Handling Bad Blocks

As you know, flash storage is never perfect. With frequent cell erasures, cells can wear out and become unreliable for writing. Some cells may even be defective right after the chip is manufactured. In the factory, a thorough test of all the chip's cells is conducted, identifying defective cells. Note that the factory testing conditions are very rigorous, with processes taking place in a temperature chamber with elevated temperatures. Replicating this process adequately at home is nearly impossible, and those cells that were identified as defective at the factory may write just fine in a finished device but can cause problems in the future. That's why it's essential not to lose the factory list of defective cells.

Because the minimum erasure unit is a block, if there's only one defective cell within a block, the entire block is marked as a defective block (or "bad block" in the original terminology). To mark a block as defective, a special marker is written to it at the factory - the byte 00 in the first position of the Out-Of-Band (OOB) area of each page in the block. In good blocks, this position contains FF, and any value different from FF is considered a sign of a defective block. However, since Qualcomm uses its own data storage format on the flash memory, the factory marker for defective blocks is not used in devices built on this platform. Instead, a data byte in the middle of each sector is used as a marker (usually at offset +1D1 from the beginning of the sector).

If this marker is not equal to FF, the entire block containing that sector is declared defective. This marker is processed by the controller hardware. When reading a sector, it is not copied into the data buffer but is instead moved to a special register called NAND_BUFFER_STATUS, from where it can be retrieved for analysis later. The position of the marker is programmable, and you can find this value among the flash parameters displayed during bootloader initialization.

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

```
"User" means that the marker is located in the data area of the sector. (If it's located in the OOB area, as is done at the factory, it will be labeled as "spare"). "1D1" is the offset from the beginning of the sector to the marker.
```

In the process of reading data, if you ignore the indication of a defective block, that block will be read as regular data but will consist mostly of zeros. If you then write this dump to another device, the block will be written as a good (non-defective) data block, consisting of zeros. This will disrupt the format of the written partition and render the device non-functional. Additionally, if you ignore the indicators of defective blocks on the target device, the factory markers will be erased, and the factory defective block will be lost forever, replaced by a good block containing data. However, since the block is potentially defective, the data can easily become corrupted during storage and disrupt the device's operation.

From all the above, it can be concluded that the method of restoring a device's firmware by flashing a complete per-sector copy of the flash from another similar donor device (commonly known as a "fulldump") is a flawed and potentially dangerous method. It can result in a completely non-functional device or a working one that is potentially prone to unpredictable failures. The correct approach is to flash individual partitions, skipping both the patient and donor's factory defective blocks. Now, the qtools suite has finally learned to work correctly with defective blocks.

qbadblock — Utility for Working with Defective Blocks

For finding defective blocks and managing markers for individual blocks within the qtools complex, there is a special utility called "qbadblocks." The following outlines its capabilities.

Construction of a list of factory defective blocks

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

```
$ ./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
```

In the list, the absolute block number and the number relative to the beginning of the partition in which the block is located

(if the partition map is available) are provided. As usual, you can specify the range of checked blocks using the -b and -l options. A copy of the list is saved in the file badblocks.lst. I strongly recommend creating this list for your device before starting any experiments and saving it in a secure location. This will allow you to restore the factory block markers in case they are accidentally erased.

Setting and Clearing the Marker

The qbadblocks utility allows you to set or clear the defect marker on any block of the flash memory, except for block 0 (which must always be good). You can use the -m and -u options for this:

Use -m followed by the block number to set the defect marker on a specific block.

```
qbadblock -m <blk>   - marks the BLK block as defective
```

Use -u followed by the block number to clear the defect marker from a specific block.

```
qbadblock -u <blk>   - Removes the defect marker from the BLK unit
```

Managing the Marker Position

It can happen that during the flash initialization process, the bootloader incorrectly identifies the position of the defect block marker. In this case, you can forcibly set the marker position to the desired byte within the data field of the sector using the -s option. For example:

```
qbadblock -s 1d1
```

This allows you to manually adjust the marker position to match the specific flash configuration or bootloader behavior.

