```
*** REL (RELative file layout)
*** Document revision: 1.1
*** Last updated: March 11, 2004
*** Compiler/Editor: Peter Schepers
*** Contributors/sources: Immers/Neufeld "Inside Commodore DOS"

  This is a filetype native to CBM drive (random access)  devices.  On  the
surface it seems similar to all other filetypes, especially  SEQ,  but  was
designed to make access to data *anywhere* in the file very fast.

  We start by examining a REL file directory entry...

00: 00 00 84 11 02 41 44 44 49 54 49 4F 4E 41 4C 20  ¨¨Ñ..ADDITIONALˇ
10: 49 4E 46 4F A0 11 0C FE 00 00 00 00 00 00 61 01  INFO†..¸¨¨¨¨¨¨a.

     Bit:$00-01: Track/Sector location of next directory sector ($00 $00 if
                 not the first entry in the sector)
             02: File type.
                 Typical values for this location are:
                   $00 - Scratched (deleted file entry)
                    80 - DEL
                    81 - SEQ
                    82 - PRG
                    83 - USR
                    84 - REL
                 Bit 0-3: The actual filetype
                          000 (0) - DEL
                          001 (1) - SEQ
                          010 (2) - PRG
                          011 (3) - USR
                          100 (4) - REL
                          Values 5-15 are illegal, but if used will produce
                          very strange results. The 1541 is inconsistent in
                          how it treats these bits. Some routines use all 4
                          bits, others ignore bit 3,  resulting  in  values
                          from 0-7.
                 Bit   4: Not used
                 Bit   5: Used only during SAVE-@ replacement
                 Bit   6: Locked flag (Set produces ">" locked files)
                 Bit   7: Closed flag  (Not  set  produces  "*", or "splat"
                          files)
          03-04: Track/sector location of first sector of file
          05-14: 16 character filename (in PETASCII, padded with $A0)
          15-16: Track/Sector location of first side-sector block (REL file
                 only)
             17: REL file record length (REL file only, max. value 254)
          18-1D: Unused (except with GEOS disks)
          1E-1F: File size in sectors, low/high byte  order  ($1E+$1F*256).
                 The approx. filesize in bytes is <= #sectors * 254

  The third byte ($84) indicates this entry is a  REL  file  and  that  the
```

three normally empty entries at offset $15, $16 and $17 are now used as they are explained above. It's the sector chain that this entry points to (called the SIDE SECTORS) which are of interest here (in this case, #17/#12). Here is a dump of that sector...

```
      00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F       ASCII
      -----------------------------------------------  ----------------
0000: 0C 13 00 FE 11 0C 0C 13 06 09 00 00 00 00 00 00  ... ............
0010: 11 02 11 0D 11 03 11 0E 11 04 11 0F 11 05 11 10  ................
0020: 11 06 11 11 11 07 11 12 11 08 11 13 11 09 11 14  ................
0030: 11 0A 11 0B 10 00 10 0A 10 14 10 08 10 12 10 06  ................
0040: 10 10 10 04 10 0E 10 02 10 0C 10 01 10 0B 10 03  ................
0050: 10 0D 10 05 10 0F 10 07 10 11 10 09 10 13 0F 07  ................
0060: 0F 11 0F 05 0F 0F 0F 03 0F 0D 0F 01 0F 0B 0F 00  ................
0070: 0F 0A 0F 14 0F 08 0F 12 0F 06 0F 10 0F 04 0F 0E  ................
0080: 0F 02 0F 0C 0F 09 0F 13 0E 07 0E 11 0E 05 0E 0F  ................
0090: 0E 03 0E 0D 0E 01 0E 0B 0E 00 0E 0A 0E 14 0E 08  ................
00A0: 0E 12 0E 06 0E 10 0E 04 0E 0E 0E 02 0E 0C 0E 09  ................
00B0: 0E 13 0D 07 0D 11 0D 05 0D 0F 0D 03 0D 0D 0D 01  ................
00C0: 0D 0B 0D 00 0D 0A 0D 14 0D 08 0D 12 0D 06 0D 10  ................
00D0: 0D 04 0D 0E 0D 02 0D 0C 0D 09 0D 13 0C 07 0C 11  ................
00E0: 0C 05 0C 0F 0C 03 0C 0D 0C 01 0C 0B 0C 00 0C 0A  ................
00F0: 0C 14 0C 08 0C 12 0C 06 0C 10 0C 04 0C 0E 0C 02  ................
```

```
  Bytes:   $00: Track location of next side-sector ($00 if last sector)
            01: Sector location of next side-sector
            02: Side-sector block number (first sector is $00, the next  is
                $01, then $02, etc)
            03: REL file RECORD size (from directory entry, max. value 254)
          04-0F: Track/sector locations of the six other side-sectors.  Note
                the first entry is this very sector we  have  listed  here.
                The next is the next t/s listed at  the  beginning  of  the
                sector. All of this information must be correct. If one  of
                these chains is $00/$00, then we have no more side sectors.
                Also, all of these (up to six) side sectors must  have  the
                same values in this range.
          10-FF: T/S chains of *each* sector of the data  portion.  When  we
                get a $00/$00, we are at the end of the chain.
```

If the speed advantage regarding this type file file isn't obvious yet, consider the following scenario... If we need to access record 4000, its only a couple of calculations to see how many bytes into the file it is...

4000 * "record length" (254) = byte offset

Once we know this, we can calculate how many sectors into the file the record is...

byte offset / 254 = # of sectors into REL file

The divisor value "254" in the above formula is the number of bytes

```
useable in each sector (256 bytes less the 2 bytes used for the forward t/s
pointer) and has no relation to the "max REL record length".

  Now that we know the number of sectors, we can look it up in our
side-sector tables to see where the record is. The speed of this system is
truly amazing, given the era of the C64, and a floppy drive.
```