MARK SPAHN’S SUDOKU ROUTINE ANNOTATIONS: May 16, 2010

PROGRAM: AGETGRID

{9,9}→dim([A])

For(R,1,9)
Prompt X

For(C,1,9)
int(10*fPart(X/10^C))→[A](R,10-C)
End
End

[A]→[C] ; Stores a copy of [A] in [C] for possible reference.
Disp [A]

Note: The Goto regimes are written so that, as far as possible, we do not have a sequence like "If...Then...Goto L:End". This is from the fear that following the destination label L, whichever "End" is encountered will be (mis)interpreted as going with this If-Then-(Else)-End statement.

PROGRAM: AASUDOKU

0100→N ; ***set options variable N to a particular choice
0→Q ; set default delay to Q=0 seconds (top speed)
0→H ; SHGRID assumes that H=0; i.e., hypothesizing is off.
ClrDraw ; Clear the graphic screen (hit 2nd, CATALOG, letter)
PlotsOff ; or find in DRAW.
AxesOff
prgmSHGRID ; Display the starting 9x9 sudoku grid
"ROWCOLBOX"→Str1 ; ROW,COL,BOX for display of three kinds of ENNEADS
prgmPRIMPROD ; Set each entry in the 3x9 matrix [B] to a product of
; primes with row 1: rows; row 2, columns; row 3: boxes.
0→Z ; Z = how many empty cells (zeros) remain in the grid

For(R,1,9)
For(C,1,9)
([A](R,C)=0)+Z→Z ; True statement has value 1; false statement 0
[End]

prgmSHZEROCT ; Show on screen the Zero Count Z (how many empty cells)
startTmr→A ; Record in A the starting time (***I use variable “oh”)
prgmSHTIME ; Show the elapsed time (here, the set-up time)
0→S ; S = which pass through the grid it is
0→M ; number of times a set of hypotheses has been generated
1→V ; *** “verified” flag
0→dim(L₁) ; Start with an empty list L₁ of tentatively filled cells
0→dim(L₄) ; Empty the solution-history list L₄
{0}→L₆ ; Start with an empty list of inconclusive hyps
; L₆(n): n = how many hyps bypassed because already tried

prgmSCAN ; SCAN the sudoku grid repeatedly, filling in cells
If 0
\nGoto BS ; If error, a Bad Sudoku
If Z=0
Goto Z ; If no more cells to fill in, solution found.
Lbl HP ; Generate Hypothesis Pairs (When "ReStart" appears in the notes, you jump to this instruction)
1→dim(L₆) ; Empty the “inconclusive hyps” list (except for its count)
0→dim(L₁) ; Start with an empty "tentatively filled cells " list L₁

---

1 In order to show program structure, program lines are indented here. They are not indented in the TI calculator. In fact, blank spaces inserted at the beginning of instructions will cause errors.
1+int(M) → M  ; M) tells which hypothesizing session, like 1), 2)
1→J
prgmERASEJ  ; Erase line J=1 on the screen
0→dim(L₁)
10^9*FPart(N/10^9)→ N  ; Set the leftmost digit of N to zero (type of ennead)
prgmCELLHYPS  ; Append cell hypotheses to L₁
0→ V  ; V=0 means "current hyp unverified (not known to be true)"
1→ H  ; HL (Hypothesis Loop) to XH (next Hypothesis) is
Lbl HC
6→ J
prgmERASEJ
Text(6*6,0,M,"",dim(L₁),", HYPS")  ; Show session number M, how many hyps in list L₁
prgmWAIT  ; ***
If dim(L₁)<H
Goto IK  ; If fewer than H hypotheses, get some ennead hyps
Lbl HL  ; like For(H,1,dim(L₁)), but can jump into and out of loop
prgmSHYP  ; Display the hypothesis, like 1:(1,4)=3; sets hyp's R,C,D
prgmWAIT  ; allow user time to read it.

; Under some circumstances, it can be decided immediately whether a hypothesis
; H:(R,C)=D is true or false. If its cell (R,C) is already filled with D,
; (set by a previous hypothesis), the hypothesis is true ("ALREADY SO").
; If cell (R,C) is filled with a nonzero value other than D, the hypothesis is
; false. If cell (R,C) is empty as D could not go into it ("BLOCKED" by a D in an
; intersecting ennead), the hypothesis is false. The hypothesis-pairs 1-2, 3-4,...,
; are "twins" in the sense that if one is true, the other is false, and if one
; is false, the other is true.
[A](R,C)→ F  ; Look at the number in the cell referred to by the hyp
If F=0
Goto F0  ; F0 ("eff zero") means the cell is empty
    ; At this point the hyp's cell has already been filled
    ; (by a previous hypothesis); record this in solution-
    ; history list L₄, in a recognizable format:
1000H+100R+10C+D+F/10→ T  ; either HRCD.D (the hyp turned out to be true) or,
If F≠ D
    T→ T
    ; If the hypothesis is wrong, -HRCD.F
T→ L₄(1+dim(L₄))  ; If cell (R,C) already contains what the hyp says
If F= D
Goto FD
Lbl FS  ; Cell (R,C) contains F, something other than 0 or D
prgmERASE8  ; Display, under line H:(R,C)=D, the line (R,C)=F≠ D
Text(8*6,0,"("",R","",C","")="",F,"","",D)
prgmWAIT
prgmERASE8
Goto NG  ; this is No Good
Lbl FD  ; (R,C) contains D, consistent with the current hyp H
prgmERASE8
Text(8*6,0,"ALREADY SO")  ; tell the user
prgmWAIT  ; pause to let him read the message
prgmERASE8
Goto XP  ; then go on to the next Pair of hypotheses
Lbl F0  ; cell (R,C) is empty
prgmDINRC  ; Can digit D be excluded immediately from filling (R,C)?
If T
    Goto F1
prgmERASE8
Text(8*6,0,"BLOCKED")  ; tell the user
prgmWAIT
prgmERASE8
Goto NG  ; and go on to "This hyp is No Good (false)"
Lbl F1  ; Otherwise, if this hyp has already been found
1→ J  ; to be inconclusive (leading to neither solution
dim(L₆)→T ; nor contradiction; i.e., is in list L₆),
Lbl F6
If L₁(H)=L₆(J)
  Goto I6
IS>(J,T)
  Goto F6
Goto FT
Lbl I6 ; Skip this inconclusive hypothesis
  L₁(1)→L₂(1) ; keep count: one more hyp-check has been bypassed
prgmERASE8
  Text(8*6,0,"SKIPD",L₆(1)) ; how many already-done hyps have been skipped
prgmWAIT
prgmERASE8
0→V
Goto XH ; go on to next hyp.
Lbl FT ; starting with an empty "cells tentatively filled" list,
  0→dim(L₂) ; putting D into cell (R,C) (and into L₂ if V=0)
prgmFILLCELL ; Make passes thru the grid, till done/error/can't
prgmSCAN
If Z=0
  Goto Z ; Did the SCAN end by filling in all the grid's cells?
If θ
  Goto NG ; Did the SCAN end by reaching a contradiction?
If V=0
  Goto F2 ; If in "this is known to be true" mode,
  .01+M→M ; note that a cell(s) was filled in during hyp-session M
If fPart(N/10)
  Goto RS ; If the flag in N says so, restart with a new set of hyps
prgmFILLCELL ; Hyp led to neither solution nor error
prgmSCAN
If fPart(H/2)
  L₂→L₃ ; If working on an odd-numbered hyp, save L₂ in L₃
L₁(H)→L₆(1+dim(L₆)) ; append this hyp to list of inconclusive hyps
prgmERASE8
prgmUNWINDL2 ; Re-empty the hypothetically filled cells
prgmWAIT
prgmUNWINDL2
0→V
Goto XH ; Go on to the next hyp, whose truth value is unknown (V=0)
Lbl NG ; The current hypothesis H:(R,C)=D is known to be false
  0→θ ; Turn off any SCAN-produced error code,
  If V=0
    Goto N1 ; lest it still be on when ReStart calls HypPairs
    L₁ BS ; Sudoku has been discovered to be inconsistent
    9→θ ; Set the error code to "bad sudoku"
    Text(8*6,0,"BAD SUDOKU")
    Text(9*6,0,"UNSOLVABLE")
prgmWAIT
Goto Q
Lbl N1
prgmUNWINDL2 ; undo the sequence of filled cells (now known to be wrong)
1→V ; Change H to the partner hyp, mark it verified true (V=1)
l→dim(L₆) ; Empty the "inconclusive hyps" list (but keep the count)
If fPart(H/2)
  Then
    1+H→H
  Else
    H-1→H
End
prgmSETHYRCD ; Set (R,C)=D according to the updated hyp H
SUDOKU PROGRAMS

