External ROMs and Device Drivers

(from QDOS Companion, Chapter 9)

The QL hardware and firmware has been designed for expansion with additional hardware, in the form of external ROMs which may also include extra hardware for peripherals. All extra ROMs have a defined format, so that the QL can recognise them and take action on finding them. There are two areas in which additional ROMs can lie — in the ROM socket at $0C000, or in a peripheral ROM from $C0000 to $FC000 in 16K blocks.

The ROM socket

This can accommodate up to 16K of ROM, and plugs in via the socket on the back of the QL. In the address map it always lies from $OC000 to $OFFFF, and is thus the only place in the memory map that user-written routines can be position dependent.

The peripheral ROMs

Up to 16 peripheral ROMs of 16K each can be added to the QL, though if more than one is added then an additional board, such as the expansion module, is required. As such ROMs can lie anywhere within the memory map, all code within them has to be position independent. They connect via the expansion bus on the left of the QL and can contain ROM, RAM or any other I/O devices, though they must start with ROM for the system to recognise them.

ROM Header Format

To tell if a ROM is connected or not, the system looks for a specific pattern in it, thus:

```
START   DC.L    $4AFB0001       ; identification word
        DC.W    BASPROCS-START  ; start of proc/fn definitions (or 0)
        DC.W    INIT-START      ; initialisation routine (or 0)
        DC.W    NAMELEN         ; length of ROM name
        DC.B    "ROM name",10   ; the name itself + LF
```

BASPROCS should be a list of SuperBASIC procedures and functions to be added to the system, as per usual QDOS documentation for BASIC extensions, or 0 if there are none to be aded. INIT should point to the initialisation routine called after power-up, or 0 if one is not required. The INIT routine will be executed in user mode, and if a successful return to QDOS is required then registers A0 (zero), A3 (points to the start of the ROM) and A6 ($28000) should not be altered. When INIT is called, the system tables and BASIC have been set up, but channel 0 is the only channel open. Other channels should *not* be opened by the INIT routine. Channel 0 is initially the top section of the screen, and all ROM names found are printed in it. For this reason, they should not be longer than 36 characters and should end in a LF character.

Peripheral ROM Problem

While this method is a very neat way of adding ROMs to the QL, there is a problem with peripheral ROMs — in QDOS 1.03 and earlier, there is an error in the 'look for peripherals' routine that terminates the search after checking for one device driver only, so other peripherals are ignored. One way to get around this is for the INIT routine in every peripheral to see if the peripheral is the first, and if it is then it has to carry on the search as the ROM should. This is complicated by the fact that the patch to fix it has to work once the ROM gets corrected. Thus the INIT routine has to look something like this:

```
INIT    BSR     NORMINIT        ; call the init routine
        CMP.L   #$C0000,A3      ; is it the first?
```

```
        BEQ.S   NEXT            ; if so
        RTS                     ; it isn't, so quit
NEXT    ADD.L   #$4000,A3       ; try next one
        CMP.L   #$100000,A3     ; at the end?
        BGE.S   NOMORE          ; if so
* if might be a ROM so let's see
        CMP.L   #$4AFB0001,(A3)
        BNE     NEXT            ; no so do next
        LEA     8(A3),A1        ; start of name
        MOVE.W  UT.MTEXT,A2
        JSR     (A2)            ; print the name
        MOVE.W  4(A3),D0        ; BASIC procedures and functions
        BEQ.S   NOBAS
        LEA     0(A3,D0.W),A1
        MOVE.W  BP.INIT,A2
        JSR     (A2)            ; add the procedures and functions
NOBAS   MOVE.W  6(A3),D0        ; INIT routine
        BEQ.S   NEXT            ; if none
        JSR     0(A3,D0.W)      ; call the INIT routine
        BEQ     NEXT            ; get next ROM
* here when all have been done
NOMORE  ADDQ.L  #4,A7           ; remove return address
        ADD.L   #$1E,(A7)       ; skip over rest of routine
        RTS                     ; and go back to it
```

This fix assumes that the number of bytes taken by the erroneous routine will remain constant from ROM to ROM. If all device drivers take similar action to the above, then there will be no need to fix the bugs anyway, and the number of bytes will remain constant.

Note that, unlike the usual use of BP.INIT, ROM procedures and functions get added *before* the built-in ones, so if there is a name clash then the external ROM version will override the usual built-in ones.