Processing of defective blocks

As you know, a flash drive is never perfect. From frequent overwriting, the cells wear out and cease to be reliably recorded. And some cells may be defective immediately after the crystal is made. At the factory, all cells of the chip are thoroughly tested, and defective cells are identified. Note that factory testing is not just about writing and reading sectors. At the factory, the testing conditions are very strict – for example, the process takes place in a temperature chamber at an elevated temperature. The process cannot be adequately reproduced at home, and those cells that have been recognized as defective at the factory can be written perfectly normally in the finished device - but cause problems in the future. Therefore, it is very important not to lose the factory list of defective cells.
Since the minimum erase unit is a block, if there is only one defective cell within a block, the entire block is marked as a badblock in the original terminology. When marking a block as defective, a special marker of the defective block is written to it at the factory — byte 00 in the very first position of the OOB of each page of the block. In good blocks, FF is located in this position, and all values other than FF are considered a sign of a defective block. But, since Qualcomm uses their own format for storing data on a flash drive, the factory marker of the defective block is not used in devices built on this platform. Instead, a byte of data in the middle of each sector (usually offset +1D1 from the beginning of the sector) is used as a token. If it is not equal to FF, then the entire block containing this sector is declared defective. This token is processed by the controller in hardware: when a sector is read, it is not copied to the data buffer, but is transferred to a special register NAND_BUFFER_STATUS, from where it can later be taken for analysis. The position of the marker is configurable programmatically — this value can be seen among the flash drive parameters displayed during the bootloader initialization process:

```
Defective Block Marker Position: user+1d1
```

```
user means that the token is in the sector's data region (if it is located in the OOB area, as is done
at the factory, then spare will be specified instead of user). 1D1 is the offset from the beginning
of the sector to the marker.
```

In the process of reading data, if you ignore the sign of a defective block, then such a block will be read as normal data, but will consist of almost only zeros. If the dump is then written to another device, the block will be written there as a good (not defective) data block consisting of zeros. This will cause a violation of the format of the partition being written, and will cause the device to malfunction. In addition, if you do not pay attention to the signs of defective blocks of the target device, then the factory markers will be erased, and the factory defective block will be lost forever - instead of it, you will get a good block containing data. However, since the unit is potentially defective, the data during storage can easily be distorted and also disrupt the operation of the device.
From all of the above, the following conclusion can be drawn. The method of restoring the firmware of the patient device by uploading a full sector-by-sector copy of the flash drive of another similar donor device (popularly called a fulldump), so loved by some repairmen, is a flawed and quite dangerous method. You can get a completely inoperable device, or a working one, but potentially ready to fail unpredictably at any time. The correct way is to fill in individual sections, skipping the factory defective blocks of both the patient and the donor. And now the qtools complex has finally learned to work correctly with defective blocks.

qbadblock is a utility for working with defective blocks

To search for defective blocks, as well as to manage the markers of individual blocks, the complex has a special item called qbadblocks. Its capabilities are discussed below.

Building a List of Factory Defective Units

The qbadblock program allows you to quickly scan the flash drive for the presence of factory defective blocks, and make a list of them. To do this, use the -d switch:

```
$ ./badblock -d
Building a list of defective blocks in the range 00000000 - 00001000
 Block 00000312 check - badblock (BSPFOTA+5f)
 Block 00000384 check - badblock (APPSBL+4)
 Checking block 0000052a - badblock (SCRUB+151)
 Checking Block 0000080f - badblock (USERDATA+ac)
 Checking Block 00000a2e - badblock (RECOVERY+52)
 Checking Block 00000b0e - badblock (ZTEDATA+d4)
 Checking Block 00000e6a - badblock (RECOVERYFS+19f)
 Checking block 00000f36 - badblock (RECOVERYFS+26b)
 Checking block 00000fab - badblock (RECOVERYFS+2e0)
* Total defective blocks: 9
```

The list contains the absolute number of the block, as well as the number relative to the beginning of the section in which the block is located (if a partition map is available). As usual, you can use the -b and -l switches to set the interval of blocks to be checked. A copy of the list is saved to the badblocks.lst file. I highly recommend that you make this list for it and keep it in a safe place before you start experimenting with a new device. This will allow you to restore the markers of the factory blocks in case they are accidentally erased.