1000H+100R+10C+D→L₄(1+dim(L₄)); Append verified-true hyp H:(R,C)=D to history
If fPart(H/2)=0
  Goto HL; If this known-true hyp is even-numbered, loop back to HL
  ; At this point, we have an odd-numbered hyp that we have learned is true, and we
  ; have in list L₃={R C D R C D ... R C D} the cells to be filled in as a
  ; consequence of it.
  prgmSHHYP; display, with a "therefore" symbol, the hyp H:(R,C)=D
  .01+M→M; Mark hypothesizing session M) as having filled in cell(s)
  prgmERASE8
  Text(8*6,0,"REFILL ",dim(L₃)/3,"..."); Re-fill grid with previously calculated cells
  prgmWAIT
  Q→Q; make delay ≤0
  [[For(I,1,dim(L₃))-2,3]
    L₃(I)→R
    L₃(I+1)→C
    L₃(I+2)→D]
  *** I have:
  [[For(I,1,dim(L3))]
    L₃(I)→T
    10fPart(T)→D
    int(T)/10→T
    int(T)→R
    10(T-R)→C]
  prgmFILLCELL
End
  -Q→Q; restore Q (assumes Q was ≥0 originally)
  prgmERASE8
  If fPart(N/10)
    Goto RS; If N-flag set up for restart when hyp found true, to RS
  Lbl XP; make H point to the first hypothesis of the next Pair
  0→V
  If fPart(H/2)
    1+H→H; If current H is odd, add 1 to skip even-numbered hyp
  Lbl XH;
  IS>(H,dim(L₁))
    Goto HL; if cell filled during this session, restart with new hyps
    Goto RS
  Lbl IK;
  10^9+N→N; add 1 to the leftmost digit of N (for ennead kind K)
  int(N/10^9)→K; get kind K of ennead to generate
  If K=4
    Goto K4; if all ENNEADS types have been exhausted, to K4
  prgmENNHYPSK
  Goto HC; Have any been added?
  Lbl K4
  If dim(L₁)>0
    Goto HF; there are hyps, but none has led to filling in a cell
    prgmERASE8
    Text(8*6,0,"NO HYPS.")
    Goto Q
  Lbl HF
  3→θ; error code for "no hyps have been generated"
  prgmERASE8
  Text(8*6,0,"NO HYPS")
  Goto Q
  Lbl HF
  8→θ; Hypotheses have Failed to find solution or contradiction
  prgmERASE8
  Text(8*6,0,"HYPS FAIL'D")
  Text(9*6,0,"TOO BAD.")
  prgmWAIT
  Goto Q
  Lbl RS
  For(J,6,9); Entire hypothesis list has been considered (or bypassed)
prgmERASEJ
End
For(J,0,Y-1)
 Pxl-off(62,J) ; Erase any dots at bottom of screen
End
Text(8*6,0,"RESTART...") ; *** I have Text(8*6,0,"NEW HYPS...")
prgmWAIT
prgmERASE8
Goto HP ; generate a fresh set of hypotheses.
Lbl Z
0→θ ; Solution found; set error code to zero
For(J,0,Y-1)
Pxl-off(62,J) ; erase line of dots at bottom of screen
End
If dim(L2)≠0 Then
 *** I have:
[(For(I,1,dim(L2))-2,3)
 L2(I)→R
 L2(I+1)→C]
 *** I have:
[(For(I,1,dim(L2))]
 int(L2(I))/10→T
 int(T)→R
 10fPart(T)→C]
 prgmRCPIXLTJ
 Pxl-off(T,J+3)
End
End
For(J,1,9)
prgmERASEJ
End ; Erase all 9 lines on screen
Text(4*6,0,"SOLVED!") ; Write success-proclaiming message.
Text(5*6,0,( PASS "S,")") ; 6 blanks between ( and PASS
5→J
3→T
prgmINS SYMBOL ; Write "in" just before "PASS"
If H=0
 Goto Q ; If hypothesizing was done,
If int(M)=1 ; If one set of set of hyps was considered,
Then
 Text(7*6,0,"CHECKED ",H) ; show how many hypotheses were checked,
 Text(8*6,0,"OF ",dim(L1)," HYPS") ; out of how many generated.
End
If int(M)>1 Then ; If there were multiple hypothesizing sessions,
 Text(7*6,0,"TO HYP ",H," OF") ; show how far you got
 Text(8*6,0,"HYP-SET ",M,"") ; in which hypothesizing session 1), 2), etc
End
Lbl Q
Pxl-off(0,94) ; Turn off the "WAIT for user response" dot in upper right
S→L4(1+dim(L4)) ; Append as the last item on the solution-history list L4
 ; the number S of passes required to find the solution.
prgmSHTIME ; Show the elapsed time; here, the solution time.

