

PRG1 RING. PRG

```

124 OTHERWISE: RETURN Stack$[nY+2]
125 ENDCASE
126 END
127
128
129 * Унарная операция
130 COMMON DEF Unary Op$
131 VAR Tmp$
132 IF VAR("0:ERROR")==0 THEN
133   X1$=COPY(Stack$[nX])
134   Tmp$=CALL(Op$,Stack$[nX])
135   IF VAR("0:ERROR")==0 THEN Stack$[nX]=Tmp$
136 ENDIF
137 END
138
139
140 * Двухместная операция
141 COMMON DEF Binary Op$
142 VAR Tmp$
143 IF VAR("0:ERROR")==0 THEN
144   X1$=COPY(Stack$[nX])
145   Tmp$=CALL(Op$,Stack$[nX],Stack$[nY])
146   IF VAR("0:ERROR")==0 THEN
147     nX=nY:Stack$[nX]=Tmp$
148     IF nX<999 THEN nY=nX+1 ELSE nY=0
149   ENDIF
150 ENDIF
151 END
152
153 * Поднятие стека перед замещением X
154 COMMON DEF StackUpFast
155 nY=nX:IF nX>0 THEN DEC nX ELSE nX=999
156 END
157
158
159 * Поднятие стека
160 COMMON DEF StackUp
161 IF VAR("0:ERROR")==0 THEN
162   StackUpFast
163   Stack$[nX]=COPY(Stack$[nY])
164 ENDIF
165 END
166
167
168 * Опустить стек
169 * - можно выкинуть строку в X из памяти, если на неё нет больше ссылок
170 COMMON DEF StackDown
171 nX=nY:IF nY<999 THEN INC nY ELSE nY=0
172 END
173
174
175 * Кольцевое передвижение информации в стеке
176 COMMON DEF StackRot
177 IF VAR("0:ERROR")==0 THEN
178   VAR nZ,nT
179   CASE nX
180     WHEN 999: nZ=1:nT=2
181     WHEN 998: nZ=0:nT=1
182     WHEN 997: nZ=999:nT=0
183     OTHERWISE: nZ=nX+2:nT=nX+3
184   ENDCASE
185   X1$=Stack$[nX]
186   Stack$[nX]=Stack$[nY]
187   Stack$[nY]=Stack$[nZ]
188   Stack$[nZ]=Stack$[nT]
189   Stack$[nT]=COPY(X1$)
190 ENDIF
191 END
192
193
194
195 * Ввести число пи
196 COMMON DEF LoadPi
197 IF VAR("0:ERROR")==0 THEN StackUpFast:Stack$[nX]=#FPI
198 END
199
200
201 * Считать X1
202 COMMON DEF LoadX1
203 IF VAR("0:ERROR")==0 THEN StackUpFast:Stack$[nX]=COPY(X1$)
204 END
205
206
207 * Обменять содержимое регистров X и Y, сохранив X в X1
208 COMMON DEF StackSwap
209 IF VAR("0:ERROR")==0 THEN X1$=COPY(Stack$[nX]):SWAP Stack$[nX],Stack$[nY]
210 END
211
212
213 * Абсолютное значение
214 COMMON DEF DoAbs
215 IF VAR("0:ERROR")==0 THEN
216   X1$=COPY(Stack$[nX])
217   IF Stack$[nX][0]="#" THEN
218     IF Stack$[nX][1]>"7" THEN GexNot Stack$[nX]:GexInc Stack$[nX]
219     LandRX
220   ELSEIF Stack$[nX][0]="#" THEN
221     Stack$[nX]=ToDec$(ABS(VAL(MID$(Stack$[nX],1))))
222   ELSEIF Stack$[nX][0]="#" THEN
223     Stack$[nX][0]="#"
224   ENDIF
225 ENDIF
226 END
227
228
229 * Определение знака числа
230 COMMON DEF DoSgn
231 IF VAR("0:ERROR")==0 THEN
232   X1$=Stack$[nX]
233   CASE X1$[0]
234     WHEN "-" : Stack$[nX]=#ONEN
235     WHEN "#":
236       IF X1$=#GZERO THEN
237         Stack$[nX]=#ZERO
238       ELSEIF X1$[1]>"7" THEN
239         Stack$[nX]=#ONEN
240       ELSE
241         Stack$[nX]=#ONE
242       ENDIF
243     WHEN "$":
