OPEN Alliance 10BASE-T1x MACPHY Serial Interface

TC6 –10BASE-T1x MACPHY Serial Interface

Author & Company	See Contributing Members on Page 2
Title	10BASE-T1x MACPHY Serial Interface
Version	1.0
Date	14 September 2020
Status	Final
Restriction Level	Public

Version Control of Document

Version	Author	Description	Date
0.1	Harald Zweck	Initial version, template only	2018-11-12
0.2	Harald Zweck	Update of initial version to a new template	2018-11-16
0.3	Martin Miller	Added Chapter 8	2019-02-11
0.4	Piergiorgio Beruto	Update to new template, add introduction	2019-06-05
0.5	Piergiorgio Beruto, Tim Baggett	Comment Resolution #1	2020-02-17
0.6	Tim Baggett	Comment Resolution #2	2020-03-30
0.7	Piergiorgio Beruto, Tim Baggett	Comment Resolution #3	2020-04-30
0.8	Tim Baggett	Comment Resolution #4	2020-05-25
0.9	Tim Baggett, Piergiorgio Beruto	Comment Resolution #5	2020-06-22
0.10	Tim Baggett	Comment Resolution #6	2020-08-09
0.11	Tim Baggett	Comment Resolution #7	2020-08-27
1.0	Tim Baggett	Editorial changes Changed Restriction Level to Public Finalized	2020-09-14

Restriction level history of Document

Version	Restriction Level	Description	Date
0.1-0.11	OPEN Technical Members Only	Internal Draft	2018-11-12
1.00	Public	Final	2020-09-14

Contributing Members

Tim Baggett (Microchip)
Tobias Belitz (Renesas)
Piergiorgio Beruto (Canova Tech)
David Brandt (Rockwell Automation)
Gergely Huszak (Kone)
Kevin Jennings (Analog Devices)
Wojciech Koczwara (Rockwell Automation)
Martin Miller (Microchip)
Viliam Vozar (On Semiconductor)
Harald Zweck (Infineon Technologies)

Contents

Foreword (Disclaimer)

OPEN Alliance: Members Only/OPEN Internal OPEN Specification

OPEN Alliance CONFIDENTIAL

Copyright Notice and Disclaimer

OPEN Alliance members whose contributions were incorporated in the OPEN Specification (the "Contributing Members") own the copyrights in the OPEN Specification, and permit the use of this OPEN Specification as follows:

> OPEN ALLIANCE MEMBERS: Members of OPEN Alliance have the right to use this OPEN Specification, subject to the Member's continued compliance with the OPEN Alliance governance documents, Intellectual Property Rights Policy, and the applicable OPEN Alliance Promoter or Adopter Agreement; and
>
> NON-MEMBERS OF OPEN ALLIANCE: Use of the OPEN Specification by anyone who is not a Member of OPEN Alliance is prohibited.

The receipt of an OPEN Specification shall not operate as an assignment or license under any patent, industrial design, trademark, or other rights as may subsist in or be contained in or reproduced in any OPEN Specification. The implementation of this OPEN Specification will require such a license.

THIS OPEN SPECIFICATION IS PROVIDED ON AN "AS IS" BASIS AND ALL WARRANTIES, EITHER EXPLICIT OR IMPLIED, ARE EXCLUDED UNLESS MANDATORY UNDER LAW. ACCORDINGLY, THE OPEN ALLIANCE AND THE CONTRIBUTING MEMBERS MAKE NO REPRESENTATIONS OR WARRANTIES WITH REGARD TO THE OPEN SPECIFICATION OR THE INFORMATION (INCLUDING ANY SOFTWARE) CONTAINED THEREIN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR PURPOSE, OR ABSENCE OF THIRD PARTY RIGHTS AND MAKE NO REPRESENTATIONS AS TO THE ACCURACY OR COMPLETENESS OF THE OPEN SPECIFICATION OR ANY INFORMATION CONTAINED THEREIN.

THE OPEN ALLIANCE AND CONTRIBUTING MEMBERS ARE NOT LIABLE FOR ANY LOSSES, COSTS, EXPENSES OR DAMAGES ARISING IN ANY WAY OUT OF USE OR RELIANCE UPON THE OPEN SPECIFICATION OR ANY INFORMATION THEREIN. NOTHING IN THIS DOCUMENT OPERATES TO LIMIT OR EXCLUDE ANY LIABILITY FOR FRAUD OR ANY OTHER LIABILITY WHICH IS NOT PERMITTED TO BE EXCLUDED OR LIMITED BY OPERATION OF LAW.

Without prejudice to the foregoing, the OPEN Specification was developed for automotive applications only. The OPEN Specification has neither been developed, nor tested for non-automotive applications.

OPEN Alliance reserves the right to withdraw, modify, or replace any OPEN Specification at any time, without notice.

Introduction

The IEEE 802.3cg project defines two 10 Mbit/s PHYs operating over a single pair of conductors. The 10BASE-T1L (Clause 146) is a long reach PHY supporting full duplex point-to-point operation over 1 km ofsingle balanced pair of conductors. The 10BASE-T1S (Clause 147) is a short reach PHY supporting full / half duplex point-to-point operation over 15 m of single balanced pair of conductors, or half duplex multidrop bus operation over 25 m of single balanced pair of conductors.

Furthermore, the IEEE 802.3cg project defines the new Physical Layer Collision Avoidance (PLCA) Reconciliation Sublayer (Clause 148) meant to provide improved determinism to the CSMA/CD media access method. PLCA works in conjunction with the 10BASE-T1S PHY operating in multidrop mode.

The aforementioned PHYs are intended to cover the low-speed / low-cost applications in industrial and automotive environment. The large number of pins (16) required by the MII interface, which is specified by the IEEE 802.3 in Clause 22, is one of the major cost factors that need to be addressed to fulfil this objective.

The MACPHY solution integrates an IEEE Clause 4 MAC and a 10BASE-T1x PHY exposing a low pin count Serial Peripheral Interface (SPI) to the host microcontroller. This also enables the addition of Ethernet functionality to existing low-end microcontrollers which do not integrate a MAC controller.

Abbreviation/Symbols

*****	*AND*
!	*NOT*
+	*OR*
CSMA/CD	*Carrier-Sense Multiple Access with Collision Detection*
FCS	*Frame Check Sequence*
gPTP	*Generalized Precision Time Protocol*
MAC	*Medium Access Control*
PCS	*Physical Coding Sublayer*
PHY	*Physical Layer device*
PLCA	*Physical Layer Collision Avoidance*
PMA	*Physical Medium Attachment*
PMD	*Physical Medium Dependent*
PTPv2	*Precision Time Protocol, Version 2*
RSVD	*Reserved*
SPI	*Serial Peripheral Interface*

1 Scope

The document describes the serial interface between a 10BASE-T1S or 10BASE-T1L (collectively referred to in this document as 10BASE-T1x) capable MACPHY and a station control unit like e.g. a microcontroller.

2 Normative References

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[1] IEEE Computer Society, "IEEE Std 802.3cg™-2019, IEEE Standard for Ethernet, Amendment 5: Physical Layer Specifications and Management Parameters for 10 Mb/s Operation and Associated Power Delivery over a Single Balanced Pair of Conductors," IEEE Standards Association, New York, 2019.

[2] IEEE Computer Society, "IEEE Standard for Ethernet, IEEE Std 802.3," IEEE Standards Association, New York.

[3] IEEE Computer Society, "IEEE 1588-2008, Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Standards Association, New York, 2008.

[4] IEEE Computer Society, "IEEE Std 802.1AS-2011, IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks," IEEE Standards Association, New York, 2011.

[5] TC14 - Interoperability & Compliance Tests for 10BASE-T1S PHYs, “10BASE-T1S PLCA Management Registers – Revision 1.0,” OPEN Alliance, 2019.

3 Terms and Definitions

For the purposes of this document, the terms and definitions given in in the normative references in Section 2 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4 Overview

The MACPHY is specified to carry both data (Ethernet frames) and control (register access) transactions over a single full-duplex serial peripheral interface.

5 Serial Interface Signals

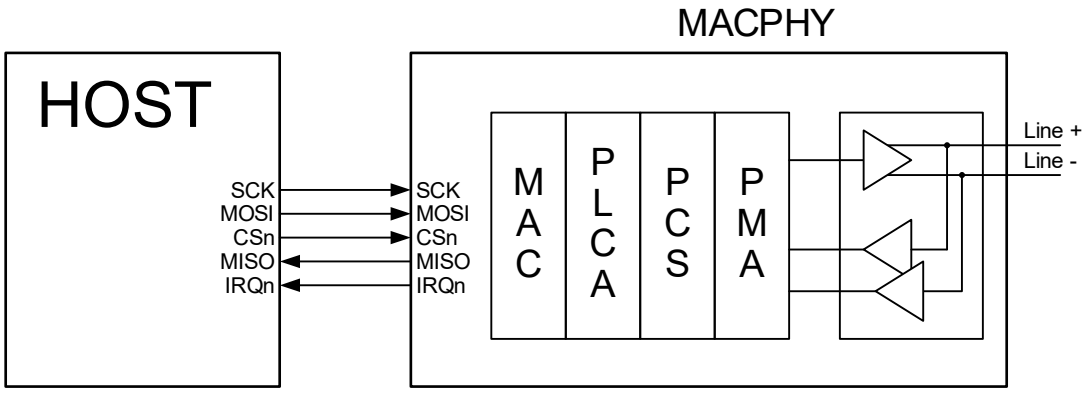

Figure 1: Pins of the OPEN serial 10BASE-T1x Interface

Table 1: OPEN serial 10BASE-T1x Interface Pin Definition

Symbol	Signal Description	Signal Source
CSn	Chip Select (Active Low)	HOST (master)
SCK	Serial Bit Clock	HOST (master)
MOSI	Master Out, Slave In Serial Data	HOST (master)
MISO	Master In, Slave Out Serial Data	MACPHY (slave)
IRQn	MACPHY Interrupt Request (Active Low)	MACPHY (slave)

6 Electrical Parameters of the Interface

6.1 DC Parameters

The DC logic levels supported by the interface is not specified and are implementation specific.

6.2 AC Parameters

Implementations of the interface shall support a maximum SPI clock (SCK) rate of at least 15 MHz. Operation can be performed at slower speeds, and vendors may support faster speeds.

7 SPI Protocols

7.1 SPI Format

The MACPHY shall support the SPI clock and phase format as illustrated in Figure 2. The clock is low when idle. The first data bit is output on MISO and MOSI when CSn is asserted low. Data is captured on the rising edge of SCK and changes on the falling edge. When CSn is deasserted the MACPHY shall set the MISO output to a high-impedance state while ignoring SCK and any input on MOSI.

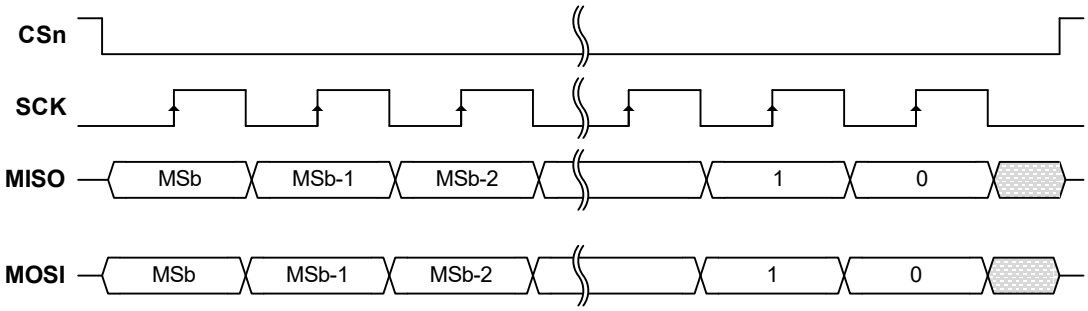

Figure 2: SPI Clock/Phase Format

Data representation is big-endian, with the most significant bit of each octet transferred first.

The SPI host (SPI master) providing the SPI clock is always the master of the transactions on the SPI bus. When the MACPHY (SPI slave) needs servicing, it may inform the SPI host by asserting the IRQn interrupt signal.

7.2 Protocol Overview

Two types of transactions are defined in the protocol: data transactions for Ethernet frame transfers and control transactions for register read/write transfers. A chunk is the basic element of data transactions and is composed of 4 bytes of overhead plus the configured payload size for each chunk. Ethernet frames are transferred over one or more data chunks. Control transactions consist or one or more register read/write control commands.

SPI transactions are initiated by the SPI host with the assertion of CSn low to the MACPHY and ends with the deassertion of CSn high. In between each SPI transaction, the SPI host may need time for additional processing and to setup the next SPI data or control transaction. The SPI transaction duration is defined by the protocol and transfer type.

The CSn shall be deasserted for a minimum period of time prior to changing between control and data transactions as illustrated in Figure 4. See resync_timer in Section 8.4. Deassertion of CSn is not required between transfer of multiple data chunks in a data transaction as shown in Figure 3 and Figure 4, or between multiple control commands in a control transaction as shown in Figure 4. Any deassertion of CSn before the expected end of a data chunk or control command aborts any ongoing transaction and reset the slave SPI bit deserializer/decoder as specified in 8.6.

An overview with multiple SPI data transactions is shown in Figure 3.

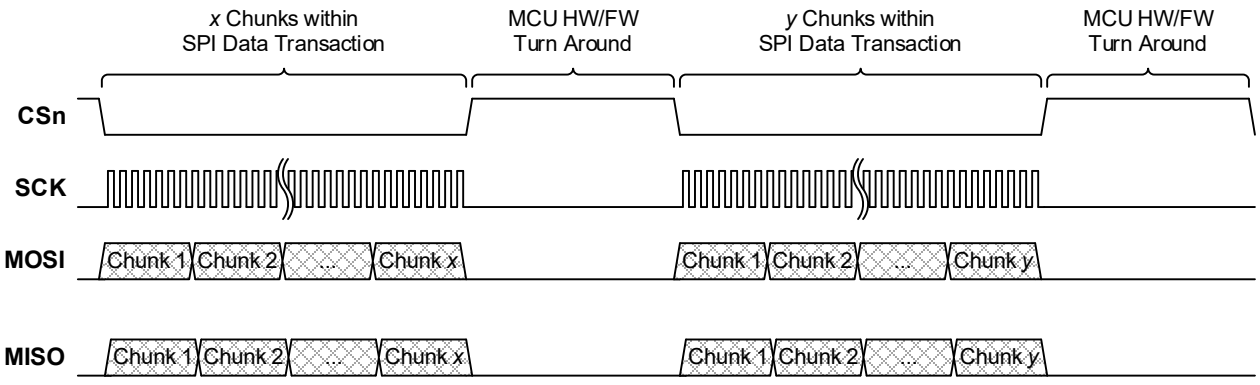

Figure 3: SPI Transaction Overview - Ethernet Frame Data Transfer

SPI data transactions consist of an equal number of transmit (TX) and receive (RX) chunks. Chunks in both transmit and receive directions may or may not contain valid frame data independent from each other, allowing for the simultaneous transmission and reception of different length frames. Each transmit data chunk begins with a 32-bit data header followed by a data chunk payload on MOSI. The data header indicates whether transmit frame data is present and provides the information to determine which bytes of the payload contain valid frame data. In parallel, receive data chunks are received on MISO. Each receive data chunk consists of a data chunk payload ending with a 32-bit data footer. The data footer indicates if there is receive frame data present within the payload or not and provides the information to determine which bytes of the payload contain valid frame data.

The details of data transactions are defined in Section 7.3

An overview illustrating a data transaction followed by a control transaction is shown below.

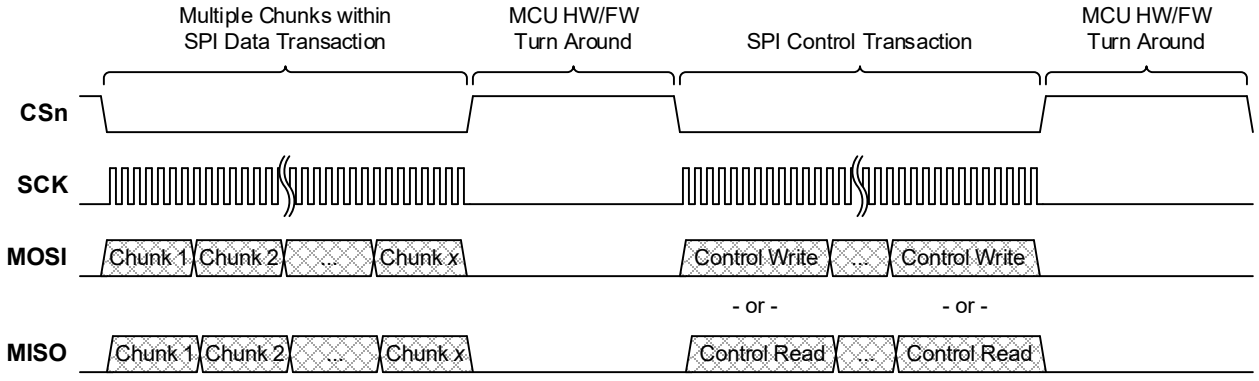

Figure 4: SPI Transaction Overview - Ethernet Frame Data Transfer Followed by Control Transfer

The SPI control transaction consist of one or more read or write control commands as illustrated in Figure 4. Each control command begins with a control header which specifies the register access type (read or write),starting address, and number of registers to read or write.

The details of control commands are defined in Section 7.4.

The SPI host must always deassert CSn to switch between data transactions and control transactions as illustrated in Figure 4 to avoid undefined behavior.

7.3 Data Transaction Protocol for Ethernet Frames

The format of the Ethernet frames transferred between the MAC client and the MAC sublayer are specified in clauses 2, 3, and 4 of IEEE 802.3 [2]. In the context of this specification, the SPI host takes the role of the MAC client and the MACPHY internally implements the MAC sublayer.

Ethernet frames are typically transferred from the SPI host to the MACPHY without any padding or frame check sequence (FCS). The MAC will automatically pad the Ethernet frame to the minimum frame size of 64 bytes and append a computed FCS. However, the Ethernet specification allows for the SPI host to optionally perform the frame padding and FCS computation prior to transfer to the MACPHY.

Similarly, the MACPHY will typically strip the FCS from received Ethernet frames prior to transfer to the SPI host. However, the Ethernet specification allows the option for the Ethernet frame to be transferred to the MAC client with the FCS.

The IEEE Ethernet standard [2] defines the behavior of the MAC and therefore is beyond the scope of this specification. As a result, support for allowing the SPI host to perform frame padding and FCS computation, or passing the FCS to the SPI host is optional. When supported, the method for configuring the MACPHY to enable these modes of frame transfer is implementation specific.

7.3.1 Ethernet Frame Protection

If the MAC supports the appending of transmit frame padding and FCS by the SPI host, and passing of FCS with received frames to the SPI host, then the MACPHY shall support the ability to validate transmit frames received from the SPI host to detect the occurrence of bit errors in the transfer of frames over SPI. When the MAC supports these features, the Transmit Frame Check Sequence Validation Capability bit in the STDCAP register shall be '1'. See Section 9.2.3.2.