This command will set the marker position to byte 0x1D1. The change will remain in effect until the device is rebooted or a new marker position is set using the same or a different byte offset..

qrflash - Handling defective blocks during reading

When reading with qrflash, both for partitions and arbitrary sets of blocks, the mode for handling defective blocks is set using the -u flag. The following modes are available:

-us — skip defective blocks. If a block is found to be defective during reading, nothing will be written to the output file. This is the default mode for reading partitions, allowing you to obtain a clean dump of the partition suitable for writing to any other device.

-uf — fill defective blocks. In this mode, the output file will have the entire defective block replaced with blocks filled with the byte 0xbb. Since such data blocks are highly unlikely to occur on a real device, this can be used to guarantee that a defective block was present in that location on the donor flash. This mode is used by default for non-partitioned reading (block numbers).

-ui — ignore defects. The program will behave as usual, reading defective blocks as regular data blocks. This mode may not have practical use, but some specialists claim that it can help salvage partially corrupted data. It is included for that purpose.

-ux — disable hardware control of defective blocks. The defective block marker will be read as a regular data byte and included in the output file. This mode is the only one that allows creating an accurate full dump of the donor flash (the infamous "fulldump") suitable for writing to other devices. However, for the reasons mentioned earlier, using fulldumps is highly discouraged. The exception is writing a fulldump to the same device from which it was extracted (for complete firmware restoration, for example). In this case, the bad block markers will be written back to their original locations, and the factory defect list will not be disrupted.

qwdirect - Handling defective blocks during writing

The qwdirect utility offers a comprehensive set of tools for managing defective blocks, and the -u key is used to control these modes. You can use this key multiple times in the command line to set the desired modes. Here are the different modes available:

us - Ignore defect flags marked in the input file. When qwdirect reads a defective block, and this mode is enabled, it writes a special block filled with 0xbb bytes to the output dump. By default, qwdrect skips these blocks and continues reading the input file. However, when writing a full dump, you cannot skip blocks because it would disrupt partition boundaries. This mode is suitable for writing specific partition images without shifts or violating block addresses. It may result in the replacement of defective blocks with garbage data on the flash memory, potentially causing partial or complete device malfunction.

uc - Simulate defective blocks from the input file. In this mode, when a block marked as defective (with 0xbb bytes) is encountered in the input file, qwdirect creates a fictitious defective block on the target flash device. Essentially, a perfectly functioning block is marked as defective. This mode is used when writing a foreign full dump to your flash device and should be used sparingly. All defective blocks from the source flash will be considered defective on the target flash, creating a difficult situation.

um - Verify the consistency of defective blocks between the input file and the flash memory. This mode is useful for writing a full dump back to the same flash device from which it was read. Since the factory defective blocks in the dump and the flash memory match, this method allows for quick and lossless device recovery. In this mode, qwdirect checks if a block marked as defective in the input file corresponds to a defective block on the flash device. If it does, the block is skipped, and the writing process continues without affecting the factory defect marker. However, if the corresponding block on the flash memory is not defective, the writing process is terminated abruptly as the input dump does not match the flash device, making further writing pointless.

ux - Disable hardware control of defective blocks. In this mode, blocks from the input dump are written to the flash memory entirely, including the defective block marker. While it allows for adequate writing of a dump to the same flash device from which it was read, it is not recommended. First, handling any operations (especially erasing) with factory defective blocks is dangerous because there's no guarantee that the marker will be written correctly during the next write. Second, in this mode, it is impossible to maintain control over the consistency of defective blocks between the dump and the flash memory, potentially leading to accidentally overwriting your device with someone else's dump and erasing all factory markers. It is advisable to use a dump created with qrflash -uf as a full dump and write it with the -um key.

ub - Skip the checking of defective blocks in the flash memory before writing. This is an EXTREMELY DANGEROUS mode and should be avoided. In normal operation, qwdirect checks whether a block is defective before writing. Using this key disables that check, allowing the target block to be erased and written as a regular block even if it's defective. This is guaranteed to destroy factory markers. This mode is strongly NOT recommended for regular use. It can only be used during debugging to rewrite manually created pseudo-defective blocks (e.g., using qbadblock -m). This mode was used by the qwdirect program before the introduction of built-in defective block control tools. Writing full dumps using this mode is a crude and unjustified intervention that can lead to numerous potential problems due to using factory defective blocks for storing data.

