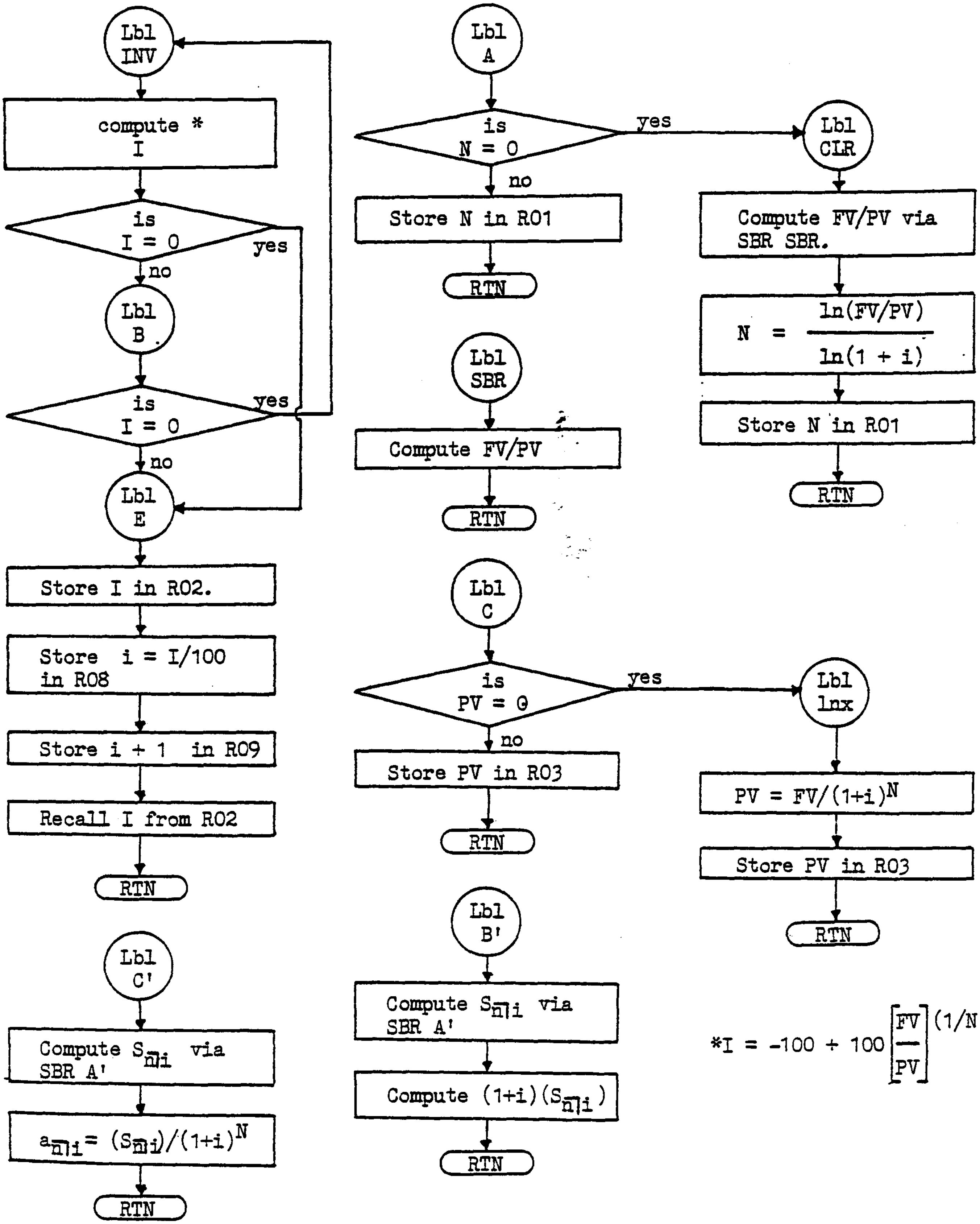
(3) PGM 18 C' evaluates $\frac{1 - (R09)^{-(R01)}}{R08}$ and stores $(R09)^{R01}$ in R12.

(4) PGM 18 D' evaluates $\left[\frac{R09}{R08}\right]\left[1 - (R09)^{-(R01)}\right]$ and stores $(R09)^{R01}$ in R12.

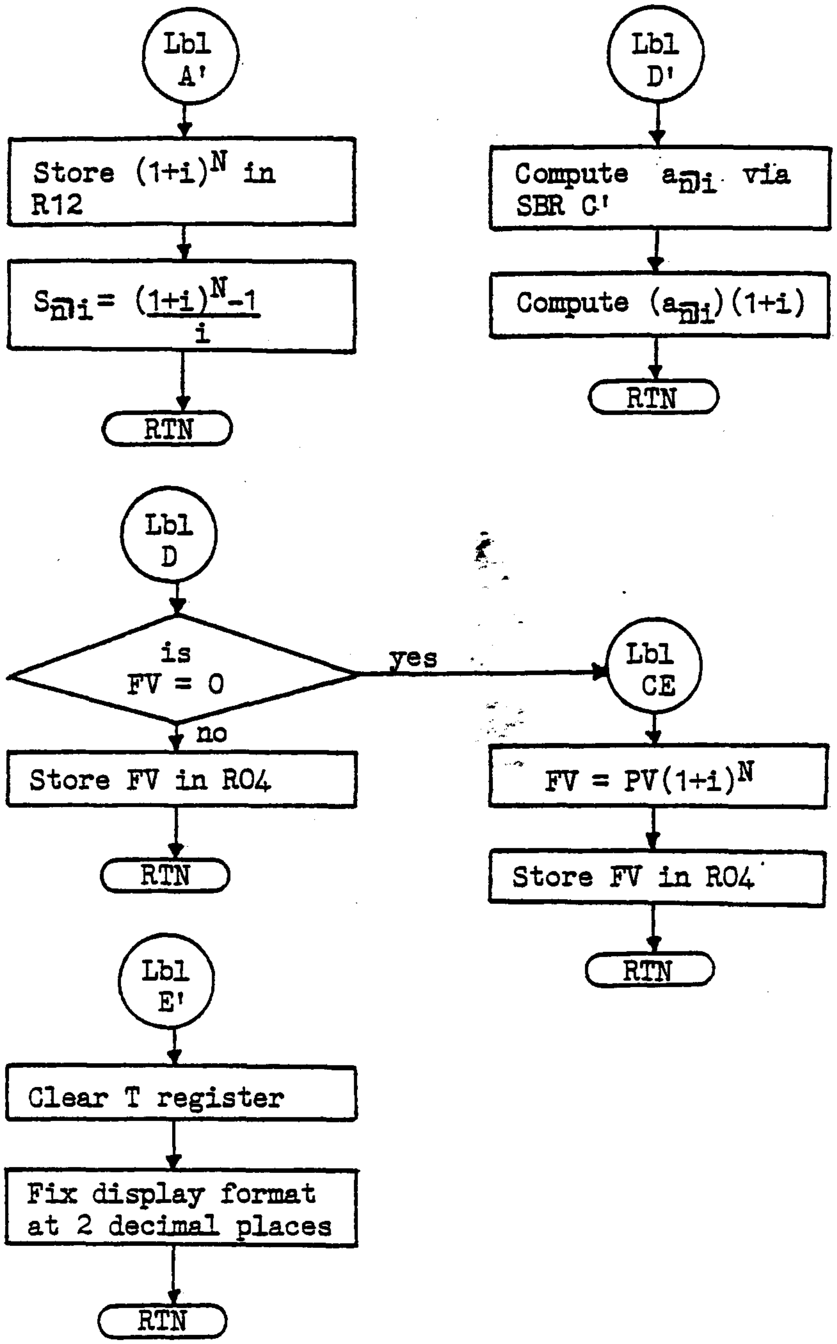
(5) PGM 18 SBR SBR evaluates (RCL 04 ÷ RCL 03) and returns with result in display without affecting pending operations.

(6) PGM 18 SBR lnx evaluates $\frac{R04}{(R09)^{R01}}$ and stores it in R03.

(7) PGM 18 SBR CE evaluates $(R03)(R09)^{R01}$ and stores it in R04.



$$*I = -100 + 100\left[\frac{FV}{PV}\right]^{(1/N)}$$


Lbl A'
Store $(1+i)^N$ in R12
$S_{\overline{n}|i} = \frac{(1+i)^N-1}{i}$
RTN
Lbl D'
Compute $a_{\overline{n}|i}$ via SBR C'
Compute $(a_{\overline{n}|i})(1+i)$
RTN
Lbl D
is FV = 0
yes
no
Store FV in R04
RTN
Lbl CE
$FV = PV(1+i)^N$
Store FV in R04
RTN
Lbl E'
Clear T register
Fix display format at 2 decimal places
RTN

ML-18 Program Listing

```
000  76 LBL
001  16 A'
002  53 (
003  53 (
004  43 RCL
005  09  09
006  45 YX
007  43 RCL
008  01  01
009  75  -
010  42 STO
011  12  12
012  01  1
013  54  )
014  55  ÷
015  43 RCL
016  08  08
017  54  )
018  92 RTN
019  76 LBL
020  17 B'
021  53  (
022  16 A'
023  65  ×
024  43 RCL
025  09  09
026  54  )
027  92 RTN
028  76 LBL
029  18 C'
030  53  (
031  16 A'
032  55  ÷
033  43 RCL
034  12  12
035  54  )
036  92 RTN
037  76 LBL
038  19 D'
039  18 C'
040  53  (
041  24 CE
042  65  ×
043  43 RCL
044  09  09
045  54  )
046  92 RTN
047  76 LBL
048  10 E'
049  29 CP
050  58 FIX
051  02  02
052  92 RTN
053  76 LBL
054  71 SBR
055  53  (
056  43 RCL
057  04  04
058  55  ÷
059  43 RCL
060  03  03
061  54  )
062  92 RTN
063  76 LBL
064  25 CLR
065  53  (
066  71 SBR
067  71 SBR
068  23 LNX
069  55  ÷
070  43 RCL
071  09  09
072  23 LNX
073  54  )
074  42 STO
075  01  01
076  92 RTN
077  76 LBL
078  11  A
079  67  EQ
080  25 CLR
081  42 STO
082  01  01
083  92 RTN
084  76 LBL
085  22 INV
086  53  (
087  01  1
088  00  0
089  00  0
090  75  -
091  24 CE
092  65  ×
093  71 SBR
094  71 SBR
095  22 INV
096  45 YX
097  43 RCL
098  01  01
099  54  )
100  94 +/-
101  67  EQ
102  15  E
103  76 LBL
104  12  B
105  67  EQ
106  22 INV
107  76 LBL
108  15  E
109  42 STO
110  02  02
111  53  (
112  24 CE
113  55  ÷
114  01  1
115  00  0
116  00  0
117  85  +
118  42 STO
119  08  08
120  01  1
121  54  )
122  42 STO
123  09  09
124  43 RCL
125  02  02
126  92 RTN
127  76 LBL
128  23 LNX
129  53  (
130  43 RCL
131  04  04
132  55  ÷
133  43 RCL
134  09  09
135  45 YX
136  43 RCL
137  01  01
138  54  )
139  42 STO
140  03  03
141  92 RTN
142  76 LBL
143  13  C
144  67  EQ
145  23 LNX
146  42 STO
147  03  03
148  92 RTN
149  76 LBL
150  24 CE
151  53  (
152  43 RCL
153  03  03
154  65  ×
155  43 RCL
156  09  09
157  45 YX
158  43 RCL
159  01  01
160  54  )
161  42 STO
162  04  04
163  92 RTN
164  76 LBL
165  14  D
166  67  EQ
167  24 CE
168  42 STO
169  04  04
170  92 RTN

001  16 A'
020  17 B'
029  18 C'
038  19 D'
048  10 E'
054  71 SBR
064  25 CLR
078  11  A
085  22 INV
104  12  B
108  15  E
128  23 LNX
143  13  C
150  24 CE
165  14  D
```

ML-19

ANNUITIES

ML-19 is second only to ML-02 in length and complexity. Overall, it is a fairly well written program with the same good points noted for ML-18, upon which it relies heavily for subroutines. For a discussion of annuities and the applicable formulas consult the Master Library Manual.

Interface procedure:

The most efficient interface procedure will depend on the particular problem at hand and the current states of flags 1-4, the T register, and R05.

T register:
The T register must always be zero when using ML-19. If all the other conditions for a particular problem are already met, then a CP will suffice for initialization.

Flags:
One and only one flag must be set for a particular problem according to the following table:

Sinking Fund:	flag 1
Ann. Due/FV:	flag 2
Ord. Ann./PV:	flag 3
Ann. Due/PV:	flag 4

The most efficient way to reset flags 1-4 if more than one is set or it is unknown which is set, is to use PGM 19 E', which also clears R05 and the T register. Then simply set the appropriate flag instead of using keys A'-D'.

R05:
For 5, of the possible 18 types of problems, R05 must be zero. These are indicated in the table of register assignments. In cases where R05 is not used or where a non-zero balloon payment exists, the contents of R05 can be ignored for now.

