```
'1:RING.PRG       v0.2.0a, 7 августа 2022
'Общие переменые и работа с ними

'Авторское право (C) Васильев Илья Владимирович, г. Москва
'Это свободная программа, защищённая авторским правом
'Распространяется под лицензией GNU GPL v3 или старше

OPTION STRICT 'Все переменные должны быть объявлены
OPTION DEFINT 'Массивы по умолчанию целые

'Формат десятичного числа
'              11111111 смещение (старшая цифра)
'01234567890123456 7 смещение
'-1234567890123456-12
'¦мантисса        ¦порядок
'+-знак числа     +-знак порядка

'                 0123456789012345 67
CONST #ZERO ="  0000000000000000 00"  'Десятичный ноль
CONST #ONE  ="  1000000000000000 00"  'Десятичная единица
CONST #FF   ="  2550000000000000 02"  'Десятичное число 255
CONST #ONEN ="-1000000000000000 00"  'Десятичная минус единица
CONST #FPI  ="  3141592653589800 00"  'Число пи
CONST #GZERO="#0000000000000000 07"  'Гекс 0000:0000
CONST #GMAX ="#7FFFFFFFF0000000 07"  'Гекс 7fff:ffff
CONST #GMAXN="#FFFFFFFFF0000000 07"  'Гекс ffff:ffff

CONST #HEX$ ="0123456789abcdef"
CONST #LIMIT=65536*65536   'Максимальное беззнаковое целое + 1

VAR Stack$[1000]           'Кольцевой стек
VAR nX, nY                 'Адрес вершины стека и ячейки под ней
VAR X1$                    'Регистр предыдущего результата
VAR R$[65536]              'Регистры данных
VAR PrgM[65536*2]          'Память программ: 64K инструмент + 64K цель

'Регистры функций
'VAR Rf045                  '0-ГРАДУСЫ, 1-РАДИАНЫ, 2-ГРАДЫ

'"Двухшаговые" команды с 16-битным номером регистра
DIM tblRefP[]=[&hf4,&hf6,&hf7,&hf8,&hf9,&hfa,&hfb,&hfc,&hfd,&hfe]

'Ссылается ли ячейка P на участок памяти программ?
COMMON DEF IsRefP(P,Pmin,Pmax)
  VAR A=PrgM[P], A1 = A>>>24
  IF (A1 AND &hf7)!=&hf7 THEN RETURN #FALSE
  A1 = (A>>>16) AND 255
  IF A1==255 THEN RETURN #FALSE
  IF FIND(tblRefP,A1)!=-1 THEN RETURN #FALSE
  A1 = A AND 65535
  IF A1==65535 THEN RETURN #FALSE
  RETURN (A1>=Pmin) && (A1<=Pmax)
END

'Найти и изменить ссылки на участок памяти
COMMON DEF Relocate Pstart,Pend,Pmin,Pmax,Pshift
  VAR I, A, B
  IF Pmin<=Pmax THEN
    FOR I=Pstart TO Pend
      IF IsRefP(I,Pmin,Pmax) THEN
        A=PrgM[I]:B=(A AND 65535)+Pshift
        PrgM[I]=(A AND &hffff0000) OR (B AND 65535)
      ENDIF
    NEXT
  ENDIF
END

'Стековые операции
'Очистить стек
COMMON DEF StackClear
  nX=996:nY=997:X1$=#ZERO
'  FILL Stack$,#ZERO,nX,4
  FILL Stack$,#ZERO
  VAR("2:shBody$")=""       'Сбрасываем незаконченный ввод
END

'Очистить RX
COMMON DEF ClearX
  Stack$[nX]=#ZERO
END

'Установить RX
COMMON DEF SetX$ A$
  Stack$[nX]=A$
END

'Установить RY
COMMON DEF SetY$ A$
  Stack$[nY]=A$
END

'Получить RX
