==========

Add *the following entries to Table 22 – ACF message types. Other entries not shown*.

Value	Name	Description	Subclause
$0E_{16}$	ACF_I2C	Inter-IC (I2C) bus message	9.4.16
$0F_{16}$	ACF_I2C_BRIEF	Abbreviated I2C bus message	9.4.17
10_{16}	ACF_SPI_SAFE	Serial Peripheral Interface (SPI) message specified by SAFESPI [https://safespi.org/]	9.4.18

========== contribution below recreated from posted PDF ==========

[EDITOR's NOTE from 1772b-I2C-contrib-bwy-v9-FINAL to this document v10. – DELETE after presentation to IEEE 1722.
1. Bits in header that lacked explicit and verbose polarity value definition now have such statements. Cause: similar bus signals in active low may confuse the reader, e.g. RD on bus vs rd in header. Avoid potential confusion.
2. Two types of read – read with early (ack) response and read with ack and data response are the controller's choice with no clear benefits for having the two variations. Remove read with early (ack) response to simplify operation.
Consequence: In Table in 9.4.16.19, the variable "RR" is replaced with "1", and references to "early response" operation in text is deleted. And Diagram in the same sub-clause revised.]

[BEGIN, not a part of the contribution to this draft]

9. Level 1

9.4. Level 2

[END, not a part of the contribution to this draft]

9.4.16. I2C message

9.4.16.1. Overview

The Inter-Integrated Circuit (I2C) ACF message provides the capability to transport I2C transaction events over a TSN network. An I2C device may be a Controller or a Target. For a given I2C bus cycle, there is only one Controller and typically there is only one target that responds. The bus cycle signals are transported as messages encoded in ACF over a TSN network.

This I2C ACF request message transparently conveys the I2C transaction bus state of the Controller over a network onto the Target's I2C bus, and any responses from the Target's bus onto the Controller's bus, each with up to one byte of data per message.

I2C protocol allows a Target device to slow the transfer rate of a transaction using a mechanism often referred to as "clock stretching" so that a slow Target device may acquire or retrieve the data and then respond with the data. This mechanism allows the Controller's bus to be stalled while waiting for a response message from the Target's bus.

The I2C ACF message protocol is designed to maximize compatibility to legacy I2C bus operation between I2C devices over a network at the cost of bandwidth and payload efficiency. See 9.4.16.19 for an illustration of this transparent mode of operation, and see Figure 78 illustration of Controller, Controller Agent, Target, and Target Agent.

For a more modern system implementation of I2C transport where the legacy I2C bus cycle compatibility is not required, the Generic Byte Bus (GBB) ACF message (9.4.14) may be used to transfer one or more bytes in a single ACF message. The use of GBB would also provide better network bandwidth utilization and minimize processing overhead.

[Editor's note –insert a reference to I2C spec, if one is available. If not, be silent.]

The I2C message inherits the following fields from the Common ACF message described in 9.4.1:

- **acf_msg_type**: 7 bits
- **acf_msg_length**: 9 bits

The I2C message has an **acf_msg_type** field of ACF_I2C (see Table 22).

The I2C message defines the following message-specific fields:

- **pad** (padding length): 2 bits
- **mtv** (message_timestamp valid): 1 bit
- **str** (start): 1 bit
- **stp** (stop): 1 bit
- **i2c_bus_id**: 11 bits
- **message_timestamp**: 8 octets
- **wr** (write): 1 bit
- **akv** (ack valid): 1 bit
- **ack** (acknowledge): 1 bit
- **rdv** (read data valid): 1 bit
- **c2t** (Controller to Target): 1 bit
- **rd** (read): 1 bit
- **trr** (target response 5 request): 1 bit
- **rsv** (reserved): 1 bit
- **transaction_num**: 8 bits
- **evt:** 4 bits
- **exception_codes: 4** bits
- **reserved:** 8 bits
- **i2c_msg_payload**: 0 to 1 quadlets

These fields are shown in Figure 77.