Input data:
Prestore any data not already in the appropriate registers according to the following list:

N:	Use STO 01. Note that -N must be prestored in R11 only for Ann. Due/PV interest calculation. In this case, use PGM 19 A.

Input data (cont.)

I: Use PGM 18 E, with I in the display, to store I in R02, i in R08, and i+1 in R09.

PMT: Use STO 03

PV/FV: Use STO 04

B.PMT: Use STO 05

Computation:
With a zero display, execute the appropriate user defined key preceded by PGM 19 ...returns with the value of the unknown variable in the display and the appropriate register.

Display format:
For PMT, PV/FV, and B.PMT, the display returns in FIX 02 format. For I, the display returns in FIX 04 format. For N, the display returns in floating decimal format.

Special Notes:

(1) For a discussion of the eight NOP's in ML-19 see the notes for ML-02.

(2) Contrary to M.L.M., R07 is not used.

(3) R06 is used only when calculating I with a B.PMT involved.

(4) R10 is used only when calculating N.

(5) R13 and R14 are used only for calculating I.

(6) The reason for using sequences such as "Lbl A GTO 378" is to speed up program execution. A label search can take as long as two seconds, depending upon how far down in program memory the label is, which results in a noticeably longer data input time than if the labels are put at the top of program memory and followed by absolute addresses. As an example, notice the difference in time for executing labels A and A' with a non-zero display. The label A sequence is noticeably faster even though it is over three times longer!

Special applications:

(1) PGM 19 E' resets flags 1-4 and clears R05 and T reg.

Register assignments are: (cont. on next page)

Problem type	Unknown var.		R01	R02	R03	R04	R05	R06	R08	R09	R10	R11	R12	R13	R14
SINKING FUND	N	INITIAL	—	I	PMT	FV	0	—	i	1+i	—	—	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	used	-N	—	—	—
	I	INITIAL	N	—	PMT	FV	0*	—	—	—	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	used	used
	PMT	INITIAL	N	I	—	FV	0	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	—	—
	FV	INITIAL	N	I	PMT	—	0	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	—	—
ANN DUE FV	N	INITIAL	—	I	PMT	FV	0*	—	i	1+i	—	—	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	used	-N	—	—	—
	I	INITIAL	N	—	PMT	FV	0*	—	—	—	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	used	used
	PMT	INITIAL	N	I	—	FV	0	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	—	—
	FV	INITIAL	N	I	PMT	—	0	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	FV	0	—	i	1+i	—	-N	used	—	—
ORD ANN PV	N	INITIAL	—	I	PMT	PV	B.PMT	—	i	1+i	—	—	—	—	—
		FINAL	N	I	PMT	PV	B.PMT	—	i	1+i	used	-N	—	—	—
	I	INITIAL	N	—	PMT	PV	B.PMT	—	—	—	—	-N*	—	—	—
		FINAL	N	I	PMT	PV	B.PMT	use	i	1+i	—	-N	used	used	used
	PMT	INITIAL	N	I	—	PV	B.PMT	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	PV	B.PMT	—	i	1+i	—	-N	used	—	—
	PV	INITIAL	N	I	PMT	—	B.PMT	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	PV	B.PMT	—	i	1+i	—	-N	used	—	—
	B.PMT	INITIAL	N	I	PMT	PV	—	—	i	1+i	—	-N*	—	—	—
		FINAL	N	I	PMT	PV	B.PMT	—	i	1+i	—	-N	used	—	—

Register assignments are (cont.):

Problem type	Unknown var.		R01	R02	R03	R04	R05	R06	R08	R09	R10	R11	R12	R13	R14
ANN DUE PV	N	INITIAL	---	I	PMT	PV	B.PMT	---	i	1+i	---	---	---	---	---
		FINAL	N	I	PMT	PV	B.PMT	---	i	1+i	used	-N	---	---	---
	I	INITIAL	N	---	PMT	PV	B.PMT	---	---	---	---	-N	---	---	---
		FINAL	N	I	PMT	PV	B.PMT	use	i	1+i	---	-N	used	used	used
	PMT	INITIAL	N	I	---	PV	B.PMT	---	i	1+i	---	-N*	---	---	---
		FINAL	N	I	PMT	PV	B.PMT	---	i	1+i	---	-N	used	---	---
	PV	INITIAL	N	I	PMT	---	B.PMT	---	i	1+i	---	-N*	---	---	---
		FINAL	N	I	PMT	PV	B.PMT	---	i	1+i	---	-N	used	---	---
	B.PMT	INITIAL	N	I	PMT	PV	---	---	i	1+i	---	-N*	---	---	---
		FINAL	N	I	PMT	PV	B.PMT	---	i	1+i	---	-N	used	---	---

*Values which are stored in the particular register during normal use of ML-19 to find the given variable but which are not needed and can be omitted during interfacing. The register can then be used for other data and will not be affected by use of ML-19.

---Not affected by ML-19 while finding the given variable if in "FINAL" row or immaterial if in "INITIAL" row.

VAL = PV or FV
depending upon the
specific problem.