PROGRAM:CELLHYPS ; Fill list L1 with the hypotheses generated from cells
 ; that must hold either of two digits. For example, if cell (8,5) might hold either
 ; a 4 or a 7, this generates two hypotheses, either (8,5)=4, or (8,5)=7. These
 ; hypotheses are represented in the hypothesis list L1 by the numbers 85.4 and 85.7
prgmSHTIME ; Show starting time for getting cell hyps
prgmSHYPGRD ; Show the little hypothesis grid
For(R,1,9)
 For(C,1,9)

SUDOKU PROGRAMS

prgmCHKEY ; Allow user to stop/start CELLYPS display
If [A](R,C)≠0 ; Look only at empty cells
  Goto XC ; XC stands for next C
0→F ; F = first-found digit that might go into cell (R,C)
0→G ; G = second-found digit that might go into cell (R,C)
1→D ; For each digit D from 1 to 9
Lbl DL ; DL stands for D-Loop
  PRIME(D)→P ; If digit D is not in the
  If fPart([B](1,R)/P)=0 ; row,
    Goto XD ; XD stands for next D
  If fPart([B](2,C)/P)=0 ; column, or
    Goto XD
  prgmSETB
  If fPart([B](3,B)/P)=0 ; box that cell (R,C) is in, D will fit in (R,C).
    Goto XD
  If G≠0 ; If two fitting digits have already been found,
    Goto XC ; no good; try the next cell
  If F=0
    Then
      D→F
    Else
      D→G
    End
  Lbl XD ; next D
  IS>(D,9)
    Goto DL ; Try the next digit D. If no digit will fit into this
cell, an inconsistency has been discovered
  If F=0
    Then
      4→θ ; unfillable cell found while looking for cell hypotheses
      8→J
      prgmERASEJ ; erase line 8
      Text(6J,0,"(",R,",",C,")=NONE")
      Return
    End
  If G=0 ; This shouldn't happen when this program is called
    Then
      5→θ
      F→D
      prgmFILLCELL
      prgmERASE8
      Text(6J,0,"CELHYP ERR")
      Return
    End
  7→J
  prgmERASEJ
  Text(6J,0,"(",R,",",C,")=""F," / ",G) ; e.g., (8,2)=5 / 7
  Pxl-On(51+(R-1)+int((R-1)/3),13+(C-1)+int((C-1)/3)) ; set dot in # grid
  10R+C+F/10→L1(1+dim(L1)) ; Append RC.F, RC.G to the hypothesis list L1
  10R+C+G/10→L1(1+dim(L1))
  If 1
    prgmWAIT
  Lbl XC ; "next C"
End
End
M/100+dim(L1)/100000→L4(1+dim(L4)) ; hyp session, number of hyps .MMhhh into L4
prgmERHYPGRD ; erase the little hypothesis grid
prgmSHTIME ; Show the ending time for getting the cell hypotheses

PROGRAM:CELLS ; SCAN through all 81 cells, looking for a cell into which
                ; only one digit will fit
1→J
prgmERASEJ ; Erase the line on which successive "(R,C)" will be shown
For(R,1,9)
prgmCHKEY ; If a key is being pressed, update delay Q
For(C,1,9)
  If [A](R,C)
    Goto C ; If cell at (R,C) is occupied, on to next cell
  prgmHILITCEL ; If an empty cell, highlight it on the screen
  Text(1*6,0,"("+R+",","+C+","+")") ; Show (R,C) on line 1
  0→F
  1→D ; Count up from digit 1 to digit 9
Lbl DL ; Digit Loop
  PRIME(D)→P ; Set P to the prime corresponding to digit D
  If fPart([B](1,R)/P)=0 ; If digit D is already in row R, try next D
    Goto XD
  If fPart([B](2,C)/P)=0 ; If digit D is already in column C, on to next D
    Goto XD
  prgmSETB ; Set B to the number of the box cell (R,C) is in
  If fPart([B](3,B)/P)=0 ; If D is already somewhere in that box, try next D
    Goto XD
  ; At this point, digit D can fit in cell (R,C)
  If F
    Goto U ; If a previous digit F can also fit in (R,C),
  ; this cell accommodate more than 1 digit, so try next cell
  D→F ; Record in F the first digit that can go in this cell
Lbl XD ; on to the next Digit
  IS>(D,9) Goto DL
  If F=0 ; If no digits were found that can fit into cell (R,C)
    Then
      1→0
      2→J
      prgmERASEJ
      Text(2*6,0,"CAN'T FILL")
      prgmWAIT
      prgmHILITCEL
      Return
    End
End
F→D ; Display unique digit that fits in cell, like (4,2)=7
Text(1*6,17,"=",D)
prgmFILLCELL
1→J
prgmERASEJ ; Erase line 1
If J=0
  Return ; Sudoku all filled in?
Goto C
Lbl U ; Unhighlight this cell, and move on to next cell
prgmHILITCEL ; Highlight the cell again, which cancels its highlighting
Lbl C
End
End
prgmSHTIME

Program:CHKEY
getKey→T ; Check whether a key is being pressed;
If T=0
  Return ; If not, do nothing.
If T=103 or T=105 ; If the "." key or the "Enter" key,
  Then
    10→Q
    Pxl-On(0,93) ; set the delay Q to "WAIT for user to press Enter"
    ; Turn on an unobtrusive pixel in the upper-right corner
    ; of the screen to show that upon finding something,
    ; the program will WAIT for the user to press "Enter".
    ; Pixel (0,93) is near the climbing-inchworm "working"
; or flashing-jingle-bells "WAITing for the user to
; press Enter" vertical row of pixels.
; Alternatively, Pxl-On(62,0) turns on a pixel in the
; lower-left corner of the screen.