244       VAR Z=VAL(MID$(X1$,1))
245       IF Z==0 THEN
246         Stack$[nX]=#ZERO
247       ELSEIF Z<0 THEN
248         Stack$[nX]=#ONEN

```



```

247 ELSEIF Z<0 THEN
248   Stack$[nX]=#ONEN
249 ELSE
250   Stack$[nX]=#ONE
251 ENDIF
252 OTHERWISE:
253 IF X1$=#ZERO THEN
254   Stack$[nX]=#ZERO
255 ELSEIF X1$[0]!="#" THEN
256   Stack$[nX]=#ONE
257 ENDIF
258 ENDCASE
259 ENDIF
260 END
261
262 'Выделение целой части
263 COMMON DEF DoInt
264 IF VAR("0:ERROR")==0 THEN
265   X1$=COPY(Stack$[nX])
266 CASE X1$[0]
267 WHEN "#": LandRX
268 WHEN "$": Stack$[nX]=ToDec$(INT(VAL(MID$(X1$,1))))
269 OTHERWISE:
270   VAR I=0: GetDigit(X1$[16])*10+GetDigit(X1$[17])
271   IF X1$[16]!="-" THEN
272     Stack$[nX]=#ZERO
273   ELSEIF I<13 THEN
274     FOR I=0+2 TO 14: Stack$[nX][I]="0": NEXT
275   ENDIF
276   ENDIF
277 ENDIF
278 END
279
280 'Выделение дробной части
281 COMMON DEF DoFrac
282 IF VAR("0:ERROR")==0 THEN
283   VAR P
284   X1$=COPY(Stack$[nX])
285 CASE X1$[0]
286 WHEN "#": Stack$[nX]=#ZERO
287 WHEN "$":
288   P=VAL(MID$(X1$,1))
289   Stack$[nX]=ToDec$(P-INT(P))
290 OTHERWISE:
291   IF X1$[16]!="-" THEN
292     P=GetDigit(X1$[16])*10+GetDigit(X1$[17])
293     IF P<13 THEN
294       VAR I=P+2: A$=#ZERO, J=1, F=0
295       FOR I=1 TO 14
296         IF X1$[I]!="0" THEN F=1
297         IF F==1 THEN
298           A$[J]=X1$[I]: INC J
299         ELSE
300           INC P1
301         ENDIF
302       NEXT
303       IF F==1 THEN 'Если дробная часть отлична от нуля
304         A$[16]="-"
305         A$[17]=STR$(P1 DIV 10)
306         A$[18]=STR$(P1 MOD 10)
307       ENDIF
308       Stack$[nX]=A$
309     ENDIF
310   ENDIF
311   ENDIF
312   ENDIF
313   ENDIF
314   ENDIF
315   ENDIF
316   ENDIF
317   ENDIF
318   ENDIF
319   ENDIF
320   ENDIF
321   ENDIF
322   ENDIF
323   ENDIF
324   ENDIF
325   ENDIF
326   ENDIF
327   ENDIF
328   ENDIF
329   ENDIF
330   ENDIF
331   ENDIF
332   ENDIF
333   ENDIF
334   ENDIF
335   ENDIF
336   ENDIF
337   ENDIF
338   ENDIF
339   ENDIF
340   ENDIF
341   ENDIF
342   ENDIF
343   ENDIF
344   ENDIF
345   ENDIF
346   ENDIF
347   ENDIF
348   ENDIF
349   ENDIF
350   ENDIF
351   ENDIF
352   ENDIF
353   ENDIF
354   ENDIF
355   ENDIF
356   ENDIF
357   ENDIF
358   ENDIF
359   ENDIF
360   ENDIF
361   ENDIF
362   ENDIF
363   ENDIF
364   ENDIF
365   ENDIF
366   ENDIF
367   ENDIF
368   ENDIF
369   ENDIF
370   ENDIF
371   ENDIF
372   ENDIF
373   ENDIF
374   ENDIF
375   ENDIF
376   ENDIF
377   ENDIF
378   ENDIF
379   ENDIF
380   ENDIF
381   ENDIF
382   ENDIF
383   ENDIF
384   ENDIF
385   ENDIF
386   ENDIF
387   ENDIF
388   ENDIF
389   ENDIF
390   ENDIF
391   ENDIF
392   ENDIF
393   ENDIF
394   ENDIF
395   ENDIF
396   ENDIF
397   ENDIF
398   ENDIF
399   ENDIF
400   ENDIF