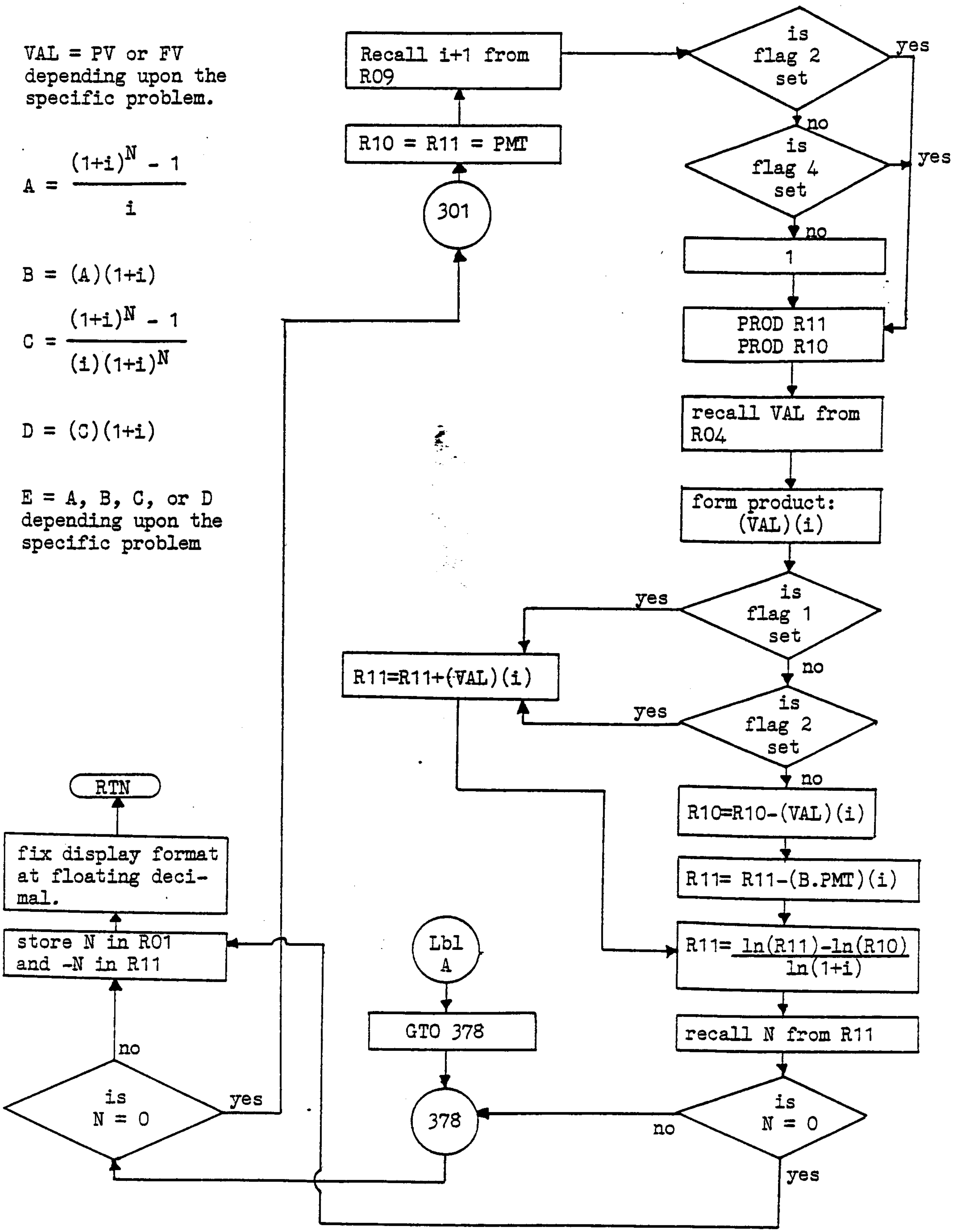
$$A = \frac{(1+i)^N - 1}{i}$$

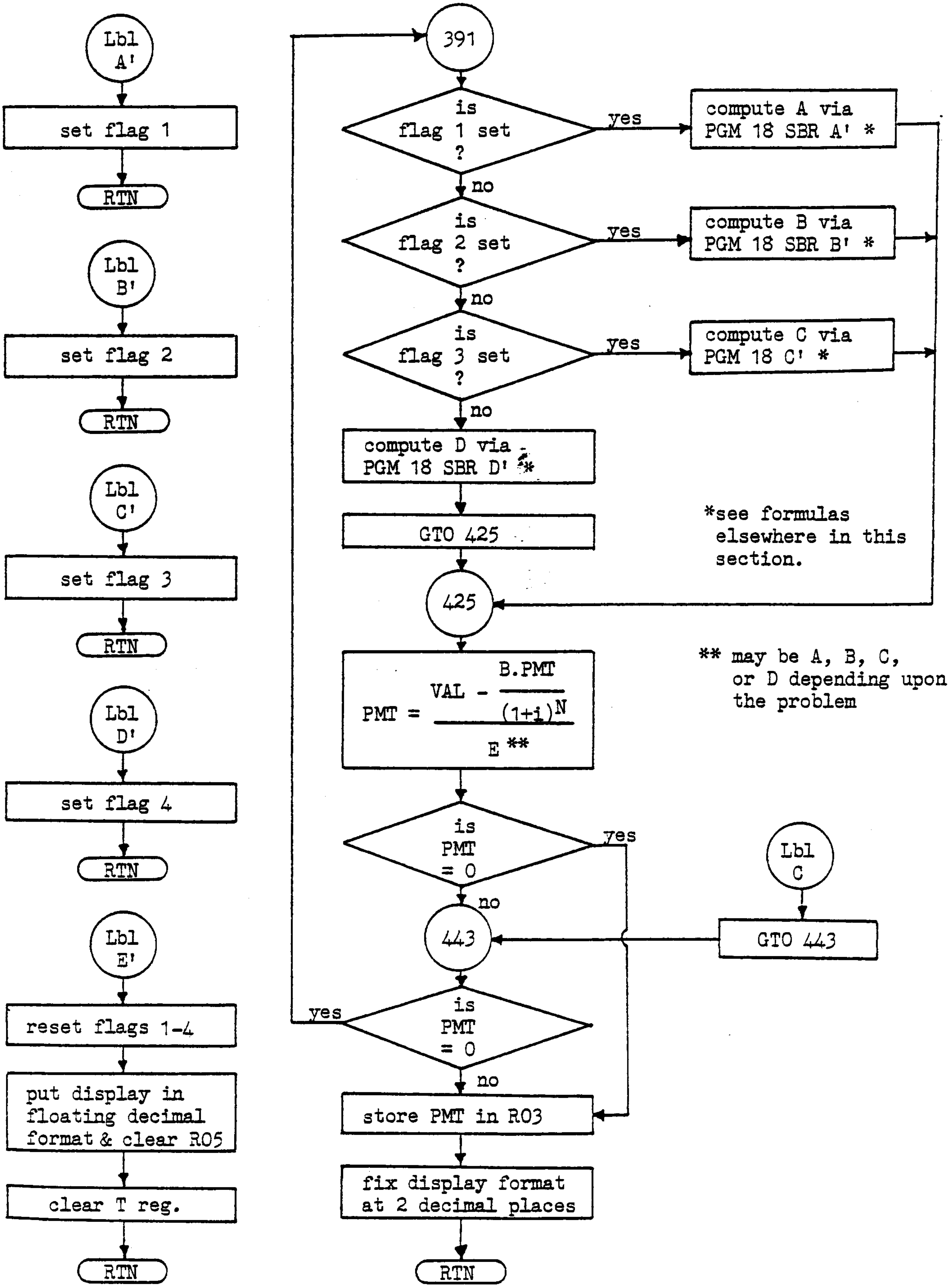
$$B = (A)(1+i)$$

$$C = \frac{(1+i)^N - 1}{(i)(1+i)^N}$$

$$D = (C)(1+i)$$

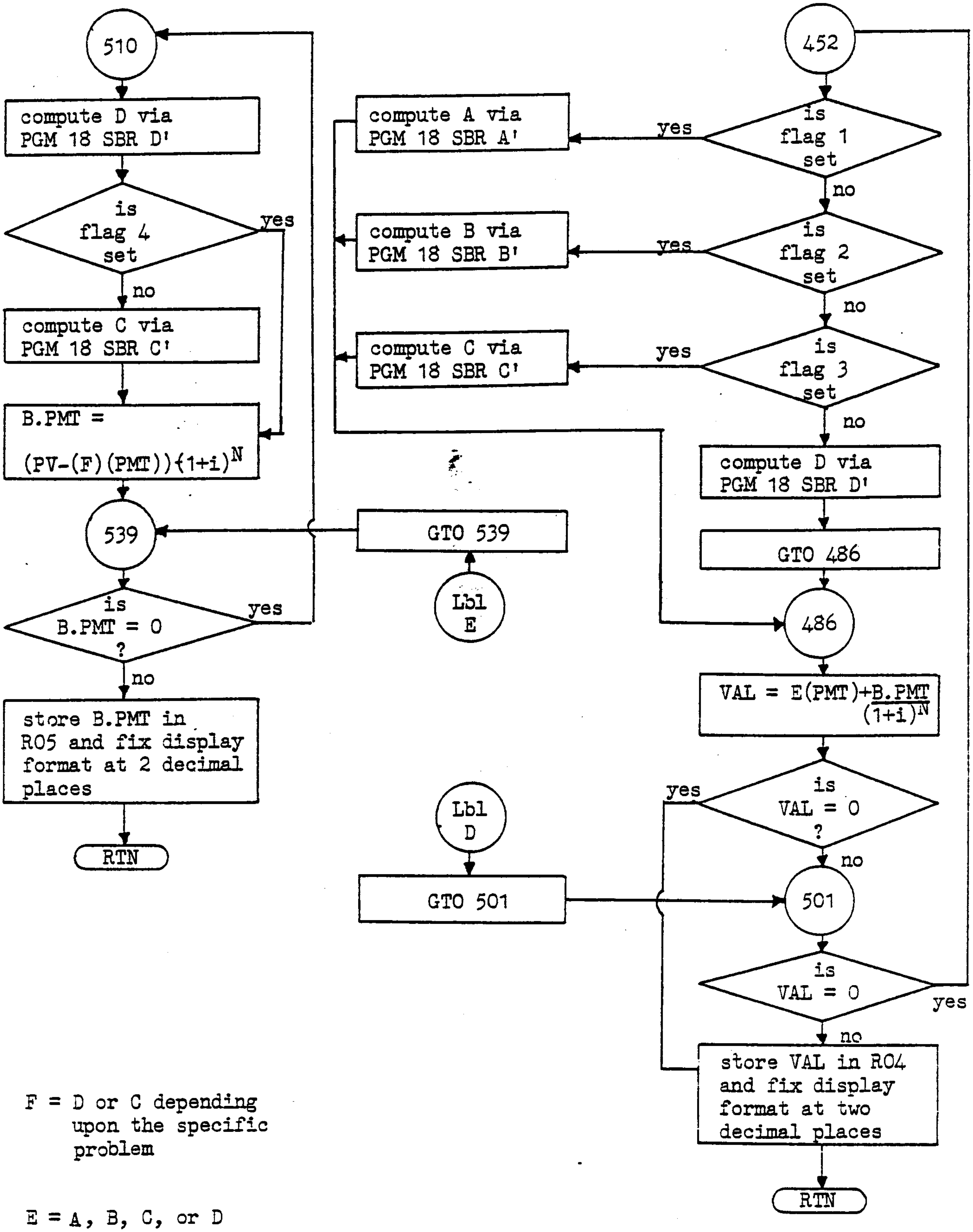
E = A, B, C, or D
depending upon the
specific problem





*see formulas elsewhere in this section.

** may be A, B, C, or D depending upon the problem


510
compute D via PGM 18 SBR D'
is flag 4 set
yes
no
compute C via PGM 18 SBR C'
B.PMT = (PV-(F)(PMT))(1+i)^N
539
is B.PMT = 0 ?
yes
no
store B.PMT in R05 and fix display format at 2 decimal places
RTN
compute A via PGM 18 SBR A'
compute B via PGM 18 SBR B'
compute C via PGM 18 SBR C'
GTO 539
Lbl E
Lbl D
GTO 501
452
is flag 1 set
yes
no
is flag 2 set
yes
no
is flag 3 set
yes
no
compute D via PGM 18 SBR D'
GTO 486
486
VAL = E(PMT)+B.PMT/(1+i)^N
is VAL = 0 ?
yes
no
501
is VAL = 0
yes
no
store VAL in R04 and fix display format at two decimal places
RTN

F = D or C depending upon the specific problem

E = A, B, C, or D depending upon the specific problem

ML-19 Program Listing

```
000  76  LBL   050  53  (     100  55  ÷     150  43  RCL
001  11  A     051  24  CE    101  43  RCL   151  10  10
002  61  GTO   052  55  ÷     102  09  09    152  54  )
003  03  03    053  43  RCL   103  55  ÷     153  42  STO
004  78  78    054  01  01    104  43  RCL   154  08  08
005  76  LBL   055  33  X²    105  08  08    155  53  (
006  12  B     056  75  -     106  54  )     156  43  RCL
007  61  GTO   057  43  RCL   107  87  IFF   157  01  01
008  02  02    058  10  10    108  01  01    158  85  +
009  92  92    059  35  1/X   109  01  01    159  33  X²
010  76  LBL   060  54  )     110  20  20    160  55  ÷
011  13  C     061  42  STO   111  53  (     161  02  2
012  61  GTO   062  08  08    112  24  CE    162  85  +
013  04  04    063  44  SUM   113  65  ×     163  43  RCL
014  43  43    064  09  09    114  43  RCL   164  14  14
015  76  LBL   065  36  PGM   115  09  09    165  55  ÷
016  14  D     066  18  18    116  75  -     166  43  RCL
017  61  GTO   067  16  A'    117  43  RCL   167  01  01
018  05  05    068  42  STO   118  14  14    168  54  )
019  01  01    069  14  14    119  54  )     169  87  IFF
020  76  LBL   070  87  IFF   120  22  INV   170  03  03
021  15  E     071  01  01    121  49  PRD   171  01  01
022  61  GTO   072  00  00    122  13  13    172  84  84
023  05  05    073  80  80    123  43  RCL   173  53  (
024  39  39    074  53  (     124  13  13    174  43  RCL
025  01  1     075  24  CE    125  44  SUM   175  11  11
026  42  STO   076  65  ×     126  08  08    176  85  +
027  09  09    077  43  RCL   127  44  SUM   177  33  X²
028  07  7     078  09  09    128  09  09    178  55  ÷
029  94  +/-   079  54  )     129  50  I×I   179  02  2
030  22  INV   080  53  (     130  77  GE    180  85  +
031  28  LOG   081  24  CE    131  00  00    181  43  RCL
032  32  X:T   082  75  -     132  65  65    182  14  14
033  53  (     083  43  RCL   133  61  GTO   183  54  )
034  43  RCL   084  10  10    134  02  02    184  22  INV
035  04  04    085  54  )     135  80  80    185  49  PRD
036  55  ÷     086  42  STO   136  53  (     186  08  08
037  43  RCL   087  13  13    137  43  RCL   187  43  RCL
038  03  03    088  53  (     138  05  05    188  08  08
039  54  )     089  43  RCL   139  55  ÷     189  44  SUM
040  42  STO   090  14  14    140  43  RCL   190  09  09
041  10  10    091  55  ÷     141  03  03    191  36  PGM
042  87  IFF   092  43  RCL   142  85  +     192  18  18
043  03  03    093  08  08    143  42  STO   193  18  C'
044  01  01    094  75  -     144  14  14    194  42  STO
045  36  36    095  43  RCL   145  43  RCL   195  06  06
046  87  IFF   096  01  01    146  01  01    196  87  IFF
047  04  04    097  65  ×     147  49  PRD   197  03  03
048  01  01    098  43  RCL   148  14  14    198  02  02
049  36  36    099  12  12    149  75  -     199  06  06
```

```
200  53  (
201  24  CE
202  65  ×
203  43  RCL
204  09  09
205  54  )
206  53  (
207  24  CE
208  85  +
209  43  RCL
210  14  14
211  55  ÷
212  43  RCL
213  01  01
214  55  ÷
215  43  RCL
216  12  12
217  75  -
218  43  RCL
219  10  10
220  54  )
221  42  STO
222  13  13
223  53  (
224  43  RCL
225  06  06
226  55  ÷
227  43  RCL
228  08  08
229  75  -
230  43  RCL
231  01  01
232  55  ÷
233  43  RCL
234  12  12
235  55  ÷
236  43  RCL
237  09  09
238  55  ÷
239  43  RCL
240  08  08
241  54  )
242  87  IFF
243  03  03
244  02  02
245  55  55
246  53  (
247  24  CE
248  65  ×
249  43  RCL
250  09  09
251  85  +
252  43  RCL
253  06  06
254  54  )
255  53  (
256  24  CE
257  85  +
258  43  RCL
259  14  14
260  55  ÷
261  43  RCL
262  12  12
263  55  ÷
264  43  RCL
265  09  09
266  54  )
267  22  INV
268  49  PRD
269  13  13
270  43  RCL
271  13  13
272  44  SUM
273  08  08
274  44  SUM
275  09  09
276  50  I×I
277  77  GE
278  01  01
279  91  91
280  53  (
281  43  RCL
282  08  08
283  65  ×
284  01  1
285  00  0
286  00  0
287  54  )
288  29  CP
289  67  EQ
290  02  02
291  95  95
292  67  EQ
293  00  00
294  25  25
295  58  FIX
296  04  04
297  36  PGM
298  18  18
299  15  E
300  92  RTN
301  68  NOP
302  43  RCL
303  03  03
304  42  STO
305  11  11
306  42  STO
307  10  10
308  43  RCL
309  09  09
310  87  IFF
311  02  02
312  03  03
313  19  19
314  87  IFF
315  04  04
316  03  03
317  19  19
318  01  1
319  49  PRD
320  11  11
321  49  PRD
322  10  10
323  53  (
324  43  RCL
325  04  04
326  65  ×
327  43  RCL
328  08  08
329  54  )
330  87  IFF
331  01  01
332  03  03
333  54  54
334  87  IFF
335  02  02
336  03  03
337  54  54
338  94  +/-
339  44  SUM
340  10  10
341  53  (
342  43  RCL
343  05  05
344  65  ×
345  43  RCL
346  08  08
347  54  )
348  94  +/-
349  44  SUM
350  11  11
351  61  GTO
352  03  03
353  56  56
354  44  SUM
355  11  11
356  43  RCL
357  11  11
358  23  LNX
359  42  STO
360  11  11
361  43  RCL
362  10  10
363  23  LNX
364  94  +/-
365  44  SUM
366  11  11
367  43  RCL
368  09  09
369  23  LNX
370  35  1/X
371  49  PRD
372  11  11
373  43  RCL
374  11  11
375  67  EQ
376  03  03
377  81  81
378  67  EQ
379  03  03
380  01  01
381  42  STO
382  01  01
383  94  +/-
384  42  STO
385  11  11
386  94  +/-
387  58  FIX
388  09  09
389  68  NOP
390  92  RTN
391  68  NOP
392  87  IFF
393  01  01
394  04  04
395  22  22
396  87  IFF
397  02  02
398  04  04
399  16  16
```

```
400   87  IFF
401   03   03
402   04   04
403   10   10
404   36  PGM
405   18   18
406   19  D'
407   61  GTO
408   04   04
409   25   25
410   36  PGM
411   18   18
412   18  C'
413   61  GTO
414   04   04
415   25   25
416   36  PGM
417   18   18
418   17  B'
419   61  GTO
420   04   04
421   25   25
422   36  PGM
423   18   18
424   16  A'
425   53   (
426   24  CE
427   55   ÷
428   53   (
429   43  RCL
430   04   04
431   75   -
432   43  RCL
433   05   05
434   55   ÷
435   43  RCL
436   12   12
437   54   )
438   54   )
439   35  1/X
440   67   EQ
441   04   04
442   46   46
443   67   EQ
444   03   03
445   91   91
446   42  STO
447   03   03
448   58  FIX
449   02   02
```

```
450   68  NOP
451   92  RTN
452   68  NOP
453   87  IFF
454   01   01
455   04   04
456   83   83
457   87  IFF
458   02   02
459   04   04
460   77   77
461   87  IFF
462   03   03
463   04   04
464   71   71
465   36  PGM
466   18   18
467   19  D'
468   61  GTO
469   04   04
470   86   86
471   36  PGM
472   18   18
473   18  C'
474   61  GTO
475   04   04
476   86   86
477   36  PGM
478   18   18
479   17  B'
480   61  GTO
481   04   04
482   86   86
483   36  PGM
484   18   18
485   16  A'
486   53   (
487   24  CE
488   65   ×
489   43  RCL
490   03   03
491   85   +
492   43  RCL
493   05   05
494   55   ÷
495   43  RCL
496   12   12
497   54   )
498   67   EQ
499   05   05
```

```
500   04   04
501   67   EQ
502   04   04
503   52   52
504   42  STO
505   04   04
506   58  FIX
507   02   02
508   68  NOP
509   92  RTN
510   68  NOP
511   36  PGM
512   18   18
513   19  D'
514   87  IFF
515   04   04
516   05   05
517   21   21
518   36  PGM
519   18   18
520   18  C'
521   53   (
522   53   (
523   24  CE
524   65   ×
525   43  RCL
526   03   03
527   75   -
528   43  RCL
529   04   04
530   54   )
531   94  +/-
532   65   ×
533   43  RCL
534   12   12
535   54   )
536   67   EQ
537   05   05
538   42   42
539   67   EQ
540   05   05
541   10   10
542   42  STO
543   05   05
544   58  FIX
545   02   02
546   68  NOP
547   92  RTN
548   76  LBL
549   16  A'
```

```
550   86  STF
551   01   01
552   92  RTN
553   76  LBL
554   17  B'
555   86  STF
556   02   02
557   92  RTN
558   76  LBL
559   18  C'
560   86  STF
561   03   03
562   92  RTN
563   76  LBL
564   19  D'
565   86  STF
566   04   04
567   92  RTN
568   76  LBL
569   10  E'
570   22  INV
571   86  STF
572   01   01
573   22  INV
574   86  STF
575   02   02
576   22  INV
577   86  STF
578   03   03
579   22  INV
580   86  STF
581   04   04
582   00   0
583   42  STO
584   05   05
585   58  FIX
586   09   09
587   29  CP
588   92  RTN

001   11   A
006   12   B
011   13   C
016   14   D
021   15   E
549   16  A'
554   17  B'
559   18  C'
564   19  D'
569   10  E'
```