Setting and Removing a Marker

The qbadblocks utility allows you to install or remove a defect marker from any block of the flash drive, except for block 0 (it must always be in good working order). To do this, use the -m and -u switches:

```
qbadblock -m <blk> - marks the blk block as defective
qbadblock -u <blk> - removes the defect marker from the blk block
```

Managing the Marker Position

It may happen that the bootloader incorrectly detects the position of the marker of the defective block during the initialization of the flash drive. In this case, you can force the position of the marker to the desired byte of the sector data field using the -s switch, for example:

```
qbadblock -s 1d1
```

This command will set the position of the token to byte 1D1. The installation will remain in effect until the device is rebooted, or until a new marker position is set.

qrflash - Handling Defective Blocks When Reading

When reading both partitions and an arbitrary set of blocks with qrflash, the mode of processing defective blocks is set using the -u switch.

-us — skip defective blocks. If the block turns out to be defective when it is read, nothing will be written to the output file. This is the default mode for reading partitions – it allows you to get a clean partition dump that can be written to any other device.

-uf — Fill in defective blocks. In this mode, a block completely filled with a byte 0xbb is written to the output file in place of the defective block. Since such data blocks will most likely never be encountered in a real device, it can be guaranteed that there was a defective block on the donor flash drive in this place. This mode is used by default for non-format reading (by block numbers).

-ui — ignore defects. The program will behave as before, reading defective blocks as normal data blocks. This mode seems to have no practical use, although some experts claim that it can save partially killed data. For them, the regime is left.

-ux — disable hardware control of defective blocks. The defective block marker will be read as a normal byte of data and will be included in the output file. This mode is the only one that allows you to make a reliable full image of the donor's flash drive (the same fulldump) that can be written to other devices. However, for the above reasons, the use of fulldumps is highly discouraged. The exception is writing a full-dump to the same device from which it was removed (for a full firmware repair, for example) — in this case, the markers of the badblocks will be written to their native places, and the factory list of defects will not be broken.

qwdirect - Handling Defective Blocks on Writing

The qwdirect utility also has a comprehensive set of tools for handling defective blocks. They are also controlled by the -u switch, but can be set multiple times on the command line to set the desired modes. The key has the following values:

-us - ignore the signs of defective blocks marked in the input file. When qrflash encounters a defective block during reading, it writes a special block consisting of a byte 0xbb to the output dump (if this mode is set). If such a block occurs in the input file, qwdrect simply skips it by default and continues reading the input file. However, when writing a full dump, blocks should not be skipped, because skipping input blocks will automatically violate partition boundaries. Skipping blocks can only be used to write images of specific sections cut from a fulldump. Therefore, by using the -us switch, it is possible to write data without shifting or violating the absolute addresses of the blocks. In this case, garbage (a block of all bytes 0xbb) will be written to the flash drive in place of the defective blocks of the original dump. This will most likely cause the device to partially or completely fail.

-uc - simulate defective blocks of the input file. In this mode, if there is a block in the input file marked as defective (0xbb bytes), then a fictitious defective block is created on the target flash drive. That is, a fully functional and living block will be marked as defective. This allows you (if you specify the -ub switch at the same time) to adequately write someone else's fulldump to your flash drive. At the same time, all defective blocks of the source donor flash drive will be declared defective on the target flash drive as well. Naturally, this mode is quite sneaky - the created pseudo-defective blocks will practically do not differ from the factory ones. And it should be used only in the MOST EXTREME cases, when you need to restore the device quickly and at any cost, and there is only someone else's fulldump in the deposit.

-um - check the correspondence of defective blocks of the file and the flash drive. This is quite a useful mode designed to fill the fulldump into the same flash drive from which it was read. Since the factory defective dump blocks and the flash drive are exactly the same, this method of dump filling allows you to quickly and without losses restore the operation of the device. In this mode, qwdirect, when it encounters a defective block marked in the file, checks whether the corresponding block on the flash drive is defective. If so, the block is simply skipped and the recording continues from the next block. The factory defect marker is not affected. If the corresponding block of the flash drive is not defective, then the writing process crashes, because in this case the dump does not correspond to the flash drive (it is someone else's) and further writing becomes meaningless.

-ux - disable hardware control of defective blocks. In this mode, the entire input dump block is written to the flash drive, including the marker of the defective block. The original dump for such an entry should be made using qrflash also with the -ux switch. This mode also allows you to adequately write the dump to the flash drive from which it was read. But using it is highly discouraged. First of all, it is quite dangerous to perform any actions (and especially erasing) with factory defective blocks — if the block is defective, then it is not a fact that the marker will be adequately written during the next recording. Secondly, in this mode it is impossible to check the compliance of defective dump blocks and flash drives, as a result of which you can accidentally upload someone else's dump to the device and erase all factory markers. It is better to use a dump taken with qrflash -uf as a fulldump and write it with the -um switch.