Return
Else
    Pxl-Off(0,93)
If T=102
    0→Q
If T=92
    1→Q
If T=93
    2→Q
If T=94
    3→Q
If T=82
    4→Q
If T=83
    5→Q
If T=84
    6→Q
If T=72
    7→Q
If T=73
    8→Q
If T=74
    9→Q
End

PROGRAM: DINRC
0→T
; Sets T to 1 (true) iff digit D can go into cell (R,C)
If [A](R,C)≠0
    Return
Prime(D)→P
If fPart([B](1,R)/P)=0
    Return
If fPart([B](2,C)/P)=0
    Return
prgmSETB
If fPart([B](3,B)/P)=0
    Return
1→T

PROGRAM: ENNEADS
1→J
prgmERASEJ
0→T
; Erase line 1
prgmINSymbol
; Position J, T to line 1, position 4
For(K,1,3)
    ; Kind 1 is row, kind 2 is column, kind 3 is 3x3 box
    Text(1*6,10,sub(Str1,1+3(K-1),3)) ; depending on K, show ROW COL BOX on line 1 pos 10
End
For(I,1,9)
    ; For all nine boxes/columns/rows
    prgmCHKEY
    If [B](K,I)=L
        Goto I
    prgmHILITENN
    Text(1*6,23,I,"?") ; Display the ennead number in proper place in line 1
End
Lbl EN
    ; Do this ennead: consider all digits and where they fit
    1→D
    Lbl DL
    ; Count up, digit D = 1 to 9
    Digit Loop
    0→F
    ; First-found cell of this ennead into which D will fit
PRIME(D) → P
If fPart([B](K,I)/P)=0; Divides evenly; thus P is a factor.
   Goto XD
   ; If this digit D is already in this ennead, on to neXt D.
   ; Note here the power of the prime product.
Text(1*6,0,D)
   ; Display the digit D in "D <in> ROW/COL/BOX I?" on line 1
1→E
   ; Look at ennead's cells E from 1 to 9
Lbl EL
   ; cell E Loop
prgmSETRCB
   ;Set Row, Column, Box for cell E of ennead of kind K, id# I
If [A](R,C)≠0
   Goto XE
   ; If cell is already occupied, on to next cell E in ennead
K→T
   ; If digit D is in any of the other two kinds of ENNEADS that cell (R,C) is in, try next cell E
T-1→T
   ; increment T, cycling 1,2,3; 1,2,3; 1,2,3
If fPart([B](T,R(T=1)+C(T=2)+B(T=3))/P)=0
   Goto XE
T-1→T
   ; in cr
T-1→T
   ; cycling T, cycling 1,2,3; 1,2,3; 1,2,3
If fPart([B](T,R(T=1)+C(T=2)+B(T=3))/P)=0
   Goto XE
If F≠0
   Goto XD
   ; At this point, cell E can accommodate digit D, so record this cell in F
E→F
   ; If there was a previous D-accommodating cell, it's not unique
Lbl XE
IS>(E,9)
   Goto EL
   ; try the next cell E
If F=0
   ; If the ennead has no cells in which this digit D Then
2→0
   ; might go, an error (or a hypothesis has led to a contradiction)
2→J
prgmERASEJ
   ; erase line 2
Text(6J,0,"NO!")
   ; Answer question like "4 in BOX 5?" with "NO!"
prgmWAIT
prgmHILITENN
Return
End
F→E
   ; Get the (unique) cell of the ennead in to which
prgmSETRCB
   ; D can go, and display it, like 7 IN BOX 5(8)
Text(1*6,27,"("E,"")")
prgmFILLCELL
   ; Fill cell (R,C) with digit D
Text(1*6,27,"? ")
   ; ? and 6 blanks to erase the "(8)" part
If Z>0
   Goto EN
prgmHILITENN
   ; If sudoku still unfinished, look at this ennead again
Return
   ; and exit to calling routine
Lbl XD
IS>(D,9)
   Goto DL
   ; try the neXt Digit
prgmSHTIME
   ; update elapsed time after each ennead
prgmHILITENN
   ; Erase the highlighting by calling it again
Lbl I
End
Lbl K
End

Program: ENNHYPK
   ; Looks at ENNEADS and finds digits that can occur in only two cells of the ennead.
; For example, if 5 can occur in box 1 in either the box's cell 3 -- which is cell
; (1,3) -- or the box's cell 4 -- which is cell (2,1) -- this routine produces the
; two companion hypotheses (1,3)=5, (2,1)=5.
; Kind of ennead is K=1 for rows, 2 for columns, 3 for boxes.
; int(10fPart(N/10^4)) is the 4th digit of N from the right; if nonzero, this tells
; which kind of ennead to consider.

prgmSHTIME         ; Show starting time for finding ennead hyps of kind K
prgmSHYPGRD        ; show the little hypothesis grid
For(K,1,9)          ; For(K,1,9)
  For(I,1,9)        ; show(K,1,9)
    prgmCHKEY       ; allow starting/stopping during hyp generation
      If [B](K,I)=L
        Goto I
      Lbl 5
      1→D
      Lbl 6
      0→F
      0→G
      lPRIME(D)→P
      If fPart([B](K,I)/P)=0
        Goto D
      1→E
      Lbl 7
      prgmSETRCB
      If [A](R,C)≠0
        Goto E
      0→T
      T+1→T
      If T=K+1
        1→T
      If fPart([B](T,R(T=1)+C(T=2)+B(T=3))/P)=0
        Goto E
      T+1→T
      If T=K+1
        1→T
      If fPart([B](T,R(T=1)+C(T=2)+B(T=3))/P)=0
        Goto E
      If G≠0
        Goto D
      If F=0
      Then
        E→FD
      Else
        E→G
      End
      Lbl E
      DS<(E,1)
      Goto 7
    If F=0
    Then
      6→θ
      8→J
      prgmERASEJ
      Text(6J,0,"NO ",θ," IN ",sub(Str1,1+3(K-1),3)," ",I)
    prgmWAIT
    Return
  End
  If G=0
  Then
    8→J
    prgmERASEJ
    7→θ
Text(6J,0,"
ENNYPS ERR")
prgmWAIT
Return
End
7→J
prgmERASEJ
Text(6J,0,D,"\,"sub(Str1,1+3(K-1),3),I,"\,"G,"\,"\",F)
prgmWAIT
F→E
prgmSETRCB
10R+C+D/10→L(1+dim(L))
Pxl-On(51+(R-1)+int((R-1)/3),13+(C-1)+int((C-1)/3)) ; set dot (R,C) in # grid
G→E
prgmSETRCB
10R+C+D/10→L(1+dim(L))
Pxl-On(51+(R-1)+int((R-1)/3),13+(C-1)+int((C-1)/3)) ; set dot (R,C) in # grid

Lbl D
DS<(D,1)
Goto 6
Lbl I
End
; End For I
Lbl K
End
; End For K
M/100+K/1000+dim(L)/10^6→L(1+dim(L)) ; .MMKccc for M)C hyps of kind K
prgmERHYPGRD ; erase the little hyp grid
prgmSHTIME ; ending time for finding all ennead hyps of kind K

Program: ERASE8
8→J
prgmERASEJ

Program: ERASEJ
; Erase line J
Text(6J,0,"\n"") ; exactly 38 spaces

Program: ERHYPGRD
; Erase the little 9-box hypothesis grid
51→F
13→G
For(T,F-1,F+11)
For(J,G-1,G+11)
Pxl-off(T,J)
End
End