ML-20

DAY OF THE WEEK
DAYS BETWEEN DATES

ML-20 determines the number of days between dates after the year 1582 and the day of the week for any date after 1582. Program execution is based on the formulas given in the Master Library Manual.

Interface procedure:

If known valid dates are to be input and the form MMDD.YYYY is not desired then:

(1) Prestore month (MM) in R01, day (DD) in R02, and year (YYYY) in R03.
(2) Execute PGM 20 SBR 086 ...returns with "factor" in display.
(3) Store factor for the first date in an available register and repeats steps (1) and (2) for the second date.
(4) Subtract the two factors to get the number of days between dates.

If the day of the week is desired for a known valid date and the form MMDD.YYYY is not desired then:

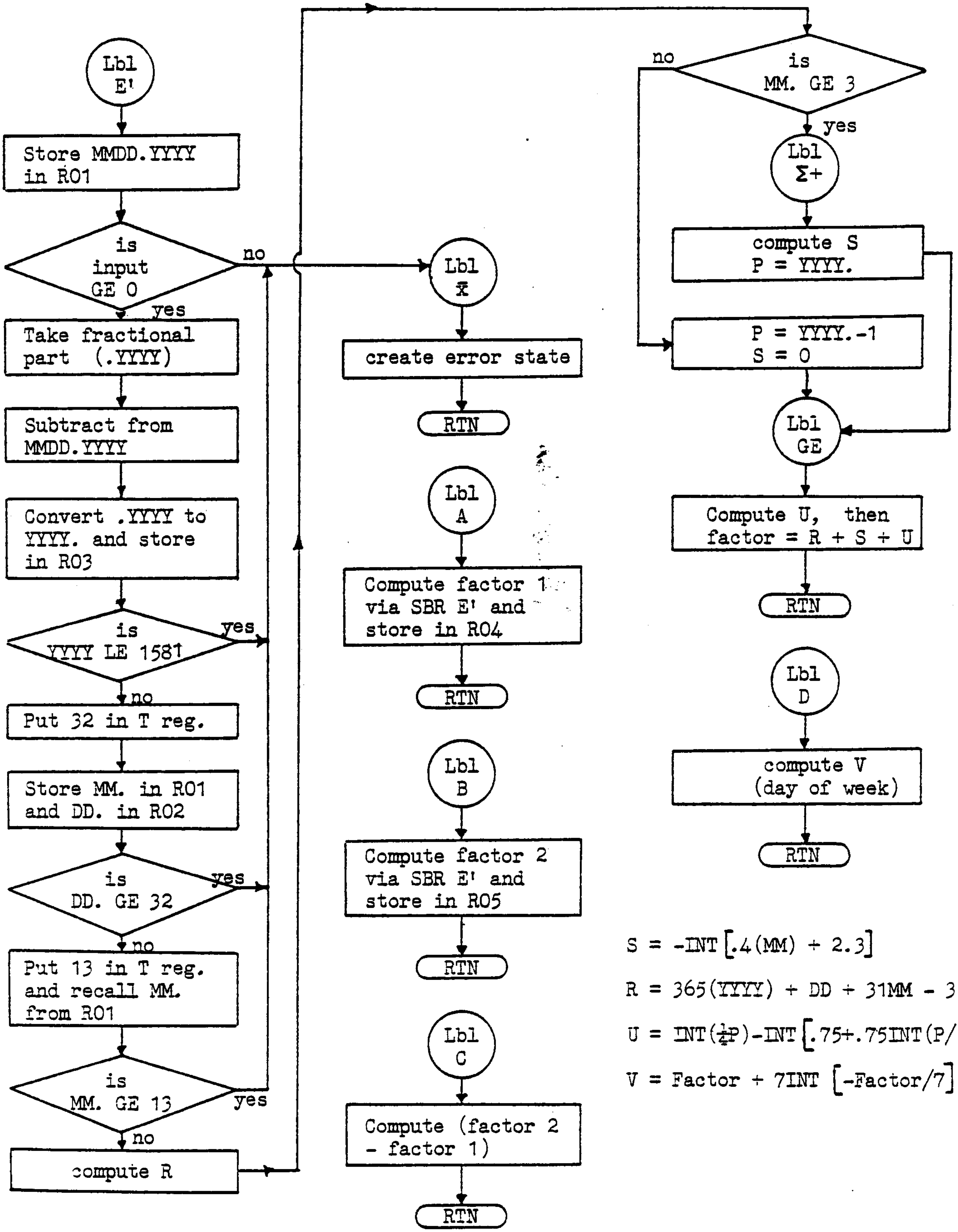
(1) Prestore month (MM) in R01, day (DD) in R02, and year (YYYY) in R03.
(2) Execute PGM 20 SBR 086 ((SBR 177 ...returns with the numeral corresponding to the day of the week as given in M.L.M. Note that the factor for the date is stored in R01. If pending operations do not exist in the calling routine, the ((may be omitted.

Normal use data:

Flags affected: none
Registers used: 1-5 AND THE treg.
Parentheses levels: 3 unless label D is used, 5 if it is used.
Subroutine levels: 1

Special applications:

PGM 20 C evaluates (RCL 05 - RCL 04) without affecting pending operations.



$S = -\mathrm{INT}[.4(MM) + 2.3]$

$R = 365(YYYY) + DD + 31MM - 31$

$U = \mathrm{INT}(\frac{1}{4}P) - \mathrm{INT}[.75 + .75\mathrm{INT}(P/100)]$

$V = \mathrm{Factor} + 7\mathrm{INT}[-\mathrm{Factor}/7]$

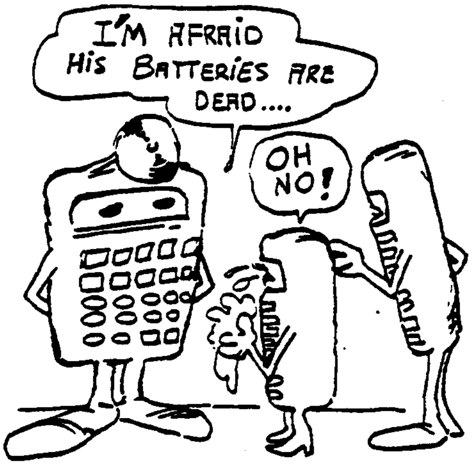
ML-20 Program Listing

```
000  76  LBL
001  78  Σ+
002  53  (
003  93  .
004  04  4
005  65  ×
006  43  RCL
007  01  01
008  85  +
009  02  2
010  93  .
011  03  3
012  54  )
013  59  INT
014  94  +/-
015  85  +
016  61  GTO
017  77  GE

018  76  LBL
019  79  x̄
020  00  0
021  35  1/X
022  92  RTN

023  76  LBL
024  10  E'
025  53  (
026  42  STO
027  01  01
028  29  CP
029  22  INV
030  77  GE
031  79  x̄
032  22  INV
033  59  INT
034  22  INV
035  44  SUM
036  01  01
037  65  ×
038  04  4
039  22  INV
040  28  LOG
041  54  )
042  42  STO
043  03  03
044  32  X:T
045  01  1
046  05  5
047  08  8
048  01  1
049  77  GE

050  79  x̄
051  03  3
052  02  2
053  32  X:T
054  53  (
055  53  (
056  43  RCL
057  01  01
058  55  ÷
059  01  1
060  00  0
061  00  0
062  54  )
063  42  STO
064  01  01
065  22  INV
066  59  INT
067  22  INV
068  44  SUM
069  01  01
070  65  ×
071  01  1
072  00  0
073  00  0
074  54  )
075  42  STO
076  02  02
077  77  GE
078  79  x̄
079  01  1
080  03  3
081  32  X:T
082  43  RCL
083  01  01
084  77  GE
085  79  x̄
086  53  (
087  03  3
088  06  6
089  05  5
090  65  ×
091  43  RCL
092  03  03
093  85  +
094  43  RCL
095  02  02
096  85  +
097  03  3
098  01  1
099  65  ×

100  43  RCL
101  01  01
102  75  -
103  03  3
104  01  1
105  85  +
106  03  3
107  32  X:T
108  43  RCL
109  01  01
110  77  GE
111  78  Σ+
112  01  1
113  22  INV
114  44  SUM
115  03  03
116  76  LBL
117  77  GE
118  53  (
119  43  RCL
120  03  03
121  55  ÷
122  04  4
123  54  )
124  59  INT
125  75  -
126  53  (
127  93  .
128  07  7
129  05  5
130  85  +
131  53  (
132  43  RCL
133  03  03
134  55  ÷
135  01  1
136  00  0
137  00  0
138  54  )
139  59  INT
140  65  ×
141  93  .
142  07  7
143  05  5
144  54  )
145  59  INT
146  54  )
147  92  RTN

148  76  LBL
149  11  A
150  10  E'
151  42  STO
152  04  04
153  00  0
154  92  RTN

155  76  LBL
156  12  B
157  10  E'
158  42  STO
159  05  05
160  00  0
161  92  RTN

162  76  LBL
163  13  C
164  53  (
165  43  RCL
166  05  05
167  75  -
168  43  RCL
169  04  04
170  54  )
171  92  RTN

172  76  LBL
173  14  D
174  53  (
175  53  (
176  10  E'
177  42  STO
178  01  01
179  94  +/-
180  55  ÷
181  07  7
182  54  )
183  59  INT
184  65  ×
185  07  7
186  85  +
187  43  RCL
188  01  01
189  54  )
190  92  RTN

001  78  Σ+
019  79  x̄
024  10  E'
117  77  GE
149  11  A
156  12  B
163  13  C
173  14  D
```

ML-21

HI-LO GAME

The HI-LO game should have been called the HI-BYE game since it quickly becomes boring. (The old luner lander game in the SR-52 master library was much more interesting.)



As a synopsis, when the user guesses a number, the calculator uses the random number generator in program 15 and an input seed to generate a number from 1-1023. The user then inputs a guess from which the calculator computes an error of which the sign determines whether a 1 or a -1 is output. (Note that OP 10 would have been much "cleaner.") When the user guesses the exact number a flashing zero is displayed and the number of guesses can be recalled.

If the calculator guesses, its first guess is always 512, which splits the possible range in half. If the user indicates that 512 is too low, it then sets the new range as 512-1023 and splits the difference with a new guess. This procedure of reduction by halves is repeated until the number is "guessed" Note that the calculator never takes more than 10 guesses to pinpoint the number.

