pCT Data Transfer Protocol (pDTP)

Ola S. Grøttvik

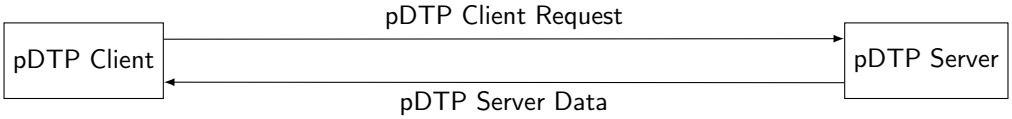
v1.1 - September 11, 2019

1 Concept

To reliably transmit data over UDP, it might be required to add a layer on top to control the offload from the pRU. This document presents a solution to the problem, the pCT Data Transfer Protocol (pDTP). The protocol enables the offload to operate in either pull, semi-push or full-push mode. The client can request the server (the pRU), to transmit a single packet of a desired size (PULL). Or the client can request a stream of certain number of packets of a desired size (SEMI-PUSH). Full-push mode, is a mode were there are no control of the stream, the server will continuously push available data to the client (FULL-PUSH).

In PULL mode, the client can either instruct the server to wait for an acknowledge or not, and also tell the server to retransmit the last packet. In SEMI-PUSH mode no acknowledges are performed before the last packet - the end-of-stream (EOS) packet. In SEMI-PUSH mode there are no re-transmit possibilities and if packets are lost, they are lost forever. In either mode, the server will always try to transmit the number of pRU words requested, but if no more data is available will only transmit what is possible. For PULL, if no data at all is available, a special error packet is sent. For SEMI-PUSH, an end-of-stream (EOS) will be transmitted. For FULL-PUSH, nothing is transmitted if no data is available, and the server will stay in the mode until it is aborted.

The maximum theoretical pDTP payload is calculated by $65,535(2^{16}-1)-20(IPv4)-8(UDP)-8(pDTPSH) = 65,499$ octets which translates to no more than 4093 pRU words. The calculation would be the same if pDTP Server Header were to be extended to 16 octets, so there are possibilitites to add more fields for clarity. However, because of diminishing returns in efficiency gain, there seem to be a practical limit of 9000 bytes in the payload. This yields an efficiency of 99% and is the maximum size of a so-called jumbo frame. Also, to make the pDTP header format as clean as possible, a limit of 255 pRU words (4080 bytes) seems to be a sound limit.



Two header formats are defined, one for the client and one for server. This is done as the requirements for the two are quite different; the client are just sending commands and the server are transmitting data as well as acknowledgements.

2 pDTP Client Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	IPv4 (20 Octets?)																															
20	160	UDP (8 Octets)																															
28	224	DTP Client Opcode				Flags				DTP Special Commands																Requested DTP Packet Size - in pRU words							

2.1 pDTP Client Opcodes

Opcode Name	Opcode	Short Description	Long Description
CLIENT_RQR	0x0	Client Read Request	Client request the server to transmit a packet of a certain amount of pRU words
CLIENT_RQT	0x1	Client Test Request	Client request the server to transmit a test packet of a certain amount of pRU words
CLIENT_RQS	0x2	Client Stream Request	Client request the server to transmit a certain number of packets of a certain amount of pRU words
CLIENT_RQFS	0x3	Client Full-Stream Request	Client requests the server to continuously transmit packets as data gets available
CLIENT_ERROR	0x4	Client Error	Timeout while waiting for packet, or some other error
CLIENT_ACK	0x5	Client Acknowledge	Acknowledge to the server that the last packet were obtained
CLIENT_ABRT	0x6	Client Abort	Instructs the server to abort the current operation, e.g. CLIENT_RQFS
CLIENT_GS	0x7	Client Get Status	Ask the server to transmits its buffer status, firmware information, and absolute clock
CLIENT_THROTTLE	0x8	Client Throttle	Ask the server to throttle the output stream, by waiting for a certain number of clk cycles between each stream packet

2.2 pDTP Client Special Commands

Opcode / Bit	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CLIENT_RQR	NO_ACK	MIN_RQ	MAXIMIZE																		RQ_PACKET_SIZE							
CLIENT_RQT	NO_ACK																				RQ_PACKET_SIZE							
CLIENT_RQS		MIN_RQ	MAXIMIZE	NO_WAIT	RQ_STREAM_SIZE																RQ_PACKET_SIZE							
CLIENT_RQSFS		MIN_RQ	MAXIMIZE	NO_WAIT																	RQ_PACKET_SIZE							
CLIENT_ERROR		UNINTERPRETABLE	TIMEOUT	RESEND_PACKET																								
CLIENT_ACK																												
CLIENT_ABRT																												
CLIENT_GS																												
CIENT_THROTTLE				WAIT_CYCLES																								