401   ENDIF
402   ENDIF
403   ENDIF
404   ENDIF
405   ENDIF
406   ENDIF
407   ENDIF
408   ENDIF
409   ENDIF
410   ENDIF
411   ENDIF
412   ENDIF
413   ENDIF
414   ENDIF
415   ENDIF
416   ENDIF
417   ENDIF
418   ENDIF
419   ENDIF
420   ENDIF
421   ENDIF
422   ENDIF
423   ENDIF
424   ENDIF
425   ENDIF
426   ENDIF
427   ENDIF
428   ENDIF
429   ENDIF
430   ENDIF
431   ENDIF
432   ENDIF
433   ENDIF
434   ENDIF
435   ENDIF
436   ENDIF
437   ENDIF
438   ENDIF
439   ENDIF
440   ENDIF
441   ENDIF
442   ENDIF
443   ENDIF
444   ENDIF
445   ENDIF
446   ENDIF
447   ENDIF
448   ENDIF
449   ENDIF
450   ENDIF
451   ENDIF
452   ENDIF
453   ENDIF
454   ENDIF
455   ENDIF
456   ENDIF
457   ENDIF
458   ENDIF
459   ENDIF
460   ENDIF
461   ENDIF
462   ENDIF
463   ENDIF
464   ENDIF
465   ENDIF
466   ENDIF
467   ENDIF
468   ENDIF
469   ENDIF
470   ENDIF
471   ENDIF
472   ENDIF
473   ENDIF
474   ENDIF
475   ENDIF
476   ENDIF
477   ENDIF
478   ENDIF
479   ENDIF
480   ENDIF
481   ENDIF
482   ENDIF
483   ENDIF
484   ENDIF
485   ENDIF
486   ENDIF
487   ENDIF
488   ENDIF
489   ENDIF
490   ENDIF
491   ENDIF
492   ENDIF
493   ENDIF
494   ENDIF
495   ENDIF
496   ENDIF
497   ENDIF
498   ENDIF
499   ENDIF
500   ENDIF
501   ENDIF
502   ENDIF
503   ENDIF
504   ENDIF
505   ENDIF
506   ENDIF
507   ENDIF
508   ENDIF
509   ENDIF
510   ENDIF
511   ENDIF
512   ENDIF
513   ENDIF
514   ENDIF
515   ENDIF
516   ENDIF
517   ENDIF
518   ENDIF
519   ENDIF
520   ENDIF
521   ENDIF
522   ENDIF
523   ENDIF
524   ENDIF
525   ENDIF
526   ENDIF
527   ENDIF
528   ENDIF
529   ENDIF
530   ENDIF
531   ENDIF
532   ENDIF
533   ENDIF
534   ENDIF
535   ENDIF
536   ENDIF
537   ENDIF
538   ENDIF
539   ENDIF
540   ENDIF
541   ENDIF
542   ENDIF
543   ENDIF
544   ENDIF
545   ENDIF
546   ENDIF
547   ENDIF
548   ENDIF
549   ENDIF
550   ENDIF
551   ENDIF
552   ENDIF
553   ENDIF
554   ENDIF
555   ENDIF
556   ENDIF
557   ENDIF
558   ENDIF
559   ENDIF
560   ENDIF
561   ENDIF
562   ENDIF
563   ENDIF
564   ENDIF
565   ENDIF
566   ENDIF
567   ENDIF
568   ENDIF
569   ENDIF
570   ENDIF
571   ENDIF
572   ENDIF
573   ENDIF
574   ENDIF
575   ENDIF
576   ENDIF
577   ENDIF
578   ENDIF
579   ENDIF
580   ENDIF
581   ENDIF
582   ENDIF
583   ENDIF
584   ENDIF
585   ENDIF
586   ENDIF
587   ENDIF
588   ENDIF
589   ENDIF
590   ENDIF
591   ENDIF
592   ENDIF
593   ENDIF
594   ENDIF
595   ENDIF
596   ENDIF
597   ENDIF
598   ENDIF
599   ENDIF
600   ENDIF
601   ENDIF
602   ENDIF
603   ENDIF
604   ENDIF
605   ENDIF
606   ENDIF
607   ENDIF
608   ENDIF
609   ENDIF
610   ENDIF
611   ENDIF
612   ENDIF
613   ENDIF
614   ENDIF
615   ENDIF
616   ENDIF
617   ENDIF
618   ENDIF
619   ENDIF
620   ENDIF
621   ENDIF
622   ENDIF
623   ENDIF
624   ENDIF
625   ENDIF
626   ENDIF