Program: FILLCELL
; Put digit D into the cell in row R, column C,
; and adjust prime-products matrix [B] accordingly
If l
prgmWAIT
D→[A](R,C)
prgmSETB
PRIME(D)→P
P*[B](1,R)→[B](1,R)
P*[B](2,C)→[B](2,C)
P*[B](3,B)→[B](3,B)
Z→Z
Pxl-On(62,W)
1+W→W
prgmSHCELL
prgmSHZEROCT
If H=0
Return
If V=1
Return
R→L₂(1+dim(L₂)); If hypothesizing provisionally, append R,C,D to list L₂
C→L₂(1+dim(L₂)); of cells that were tentatively filled in
D→L₂(1+dim(L₂)); pursuant to the current hypothesis

PROGRAM HILITCEL
; Highlight cell (R,C); used by routine cells
; Calling this routine twice restores the status quo. Slows processing about 10%
prgmRCPIXLTJ
Pxl-Change(T+2,J)
Pxl-Change(T+4,J)
Pxl-Change(T+2,J+2)
Pxl-Change(T+4,J+2)
Pxl-Change(T+2,J+1)
Pxl-Change(T+3,J)
Pxl-Change(T+3,J+2)
Pxl-Change(T+4,J+1)

PROGRAM HILITENN
; Highlight the current ennead being considered
If K=3
  Goto BX
If K=2
  Goto CL
Lbl RW
I→R
1→C
prgmRCPIXLTJ
Pxl-Change(T+2,J-5)
Pxl-Change(T+3,J-5)
Pxl-Change(T+3,J-4)
Pxl-Change(T+4,J-5)
Return
Lbl CL
1→R
I→C
prgmRCPIXLTJ
Pxl-Change(T-3,J)
Pxl-Change(T-3,J+1)
Pxl-Change(T-2,J+1)
Pxl-Change(T-3,J+2)
Return
Lbl BX
3int((I-1)/3)+2→R
1→C
prgmRCPIXLTJ
Pxl-Change(T+2,J-5)
Pxl-Change(T+2,J-4)
Pxl-Change(T+4,J-5)
Pxl-Change(T+4,J-4)
I-1→C
C-3int(C/3)→C
3C+2→C
1→R
prgmRCPIXLTJ
Pxl-Change(T-3,J)
Pxl-Change(T-2,J)
Pxl-Change(T-3,J+2)
Pxl-Change(T-2,J+2)

PROGRAM HYPPAIRS
; Generate pairs of hypotheses and list them in L₁
prgmSHTIME
; Show starting and stopping times for hyp generation
6→J
prgmERASEJ
Text(6J,0,"GET HYP...")
If 1
prgmWAIT ; To bypass WAITing, change the 1 to a zero
51→W
13→V ; (row W, col V)
For(J,V-1,V+11)
  Pxl-On(W+3,J)
  Pxl-On(W+7,J)
End
For(T,W-1,W+11)
  Pxl-On(T,V+3)
  Pxl-On(T,V+7)
End
0→dim(L₁)
If int(10fPart(N/10^4))
  prgmENNHYPS ; If digit 4 of options variable N is on (nozero),
  prgmCELLHYPS ; generate ennead-based hypotheses
  M/100+dim(L₁)/100000→L₄(1+dim(L₄))
6→J
prgmERASEJ ; erase line 6
prgmERASEJ ; erase line 7
Text(6*6,0,M," \),dim(L₁)," HYPS" ); Show on line 6 how many hypotheses were generated
For(T,W-1,W+11)
  For(J,V-1,V+11)
    Pxl-Off(T,J)
  End
End
If dim(L₁)=0
  Then
    3→θ
  prgmERASE8
    Text(6J,0,"NO HYPS.")
  Return
End
prgmSHTIME ; ending time for hypothesis generation

PROGRAM: INSYMBOL ; Draws on screen the lowercase word "in"
6J+3→J ; Set J to the top of the lowercase n in "in"
Pxl-On(J-2,T) ; The dot on the i
Pxl-On(J,T) ; The riser of the i
Pxl-On(J+1,T)
Pxl-On(J+2,T)
Pxl-On(J+2,T+2)
Pxl-On(J+1,T+2)
Pxl-On(J+2,T+2)
Pxl-On(J,T+3) ; Top of the n
Pxl-On(J,T+4)
Pxl-On(J+1,T+4) ; Right riser of the n
Pxl-On(J+2,T+4)

PROGRAM: PRIMPROD ; This PROGRAM fills [B] with prime products.
{2,3,5,7,11,13,17,19,23}→PRIME
prod(PRIME→L ; L = 2*3*5*7*11*13*17*19*23 = 223,092,870
{3,9}→dim([B])
Fill(1,[B])
For (R,1,9)
  For (C,1,9)
    [A](R,C)→D ; D is digit in original array position, thus 0≤D≤9.
    If D≠0
      Then
        .PRIME(D)→P ; Assigns a prime corresponding to D; for example, if D=5,
                     ; P would be the fifth prime, 11.
      prgmSETB ; 3int((R-1)/3)+int((C-1)/3)+1→B, box containing (R,C)
SUDOKU PROGRAMS

PROGRAM: RCPIXLTJ
1+X+6(R-1)+2int((R-1)/3)→T ; Make T, J the pixel row, column of the upper-
3+Y+5(C-1)+3int((C-1)/3)→J ; left corner of the 6x4-pixel numeral in cell (R, C)

PROGRAM: SCAN
0→θ ; Start error-free
Repeat W=0 ; until a SCAN thru the grid produces no filled cells
  1+S→S ; Increment the number of passes thru the sudoku grid
  0→W ; Start the pass with no cells filled
  For(J,0,Y-1)
    Pxl-off(62,J) ; Erase "cells filled this pass" row of dots
  End
  1→J ; End For
  prgmERASEJ ; Erase line 1 on the screen
  prgmCELLS
  If θ or Z=0 ; If not hypothesizing (Delete this "if H=0" condition?),
    Goto 9 ; hypothesis H:(R,C)=D, number ff of cells filled in
  prgmENNEADS
  If θ or Z=0 ; under this hyp, and error code θ (if any) that ended
    Goto 9 ; this hypothesis run, in format HRCD.ffθ
  If H=0 ; If not hypothesizing (Delete this "if H=0" condition?),
    S+W/100→L4(1+dim(L4)) ; append to the solution history
    ; how many cells W were fill in each pass S
    prgmSHPROG
  End ; End of Repeat W=0
Lbl 9 ; If hypothesis, add info to solution-history list L4
  If H=0
    Return
  prgmSETHYRCD
    ; If hyp led to an error, note with "-"
    T→T
    T→L4(1+dim(L4)) ; Append this number to list L4
  End