- NO_ACK - Server will not wait for an acknowledge from client
- RQ_PACKET_SIZE - The number of pRU words requested in each packet - max 255 pRU words, i.e. ~ 4 kB
- RQ_STREAM_SIZE - Number of packets to transmit without listening for acknowledge - max $2^{16}-1 = 65.535$ packets
- MIN_RQ - Will not send a packet if buffer has less than RQ_PACKET_SIZE. Pull: Server will send an ERROR. Semi push: Server will transmit an EOS. Full push: Server will wait until buffer is filled with required amount.
- MAXIMIZXE - Maximize the number of pRU words transmitted, based on the data available. Can be used together with MIN_RQ.
- NO_WAIT - Server will not wait for buffer to be filled with data if buffer is empty or is filled with less than MIN_RQ. Result is EOS in both RQS and RQFS.
- UNINTERPRETABLE - The received packet was not possible to understand
- TIMEOUT - The client timed out while waiting for something
- RESEND_PACKET - Ask the server to resend the last packet - only available after CLIENT_RQR
- WAIT_CYCLES - The number of clock cycles between each stream packet transmitted from the server.

3 pDTP Server Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	IPv4 (20 Octets?)																															
20	160	UDP (8 Octets)																															
28	224	DTP Server Opcode				FLAGS				DTP Packet ID / Buffer Fill Count																Actual DTP Packet Size / Version Number							
32	256	ABS_TIME (System Clock Cycles)																															
36	288	Payload (0 - 255 pRU words)																															

3.1 pDTP Server Opcodes

Opcode Name	Opcode	Short Description	Long Description
SERVER_WRITE	0x0	Server Write Packet	Server sends a packet
SERVER_STREAM	0x1	Server Stream Packet	A part of a stream of packets
SERVER_ERROR	0x2	Server Error	Timeout while waiting for ack, uninterpretable received packet or no data available
SERVER_EOS	0x3	Server End-Of-Stream	Server is finished transmitting all possible packets
SERVER_STATUS	0x4	Server Status	Server Status

3.2 pDTP Special Commands

Opcode / Bit	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SERVER_WRITE	FULL	ALMOST_FULL			PACKET_ID																AC_PACKET_SIZE							
SERVER_STREAM	FULL	ALMOST_FULL			PACKET_ID																AC_PACKET_SIZE							
SERVER_ERROR	INVALID_RQ	MIN_RQ	EMPTY	TIMEOUT	BUFFER_FILL_COUNT																							
SERVER_EOS		MIN_RQ			BUFFER_FILL_COUNT																							
SERVER_STATUS	FULL	ALMOST_FULL	EMPTY		BUFFER_FILL_COUNT																VERSION_NUMBER							

- PACKET_ID - Incremented counter of all transmitted packets
- FULL - The offload data buffer is full
- ALMOST_FULL - The offload data buffer has less than 20% space left
- EMPTY - The offload data buffer is empty
- TIMEOUT - Timeout while waiting for something or uninterpretable
- NO_DATA - No data was available on server when receiving request
- BUFFER_FILL_COUNT - Amount of pRU words stored in offload buffer
- INVALID_RQ - Client asked about something unknown to the server
- VERSION_NUMBER - Current version of pDTP protocol

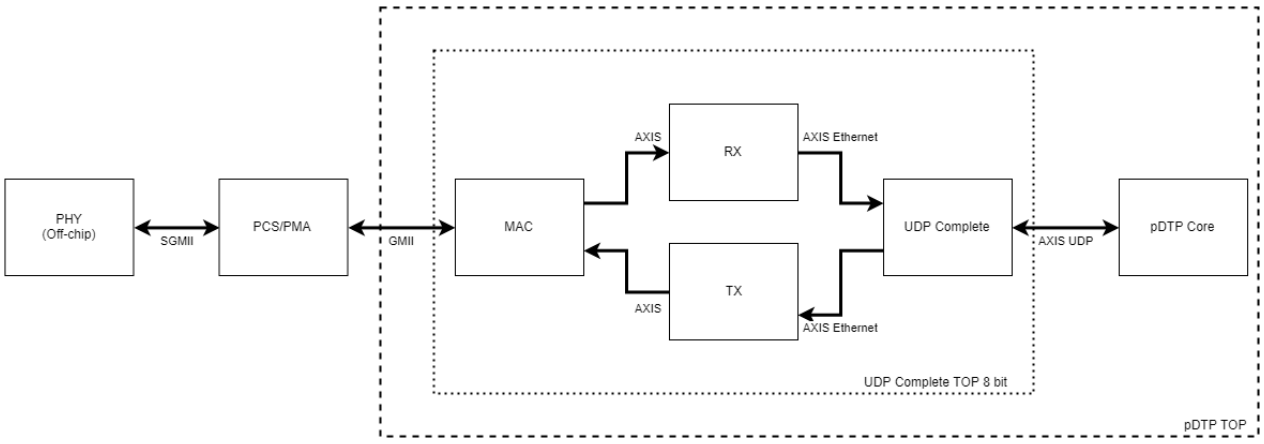
3.3 SERVER_STATUS

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	IPv4 (20 Octets)																															
20	160	UDP (8 Octets)																															
28	224	SERVER_STATUS				FULL		ALMOST_FULL		EMPTY								BUFFER_FILL_COUNT								VERSION_NUMBER							
32	256	ABS_TIME (System Clock Cycles)																															
36	288	BUILD_DATE (YYMMDDHH)																															
40	320	GIT_HASH																															

