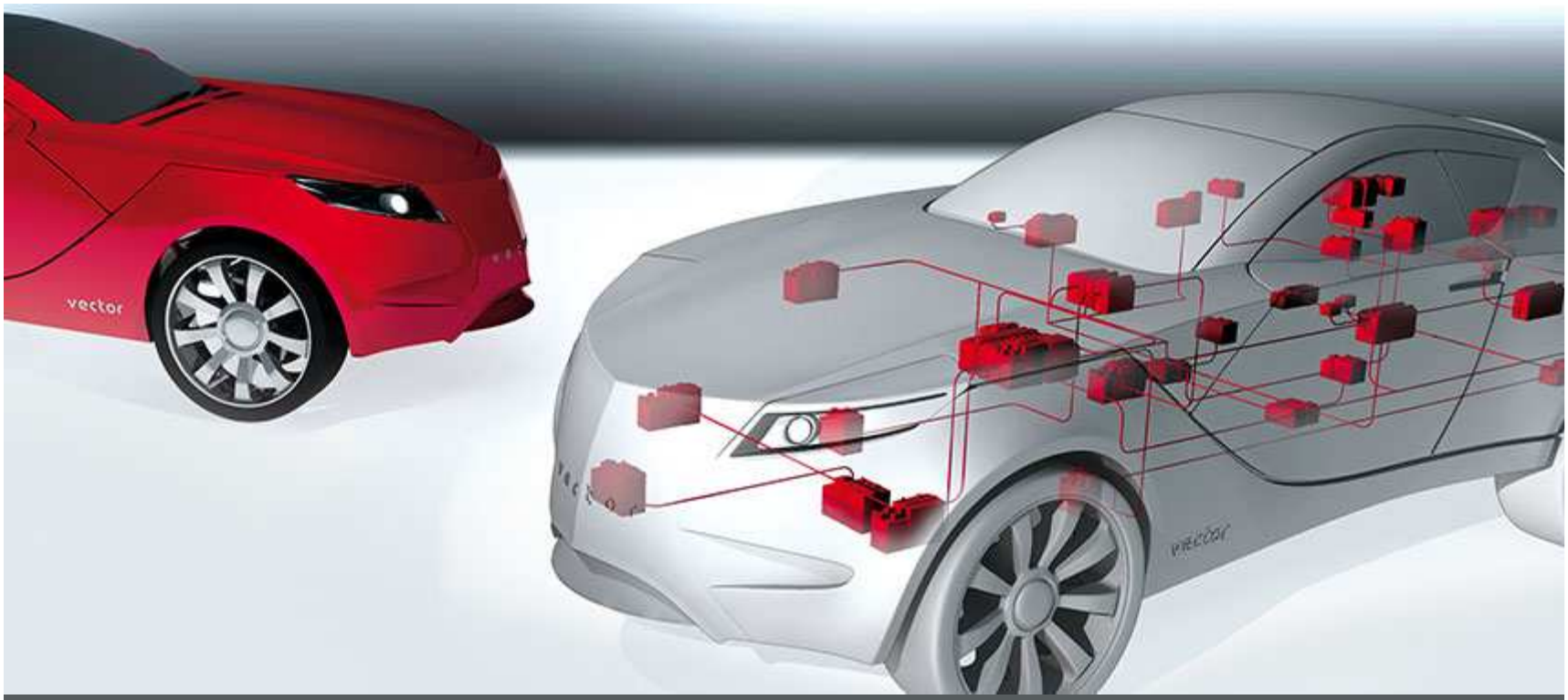
VECTOR



Extension of ASAM MDF for channel-oriented storage

ASAM Workshop 2016-07-05 in Munich

V0.1 | 2016-06-16

VECTOR

Overview

- Motivation
- Requirements
- Proposed Solution
- Compatibility
- Special Use Cases
- Discussion

