```
Calc.HowManyUp          PROCEDURE(*BYTE CutGrain,       |
                                  *BYTE HowManyUpWide,  |
                                  *BYTE HowManyUpLong,  |
                                   REAL FullWidth,      |
                                   REAL FullLength,     |
                                   REAL CutWidth,       |
                                   REAL CutLength,      |
                                   REAL BleedTop,       |
                                   REAL BleedBottom,    |
                                   REAL BleedLeft,      |
                                   REAL BleedRight,     |
                                   BYTE MeasureSys)

LocalVars               GROUP,PRE(LOC),AUTO
FullWidth                 ULONG
FullLength                ULONG
CutWidth                  ULONG
CutLength                 ULONG
BaseL1Out                 LONG
BaseL2Out                 LONG
BaseS1Out                 LONG
BaseS2Out                 LONG
BaseLongYield             LONG
BaseShortYield            LONG
TempLong                  ULONG
TempBleed                 REAL
                        .
TMP:CutWidth            REAL
TMP:CutLength           REAL

  CODE

  CLEAR(LocalVars)
  IF FullWidth * FullLength * CutWidth * CutLength = 0
    RETURN(0)
  .
           ! ... This function calculates how many products can be printed
           !     on a specified size sheet. It does not take into account
           !     switch-cuts. A similar function, HowManyOut(), does.
           !     Switch-cuts are excluded because of grain considerations
           !     and watermarks for letterheads.

           ! ... The reason we can't simply add LOC:BleedLeft to LOC:BleedRight
           !     instead of parsing the combined total is that fractions are rounded.
           !     1/16 = 1.588mm, but twice 1/16 (or 1/8) = 3.175mm, not 3.176mm
           !     Keep in mind that the range of Parse.BleedFraction is
           !     1/16" to 1", which limits the individual bleeds (left, right,
           !     top, bottom) to 1/2".

  LOC:TempBleed = Parse.BleedFraction(BleedLeft + BleedRight,MeasureSys)
  TMP:CutWidth  = CutWidth  + LOC:TempBleed
  LOC:TempBleed = Parse.BleedFraction(BleedTop  + BleedBottom,MeasureSys)
  TMP:CutLength = CutLength + LOC:TempBleed

  !Convert REALs to LONGS to avoid fractional errors
  LOC:FullWidth  = FullWidth     * 1000
  LOC:FullLength = FullLength    * 1000
  LOC:CutWidth   = TMP:CutWidth  * 1000
  LOC:CutLength  = TMP:CutLength * 1000

  !If width > length, switch
  IF LOC:FullLength < LOC:FullWidth
    LOC:TempLong   = LOC:FullWidth
    LOC:FullWidth  = LOC:FullLength
    LOC:FullLength = LOC:TempLong
  .
  IF LOC:CutLength < LOC:CutWidth
    LOC:TempLong  = LOC:CutWidth
    LOC:CutWidth  = LOC:CutLength
    LOC:CutLength = LOC:TempLong
  .
  LOC:BaseL1Out      = LOC:FullWidth / LOC:CutWidth     !First, try full width
  LOC:BaseL2Out      = LOC:FullLength / LOC:CutLength   !  divided by cut width
  LOC:BaseLongYield  = LOC:BaseL1Out * LOC:BaseL2Out

  LOC:BaseS1Out      = LOC:FullWidth / LOC:CutLength    !Next, try full width
  LOC:BaseS2Out      = LOC:FullLength / LOC:CutWidth    !  divided by cut length
  LOC:BaseShortYield = LOC:BaseS1Out * LOC:BaseS2Out

  IF LOC:BaseLongYield > LOC:BaseShortYield             !Take the larger of the two
    CutGrain      = eGrainLong
    HowManyUpWide = LOC:BaseL1Out
    HowManyUpLong = LOC:BaseL2Out
    RETURN(LOC:BaseLongYield)
  ELSE
    CutGrain      = eGrainShort
    HowManyUpWide = LOC:BaseS1Out
    HowManyUpLong = LOC:BaseS2Out
    RETURN(LOC:BaseShortYield)
  .
```