627   ENDIF
628   ENDIF
629   ENDIF
630   ENDIF
631   ENDIF
632   ENDIF
633   ENDIF
634   ENDIF
635   ENDIF
636   ENDIF
637   ENDIF
638   ENDIF
639   ENDIF
640   ENDIF
641   ENDIF
642   ENDIF
643   ENDIF
644   ENDIF
645   ENDIF
646   ENDIF
647   ENDIF
648   ENDIF
649   ENDIF
650   ENDIF
651   ENDIF
652   ENDIF
653   ENDIF
654   ENDIF
655   ENDIF
656   ENDIF
657   ENDIF
658   ENDIF
659   ENDIF
660   ENDIF
661   ENDIF
662   ENDIF
663   ENDIF
664   ENDIF
665   ENDIF
666   ENDIF
667   ENDIF
668   ENDIF
669   ENDIF
670   ENDIF
671   ENDIF
672   ENDIF
673   ENDIF
674   ENDIF
675   ENDIF
676   ENDIF
677   ENDIF
678   ENDIF
679   ENDIF
680   ENDIF
681   ENDIF
682   ENDIF
683   ENDIF
684   ENDIF
685   ENDIF
686   ENDIF
687   ENDIF
688   ENDIF
689   ENDIF
690   ENDIF
691   ENDIF
692   ENDIF
693   ENDIF
694   ENDIF
695   ENDIF
696   ENDIF
697   ENDIF
698   ENDIF
699   ENDIF
700   ENDIF
701   ENDIF
702   ENDIF
703   ENDIF
704   ENDIF
705   ENDIF
706   ENDIF
707   ENDIF
708   ENDIF
709   ENDIF
710   ENDIF
711   ENDIF
712   ENDIF
713   ENDIF
714   ENDIF
715   ENDIF
716   ENDIF
717   ENDIF
718   ENDIF
719   ENDIF
720   ENDIF
721   ENDIF
722   ENDIF
723   ENDIF
724   ENDIF
725   ENDIF
726   ENDIF
727   ENDIF
728   ENDIF
729   ENDIF
730   ENDIF
731   ENDIF
732   ENDIF
733   ENDIF
734   ENDIF
735   ENDIF
736   ENDIF
737   ENDIF
738   ENDIF
739   ENDIF
740   ENDIF
741   ENDIF
742   ENDIF
743   ENDIF
744   ENDIF
745   ENDIF
746   ENDIF
747   ENDIF
748   ENDIF
749   ENDIF
750   ENDIF
751   ENDIF
752   ENDIF
753   ENDIF
754   ENDIF
755   ENDIF
756   ENDIF
757   ENDIF
758   ENDIF
759   ENDIF
760   ENDIF
761   ENDIF
762   ENDIF
763   ENDIF
764   ENDIF
765   ENDIF
766   ENDIF
767   ENDIF
768   ENDIF
769   ENDIF
770   ENDIF
771   ENDIF
772   ENDIF
773   ENDIF
774   ENDIF
775   ENDIF
776   ENDIF
777   ENDIF
778   ENDIF
779   ENDIF
780   ENDIF
781   ENDIF
782   ENDIF
783   ENDIF
784   ENDIF
785   ENDIF
786   ENDIF
787   ENDIF
788   ENDIF
789   ENDIF
790   ENDIF
791   ENDIF
792   ENDIF
793   ENDIF
794   ENDIF
795   ENDIF
796   ENDIF
797   ENDIF
798   ENDIF
799   ENDIF
800   ENDIF
801   ENDIF
802   ENDIF
803   ENDIF
804   ENDIF
805   ENDIF
806   ENDIF
807   ENDIF
808   ENDIF
809   ENDIF
810   ENDIF
811   ENDIF
812   ENDIF
813   ENDIF
814   ENDIF
815   ENDIF
816   ENDIF
817   ENDIF
818   ENDIF
819   ENDIF
820   ENDIF
821   ENDIF
822   ENDIF
823   ENDIF
824   ENDIF
825   ENDIF
826   ENDIF
827   ENDIF
828   ENDIF
829   ENDIF
830   ENDIF
831   ENDIF
832   ENDIF
833   ENDIF
834   ENDIF
835   ENDIF
836   ENDIF
837   ENDIF
838   ENDIF
839   ENDIF
840   ENDIF
841   ENDIF
842   ENDIF
843   ENDIF
844   ENDIF
845   ENDIF
846   ENDIF
847   ENDIF
848   ENDIF
849   ENDIF
850   ENDIF
851   ENDIF