VECTOR

Motivation

- MDF stores signal values in records
  - typically one time stamp and one or more signal values + invalidation bits (optional)
  - no duplication of time stamps (smaller file size)
  - fine for online-writing use case
- For offline evaluation, typically all (or a range of) consecutive signal values must be read
  - ODS offers API to receive signal values as array
- Reading all signal values from MDF requires
  - either reading each record and extracting the signal value,
  - or alternating read and seek operations to read only the relevant bytes of the signal value
- Lower performance than for "read-optimized" formats (like HDF5, …) where the values are stored consecutively
  - especially for records with many signals unnecessary bytes must be read

=> Aim: Transform MDF file for optimized reading (write once, read many)

VECTOR

Requirements

- Storage of signal values in such a manner that reading is faster
  - ideally all signal values are stored consecutively
- No loss of information (same information as in source file)
  - only re-organization of MDF file
- Each signal value still can be associated with a time stamp
  - avoid duplication of time stamps
- Try to stay compatible to existing MDF 4.1 format

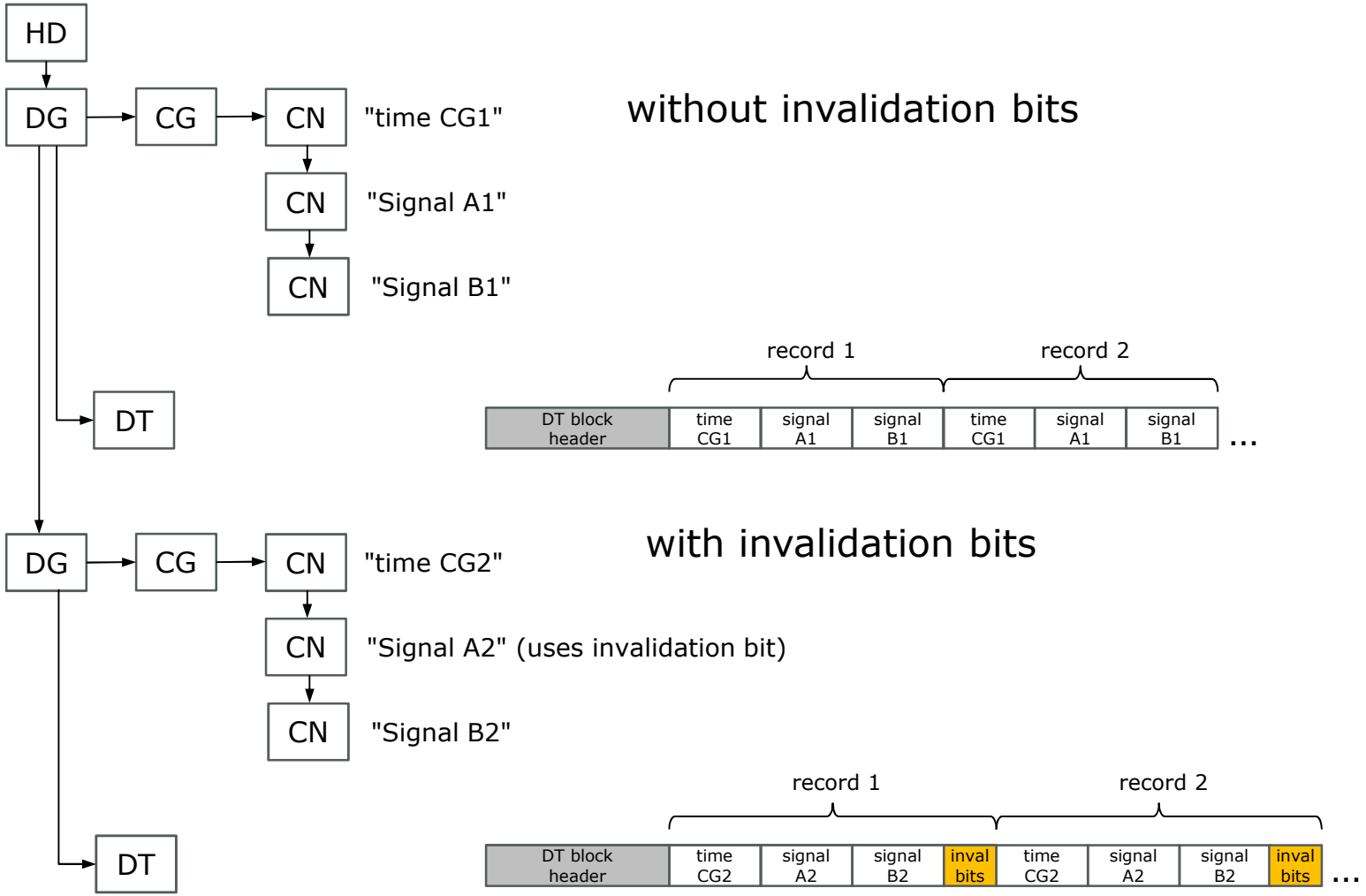
VECTOR

Proposed Solution: Re-organization of MDF file

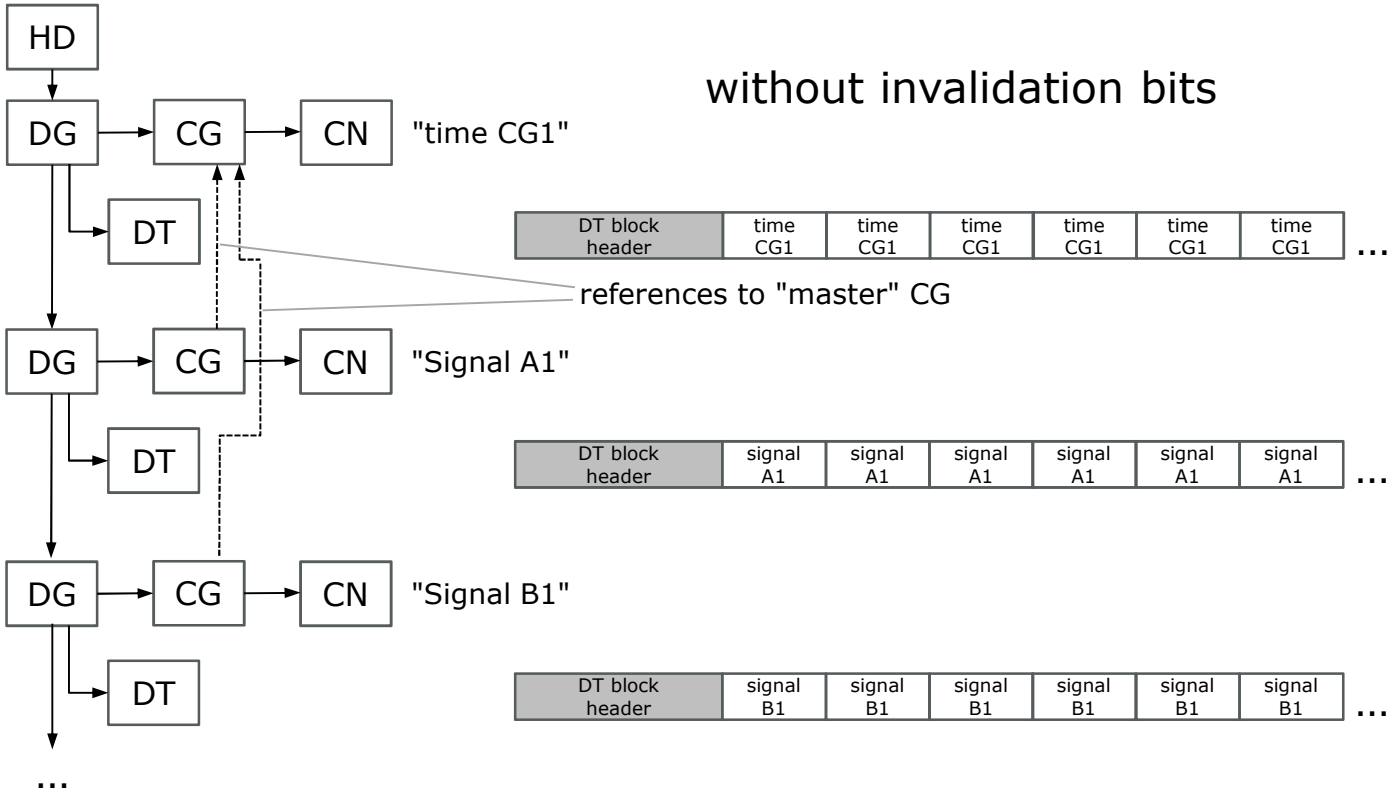
- One channel (CN) per channel group (CG)
  - without invalidation bit, the record only contains the signal value
    - signal values are stored consecutively (except when using distributed data blocks)
  - with invalidation bit, one extra byte per record is required
    - between each signal value, there is the extra byte for the invalidation bit