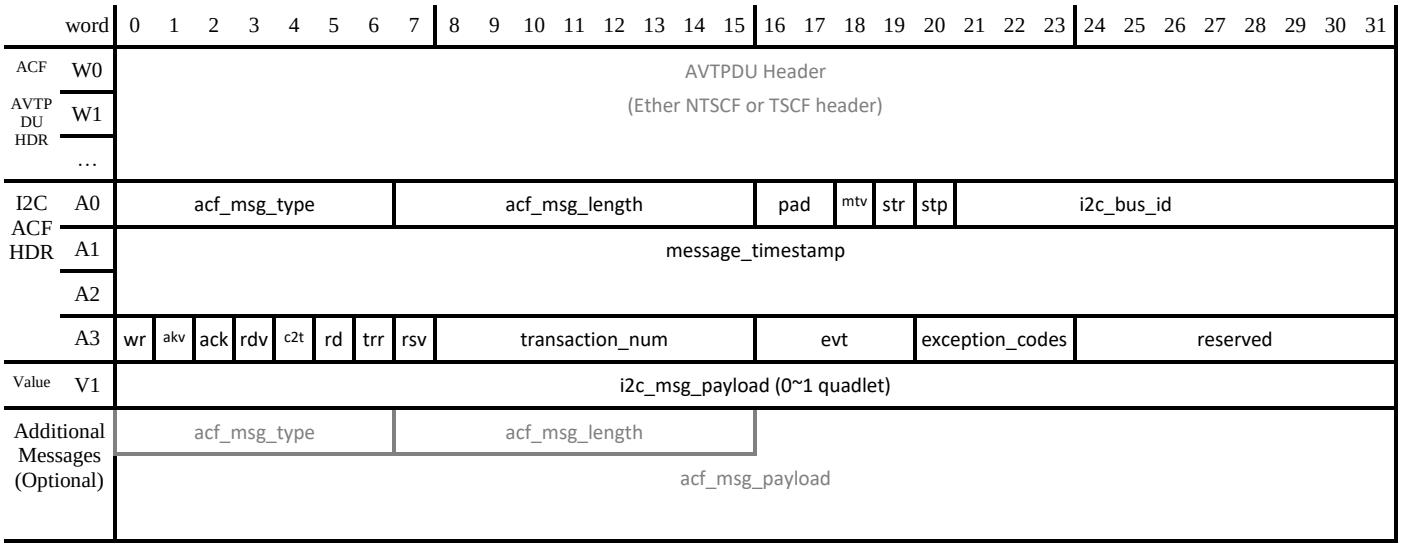

Figure 77 -- I2C ACF message

9.4.16.2. pad (padding length) field

The **pad** field indicates the length of padding at the end of the message payload in octets. The Talker adds padding to the original data so that this message ends on a quadlet boundary. The Listener removes the padding when it parses the ACF message to obtain the original base message. A value of zero (0) in the **pad** field indicates that there is no padding in the ACF message payload. For the I2C ACF message, the **i2c_msg_payload** may be null or one byte; therefore, the only valid **pad** values are 0 or 3. See 9.4.1.6 for examples showing the proper placement of a single byte in the **i2c_msg_payload** quadlet.

9.4.16.3. mtv (message_timestamp valid) field

The **mtv** (message timestamp valid) field contains a bit indicating whether the 8-octet **message_timestamp** field contains valid data.

The bit is set to one (1) if the **message_timestamp** field contains a valid timestamp.

The bit is set to zero (0) if the **message_timestamp** field is not valid. An AVTP Listener ignores the **message_timestamp** in any message where **mtv** is set to zero (0) as its contents are undefined (see 9.4.1.5).

9.4.16.4. str (start) field

The **str** (start) field contains a bit that indicates the I2C bus START or RE-START signal, when c2t field is one (1). The str field is ignored when the c2t field value is a zero (0).

9.4.16.5. stp (stop) field

The **stp** (stop) field contains a bit that indicates I2C bus STOP signal.

This bit is set to one (1) if the I2C bus STOP signal is asserted.

This bit is set to zero (0) if the I2C bus STOP signal is deasserted.

9.4.16.6. i2c_bus_id field

The 11-bit **i2c_bus_id** field provides an identifier for the byte bus on which this message originated or is targeted. The value of **i2c_bus_id** is application specific. It has several potential uses, including the identification of the originating and terminating I2C bus transport when using Ethernet as backbone to carry messages from a multiplicity of I2C buses.

If not specified for a particular application, the value of **i2c_bus_id** shall be set to zero (0) by the Talker.