852   ENDIF
853   ENDIF
854   ENDIF
855   ENDIF
856   ENDIF
857   ENDIF
858   ENDIF
859   ENDIF
860   ENDIF
861   ENDIF
862   ENDIF
863   ENDIF
864   ENDIF
865   ENDIF
866   ENDIF
867   ENDIF
868   ENDIF
869   ENDIF
870   ENDIF
871   ENDIF
872   ENDIF
873   ENDIF
874   ENDIF
875   ENDIF
876   ENDIF
877   ENDIF
878   ENDIF
879   ENDIF
880   ENDIF
881   ENDIF
882   ENDIF
883   ENDIF
884   ENDIF
885   ENDIF
886   ENDIF
887   ENDIF
888   ENDIF
889   ENDIF
890   ENDIF
891   ENDIF
892   ENDIF
893   ENDIF
894   ENDIF
895   ENDIF
896   ENDIF
897   ENDIF
898   ENDIF
899   ENDIF
900   ENDIF
901   ENDIF
902   ENDIF
903   ENDIF
904   ENDIF
905   ENDIF
906   ENDIF
907   ENDIF
908   ENDIF
909   ENDIF
910   ENDIF
911   ENDIF
912   ENDIF
913   ENDIF
914   ENDIF
915   ENDIF
916   ENDIF
917   ENDIF
918   ENDIF
919   ENDIF
920   ENDIF
921   ENDIF
922   ENDIF
923   ENDIF
924   ENDIF
925   ENDIF
926   ENDIF
927   ENDIF
928   ENDIF
929   ENDIF
930   ENDIF
931   ENDIF
932   ENDIF
933   ENDIF
934   ENDIF
935   ENDIF
936   ENDIF
937   ENDIF
938   ENDIF
939   ENDIF
940   ENDIF
941   ENDIF
942   ENDIF
943   ENDIF
944   ENDIF
945   ENDIF
946   ENDIF
947   ENDIF
948   ENDIF
949   ENDIF
950   ENDIF
951   ENDIF
952   ENDIF
953   ENDIF
954   ENDIF
955   ENDIF
956   ENDIF
957   ENDIF
958   ENDIF
959   ENDIF
960   ENDIF
961   ENDIF
962   ENDIF
963   ENDIF
964   ENDIF
965   ENDIF
966   ENDIF
967   ENDIF
968   ENDIF
969   ENDIF
970   ENDIF
971   ENDIF
972   ENDIF
973   ENDIF
974   ENDIF
975   ENDIF
976   ENDIF
977   ENDIF
978   ENDIF
979   ENDIF
980   ENDIF
981   ENDIF
982   ENDIF
983   ENDIF
984   ENDIF
985   ENDIF
986   ENDIF
987   ENDIF
988   ENDIF
989   ENDIF
990   ENDIF
991   ENDIF
992   ENDIF
993   ENDIF
994   ENDIF
995   ENDIF
996   ENDIF
997   ENDIF
998   ENDIF
999   ENDIF
1000  ENDIF

```

```

000370 IF VAR("0:ERROR")==0 THEN
000371 X1$=COPY(Stack$[nX])
000372 Stack$[nX]=ToGex$(Stack$[nX])
000373 GexNot Stack$[nX]
000374 ENDIF
000375 END
000376
000377
000378 ; Генерация случайного числа от 0 до 1
000379 ; - добавить семя
000380 COMMON DEF DoRnd
000381 IF VAR("0:ERROR")==0 THEN
000382 StackUpFast
000383 Stack$[nX]=ToDec$(RNDP())
000384 ENDIF
000385 END
000386
000387 ; Преобразовать не-цифру в "0"
000388 DEF ToDigit$(A$)
000389 IF A$>="0" && A$<="9" THEN RETURN A$
000390 RETURN "0"
000391 END
000392
000393
000394 ; Получить значение цифры
000395 COMMON DEF GetDigit(A$)
000396 IF A$>="0" && A$<="9" THEN RETURN VAL(A$)
000397 IF A$>="A" && A$<="F" THEN RETURN ASC(A$)-ASC("A")+10
000398 IF A$>="a" && A$<="f" THEN RETURN ASC(A$)-ASC("a")+10
000399 RETURN 0
000400 END
000401
000402 ; Проверка, является ли содержимое RX отрицательным
000403 ; - нужно ли проверять знак вводимого числа?