When supported, validation of transmit frames over the SPI first requires the configuration of the MAC to expect the SPI host to perform frame padding and appending the FCS. Details for configuring the MAC are implementation specific. Validation of transmit frames received from SPI is enabled by setting the Transmit Frame Check Sequence Validation Enable bit in the CONFIG0 register. See Section 9.2.5.3. As frames are received from the SPI, the MACPHY will validate the FCS located in the last 4 bytes of the frame. If the frame is determined invalid, the Transmit Frame Check Sequence Error indication in the STATUS0 register shall be set (see Section 9.2.8.2). If practical, it is recommended that the MACPHY also drop the invalid frame prior to passing it to the MAC to avoid transmission of broken packets onto the network. The SPI host may monitor the Transmit Frame Check Sequence Error indication to detect and count invalid frames received by the MACPHY on MOSI. If unmasked, the Transmit Frame Check Sequence Error indication may set the Extended Status (EXST) interrupt bit (see Section 7.3.7) and assert the IRQn pin if there is no ongoing data chunk transfer.

Detection of errors in frames sent to the SPI host requires configuring the MAC to pass the FCS with the received frame to the SPI host. That is, the MAC will not strip the FCS from frames received from the network. Details for configuring the MAC are implementation specific. The SPI host may then validate the FCS to determine the validity of the received frame. Normally the MAC is configured to filter all invalid frames received from the network and not pass them to the host, therefore, the invalid frames that are detected by the SPI host will be known to have suffered a bit error on MISO. If the MAC is configured to pass all frames, including invalid frames, then the host can identify frames that incurred a bit error on MISO by comparing the number of invalid frames received at the host with invalid frame counters typically present in the MAC.

When transmit or receive frame bit errors are detected on the SPI, the retry of frames is performed by higher protocol layers that are beyond the scope of this specification.

7.3.2 Transmit Data Chunk

A transmit data chunk consists of a 4-byte data header followed by the transmit data chunk payload.

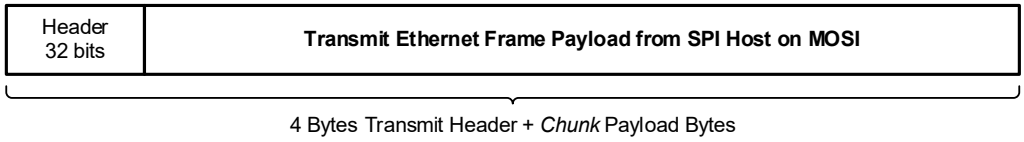

Figure 5: Transmit Data Chunk

7.3.3 Receive Data Chunk

A receive data chunk consists of the receive data chunk payload followed by a 4-byte data footer at the end.

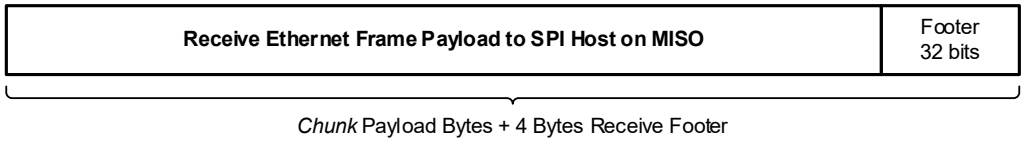

Figure 6: Receive Data Chunk

7.3.4 Chunk Payload Size

The MACPHY shall support a default data chunk payload size of 64 bytes. Data chunk payload sizes of 32, 16, or 8 bytes may also be supported. The data chunk payload is always a multiple of 4 bytes. If additional data chunk payload sizes are supported, this must be configured before enabling normal operation. Changing the data chunk payload size during normal operation may result in unexpected behaviour and loss of data. Therefore, once configured, the data chunk payload size should not be changed without the MACPHY being reset.

The total size of the data chunk is equal to the data chunk payload size plus the four bytes of the 32-bit data header/footer.

7.3.5 Data Chunk Transactions

SPI data transactions consist of 1 to *N* chunks of data on MOSI (transmit data) and MISO (receive data). The 4-byte data header occurs at the beginning of each transmit data chunk on MOSI and the 4-byte data footer occurs at the end of each receive data chunk on MISO. The data header and footer contain the information needed to determine the validity and location of the transmit and receive frame data within the data chunk payload. Ethernet frames shall be aligned to a 32-bit boundary within the data chunk payload, as mandated by Section 8.6.

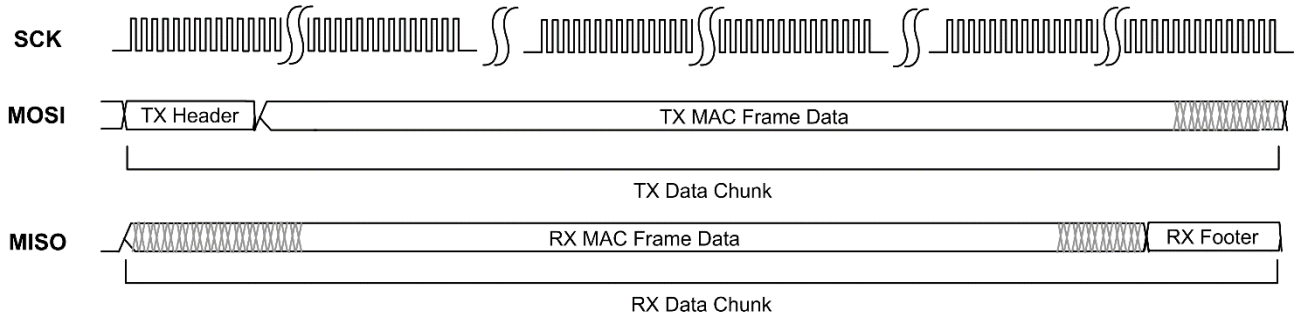

Figure 7: SPI Chunk Data Transfer

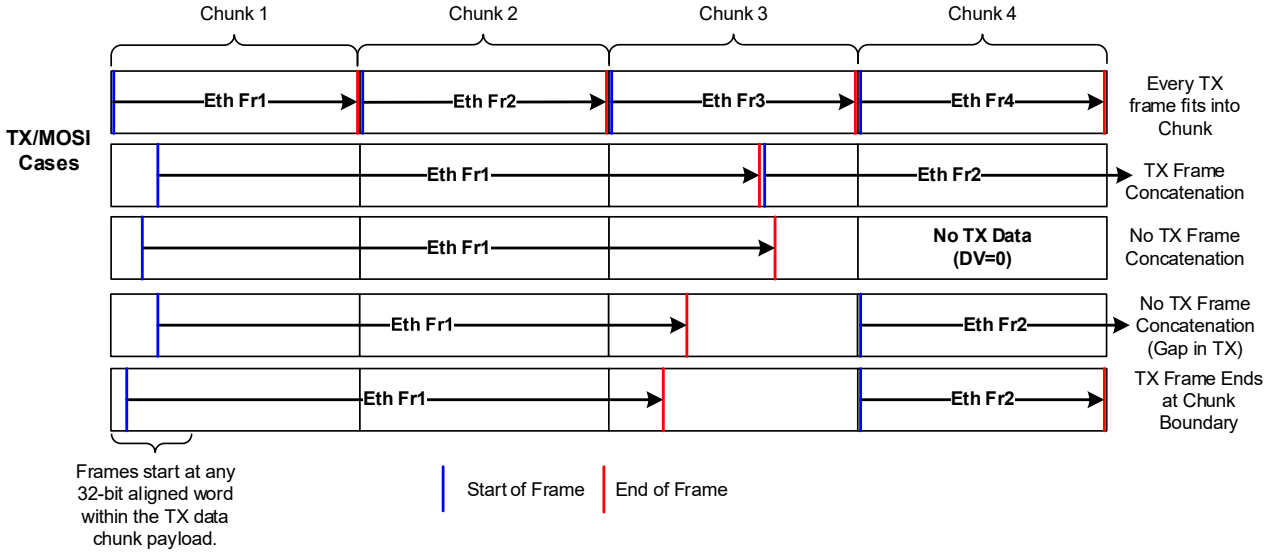

Figure 8: Transmit Data Chunk Cases

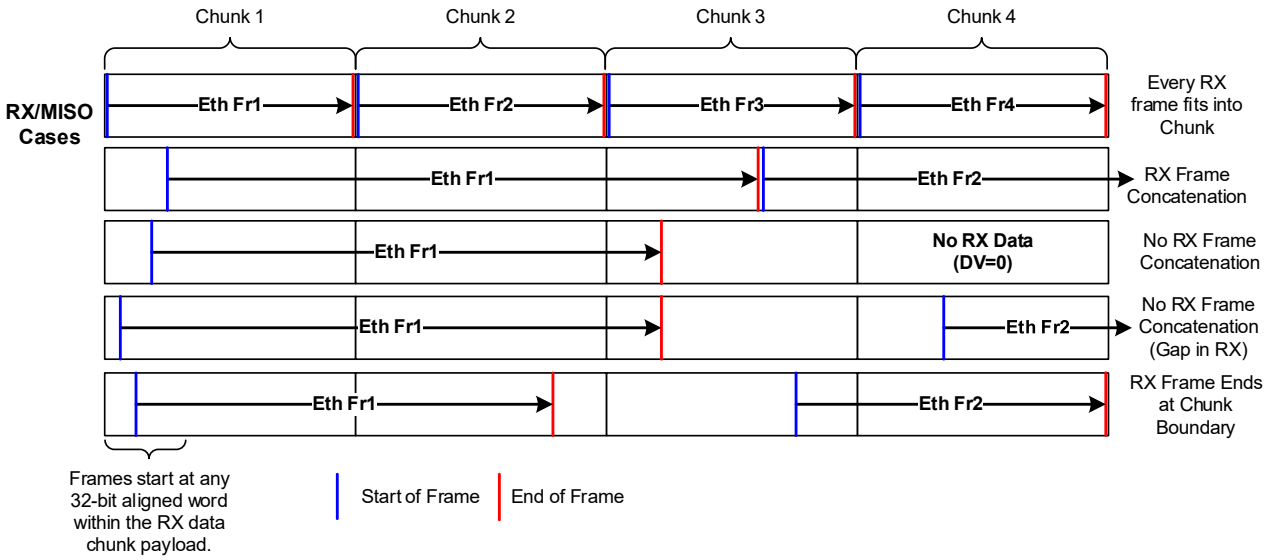

Figure 9: Receive Data Chunk Cases

7.3.6 Transmit Data Header

The transmit data header is shown in Figure 10 below.

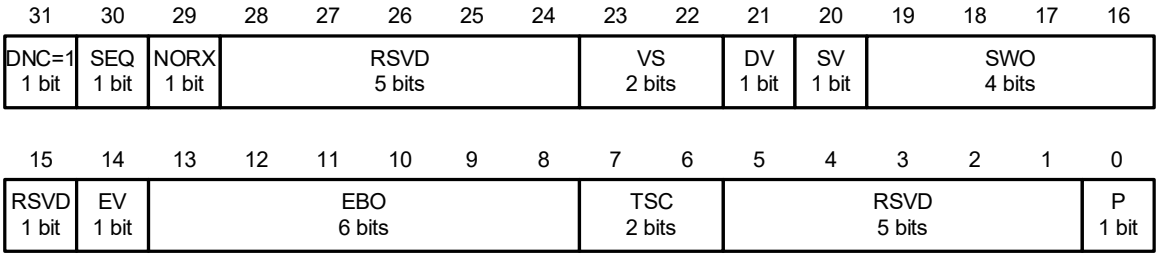

Figure 10: Transmit Data Header

Bit	Label	Name
31	DNC	Data-Not-Control
30	SEQ	Data Chunk Sequence
29	NORX	No Receive
28..24	RSVD	Reserved
23..22	VS	Vendor Specific
21	DV	Data Valid
20	SV	Start Valid
19..16	SWO	Start Word Offset
15	RSVD	Reserved
14	EV	End Valid
13..8	EBO	End Byte Offset
7..6	TSC	Transmit Frame Timestamp Capture
5..1	RSVD	Reserved
0	P	Header Parity Bit

Table 2: Transmit Data Header

DNC — Data-Not-Control flag. This flag specifies the type of SPI transaction. For TX data chunks, this bit shall be '1'.

- 0 Control command
- 1 Data chunk

SEQ — Data Chunk Sequence. This bit is used to indicate an even/odd transmit data chunk sequence to the MACPHY.

NORX — No Receive flag. The SPI host may set this bit to indicate to the MACPHY that it will not process receive frame data that may be in the current receive data chunk. For normal operation, the SPI host shall set NORX = 0 indicating that it will accept and process any receive frame data within the current chunk.

DV — Data Valid flag. The SPI host uses this bit to indicate whether the current chunk contains valid transmit frame data (DV = 1) or not (DV = 0). When '0', the MACPHY ignores the chunk payload. Note that the receive path is unaffected by the setting of the DV bit in the data header.

VS — Vendor Specific. These bits are implementation specific. If the MACPHY does not implement these bits, the host shall set them to '0'.

SV	Start Valid flag. The SPI host shall set this bit when the beginning of an Ethernet frame is present in the current transmit data chunk payload. Otherwise, this bit shall be zero. This bit is not to be confused with the Start-of-Frame Delimiter (SFD) byte described in IEEE 802.3 [2].
SWO	Start Word Offset. When SV = 1, this field shall contain the 32-bit word offset into the transmit data chunk payload that points to the start of a new Ethernet frame to be transmitted. The host shall write this field as zero when SV = 0.
EV	End Valid flag. The SPI host shall set this bit when the end of an Ethernet frame is present in the current transmit data chunk payload. Otherwise, this bit shall be zero.
EBO	End Byte Offset. When EV = 1, this field shall contain the byte offset into the transmit data chunk payload that points to the last byte of the Ethernet frame to transmit. This field shall be zero when EV = 0.
TSC	Timestamp Capture. Request a timestamp capture when the frame is transmitted onto the network. 00 Do not capture a timestamp 01 Capture timestamp into timestamp capture register A 10 Capture timestamp into timestamp capture register B 11 Capture timestamp into timestamp capture register C
P	Parity. Parity bit calculated over the transmit data header. Method used is odd parity. See Section 8.5.2.
RSVD	Reserved: All reserved bits shall be ‘0’.

The transmit data header and control command header share the location and meaning of the Data-Not-Control (DNC) flag location in the header definition. This allows the MACPHY to determine whether the transfer is for data or control.

When supported and enabled by the SEQE bit in the CONFIG0 register (see Table 12), the Sequence (SEQ) bit acts as an even/odd bit (i.e., the least-significant bit of a transmit data chunk counter) for identifying the sequence of transmit data chunks from the SPI host. The SPI host shall toggle the SEQ bit for each new transmit data chunk transferred. When the MACPHY receives a transmit data chunk in which the SEQ bit has not changed from the previous chunk, it assumes the SPI host is resending the previous chunk and the current chunk is not accepted. This feature is disabled by default and only supported in store-and-forward operation. When unsupported, disabled, or in cut-through operation, the MACPHY shall ignore the SEQ bit in the data header, and the SPI host should write SEQ = 0.

The No Receive (NORX) flag is used by the SPI host to perform flow control to the MACPHY. When NORX = 1, the SPI host will ignore the current receive data chunk payload. When the MACPHY receives a data header with NORX = 1, it will set DV = 0 within the footer of the current receive data chunk, regardless of any receive frame data being available. The MACPHY will retain any available receive frame data and attempt to resend it on MISO during the next data chunk. The SPI host will indicate to the MACPHY that it will receive and process the current receive data chunk payload by sending NORX = 0.

The Data Valid (DV) flag indicates to the MACPHY whether the current transmit data chunk payload contains valid transmit Ethernet frame data. When DV = 1, the SPI host shall set the SV and EV flags accordingly to locate the frame data boundaries within the chunk payload. If either or both SV and EV are set to ‘1’, either or

both of SWO and EBO are also set to locate the boundary of the valid frame data. It is possible that during long frame transfers DV = 1 and both SV = 0 and EV = 0. It is also possible that a single minimum size 64-byte Ethernet frame (or a smaller, unpadded frame) fits entirely within the transmit data chunk payload of a single chunk and both SV = 1 and EV = 1 (i.e., both the start and end of the frame are present within the data chunk payload). Regardless of size, the transfer of multiple frames in a single data chunk payload is impossible. The MACPHY ignores the transmit data chunk payload when DV = 0.

Start Valid (SV) flag indicates whether the transmit data chunk payload contains a valid beginning of an Ethernet frame. When SV = 1, the SWO shall be set accordingly to locate the beginning of the frame within the transmit data chunk payload. When SV = 1 the Start Word Offset (SWO) field is set to the offset, expressed in 32-bit words, of the first data byte within the payload of the chunk. That is, if the first byte of the payload contains the first byte of the Ethernet frame, SWO is set to '0'. Note that this implies that the offset of the first data byte of the frame is always aligned to a 32-bit boundary within the transmit data chunk payload such that the first byte of the frame is always the most significant byte of the 32-bit word. The allowed range of SWO is 0 to 15. When SV = 0, the SPI host shall write this field as all zero.

The End Valid (EV) flag indicates if the transmit data chunk payload contains the end of an Ethernet frame. If EV = 1, then EBO shall be set accordingly to point to the position of the last byte of the frame within the transmit data chunk payload.

When EV = 1, the SPI host sets the End Byte Offset (EBO) to the offset of the last byte of the Ethernet frame within the chunk payload. When EV=0, the SPI host shall write EBO as zero. The first byte of the transmit data chunk payload is located at an offset of zero.

The transmit Timestamp Capture (TSC) field is used by the SPI host to request the capture of a timestamp when the frame is transmitted onto the network. This field is only valid when SV = 1 and shall be ignored when SV = 0 or the timestamp feature is not supported. The SPI host shall set TSC = 00 at all other times. Additional details for frame timestamp capturing is found in Section 7.8.

Parity (P) uses odd parity to provide for transmit data header protection as described in Section 8.5.2. When a header is received from the SPI host with a parity error, the MACPHY handles the error as described in Section 7.5.

7.3.7 Receive Data Footer

The receive data footer is shown in Figure 11 below.

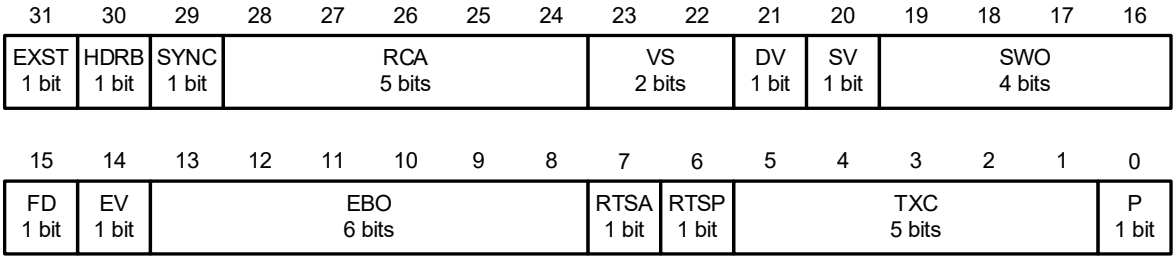

Figure 11: Receive Data Footer

Bit	Label	Name
31	EXST	Extended Status
30	HDRB	Received Header Bad
29	SYNC	Configuration Synchronized
28..24	RCA	Receive Chunks Available
23..22	VS	Vendor Specific
21	DV	Data Valid
20	SV	Start Valid
19..16	SWO	Start Word Offset
15	FD	Frame Drop
14	EV	End Valid
13..8	EBO	End Byte Offset
7	RTSA	Receive Frame Timestamp Added
6	RTSP	Receive Frame Timestamp Parity
5..1	TXC	Transmit Credits
0	P	Footer Parity Bit

Table 3: Receive Data Footer

EXST Extended Status. This bit is set when any bit in the STATUS0 or STATUS1 registers are set and not masked.