- Master channels (time, angle, distance) are stored equally
  - master channels cannot have invalidation bits
    => time stamps are always stored consecutively
- Introduce a new link in CG block to reference the CG with master channel
  - both CGs must have equal number of samples
  - multiple master channels
    - CG of first master channel references the next CG with next master channel
    - alternative: "master" CG contains all master channels (record-based storage)
- In case of equidistant time stamps, a virtual master channel can be used
  - either in own CG or in each CG with single CN block for signal

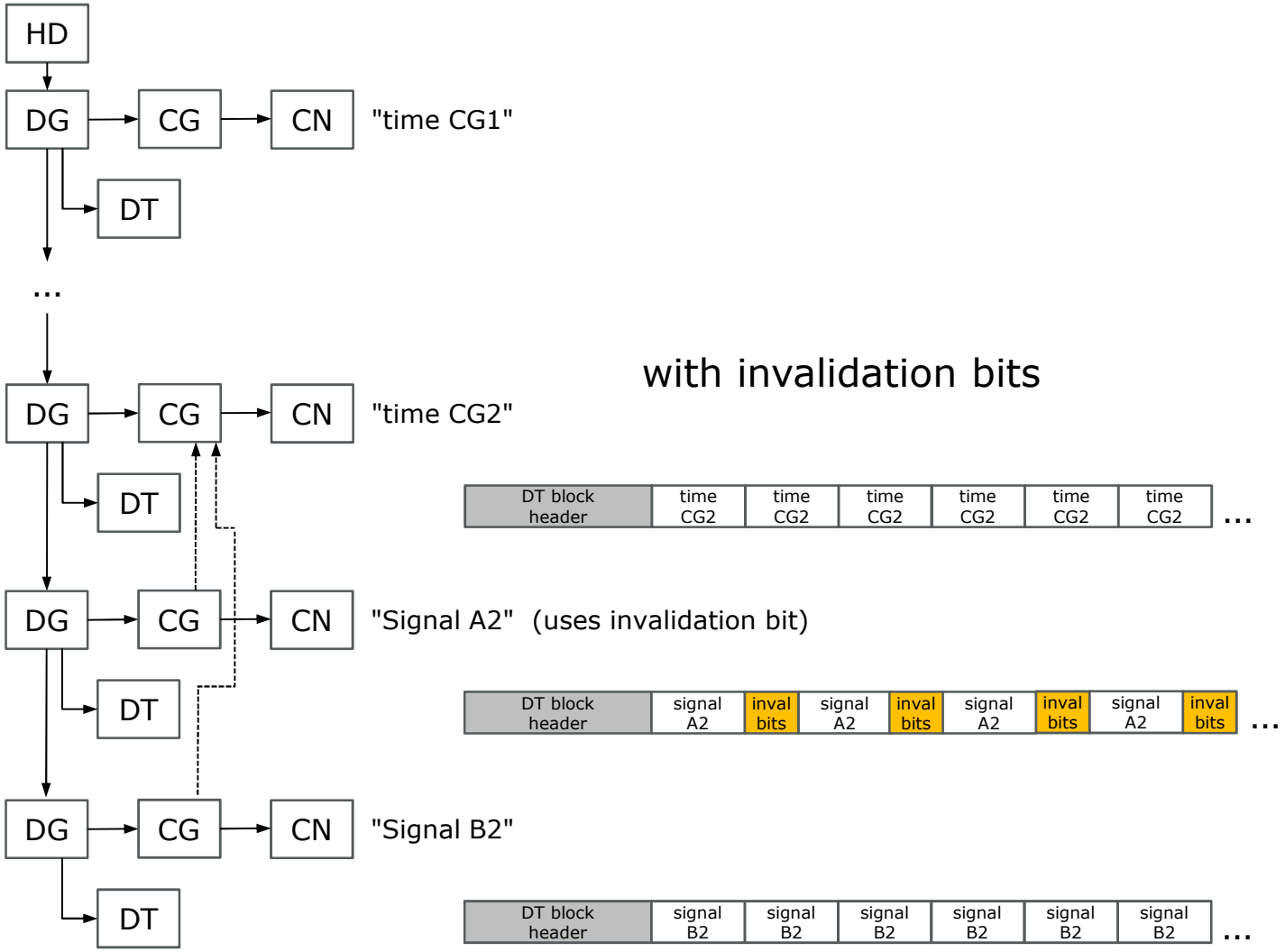
Example for sorted MDF file (with record storage)