Normal use data:

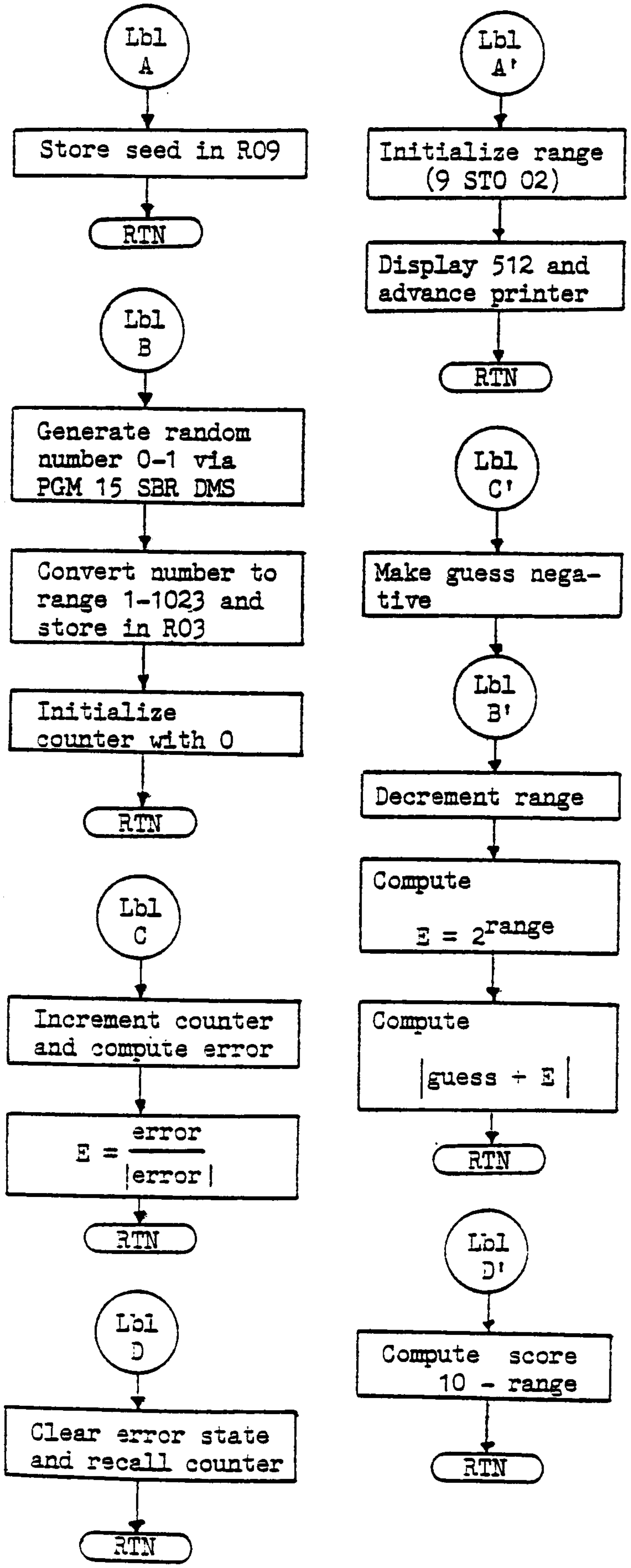
Flags affected: none
Parentheses levels: 4
Subroutine levels: 1
Registers used: R02, R03, R04, R05, R07 and R09

Special notes:

Contrary to Master Library Manual, register 01 is not used but register 07 is. (by PGM 15)

Interface procedure:

Can't imagine why you'd want to but if you do, simply follow the user instructions and precede each user defined key with PGM 21; except for labels D and A where you can use CE RCL 04 and STO 09 respectively.


Lbl A
Store seed in R09
RTN
Lbl B
Generate random number 0-1 via PGM 15 SBR DMS
Convert number to range 1-1023 and store in R03
Initialize counter with 0
RTN
Lbl C
Increment counter and compute error
E = error / |error|
RTN
Lbl D
Clear error state and recall counter
RTN
Lbl A'
Initialize range (9 STO 02)
Display 512 and advance printer
RTN
Lbl C'
Make guess negative
Lbl B'
Decrement range
Compute E = 2^range
Compute |guess ÷ E|
RTN
Lbl D'
Compute score 10 - range
RTN

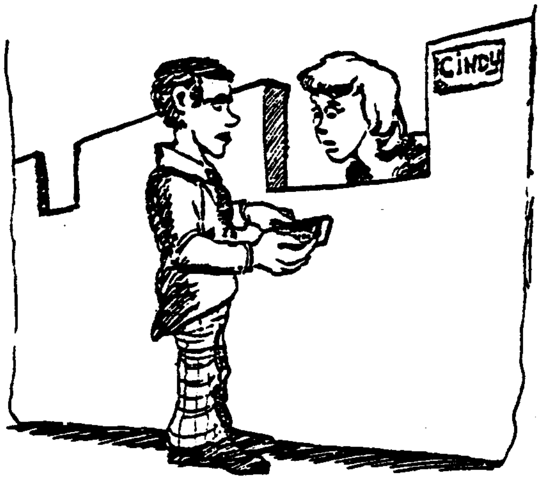
ML-21 Program Listing

```
000   76  LBL
001   13  C
002   32  X:T
003   01   1
004   44  SUM
005   04   04
006   53   (
007   32  X:T
008   75   -
009   43  RCL
010   03   03
011   54   )
012   42  STO
013   05   05
014   53   (
015   35  1/X
016   50  I×I
017   65   ×
018   43  RCL
019   05   05
020   54   )
021   92  RTN
-----------------
022   76  LBL
023   18  C'
024   94  +/-
025   76  LBL
026   17  B'
027   32  X:T
028   01   1
029   22  INV
030   44  SUM
031   02   02
032   53   (
033   02   2
034   45  YX
035   43  RCL
036   02   02
037   85   +
038   32  X:T
039   54   )
040   50  I×I
041   92  RTN
-----------------
042   76  LBL
043   12  B
044   53   (
045   36  PGM
046   15   15
047   71  SBR
048   88  DMS
049   65   ×
050   01   1
051   00   0
052   02   2
053   03   3
054   85   +
055   01   1
056   54   )
057   59  INT
058   42  STO
059   03   03
060   00   0
061   42  STO
062   04   04
063   92  RTN
-----------------
064   76  LBL
065   14  D
066   24  CE
067   43  RCL
068   04   04
069   92  RTN
-----------------
070   76  LBL
071   16  A'
072   09   9
073   42  STO
074   02   02
075   05   5
076   01   1
077   02   2
078   98  ADV
079   92  RTN
-----------------
080   76  LBL
081   19  D'
082   53   (
083   01   1
084   00   0
085   75   -
086   43  RCL
087   02   02
088   54   )
089   92  RTN
-----------------
090   76  LBL
091   11  A
092   42  STO
093   09   09
094   92  RTN
-----------------

001   13   C
023   18   C'
026   17   B'
043   12   B
065   14   D
071   16   A'
081   19   D'
091   11   A
```

ML-22

CHECKING/SAVINGS ACCOUNT MANAGEMENT

ML-22 is a prime example of how to make an easy job hard and waste a lot of memory space (i.e. program steps) doing it.



"But I can't be overdrawn....my calculator says I still have $314.15 in my account...."

The call to PGM 18 B at step 059 stores %I in R02, i in R08, and i+1 in R09 (i = %I/100). The i in R08 is not used at all and the i+1 in R09 is used only once. The sequence PGM 18 B plus the registers used to store i and i+1 results in a total minimum of 2(8)+3 or 19 equivalent program steps. By contrast, the sequence STO 02....(RCL 02 ÷ 100 + 1)... is only a maximum of 12 steps.

The call to PGM 18 A at step 066 simply stores N in R01. The sequence CP PGM 18 A is obviously longer than simply STO 01, and considerably slower.

The call to PGM 18 SBR CE at step 074 calculates:

$$FV = PV(1+i)^N$$

but requires that data be moved from R06 to R03 and PGM 18 uses R04, which results in wasting two more memories. Thus, 2(8)+9 gives 25 equivalent steps versus 18 steps for ...(RCL 06 X (RCL 02 ÷ 100 + 1) y^x RCL 01).... Note however that 10 of these steps were previously considered when 1+i was calculated. To summarize, the program has called another program three times and used the equivalent of 48 program steps to do what could be done directly with 22 steps.

A more subtle fault is the use of both a pointer and a flag for routing data to and from only two memories. Note the difference between labels E and E'. E' uses 18 steps and a register to do the opposite of what E does in 13 steps.

Label A is used only to access label E'. It would have been better to simply call the label E' routine A, thus saving 3 steps and a subroutine level.

Register assignments are:

R01: Number of periods (N)
R02: Interest rate (%) per period
R03: Savings balance before compounding (PV)
R04: Savings balance after compounding (FV)
R05: Current checking balance
R06: Current savings balance
R07: Interest rate per year (%I)
R08: Decimal interest rate per year (i)
R09: i+1
R10: Balance pointer

Interface procedure:

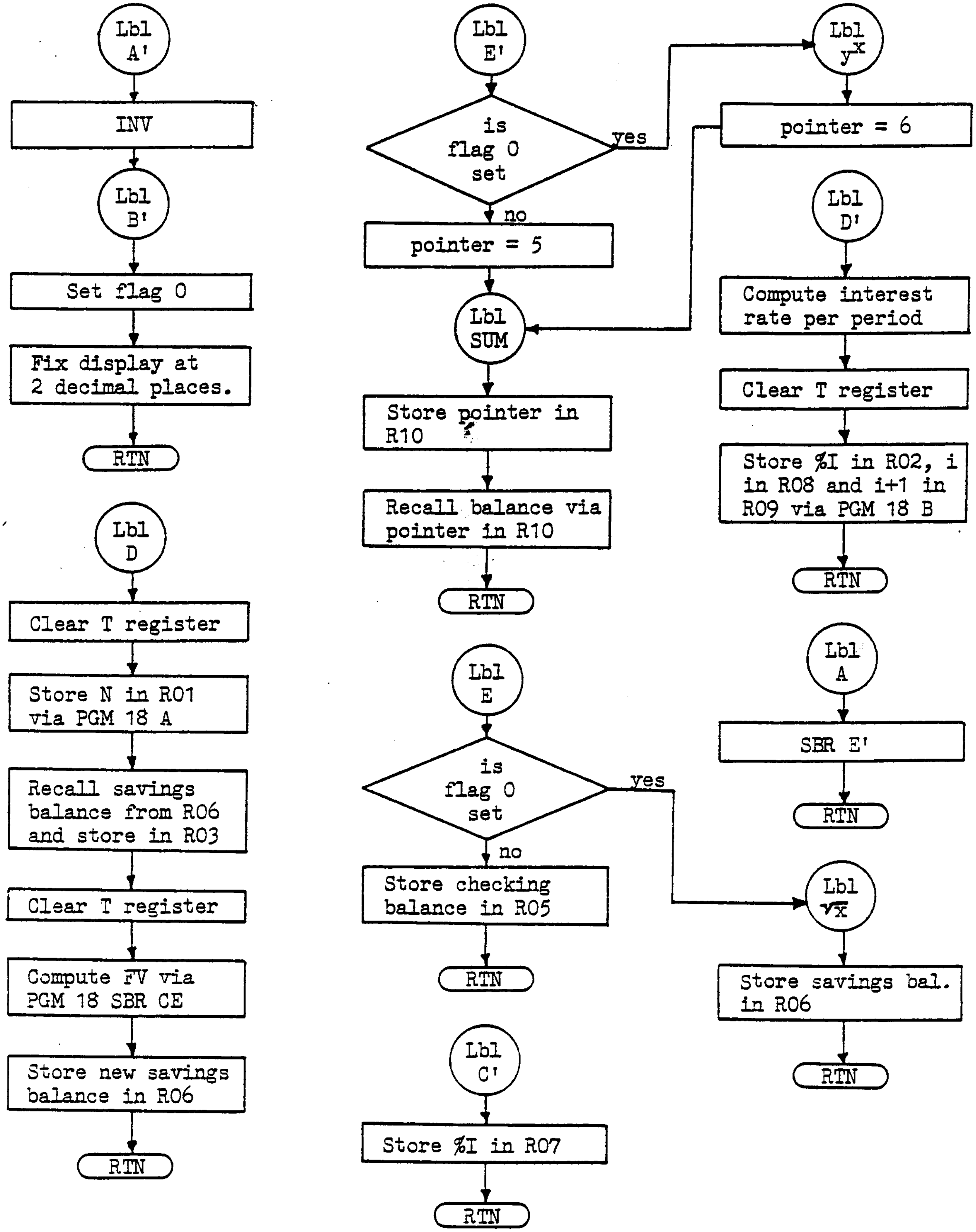
To interface the entire program with another would require that registers 1-10 be reserved (equivalent of 80 program steps) and would need 3 steps each to access 9 user defined keys, for a total of 107 equivalent program steps.

By contrast, the following program is much easier to understand, uses only 71 program steps and 4 registers for an equivalent 103 program steps. It is functionally equivalent to ML-22 with the exception of display results after executing A' or B'. It should be easier to expand or modify for personal needs than to try and interface ML-22 with program memory.

```
000  76  LBL
001  11   A
002  73  RC*
003  04   04
004  92  RTN
005  76  LBL
006  13   C
007  22  INV
008  76  LBL
009  12   B
010  74  SM*
011  04   04
012  73  RC*
013  04   04
014  92  RTN
015  76  LBL
016  14   D
017  53   (
018  32  X:T
019  43  RCL
020  01   01
021  45  YX
022  32  X:T
023  54   )
024  49  PRD
025  02   02
026  43  RCL
027  02   02
028  92  RTN
029  76  LBL
030  15   E
031  72  ST*
032  04   04
033  58  FIX
034  02   02
035  92  RTN
036  76  LBL
037  16   A'
038  03   3
039  42  STO
040  04   04
041  92  RTN
042  76  LBL
043  17   B'
044  02   2
045  42  STO
046  04   04
047  92  RTN
048  76  LBL
049  18   C'
050  42  STO
051  01   01
052  92  RTN
053  76  LBL
054  19   D'
055  22  INV
056  49  PRD
057  01   01
058  43  RCL
059  01   01
060  32  X:T
061  01   1
062  00   0
063  00   0
064  22  INV
065  49  PRD
066  01   01
067  32  X:T
068  69  OP
069  21   21
070  92  RTN
```

Special application:

PGM 22 E will store the input in R06 if flag 0 is set or R05 if it is not set.