This section discusses full dumps and cloning of a complete flash drive.

Many repair technicians and researchers use full sector-by-sector flash dumps, often referred to as "fulldumps," to simplify their work. It is assumed that such a dump allows for quick firmware recovery for a device. However, the presence of defective blocks on the flash drive of both the donor device and the target patient device can significantly hinder the use of this straightforward method.

There are two fundamentally different situations to consider:

Donor and patient are the same device, meaning you are attempting to write a fulldump to the device from which it was previously read. This is the most favorable scenario. Defective blocks on the flash drive are in the same positions in both the dump and on the flash drive, and they can be easily bypassed during the writing process. You should read the fulldump from the device using the qrflash command with the -uf flag (which is implied by default when reading without using a partition map). Writing is done using qwdirect with the -um flag. During this process, the matching of defective markers in the dump and on the flash drive is checked, and defective blocks are skipped. If a defective block is detected in the dump, and the corresponding block on the flash drive is good, or vice versa, if a defective block is encountered on the flash drive that is not marked as defective in the dump, the writing process will fail. This would indicate that you are attempting to load someone else's dump into the device, or that new defective blocks have appeared on the flash drive. In either case, it becomes impossible to adequately write the fulldump.

The donor and the patient are different devices, and you are attempting to clone the firmware. If there are no defective blocks on either flash drive, there are no problems, and the fulldump can be read and written, successfully reviving the patient device. However, if there is at least one defective block on either flash drive, using the fulldump is NOT

RECOMMENDED.

But if you can't resist the temptation... You can read the firmware using qrflash -ux and write it using qwdirect -ux. In this case, the defective markers of the donor flash drive will be written to the patient's flash drive as is, and perfectly functional blocks of the patient will be marked as defective. Nevertheless, the process will complete correctly, and the patient device will work. The situation becomes worse when defective blocks exist on the patient's flash drive. With this method of writing, the factory markers will be erased, and real data will be written to defective blocks. The outcome of such writing is unpredictable, ranging from complete device malfunction to severe operational failures in the distant future. Therefore, this method of firmware cloning should be considered extremely inadvisable. A much better approach is to copy the firmware by partition.

Copying firmware by partition.

Copying by partitions is the correct method for flashing firmware. This is the method used by official firmware update programs and technological utilities like QPST. It allows you to work with factory defective blocks without any address shifting. Let's consider both stages - reading firmware from the donor device and writing it to the recipient device.

Reading is done using the qrflash utility in partition mode. Initially, all partitions are read from the device (typically, in OFP format), while bypassing defective blocks:

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

The -us key, which activates the mode for skipping defective blocks, is enabled by default when working with partitions. You don't need to specify it explicitly. Then, save the device's partition map to a separate file:

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

Now, all the read files with the .oob extension and the part.lst file will constitute a firmware image suitable for flashing onto any similar device. You can archive them and store them for future use.

Writing is done using qwdirect, but there won't be full automation as in reading. You'll need to give a separate write command for each partition:

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

You can find the starting block number in the partition map that was saved during the reading process. The partition number in the map is the first two digits of the file name containing the partition dump. This way, you can write all partitions or only the ones that need to be restored. Any defective blocks found on the recipient device's flash memory will be skipped during writing. As a result, the factory defects won't be lost, and the block addressing won't be disrupted since writing begins at the partition boundary, not from the beginning of the flash memory.