Example for channel-oriented storage (first group)



Example for invalidation bits (second group)

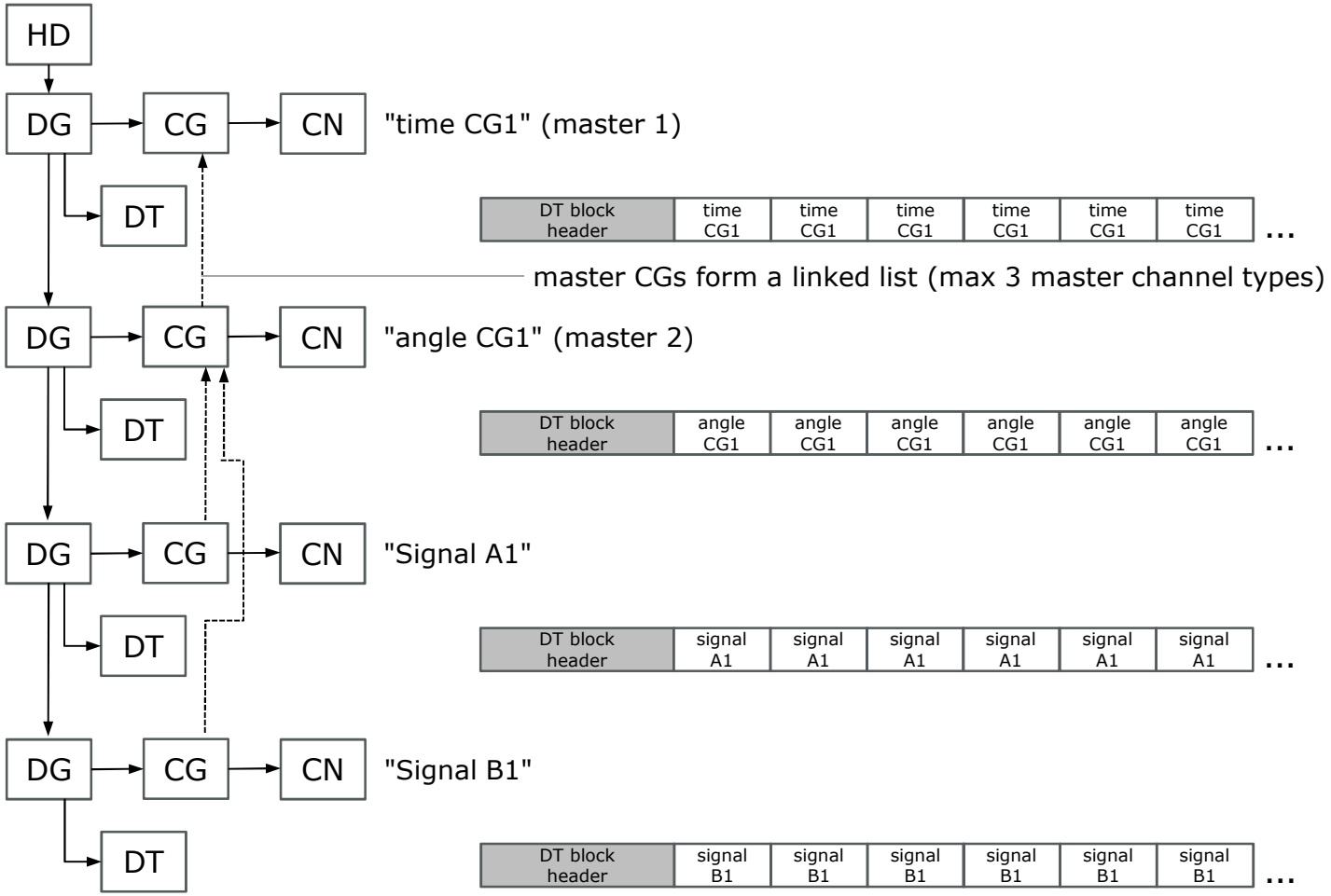


VECTOR

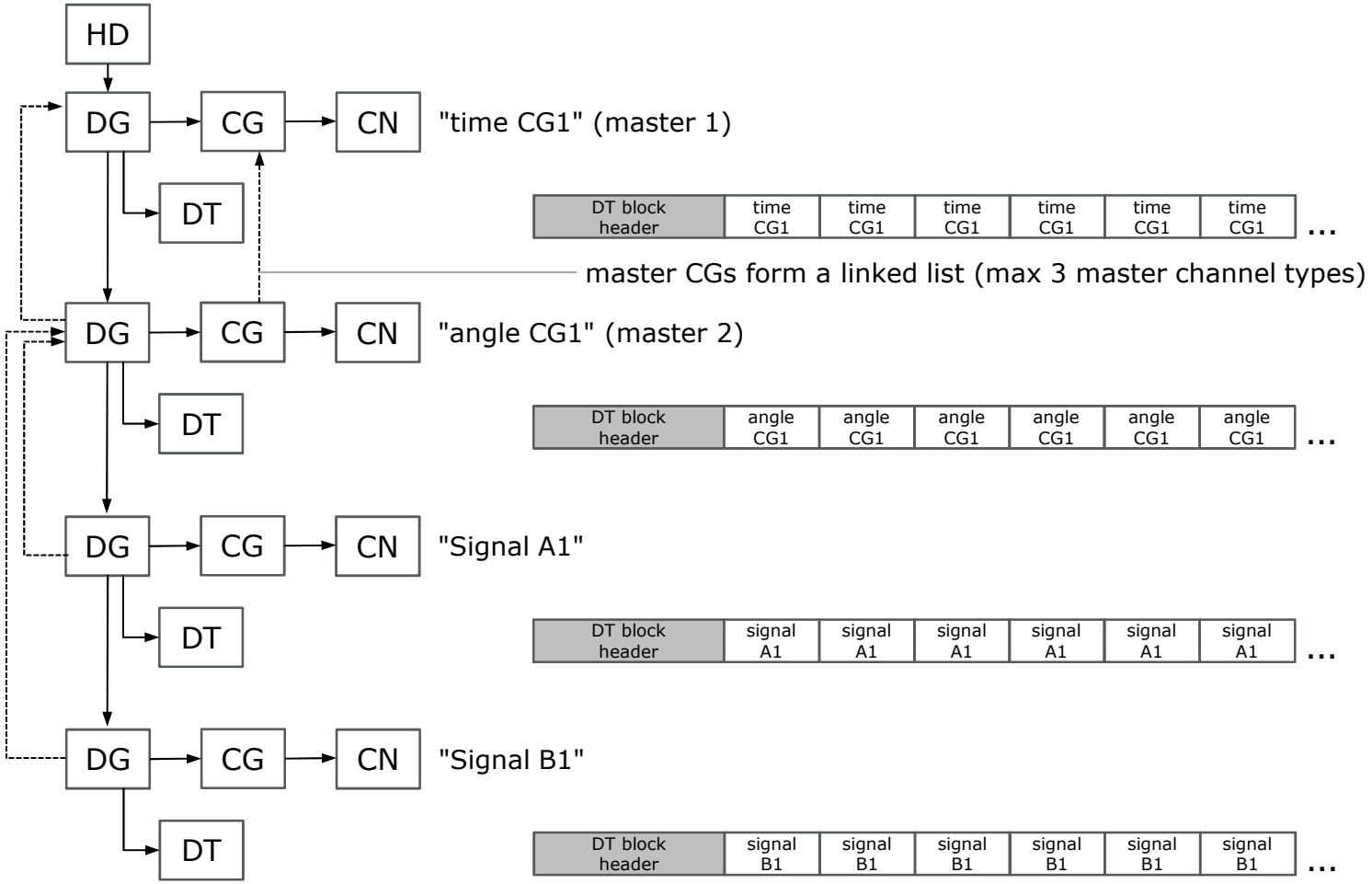
Reference to master channel(s)

