A small description of DV3 drivers

For many device drivers, SMSQ/E uses “DV3” compliant drivers. These drivers may offer a certain number of facilities through well established, and already existing (and coded!) mechanisms, thus immensely reducing the amount of coding a device driver writer will have to do. In many cases, a device driver writer would only have to care for the “physical” routines.

Indeed, DV3 device drivers are normally split into four parts

From lower to higher level, these are :

1 - The physical routines
2 - The filesystem related routines
3 - The I/O and other OS calls related routines
4 - A control thing also used by the basic interface for certain device related commands

The important thing here to realise for a DV3 compliant device driver writer is that he doesn't need to care about levels 2 and 3. The routines for them already exist and can be used automatically.

1 - The physical routines

These are the routines that allow physical access to a drive, the most important ones being those that allow read/writing one sector. SMSQE accesses all common read/write operations to the disk through 2 single routines: Read one sector (dv3_xxx_rsect_asm) and write one sector (dv3_xxx_wsect_asm).

So every operation, even if only for a single byte, will, in the end, always result in one sector being read/written. The advantage is that a driver writer has to implement only these two routines to get, nearly automatically, a fully blown device driver.

There are some other physical routines:
The routine to format a disk (xxx_mformat_asm), to check whether a disk is write protected (xxx_ckwp), whether it is ready (e.g. to check that a drive door is closed), to start/stop a drive and some others. Most of these could be (safely) ignored, in which case the facility provided simply isn't there - for example, it wouldn't be possible to format a drive

It goes without saying that these routines are strictly dependent on the actual hardware to be used (real or emulated).

The files for the physical routines are usually placed in the “dv3_” directory, making an appropriate subdirectory for each machine (e.g. dv3_gold_, dv3_atari, dv3_q40, dv3_qpc_ etc.).

2 - The filesystem related routines

These are the routines that pertain to a specific file system. They contain information about the type of file system, and routines to handle file system specific tasks, such a figuring out how a disk should be laid out; which entry in a FAT would belong to what file; how the root sector is laid out and to be handled, where sectors should be read from and written to etc.

2.1 - Known filesystems

There are 3 groups of filesystems currently SMSQE knows about:

2.1.1 QL5A/QL5B/QLWA

This is the standard Qdos filesystem, used in floppies (QL5A/QL5B) and in hard disks (QLWA). The latter may be real ones (Atari, Qx0) or qxl.win type container files. The files for this can be found in the “dv3_qlf_” subdirectory.

2.1.2 MSDOS

This is the filesystem that can handle MSDOS type FAT drives (including Atari type hard disk partitions). For the time being, only FAT12 or FAT16 formatted disks can be handled.

There are two file systems here:

-- Fat12 and small Fat16

The files that are in the “dv3_msd_” subdirectory can handle FAT12 and small FAT16 drives (not more that 32 MiB) - mostly used, I believe, to read DOS floppy disks and perhaps "small" native partitions on Atari and Qx0 (remember, at the time SMSQE came out, 32 MiB were pretty huge!).

-- Medium sized Fat16

The files that are in the “dv3_msd16_” subdirectory can handle larger FAT16 drives - but this is still limited : the FAT16 partition must be bigger than 32 MiB and smaller then 256 MiB. Actually, this should be a FAT16 partition formatted with 512 byte sectors, each cluster holding 8 sectors.

2.1.3 QUBIDE (experimental)

This will handle the **first partition** of a Qubide formatted disk, either a real one or a container file. The files for this are in the “dv3_qw1_” subdirectory.

2.2 - Using the filesystems

The system should automatically recognize what format a drive is for, without further input from the device driver writer. For this, each filing system contains a "table" :

- qlf_table to be found in “dv3_qlf_table_asm” for QDOS QLWA drives;
- qw1_table to be found in “dv3_qw1_table_asm” for Qubide QLW1 drives;
- msd_table to be found in “dv3_msd_table_asm” for MSDOS FAT12 and FAT16 (small) drives;
- msd_table to be found in “dv3_msd16_table_asm” for MSDOS FAT16 (medium) drives.

(Note that the last two are mutually exclusive).

If a device driver writer wants his device to recognize a specific format, the corresponding table should be specifically included, like so :

```
lea     qlf_table,a1
jmp     dv3_addfd
```

For an example, look at “smsq_atari_driver_dv3_asm”.

The tables are chained tables, so, for example, the qw1_table refers to the qlf_table which in turn refers to the msd_table. By linking in the first, all three are linked in. If only the qlf table is wanted, without the msd table, make a copy of the qlf_table file, delete the link to the msd table in there and include that in the compilation.

3 - The I/O and other OS calls related routines

These are the routines that handle all I/O calls : opening and closing channels, all I/O calls to them (except for formatting), slave blocks, etc. These routines are there and they work – a device driver writer doesn’t need to (re)write them.

4 - The Control Thing

The Control Thing implements the routines for the various basic commands usually associated with a DV3 device, notably XXX_USE. The keywords then use the Control Thing, through the mechanism provided by the “sbext_ut_proc_asm” file. The Control Thing is an extension thing, it can thus also be called from machine code or other high level languages which have a mechanism for calling (extension) things.

The Control Thing is usually called XXX Control, where XXX is the device name, e.g. “WIN Control”.

A template Control Thing can be found in “dv3_hd_thing_asm” implementing many of the keywords. Here one picks and chooses. Some keywords (such as WIN_USE) will make sense for a device, others (such as WIN_SLUG) may not.

5 - Writing a DV3 device driver

Writing a DV3 device driver using an existing file system thus mainly involves writing the necessary physical routines and the initialisation routine for the driver, and then lining them with all the rest of the DV3 system.

As an example for the linking part, look at smsq_gold_dv3_asm and ...link.

The initialisation is done via an “init” routine which uses a pretty standard approach. Templates for this mechanism can be found in “dv3_gold_init_asm” (for floppies), “dv3_q40_hd_init_asm” or “dv3_qpc_hd_init_asm” for hard disks. These contain standard tables which a device driver writer can just copy/adapt.

DV3 device drives also depend on the device definition block being set up in a standard way, and containing information in a standard way at standard offsets. The structure of

that can be found in “dv3_keys” in general, “dv3_hd_keys” for additional offsets for device definition blocks for hard disks, and “dv3_fd_keys” for device definition blocks for floppy disks.

W. Lenerz

NB. This document is probably not exempt from error. If you find any, please point them out to me, so that I can correct them.

Todo : explain the format of the init table and how this is all tied together