000404 COMMON DEF IsNegative(*)
000405 VAR A$
000406 IF DEFARGC()==0 THEN A$=Stack$[nX] ELSE A$=DEFARG(0)
000407 CASE A$[0]
000408 WHEN "#": RETURN A$[1]>"7"
000409 WHEN "$": RETURN VAL(MID$(A$,1))<0
000410 OTHERWISE: RETURN A$[0]!="0"
000411 END
000412
000413 ; Проверка, записан ли в RX ноль
000414 COMMON DEF IsZero(*)
000415 VAR A$
000416 IF DEFARGC()==0 THEN A$=Stack$[nX] ELSE A$=DEFARG(0)
000417 CASE A$[0]
000418 WHEN "#": RETURN A$=#GZERO
000419 WHEN "$": RETURN VAL(MID$(A$,1))=0
000420 OTHERWISE: RETURN A$=#ZERO
000421 END
000422
000423 ; Преобразовать десятичное число в гекс
000424 ; - каждая тетрада BCD-числа преобразуется независимо
000425 ; - большие положительные в ffff:ffff
000426 ; - большие отрицательные в ffff:ffff
000427 ; - добавить отрицательные целые в гекс
000428 COMMON DEF ToGex$(A$)
000429 VAR G$
000430 IF A$[0]!="0" THEN RETURN A$ ; Если уже гекс, не трогать
000431 IF A$[0]!="$": THEN A$=ToDec$(VAL(MID$(A$,1)))
000432 N=GetDigit(A$[17])+1
000433 IF A$[0]!="-" THEN RETURN #GZERO
000434 IF A$[16]!="0" THEN RETURN #GMAXN ELSE RETURN #GMAX
000435 ENDIF
000436 G$=#GZERO
000437 FOR I=1 TO N: G$[8-N+I]=A$[I]: NEXT I
000438 RETURN G$
000439 END
000440
000441 ; Преобразовать целое в гекс
000442 COMMON DEF MkGex$(N)
000443 VAR G$=#GZERO, I=8
000444 WHILE N<0: INC N, #LIMIT: WEND
000445 WHILE N!=0
000446 G$[I]=HEX$(N AND 15): DEC I
000447 N=INT(N/16)
000448 WEND
000449 RETURN G$
000450 END
000451
000452 ; Считать гекс, как двоичное целое со знаком
000453 COMMON DEF ReadGex(A$)
000454 VAR I, G=GetDigit(A$[1])
000455 FOR I=2 TO 8: G=G*16+GetDigit(A$[I]): NEXT I
000456 IF A$[1]>"7" THEN DEC G, #LIMIT
000457 RETURN G
000458 END
000459
000460 ; Преобразовать десятичное число в вещественное
000461 COMMON DEF ToReal(A$)
000462 VAR I, B$=ToDigit$(A$[1])+"." C$
000463 FOR I=2 TO 14: PUSH B$, ToDigit$(A$[I]): NEXT I
000464 IF A$[15]!="." THEN PUSH B$, "E" ELSE PUSH B$, "E-"
000465 PUSH B$, ToDigit$(A$[16])+ToDigit$(A$[17])
000466 IF A$[0]!="0" THEN RETURN VAL(B$) ELSE RETURN -VAL(B$)
000467 END
000468
000469 ; Преобразовать в десятичное число
000470 ; - продумать отрицательные гексы
000471 COMMON DEF Land(A$)
000472 CASE A$[0]
000473 WHEN "#":
000474 VAR I, G=GetDigit(A$[1])
000475 FOR I=2 TO 8: G=G*10+GetDigit(A$[I]): NEXT I
000476 RETURN G
000477 WHEN "$": RETURN VAL(MID$(A$,1))
000478 WHEN "G": WHEN "M": WHEN "S":
000479 A$[0]="0"
000480 WHEN "g": WHEN "m": WHEN "s":
000481 A$[0]="0"
000482 OTHERWISE: RETURN ToReal(A$)
000483 END
000484
000485
000486
000487
000488
000489
000490
000491
000492
000493
000494

```



```

493 END CASE
494 RETURN ToReal(A$)
495 END
496
497
498 'убедиться, что в RX число
499 COMMON DEF LandRX
500 VAR S$=Stack$(nX)[0]
501 IF S$="#." OR S$="$" THEN Stack$(nX)=ToDec$(Land(Stack$(nX)))
502 END
503
504
505 'преобразовать число в десятичное
506 COMMON DEF ToDec$(A)
507 VAR B$=#ZERO, N$=STR$(ABS(A))
508 VAR I,L=LEN(N$),E
509 IF A<0 THEN B$[0]="-"
510 CASE TYPEOF(A)
511 WHEN #T_INT: 'Преобразуем целое
512 FOR I=1 TO L: B$[I]=N$[I-1]: NEXT I