COMMON DEF GetX$()
  RETURN Stack$[nX]
END

'Получить RY
COMMON DEF GetY$()
  RETURN Stack$[nY]
END

'Получить RZ
COMMON DEF GetZ$()
  IF nY<999 THEN RETURN Stack$[nY+1] ELSE RETURN Stack$[0]
END

'Получить RT
COMMON DEF GetT$()
  CASE nY
    WHEN 999:  RETURN Stack$[1]
    WHEN 998:  RETURN Stack$[0]
    OTHERWISE: RETURN Stack$[nY+2]
  ENDCASE
```

```
    OTHERWISE: RETURN Stack$[nY+2]
  ENDCASE
END

'Унарная операция
COMMON DEF Unary Op$
  VAR Tmp$
  IF VAR("0:ERROR")==0 THEN
    X1$=COPY(Stack$[nX]) 'На МК-56 и МК-161 X1 обновляется даже при ошибке
    Tmp$=CALL(Op$,Stack$[nX])
    IF VAR("0:ERROR")==0 THEN Stack$[nX]=Tmp$
  ENDIF
END

'Двухместная операция
COMMON DEF Binary Op$
  VAR Tmp$
  IF VAR("0:ERROR")==0 THEN
    X1$=COPY(Stack$[nX]) 'На МК-56 и МК-161 X1 обновляется даже при ошибке
    Tmp$=CALL(Op$,Stack$[nX],Stack$[nY])
    IF VAR("0:ERROR")==0 THEN
      nX=nY:Stack$[nX]=Tmp$
      IF nX<999 THEN nY=nX+1 ELSE nY=0
    ENDIF
  ENDIF
END

'Поднятие стека перед замещением X
COMMON DEF StackUpFast
  nY=nX:IF nX>0 THEN DEC nX ELSE nX=999
END

'Поднятие стека
COMMON DEF StackUp
  IF VAR("0:ERROR")==0 THEN
    StackUpFast
    Stack$[nX]=COPY(Stack$[nY])
  ENDIF
END

'Опустить стек
'- можно выкинуть строку в X из памяти, если на неё нет больше ссылок
COMMON DEF StackDown
  nX=nY:IF nY<999 THEN INC nY ELSE nY=0
END

'Кольцевое передвижение информации в стеке
COMMON DEF StackRot
  IF VAR("0:ERROR")==0 THEN
    VAR nZ, nT
    CASE nX
      WHEN 999: nZ=1:nT=2
      WHEN 998: nZ=0:nT=1
      WHEN 997: nZ=999:nT=0
      OTHERWISE: nZ=nX+2:nT=nX+3
    ENDCASE
    X1$=Stack$[nX]
    Stack$[nX]=Stack$[nY]
    Stack$[nY]=Stack$[nZ]
    Stack$[nZ]=Stack$[nT]
    Stack$[nT]=COPY(X1$)
  ENDIF
END

'Ввести число пи
COMMON DEF LoadPi
  IF VAR("0:ERROR")==0 THEN StackUpFast:Stack$[nX]=#FPI
END

'Считать X1
COMMON DEF LoadX1
  IF VAR("0:ERROR")==0 THEN StackUpFast:Stack$[nX]=COPY(X1$)
END

'Обменять содержимое регистров X и Y, сохранив X в X1
COMMON DEF StackSwap
  IF VAR("0:ERROR")==0 THEN X1$=COPY(Stack$[nX]):SWAP Stack$[nX],Stack$[nY]
END

'Абсолютное значение
COMMON DEF DoAbs
  IF VAR("0:ERROR")==0 THEN
    X1$=COPY(Stack$[nX])
    IF Stack$[nX][0]=="#" THEN
      IF Stack$[nX][1]>"7" THEN GexNot Stack$[nX]:GexInc Stack$[nX]
      LandRX
    ELSEIF Stack$[nX][0]=="$" THEN
      Stack$[nX]=ToDec$(ABS(VAL(MID$(Stack$[nX],1))))
    ELSEIF Stack$[nX][0]=="-" THEN
      Stack$[nX][0]=" "
    ENDIF
  ENDIF
END

'Определение знака числа
COMMON DEF DoSgn
  IF VAR("0:ERROR")==0 THEN
    X1$=Stack$[nX]
    CASE X1$[0]
      WHEN "-": Stack$[nX]=#ONEN
      WHEN "#":
        IF X1$==#GZERO THEN
