IEEE 1722 GPIO
Changes to GBB for supporting it

David Ortiz

IEEE 1722b Task Force – 5th July 2022

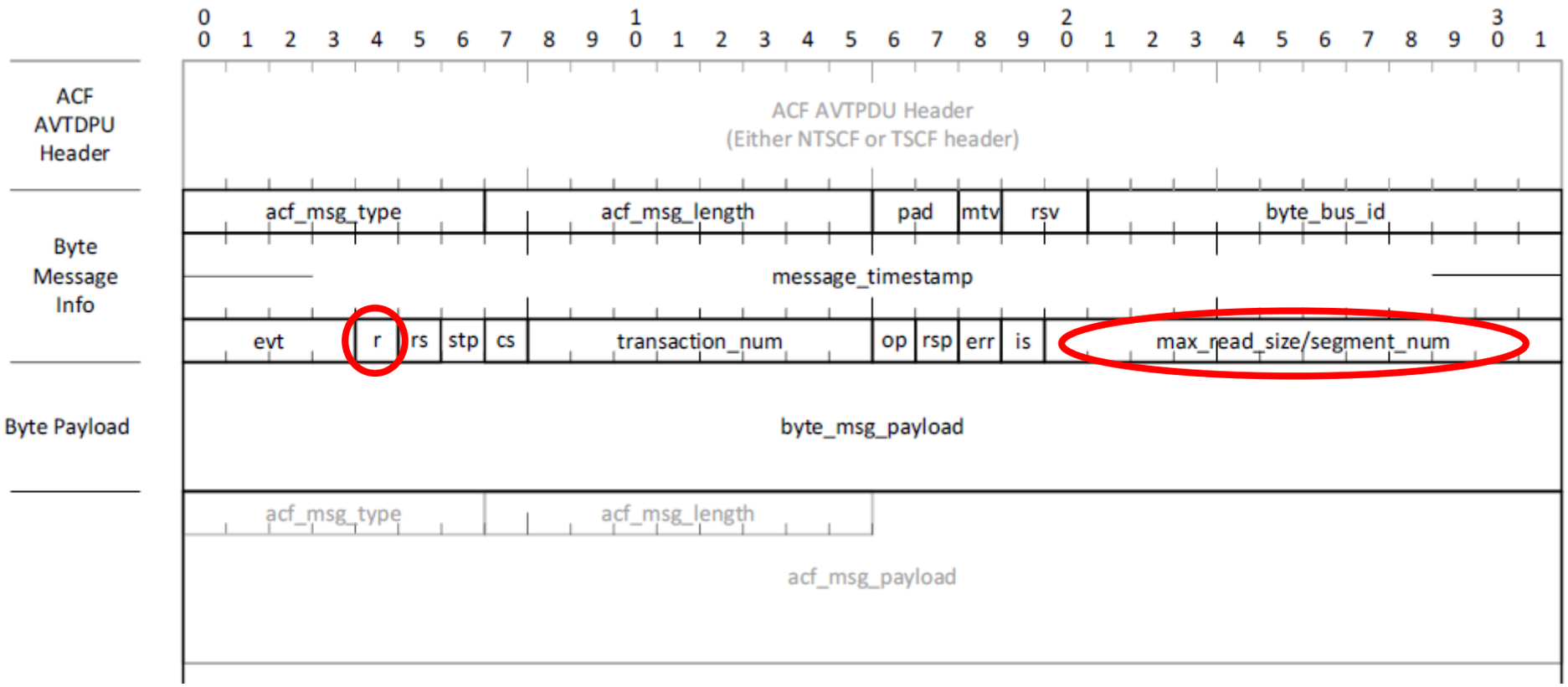
Introduction & objectives I

- In previous presentation the option to cover GPIO functionality through a newly defined GPIO ACF was described.
  - The key functionality offered by the new GPIO ACF was possibility of having asynchronous GPIO value updates from Sensor node. The other features could be handled by proper definition of a Memory-mapped GPIO device. The memory-mapped GPIO device could be managed through existing ACF messages.
- Instead of defining a new GPIO ACF, an alternative would be adding possibility for refresh (asynchronous) updates on other existing protocols.
- From the existing defined ACF the best option would be to modify the GBB protocol.
- Analysis based on latest description available at the private area (1722b-Clause9-ver01_brian_edem.pdf).
- Comment resolution on the max_read_size conversion to ‘read_size’ also considered and needed. In any case here field is still referred to max_read_size.
- This is presented on the next slides.

Current GBB ACF Limitations

- Current GBB description limits its application to a request/response scenario:
  - Host (Talker according to 9.4.14.10) sends a request message (read or write) to the device.
  - The device (Listener according to 9.4.14.10) sends a response back to the host.
  - In case of write requests, the response is optional.
  - In case of read requests and when the response is big, there might be several responses, each one having a different segment_num field.
    - Every request/response is unique (different values on rsp or segment_num, or transaction_num). Duplicated messages are discarded.
- With current description it is not possible to provide ‘refresh responses’ where the Listener/sensor could send messages back to the Host when its internal state (the values sampled on GPIO inputs) changes.