513 B$[17]=#HEX$(L-1)
514 RETURN B$
515 WHEN #T_REAL: 'Преобразуем вещественное
516 A=ABS(A)
517 IF A<1E-99 THEN RETURN #ZERO 'Машинный ноль
518 IF A<1 THEN
519 E=INT(CEIL(-LOG(A,10))) 'Двоичная математика может дать ошибку
520 A=A*POW(10,E)
521 N$=FORMAT$("%.13F",A)
522 IF A>10 THEN
523 DEC E: N$[2]=N$[1] 'Считаем, что ошибка только на один разряд
524 '12.34 → 12234 (1.234)
525 ELSEIF A<1 THEN
526 INC E: N$=MID$(N$,1): N$[0]=N$[1] '0.123 → 1123 (1.23)
527 ENDIF
528 IF E>0 THEN B$[15]="-"
529 B$[16]=#HEX$(E DIV 10)
530 B$[17]=#HEX$(E MOD 10) 'Считаем, что порядок меньше 100
531 ELSEIF A=10 THEN
532 E=INT(LOG(A,10)) 'Двоичная математика может дать ошибку
533 N$=FORMAT$("%.13F",A)
534 IF A>10 THEN
535 INC E: N$[2]=N$[1] 'Считаем, что ошибка только на один разряд
536 '12.34 → 12234 (1.234)
537 ELSEIF A<1 THEN
538 DEC E: N$=MID$(N$,1): N$[0]=N$[1] '0.123 → 1123 (1.23)
539 ENDIF
540 B$[16]=#HEX$(E DIV 10)
541 B$[17]=#HEX$(E MOD 10) 'Считаем, что порядок меньше 100
542 ELSE
543 N$=FORMAT$("%.13F",A) 'Числа от 1 до 10, не включая 10
544 ENDIF
545 L=LEN(N$)
546 B$[1]=N$[0]
547 IF L>14 THEN L=15
548 FOR I=2 TO L-1: B$[I]=N$[I]: NEXT I
549 RETURN B$
550 WHEN #T_STR:
551 RETURN A 'Возможно, на входе уже десятичное
552 OTHERWISE:
553 RETURN #ZERO 'По умолчанию ноль
554 END CASE
555 END
556
557 'преобразовать V# в целое и ограничить диапазоном [MI,MA]
558 COMMON DEF CAP(V#,MI,MA)
559 IF V#<MI THEN
560 RETURN MI
561 ELSEIF V#>MA THEN
562 RETURN MA
563 ELSE
564 RETURN INT(V#)
565 ENDIF
566 END
567
568
569 'Считать угол формата M, перевести в градусы
570 '— игнорирует знак мантиссы
571 DEF GetM$(A$)
572 VAR E=VAL(A$[16]+A$[17]), sE$=A$[15]
573 IF sE$="." THEN 'Отрицательный порядок
574 RETURN VAL(A$[1]+". "+MID$(A$,2,13))*POW(10,-E)/6
575 ELSEIF E>12 THEN
576 RETURN VAL(A$[1]+". "+MID$(A$,2,13))*POW(10,E)
577 ELSE
578 VAR rM#=VAL(A$[2+E]+". "+MID$(A$,3+E,12-E))/6
579 RETURN VAL(MID$(A$,1,E+1))+rM#
580 ENDIF
581 END
582
583
584 'Считать угол формата MC, перевести в градусы
585 '— игнорирует знак мантиссы
586 '— не проверено
587 DEF GetS$(A$)
588 VAR E=VAL(A$[16]+A$[17]), sE$=A$[15], iM, rS#
589 IF sE$="." THEN 'Отрицательный порядок
590 CASE E
591 WHEN 1: 'mmccc
592 iM=VAL(A$[1]+A$[2])
593 rS#=VAL(A$[3]+A$[4]+". "+MID$(A$,5,10))/6
594 RETURN (iM*60+rS#)/360
595 WHEN 2: 'mcccc
596 iM=VAL(A$[1])
597 rS#=VAL(A$[2]+A$[3]+". "+MID$(A$,4,11))/6
598 RETURN (iM*60+rS#)/360
599 OTHERWISE: 'cccc
600 RETURN VAL(A$[1]+". "+MID$(A$,2,13))*POW(10,2-E)/36
601 END CASE
602 ELSEIF E>12 THEN 'ffff
603 RETURN VAL(A$[1]+". "+MID$(A$,2,13))*POW(10,E)
604 ELSE
605 CASE E
606 WHEN 12: RETURN VAL(MID$(A$,1,13))+VAL(A$[14])/6
607 WHEN 11: RETURN VAL(MID$(A$,1,12))+VAL(A$[13]+A$[14])/60 '::ffffm
608 OTHERWISE: 'fmmccc
609 iM=VAL(A$[2+E]+A$[3+E])
610 rS#=VAL(A$[4+E]+". "+MID$(A$,5+E,10-E))/6
611 RETURN VAL(MID$(A$,1,E+1))+iM*6+rS#
612 END CASE
613 ENDIF
614 END
615
616
617 'Привести аргумент к радианам

```

```

615 *Привести аргумент к радианам
616 COMMON DEF ToRad(A$)
617 CASE A$[0]
618 WHEN "G": WHEN "g":
619 WHEN "T": RETURN RAD(Land(A$))