-ub - do not check the defectiveness of the flash drive blocks before writing. This is a VERY DANGEROUS mode. In normal operation, qwdirect checks to see if the block is defective before writing the next block. With this key, you can disable this check — the target block will be erased and written as a normal block, even if it is defective. This is guaranteed to destroy the factory markers. In fact, this mode is NOT recommended for use. It can only be used during debugging, to overwrite manually created pseudo-defective blocks (e.g. using qbadblock -m). This is the mode used by qwdirect before it had built-in defect block controls. Writing fulldumps using this mode is a crude and unwarranted intervention that leads to a lot of potential problems due to the use of factory defective blocks for data storage.

About fulldumps and cloning a full flash drive.

To simplify their lives, many repairmen and researchers use full dumps of flash drives (fulldumps) to simplify their lives. It is assumed that such a dump allows you to quickly restore the firmware of the device. However, the presence of defective blocks on the flash drive of the donor device and the patient being restored, for the reasons stated above, completely prevents the use of such a simple method.
A distinction should be made between 2 fundamentally different situations.
1. The donor and the patient are the same device, i.e. we are trying to record a full-dump on the device from which this dump was once read. This is the most favorable case. Defective blocks are in the same places both in the dump and on the flash drive, and it is enough to simply bypass them when writing. The fulldump should be read from the device with the qrflash command with the -uf switch (which is what is implied by default when reading without using a partition map). Writing is done with qwdirect with the -um switch. At the same time, the correspondence of defective markers in the dump and on the flash drive is checked, and defective blocks are skipped. If there is a sign of a defective block in the dump, and the corresponding block on the flash drive turns out to be good, or, conversely, there is a defective block on the flash drive that is not marked in the dump, the recording will crash. This will mean that we are trying to fill the device with someone else's dump, or new defective blocks have formed on the flash drive. In any case, an adequate fulldump recording becomes impossible.

2. The donor and the patient are different devices, and we are trying to clone the firmware. If there are no defective blocks on the donor's and the patient's flash drives, there are no problems, the full-dump is counted, it will flood and the patient will revive. If there is at least one defective block on any of the flash drives, then you CANNOT use a fulldump.
But if you can't, but you really want to... You can read the firmware with qrflash -ux, and write it with qwdirect -ux as well. In this case, the defective markers of the donor flash drive will be recorded on the patient's flash drive as is, and the patient's fully alive blocks will be declared defective. Although, in this case, everything will be recorded correctly and the patient will earn. It is worse if there are defective blocks on the patient's flash drive. With this method of recording, the factory markers will be erased, and the real data will be written to the defective blocks. The result of such a recording

is unpredictable — from complete inoperability of the device to enchantingly elusive failures in the operation of the device in the distant future. So this method of cloning firmware should be recognized as extremely unsuccessful. A much more correct way would be to copy the firmware by sections.

Copying firmware by partitions.

Partition copying is the correct way to upload firmware. It is used by proprietary firmware update programs and technological utilities such as QPST. At the same time, all factory defective units are carefully dispensed with without any addressing shifts. Let's take a look at both stages – reading the firmware from the donor device and writing it to the patient device.
Reading is done with the help of the qrflash utility in the mode of working with partitions. First, we read all the sections from the device (of course, from the ALE), skipping the defective blocks:

```
qrflash -s@ -x
```

The -us switch, which enables the mode of skipping defective blocks, is enabled by default in the partition mode — you can choose not to specify it. Then save the device partition map to a separate file:

```
qrflash -s@ -m >part.lst
```

Now all read *.oob file and part.lst file will form a firmware image suitable for uploading to any similar device. You can archive them and put them in storage.
Writing is done with qwdirect, but there will be no full automation like reading. For each section, you will have to give a separate write command:

```
qwdirect -fi -b<start> <file>
```

The number of the starting block can be found in the section map, previously saved during reading. And the partition number in the map is the first 2 digits of the file name that contains the partition dump. In this way, you can write all the partitions, or only some of the partitions that need to be restored. Any defective blocks found on the patient's flash drive will be skipped during the recording. As a result, factory defects will not be lost, and block addressing will not be broken, since the recording does not begin at the beginning of the flash drive, but at the partition boundary.