HDRB Received Header Bad. When set, indicates that the MACPHY received a control or data header with a parity error.

SYNC Configuration Synchronized flag. This bit reflects the state of the SYNC bit in the CONFIG0 configuration register (see Table 12). A zero indicates that the MACPHY configuration may not be as expected by the SPI host. Following configuration, the SPI host sets the corresponding bit in the configuration register which is reflected in this field.

RCA Receive Chunks Available. The RCA field indicates to the SPI host the minimum number of additional receive data chunks of frame data that are available for reading beyond the current receive data chunk. This field is zero when there is no receive frame data pending in the MACPHY's buffer for reading.

VS Vendor Specific. These bits are implementation specific. If not implemented, the MACPHY shall set these bits to '0'.

DV Data Valid flag. The MACPHY uses this bit to indicate whether the current receive data chunk contains valid receive frame data (DV = 1) or not (DV = 0). When '0', the SPI host shall ignore the chunk payload.

SV Start Valid flag. The MACPHY sets this bit when the current chunk payload contains the start of an Ethernet frame. Otherwise, this bit is zero. The SV bit is not to be confused with the Start-of-Frame Delimiter (SFD) byte described in IEEE 802.3 [2].

SWO Start Word Offset. When SV = 1, this field contains the 32-bit word offset into the receive data chunk payload containing the first byte of a new received Ethernet frame. When a receive timestamp has been added to the beginning of the received Ethernet frame (RTSA = 1) then SWO points to the most significant byte of the timestamp as described in detail in Section 7.8. This field will be zero when SV = 0.

FD	Frame Drop. When set, this bit indicates that the MAC has detected a condition for which the SPI host should drop the received Ethernet frame. This bit is only valid at the end of a received Ethernet frame (EV = 1) and shall be zero at all other times.
EV	End Valid flag. The MACPHY sets this bit when the end of a received Ethernet frame is present in this receive data chunk payload.
EBO	End Byte Offset: When EV = 1, this field contains the byte offset into the receive data chunk payload that locates the last byte of the received Ethernet frame. This field is zero when EV = 0.
RTSA	Receive Timestamp Added. This bit is set when a 32-bit or 64-bit timestamp has been added to the beginning of the received Ethernet frame. The MACPHY shall set this bit to zero when SV = 0.
RTSP	Receive Timestamp Parity. Parity bit calculated over the 32-bit/64-bit timestamp added to the beginning of the received Ethernet frame. Method used is odd parity as described in Section 8.5.2. The MACPHY shall set this bit to zero when RTSA = 0.
TXC	Transmit Credits: This field contains the minimum number of transmit data chunks of frame data that the SPI host can write in a single transaction without incurring a transmit buffer overflow error.
P	Parity. Parity bit calculated over the receive data footer. Method used is odd parity. See Section 8.5.2.
RSVD	Reserved: All reserved bits must be ‘0’.

If the MACPHY detects an invalid header from the SPI host, due to an incorrect parity check, then the MACPHY sets the Header Bad (HDRB) bit. The HDRB bit is zero otherwise. Additional details describing the handling of header parity errors are specified in Section 7.5.

The Configuration Synchronized (SYNC) bit which reflects the state of the SYNC bit in the CONFIG0 register. See Table 12. See Section 7.6 for details on synchronizing the MACPHY configuration with the SPI host.

The Data Valid (DV) flag indicates to the SPI host whether the current receive data chunk payload contains valid receive Ethernet frame data. When DV = 1, the SV and EV flags are set accordingly to locate the frame data boundaries within the chunk payload. If either or both SV and EV are set to ‘1’, either or both of SWO and EBO are also set to locate the boundary of the valid frame data. It is possible that during long frame transfers DV = 1 and both SV = 0 and EV = 0. It is also possible that a single Ethernet frame fits entirely within the receive data chunk payload and both SV = 1 and EV = 1 (e.g. 64-byte Ethernet Frame). Regardless of size, the transfer of multiple frames in a single data chunk payload is impossible. The SPI host ignores the receive data chunk payload when DV = 0.

The Frame Drop (FD) flag is used as an indicator to the SPI host that the received data frame ending in the current data chunk should be dropped (ignored). Some MACs implement the ability to transfer bad received frames to the station entity for debugging purposes. This bit allows the MACPHY to transfer such error frames with an indication that the frame is correct or not. See Section 7.9 for additional usage details in receive cut-through operation. This bit is only valid in a data chunk containing the end of a received Ethernet frame when DV = 1 and EV = 1. It is zero in all other conditions.

The Extended Status (EXST) bit is used by the device to communicate events and errors to the SPI host which need servicing. The EXST bit is set any time a status bit is set within the STATUS0 or STATUS1 registers and not masked. See Sections 9.2.8 through 9.2.12. The SPI host can use this bit to determine and schedule

appropriate control commands to read the status registers and maintain proper operation of the MACPHY. A MACPHY implementation may include additional sources for setting the EXST bit beyond the STATUS0 and STATUS1 registers in this specification.

Start Valid (SV) flag indicates whether the receive data chunk payload contains a valid beginning of an Ethernet frame. When SV = 1, the SWO is set accordingly to locate the beginning of the frame within the receive data chunk payload. When SV = 1 the Start Word Offset (SWO) field is set to the offset, expressed in 32-bit words, of the first data byte within the payload of the chunk. That is, if the first byte of the payload contains the first byte of the Ethernet frame, SWO is set to '0'. Note that this implies that the offset of the first data byte of the frame is always aligned to a 32-bit boundary within the receive data chunk payload such that the first byte of the frame is always the most significant byte of the 32-bit word. The allowed range of SWO is 0 to 15. When SV = 0, the MACPHY writes this field as all zero and it shall be ignored by the SPI host.

Received frames are allowed to begin anywhere within the receive data chunk payload on a 32-bit word boundary. However, if the Zero-Align Receive Frame Enable (ZARFE) bit in CONFIG0 is set, then the MACPHY aligns the start of received frames to the first data word of the receive data chunk payload (SWO = 0). See Section 9.2.5.5. Additionally, received frames may be limited to start only in the first data word of the first receive data chunk payload following the assertion of CSn when the CSn Align Receive Frame Enable (CSARFE) bit is set in CONFIG0 (see Section 9.2.5.4). Enabling of these options will reduce performance especially in high-bandwidth applications, but may reduce memory management complexity in lower end SPI host microcontrollers.

The End Valid (EV) flag indicates if the receive data chunk payload contains the end of a received Ethernet frame. If EV = 1, then EBO points to the position of the last byte of the frame within the receive data chunk payload.

When EV = 1, the MACPHY sets the End Byte Offset (EBO) to the offset of the last byte of the Ethernet frame within the chunk payload. When EV = 0, EBO is set to zero. The first byte of the receive data chunk payload is located at an offset of zero.

The Transmit Credit (TXC) field is used to indicate to the SPI host the minimum number of consecutive transmit data chunks of frame data (DV = 1) the SPI host may send to the MACPHY without incurring a transmit buffer overflow condition. The TXC field essentially denotes the minimum amount of free buffer space (in chunks) that the MACPHY has available for accepting frame data written by the SPI host following reception of the data footer. This is a form of transmit flow control and can be used to optimize use of the available SPI bus bandwidth while also avoiding data loss. The TXC field is also available by reading the Buffer Status Register (see Section 9.2.10.2).

The Receive Chunks Available (RCA) field provides the SPI host with the minimum number of additional receive data chunks of frame data (DV = 1) available for reading by the SPI host. When the SPI host receives a data footer with RCA > 0, it can immediately perform additional data chunk transactions to read the available receive frame data. The MACPHY indicates with RCA = 0 that there is no receive frame data present in the MACPHY's buffers at the time the footer was assembled. In this case, the SPI host may continue polling for receive data by transferring (possibly empty) transmit data chunks. If the previously conveyed footer prior to CSn being deasserted indicated RCA = 0, the SPI host may also wait for receive data to become available and the MACPHY to assert the IRQn pin (see Section 7.7). The RCA field is also available by reading the Buffer Status Register (see Section 9.2.10.3).

Parity (P) uses odd parity to provide for receive data footer protection as described in Section 8.5.2.

Upon assertion of CSn, the MACPHY defaults into a data state and begins immediately outputting receive chunk payload data until the reception of the DNC bit of the header. If the header is received with DNC = 1 indicating a data chunk, the MACPHY remains in the data state processing any received transmit chunk payload data and outputting receive chunk payload data followed by the data footer. However, if the current header is received with DNC = 0, indicating a control command, the MACPHY transitions into a control state within the first 32-bit word of the transaction. If the MACPHY has receive frame data ready to be transferred to the SPI host at the time it receives a control command, the receive frame transfer is deferred until the next data transaction.

7.3.8 Data Chunk Transfer Errors

The MACPHY shall detect and report at least the following errors in the transfer of frame data chunks. When a data chunk transfer error occurs, an error status is reported in the STATUS0 register and the data footer EXST bit is set. Ethernet frames being transferred at the time the error is detected may result in the frame being lost (dropped).

7.3.8.1 Transmit Protocol Error

The transmit protocol error occurs when the MACPHY detects protocol errors in the transfer of transmit data chunks. These errors are usually due to SPI host firmware issues and should not occur in normal operation. The Transmit Protocol Error (TXPE) bit is set within the STATUS0 register when the following errors are detected.

Data Valid Without Start

A data header was received by the MACPHY indicating valid frame data (DV = 1) without a prior start-of-frame indication (SV = 1). The MACPHY ignores the transmit data chunk and continue waiting for a data header with SV = 1.

Repeated Frame Start

The MACPHY received two data headers indicating a start-of-frame (SV = 1) without an end-of-frame (EV = 1). The MACPHY may drop frame data until it sees an end-of-frame indicator (EV = 1) and begin accepting new frame data with a successive start-of-frame indicator (SV = 1). Alternatively, the MACPHY may drop the frame data from the previous start-of-frame indicator and begin accepting new frame data with the second (repeated) start-of-frame indicator.

7.3.8.2 Transmit Buffer Overflow Error

The SPI host may overflow the transmit buffer by attempting to write transmit frame data to the MACPHY when there is no transmit buffer space available as indicated by the Transmit Credit (TXC) field of the previous data footer being zero. In this condition, the MACPHY ignores the transmit data chunk and sets the Transmit Buffer Overflow Error (TXBOE) bit in the STATUS0 register.

When a transmit buffer overflow occurs, if data header SEQ bit functionality is supported and enabled (see Section 9.2.5.12) then frame data already in the MACPHY buffer is not dropped and the SPI host should resend the transmit data chunk with the same SEQ number once transmit credits are available. Failure of the SPI host to properly resend the data chunk may result in a corrupted frame being transmitted if the SPI host is relying on the MAC to add the frame check sequence to the frame.

If data header SEQ bit functionality is not supported or is disabled when a transmit buffer overflow occurs, then the frame shall be dropped by the MACPHY. The SPI host should detect the overflow condition in the STATUS0 register and perform appropriate actions to retransmit the frame.

7.3.8.3 Transmit Buffer Underflow Error

The transmit buffer underflow can only occur in cut-through mode of operation. Since the MACPHY may begin transmitting frame data onto the network before the entire frame is received from the SPI host, the SPI host must always send frame data to the MACPHY faster than the network. Should the SPI host fail to keep up with the network rate, the MAC may underflow on reads from the transmit buffer. When this occurs, the MACPHY shall terminate the frame being transmitted onto the network in such a way that it is invalid and cannot be received as a valid Ethernet frame. For instance, this shall result in the MAC asserting the TX_ER MII signal to the PHY or appending an invalid FCS to the frame. Additionally, the MACPHY shall ignore any additional frame data received from the SPI host until the the end of frame has been indicated with End Valid set (EV = 1).

The MACPHY sets the Transmit Buffer Underflow Error (TXBUE) bit in the STATUS0 register when a transmit buffer underflow occurs.

7.3.8.4 Loss of Framing Error

The CSn signal is deasserted prior to the expected end of the data chunk or control command. Any transmit frame data received in the short chunk is ignored by the MACPHY.

The MACPHY sets the Loss of Framing Error (LOFE) bit in the STATUS0 register when a loss of framing error occurs.

The MACPHY terminates and drops any transmit frame in progress on MOSI when the loss of framing error occurs. The SPI host should detect the loss of framing error condition in the STATUS0 register and perform appropriate actions to retransmit the frame.

The MACPHY terminates any receive frame in progress of being sent to the SPI host on MISO as a result of the short receive data chunk. The terminated receive frame is dropped, or may be automatically retried to the SPI host if supported by the implementation.

7.3.8.5 Receive Buffer Overflow

A receive buffer overflow occurs when the MACPHY receives frames from the network that fills the internal buffers. This occurs when the SPI host does not read frame data from the MACPHY fast enough. A receive buffer overflow error may occur in both store-and-forward and cut-through modes of operation.

The MACPHY sets the Receive Buffer Overflow (RXBOE) bit in the STATUS0 register when a receive buffer overflow occurs.

Once a receive buffer overflow occurs, the MACPHY terminates the frame being received by the PHY. No portion of the frame is transferred to the SPI host in store-and-forward mode. When operating in cut-through mode, the MACPHY terminates the frame (EV = 1) with Frame Drop set (FD = 1).

It is impossible for a receive buffer underflow to occur. Should there be no frame data to receive from the MACPHY when the SPI host initiates a data chunk transfer, the MACPHY sends an empty receive chunk payload with a data footer indicating no Data Valid (DV = 0).

7.4 Control Transaction Protocol

Control transactions consist of one or more control commands. Control commands are used by the SPI host to read and write registers within the MACPHY. Each control commands are composed of a 32-bit control command header followed by register data.

7.4.1 Control Command Header

The control command header is defined Figure 12 below.

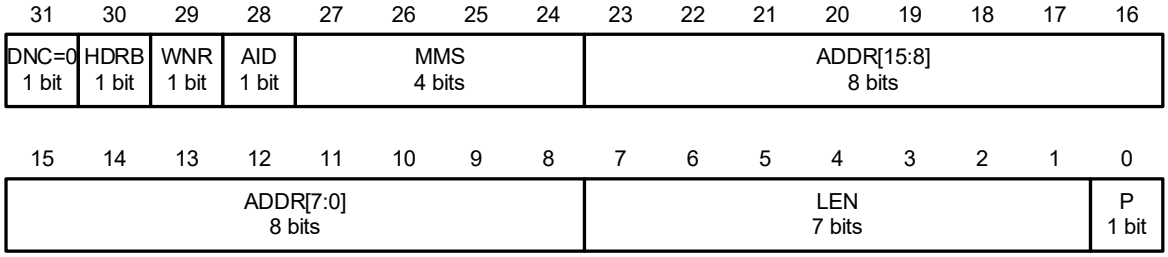

Figure 12: Control Command Header

Bit	Label	Name
31	DNC	Data-Not-Control
30	HDRB	Received Header Bad
29	WNR	Write-Not-Read
28	AID	Address Increment Disable
27..24	MMS	Memory Map Selector
23..8	ADDR	Address
7..1	LEN	Length
0	P	Parity Bit

Table 4: Control Command Header

DNC — Data-Not-Control flag. This flag specifies the type of SPI transaction. For control commands, this bit shall be '0'.

- 0 Control command
- 1 Data chunk

HDRB — Received Header Bad. When set by the MACPHY, indicates that a header was received with a parity error. The SPI host should always clear this bit. The MACPHY ignores the HDRB value sent by the SPI host on MOSI.

WNR — Write-Not-Read. This bit indicates if data is to be written to registers (when set) or read from registers (when clear).

AID — Address Increment Disable. When clear, the address will be automatically post-incremented by one following each register read or write. When set, address auto increment is disabled allowing successive reads and writes to occur at the same register address.

MMS — Memory Map Selector. This field selects the specific register memory map to access. See Table 6.

ADDR — Address. Address of the first register within the selected memory map to access.

LEN — Length. Specifies the number of registers to read/write. This field is interpreted as the number of registers minus 1 allowing for up to 128 consecutive registers read or written starting at the address specified in ADDR. A length of zero shall read or write a single register.

P Parity. Parity bit calculated over the control command header. Method used is odd parity. See Section 8.5.2.

The data header and control command header share the location and meaning of the Data-Not-Control flag (DNC) location in the header definition. This allows the MACPHY to determine whether the transfer is for data or control.

The MACPHY sets the Header Bad (HDRB) bit to '1' any time a header was received from the host with a parity error, and '0' otherwise. Additional details describing the handling of header parity errors are specified in Section 7.5.

The Write-Not-Read (WNR) bit indicates the type of the control command to be performed. Registers are read when WNR = 0 and written when WNR = 1.

Normally, when reading or writing multiple consecutive registers (LEN > 0), the address pointer will be post-incremented by one for each register accessed. The MACPHY may optionally support the ability to disable the auto-incrementing of the register address when the Address Increment Disable (AID) bit is set. When AID is set, the MACPHY continues to read or write the same register specified by the MMS and ADDR fields repeatedly, without incrementing, until the control command has completed. If this feature is not supported, the MACPHY ignores the AID field and the SPI host shall always clear AID.

Parity (P) uses odd parity to provide for control header protection as described in Section 8.5.2. When a header is received from the SPI host with a parity error, the MACPHY handles the error as described in Section 7.5.

The MACPHY may default to data mode at the assertion of CSn and remain in the data state until it receives a header with the DNC = 0 (indicating a control command). When this occurs, the MACPHY implementation needs some special consideration. Since any receive Ethernet frame data that is ready to be sent to the SPI host must be sent immediately upon assertion of CSn, it may happen that receive Ethernet frame data is pre-loaded into a shift register and sent out on MISO until the MACPHY state machine switches to the control state. The MACPHY must not lose any receive Ethernet frame data by preserving the data loaded into the shift register so that it can be reloaded on the next data transfer. Any receive Ethernet frame data must stop being sent on MISO within 31 bits. The first 32 bits sent out on MISO for control read and writes are considered invalid and shall not be used by the SPI host.

7.4.2 Control Write

A control write command is shown in Figure 13. The MACPHY ignores the final 32 bits of data from the SPI host at the end of the control write command. The write command and data is also echoed from the MACPHY back to the SPI host to enable the SPI host to identify which register write failed in the case of any bus errors. The format of the register write data and echoed register write data are described in Section 7.4.4.

Control write commands can write either a single register or multiple consecutive registers. When multiple consecutive registers are written, the address is automatically post-incremented by the MACPHY unless the Address Increment Disable (AID) bit is set in the control header. Writing to any unimplemented or undefined registers or register memory maps (MMS) shall be ignored and yield no effect.