          Stack$[nX]=#ZERO
        ELSEIF X1$[1]>"7" THEN
          Stack$[nX]=#ONEN
        ELSE
          Stack$[nX]=#ONE
        ENDIF
      WHEN "$":
        VAR Z=VAL(MID$(X1$,1))
        IF Z==0 THEN
          Stack$[nX]=#ZERO
        ELSEIF Z<0 THEN
          Stack$[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 'X указывает на строку, отличную от X1
255          ELSEIF X1$[0]!="#" THEN
256            Stack$[nX]=#ONE
257          ENDIF
258      ENDCASE
259    ENDIF
260  END
261
262
263  'Выделение целой части
264  COMMON DEF DoInt
265    IF VAR("0:ERROR")==0 THEN
266      X1$=COPY(Stack$[nX])
267      CASE X1$[0]
268        WHEN "#": LandRX
269        WHEN "$": Stack$[nX]=ToDec$(INT(VAL(MID$(X1$,1))))
270        OTHERWISE:
271          VAR I,I0=GetDigit(X1$[16])*10+GetDigit(X1$[17])
272          IF X1$[15]=="-" THEN
273            Stack$[nX]=#ZERO
274          ELSEIF I0<13 THEN
275            FOR I=I0+2 TO 14:Stack$[nX][I]="0":NEXT
276          ENDIF 'порядок
277      ENDCASE
278    ENDIF 'Error
279  END
280
281
282  'Выделение дробной части
283  COMMON DEF DoFrac
284    IF VAR("0:ERROR")==0 THEN
285      VAR P
286      X1$=COPY(Stack$[nX])
287      CASE X1$[0]
288        WHEN "#": Stack$[nX]=#ZERO
289        WHEN "$":
290          P=VAL(MID$(X1$,1))
291          Stack$[nX]=ToDec$(P-INT(P))
292        OTHERWISE:
293          IF X1$[15]!="-" THEN
294            P=GetDigit(X1$[16])*10+GetDigit(X1$[17])
295            IF P<13 THEN
296              VAR I,P1=1,A$=#ZERO,J=1,F=0
297              FOR I=P+2 TO 14
298                IF X1$[I]!="0" THEN F=1
299                IF F==1 THEN
300                  A$[J]=X1$[I]:INC J
301                ELSE
302                  INC P1
303                ENDIF
304              NEXT
305              IF F==1 THEN ' Если дробная часть отлична от нуля
306                A$[15]="-"
307                A$[16]=STR$(P1 DIV 10)
308                A$[17]=STR$(P1 MOD 10)
309              ENDIF
310              Stack$[nX]=A$
311            ENDIF ' X<1e13
312          ENDIF 'X>=1
313      ENDCASE
314    ENDIF 'Error
315  END
316
317
318  'Определение максимального значения
319  'Реализована версия из МК-161
320  '- исправить отрицательные гексы
321  COMMON DEF DoMax
322    IF VAR("0:ERROR")==0 THEN
323      X1$=COPY(Stack$[nX])
324      IF Land(Stack$[nY])>Land(X1$) THEN SWAP Stack$[nX],Stack$[nY]
325    ENDIF
326  END
327
328
329  'Логическое умножение
330  COMMON DEF DoAnd
331    IF VAR("0:ERROR")==0 THEN
332      VAR I, A$=ToGex$(Stack$[nX]), B$=ToGex$(Stack$[nY])
333      X1$=Stack$[nX]:StackDown
334      Stack$[nX]=#GZERO