9.4.16.7. message_timestamp field

See 9.4.1.5.

9.4.16.8. wr (write) field

The **wr** (write) field contains a bit indicating a write operation and that **i2c_msg_payload** contains valid write data. Note: wr field does not reflect I2C bus transaction direction bit (R/W).

This bit is set to one (1) if the **wr** is asserted.

This bit is set to zero (0) if the **wr** deasserted.

9.4.16.9. akv (ack valid) field

The **akv** (ack valid) field contains a bit indicating the **ack** field (9.4.16.10) is valid.

This bit is set to one (1) if the **akv** is asserted.

This bit is set to zero (0) if the **akv** is deasserted.

9.4.16.10. ack (acknowledge) field

The **ack** (acknowledge) field contains a bit indicating the bus acknowledge bit from the I2C bus transaction.

This bit is set to one (1) if the **ack** is asserted.

This bit is set to zero (0) if the **ack** is deasserted.

9.4.16.11. rdv (read data valid) field

The **rdv** (read data valid) field contains a bit indicating the **rd** bit (9.4.16.13) is valid as well as the contents of i2c_msg_payload.

This bit is set to one (1) if the **rdv** is asserted.

This bit is set to zero (0) if the **rdv** is deasserted.

9.4.16.12. c2t (Controller to Target) field

The **c2t** (Controller to Target) field contains a bit indicating the direction of this I2C message.

This bit is set to one (1) if the I2C message is from Controller to Target.

This bit is set to zero (0) if the I2C message is from Target to Controller.

9.4.16.13. rd (read) field

The **rd** (read) field contains a bit indicating a read operation on the I2C bus. See **rdv** (9.4.16.11) whether i2c_msg_payload contains valid read data. Note: **rd** field does not reflect I2C bus transaction direction bit (R/W).

This bit is set to one (1) if the **rd** is asserted.

This bit is set to zero (0) if the **rd** is deasserted.

9.4.16.14. trr (target response request) field

The **trr** (target response request) field contains a bit indicating Controller Agent requests the Target Agent to send a response when the Target's I2C bus cycle ended. See 9.4.16.19 for the Request and Response message types.

This bit is set to one (1) if the Controller Agent requests the Target Agent to send a message in response to a request with the **stp** bit set (CR4-WE or CR7-RE). Upon receipt, Target Agent shall respond with the message TR5-End when bus cycle terminates with a STOP.

This bit is set to zero (0) if the Controller Agent does not request Target Agent to send any response to a request with the **stp** bit (CR4-WE or CR7-RE) set unless an error occurs. Upon receipt, Target Agent shall not send any response.

This bit set to one (1) if the Controller requests the Target Agent to send a message in responds to a request with the <bold>stp</bold> bit set. Upon receipt, Target shall generate a response message when bus cycle terminates with a STOP.

This bit is set to zero (0) if the Controller does not request the Target to send any response to a request with the <bold>stp</bold> bit set unless an error occurs.

9.4.16.15. transaction_num field

The **transaction_num** field contains 8 bits indicating the sequence of I2C messages in a stream. It is used to associate Target Agent's response messages to request messages generated by Controller Agent. This field may start at any value for the first I2C transaction message. The **transaction_num** field value shall be maintained for each **i2c_bus_id field** value. The Target Agent's response messages shall use the received **transaction_ num** value in the corresponding Controller's request message associated with the given i2c_bus_id for all responses. The **transaction_ num** field shall be incremented by one (1), with wrapping, on the start of each new I2C request message. This field is to wrap from 11111111_2 to 00000000_2 (FF_{16} to 00_{16}).

The value of this field is determined by the Controller Agent. Target can use the **transaction_ num** to detect I2C messages lost in transit. This field is also used to associate an operation request with the correct operation response(s) that are returned to the Controller Agent's message(s) (See 9.4.16.19).

A Target Agent shall be able to tolerate a starting **transaction_ num** of any value.

The transaction_num field may be used to support optional message retransmission upon loss of message detection described in 9.4.16.20.

9.4.16.16. evt (event) field

The **evt** (event) field contains 4 bits that are reserved for upper-level protocols to indicate events in the messages. The specific use of these bits is outside the scope of this standard.