PROGRAM: SETB ; Tells what box B the cell in row R, column C is in
3int((R-1)/3)+int((C-1)/3)+1→B

Program: SEHYRCD
int(L4(H)/10)→R ; Get R, C, D, from H-th number RC.D in hypothesis list L4
10fPart(int(L4(H))/10)→C
10fPart(L4(H))→D

Program: SHHYGRD ; Show a little 9-box hypothesis grid
For(J,6,9)
  prgmERASEJ
End ; Erase lines
int(N/10^9)→K ; Look at leftmost digit of options variable N
If K=0
  Text(6*6,0,"CELL HYPS...") ; Show the grid's title
Else

Page 14 of 22
Text(6*6,0,sub(Str1,1+3(K-1),3)," HYPS...")
End ; End If K=0
5I→F
13→G
For(T,F-1,F+11)
  For(J,G-1,G+11,4)
    Pxl-on(T,J)
  End ; End For J
End ; End For T
For(J,G-1,G+11)
  For(T,F-1,F+11,4)
    Pxl-on(T,J)
  End ; End For T
End ; End For J

PROGRAM: SetHyYRCD
int(L(1)(H)/10)→R ; Get R, C, D, from H-th number RC.D in hypothesis list L
10fPart(int(L(1)(H))/10)→C
10fPart(L(1)(H))→D

PROGRAM: SETRCB ; Given cell E of ennead of kind K, id# I, set R, C, B to the row, column, and 3x3 box this cell is in
If K=1
  I→R
  E→C
Else
  If K=2
    E→R
    I→C
  Else
    I→B
    3int((B-1)/3)+int((E-1)/3)+1→R
    3(B-R)+E→C
  ; Derivation of above line: (C-1) = 3{(B-1) mod 3} + [(E-1) mod 3]
  ;       = 3{(B-1) - 3int((B-1)/3)} + [(E-1) - 3int((E-1)/3)]
  ;       = 3(B-1) + (E-1) -3{3int((B-1)/3) + int((E-1)/3)}
  ;       = 3(B-1) + (E-1) -3(R-1)
  ;       C - 1 = 3B - 3 + E - 1 - 3R + 3 = 3(B-R) + E - 1
  ;       C = 3(B-R)+E
End
If K≠3
  prgmSETB
End

PROGRAM: SHCELL ; Show on the screen the content of the cell at (R,C)
prgmRPXLTJ ; Make T=pixel row, J=pixel column for cell (R,C)
If Pxl-Test(T+6,J+3); If the cell below (R,C) has a "tentative" dot,
    .5+T→T ; set fractional part of T to nonzero (as a flag).
If [A](R,C) ; If the number in the cell is nonzero,
    Then
      Text(int(T),J,[A](R,C)) ; show it as a standard 6x4(5x3)-pixel numeral,
      If H(V=0)
        Pxl-On(int(T),J+3) ; mark this cell with a "tentatively filled in" dot.
      Else
        Text(T,J,"    ") ; erase whatever was there with four blanks,
        Pxl-On(int(T)+3,J+1) ; and show an empty cell with a centered dot.
      End
    If fPart(T)
      Pxl-On(int(T)+6,J+3) ; If cell below (R,C) was dotted, restore the dot.
End

PROGRAM: SHGRID ; shows the 9x9-cell grid
ClrDraw  ; clear the graphic screen
If Q>9
  Pxl-On(0,94)  ; If "WAIT for Enter" flag is on, show its "ON" dot
If 0
  Text(0,17,N)
If 0  ; Change to "If 1" in the rare case when tackling an
Then  ; "intractable" sudoku by running routine OSCAN
  "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"→Str2  ; Display value of control variable O ("oh"
  int(0/16)→T
  0→16T→J  ; in hexadecimal on the top line.
  Text(0,20,sub(Str2,T+1,1)+sub(Str2,J+1,1))
End  ; End of Then clause
2→X  ; (X,Y) is the pixel address of grid's upper-left corner
40→U  ; (X,Y)=(2,40) puts grid at lower-right of screen
U→Y  ; Other routines use (X,Y) as a global constants
   ; Draw the tic-tac-toe (#) lines:
X+20→R
For(C,U,U+54)  ; top horizontal line
  Pxl-On(R,C)
End
X+40→R
For(C,U,U+54)  ; bottom horizontal line
  Pxl-On(R,C)
End
U+18→C
For(R,X,X+60)  ; left vertical line
  Pxl-On(R,C)
End
U+36→C
For(R,X,X+60)  ; right vertical line
  Pxl-On(R,C)
End    ; Draw the square around the # pattern:
X→R
For(C,U,U+54)  ; top line, left to right
  Pxl-On(R,C)
End
U+54→C
For(R,X,X+60)  ; right line, top to bottom
  Pxl-On(R,C)
End
X+60→R
For(C,U+54,U,\1)  ; bottom line, right to left
  Pxl-On(R,C)
End
U→C
For(R,X+60,X,\1)  ; left line, bottom to top
  Pxl-On(R,C)
End
   ; Make sure Y has the correct value (SHCELL needs it)
  [Mystery: What changed Y since it got set to U at the
  beginning of this routine? Answer: TI-83 manual, page
  A-49: The TI-83 can update X, Y, R, q, and T during
  graphing, so you may want to avoid using these variables
  to store non-graphing data.]
1→V  ; Ensure that "tentative" dot is not shown by SHCELL
For(R,1,9)
  For(C,1,9)
    prgmSHCELL
  End
End
PROGRAM:SHHYP  ; Show hypothesis H:(R,C)=D
If \( V=1 \);

\[
\text{If the "hypothesis H Verified to be true" flag is on,}
\]

Then

\[
\text{write a "therefore" sign in the first four columns}
\]

\[
\text{top dot of the .}
\]

\[
\text{lower-left dot of the .}
\]

\[
\text{lower-right dot of the .}
\]

\[
\text{start Text one 4-pixel-wide character to the right}
\]

End

prgmSETHYRCD

\[
\text{get the values R,C,D corresponding to hypothesis That}
\]

\[
\text{(7*6,J,H,"(:",R,,",",C,"="),D)} ; \text{Show H:(R,C)=D on line 7}
\]

PROGRAM:SHPROG

\[
\text{Show progress}
\]

For \((J,1,4)\)

\[
\text{prgmERASEJ} ; \text{Erase lines 1-4, then write on lines 3, 4, e.g.:
}\]

\[
\text{Text(3*6,0,W," FILLED"} : 2 FILLED
\]

End

4→J

0→T

prgmINSymbol

\[
\text{Text(4*6,6,"PASS ",S) ; in PASS 3}
\]