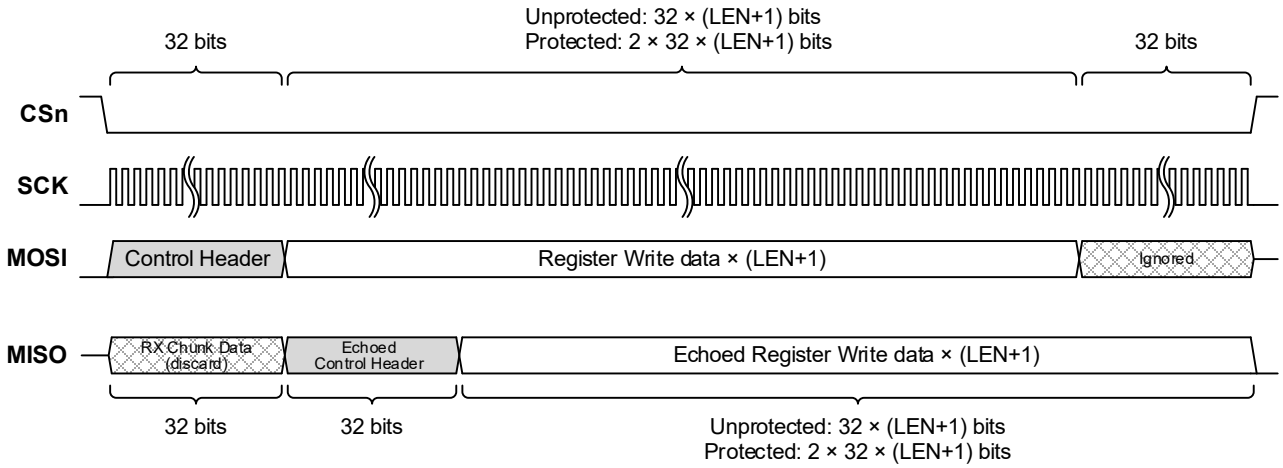

Figure 13: Control Commnd Write Transaction

When a control write command is followed by another control command (read or write), the new control header shall immediately follow the last word of the echoed register write data. The LEN field is used to calculate the expected length of the echoed register write data, and therefore the location of the next expected control header. The SPI host shall deassert CSn following the last word of the echoed register write data when the write command is the last command of the transaction. See Figure 13.

7.4.3 Control Read

A control read command is shown in Figure 14. The MACPHY ignores all data from the SPI host following the control header for the remainder of the control read command. Control read commands can read either a single register or multiple consecutive registers. When multiple consecutive registers are read, the address is automatically post-incremented by the MACPHY unless the Address Increment Disable (AID) bit in the control header is set. The MACPHY shall respond to all register memory maps (MMS) and addresses. Undefined and unimplemented register memory maps (MMS) and addresses shall return a value of zero when read. Writes to undefined and unimplemented registers memory maps and registers, read-only registers, and reserved registers and bits shall have no effect. The format of the register read data is described in Section 7.4.4.

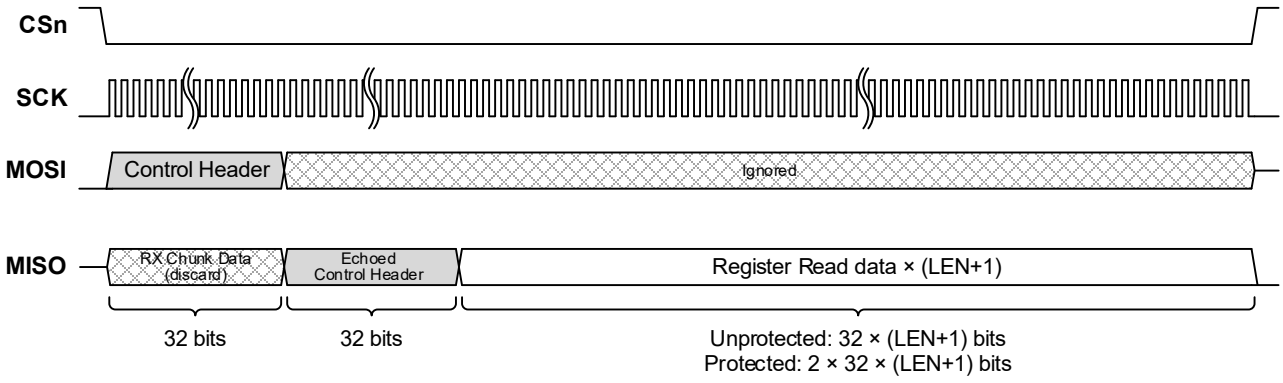

Figure 14: Control Command Read Transaction

When a control read command is followed by another control command (read or write), the new control header shall immediately follow the last word of the register read data. The LEN field is used to calculate the expected length of the register read data, and therefore the location of the next expected control header. The

SPI host shall deassert CSn following the last word of the register read data when the read command is the last command of the transaction. See Figure 14.

7.4.4 Register Data Format and Protection

All control data is transferred as 32-bit words to support access to registers up to 32 bits wide. Registers which are less than 32 bits in width shall be right aligned with leading zeros within the 32-bit word as shown in Figure 15. Register data shall be transferred most significant byte first and most significant bit first within the 32-bit word.

The register data being read or written can be protected against simple bit errors. When enabled by setting the Protection Enable (PROTE) bit in the CONFIG0 register (see Table 12), protection is accomplished by duplication of each 32-bit word containing register data with its ones' complement. Errors are detected at the receiver by performing a simple exclusive-OR (XOR) of each received 32-bit word containing register data with its received complement and detecting if there are any zeros in the result.

Should any errors be detected, the MACPHY shall assert the Control Data Protection Error (CDPE) bit in STATUS0 and not perform a write of any register data in which an error has been detected. The MACPHY shall not attempt to correct any errors when it echoes the received register write data back to the SPI host.

The SPI host may also validate and compare the echoed register write data to what it sent to the MACPHY to determine which registers the MACPHY received incorrectly and did not write. Additionally, register data read by the MACPHY should be validated by the SPI host to prevent performing incorrect actions as a result of incorrect register values.

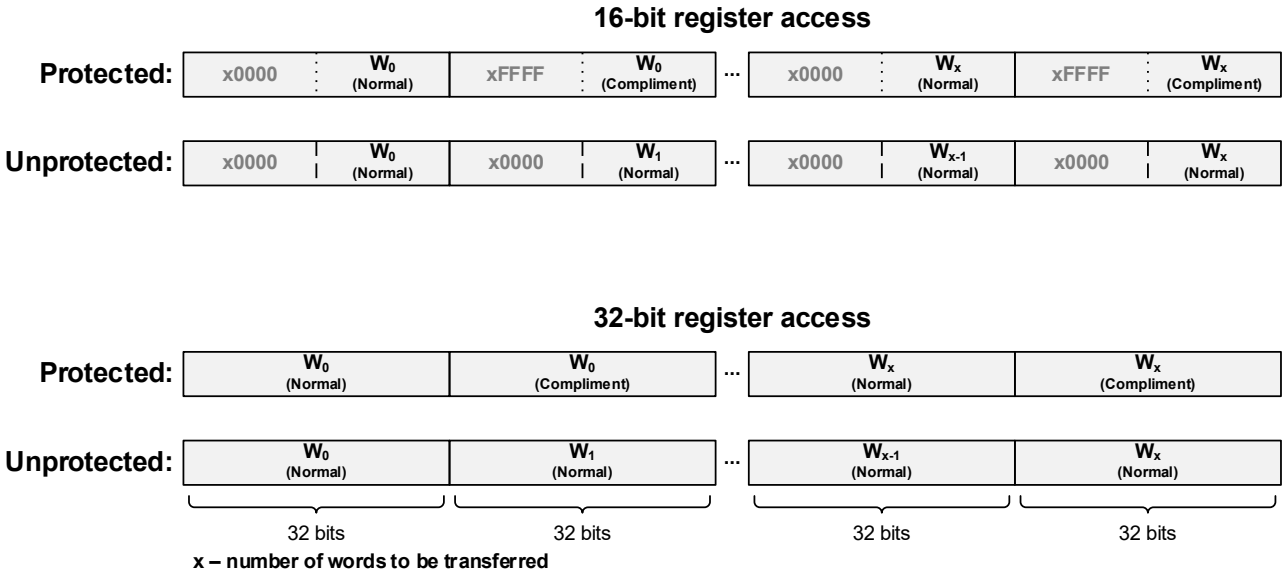

Figure 15: Control Command Read/Write Register Data Transfer Format

7.5 Header Parity Error Handling

The MACPHY verifies the parity of each received header. Should a header (data or control) be received with a parity error, the MACPHY cannot determine anything about the transaction. Therefore, it terminates transmission of receive chunk data on MISO within the first 32-bit word. Beginning with the second 32-bit

word repeating for each subsequent word through the remainder of the transaction until CSn is deasserted, the MACPHY outputs a 32-bit constant value of 0x40000000 to allow the SPI host to identify the error.

The constant value of 0x40000000 represents an assumed echoed control header (DNC = 0), with a Header Bad indicator (HDRB = 1), and correct odd parity (P = 0). The SPI host should monitor for the reception of this constant value in the location it would expect to find the echoed control header (in case of control commands) and data footer (in the case of data chunks). When HDRB is received set, the SPI host should deassert CSn at the appropriate opportunity and restart the desired control or data transaction.

As it cannot be determined if a control command or data chunk were intended in the bad header field, all data received from the SPI host on MOSI is ignored by the MACPHY until CSn is deasserted. The MACPHY drops any frame being received from the SPI host when the error occurred and ignores any additional data sent by the SPI host on MOSI until a new frame starts (SV = 1). Additionally, the MACPHY terminates any frame being sent to the SPI host on MISO when the error occurred. The MACPHY continues receiving and sending frames again once CSn has been deasserted. The SPI host, upon detecting the error code, may retransmit the frames that were dropped as a result of the error assuming the frame that was terminated remains in the SPI host's memory buffer. If a frame being sent from the MACPHY to the SPI host was terminated as a result of the error, then the MACPHY indicates the termination to the SPI host by sending DV = 1, EV = 1, and FD = 1 in the first data footer following a new assertion of CSn. Any completed frames that were received by the MACPHY from the SPI host in transmit chunks prior to the error will be processed. The SPI host should also process any completed frames sent by the MACPHY in receive chunks before the error occurred.

Additionally, any control commands prior to the header parity error will be performed. The SPI host should monitor the received data from the MACPHY to identify the error code in the echoed control command word to determine which control commands failed so they may be repeated in a subsequent control transaction.

If data header SEQ bit functionality is supported and enabled, then frames in progress when the error occurred are not terminated. Instead, the SPI host may continue sending the frame by resending the transmit data chunk with the same SEQ value that was ignored by the MACPHY when the error was detected. Similarly, the MACPHY may continue sending the receive frame to the SPI host following CSn deassertion and assertion starting with the receive data chunk that was to be sent when the error was detected.

7.6 Configuration Synchronization Handling

Following a reset or any other detectable event that may potentially cause the loss of configuration, the MACPHY shall clear the SYNC bit in the CONFIG0 register to its default value of '0' (see Section 9.2.5.2). This is used to indicate that the MACPHY is in an 'initial' unconfigured state. The value of the CONFIG0 register SYNC bit is always transmitted to the SPI host as the SYNC bit in the data footer at the end of receive data chunks (see Section 7.3.7). Additionally, as a result of a reset, the RESETC bit in the STATUS 0 register shall be set to '1' (see Section 9.2.8.8).

When in this state with SYNC = 0, the MACPHY ignores all transmit data chunks and will not transfer Ethernet frame data within receive data chunks.

It is possible for the MACPHY and SPI host to attempt operation with different data chunk payload size when the MACPHY is not configured resulting in incorrect operation. This may occur, for example, when the MACPHY defaults to a data chunk payload size of 64 bytes but the SPI host continues operating with a data chunk payload size of 32 bytes or less. In this condition, the SPI host will not be able to properly locate the

data footer at the end of the receive data chunk. Therefore, when the MACPHY is unconfigured (SYNC = 0), it monitors each header to determine if the transaction is a data transaction by validating the header parity and detecting DNC = 1. If so, the MACPHY transmits a complete data footer on MISO beginning with the second 32-bit word of the data transaction and continuing until CSn is deasserted. If the MACPHY detects a header for a control command, it continues to process the control command as normally, allowing the SPI host to read/write registers and configure the MACPHY. The first 32-bit word output from the MACPHY when SYNC = 0 is undefined.

When a data footer is received with SYNC = 0, the SPI host shall configure the MACPHY for proper operation before attempting to transfer any Ethernet frames. Once the MACPHY has been configured, the SPI host shall write the CONFIG0 register setting SYNC = 1. Once set by the SPI host, it shall not be possible to write a '0' to the CONFIG0 register SYNC bit causing it to clear again. Once SYNC is set to a '1', the MACPHY begins normal operation, processing transmit data chunks with frame data for transmission to the network, and generating receive data chunks with frame data received from the network.

7.7 IRQn generation and handling

The IRQn is asserted when any of the following internal assertion conditions are met and deasserted when all of the following internal deassertion conditions are met (logical OR).

RCA – Receive Chunks Available

Asserted:	The MACPHY detects CSn deasserted and the previous data footer had no receive data chunks available (RCA = 0). The IRQn pin will be asserted when receive data chunks become available for reading while CSn is deasserted.
Deasserted:	On reception of the first data header following CSn being asserted.

TXC – Transmit Chunk Credits Available

Asserted:	The MACPHY detects CSn deasserted and the previous data footer indicated less than TXCTHRESH of transmit credits available (TXC < TXCTHRESH). The IRQn pin will be asserted when transmit credits become available while CSn is deasserted. The minimum number of transmit credits that must be available before asserting IRQn is configured by the TXCTHRESH field of the CONFIG0 register (see Table 12).
Deasserted:	On reception of the first data header following CSn being asserted.

EXST – Extended Status Event

Asserted:	The MACPHY detects CSn deasserted and the previous data footer had no Extended Status assertion (EXST = 0). The IRQn pin will be asserted when an event occurs that would cause the extended status bit become set while CSn is deasserted. The Extended Status bit is set at any time a bit in the STATUS0 or STATUS 1 registers in 9.2.8 and 9.2.9, respectively, has been set that has not been masked by the IMASK0 or IMASK1 registers in 9.2.11 and 9.2.12, respectively. Vendors may implement additional status bits that may cause the Extended Status bit to be set.
Deasserted:	On reception of the first data header following CSn being asserted.

SYNC – Configuration Required (SYNC bit cleared)

Asserted: The IRQn pin will be asserted when the MACPHY has been reset and the CONFIG0 register SYNC bit is clear (see Table 12). The assertion of IRQn in this case indicates that the MACPHY is ready to be configured by the SPI host.

Deasserted: On reception of the first data header following CSn being asserted, or the SPI host writes to the STATUS0 register clearing the RESETC bit (see Section 9.2.8.8).

When the SPI host detects an asserted IRQn from the MACPHY, it should initiate a data chunk transfer to obtain a current data footer. If the source of the interrupt was receive frame data becoming available, then the MACPHY may begin sending received data to the SPI host immediately. If the SPI host knows that transmit credits were available (TXC ≠ 0) when it received the last data footer prior to deasserting CSn, it may also write transmit frame data to the MACPHY, otherwise if it is possible that the assertion of the interrupt is due to transmit credits becoming available (the last data footer contained TXC = 0) the SPI host shall not transmit frame data (DV = 0).

Once the SPI host has received an data footer indicating the source of the interrupt, it should act to resolve the interrupt. If the interrupt was caused by receive data chunks available or transmit credits greater than the configured threshold being available (TXC ≥ TXCTHRESH, see Table 12), the interrupt is already resolved by the data chunk transaction previously completed to receive the current data footer. The SPI host shall continue reading receive frame data, or update its internal transmit buffer credit counter. If the source of the interrupt was an Extended Status (EXST = 1), the SPI host shall perform a control command read of the STATUS0 register and any additional vendor specific registers which contain status bits that may cause EXST = 1 and act accordingly to resolve these interrupt sources. Should the data footer indicate that the interrupt was due to a loss of configuration Synchronization (SYNC = 0, indicating a reset of the MACPHY), the SPI host shall terminate all pending frame transfers, reconfigure the MACPHY, and write a ‘1’ to the CONFIG0 register SYNC bit.

7.8 Frame Timestamp Capture

The MACPHY may optionally support the capturing of timestamps on Ethernet frames received from and transmitted to the network. Support for this option is indicated by the Frame Timestamp Capability (FTSC) bit in the STDCAP register (see Table 10). When supported, this feature is enabled and configured by the Frame Timestamp Enable (FTSE) and Frame Timestamp Select (FTSS) bits in the CONFIG0 register (see Table 12).

When frame timestamp capturing is supported, the MACPHY implements the capturing of 32-bit and 64-bit timestamps. The FTSS bit in the CONFIG0 register (Table 12) selects the size and format of the timestamps that will be appended to frames on ingress from the network and captured on egress to the network.

The format of the 32-bit and 64-bit timestamps are shown in Figure 16. Both formats are based on the timestamp format standardized in IEEE 1588-2008 [3] (PTPv2) and IEEE 802.1AS-2011 [4] (gPTP). Each timestamp consists of a 30-bit field containing a nanosecond counter. The nanosecond field rolls over to zero every 10^9 nanoseconds.

The 32-bit timestamp utilizes bits 30 and 31 to indicate seconds and will roll over every 4 seconds.

Rather than implementing a standard 48-bit seconds field as defined in PTPv2 and gPTP, the 64-bit timestamp defines a 32-bit seconds. Assuming the standard epoch, the 64-bit timestamp will rollover in the year 2106. Bits 30 and 31 are always zero.

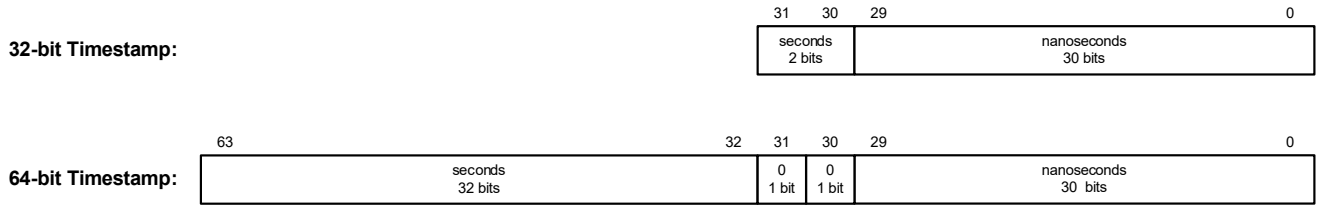

Figure 16: Frame Timestamp Capture Formats

7.8.1 Receive Frame Timestamp Capturing

When frame timestamp capturing is enabled, a received frame may have a timestamp appended to the beginning of the frame in receive data chunk payload. The timestamp shall be captured at the detection of Start-of-Frame Delimiter (SFD) ingress to the receiver. The SPI host can detect that a timestamp has been appended to the frame by examining the Receive Timestamp Appended (RTSA) bit of the data footer.