NOTE— One example usage of the **evt** field is the ability to change a message's bit rate speed for applications where fast and slow devices coexist on the same target I2C bus. This is needed when a slower device can only accept its data if the bus's bit rate speed is one that it supports and faster devices on the same bus need to get their data faster.

9.4.16.17. exception codes field

The **exception codes field** contains 4 bits that encodes an I2C bus transaction exceptions that may have occurred during the bus transaction. The Target Agent uses the exception code to indicate conditions that caused abnormal I2C bus cycle termination, or error conditions that should cause termination of bus cycle, or reporting exception condition while performing proper operations such as duplicate request handing, or others.

It may be possible, though unlikely, to experience multiple exceptions in the same response function. When multiple exceptions are detected, it is recommended that the lowest binary value encoding to be selected; however, the Target agent should select the most relevant exception code in a given bus cycle.

exception codes <bit 0, 1, 2, 3>	meaning
0000	No exceptions detected
0001 - 0011	Reserved for future use
0100 - 0111	User defined exceptions or errors.
1000	Bus timeout – transaction did not complete due to timeout waiting for an **ack or data.**
1001	Bus busy – transaction could not start due to excessive time waiting for bus idle.
1010	Controller conflict – i2c_bus_id does not match initial request. The behavior of a Target Agent that is capable of queuing requests from multiple Controller Agents is beyond the scope of this standard
1011	**transaction num** sequence error –sequence does not match expected value from the previous message.
1100	Start error – **str** (START) field is not set in a I2C Controller message or **str** (START) field is set when not expected and corresponding bus transaction terminated.
1101 - 1111	Reserved for future use

Table xx – I2C ACF exception codes field values

9.4.16.18. i2c_msg_payload field

The **i2c_msg_payload** field is a variable-length field that contains from 0 to 1 quadlet of message data and if 1 quadlet in size, it shall contain exactly one byte of valid data. It is permissible for **acf_msg_length** field to indicate 5 quadlets of message when no valid data is expected. In such case, the receiver shall deem the payload quadlet as null data.

The overall size of the **i2c_msg_payload** field shall always be an integer multiple of quadlets. See 9.4.1.6 for examples of various and different messages lengths, their **pad** values and quadlet sizes, and the data placement in the payload field.

9.4.16.19. Transparent I2C ACF Application

In figure 78, the device A, may initiate a read or write to the device D over the I2C Bus. The Controller A sends a request (read or write); the request is received and encoded in I2C ACF by the 1722 I2C Controller Agent ("Controller Agent") and sent over the network; the request is received by the 1722 I2C Target Agent ("Target Agent") and appropriate bus signals are asserted onto the I2C Bus connected to the device D. Devices A and D do not need to be aware of the presence of the intermediate agents nor the network.

Each I2C bus state and signals from the Controller is encoded in one of the I2C Requests by the Controller Agent, and each I2C Request is processed by the Target Agent and one of the I2C Response is sent back for each of the I2C Request made. A Target Agent shall respond to each I2C Request with an I2C Response that reflects the I2C bus transaction state. For the Requests CR4-WE (Write with End) and CR7-RE (Read with End), a Target Agent respond with TR5-End to indicate end of I2C bus activity, if the **trr** field is set to one (1). Unlike other I2C Requests or I2C Responses, TR5-End does not have corresponding I2C bus state change upon its receipt.