620 WHEN "S": RETURN RAD(GetM$(A$))
621 WHEN "M": RETURN RAD(GetS$(A$))
622 WHEN "m": RETURN RAD(GetM$(A$))
623 WHEN "s": RETURN -RAD(GetS$(A$))
624 ENDCASE
625 RETURN Land(A$)
626 END
627
628 *Привести радианы к градусной мере
629 COMMON DEF ComRad(V#)
630 CASE Rf045
631 WHEN 0: RETURN DEG(V#)
632 WHEN 1: RETURN V#
633 WHEN 2: RETURN DEG(V#)/0.9
634 ENDCASE
635 END
636
637 *Память программ
638 *Очистить память программ, позже -- восстановить из энергонезависимой
639 *Очищает также память программ цели
640 *Самостоятельно вызывает StrClear
641 COMMON DEF PrgClear
642 FILL PrgM, -1
643 StrClear
644 END
645
646 *Считать ячейку памяти программ
647 *из несуществующих адресов считывается ffff:ffff
648 COMMON DEF ReadPrgM
649 IF VAR(0:ERROR)=0 THEN
650 VAR A$=ReadGex(ToGex$(Stack$(nX)))
651 X1$=Stack$(nX)
652 IF A<0: IF A>65535 THEN A=-1 ELSE A=PrgM[A]
653 Stack$(nX)=MkGex$(A)
654 ENDIF
655 END
656
657 *Регистровые операции
658 *Очистить регистры, позже -- восстановить из энергонезависимой памяти
659 COMMON DEF RegClear
660 FILL R$, #ZERO
661 END
662
663 *Записать регистр X в регистр функций N
664 COMMON DEF STOF N
665 CASE N
666 WHEN &hf045: Rf045=CAP(ToReal(Stack$(nX)),0,2)
667 ENDCASE
668 END
669
670 *Запись содержимого регистра X в регистр N
671 COMMON DEF TSTO N
672 IF N>=0 THEN
673 IF N>=&hf000 && N<&h10000 THEN
674 STOF N
675 ELSEIF N<&h10000 THEN
676 R$(N)=COPY(Stack$(nX))
677 ELSEIF N<&h20000 THEN
678 PRGM[N]=ReadGex(ToGex$(Stack$(nX)))
679 ENDIF
680 ENDIF
681 END
682
683 *Косвенная запись содержимого регистра X
684 *по содержимому адресного регистра N
685 *проверить диапазон N
686 COMMON DEF KSTO N
687 VAR A$=ToGex$(R$(N))
688 R$(N)=A$
689 STO ReadGex(A$)
690 END
691
692 *Косвенная запись содержимого регистра X, N от 0 до 15
693 COMMON DEF KSTO1 N
694 VAR A$=ToGex$(R$(N))
695 CASE N
696 WHEN 0: WHEN 1: WHEN 2: WHEN 3: GexDec A$
697 WHEN 4: WHEN 5: WHEN 6: GexInc A$
698 ENDCASE
699 R$(N)=A$
700 STO ReadGex(A$)
701 END
702
703 *Вызов в X содержимого регистра функций N
704 *если регистра нет, в стек загружается 0
705 COMMON DEF RCLF N
706 CASE N
707 WHEN &hf045:
708 StackUpFast: Stack$(nX)=ToDec$(Rf045)
709 OTHERWISE:
710 StackUpFast: Stack$(nX)=#ZERO
711 ENDCASE
712 END
713
714 *Вызов в X содержимого регистра R
715 *если регистра нет, в стек загружается 0
716 COMMON DEF RCL N
717 IF N<0: IF N>=&h20000 THEN
718 StackUpFast: Stack$(nX)=#ZERO
719 ELSEIF N>=&hf000 && N<&h10000 THEN
720 RCLF N
721 ELSEIF N<&h10000 THEN
722 StackUpFast: Stack$(nX)=COPY(R$(N))
723 ELSE
724 StackUpFast: Stack$(nX)=ToDec$(PrgM[N])
725 ENDIF
726 END
727
728 *Косвенный вызов в регистр X по содержимому адресного регистра N
729 *проверить диапазон N
730 COMMON DEF KRCL N
731 VAR A$=ToGex$(R$(N))
732 R$(N)=A$
733 RCL ReadGex(A$)
734 END
735
736 *Косвенный вызов в регистр X, N от 0 до 15
737 COMMON DEF KRCL1 N
738 VAR A$=ToGex$(R$(N))
739 CASE N
740 WHEN 0: WHEN 1: WHEN 2: WHEN 3: GexDec A$
741 WHEN 4: WHEN 5: WHEN 6: GexInc A$
742 ENDCASE
743 R$(N)=A$
744 RCL ReadGex(A$)
745 END
746

```