GBB ACF modifications I



- *Redefine ‘r’ (reserved) bit to ‘rfs’ refresh.*
- *Change description of max_read_size/segment_num when rfs bit is set. Refresh messages must always fit in a single ACF to allow redefining of the field meaning in this case.*
- *The same changes would be applied to the abbreviated GBB message format.*

GBB ACF modifications II

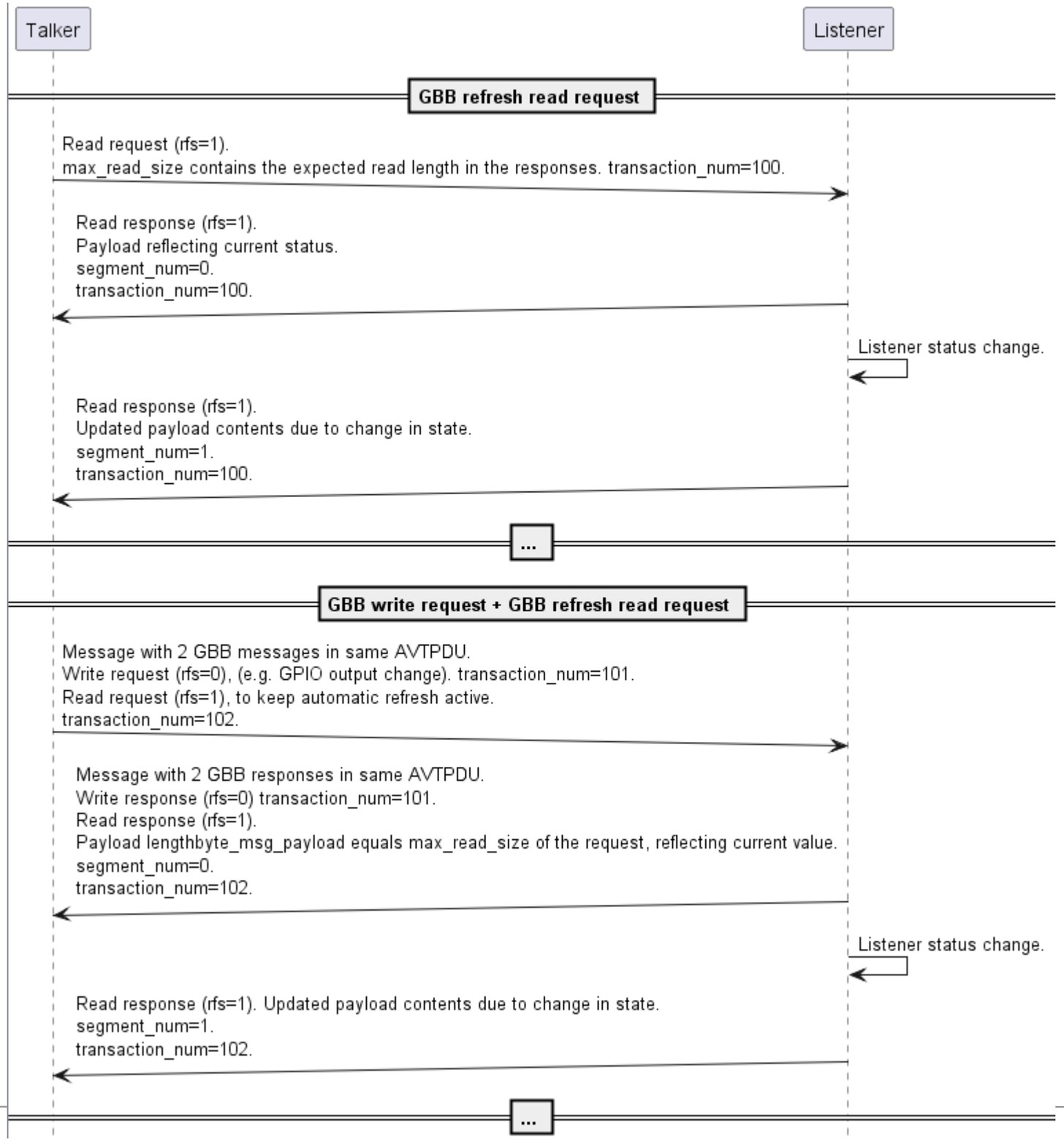
- *rfs: This bit is used to manage refresh messages. It can only be set in read messages.*
  - *When this bit is set to 1 on a request read message, it is used by the Talker to enable the refresh functionality on the Listener. In this case the read_size field of the message must be small so the associated read data fits in a single ACF response message.*
  - *The listener will set the rfs bit in all subsequent responses to the request message.*
  - *The different messages sent by the Listener will have incrementing values on the segment_num field of the message, so it is possible to identify every single message, detect lost messages, or discard repeated messages.*
  - *The first refresh message contains the response corresponding to the ‘initial state’ of the Listener.*
  - *If the state of the listener changes, it might send a new refresh message, incrementing the segment_num value by 1.*
  - *When the Talker sends a new request message having the same byte_bus_id, the Listener shall stop sending refresh response messages associated with the previous request. Messages with different byte_bus_id do not have any impact on the operation of the refresh functionality.*

GBB ACF modifications III

- *max_read_size/sequence_num:*
  - *The description of this field must be changed to reflect the new functionality when the rfs bit is set.*

- *The way implementors might use this new ‘refresh response’ functionality of GBB would be outside the standard.*
- *An example use case of the refresh functionality might be added to an informative annex if desired. That example might show how to use that functionality for a memory-mapped GPIO block.*
- *The use of the new ‘refresh read’ functionality is not limited to GPIO, it could be useful in other scenarios.*
- *The main advantage of supporting refresh functionality is to limit the bandwidth occupation of the channel due to polling, without compromising critical latency.*

GBB ACF use case



Thank you!