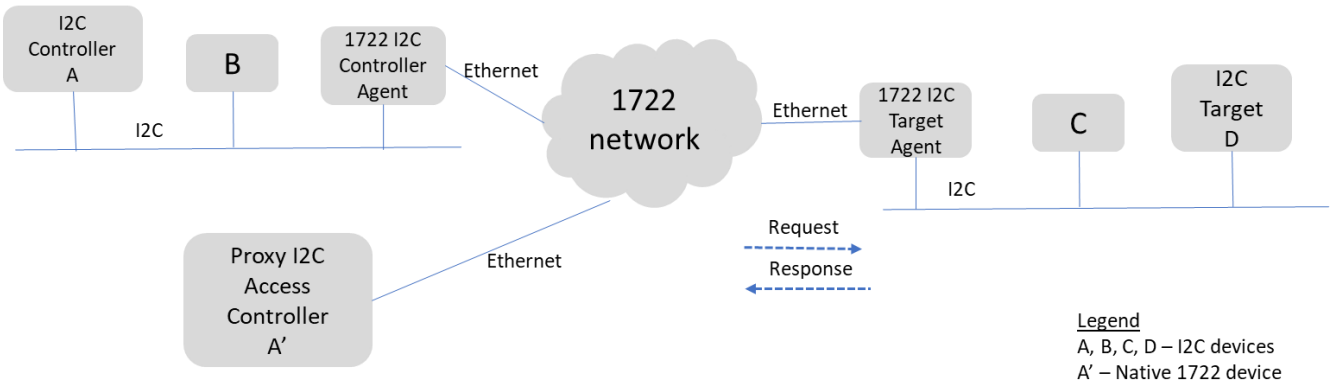

Figure 78 -- Network and 1722 I2C Agent connections to I2C Bus

The 1722 I2C Controller Agent (may issue the following I2C Request types, conveyed via I2C bus states and signals:

CR1-Start: Bus Transaction Start with a read or write
CR2-AC: Address Continuation (10 bit address access)
CR3-WC: Write Continue
CR4-WE: Write with End (STOP)
CR5-WR: Write with Restart
CR6-RC: Read Continue
CR7-RE: Read with End (STOP)
CR8-RR: Read with Restart

The 1722 I2C Target Agent may issue the following I2C Response types, conveyed via I2C bus states and signals:

TR1-NACK: Bus transaction NACK (not acknowledge)

TR2-ACK: Bus transaction ACK (acknowledge)
TR3-RD: Read Data
TR4-RAD: Target ACK and Read Data
TR5-End: Bus transaction End Confirm (CR4 or CR7 End response)

The Table xx shows encodings of various I2C ACF field values associated with I2C Request types or the I2C Response types. The **transaction_num**, **exception_codes**, **trr**, and **evt** fields are excluded from the table to remain concise.

Blank space indicate don't care. ~~The variable RR (Read Request) used on **rd** field may be set to a one (1) if a read operation with Target's early response is indicated, or set to a zero (0) if a read operation with Target's response to a read request is indicated.~~ When the Read Request is indicated from the Controller Agent's request, the Target Agent's response will be returned with the first byte of read data. The Figure 79 illustrates representative bus transactions ~~both an early response and a read request~~.

Request or Response	I2C_msg_ payload	**wr**	**akv**	**ack**	**rdv**	**c2t**	**rd**	**str**	**stp**
Controller									
CR1-Start	<addr>	1	0		1	1	1~~RR~~	1	0
CR2-AC	<addr>	1	0		1	1	1~~RR~~	0	0
CR3-WC	<data>	1	0		0	1	0	0	0
CR4-WE		0	0		0	1	0	0	1
CR5-WR	<addr>	1	0		1	1	1~~RR~~	1	0
CR6-RC		0	1	1	0	1	1	0	0
CR7-RE		0	1	0	0	1	0	0	1
CR8-RR	<addr>	1	1	0	1	1	1~~RR~~	1	0
Target									
TR1-NACK			1	0	0	0			
TR2-ACK			1	1	0	0			
TR3-RD	<data>		0		1	0	1		
TR4-RAD	<data>		1	1	1	0	1		
TR5-END			0		0	0			

Table xx – I2C ACF Controller Request and Target Response field values

Here are example I2C bus tractions to help illustrate the application of I2C Request and Response pairs for
a write operation terminated by the Target,
a write operation terminated by the Controller,
a read operations with Target's early response,
a read operation with Target's read request, and
a write operation followed by a re-start into a read operation with Target's response to a read request.