When a receive timestamp has been appended to the frame, as indicated by RTSA = 1, the size of the appended timestamp may be 32 or 64 bits as configured by the Frame Timestamp Select (FTSS) bit in the CONFIG0 register (Table 12). When enabled, the SV bit will be set to '1' and SWO will indicate the word offset into the receive data chunk payload for the beginning of the appended timestamp. The received frame immediately follows the timestamp value.

The timestamp shall begin on any 32-bit word boundary of the receive data chunk payload and is transferred most significant bit first. Regardless of the size of the configured timestamp, 32 or 64 bits, the received frame also begins on a 32-bit word boundary. It is permissible for one or both of the 32-bit words of a timestamp to appear at the end of one receive data chunk payload and the frame to begin at the beginning of the next receive data chunk payload. In such case, the SV and SWO indicate the beginning of the appended timestamp value, and not the beginning of the frame.

Since the appended receive timestamp is not covered by the embedded frame check sequence (FCS), an odd parity check is computed over the appended 32-bit or 64-bit timestamp and inserted as the Receive Timestamp Parity (RTSP) bit into the same data footer that indicates SV = 1 and the Start Word Offset (SWO) of the appended timestamp.

When frame timestamp capturing is unsupported or disabled, the RTSA and RTSP bits in the data footer shall be '0'. Additionally, RTSA and RTSP shall be zero any time SV = 0.

7.8.2 Transmit Frame Timestamp Capturing

Support for transmit frame timestamp capturing is enabled when receive frame timestamp capturing is enabled. The timestamp will be captured at the detection of Start-of-Frame Delimiter (SFD) egress from the transmitter.

When the SPI host wishes to capture the timestamp of a frame as it is transmitted on to the network, it shall transmit the frame to the MACPHY in a transmit data chunk containing non-zero value for the Timestamp Capture (TSC) field in the data header. The non-zero value of the TSC field indicates which of three Transmit

Timestamp Capture registers (TTSCA, TTSCB, and TTSCC) to place the captured timestamp. See Sections 9.2.13-9.2.18. The TSC field is only valid when the transmit data chunk contains the start of an Ethernet frame and SV = 1, otherwise the MACPHY shall ignore the TSC field. A TSC field value of zero indicates to the MACPHY that it shall not capture a timestamp on the transmission of the frame to the network.

There may be a delay from when the MACPHY receives the frame from the SPI host until it is transmitted onto the network, for example, due to network access controls (CSMA/CD and PLCA). Once the frame has been transmitted and the timestamp captured into the requested timestamp capture register, the appropriate Transmit Timestamp Capture Available (TTSCAA, TTSCAB, TTSCAC) status bit will be asserted within the STATUS0 register. See Section 9.2.8. If unmasked, the assertion of these status bits will set the Extended Status bit within the data footer indicating to the SPI host that is should perform a control command read of the STATUS0 register to determine the source of the Extended Status, and read the appropriate timestamp capture register. The format of the timestamp captured into the transmit timestamp capture register is configured by the Frame Timestamp Select (FTSS) bit in the CONFIG0 register (Table 12).

When frame timestamp capturing is unsupported or disabled, the MACPHY shall ignore the TSC field in the data header and the SPI host should write TSC = <00>.

7.9 Cut-Through Operation

Normal transfer of frame data through the MACPHY occurs in a store-and-forward mode of operation. In this mode, a full frame of data is received by the MACPHY from the SPI host (or the network) before it is transmitted out to the network (or the SPI host). If supported by the MACPHY, an optional cut-through mode may be enabled which can reduce latency of transferring frames through the MACPHY.

Support for cut-through mode of operation is indicated by the Cut-Through Capability (CTC) bit in the STDCAP register (see Table 10). Cut-through may be enabled in either the receive or transmit directions (or both) by respectively setting the Receive Cut-Through Enable (RXCTE) or Transmit Cut-Through Enable (TXCTE) bits in the CONFIG0 register (see Table 12). Additional vendor specific configuration may be necessary prior to enabling cut-through operation. If cut-through operation is desired, it shall be configured and enabled prior to any frame data being transferred through the MACPHY and before the configuration synchronization (SYNC) bit is set within the CONFIG0 register.

Store-and-forward operation requires every frame to be received from the network and processed by the MAC in its entirety prior to being sent to the SPI host. This permits the MAC to validate received frames preventing undesired frames or frames with errors from being forwarded to the SPI host. Examples of frame errors include frame check sequence errors, alignment errors (odd number of received nibbles), runt frames, frames too large, etc. Additionally, the MAC can filter frames to allow only frames addressed to the local individual MAC address or group address to be received. Receive cut-through operation means that the MACPHY begins transferring the received frames to the SPI host before the complete frame has been received. As a result, invalid or undesired received frames may be transferred over the SPI before being detected by the MAC. To avoid placing the burden of validating frames onto the SPI host, the MACPHY sets the Frame Drop (FD) flag in the data footer to a ‘1’ indicating to the SPI host that the MAC has determined that the frame shall be dropped (ignored).

8 SPI Protocol State Diagrams

The detailed behavior of the SPI protocol elements described in Section 7, along with the interaction with the MAC layer, is captured by the following state diagrams, functions and related variables.

The MACPHY transmit and receive functionality shall conform to the state diagrams and related constants, variables, timers and functions defined in this section. Should there be a discrepancy between a state diagram and descriptive text, the state diagram prevails.

8.1 State diagram notation

The convention specified in Clauses 147.1.3.1 and 147.1.3.2 of [1] are adopted with the following exceptions:

- Statements inside the state boxes are blocking, therefore executed sequentially from top to bottom.
- Operators precedence is defined in Table 5 with the highest priority operator listed at the top.

Table 5: Operator Precedence (Highest to Lowest)

OPERATOR	DESCRIPTION
[], [:]	subscript, subscript range
!	logical NOT
&	address-of
*, /, %	* : logical AND, arithmetic multiply / : integer division % : remainder
+, -	+ : logical OR, arithmetic sum - : arithmetic subtraction
~, ^	bitwise NOT, bitwise XOR
len()	returns the length of the supplied array or bit stream
=, ≠, <, >	equal, not equal, less, greater comparisons
⇐	assignment

8.2 Variables

The following variables are used in the state diagrams.

SCK — This variable is TRUE on the positive edge of the input SPI clock. Otherwise it is FALSE.
Values: TRUE or FALSE

MOSI — The SPI MOSI (master-out, slave-input) pin value.
Values: ‘1’ or ‘0’

MISO — The SPI MISO (master-in, slave-out) pin value.
Values: ‘1’ or ‘0’

CS	The SPI CSn (chip-select) pin value. The CSn signal is active-low (ASSERTED = 0). Values: ASSERTED or DEASSERTED
IRQ	The MACPHY IRQn pin value. The IRQn pin is active-low (ASSERTED = 0). Values: ASSERTED or DEASSERTED
cfg_sync	This variable reflects the value of the SYNC bit in the CONFIG0 register (see Section 9.2.5.2). It is TRUE when SYNC is set, FALSE otherwise. Values: TRUE or FALSE
ctl_protected	This variable reflects the value of the PROTE bit in the CONFIG0 register (see Section 9.2.5.11). It is TRUE when PROTE is set, FALSE otherwise. Values: TRUE or FALSE
chunk_sz	This variable is set after the configured data chunk payload size, CPS, (see Section 9.2.5.13) in bytes. Values: 8, 16, 32 or 64
exst	This variable reflects the logical-OR of all internal interrupt status sources (see Sections 9.2.8 and 9.2.9) of the MACPHY logically AND with their masks (see Sections 9.2.11 and 9.2.12). It is TRUE when any of the unmasked interrupt status sources are asserted.
pexst	Internal state variable used to track if the EXST bit was sent in the previous data footer. Values: TRUE, FALSE or X
tx_stop	Internal state variable used to indicate a transmit stop condition. This happens when the transmit credit count reaches 0 and the SPI host is waiting for the IRQn to be asserted. Values: TRUE or FALSE
pseq	Internal state variable used to track the SEQ data header bit coming from the SPI host to detect whether a chunk has been transferred twice. Values: TRUE, FALSE or X
pseq_en	Internal variable used for indicating that support for SEQ data header bit is enabled. It is set within the state diagrams according to the SEQE bit in the CONFIG0 register (see Section 9.2.5.12).
xmitting	Internal state variable used to keep track of whether the SPI host is in the process of transmitting a frame. It is used for protocol errors checking and for IRQn generation. Values: TRUE or FALSE
recv	Internal state variable used to keep track of whether the SPI host is in the process of receiving a frame. It is used for protocol errors checking and for IRQn generation. Values: TRUE or FALSE
resync_rqueue	Internal state variable used when terminating a receive frame being sent to the SPI host. When TRUE, will cause the first 32-bit word sent to the SPI host following CSn assertion to terminate a previously started frame. Values: TRUE or FALSE

chunk_error — Internal state variable used to take special actions when an error is detected while transferring data chunks.
Values: TRUE or FALSE

cnt — General purpose counter.
Values: 32-bit unsigned integer

ctsz — Internal variable used for computing the effective size, in bits, of a control command. It is set within the state diagrams according to the PROTE bit in the CONFIG0 register (see Section 9.2.5.11).
Values: 32-bit unsigned integer

reset — This variable reflects the logical-OR of all reset sources of the MACPHY and is TRUE when any of the reset sources are asserted. Reset sources include power-on reset (POR), software reset (see Section 9.2.4.2), and an external RESET pin (if implemented).

rxc_avail — This variable reflects the number of additional receive data chunks of frame data within the MAC buffer available for reading by the SPI host. This value is available as RCA in the Buffer Status Register (see Section 9.2.10.3).
Values: 32-bit unsigned integer

txc_free — This variable reflects the number of consecutive transmit data chunks of frame data that the SPI host can write without overflowing the the MAC buffer. This value is available as TXC in the Buffer Status Register (see Section 9.2.10.2).
Values: 32-bit unsigned integer

txc_thresh — Internal variable used for the minimum number of free chunks within the MAC transmit buffer that needs to be available before asserting IRQn. It is set within the state diagrams according to the TXCTHRESH field in the CONFIG0 register (see Section 9.2.5.6).

txb — This variable represents a structured buffer reflecting an SPI to MACPHY transfer (both control and data), including header and payload. The layout of the 'txb' structure is represented by the following SystemVerilog™-like definition.

```
union {
   struct {
      logic DNC;
      logic SEQ;
      logic NORX;
      logic [4:0] RSVD0;
      logic [1:0] VS;
      logic DV;
      logic SV;
      logic [3:0] SWO;
      logic RSVD1;
      logic EV;
      logic [5:0] EBO;
      logic [1:0] TSC;
      logic [4:0] RSVD2;
      logic P;
```

```
   } hdr;

   struct
   {
      logic DNC;
      logic HDRB;
      logic WNR;
      logic AID;
      logic [3:0] MMS;
      logic [15:0] ADDR;
      logic [6:0] LEN;
      logic P;
   } ctl;

   byte [0 : chunk_sz+4 - 1] data;
   logic [0 : (chunk_sz+4) * 8 - 1] bit;
} txb;
```

rxb This variable represents a structured buffer reflecting a MACPHY to SPI transfer (both control and data), including data footer and payload. The layout of the ‘rxb’ structure is represented by the following SystemVerilog-like definition.

```
union {
   struct {
      byte [0 : chunk_sz-1] dummy;
      logic EXST;
      logic HDRB;
      logic SYNC;
      logic [4:0] RCA
      logic [1:0] VS;
      logic DV;
      logic SV;
      logic [3:0] SWO;
      logic FD;
      logic EV;
      logic [5:0] EBO;
      logic RTSA;
      logic RTSP;
      logic [4:0] TXC;
      logic P;
   } ftr;

   byte [0 : chunk_sz+4 - 1] data;
   logic [0 : (chunk_sz+4) * 8 - 1] bit;
} rxb;
```

8.3 Constants

The following constants are used in the state diagrams.

HIGHZ	High-Impedance. This is a special constant representing that the digital or analog pin shall configure its output with an equivalent series resistance greater than 33 kΩ.
X	Undefined value. Implementations are allowed to define any value in place of 'X'.
{0}	Abbreviation to indicate a structure/union with all of its members and sub-members set to '0'.
ASSERTED	Indicates either '1' (HIGH) or '0' (LOW) as appropriate for the target signal to be active.
DEASSERTED	Indicates either '0' (LOW) or '1' (HIGH) as appropriate for the target signal not to be active.
TRUE	A true conditions, defined as a logical '1'
FALSE	A false condition, defined as a logical '0'
ENOERR	No error condition. This represents clearing any error bit in the STATUS0 register (see Section 9.2.8). This includes the TXPE, TXBOE, TXBUE, and the RXBOE bits.
ELOF	Loss-of-framing condition, indicating CSn was deasserted before the expected end of a data chunk or control command read/write. This error maps on the LOFE bit in the STATUS0 register (see Section 9.2.8.7).
EPROTO	Protocol Sequence Error. This error maps on the TXPE bit in the STATUS0 register (see Section 9.2.8.14).
ETXBUSY	Transmit Overflow Error. This maps to the TXBOE bit in the STATUS0 register (see Section 9.2.8.13).
ERESET	Reset Complete. This maps to the RESETC bit in the STATUS0 register (see Section 9.2.8.8).
EHDR	Header Parity Error. This maps to the HDRE bit in the STATUS0 register (see Section 9.2.8.9).

8.4 Timers

The following timers are used in the state diagrams.

resync_timer	Minimum chip-select deassertion time as required by the implementation.

8.5 Functions

The following tasks and functions are used in the state diagrams.

8.5.1 TRI

This function returns the supplied value ‘val’ when ‘en’ is true, otherwise it returns high-impedance (HIGHZ).

```
TRI(val, en):
IF en = ASSERTED THEN
   return val
ELSE
   return HIGHZ
END
```

8.5.2 CRC1

This function returns an odd parity bit computed over the bits supplied in ‘data’. A ‘1’ is returned when the number of bits set to one in the supplied bit stream is even (resulting in an odd number of ones when the parity is included), ‘0’ otherwise.

```
CRC1(data):
res ⇐ 1
FOR (i ⇐ 0; i < len(data); i ⇐ i + 1) DO
   res ⇐ res ^ data[i]
DONE
return res
```

8.5.3 read_reg

This function shall read the register in the MMS/ADDR specified in the control header 'ctl' and return the 32-bit value in the buffer 'buf'.

Additionally, if the system is configured for protected control commands, this function shall append the negated read value to the same buffer.

```
read_reg(ctl, buf)
```

8.5.4 write_reg

This function shall write the 32-bit word contained in ‘buf’ to the register in the MMS/ADDR specified in the control header 'ctl'.

```
write_reg(ctl, buf)
```

8.5.5 set_ext_status

This function sets the specified error ‘code’ in the STATUS0 register (See Section 9.2.8) and updates the data footer ‘ftr’ based on the unmasked status register bits. The global variable chunk_error is also set if ‘code’ indicates any data transfer error.

```
set_ext_status(ftr, code):
IF code != ENOERR THEN
   chunk_error ⇐ TRUE
ELSE
   chunk_error ⇐ FALSE
```

```
END
ftr.EXST ⇐ exst
```

8.5.6 check_proto

This function checks the state of the 'xmitting' variable against the received protocol header to identify possible out of sequence operations. It returns TRUE if the current state is consistent, FALSE otherwise. Eventually, the variable 'xmitting' is updated to reflect the new status.

```
check_proto(xmitting, hdr):
valid ⇐ (xmitting ^ hdr.SV + hdr.SV * hdr.EV) = 1
IF valid THEN
   IF xmitting THEN
      IF hdr.EV * !hdr.SV THEN
         xmitting ⇐ FALSE
      END
   ELSE
      IF hdr.SV * !hdr.EV
         xmitting ⇐ TRUE
      END
   END
END
return valid
```

8.5.7 update_rx

This function checks whether the MAC has any receive frame data ready to be conveyed to the SPI host and updates the receive data chunk payload / footer accordingly. If the receive chunk has been already updated by a previous call to update_rx, successive calls have no effect until the chunk is delivered to the SPI host. The 'rxb' parameter is the receive buffer to be updated (including footer). The 'off' parameter specifies at which offset the data shall be copied within the rxb buffer.

```
update_rx(rxb, offs):
IF rxb.ftr.DV = 0 THEN
   n ⇐ rqueue_read(chunk_sz - offs, &rxb.data[offs])
   rxb.ftr.RCA ⇐ 0
   IF n ≠ 0 THEN
      rxb.ftr.DV ⇐ 1
      rxb.ftr.SWO ⇐ offs / 4
      rxb.ftr.EBO ⇐ 0
      rxb.ftr.SV ⇐ 0;
      rxb.ftr.EV ⇐ 0;

      IF (rqueue_sfd() >= 0) THEN
         rxb.ftr.SV ⇐ 1;
         rxb.ftr.SWO ⇐ rxb.ftr.SWO + rqueue_sfd() / 4
      END

      IF (rqueue_eof() >= 0) THEN
         rxb.ftr.EV ⇐ 1;
         rxb.ftr.EBO ⇐ offs + rqueue_eof()
      END
```

```
            rxc_avail = rqueue_chunks()

            IF rxc_avail > 31 THEN
               rxb.ftr.RCA ⇐ 31
            ELSE
               rxb.ftr.RCA ⇐ rxc_avail
            END
         END
         rxb.ftr.P ⇐ CRC1(rxb.ftr.bit[0:30])
      END
```

8.5.8 rqueue_read

This function shall copy up to 'n' bytes from MAC receive memory into 'buf' and return the number of bytes copied.

When reading past the end of a frame, if there is not enough data in the MAC receive memory to fulfill the entire request (`rqueue_sz() < n`), rqueue_read() shall stop reading at the end of the current frame[1]. Repeated calls to this function return the same data until rqueue_pop() is called.

```
rqueue_read(n, buf)
```

8.5.9 rqueue_sz

This function returns the number of bytes available in the MAC receive memory.

```
rqueue_sz()
```

8.5.10 rqueue_chunks

This function returns the number of chunks available in the MAC receive memory. This function should take into account chunk size, padding between frames, and the limitation that only one frame may start and end within a single chunk.

```
rqueue_chunks()
```

8.5.11 rqueue_sfd

If the last call to rqueue_read() contained the start of a frame, this function returns the relative offset of the first byte of the frame within the read data. Otherwise this function returns a negative value.

According to the rqueue_read() definition, the returned offset is supposed to be aligned to a 32-bit boundary.

```
rqueue_sfd()
```

8.5.12 rqueue_eof

If the last call to rqueue_read() contained the end of a frame, this function returns the relative offset of the last byte of the frame within the read data. Otherwise this function returns a negative value.

```
rqueue_eof()
```

[1] This can only happen when operating in receive cut-through mode

8.5.13 rqueue_pop

This function removes from the queue the number of bytes read during the last call to rqueue_read().