335      FOR I=1 TO 8
336        Stack$[nX][I]=#HEX$[GetDigit(A$[I]) AND GetDigit(B$[I])]
337      NEXT
338    ENDIF
339  END
340
341
342  'Логическое сложение
343  COMMON DEF DoOr
344    IF VAR("0:ERROR")==0 THEN
345      VAR I, A$=ToGex$(Stack$[nX]), B$=ToGex$(Stack$[nY])
346      X1$=Stack$[nX]:StackDown
347      Stack$[nX]=#GZERO
348      FOR I=1 TO 8
349        Stack$[nX][I]=#HEX$[GetDigit(A$[I]) OR GetDigit(B$[I])]
350      NEXT
351    ENDIF
352  END
353
354
355  'Логическое сложение по модулю 2
356  COMMON DEF DoXor
357    IF VAR("0:ERROR")==0 THEN
358      VAR I, A$=ToGex$(Stack$[nX]), B$=ToGex$(Stack$[nY])
359      X1$=Stack$[nX]:StackDown
360      Stack$[nX]=#GZERO
361      FOR I=1 TO 8
362        Stack$[nX][I]=#HEX$[GetDigit(A$[I]) XOR GetDigit(B$[I])]
363      NEXT
364    ENDIF
365  END
366
367
368  'Логическая инверсия
369  COMMON DEF DoNot
370    IF VAR("0:ERROR")==0 THEN
371      X1$=COPY(Stack$[nX])
```

```
370   IF VAR("0:ERROR")==0 THEN
371     X1$=COPY(Stack$[nX])
372     Stack$[nX]=ToGex$(Stack$[nX])
373     GexNot Stack$[nX]
374   ENDIF
375 END
376
377
378 'Генерация случайного числа от 0 до 1
379 '- добавить семя
380 COMMON DEF DoRnd
381   IF VAR("0:ERROR")==0 THEN
382     StackUpFast
383     Stack$[nX]=ToDec$(RNDF())
384   ENDIF
385 END
386
387
388 'Преобразовать не-цифру в "0"
389 DEF ToDigit$(A$)
390   IF A$>="0" && A$<="9" THEN RETURN A$
391   RETURN "0"
392 END
393
394
395 'Получить значение цифры
396 COMMON DEF GetDigit(A$)
397   IF A$>="0" && A$<="9" THEN RETURN VAL(A$)
398   IF A$>="A" && A$<="F" THEN RETURN ASC(A$)-ASC("A")+10
399   IF A$>="a" && A$<="f" THEN RETURN ASC(A$)-ASC("a")+10
400   RETURN 0
401 END
402
403
404 'Проверка, является ли содержимое RX отрицательным
405 '- нужно ли проверять знак вводимого числа?
406 COMMON DEF IsNegative(*)
407   VAR A$
408   IF DEFARGC()==0 THEN A$=Stack$[nX] ELSE A$=DEFARG(0)
409   CASE A$[0]
410     WHEN "#":  RETURN A$[1]>"7"
411     WHEN "$":  RETURN VAL(MID$(A$,1))<0
412     OTHERWISE: RETURN A$[0]!=" "
413   ENDCASE
414 END
415
416
417 'Проверка, записан ли в RX ноль
418 COMMON DEF IsZero(*)
419   VAR A$
420   IF DEFARGC()==0 THEN A$=Stack$[nX] ELSE A$=DEFARG(0)
421   CASE A$[0]
422     WHEN "#":  RETURN A$==#GZERO
423     WHEN "$":  RETURN VAL(MID$(A$,1))==0
424     OTHERWISE: RETURN A$==#ZERO
425   ENDCASE
426 END
427
428
429 'Преобразовать десятичное число в гекс
430 '- каждая тетрада bcd-числа преобразуется независимо
431 '- большие положительные в 7fff:ffff
432 '- большие отрицательные в ffff:ffff
433 '- добавить отрицательные целые в гекс
434 COMMON DEF ToGex$(A$)
435   VAR G$, I, N