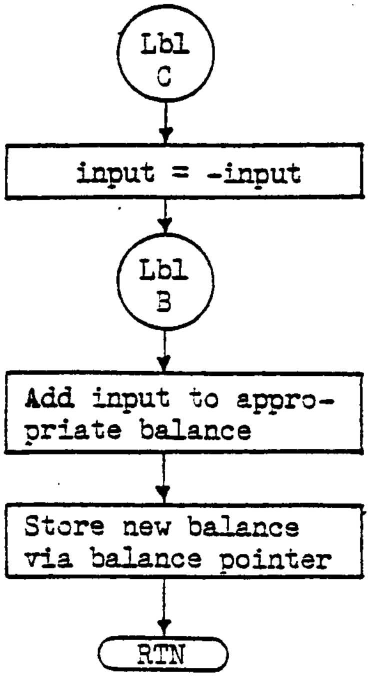

Lbl A'
INV
Lbl B'
Set flag 0
Fix display at 2 decimal places.
RTN
Lbl D
Clear T register
Store N in R01 via PGM 18 A
Recall savings balance from R06 and store in R03
Clear T register
Compute FV via PGM 18 SBR CE
Store new savings balance in R06
RTN
Lbl E'
is flag 0 set
yes
no
pointer = 5
Lbl SUM
Store pointer in R10
Recall balance via pointer in R10
RTN
Lbl E
is flag 0 set
yes
no
Store checking balance in R05
RTN
Lbl C'
Store %I in R07
RTN
Lbl y^x
pointer = 6
Lbl D'
Compute interest rate per period
Clear T register
Store %I in R02, i in R08 and i+1 in R09 via PGM 18 B
RTN
Lbl A
SBR E'
RTN
Lbl $\sqrt{x}$
Store savings bal. in R06
RTN

ML-22 Program Listing

```
000  76 LBL
001  10 E'
002  87 IFF
003  00  00
004  45 YX
005  05  5
006  61 GTO
007  44 SUM
008  76 LBL
009  45 YX
010  06  6
011  76 LBL
012  44 SUM
013  42 STO
014  10  10
015  73 RC*
016  10  10
017  92 RTN
018  76 LBL
019  13 C
020  94 +/-
021  76 LBL
022  12 B
023  53  (
024  24 CE
025  85  +
026  10 E'
027  54  )
028  72 ST*
029  10  10
030  92 RTN
031  76 LBL
032  11 A
033  10 E'
034  92 RTN
035  76 LBL
036  16 A'
037  22 INV
038  76 LBL
039  17 B'
040  86 STF
041  00  00
042  58 FIX
043  02  02
044  92 RTN
045  76 LBL
046  18 C'
047  42 STO
048  07  07
049  92 RTN
```

```
050  76 LBL
051  19 D'
052  53  (
053  35 1/X
054  65  x
055  43 RCL
056  07  07
057  54  )
058  29 CP
059  36 PGM
060  18  18
061  12  B
062  92 RTN
063  76 LBL
064  14 D
065  29 CP
066  36 PGM
067  18  18
068  11  A
069  43 RCL
070  06  06
071  42 STO
072  03  03
073  29 CP
074  36 PGM
075  18  18
076  71 SBR
077  24 CE
078  42 STO
079  06  06
080  92 RTN
081  76 LBL
082  15 E
083  87 IFF
084  00  00
085  34 √X
086  42 STO
087  05  05
088  92 RTN
089  76 LBL
090  34 √X
091  42 STO
092  06  06
093  92 RTN
```

```
001  10 E'
009  45 YX
012  44 SUM
019  13  C
022  12  B
032  11  A
036  16 A'
039  17 B'
046  18 C'
051  19 D'
064  14  D
082  15  E
090  34 √X
```


Lbl C
input = -input
Lbl B
Add input to appropriate balance
Store new balance via balance pointer
RTN

ML-23

DMS OPERATIONS

ML-23 does simple arithmetic operations (+,-,x,÷) on numbers in dd.mmss format. Since the program structure is so simple, no flowchart is needed and explanation of the coding appears with the program listing.

Normal use data:

Registers used: R01
Parentheses levels: 2 (contrary to M.L.M. Appendix A)
Subroutine levels: none (contrary to M.L.M. Appendix A)

Interface procedure:

Simply follow the user instructions in the M.L.M. and precede each user defined key with "PGM 23".

Special notes:

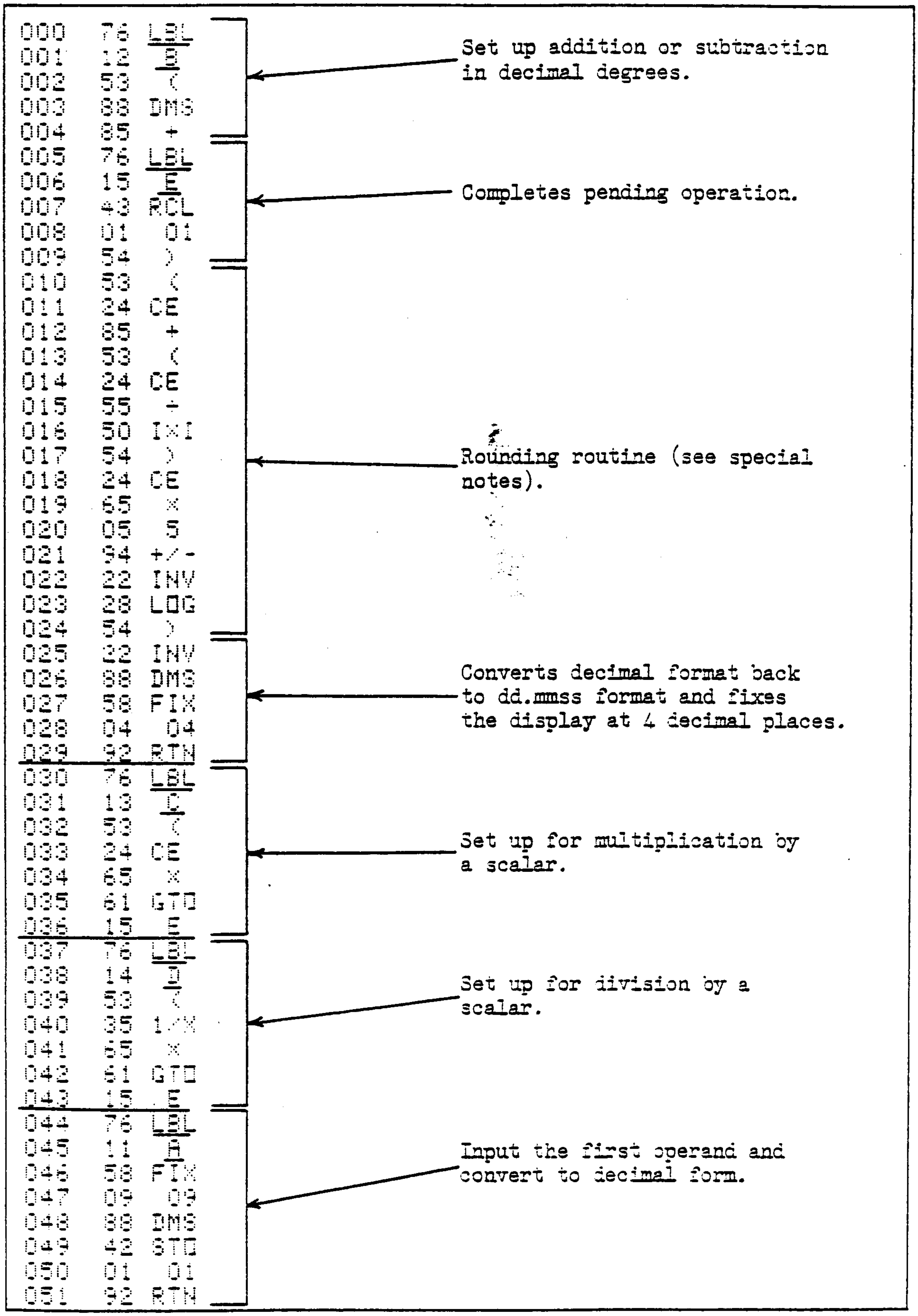
(1) ML-23 leaves the display in fix 4 format.

(2) The "rounding" portion of the program actually only increases the magnitude of the decimal form of the answer by .00001 before conversion back to dd.mmss format. It is apparently intended to keep the display from showing either 60 minutes or 60 seconds.

For example, download PGM 23 with "PGM 23 OP 09" and delete the "rounding" routine, steps 010-024. After hitting RST to make sure you are using the modified program, run example 1 in the M.L.M. and note that the displayed result 11.1960 instead of 11.2000 as with the original version.

This is not fail-safe though. Using ML-23 according to the user instructions, divide 16.3958 by 50 and note that the result is .1960.

ML-23 Program Listing



```
000  76  LBL
001  12   B
002  53   (
003  88  DMS
004  85   +
```

Set up addition or subtraction in decimal degrees.

```
005  76  LBL
006  15   E
007  43  RCL
008  01   01
009  54   )
```

Completes pending operation.

```
010  53   (
011  24  CE
012  85   +
013  53   (
014  24  CE
015  55   ÷
016  50  I×I
017  54   )
018  24  CE
019  65   ×
020  05   5
021  94  +/-
022  22  INV
023  28  LOG
024  54   )
```

Rounding routine (see special notes).

```
025  22  INV
026  88  DMS
027  58  FIX
028  04   04
029  92  RTN
```

Converts decimal format back to dd.mmss format and fixes the display at 4 decimal places.

```
030  76  LBL
031  13   C
032  53   (
033  24  CE
034  65   ×
035  61  GTO
036  15   E
```

Set up for multiplication by a scalar.

```
037  76  LBL
038  14   D
039  53   (
040  35  1/X
041  65   ×
042  61  GTO
043  15   E
```

Set up for division by a scalar.

```
044  76  LBL
045  11   A
046  58  FIX
047  09   09
048  88  DMS
049  42  STO
050  01   01
051  92  RTN
```

Input the first operand and convert to decimal form.

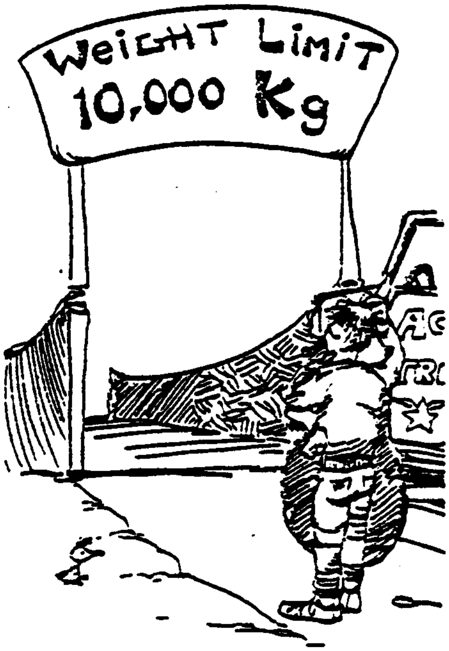
ML-24 , ML-25

UNIT CONVERSIONS

Both ML-24 and ML-25 are essentially mechanizations of the general formula:

$y = kx$ where k is some conversion constant

Note that if $y = kx$ then $1/x = (k)(1/y)$. Thus the same routine can be used for conversions in both directions by taking the reciprocal of inputs and outputs.



Normal use data:

Parentheses levels: 2
Subroutine levels: 1

Interface procedure:

Simply follow the user instructions in the M.L.M. and precede each user defined key with "PGM 24" or "PGM 25".

Special note:

The GTO A which ends PGM 25 has no functional usage in the program. It is apparently there to keep a R/S after executing label E' from trying to execute past step 123.