This shall not include any padding added by the read function as a result of crossing a frame boundary. This means that this function shall remove from the queue the effective number of frame(s) bytes read.

```
rqueue_pop()
```

8.5.14 rqueue_reset

This function resets the receive queue terminating and dropping any frame currently in transmission to the SPI host. If the last call to rqueue_read() contained frame data, this function will continue to read and pop data from the receive queue until the end of the frame.

```
rqueue_reset()
```

8.5.15 tqueue_sz

This function returns the number of bytes available in the MAC transmit memory.

```
tqueue_sz()
```

8.5.16 tqueue_write

This function appends exactly 'n' bytes from 'buf' to MAC transmit memory.

The 'eof' parameter is set to indicate that the last byte of 'buf' is the last byte of the frame to be transmitted. The MAC is supposed to start the transmission of the frame when the 'eof' parameter is TRUE.

The 'sfd' parameter is set to indicate that the first byte of 'buf' is the beginning of a new frame. This indication is supposed to be used internally by the MAC transmit logic.

```
tqueue_write(n, buf, sfd, eof)
```

8.5.17 tqueue_reset

This function resets the transmit queue terminating and dropping any frame currently being received from the SPI host.

```
tqueue_reset()
```

8.6 TX/RX State Diagrams

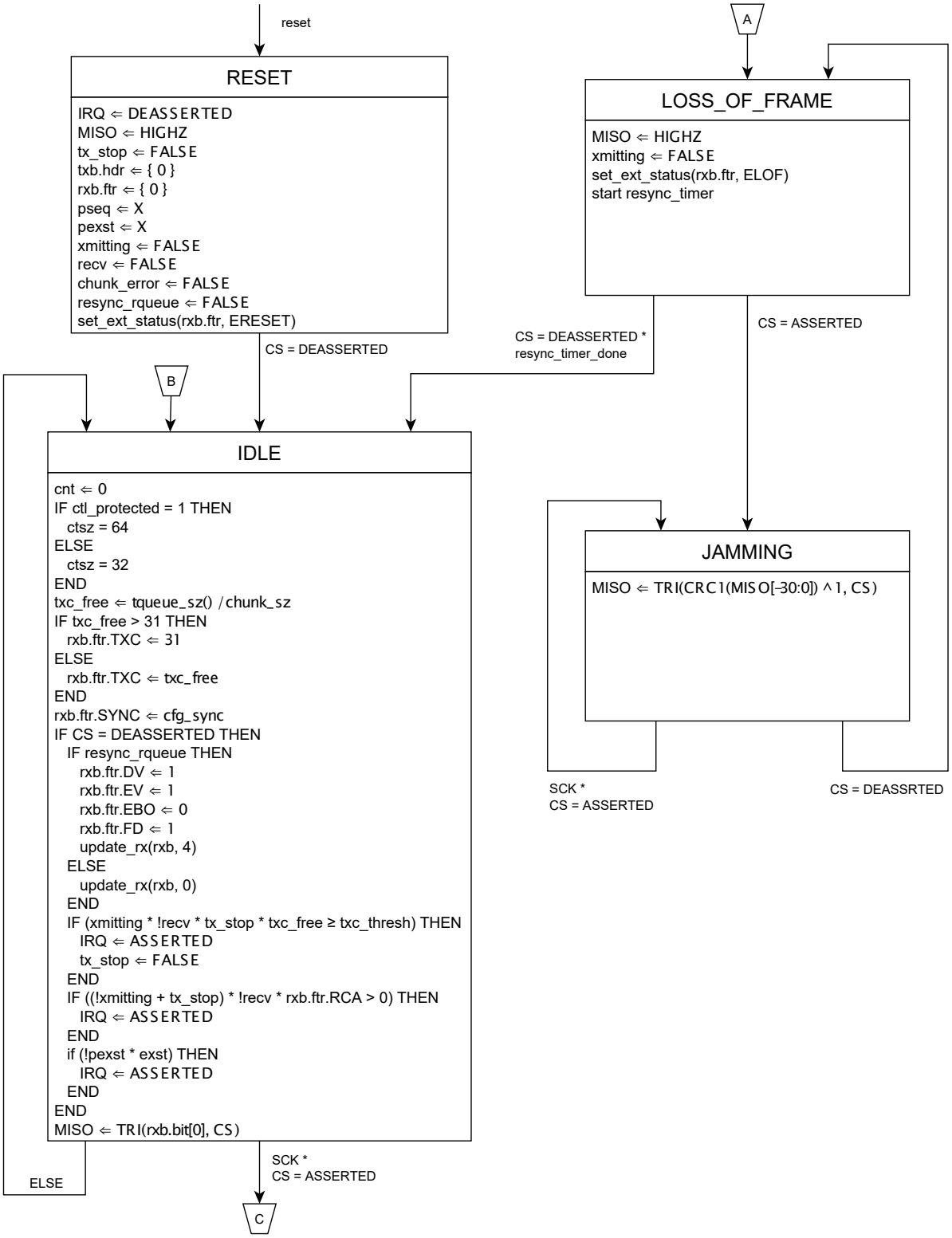

Figure 17: TX/RX State Diagram (part 1 of 5)

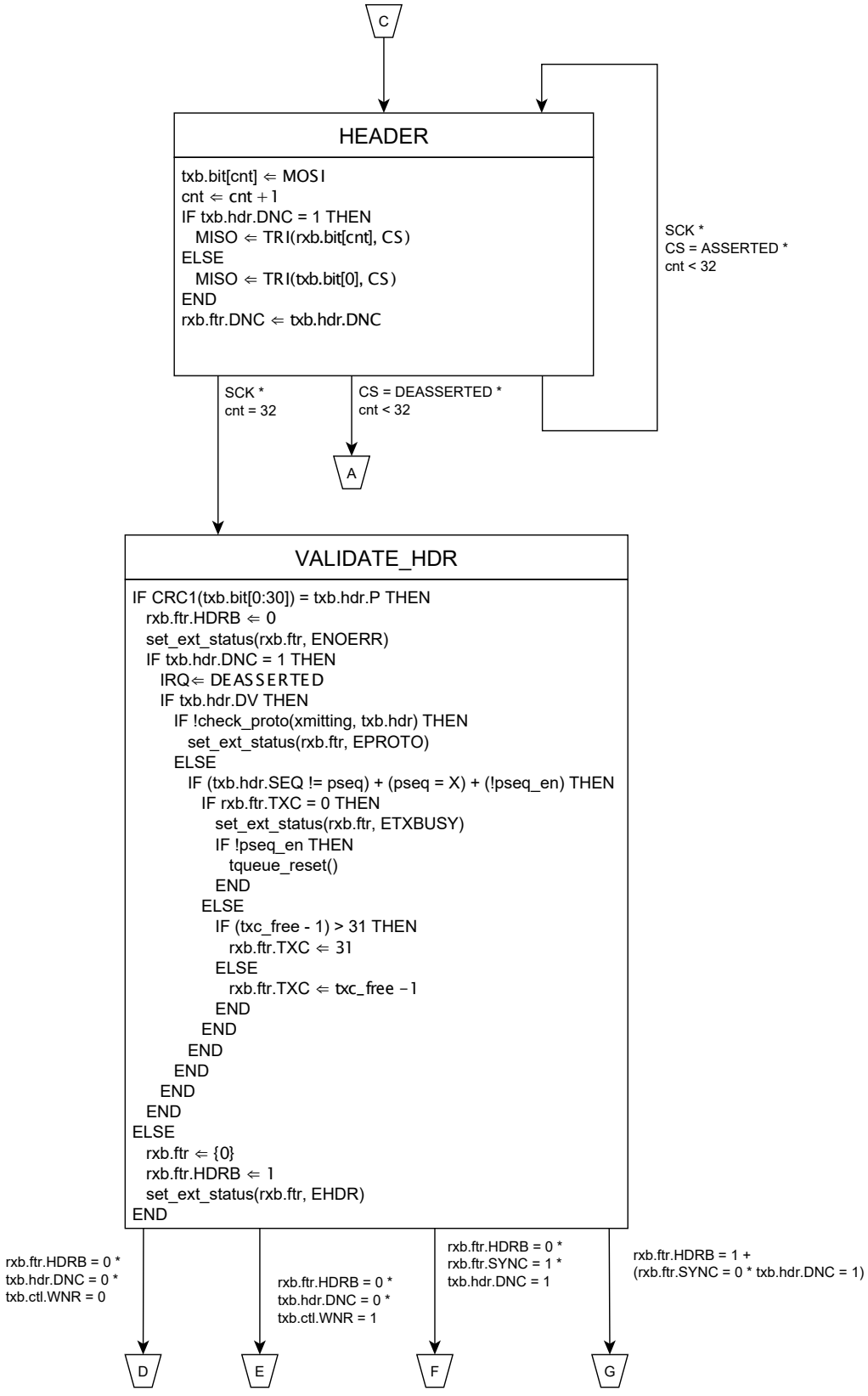

Figure 18: TX/RX State Diagram (part 2 of 5)

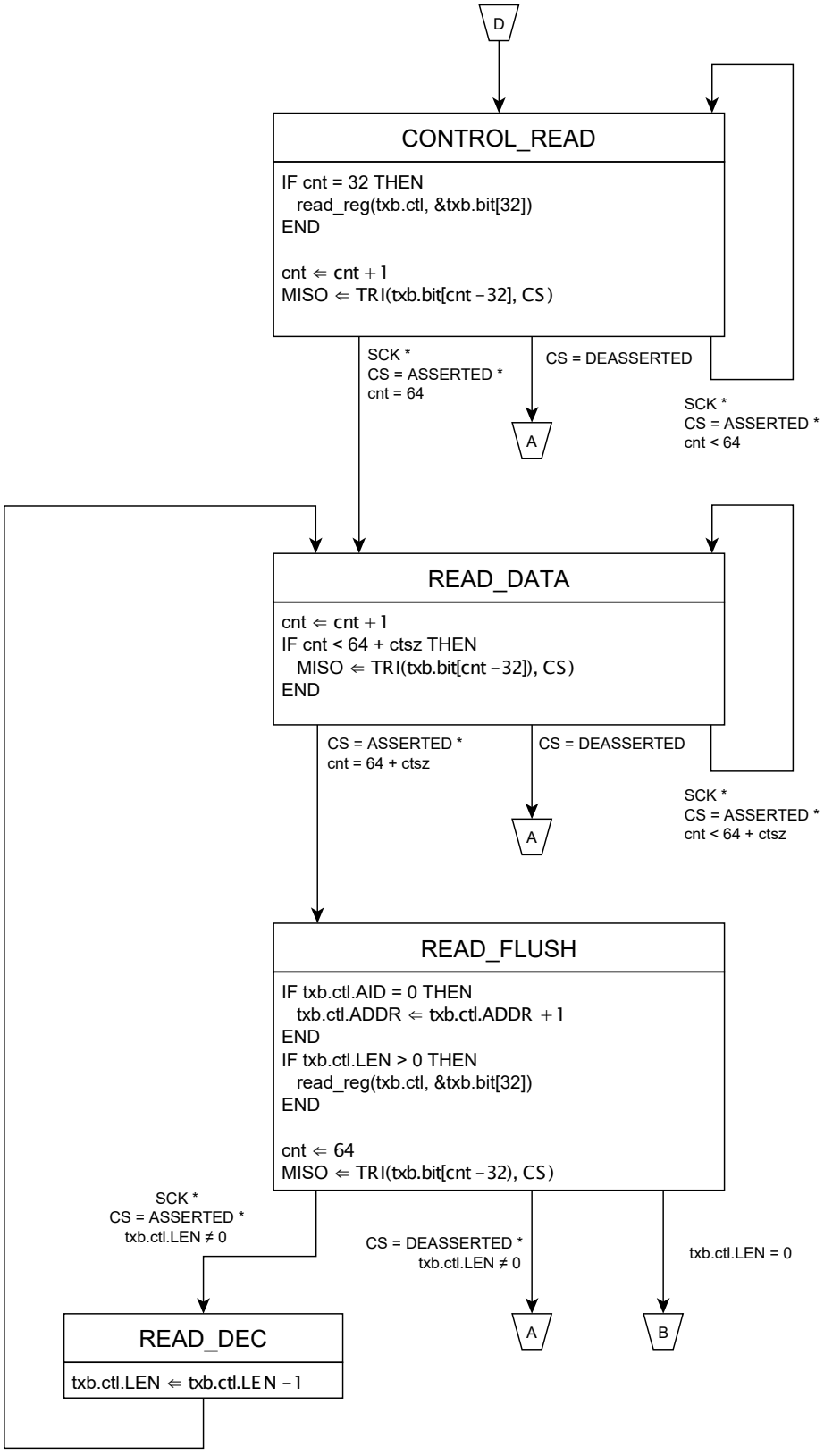

Figure 19: TX/RX State Diagram (part 3 of 5)

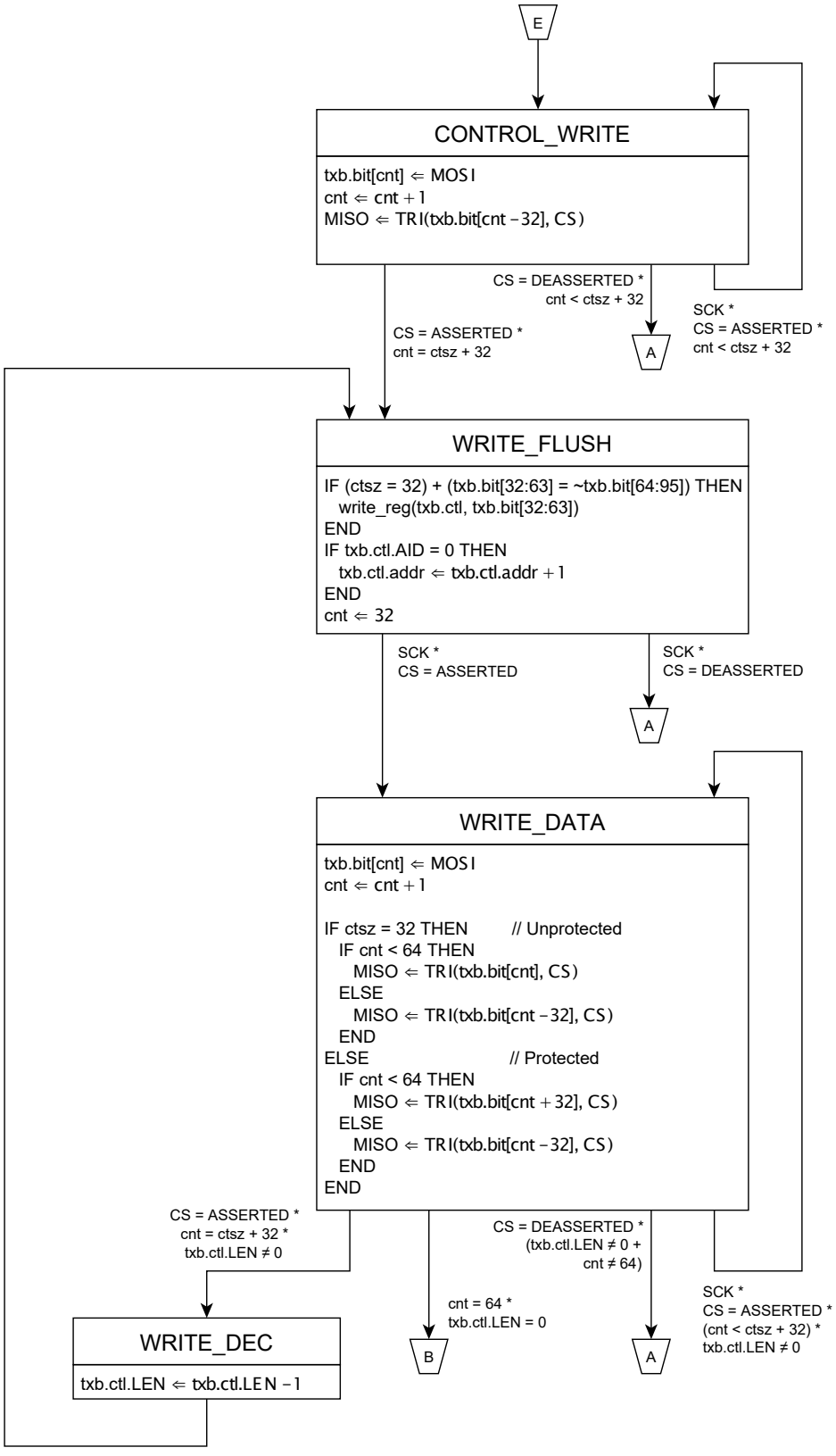

Figure 20: TX/RX State Diagram (part 4 of 5)

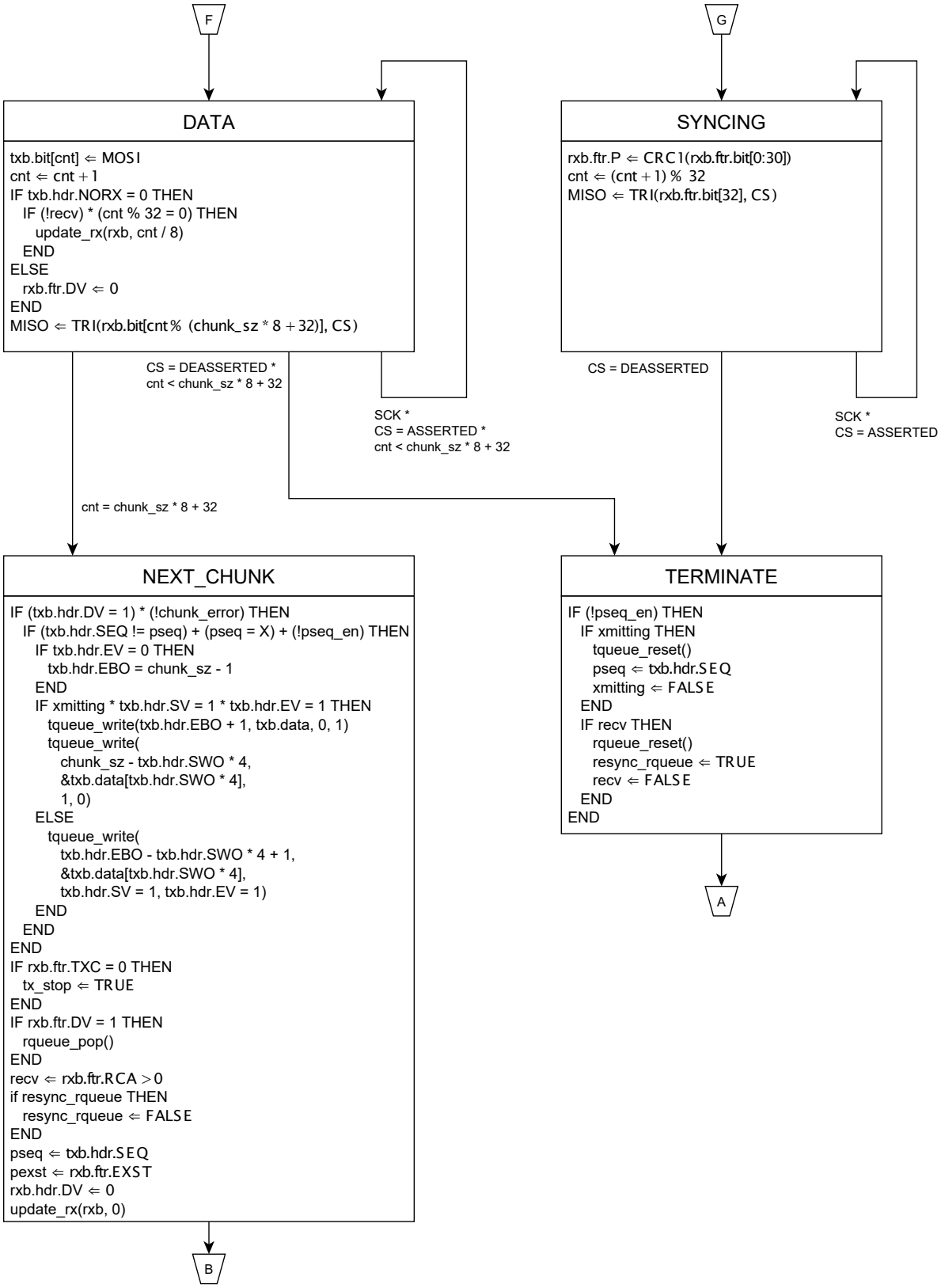

Figure 21: TX/RX State Diagram (part 5 of 5)

9 Control and Status Registers

9.1 Register Memory Map Selector

The register memory map selector (MMS) field of the control command header (see Section 7.4.1) allows for 16 memory maps to be allocated as groups of registers. Each memory map may consist of up to 65536 registers. Each MMS is assigned as shown in Table 6.