436   IF A$[0]=="#" THEN RETURN A$ 'Если уже гекс, не трогать
437   IF A$[0]=="$" THEN A$=ToDec$(VAL(MID$(A$,1)))
438   N=GetDigit(A$[17])+1
439   IF A$[0]=="-" || A$[15]=="-" THEN RETURN #GZERO
440   IF A$[16]!="0" || N>8 THEN
441     IF A$[0]=="-" THEN RETURN #GMAXN ELSE RETURN #GMAX
442   ENDIF
443   G$=#GZERO
444   FOR I=1 TO N:G$[8-N+I]=A$[I]:NEXT
445   RETURN G$
446 END
447
448
449 'Преобразовать целое в гекс
450 COMMON DEF MkGex$(N)
451   VAR G$=#GZERO, I=8
452   WHILE N<0:INC N,#LIMIT:WEND
453   WHILE N!=0
454     G$[I]=#HEX$[N AND 15]:DEC I
455     N=INT(N/16)
456   WEND
457   RETURN G$
458 END
459
460
461 'Считать гекс, как двоичное целое со знаком
462 COMMON DEF ReadGex(A$)
463   VAR I, G=GetDigit(A$[1])
464   FOR I=2 TO 8:G=G*16+GetDigit(A$[I]):NEXT
465   IF A$[1]>"7" THEN DEC G,#LIMIT
466   RETURN G
467 END
468
469
470 'Преобразовать десятичное число в вещественное
471 COMMON DEF ToReal(A$)
472   VAR I, B$=ToDigit$(A$[1])+".", C$
473   FOR I=2 TO 14:PUSH B$,ToDigit$(A$[I]):NEXT
474   IF A$[15]==" " THEN PUSH B$,"E" ELSE PUSH B$,"E-"
475   PUSH B$,ToDigit$(A$[16])+ToDigit$(A$[17])
476   IF A$[0]==" " THEN RETURN VAL(B$) ELSE RETURN -VAL(B$)
477 END
478
479
480 'Преобразовать в десятичное число
481 '- продумать отрицательные гексы
482 COMMON DEF Land(A$)
483   CASE A$[0]
484     WHEN "#":
485       VAR I, G=GetDigit(A$[1])
486       FOR I=2 TO 8:G=G*10+GetDigit(A$[I]):NEXT
487       RETURN G
488     WHEN "$": RETURN VAL(MID$(A$,1))
489     WHEN "G": WHEN "M": WHEN "S":
490       A$[0]=" "
491     WHEN "g": WHEN "m": WHEN "s":
492       A$[0]="-"
493   ENDCASE
494   RETURN ToReal(A$)
```

```
    ENDCASE
    RETURN ToReal(A$)
END

'Убедиться, что в RX число
COMMON DEF LandRX
    VAR S$=Stack$[nX][0]
    IF S$=="#" OR S$=="$" THEN Stack$[nX]=ToDec$(Land(Stack$[nX]))
END

'Преобразовать число в десятичное
COMMON DEF ToDec$(A)
    VAR B$=#ZERO, N$=STR$(ABS(A))
    VAR I,L=LEN(N$),E
    IF A<0 THEN B$[0]="-"
    CASE TYPEOF(A)
      WHEN #T_INT:                          'Преобразуем целое
        FOR I=1 TO L:B$[I]=N$[I-1]:NEXT
        B$[17]=#HEX$[L-1]
        RETURN B$
      WHEN #T_REAL:                         'Преобразуем вещественное
        A=ABS(A)
        IF A<1E-99 THEN RETURN #ZERO 'Машинный ноль
        IF A<1 THEN
          E=INT(CEIL(-LOG(A,10)))           'Двоичная математика может дать ошибку
          A=A*POW(10,E)
          N$=FORMAT$("%.13F",A)
          IF A>=10 THEN                     'Считаем, что ошибка только на один разряд
            DEC E:N$[2]=N$[1]               '12.34 → 12234 (1.234)
          ELSEIF A<1 THEN
            INC E:N$=MID$(N$,1):N$[0]=N$[1] '0.123 → 1123 (1.23)