ML-24 Program Listing

```
000   76  LBL
001   11    A
002   53    (
003   24   CE
004   65    x
005   02    2
006   93    .
007   05    5
008   04    4
009   54    )
010   92  RTN
011   76  LBL
012   12    B
013   53    (
014   24   CE
015   65    x
016   93    .
017   03    3
018   00    0
019   04    4
020   08    8
021   54    )
022   92  RTN
023   76  LBL
024   13    C
025   53    (
026   24   CE
027   65    x
028   93    .
029   09    9
030   01    1
031   04    4
032   04    4
033   54    )
034   92  RTN
035   76  LBL
036   14    D
037   53    (
038   24   CE
039   65    x
040   01    1
041   93    .
042   06    6
043   00    0
044   09    9
045   03    3
046   04    4
047   04    4
048   54    )
049   92  RTN
```

```
050   76  LBL
051   15    E
052   53    (
053   24   CE
054   65    x
055   93    .
056   08    8
057   06    6
058   08    8
059   09    9
060   07    7
061   06    6
062   02    2
063   04    4
064   54    )
065   92  RTN
066   76  LBL
067   16   A'
068   35  1/X
069   11    A
070   35  1/X
071   92  RTN
072   76  LBL
073   17   B'
074   35  1/X
075   12    B
076   35  1/X
077   92  RTN
078   76  LBL
079   18   C'
080   35  1/X
081   13    C
082   35  1/X
083   92  RTN
084   76  LBL
085   19   D'
086   35  1/X
087   14    D
088   35  1/X
089   92  RTN
090   76  LBL
091   10   E'
092   35  1/X
093   15    E
094   35  1/X
095   92  RTN
```

```
001   11    A
012   12    B
024   13    C
036   14    D
051   15    E
067   16   A'
073   17   B'
079   18   C'
085   19   D'
091   10   E'
```

ML-25 Program Listing

```
000  76  LBL
001  11   A
002  53   (
003  53   (
004  24  CE
005  75   -
006  03   3
007  02   2
008  54   )
009  65   x
010  05   5
011  55   ÷
012  09   9
013  54   )
014  92  RTN
015  76  LBL
016  12   B
017  53   (
018  24  CE
019  65   x
020  93   .
021  00   0
022  02   2
023  09   9
024  05   5
025  07   7
026  03   3
027  05   5
028  02   2
029  09   9
030  06   6
031  54   )
032  92  RTN
033  76  LBL
034  13   C
035  53   (
036  24  CE
037  65   x
038  03   3
039  93   .
040  07   7
041  08   8
042  05   5
043  04   4
044  01   1
045  01   1
046  07   7
047  08   8
048  04   4
049  54   )

050  92  RTN
051  76  LBL
052  14   D
053  53   (
054  24  CE
055  65   x
056  02   2
057  08   8
058  93   .
059  03   3
060  04   4
061  09   9
062  05   5
063  02   2
064  03   3
065  01   1
066  03   3
067  54   )
068  92  RTN
069  76  LBL
070  15   E
071  53   (
072  24  CE
073  65   x
074  93   .
075  04   4
076  05   5
077  03   3
078  05   5
079  09   9
080  02   2
081  03   3
082  07   7
083  54   )
084  92  RTN
085  76  LBL
086  16   A'
087  53   (
088  24  CE
089  65   x
090  01   1
091  93   .
092  08   8
093  85   +
094  03   3
095  02   2
096  54   )
097  92  RTN
098  76  LBL
099  17   B'

100  35  1/X
101  12   B
102  35  1/X
103  92  RTN
104  76  LBL
105  18   C'
106  35  1/X
107  13   C
108  35  1/X
109  92  RTN
110  76  LBL
111  19   D'
112  35  1/X
113  14   D
114  35  1/X
115  92  RTN
116  76  LBL
117  10   E'
118  35  1/X
119  15   E
120  35  1/X
121  92  RTN
122  61  GTO
123  11   A

001  11   A
016  12   B
034  13   C
052  14   D
070  15   E
086  16   A'
099  17   B'
105  18   C'
111  19   D'
117  10   E'
```

APPENDIX A

REGISTERS VS. PROGRAM MEMORY

The owner's manual is very brief in its discussion of the tradeoff between program memory and data registers. In actuality, each register corresponds to a specific eight steps in program memory. For example, with the 59, R70 corresponds to steps 392-399. This relationship is fixed. When the 58/59 partitioning is set what you are actually doing is instructing the calculator to treat a certain section of its total memory as data registers and the rest as program memory.

First let's consider what happens if you only need 24 data registers and no more for a specific program. This means that to use say R00-R23 you must repartition for a minimum of 30 data registers, which if you have a 59, leaves you with 720 program steps, 000-719. The following discussion is concerned with the 59 but applicable to the 58 keeping its memory size in mind. Note from the included table for 59 register vs. program memory assignments that R00-R23 correspond to program steps 768-959. This means that you are wasting 48 program steps, 720-767. Now if you only have a 500 step program to begin with, you probably will not be too excited about the waste of 48 steps. But what if you've done everything you can think of to try and cram a 750 step program into a 720 step partition with the exception of pulling out what is left of your hair by this time? Well, there is a sneaky way to get around this problem and get an "effective" partition of 767.23. T.I. is very careful not to point out that repartitioning is possible under program control as well as from the keyboard. When you need to access R20-R23 you must be below step 719. Put 3 OP 17 into your program to repartition to 719.29. Now you can use R20-R23. Note however, that the portion of your program which now "resides" in R24-R29 (steps 720-767) is now vulnerable to overwrite by memory operations so be careful where you put your data! Your program is also restricted to operating in steps 000-719. When you need to run the portion of your program above step 719 but below step 768 (where R23 starts) then use 2 OP 17 to repartition to 799.19. But note that R20-R23 are now inaccessible as data registers and any data they might contain now forms a "program" in steps 768-799 which may give some very strange results if you try to run it!

Incidentally, it is a good idea to record all your programs on the 59's magnetic cards in the default partition of 479.59 and let the program repartition itself to the proper mix. This eliminates the problem of trying to read a card in one partition that was recorded in another, or the need to manually repartition before reading. Don't forget though to have it repartition back to 479.59 when it's through running.

Now that we've introduced the idea that numbers put into data registers can be seen as program after repartitioning, and vice versa, let's take a closer look at the details. The mechanics are easiest explained with a specific example:

(1) Again, assuming a 59, partition to 159.99 with 10 OP 17.
(2) Multiply pi by 1 X 10 to the -15 and store the result in R99.
(3) Repartition to 479.59 with 6 OP 17 and examine or list steps 160-167. They should look like:

```
160:  54
161:  01
162:  59
163:  53          (you've actually created 3
164:  26           pseudos at this point but
165:  59           that's a different story
166:  41           ...see Appendix B)
167:  31
```

String these all together starting from the bottom to get 3141592653590154. It may or may not be obvious that the first 13 digits are the number pi. The next two are the exponent. The last is a sign digit which takes on the values:

Mantissa	Exponent	Sign digit
+	+	0
-	+	2
+	-	4
-	-	6

In general, for a block of eight steps representing a data register, if we assign the following letters to each step:

OP
MN
KL
IJ
GH
EF
CD
AB

The number in the register is:

$$A.BCDEFGHIJKLM \text{ X } 10^{NO}$$

with the signs determined by P as previously given.

Note however, that when you store a number in a data register, position P can only be 0, 2, 4, or 6. If you put a two digit instruction code in OP and position P is not not one of the allowed digits, then when you try to recall the "number" from the correponding data register, 1, 3, 5, or 7 transfers to the display register as 0, 2, 4, or 6 respectively. An 8 or 9 for digit P sets an overflow error state. There is one more complication to be aware of. If there are any leading zeros in the sequence ABCDEFG... then the display register shifts the digits to the left until A is non-zero.

TI-59 Register vs. program memory assignments

160	R99	208	R93	256	R87	304	R81	352	R75	400	R69
161	R99	209	R93	257	R87	305	R81	353	R75	401	R69
162	R99	210	R93	258	R87	306	R81	354	R75	402	R69
163	R99	211	R93	259	R87	307	R81	355	R75	403	R69
164	R99	212	R93	260	R87	308	R81	356	R75	404	R69
165	R99	213	R93	261	R87	309	R81	357	R75	405	R69
166	R99	214	R93	262	R87	310	R81	358	R75	406	R69
167	R99	215	R93	263	R87	311	R81	359	R75	407	R69
168	R98	216	R92	264	R86	312	R80	360	R74	408	R68
169	R98	217	R92	265	R86	313	R80	361	R74	409	R68
170	R98	218	R92	266	R86	314	R80	362	R74	410	R68
171	R98	219	R92	267	R86	315	R80	363	R74	411	R68
172	R98	220	R92	268	R86	316	R80	364	R74	412	R68
173	R98	221	R92	269	R86	317	R80	365	R74	413	R68
174	R98	222	R92	270	R86	318	R80	366	R74	414	R68
175	R98	223	R92	271	R86	319	R80	367	R74	415	R68
176	R97	224	R91	272	R85	320	R79	368	R73	416	R67
177	R97	225	R91	273	R85	321	R79	369	R73	417	R67
178	R97	226	R91	274	R85	322	R79	370	R73	418	R67
179	R97	227	R91	275	R85	323	R79	371	R73	419	R67
180	R97	228	R91	276	R85	324	R79	372	R73	420	R67
181	R97	229	R91	277	R85	325	R79	373	R73	421	R67
182	R97	230	R91	278	R85	326	R79	374	R73	422	R67
183	R97	231	R91	279	R85	327	R79	375	R73	423	R67
184	R96	232	R90	280	R84	328	R78	376	R72	424	R66
185	R96	233	R90	281	R84	329	R78	377	R72	425	R66
186	R96	234	R90	282	R84	330	R78	378	R72	426	R66
187	R96	235	R90	283	R84	331	R78	379	R72	427	R66
188	R96	236	R90	284	R84	332	R78	380	R72	428	R66
189	R96	237	R90	285	R84	333	R78	381	R72	429	R66
190	R96	238	R90	286	R84	334	R78	382	R72	430	R66
191	R96	239	R90	287	R84	335	R78	383	R72	431	R66
192	R95	240	R89	288	R83	336	R77	384	R71	432	R65
193	R95	241	R89	289	R83	337	R77	385	R71	433	R65
194	R95	242	R89	290	R83	338	R77	386	R71	434	R65
195	R95	243	R89	291	R83	339	R77	387	R71	435	R65
196	R95	244	R89	292	R83	340	R77	388	R71	436	R65
197	R95	245	R89	293	R83	341	R77	389	R71	437	R65
198	R95	246	R89	294	R83	342	R77	390	R71	438	R65
199	R95	247	R89	295	R83	343	R77	391	R71	439	R65
200	R94	248	R88	296	R82	344	R76	392	R70	440	R64
201	R94	249	R88	297	R82	345	R76	393	R70	441	R64
202	R94	250	R88	298	R82	346	R76	394	R70	442	R64
203	R94	251	R88	299	R82	347	R76	395	R70	443	R64
204	R94	252	R88	300	R82	348	R76	396	R70	444	R64
205	R94	253	R88	301	R82	349	R76	397	R70	445	R64
206	R94	254	R88	302	R82	350	R76	398	R70	446	R64
207	R94	255	R88	303	R82	351	R76	399	R70	447	R64

TI-59 Register vs. program memory assignments (cont.)

448	R63	496	R57	544	R51	592	R45	640	R39	688	R33
449	R63	497	R57	545	R51	593	R45	641	R39	689	R33
450	R63	498	R57	546	R51	594	R45	642	R39	690	R33
451	R63	499	R57	547	R51	595	R45	643	R39	691	R33
452	R63	500	R57	548	R51	596	R45	644	R39	692	R33
453	R63	501	R57	549	R51	597	R45	645	R39	693	R33
454	R63	502	R57	550	R51	598	R45	646	R39	694	R33
455	R63	503	R57	551	R51	599	R45	647	R39	695	R33
456	R62	504	R56	552	R50	600	R44	648	R38	696	R32
457	R62	505	R56	553	R50	601	R44	649	R38	697	R32
458	R62	506	R56	554	R50	602	R44	650	R38	698	R32
459	R62	507	R56	555	R50	603	R44	651	R38	699	R32
460	R62	508	R56	556	R50	604	R44	652	R38	700	R32
461	R62	509	R56	557	R50	605	R44	653	R38	701	R32
462	R62	510	R56	558	R50	606	R44	654	R38	702	R32
463	R62	511	R56	559	R50	607	R44	655	R38	703	R32
464	R61	512	R55	560	R49	608	R43	656	R37	704	R31
465	R61	513	R55	561	R49	609	R43	657	R37	705	R31
466	R61	514	R55	562	R49	610	R43	658	R37	706	R31
467	R61	515	R55	563	R49	611	R43	659	R37	707	R31
468	R61	516	R55	564	R49	612	R43	660	R37	708	R31
469	R61	517	R55	565	R49	613	R43	661	R37	709	R31
470	R61	518	R55	566	R49	614	R43	662	R37	710	R31
471	R61	519	R55	567	R49	615	R43	663	R37	711	R31
472	R60	520	R54	568	R48	616	R42	664	R36	712	R30
473	R60	521	R54	569	R48	617	R42	665	R36	713	R30
474	R60	522	R54	570	R48	618	R42	666	R36	714	R30
475	R60	523	R54	571	R48	619	R42	667	R36	715	R30
476	R60	524	R54	572	R48	620	R42	668	R36	716	R30
477	R60	525	R54	573	R48	621	R42	669	R36	717	R30
478	R60	526	R54	574	R48	622	R42	670	R36	718	R30
479	R60	527	R54	575	R48	623	R42	671	R36	719	R30
480	R59	528	R53	576	R47	624	R41	672	R35	720	R29
481	R59	529	R53	577	R47	625	R41	673	R35	721	R29
482	R59	530	R53	578	R47	626	R41	674	R35	722	R29
483	R59	531	R53	579	R47	627	R41	675	R35	723	R29
484	R59	532	R53	580	R47	628	R41	676	R35	724	R29
485	R59	533	R53	581	R47	629	R41	677	R35	725	R29
486	R59	534	R53	582	R47	630	R41	678	R35	726	R29
487	R59	535	R53	583	R47	631	R41	679	R35	727	R29
488	R58	536	R52	584	R46	632	R40	680	R34	728	R28
489	R58	537	R52	585	R46	633	R40	681	R34	729	R28
490	R58	538	R52	586	R46	634	R40	682	R34	730	R28
491	R58	539	R52	587	R46	635	R40	683	R34	731	R28
492	R58	540	R52	588	R46	636	R40	684	R34	732	R28
493	R58	541	R52	589	R46	637	R40	685	R34	733	R28
494	R58	542	R52	590	R46	638	R40	686	R34	734	R28
495	R58	543	R52	591	R46	639	R40	687	R34	735	R28

TI-59 Register vs. program memory assignments (cont.)