Table 6: Register Memory Map Selector (MMS) Assignment

MMS	Width	Memory Map Description
0	32	Standard Control and Status
1	-	MAC
2	16	PHY – PCS Registers (MMD 3)
3	16	PHY – PMA/PMD Registers (MMD 1)
4	16	PHY – Vendor Specific and PLCA Registers (MMD 31)
5	16	PHY – Auto-Negotiation Registers (MMD 7)
6	16	PHY – Power Unit (MMD 13)
7-9	-	Reserved
10-15	-	Vendor Specific

An implementation shall support at least one of two PHY register access methods: direct and indirect. The access methods supported by an implementation are indicated by the Indirect PHY Register Access Capability (IPRAC) and Direct PHY Register Access Capability (DPRAC) bits in the Standard Capabilities Register (0x0002). See Sections 9.2.3.3 and 9.2.3.4.

Indirect access to PHY registers as defined in IEEE 802.3 [1,2] is performed through the indirect MDIO/MDC registers MDIOACCn. See Section 9.2.19. Eight MDIOACCn registers are defined to permit the reading and/or writing of up to eight PHY registers with a single control command. The indirect access method permits access to all Clause 22 and Clause 45 registers. It guards against future changes to IEEE 802.3 [2] register mappings and it allows access to future devices that may have multiple PHY instances. Indirect access is slower and requires more control commands than direct access.

For more efficient access to PHY registers, the PHY registers may be mapped directly to SPI register memory space. When registers are mapped directly they shall be mapped according to this specification. As shown in Table 6, MMS 2 through MMS 6 are allocated for the direct mapping of PHY registers. Clause 22 standard registers and Clause 22 extended registers (Clause 29) are directly mapped into MMS 0 as shown in Table 7. MMS 7 through MMS 9 are reserved for future standardization.

Vendor specific registers may be mapped into MMS 10 though MMS 15. When directly mapped, PHY vendor specific registers in MMD 30 or MMD 31 would be mapped into the vendor specific MMS 10 through MMS 15.

9.2 Register Memory Map 0 - Standard Control and Status Registers

Register memory map 0 is allocated to standard MACPHY registers. The registers within MMS 0 are 32 bits in width.

Table 7 shows the mapping of standard registers within MMS 0.

Table 7: Standard Control and Status registers (MMS 0)

Address	Name	Description
0x0000	IDVER	Identification Version Register
0x0001	PHYID	PHY Identification Register
0x0002	STDCAP	Standard Capabilities Register
0x0003	RESET	Reset Control and Status Register
0x0004	CONFIG0	Configuration Register #0
0x0005	CONFIG1	Configuration Register #1
0x0006	CONFIG2	Configuration Register #2
0x0007	RSVD	Reserved for future expansion
0x0008	STATUS0	Status Register #0
0x0009	STATUS1	Status Register #1
0x000A	RSVD	Reserved for future expansion
0x000B	BUFSTS	Buffer Status Register
0x000C	IMSK0	Interrupt Mask #0
0x000D	IMSK1	Interrupt Mask #1
0x000E - 0x000F	RSVD	Reserved for future expansion
0x0010	TTSCAH	Transmit Timestamp Capture Register A (High)
0x0011	TTSCAL	Transmit Timestamp Capture Register A (Low)
0x0012	TTSCBH	Transmit Timestamp Capture Register B (High)
0x0013	TTSCBL	Transmit Timestamp Capture Register B (Low)
0x0014	TTSCCH	Transmit Timestamp Capture Register C (High)
0x0015	TTSCCL	Transmit Timestamp Capture Register C (Low)
0x0016 - 0x001F	RSVD	Reserved for future expansion
0x0020	MDIOACC0	MDIO Access Register 0
0x0021	MDIOACC1	MDIO Access Register 1
0x0022	MDIOACC2	MDIO Access Register 2
0x0023	MDIOACC3	MDIO Access Register 3
0x0024	MDIOACC4	MDIO Access Register 4
0x0025	MDIOACC5	MDIO Access Register 5
0x0026	MDIOACC6	MDIO Access Register 6
0x0027	MDIOACC7	MDIO Access Register 7
0x0028 - 0xFEFF	RSVD	Reserved for future expansion
0xFF00 – 0xFF1F	Clause 22	PHY Clause 22 Standard Registers
0xFF20 – 0xFF3F	Clause 29	PHY Clause 29 (Clause 22 Extended) Registers
0xFF40 - 0xFFFF	RSVD	Reserved for future expansion

9.2.1 Identification Register (0x0000)

	IDVER	Identification Version Register		
Bit	**Label**	**Description**		**Default**
31..8	RSVD	Reserved for future use	RO	0
7..4	MAJVER	OA Major Version	RO	0001
3..0	MINVER	OA Minor Version	RO	0000

Table 8: IDVER - Identification Version Register

9.2.1.1 RSVD

Reserved for future use. Writing to these bits shall have no effect on the MACPHY.

9.2.1.2 MAJVER

Major version identifier of the OPEN Alliance Serial 10BASE-T1x MACPHY Interface Specification supported by this device

9.2.1.3 MINVER

Minor version identifier of the OPEN Alliance Serial 10BASE-T1x MACPHY Interface Specification supported by this device.

9.2.2 PHY Identification Register (0x0001)

	PHYID	PHY Identification Register		
Bit	**Label**	**Description**		**Default**
31..10	OUI	Organizationally Unique Identifier (bits 2:23)	RO	-
9..4	MODEL	Manufacturer's Model Number	RO	-
3..0	REVISION	Manufacturer's Revision Number	RO	-

Table 9: PHYID – PHY Identification Register

The format of the PHY ID register is the same as described in clause 22.2.4.3.1 of [2].

9.2.2.1 OUI

The 22 bits in the OUI field correspond to the 22 most significant bits of the manufacturer's assigned 24-bit Organizationally Unique Identifier (OUI). The OUI is arranged into the PHYID register such that OUI bit 2 is located at PHYID bit 31 and OUI bit 23 is located at PHYID bit 10.

9.2.2.2 MODEL

The manufacturer's model number is used to identify the device. Use of this field is vendor specific and beyond the scope of this document.

9.2.2.3 REVISION

The manufacturer's product revision number is used to indicate a revision level of the device. Typically, this field contains an indication of the silicon revision, however use of this field is vendor specific and beyond the scope of this document.

9.2.3 Standard Capabilities Register (0x0002)

	STDCAP	Standard Capabilities Register		
Bit	**Label**	**Description**		**Default**
31..11	RSVD	Reserved for future use	RO	0
10	TXFCSVC	Transmit Frame Check Sequence Validation Capability	RO	-
9	IPRAC	Indirect PHY Register access Capability	RO	-
8	DPRAC	Direct PHY Register Access Capability	RO	-
7	CTC	Cut-Through Capability	RO	-
6	FTSC	Frame Timestamp Capability	RO	-
5	AIDC	Address Increment Disable Capability	RO	-
4	SEQC	Transmit data header Sequence Capability	RO	-
3	RSVD	Reserved for future use	RO	0
2..0	MINCPS	Minimum supported Chunk Payload Size	RO	-

Table 10: STDCAP – Standard Capabilities Register

9.2.3.1 RSVD

Reserved for future use. Writing to these bits shall have no effect on the MACPHY.

9.2.3.2 TXFCSVC

Transmit Frame Check Sequence Validation Capability. Indicates the ability to validate the FCS appended by and received from the SPI host. When this bit is set it also indicates the ability of the MAC to be configured to accept egress frames with padding and FCS appended by the SPI host, and send ingress frames to the SPI host with the received FCS.

0 Transmit FCS validation is not supported
1 Transmit FCS validation is supported

9.2.3.3 IPRAC

Indirect PHY Register Access Capability. Indicates if PHY registers are indirectly accessible through the MDIO/MDC registers MDIOACCn.

0 PHY registers are not indirectly accessible
1 PHY registers are indirectly accessible

9.2.3.4 DPRAC

Direct PHY Register Access Capability. Indicates if PHY registers are directly accessible within the SPI register memory space.

0 PHY registers are not directly accessible
1 PHY registers are directly accessible

9.2.3.5 CTC

Cut-Through Capability. Indicates if the MACPHY device supports cut-through transfer of frames through the MACPHY to/from the network.

0 Cut-through frame transfer is not supported
1 Cut-through frame transfer is supported

9.2.3.6 FTSC

Frame Timestamp Capability. Indicates if the MACPHY device supports the capturing of timestamps on frame network ingress/egress.

0 Timestamp capture on frame ingress/egress is not supported
1 Timestamp capture on frame ingress/ egress is supported

9.2.3.7 AIDC

Address Increment Disable Capability. Indicates if the MACPHY device supports the disabling of the automatic post-incrementing of the register address in control command reads and writes through the AID bit in the control header. See Section 7.4.1.

0 Address increment disable is not supported
1 Address increment disable is supported

9.2.3.8 SEQC

Transmit data header Sequence Capability. Indicates if the MACPHY supports monitoring the SEQ bit sent by the SPI host in the data header.

0 Transmit data header Sequence bit monitoring is not supported
1 Transmit data header Sequence bit monitoring is supported

9.2.3.9 MINCPS

Minimum supported Chunk Payload Size. Indicates the minimum size data chunk payload that may be configured into the CPS field of the CONFIG0 register. The minimum supported data chunk payload Size is 2^N, where N is the value of this bitfield.

3 Minimum supported data chunk payload size is 8 bytes
4 Minimum supported data chunk payload size is 16 bytes
5 Minimum supported data chunk payload size is 32 bytes
6 Minimum supported data chunk payload size is 64 bytes

All other values are reserved.

9.2.4 Reset Control and Status Register (0x0003)

	RESET	Reset Control and Status Register		
Bit	**Label**	**Description**		**Default**
31..1	RSVD	Reserved for future use	RO	0
0	SWRESET	Software Reset	R/WSC	0

WSC = Self Clearing when written to '1'

Table 11: RESET – Reset Control and Status Register

9.2.4.1 RSVD

Reserved for future use. Writing to these bits shall have no effect on the MACPHY.

9.2.4.2 SWRESET

MACPHY Software Reset. The action of writing a '1' to this bit shall fully reset the MACPHY, including the integrated PHY, to an initial state including but not limited to resetting all state machines and registers to their default value. When this bit is set, the reset shall not occur until CSn is deasserted to allow for the control command write to complete. This bit is self-clearing.

9.2.5 Configuration Register #0 (0x0004)

	CONFIG0	Configuration Register 0		
Bit	**Label**	**Description**		**Default**
31..16	RSVD	Reserved for future use	RO	0
15	SYNC	Configuration Synchronization	RW1	0
14	TXFCSVE	Transmit Frame Check Sequence Validation Enable	RW	0
13	CSARFE	CSn Align Receive Frame Enable	RW	0
12	ZARFE	Zero-Align Receive Frame Enable	RW	0
10..11	TXCTHRESH	Transmit Credit Threshold	RW	00
9	TXCTE	Transmit Cut-Through Enable	RW	0
8	RXCTE	Receive Cut-Through Enable	RW	0
7	FTSE	Frame Timestamp Enable	RW	0
6	FTSS	Frame Timestamp Select	RW	0
5	PROTE	Control data read/write Protection Enable	RW	0
4	SEQE	Transmit data header Sequence bit support Enable	RW	0
3	RSVD	Reserved for future use	RO	0
2..0	CPS	Chunk Payload Size	-	110

Table 12: CONFIG0 - Configuration Register 0

9.2.5.1 RSVD

Reserved for future use. Writing to these bits shall have no effect on the MACPHY.

9.2.5.2 SYNC

Configuration Synchronization. The state of this bit is reflected in the data footer SYNC bit. This bit shall default to '0' upon reset. Once written to a '1' by the SPI host, writing '0' shall not clear this bit.

0 The MACPHY has been reset and is not configured
1 The MACPHY is configured

9.2.5.3 TXFCSVE

Transmit Frame Check Sequence Validation Enable. When set, the final 4 octets of all Ethernet frames received will be validated as an Ethernet FCS. See Section 7.3.1. In addition to setting this bit, the MAC must be configured to expect the SPI host to pad frames to the minimum frame size and append the FCS to the frame. The MAC should also be configured to pass the FCS to the SPI host with the received FCS for validation.

Details for configuring the MAC is implementation specific and beyond the scope of this specification.

9.2.5.4 CSARFE

CSn Align Receive Frame Enable. When set, all receive Ethernet frames data shall start at the beginning of the first receive data chunk payload following CSn assertion. The Start Word Offset (SWO) will always be zero. Receive frames may begin within any receive data chunk of the transaction when this bit is clear.

9.2.5.5 ZARFE

Zero-Align Receive Frame Enable. When set, all receive Ethernet frames data shall be aligned to start at the beginning of any receive data chunk payload with a Start Word Offset (SWO) of zero. Receive frames may begin anywhere within the receive data chunk payload when this bit is clear.

9.2.5.6 TXCTHRESH

Transmit Credit Threshold. This field configures the minimum number of transmit credits (TXC) of free buffer chunks that must be available for writing before IRQn will be asserted.

00 ≥ 1 credit (default)
01 ≥ 4 credits
10 ≥ 8 credits
11 ≥ 16 credits

9.2.5.7 TXCTE

Transmit Cut-Through Enable. When supported by this MACPHY device, this bit enables the cut-through mode of frame transfer through the MACPHY device from the SPI host to the network.

9.2.5.8 RXCTE

Receive Cut-Through Enable. When supported by this MACPHY device, this bit enables the cut-through mode of frame transfer through the MACPHY device from the network to the SPI host.

9.2.5.9 FTSE

Frame Timestamp Enable. When supported by this MACPHY device, this bit enables capturing of frame network ingress/egress timestamps. See Section 7.8 for details.

0 Frame ingress/egress timestamps are disabled
1 Frame ingress/ egress timestamps are enabled

9.2.5.10 FTSS

Frame Timestamp Select. When supported by this MACPHY device and enabled by FTSE = 1, this bit configures the size and format of the timestamps appended to ingress frames and capture on request of egress frames. See Section 7.8 for timestamp format and size details.

0 32-bit timestamps
1 64-bit timestamps

9.2.5.11 PROTE

Control data read/write Protection Enable. When set, all control data written to and read from the MACPHY will be transferred with its complement for detection of bit errors as defined in Section 7.4.

0 Control data read/write protection is disabled (unprotected)
1 Control data read/write rotection is enabled (protected)

Alternative methods may be implemented for enabling and disabling control data read/write protection at reset (e.g., external configuration pin or non-volatile memory). When this is implemented, the PROTE bit should default to the configured state.

9.2.5.12 SEQE

Transmit data header Sequence bit support Enable. When supported by this MACPHY device, this bit enables MACPHY monitoring of the SEQ bit sent in the transmit data header by the SPI host.

0 Transmit data header Sequence bit monitoring is disabled. The MACPHY ignores the SEQ bit in the data header.

1 Transmit data header Sequence bit monitoring is enabled. The MACPHY monitors the SEQ bit in the data header and accepts transmit data payloads only when the SEQ bit changes.

When unsupported by MACPHY implementation, this bit shall be read-only with a value of '0'.

9.2.5.13 CPS

Chunk Payload Size. Configures the data chunk payload size to 2^N, where N is the value of this bitfield. The minimum possible data chunk payload size is 8 bytes or N = 3. The default data chunk payload size is 64 bytes, or N = 6. Once the configuration Synchronization (SYNC) bit has been set, the data chunk payload size shall not be changed without a reset of the MACPHY. The minimum supported data chunk payload size for this MACPHY device is indicated in the CPSMIN field of the CAPABILITY register.

3 Data chunk payload size is configured to 8 bytes
4 Data chunk payload size is configured to 16 bytes
5 Data chunk payload size is configured to 32 bytes
6 Data chunk payload size is configured to 64 bytes

All other values are reserved.

9.2.6 Configuration Register #1 (0x0005)

CONFIG1 Configuration Register 1

Bit	Label	Description		Default
31..0	RSVD	Reserved for future use	RO	0

Table 13: CONFIG1 - Configuration Register 1

9.2.6.1 RSVD

Reserved for future use. Writing to these bits shall have no effect on the MACPHY.

9.2.7 Configuration Register #2 (0x0006)

CONFIG2	Configuration Register 2			
Bit	**Label**	**Description**		**Default**
31..0	VS	Vendor Specific	R/W	0

Table 14: CONFIG2 - Configuration Register 2

9.2.7.1 VS

Vendor Specific. Configuration bits reserved for vendor specific implementation.

9.2.8 Status Register #0 (0x0008)

STATUS0	Status Register 0			
Bit	**Label**	**Description**		**Default**
31..13	RSVD	Reserved for future use	RO	0
12	CDPE	Control Data Protection Error	R/W1C	0
11	TXFCSE	Transmit Frame Check Sequence Error	R/W1C	0
10	TTSCAC	Transmit Timestamp Capture Available C	R/W1C	0
9	TTSCAB	Transmit Timestamp Capture Available B	R/W1C	0
8	TTSCAA	Transmit Timestamp Capture Available A	R/W1C	0
7	PHYINT	PHY Interrupt	R/W1C	0
6	RESETC	Reset Complete	R/W1C	1
5	HDRE	Header Error	R/W1C	0
4	LOFE	Loss of Framing Error	R/W1C	0
3	RXBOE	Receive Buffer Overflow Error	R/W1C	0
2	TXBUE	Transmit Buffer Underflow Error	R/W1C	0
1	TXBOE	Transmit Buffer Overflow Error	R/W1C	0
0	TXPE	Transmit Protocol Error	R/W1C	0

W1C = Write ‘1’to Clear

Table 15: STATUS0 - Status Register

Status bits are edge triggered and set on the assertion of the underlying source.

9.2.8.1 RSVD

Reserved for future use.