          ENDIF
          IF E>0 THEN B$[15]="-"
          B$[16]=#HEX$[E DIV 10]            'Считаем, что порядок меньше 100
          B$[17]=#HEX$[E MOD 10]
        ELSEIF A>=10 THEN
          E=INT(LOG(A,10))                  'Двоичная математика может дать ошибку
          A=A/POW(10,E)
          N$=FORMAT$("%.13F",A)
          IF A>=10 THEN                     'Считаем, что ошибка только на один разряд
            INC E:N$[2]=N$[1]               '12.34 → 12234 (1.234)
          ELSEIF A<1 THEN
            DEC E:N$=MID$(N$,1):N$[0]=N$[1] '0.123 → 1123 (1.23)
          ENDIF
          B$[16]=#HEX$[E DIV 10]            'Считаем, что порядок меньше 100
          B$[17]=#HEX$[E MOD 10]
        ELSE
          N$=FORMAT$("%.13F",A)             'Числа от 1 до 10, не включая 10
        ENDIF
        L=LEN(N$)
        B$[1]=N$[0]
        IF L>14 THEN L=15
        FOR I=2 TO L-1:B$[I]=N$[I]:NEXT
        RETURN B$
      WHEN #T_STR:
        RETURN A                            'Возможно, на входе уже десятичное
      OTHERWISE:
        RETURN #ZERO                        'По умолчанию ноль
    ENDCASE
END

'Преобразовать V# в целое и ограничить диапазоном [MI,MA]
COMMON DEF CAP(V#,MI,MA)
    IF V#<=MI THEN
      RETURN MI
    ELSEIF V#>=MA THEN
      RETURN MA
    ELSE
      RETURN INT(V#)
    ENDIF
END

'Считать угол формата M, перевести в градусы
'- игнорирует знак мантиссы
DEF GetM#(A$)
    VAR E=VAL(A$[16]+A$[17]),sE$=A$[15]
    IF sE$!=" " THEN 'Отрицательный порядок
      RETURN VAL(A$[1]+"."+MID$(A$,2,13))*POW(10,-E)/6
    ELSEIF E>12 THEN
      RETURN VAL(A$[1]+"."+MID$(A$,2,13))*POW(10,E)
    ELSE
      VAR rM#=VAL(A$[2+E]+"."+MID$(A$,3+E,12-E))/6
      RETURN VAL(MID$(A$,1,E+1))+rM#
    ENDIF
END

'Считать угол формата MC, перевести в градусы
'- игнорирует знак мантиссы
'- не проверено
DEF GetS#(A$)
    VAR E=VAL(A$[16]+A$[17]), sE$=A$[15], iM, rS#
    IF sE$!=" " THEN 'Отрицательный порядок
      CASE E
        WHEN 1: 'MMCCcc
          iM=VAL(A$[1]+A$[2])
          rS#=VAL(A$[3]+A$[4]+"."+MID$(A$,5,10))
          RETURN (iM*60+rS#)/3600
        WHEN 2: 'MCCccc
          iM=VAL(A$[1])
          rS#=VAL(A$[2]+A$[3]+"."+MID$(A$,4,11))
          RETURN (iM*60+rS#)/3600
        OTHERWISE: 'Ccccc..
          RETURN VAL(A$[1]+"."+MID$(A$,2,13))*POW(10,2-E)/36
      ENDCASE
    ELSEIF E>12 THEN '..rrrr
      RETURN VAL(A$[1]+"."+MID$(A$,2,13))*POW(10,E)
    ELSE
      CASE E
        WHEN 12: RETURN VAL(MID$(A$,1,13))+VAL(A$[14])/6          ':.rrrM
        WHEN 11: RETURN VAL(MID$(A$,1,12))+VAL(A$[13]+A$[14])/60 ':.rrMM
        OTHERWISE: '..rMMCCcc..