736	R27	784	R21	832	R15	880	R09	928	R03
737	R27	785	R21	833	R15	881	R09	929	R03
738	R27	786	R21	834	R15	882	R09	930	R03
739	R27	787	R21	835	R15	883	R09	931	R03
740	R27	788	R21	836	R15	884	R09	932	R03
741	R27	789	R21	837	R15	885	R09	933	R03
742	R27	790	R21	838	R15	886	R09	934	R03
743	R27	791	R21	839	R15	887	R09	935	R03
744	R26	792	R20	840	R14	888	R08	936	R02
745	R26	793	R20	841	R14	889	R08	937	R02
746	R26	794	R20	842	R14	890	R08	938	R02
747	R26	795	R20	843	R14	891	R08	939	R02
748	R26	796	R20	844	R14	892	R08	940	R02
749	R26	797	R20	845	R14	893	R08	941	R02
750	R26	798	R20	846	R14	894	R08	942	R02
751	R26	799	R20	847	R14	895	R08	943	R02
752	R25	800	R19	848	R13	896	R07	944	R01
753	R25	801	R19	849	R13	897	R07	945	R01
754	R25	802	R19	850	R13	898	R07	946	R01
755	R25	803	R19	851	R13	899	R07	947	R01
756	R25	804	R19	852	R13	900	R07	948	R01
757	R25	805	R19	853	R13	901	R07	949	R01
758	R25	806	R19	854	R13	902	R07	950	R01
759	R25	807	R19	855	R13	903	R07	951	R01
760	R24	808	R18	856	R12	904	R06	952	R00
761	R24	809	R18	857	R12	905	R06	953	R00
762	R24	810	R18	858	R12	906	R06	954	R00
763	R24	811	R18	859	R12	907	R06	955	R00
764	R24	812	R18	860	R12	908	R06	956	R00
765	R24	813	R18	861	R12	909	R06	957	R00
766	R24	814	R18	862	R12	910	R06	958	R00
767	R24	815	R18	863	R12	911	R06	959	R00
768	R23	816	R17	864	R11	912	R05		
769	R23	817	R17	865	R11	913	R05		
770	R23	818	R17	866	R11	914	R05		
771	R23	819	R17	867	R11	915	R05		
772	R23	820	R17	868	R11	916	R05		
773	R23	821	R17	869	R11	917	R05		
774	R23	822	R17	870	R11	918	R05		
775	R23	823	R17	871	R11	919	R05		
776	R22	824	R16	872	R10	920	R04		
777	R22	825	R16	873	R10	921	R04		
778	R22	826	R16	874	R10	922	R04		
779	R22	827	R16	875	R10	923	R04		
780	R22	828	R16	876	R10	924	R04		
781	R22	829	R16	877	R10	925	R04		
782	R22	830	R16	878	R10	926	R04		
783	R22	831	R16	879	R10	927	R04		

TI-58 Register vs program memory assignments

000	R59	048	R53	096	R47	144	R41	192	R35	240	R29
001	R59	049	R53	097	R47	145	R41	193	R35	241	R29
002	R59	050	R53	098	R47	146	R41	194	R35	242	R29
003	R59	051	R53	099	R47	147	R41	195	R35	243	R29
004	R59	052	R53	100	R47	148	R41	196	R35	244	R29
005	R59	053	R53	101	R47	149	R41	197	R35	245	R29
006	R59	054	R53	102	R47	150	R41	198	R35	246	R29
007	R59	055	R53	103	R47	151	R41	199	R35	247	R29
008	R58	056	R52	104	R46	152	R40	200	R34	248	R28
009	R58	057	R52	105	R46	153	R40	201	R34	249	R28
010	R58	058	R52	106	R46	154	R40	202	R34	250	R28
011	R58	059	R52	107	R46	155	R40	203	R34	251	R28
012	R58	060	R52	108	R46	156	R40	204	R34	252	R28
013	R58	061	R52	109	R46	157	R40	205	R34	253	R28
014	R58	062	R52	110	R46	158	R40	206	R34	254	R28
015	R58	063	R52	111	R46	159	R40	207	R34	255	R28
016	R57	064	R51	112	R45	160	R39	208	R33	256	R27
017	R57	065	R51	113	R45	161	R39	209	R33	257	R27
018	R57	066	R51	114	R45	162	R39	210	R33	258	R27
019	R57	067	R51	115	R45	163	R39	211	R33	259	R27
020	R57	068	R51	116	R45	164	R39	212	R33	260	R27
021	R57	069	R51	117	R45	165	R39	213	R33	261	R27
022	R57	070	R51	118	R45	166	R39	214	R33	262	R27
023	R57	071	R51	119	R45	167	R39	215	R33	263	R27
024	R56	072	R50	120	R44	168	R38	216	R32	264	R26
025	R56	073	R50	121	R44	169	R38	217	R32	265	R26
026	R56	074	R50	122	R44	170	R38	218	R32	266	R26
027	R56	075	R50	123	R44	171	R38	219	R32	267	R26
028	R56	076	R50	124	R44	172	R38	220	R32	268	R26
029	R56	077	R50	125	R44	173	R38	221	R32	269	R26
030	R56	078	R50	126	R44	174	R38	222	R32	270	R26
031	R56	079	R50	127	R44	175	R38	223	R32	271	R26
032	R55	080	R49	128	R43	176	R37	224	R31	272	R25
033	R55	081	R49	129	R43	177	R37	225	R31	273	R25
034	R55	082	R49	130	R43	178	R37	226	R31	274	R25
035	R55	083	R49	131	R43	179	R37	227	R31	275	R25
036	R55	084	R49	132	R43	180	R37	228	R31	276	R25
037	R55	085	R49	133	R43	181	R37	229	R31	277	R25
038	R55	086	R49	134	R43	182	R37	230	R31	278	R25
039	R55	087	R49	135	R43	183	R37	231	R31	279	R25
040	R54	088	R48	136	R42	184	R36	232	R30	280	R24
041	R54	089	R48	137	R42	185	R36	233	R30	281	R24
042	R54	090	R48	138	R42	186	R36	234	R30	282	R24
043	R54	091	R48	139	R42	187	R36	235	R30	283	R24
044	R54	092	R48	140	R42	188	R36	236	R30	284	R24
045	R54	093	R48	141	R42	189	R36	237	R30	285	R24
046	R54	094	R48	142	R42	190	R36	238	R30	286	R24
047	R54	095	R48	143	R42	191	R36	239	R30	287	R24

TI-58 Register vs program memory assignments (cont.)

288	R23	336	R17	384	R11	432	R05
289	R23	337	R17	385	R11	433	R05
290	R23	338	R17	386	R11	434	R05
291	R23	339	R17	387	R11	435	R05
292	R23	340	R17	388	R11	436	R05
293	R23	341	R17	389	R11	437	R05
294	R23	342	R17	390	R11	438	R05
295	R23	343	R17	391	R11	439	R05
296	R22	344	R16	392	R10	440	R04
297	R22	345	R16	393	R10	441	R04
298	R22	346	R16	394	R10	442	R04
299	R22	347	R16	395	R10	443	R04
300	R22	348	R16	396	R10	444	R04
301	R22	349	R16	397	R10	445	R04
302	R22	350	R16	398	R10	446	R04
303	R22	351	R16	399	R10	447	R04
304	R21	352	R15	400	R09	448	R03
305	R21	353	R15	401	R09	449	R03
306	R21	354	R15	402	R09	450	R03
307	R21	355	R15	403	R09	451	R03
308	R21	356	R15	404	R09	452	R03
309	R21	357	R15	405	R09	453	R03
310	R21	358	R15	406	R09	454	R03
311	R21	359	R15	407	R09	455	R03
312	R20	360	R14	408	R08	456	R02
313	R20	361	R14	409	R08	457	R02
314	R20	362	R14	410	R08	458	R02
315	R20	363	R14	411	R08	459	R02
316	R20	364	R14	412	R08	460	R02
317	R20	365	R14	413	R08	461	R02
318	R20	366	R14	414	R08	462	R02
319	R20	367	R14	415	R08	463	R02
320	R19	368	R13	416	R07	464	R01
321	R19	369	R13	417	R07	465	R01
322	R19	370	R13	418	R07	466	R01
323	R19	371	R13	419	R07	467	R01
324	R19	372	R13	420	R07	468	R01
325	R19	373	R13	421	R07	469	R01
326	R19	374	R13	422	R07	470	R01
327	R19	375	R13	423	R07	471	R01
328	R18	376	R12	424	R06	472	R00
329	R18	377	R12	425	R06	473	R00
330	R18	378	R12	426	R06	474	R00
331	R18	379	R12	427	R06	475	R00
332	R18	380	R12	428	R06	476	R00
333	R18	381	R12	429	R06	477	R00
334	R18	382	R12	430	R06	478	R00
335	R18	383	R12	431	R06	479	R00

APPENDIX B

PSEUDOS

Out of a possible 100 two-digit instruction codes, the 58/59 has 92 which are acknowledged by T.I. as valid. (These are the codes for the keyboard functions, not to be confused with the 40 special op codes which may follow an OP keystroke.) The remaining 8 codes have been dubbed "pseudos" and exploration of their functions is still a major frontier.[1] These codes are 21, 26, 31, 41, 46, 51, 56, and 82.

Creating pseudos:

Pseudos may be synthesized in program memory via the merging feature of the 58/59. For example, to create P31, in LRN mode key STO 31 then delete the STO.

The printer assigns a certain mnemonic to each pseudo as appears to the right.

```
000   21 2ND
001   26 2ND
002   31 LRN
003   41 SST
004   46 INS
005   51 BST
006   56 DEL
007   82 HIR
```

P82: HIR:

P82 is a very useful pseudo which gives access to 8 internal registers dubbed HIR1, HIR2, etc. The sequence HIR MN operates on a HIR register as follows:

"N" is the HIR register to be accessed, 1-8.
"M" is the operation desired according to the list:

STO:	0	PROD:	4
RCL:	1	INV SUM:	5
SUM:	3	INV PROD:	6,7,8,or9

Note that MN may be synthesized the same way that the pseudo was or if the reader is familiar with the instruction code versus keystroke list, the particular keystroke that gives the desired MN may be used.

The HIR registers are normally used for:

(1) Nested arithmetic operations:
First operand goes into HIR1, second into HIR2, etc.

(2) P/R and INV P/R:
Uses first two available and HIR7 and HIR8.

(3) D.MS and INV D.MS:
Uses first two available and HIR8.

[1] The author uses the terminology adopted by 52 Notes. (see Forward)

(4) Σ+ and Σ- :
Uses HIR7 and HIR8.

(5) x̄:
Uses first available HIR.

(6) OP codes:
OP 11: First two available
OP 12: First three available
OP 13: First four available
OP 14: First three available and HIR8
OP 15: First three available

HIR registers are not affected by CLR or CMs. HIRS 5-8 can be cleared by OP 00 if the 58/59 is connected to a PC-100A.

<u>P31: LRN</u>:

P31 will cause the calculator to go into LRN mode when encountered during program execution. It is a very useful prompt for adding function subroutines, for example, those like are needed by ML-08 or ML-09.

<u>P21; 2ND</u>:

If P21 is followed by SIN, COS, or TAN the calculator "crashes" (will not respond to keyboard commands to halt.)

<u>P26, P41, P46, P51, and P56</u>:

Not much is known about these pseudos at this time. The author would be very interested in hearing about any significant results from exploration by the reader in this area.

A CLEAR CUT CASE OF ABUSE CAPTAIN....
SOME BODY GAVE HIM THE FINGER!