9.2.8.2 CDPE

Control Data Protection Error. When control data read/write protection is enabled (see PROTE, Section 9.2.5.11), this bit indicates that the MACPHY detected an error in protected control write data received from the host. When control data protection is not implemented, this bit shall be reserved with a read-only value of zero.

9.2.8.3 TXFCSE

Transmit Frame Check Sequence Error. When set, this bit indicates that a frame was received from the SPI host with an invalid FCS appended. When transmit FCS validation is not implemented (TXFCSVC is clear, see Section 9.2.3.2), this bit shall be reserved with a read-only value of zero.

9.2.8.4 TTSCAC

Transmit Timestamp Capture Available C. When set, this bit indicates that a timestamp was captured for a frame transmitted onto the network and is available within the Transmit Timestamp Capture C (TTSCC) resisters. When frame timestamp capturing is not implemented (FTSC is clear, see Section 9.2.3.6), this bit shall be reserved with a read-only value of zero.

9.2.8.5 TTSCAB

Transmit Timestamp Capture Available B. When set, this bit indicates that a timestamp was captured for a frame transmitted onto the network and is available within the Transmit Timestamp Capture B (TTSCB) resisters. When frame timestamp capturing is not implemented (FTSC is clear, see Section 9.2.3.6), this bit shall be reserved with a read-only value of zero.

9.2.8.6 TTSCAA

Transmit Timestamp Capture Available A. When set, this bit indicates that a timestamp was captured for a frame transmitted onto the network and is available within the Transmit Timestamp Capture C (TTSCA) resisters. When frame timestamp capturing is not implemented (FTSC is clear, see Section 9.2.3.6), this bit shall be reserved with a read-only value of zero.

9.2.8.7 PHYINT

Physical Layer Interrupt. When set, this bit indicates a service request from the underlying physical layer block. Many physical layer implementations support an interrupt output for signaling events to the station controller. This bit is optional and will be implemented only if the underlying physical layer supports generating interrupts to a higher level. When not implemented, this bit shall be reserved with a read-only value of zero.

9.2.8.8 RESETC

Reset Complete. This bit is set when the MACPHY reset is complete and ready for configuration. When it is set, it will generate a non-maskable interrupt assertion on IRQn to alert the SPI host. Additionally, setting of the RESETC bit shall also set EXST = 1 in the receive data footer until this bit is cleared by action of the SPI host writing a '1'.

9.2.8.9 HDRE

Header Error. When set, this bit indicates that the MACPHY has detected an invalid header received from the SPI host. The invalid header will be due to a parity check error.

9.2.8.10 LOFE

Loss of Framing Error. When set, this bit indicates that the MACPHY has detected an early deassertion of CSn prior to the expected end of a data chunk or control command.

9.2.8.11 RXBOE

Receive Buffer Overflow Error. When set, this bit indicates that the receive buffer (from the network) has overflowed and receive frame data was lost.

9.2.8.12 TXBUE

Transmit Buffer Underflow Error. When set, this bit indicates that the transmit buffer (from the SPI host) has underflowed and the transmit frame may have been lost.

9.2.8.13 TXBOE

Transmit Buffer Overflow Error. When set, this bit indicates that the transmit buffer (from the SPI host) has overflowed and the transmit frame may have been lost.

9.2.8.14 TXPE

Transmit Protocol Error. When set, this bit indicates that a transmit data chunk protocol error has occurred.

- Data chunk received with DV= 1 but without a prior SV = 1
- Data chunk received with SV= 1 but with no EV = 1 (repeated SV = 1 received)

9.2.9 Status Register #1 (0x0009)

STATUS1 Status Register 1

Bit	Label	Description		Default
31..0	VS	Vendor Specific	R/W1C	0

W1C = Write '1'to Clear

Table 16: STATUS1 - Status Register

9.2.9.1 VS

Vendor Specific. Status bits reserved for vendor specific implementation.

9.2.10 Buffer Status Register (0x000B)

BUFSTS Buffer Status Register

Bit	Label	Description		Default
31..5	RSVD	Reserved	RO	0
15..8	TXC	Transmit Credits Available	RO	--
7..0	RCA	Receive Chunks Available	RO	0x00

Table 17: BUFSTS – Buffer Status Register

9.2.10.1 RSVD

Reserved for future use.

9.2.10.2 TXC

Transmit Credits Available. This field contains the number of consecutive transmit data chunks of frame data the SPI host can write without overflowing the MAC. Reading this field allows the SPI host to queue up the number of transmit chunks available into a single DMA, if desired.

The value in this field saturates at 31 and is sent in the 5-bit TXC field of every receive data footer. See Section 7.3.7.

The default (maximum) number of transmit buffer credits available is implementation specific.

9.2.10.3 RCA

Receive Chunks Available. This field contains the number of additional receive data chunks of frame data currently available for the SPI host to read. Reading this field allows the SPI host to queue up the number of receive chunks available into a single DMA, if desired.

The value in this field saturates at 31 and is sent in the 5-bit RCA field of every receive data footer. See Section 7.3.7.

The maximum number of receive chunks that may be available is implementation specific.

9.2.11 Interrupt Mask Register #0 (0x000C)

	IMASK0	Interrupt Mask Register 0		
Bit	**Label**	**Description**		**Default**
31..13	RSVD	Reserved for future use	RO	0
12	CDPEM	Control Data Protection Error Mask	RW	1
11	TXFCSEM	Transmit Frame Check Sequence Error Mask	RW	1
10	TTSCACM	Transmit Timestamp Capture Available C Mask	RW	1
9	TTSCABM	Transmit Timestamp Capture Available B Mask	RW	1
8	TTSCAAM	Transmit Timestamp Capture Available A Mask	RW	1
7	PHYINTM	Physical Layer Interrupt Mask	RW	1
6	RESETCM	RESET Complete Mask	RO	0
5	HDREM	Header Error Mask	RW	1
4	LOFEM	Loss of Framing Error Mask	RW	1
3	RXBOEM	Receive Buffer Overflow Error Mask	RW	1
2	TXBUEM	Transmit Buffer Underflow Error Mask	RW	1
1	TXBOEM	Transmit Buffer Overflow Error Mask	RW	1
0	TXPEM	Transmit Protocol Error Mask	RW	1

Table 18: IMASK0 – Interrupt Mask Register 0

9.2.11.1 RSVD
Reserved for future use.

9.2.11.2 CDPEM
Control Data Protection Error Mask. Setting this bit to ‘1’ prevents the Control Data Protection Error status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.3 TXFCSEM
Transmit Frame Check Sequence Error Mask. Setting this bit to ‘1’ prevents the Transmit Frame Check Sequence Error status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.4 TTSCACM
Transmit Timestamp Capture Available Mask C. Setting this bit to ‘1’ prevents the Transmit Timestamp Capture Available C status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.5 TTSCABM
Transmit Timestamp Capture Available Mask B. Setting this bit to ‘1’ prevents the Transmit Timestamp Capture Available B status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.6 TTSCAAM
Transmit Timestamp Capture Available Mask A. Setting this bit to ‘1’ prevents the Transmit Timestamp Capture Available A status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.7 PHYINTM
Physical Layer Interupt Mask. Setting this bit to ‘1’ prevents the physical layer interrupt (PHYINT) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.8 RESETCM

Reset Complete Mask. This bit is reserved as a mask for the Reset Complete (RESETC) status bit. This bit is read only and always zero as the RESETC status bit is a non-maskable interrupt that will cause IRQn to always assert when RESETC is set.

9.2.11.9 HDREM

Header Error Mask. Setting this bit to '1' prevents the Header Error (HDRE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.10 LOFEM

Loss of Framing Error Mask. Setting this bit to '1' prevents the Loss of Framing Error (LOFE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.11 RXBOEM

Receive Buffer Overflow Error Mask. Setting this bit to '1' prevents the Receive Buffer Overflow Error (RXBOE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.12 TXBUEM

Transmit Buffer Underflow Error Mask. Setting this bit to '1' prevents the Transmit Buffer Underflow Error (TXBUE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.13 TXBOEM

Transmit Buffer Overflow Error Mask. Setting this bit to '1' prevents the Transmit Buffer Overflow Error (TXBOE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.11.14 TXPEM

Transmit Protocol Error Mask. Setting a this bit to '1' prevents the Transmit Protocol Error (TXPE) status bit in STATUS0 from asserting the footer EXST bit.

9.2.12 Interrupt Mask Register #1 (0x000D)

IMASK1 Interrupt Mask Register 1

Bit	Label	Description		Default
31..0	VS	Vendor Specific	-	-

Table 19: IMASK1 – Interrupt Mask Register 1

9.2.12.1 VS

Vendor Specific mask bits. When set to a '1', the corresponding status in STATUS1 is prevented from asserting the footer EXST bit.

9.2.13 Transmit Timestamp Capture Register A - High (0x0010)

TTSCAH	Transmit Timestamp Capture Register A (High)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCH	Transmit Timestamp A (bits 63-32)	RO	-

Table 20: TTSCAH – Transmit Timestamp Capture Register A (High)

9.2.13.1 TTSCH

Transmit Timestamp Capture. This field contains the upper 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.14 Transmit Timestamp Capture Register A - Low (0x0011)

TTSCAL	Transmit Timestamp Capture Register A (Low)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCL	Transmit Timestamp A (bits 31-0)	RO	-

Table 21: TTSCAL – Transmit Timestamp Capture Register A (Low)

9.2.14.1 TTSCL

Transmit Timestamp Capture. This field contains the lower 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.15 Transmit Timestamp Capture Register B - High (0x0012)

TTSCBH	Transmit Timestamp Capture Register B (High)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCH	Transmit Timestamp B (bits 63-32)	RO	-

Table 22: TTSCBH – Transmit Timestamp Capture Register B (High)

9.2.15.1 TTSCH

Transmit Timestamp Capture. This field contains the upper 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.16 Transmit Timestamp Capture Register B - Low (0x0013)

TTSCBL	Transmit Timestamp Capture Register B (Low)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCL	Transmit Timestamp B (bits 31-0)	RO	-

Table 23: TTSCBL – Transmit Timestamp Capture Register B (Low)

9.2.16.1 TTSCL

Transmit Timestamp Capture. This field contains the lower 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.17 Transmit Timestamp Capture Register C - High (0x0014)

TTSCCH	Transmit Timestamp Capture Register C (High)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCH	Transmit Timestamp C (bits 63-32)	RO	-

Table 24: TTSCCH – Transmit Timestamp Capture Register C (High)

9.2.17.1 TTSCH

Transmit Timestamp Capture. This field contains the upper 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.18 Transmit Timestamp Capture Register C - Low (0x0015)

TTSCCL	Transmit Timestamp Capture Register C (Low)			
Bit	**Label**	**Description**		**Default**
31..0	TTSCL	Transmit Timestamp C (bits 31-0)	RO	-

Table 25: TTSCCL – Transmit Timestamp Capture Register C (Low)

9.2.18.1 TTSCL

Transmit Timestamp Capture. This field contains the lower 32 bits of the captured timestamp for when the requested frame was transmitted. See Figure 16 for timestamp format details.

9.2.19 MDIO Access Registers 0-7 (0x0020-0x0027)

MDIOACCn	MDIO Access Registers 0-7			
Bit	**Label**	**Description**		**Default**
31	TRDONE	Transaction Done	RW	1
30	TAERR	Turnaround Error	RO	0
29..28	ST	Start of Frame	RW	00
27..26	OP	Operation Code	RW	11
25..21	PRTAD	Port Address	RW	00000
20..16	DEVAD	Device Address	RW	00000
15..0	DATA	Data/Address Value	RW	0x00

Table 26: MDIOACCn – MDIO Access Register 0-7

9.2.19.1 TRDONE

Transaction Done. This bit should be written to '0' by the SPI host to initiate an MDIO transaction. The MACPHY will set this bit to '1' when the MDIO transaction has completed.

9.2.19.2 TAERR

Turnaround Error. The MACPHY will set this bit to '1' when a turnaround error occurs during the MDIO transaction. If this occurs the contents of the DATA field for Read and Read-post-increment-address operations should be ignored.

9.2.19.3 ST

Start of Frame. This field selects between Clause 45 and Clause 22 MDIO access, and should be set as follows:

00 Clause 45 MDIO frame
01 Clause 22 MDIO frame

9.2.19.4 OP

Operation Code. The following encoding is used:

00 Address (Clause 45 only)
01 Write
10 Post-read-increment-address (Clause 45 only)
11 Read

9.2.19.5 PRTAD

Port Address. This is the address of target port (PHY). This is called Port Address (PRTAD) for Clause 45, and called PHY Address (PHYAD) for Clause 22.

9.2.19.6 DEVAD

Device Address. For Clause 45 accesses, this is the address of the target Memory-Mapped Device (MMD) within the target port. For Clause 22 accesses, this is the address of the target register.

9.2.19.7 DATA

Data/Address Value. This should be treated according to the Operation code (OP) as follows:

- For a Write operation (Clause 45 or Clause 22) the SPI host should set this to the 16-bit value to be written.
- For an Address operation (Clause 45) the SPI host should set this to the 16-bit register address value.
- For a Read operation (Clause 45 or Clause 22), or for a Read-post-increment-address operation (Clause 45), the SPI host should set this value to '0'. On completion of the MDIO transaction (as indicated by TRDONE), the MACPHY sets this to the 16-bit value read.

9.3 Register Memory Map 1 - MAC Registers

Register memory map 1 (MMS = 1) is allocated for the implementation of MAC control and status registers. The register width and details are implementation dependent and beyond the scope of this specification.

9.4 Register Memory Map 2 – IEEE 802.3 PHY PCS Registers

Register memory map 2 (MMS = 2) is allocated for the implementation of IEEE 802.3 [1,2] PHY PCS registers (MMD 3). The registers within this memory are 16 bits in width.

Table 27: 802.3 MMD 3 PCS PHY Register mapping to MACPHY MMS 2

MMS	SPI Address		Register	MMD	Address	802.3 Format
2	0x0000	PCS	PCS control 1	3	0x0000	3.0
	⋮				⋮	⋮
	0x08E6		10BASE-T1L PCS control		0x08E6	3.2278
	0x08E7		10BASE-T1L PCS status		0x08E7	3.2279
	⋮				⋮	⋮
	0x08F3		10BASE-T1S PCS control		0x08F3	3.2291
	0x08F4		10BASE-T1S PCS status		0x08F4	3.2292
	0x08F5		10BASE-T1S PCS diagnostic 1		0x08F5	3.2293
	0x08F6		10BASE-T1S PCS diagnostic 2		0x08F6	3.2294
	⋮				⋮	⋮
	0x7FFF				0x7FFF	3.32767
	0x8000	PCS Vendor			0x8000	3.32768
	⋮				⋮	⋮
	0xFFFF				0xFFFF	3.65535

9.5 Register Memory Map 3 – IEEE 802.3 PHY PMA/PMD Registers

Register memory map 3 (MMS = 3) is allocated for the implementation of IEEE 802.3 [1,2] PHY PMA/PMD registers (MMD 1). The registers within this memory are 16 bits in width.

Table 28: 802.3 MMD 1 PMA/PMD PHY Register mapping to MACPHY MMS31

MMS	SPI Address		Register	MMD	Address	802.3 Format
3	0x0000	PMA/PMD	PMA/PMD control 1	1	0x0000	1.0
	⋮				⋮	⋮
	0x0008		PMA/PMD Status 2		0x0008	1.8
	⋮				⋮	⋮
	0x0012		BASE-T1 PMA/PMD extended ability		0x0012	1.18
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0x0834		BASE-T1 PMA/PMD control		0x0834	1.2100
	⋮				⋮	⋮
	⋮				⋮	⋮
	0x08F6		10BASE-T1L PMA control		0x08F6	1.2294
	0x08F7		10BASE-T1L PMA status		0x08F7	1.2295
	0x08F8		10BASE-T1L test mode control		0x08F8	1.2296
	0x08F9		10BASE-T1S PMA control		0x08F9	1.2297
	0x08FA		10BASE-T1S PMA status		0x08FA	1.2298
	0x08FB		10BASE-T1S test mode control		0x08FB	1.2299
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0x7FFF				0x7FFF	1.32767
	0x8000	PMA/PMD Vendor			0x8000	1.32768
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0xFFFF				0xFFFF	1.65535

9.6 Register Memory Map 4 – IEEE 802.3 PHY Vendor Specific and PLCA Registers

Register memory map 4 (MMS = 4) is allocated for the implementation of IEEE 802.3 [1,2] PHY Vendor Specific and PLCA registers [5] (MMD 31). The registers within this memory are 16 bits in width.

Table 29: 802.3 MMD 30/31 Vendor Specific PHY Register mapping to MACPHY MMS 1

MMS	SPI Address		Register	MMD	Address	802.3 Format
4	0x3000	Vendor Specific		31	0x0000	31.0
	⋮				⋮	⋮
	0xC9FF				0xC9FF	31.51711
	0xCA00	Open Alliance Specific	IDVER (PLCA ID Version)		0xCA00	31.51712
	0xCA01		CTRL0 (PLCA Control 0)		0xCA01	31.21713
	0xCA02		CTRL1 (PLCA Control 1)		0xCA02	31.51714
	0xCA03		STATUS (PLCA Status)		0xCA03	31.51715
	0xCA04		TOTMR (PLCA TO Control)		0xCA04	31.51716
	0xCA05		BURST (PLCA Burst Control)		0xCA05	31.51717
	⋮				⋮	
	0xFFFF				0xFFFF	31.33802

9.7 Register Memory Map 5 – IEEE 802.3 PHY Auto-Negotiation Registers

Register memory map 5 (MMS = 5) is allocated for the implementation of IEEE 802.3 [1,2] PHY Auto-Negotiation registers (MMD 7). The registers within this memory are 16 bits in width.

Table 30: 802.3 MMD 3 PCS Auto-Negotiation Register mapping to MACPHY MMS 5

MMS	SPI Address		Register	MMD	Address	802.3 Format
5	0x0000	Auto-Negotiation	AN Control	7	0x0000	7.0
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0x020E		10BASE-T1 AN Control		0x020E	7.526
	0x020F		10BASE-T1 AN Status		0x020F	7.527
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0x7FFF				0x7FFF	7.32767
	0x8000	Auto-Negotiation Vendor			0x8000	7.32768
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0xFFFF				0xFFFF	7.65535

9.8 Register Memory Map 6 – IEEE 802.3 PHY Power Unit Registers

Register memory map 6 (MMS = 6) is allocated for the implementation of IEEE 802.3 [1,2] PHY Power Unit registers (MMD 13). The registers within this memory are 16 bits in width.

Table 31: 802.3 MMD 13 PowerUnit PHY Register mapping to MACPHY MMS 6

MMS	SPI Address		Register	MMD	Address	802.3 Format
6	0x0000	Power Unit	PoDL PSE Control	13	0x0000	13.0
	0x0001		PoDL PSE Status 1		0x0001	13.1
	0x0002		PoDL PSE Status 2		0x0002	13.2
	0x0003		PoDL PSE Status 3		0x0003	13.3
	0x0004		PoDL PSE Status 4		0x0004	13.4
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	⋮				⋮	⋮
	0xFFFF				0xFFFF	13.65535