          iM=VAL(A$[2+E]+A$[3+E])
          rS#=VAL(A$[4+E]+"."+MID$(A$,5+E,10-E))
          RETURN VAL(MID$(A$,1,E+1))+(iM*6+rS#)/360
      ENDCASE
    ENDIF
END

'Привести аргумент к радианам
```

```
'Привести аргумент к радианам
COMMON DEF ToRad#(A$)
  CASE A$[0]
    WHEN "G": WHEN "g":
      RETURN RAD(Land(A$))
    WHEN "M": RETURN RAD(GetM#(A$))
    WHEN "S": RETURN RAD(GetS#(A$))
    WHEN "m": RETURN -RAD(GetM#(A$))
    WHEN "s": RETURN -RAD(GetS#(A$))
  ENDCASE
  RETURN Land(A$)
END

'Привести радианы к градусной мере
'COMMON DEF FromRad(V#)
'  CASE Rf045
'    WHEN 0: RETURN DEG(V#)
'    WHEN 1: RETURN V#
'    WHEN 2: RETURN DEG(V#)/0.9
'  ENDCASE
'END

'Память программ
'Очистить память программ, позже -- восстановить из энергонезависимой
'Очищает также память программ цели
'- Самостоятельно вызывает StrClear
COMMON DEF PrgClear
  FILL PrgM,-1
  StrClear
END

'Считать ячейку памяти программ
'- из несуществующих адресов считывается ffff:ffff
COMMON DEF ReadPrgM
  IF VAR("0:ERROR")==0 THEN
    VAR A=ReadGex(ToGex$(Stack$[nX]))
    X1$=Stack$[nX]
    IF A<0 !! A>65535 THEN A=-1 ELSE A=PrgM[A]
    Stack$[nX]=MkGex$(A)
  ENDIF
END

'Регистровые операции
'Очистить регистры, позже -- восстановить из энергонезависимой памяти
COMMON DEF RegClear
  FILL R$,#ZERO
END

'Записать регистр X в регистр функций N
COMMON DEF STOF N
'  CASE N
'    WHEN &hf045: Rf045=CAP(ToReal(Stack$[nX]),0,2)
'  ENDCASE
END

'Запись содержимого регистра X в регистр N
COMMON DEF STO N
  IF N>=0 THEN
    IF N>=&hf000 && N<&h10000 THEN
      STOF N
    ELSEIF N<&h10000 THEN
      R$[N]=COPY(Stack$[nX])
    ELSEIF N<&h20000 THEN
      PRGM[N]=ReadGex(ToGex$(Stack$[nX]))
    ENDIF
  ENDIF
END

'Косвенная запись содержимого регистра X
'по содержимому адресного регистра N
'- проверить диапазон N
COMMON DEF KSTO N
  VAR A$=ToGex$(R$[N])
  R$[N]=A$
  STO ReadGex(A$)
END

'Косвеная запись содержимого регистра X, N от 0 до 15
COMMON DEF KSTO1 N
  VAR A$=ToGex$(R$[N])
  CASE N
    WHEN 0: WHEN 1: WHEN 2: WHEN 3: GexDec A$
    WHEN 4: WHEN 5: WHEN 6: GexInc A$
  ENDCASE
  R$[N]=A$
  STO ReadGex(A$)
END

'Вызов в X содержимого регистра функций N
'+ если регистра нет, в стек загружается 0
COMMON DEF RCLF N
  CASE N
'    WHEN &hf045:
'      StackUpFast:Stack$[nX]=ToDec$(Rf045)
    OTHERWISE:
      StackUpFast:Stack$[nX]=#ZERO
  ENDCASE
END

'Вызов в X содержимого регистра R
'+ если регистра нет, в стек загружается 0
COMMON DEF RCL N
  IF N<0 !! N>=&h20000 THEN
    StackUpFast:Stack$[nX]=#ZERO
  ELSEIF N>=&hf000 && N<&h10000 THEN
    RCLF N
  ELSEIF N<&h10000 THEN
    StackUpFast:Stack$[nX]=COPY(R$[N])
  ELSE
    StackUpFast:Stack$[nX]=ToDec$(PrgM[N])
  ENDIF
END

'Косвенный вызов в регистр X по содержимому адресного регистра N
'- проверить диапазон N
COMMON DEF KRCL N
  VAR A$=ToGex$(R$[N])
  R$[N]=A$
  RCL ReadGex(A$)
END

'Косвенный вызов в регистр X, N от 0 до 15
COMMON DEF KRCL1 N
  VAR A$=ToGex$(R$[N])
  CASE N
    WHEN 0: WHEN 1: WHEN 2: WHEN 3: GexDec A$
    WHEN 4: WHEN 5: WHEN 6: GexInc A$
  ENDCASE
  R$[N]=A$
  RCL ReadGex(A$)
END
```

PRG1 RING.PRG