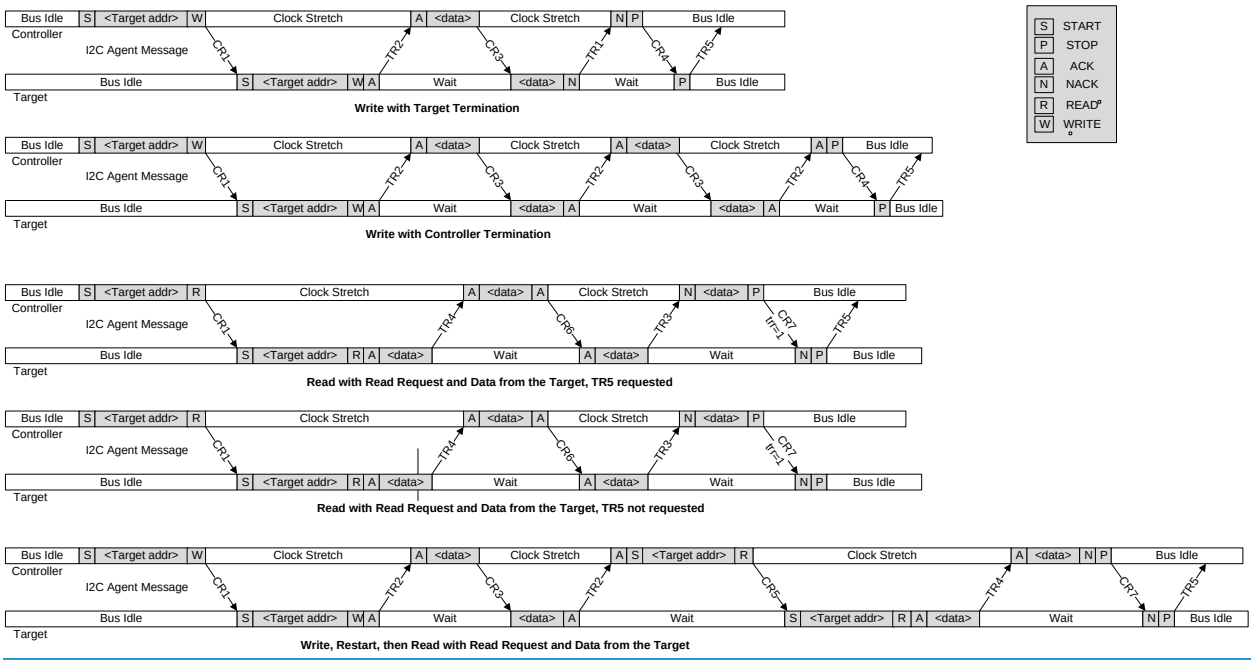

Figure 79 -- I2C Bus and Agent Request: Response Examples

9.4.16.20. Lost Message Handling

In figure 78 the Controller Agent request message may be lost due to congestion (if bandwidth reservation is not used) or due to random error (bit-error-rate of given physical layer), or any other reason. The Controller Agent, at its option, may implement a timer to help detect message loss and then re-transmit the same message to the Target. In this case, Target sees the request message for the first time, and responds normally. If, instead, the response from the Target was lost, the Controller Agent still detects message loss. If Controller Agent re-transmits the same message to the Target when the response message was lost, the Target Agent sees the same message from the Controller twice. The same message is detected when **transaction_num** did not increment from the message immediately preceding the current message and the same i2c_bus_id field value is used.

The Controller's message loss detection and re-transmission is optional and beyond the scope of this standard. The appropriate message-loss timer value is often specific to the system and should be set to a lower value than the I2C bus timeout value in the system.

The Target Agent shall respond to the detected duplicate message from the same Controller with the same response as the previous message in order to support the Controller's lost message handling function.

9.4.16.21. Proxy I2C Access Controller

In figure 78, the Controller and I2C Controller Agent function may be entirely embodied in the Ethernet node, as illustrated as the Proxy I2C Access Controller. In this case, there may not be a physical I2C bus on the Controller, while the Target may be a legacy I2C bus device. Neither the I2C Target Agent nor the Target can distinguish whether the Controller has a physical I2C bus. Therefore, it is recommended that the interoperability considerations for I2C ACF to be performed at the I2C Controller Agent and the I2C Target Agent.

[Editor's note: Check on all the shall statements and PICS pro forma to ensure interoperability test points are all on the Target and Target Agent side whenever and where ever possible.]

9.4.17. Abbreviated I2C message

[Editor's note: Don to consider this in overall CL9 context: It is perhaps worth indicating that the Abbreviated I2C ACF message is similar to "I2C ACF message" except that it doesn't have a timestamp. Clause 9.4.17 describes only the differences between "Abbreviated I2C ACF message" and "I2C ACF message"]

9.4.17.1. Overview

The Abbreviated Inter-Integrated Circuit (I2C) ACF message provides the capability to transport messages on I2C over a TSN network. An I2C device may be a Controller or a Target. For a given I2C bus cycle, there is only one Controller and typically there is only one Target that responds. The bus cycle signals are transported as messages encoded in ACF over a TSN network.