PROGRAM:SHTIME

\[
\text{Show the elapsed time since the program began}
\]

startTmr-A→T

\[
\text{Number of seconds since the program began}
\]

\[
\text{int(T/60)} \rightarrow J
\]

\[
\text{how many minutes}
\]

\[
T-60J \rightarrow T
\]

\[
\text{how many sub-minute seconds}
\]

\[
"" \rightarrow \text{Str2}
\]

\[
\text{empty string}
\]

\[
\text{If T≤9}
\]

\[
"0" \rightarrow \text{Str2}
\]

\[
\text{For times like 1:05}
\]

\[
\text{Text(0,0,J,":",Str2,T)} \text{Show time in upper-left corner like 23:17}
\]

PROGRAM:SHZEROCt

\[
\text{Show the number Z (which represents how many of}
\]

\[
30 \rightarrow T
\]

\[
\text{the number in 9x9 matrix [A] are zero}
\]

\[
\text{Text(0,T," ")} \text{8 blanks, enough space for 2 digits}
\]

\[
\text{If Z=0}
\]

\[
\text{Return}
\]

\[
\text{If zero, don't display anything}
\]

\[
\text{Display the number Z, right-justified}
\]

PROGRAM:UNWINDL2

\[
\text{Cancel out all the cells R, C in list L_2 that were added}
\]

\[
to the grid; adjust empty-cell count Z and prime-products matrix [B] accordingly
\]

9→J

prgmERASEJ

\[
\text{Erase line 9}
\]

\[
\text{If dim(L_2)=0}
\]

\[
\text{Goto RT}
\]

\[
\text{If list of tentatively added cells is empty}
\]

\[
\text{Text(6J,0,"EMPTY ",dim(L_2)/3,"...")} \text{How many tentatively filled cells will be re-}
\]

\[
\text{emptied}
\]

prgmWAIT

For \((I,dim(L_2)-2,1,3)\)

\[
L_2(I) \rightarrow R
\]

\[
L_2(I+1) \rightarrow C
\]

\[
L_2(I+2) \rightarrow D
\]

prgmSETB

\[
0 \rightarrow [A](R,C)
\]

\[
\text{Empty the cell}
\]

prgmSHCELL

\[
1+Z \rightarrow Z
\]

prgmSHZEROCt

\[
\text{Update the product-of-primes matrix [B],}
\]

\[
[B](1,R)/P \rightarrow [B](1,R)
\]

\[
[B](2,C)/P \rightarrow [B](2,C)
\]

\[
[B](3,B)/P \rightarrow [B](3,B)
\]

\[
\text{which gives the state of the sudoku grid}
\]
SUDOKU PROGRAMS

End
Lbl RT
2→J
prgmERASEJ
prgmERASE8
9→J
prgmERASEJ
0→dim(L₁)

PROGRAM: WAIT
If Q≤0
Then
Return
End
If Q>9
Then
Pause
Return
End
For(T,1,625Q)
End

This routine changes the value of the display delay Q.
   ; It checks whether any key is being pressed; if not, it does nothing. If the
   ; decimal point ("full stop") or Enter key is being pressed, Q is set to 10,
   ; so that henceforth the user must press Enter to proceed. If the toggle key 
   ; "(-)" is being pressed, the program goes from "WAIT for the Enter key to be pressed" 
   ; mode to "2-second delay" mode, or vice versa. If one of the digit keys 1-9 is 
   ; being pressed, henceforth the program WAITs for a delay of that many seconds. 
   ; The routine CHKEY is called frequently: in cells after For(R,1,9), and in ENNEADS
   ; after For(I,1,9), that is, the keyboard is checked whenever a new ennead or a new
   ; row of cells is being considered.

Solving an Intractable Sudoku

```
001 004 000
000 060 305
000 900 000
800 000 703
000 000 028
500 070 600
300 080 006
009 200 000
040 001 000
```

The following routines solve the above "intractable" sudoku, which has two pairs of row hypotheses -- 1=(2,4)or(2,8), 8=(6,4)or(6,6) -- and three pairs of column hypotheses -- 1=(5,1)or(8,1), 6=(1,8)or(3,8), 3=(8,8)or(9,8). It also has 9 pairs of box hypotheses, which are detailed in the routine SetMatrxA. The 5 row and cell hyps, which appear to be independent of each other, yield 2⁵=32 starting sudokus to check, but each takes about half an hour to check, and some remain unresolved. So let's try filling-in more cells per sudoku-to-test. The 9 box hyp choices produce 2⁹=512 starting sudokus to check, but let's try out only the last 7 of them, for 2⁷=128 starting sudokus to check. To get started, the intractable sudoku must be stored in matrix [C].

The user can input the matrix by AGTGRID, which put a copy into matrix [C].

PROGRAM: J
   ; Set J to K-th bit of variable 0 ("oh"; rightmost is bit 1)
   int(2fPart(0/2^K))→J

PROGRAM: SETMTRXA
   ; Copy matrix [C] to [A], then modify as specified in O (oh)
   [C]→[A]
   If 0
   Then
   9→K
   prgmJ
   If J

Page 18 of 22
then
  1→[A](2,4)
else
  1→[A](3,5)
end
8→K
prgmJ
if J
  then
    6→[A](1,8)
  else
    6→[A](3,8)
end
end
7→K
prgmJ
if J
  then
    8→[A](6,4)
  else
    8→[A](6,6)
end
6→K
prgmJ
if J
  then
    5→[A](4,8)
  else
    5→[A](5,7)
end
5→K
prgmJ
if J
  then
    8→[A](8,2)
  else
    8→[A](9,3)
end
4→K
prgmJ
if J
  then
    4→[A](7,4)
  else
    4→[A](8,5)
end
3→K
prgmJ
if J
  then
    9→[A](7,6)
  else
    9→[A](9,5)
end
2→K
prgmJ
if J
  then
    6→[A](8,6)
  else
6→[A](9,4)
End
1→K
prgmJ
If J
Then
  3→[A](8,8)
Else
  3→[A](9,8)
End

The following is the main (control) program for searching through the $2^7=127$ versions of the above sudoku in which 7 extra cells are filled in. Before beginning for the first time, do 0→dim(L₅) to empty list L₅, which records the results of trying variant starting sudoku grids.

Note: When OSCAN is done (or is interrupted), sum(int(L₅)) will give the number of seconds it took to check all the sudokus whose O-value is listed in L₅.