- Possibilities
  - New (optional) link in CG block to reference "master" with master channel
    - could introduce a new bit in cg_flags to indicate "channel-oriented" storage and reference to "master" CG
    - flag is required in case new link should be optional
  - Possibilities for multiple master channels
    - "master" CG could contain all master channels
    - several master CGs as linked list (define strict ordering?)
    - own link to master CG for each sync type (max 3 additional links)
  - Instead of each CG pointing to the "master" CG, we also could model a linked list for all CGs "sharing" the master channel(s)
    - not to be confused with linked list of CGs in case of unsorted DG
    - recycling of cg_cg_next link confuses old tools => should be a new link
  - Instead of CG we could reference the DG (assuming sorted DG/MDF file)
    - MDF only stores parent -> child link, but not vice versa
    - easier to find the matching DG block

Example for multiple master channels



Example for link to DG



VECTOR

Special use cases

- Arrays (CA block)
  - Use "DG template" (one data block for each element of array)
    => values for each element are stored consecutively
    (unless invalidation bit is used)

- Structures
  - "Compact" structure (byte range with all member values)
    => reading each member value still is like extracting value from record
  - A structure alternatively can be model as "fragmented" structure with a channel hierarchy (CH) block
    => each member channel can be in own CG
    (except if bytes for channels "overlap" => data duplication required)

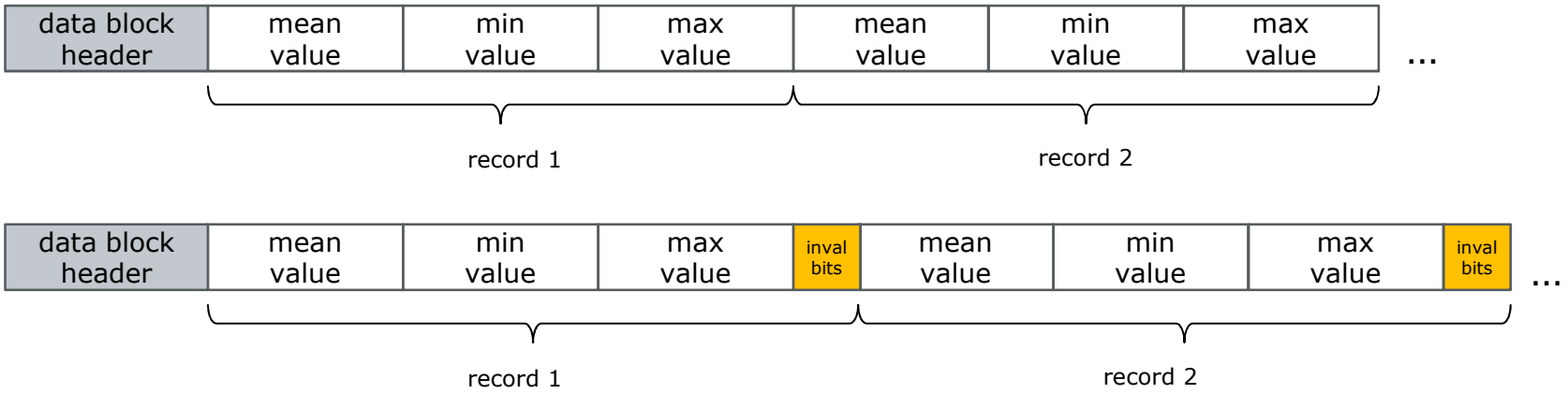
- Variable length signal data (VLSD)
  - Signal value defines start offset in SD block => no change