For a more modern system implementation of I2C transport where the legacy I2C bus cycle compatibility is not required, the Generic Byte Bus (GBB) ACF message (9.4.14) may be used to transfer one or more bytes in a single ACF message. The use of GBB would also provide better network bandwidth utilization and minimize processing overhead.

The I2C message inherits the following fields from the Common ACF message described in 9.4.1:

— **acf_msg_type**: 7 bits
— **acf_msg_length**: 9 bits

The I2C message has an **acf_msg_type** field of ACF_I2C_BRIEF (see Table 22).

The I2C message defines the following message-specific fields:

– **pad** (padding length): 2 bits
– **mtv** (message_timestamp valid): 1 bit
– **str** (start): 1 bit
– **stp** (stop): 1 bit
– **i2c_bus_id**: 11 bits
– **wr** (write): 1 bits
– **akv** (ack valid): 1 bit
– **ack** (acknowledge): 1 bit
– **rdv** (read data valid): 1 bit
– **c2t** (Controller to Target): 1 bit
– **rd** (read): 1 bit
– **trr (**target response 5 request): 1 bit
– **rsv** (reserved): 1 bit
– **transaction_num**: 8 bits
– **evt:** 4 bits
– **exception_codes:** 4 bits
– **reserved:** 8 bits
– **i2c_msg_payload**: 0 to 1 quadlets

These fields are shown in 9.4.16.1.

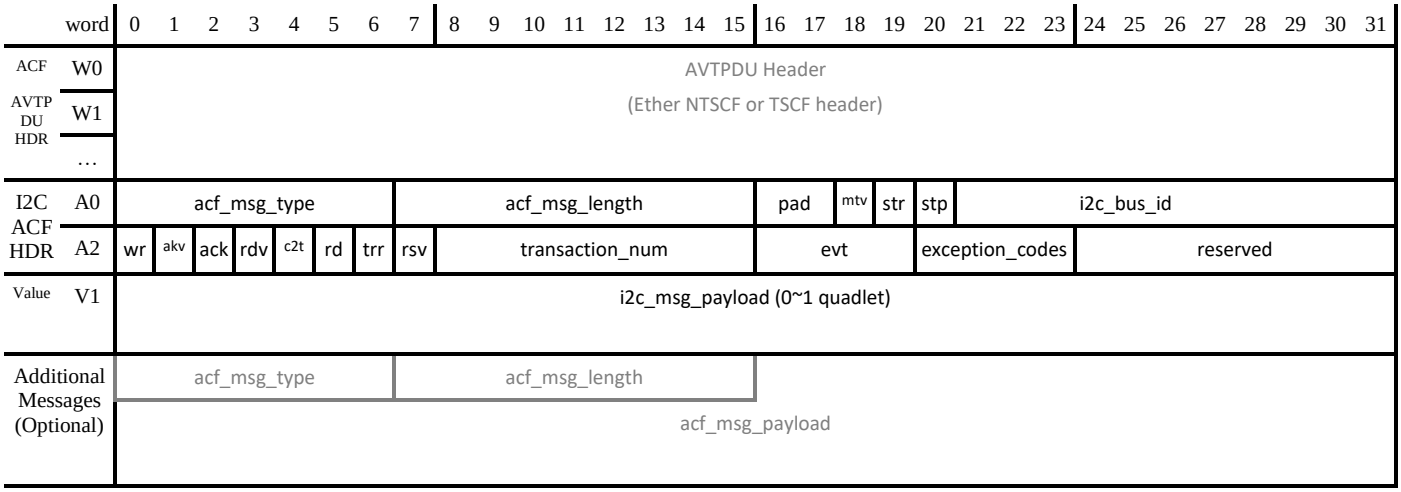

Figure 80 – Abbreviated I2C ACF message

9.4.17.2. mtv (message_timestamp_valid) field

The **mtv** field shall set to zero (0) for all transmitted BYTE_BUS_BRIEF messages and ignored on all received BYTE_BUS_BRIEF messages.