Program: OSCAN ; Test the sudoku for each value of O between 0 and $2^7-1$
Prompt O ; Get the starting value of O, e.g., 127
Lbl OL ; "oh loop"
prgmSETMTRXA [or prgmGETMTRXA] ; set [A] to starting sudoku corresponding to O value
prgmAASUDOKU ; solve this sudoku, If possible
  ; Append the result to list L₅, in format t.ss000, where t=how many seconds it took
  ; to check this sudoku, ss = how many passes were made thru the sudoku grid,
  ; ooo = value of the control variable O ("oh") for this sudoku.
(startTmr-A)+S/100+O/10⁵→L₅(1+dim(L₅))
If θ=8
  .008→L₅(1+dim(L₅)) ; If ended unresolved, mark with a following ".008" entry
If θ=0
  Return ; If sudoku has a solution, done (at top level, Return=Stop)
Lbl XO
  ds<(O,0)
goto OL ; ds<("oh","zero") decrement O ("oh") to 0 ("zero")
  ; Counting down is more convenient, because counting up requires you to fill in
  ; "is>(O,...)" with $2^n-1$ for some n, a number that varies with the problem.

Note: For this particular sudoku, counting down from O=127 to zero turned out to be much better than counting up, because the solution was found at O=122, near the beginning of the search. The sudoku for O=119 took a long time to verify as an inconsistent sudoku that has no solution; it took 90:53 with N=0000, 122:19 with N=0001, 54:53 with N=0010, 40:36 with N=0011. Note that with the right settings in N, the shortest time can be only one-third the longest time, but this depends on the particular sudoku.

Solving another intractable sudoku

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>039</td>
</tr>
<tr>
<td>000</td>
<td>001</td>
<td>005</td>
</tr>
<tr>
<td>003</td>
<td>050</td>
<td>800</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008</td>
<td>090</td>
<td>006</td>
</tr>
<tr>
<td>070</td>
<td>002</td>
<td>000</td>
</tr>
<tr>
<td>100</td>
<td>400</td>
<td>000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>009</td>
<td>080</td>
<td>050</td>
</tr>
<tr>
<td>020</td>
<td>000</td>
<td>600</td>
</tr>
<tr>
<td>400</td>
<td>700</td>
<td>000</td>
</tr>
</tbody>
</table>

This sudoku is found to yield 0 cell hypotheses, 1 row hypothesis -- (3=ROW24'5), which is the same as the box hypothesis 3=BOX2(4'5 -- 2 column hypotheses, and 8 box hypotheses. These 8 box hypotheses, which all appear to be independent of each other, are detailed in the following routine GetMtrxA, which is used in OSCAN exactly as SetMtrxA is; just make sure to change "SetMtrxA" in OSCAN to "GetMtrxA". And when
OSCAN asks for a starting value for "oh", give it a countdown starting value in the range 0 to $2^8 - 1 = 255$. And remember to first input the sudoku using AGetGrid, which stores a copy of it in matrix [C].

When using OSCAN, modify ShGrid to display the value of control variable O ("oh").

**PROGRAM: GETMTRXA**

[C]→A ; Copy matrix [C] to [A], then modify as specified in O (oh)

8→K
prgmJ
If J
Then
  3→A(2,4)
Else
  3→A(2,5)
End
7→K
prgmJ
If J
Then
  6→A(2,8)
Else
  6→A(3,8)
End
6→K
prgmJ
If J
Then
  2→A(4,1)
Else
  2→A(6,3)
End
5→K
prgmJ
If J
Then
  4→A(4,2)
Else
  4→A(5,3)
End
4→K
prgmJ
If J
Then
  9→A(5,1)
Else
  9→A(6,2)
End
3→K
prgmJ
If J
Then
  8→A(5,4)
Else
  8→A(6,6)
End
2→K
prgmJ
If J
Then
  8→A(8,1)
Else
   6→A(9,2)
End
1→K
prgmJ
If J
   2→A(7,4)
Then
Else
   2→A(9,5)
End

SUMMARY OF VARIABLES
A: time when the PROGRAM started, in seconds since the Big Bang, modulo 10^{10}
[A]: 9x9 matrix holding the cells of the sudoku grid
B: box number
[B]: 3x9 matrix of prime products
C: column number
D: digit in original array [A] being tested
E: which cell of the I-th ennead of kind K (e.g., E=5,K=3,I=4: 5th cell of box 4)
F: first-found digit that will fit in a given cell (in CELLS, CELLHYPS)
G: second-found digit that will fit in cell (R,C)
H: If zero, not hypothesizing; if nonzero, which hypothesis is being considered
I: number of entry within an ennead of type K
J: with T, used for various very temporary purposes
K: kind of ennead (1=row, 2=column, 3=box)
L: a constant (=2*3*5*7*11*13*17*17*23) telling when an ennead is full (PRIMPROD)
L_1: list of hypotheses, in form {RC.D RC.D ... RC.D} meaning (R,C)=D
L_2: list of cells filled in tentatively under hyp; of form {R C D R C D ... R C D}
L_3: copy of L_2
L_4: records the solution-history, e.g., entry -3456.052 means hyp 3:(4,5)=6 filled in
   5 cells then reached a contradiction (error code θ=2).
L_5: Originally empty; saves the results of checking "derivative" sudokus (see OSCAN)
M: how many times hypothesizing was initiated; frac part = # of verified hyps
N: each digit specifies a user-chosen options (see beginning of AASUDOKU)
O: Used as the control variable in OSCAN; its bits give binary hypothesis choices
P: the relevant prime number (from the list PRIME)
Q: how many seconds to WAIT in the routine WAIT (default is set in AASUDOKU)
when exited, the number dim(L_5) of hypotheses that were generated
R: row number
S: which pass through the sudoku grid is currently being executed
T: used very temporarily for various purposes
U: used for various purposes
V: If 1, indicates that the current hypothesis H is known true (see FILLCELL)
W: number of cells filled during the current pass (W,U used in HYPPAIRS)
X: X,Y denotes the pixel row and column of the upper-left corner of the grid
Y: on the screen; they are constants and must not be altered
Z: number of still-unfilled cells (filled with zero)
θ: error flag; if nonzero, nature of the trouble; if zero, copacetic

The values that θ can assume, and where these values are set
0: Lbl NG (AASODOKU), Lbl Z (AASODOKU) solution found
1: CAN'T FILL (r,c) (Cells)
2: NO! digit D in an ennead (ENNEADS)
3: no hypotheses found (HYPPAIRS)
4: (R,C)=NONE can't fill a cell (CELLHYPS)
5: CELLHYPS error (CELLHYPS)
6: "NO d in ROW/COL/BOX I" (ENNHYPS)
7: EnnHyps error (ENNHYPS)
8: Lbl HF (AASODOKU) hyps failed; sudoku is unresolved (no solution, no contradiction)
9: Lbl BS (AASODOKU) bad sukoku, inconsistent and unsolvable