- Virtual channels
  - suggestion: store in "master" CG

VECTOR

Sample Reduction

- Data block for SR contains still records
  - mean, minimum and maximum value of channel



- Possibly introduce a new kind of sample reduction => discussion required
  - could only store one type of value (min/max/mean)
  - could also omit mean value (not required by some tools)

VECTOR

Data Blocks

- Distribution of data blocks is still possible
  - signal values are only consecutively stored within one data block
  - maybe disallow splitting of a signal value (i.e. record) for this use case

- Compression of data blocks is still possible
  - transposition before deflate may have less effect

VECTOR

Compatibility

- Old tools may only see the single signal without time channel
  - signal values can be displayed with index-based axis
  - full compatibility in case of virtual master channel (in CG of signal)

- For compatibility, we could use the "Default X" channel reference (MDF 4.1)
  - assign a reference to the (primary) master channel stored in other CG
  - conflict in case a real X channel reference (e.g. for XY diagram) is used

VECTOR

Additional Benefit

- The shown solution also offers to implement the "add channel" use case
  - Assume we want to add a channel with equal time raster as an existing channel group (e.g. result of a calculation)
    - > Either we need to extend each record which requires rewriting of complete records for original CG
    - > Or we need to duplicate the master channel values
  - With suggested solution, simply add a "channel-oriented" CG and reference the original CG with the master channel(s)
    - > only new blocks need to be appended to existing MDF file (+ extending linked list of DG blocks, i.e. change one reference)

VECTOR

Disadvantages of the proposed solution

- Increased complexity for MDF format
  - always for new feature
  - maybe additional complexity due to compatibility
- Duplication of DG/CG blocks
  - additional size is less than 200 bytes
  - blocks for comment and source info block (TX/MD/SI) can be shared
- Cycle count should be equal for channel CG and "master" CG
  - try to describe a (sensible) instruction how to handle such a mismatch
- Usage of invalidation bits
  - Signal values are not stored consecutively any longer
    - but invalidation bits must be read anyway?
  - 1 bit is blown up to 1 byte
    - could even use 2 bytes to model ODS structure
- No 100% Compatibility
  - Older tools may not see the time channel (e.g. no support for default X)
  - However: same problem as for compression introduced in MDF 4.1

VECTOR

Summary

- Suggested solution offers
  - fast read access for consecutively stored signal values
  - simple re-organization of file (no loss of information)
  - support for special use cases (arrays, structures)
  - compatibility to MDF 4.1
  - additional benefit by supporting "add channel" use case

VECTOR

Questions from Open Discussion

- Since suggested solution is not 100% compatible:
  - Should we allow old tools to display signal values for CN-oriented storage without time channel (index-based)? Default X may not be supported…
  - Or - for security - should we take care that old tools cannot display the signal value to avoid misinterpretation?
- Try to stay compatible (MDF 4.2) or release new major version (MDF 5.0)?
  - MDF 5.0 automatically prevents display in older tools and may allow other (easier/more efficient) solutions for CN-oriented storage
  - MDF 5.0 may have acceptance problems
    - new file extension mf5
    - not readable with old tools
  - Release of MDF 5.0 should include further features
    - more implementation effort, short-term release unlikely
- Huge number of DG/CG blocks may be slow for reading?
  - AVL suggests using reference to own data block from CN
- Revise invalidation bits
  - introduce new bits / harmonize with bits from ODS?
  - try to store invalidation bits for fast access (consecutive values)?

For more information about Vector
and our products please visit

www.vector.com

Author:
Schneider, Otmar
Vector Germany