4 Firmware

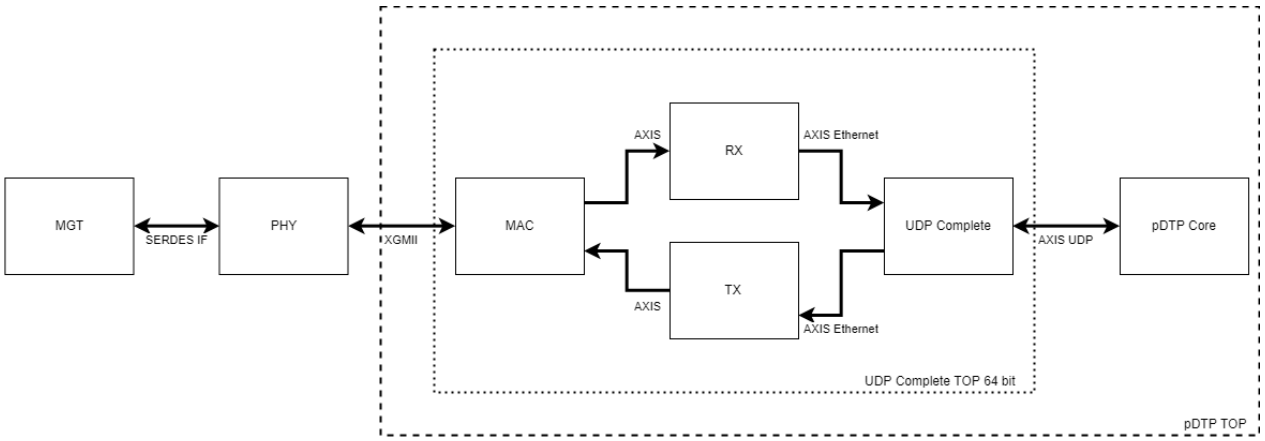
4.1 1 Gb/s - SGMII - Standard Ethernet

This approach are using a standard Ethernet cable that are connected to an off-chip SGMII PHY chip until reaching the FPGA. A PCS/PMA Xilinx IP are converting SGMII to GMII before reaching the open source MAC and the rest of the logic. All stream are 8-bit per clock cycle. Clocks are 125 MHz, as 125M*8 = 1G.



4.2 10 Gb/s - XGMII - SFP+/QSFP+

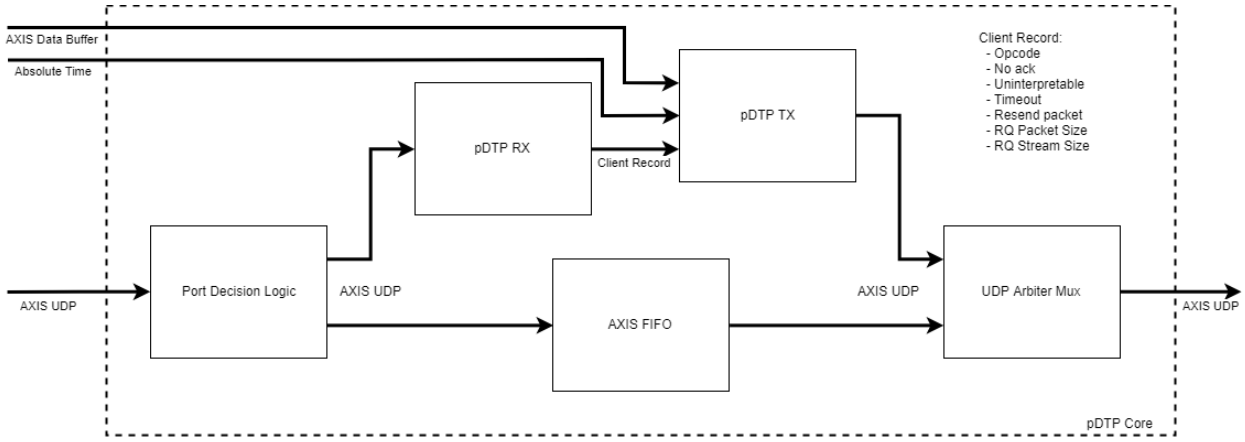
This approach are using a SFP+/QSFP+ cables, i.e. each QSFP+ connector holds 4x10Gb/s links. The links are connected to a Multigigabit Transceiver(MGT) on the FPGA, which are then connected to an open source PHY and the rest of the logic. All streams are 64-bit per clock cycle. Clocks are 156.25 MHz, as 156.25*64 = 10G.



5 pDTP Core

The pDTP Core module has the possibility of looping back UDP packets. This decision is done by the Port Decision Logic. A loopback packet will be transmitted via the AXIS FIFO into the arbiter mux.

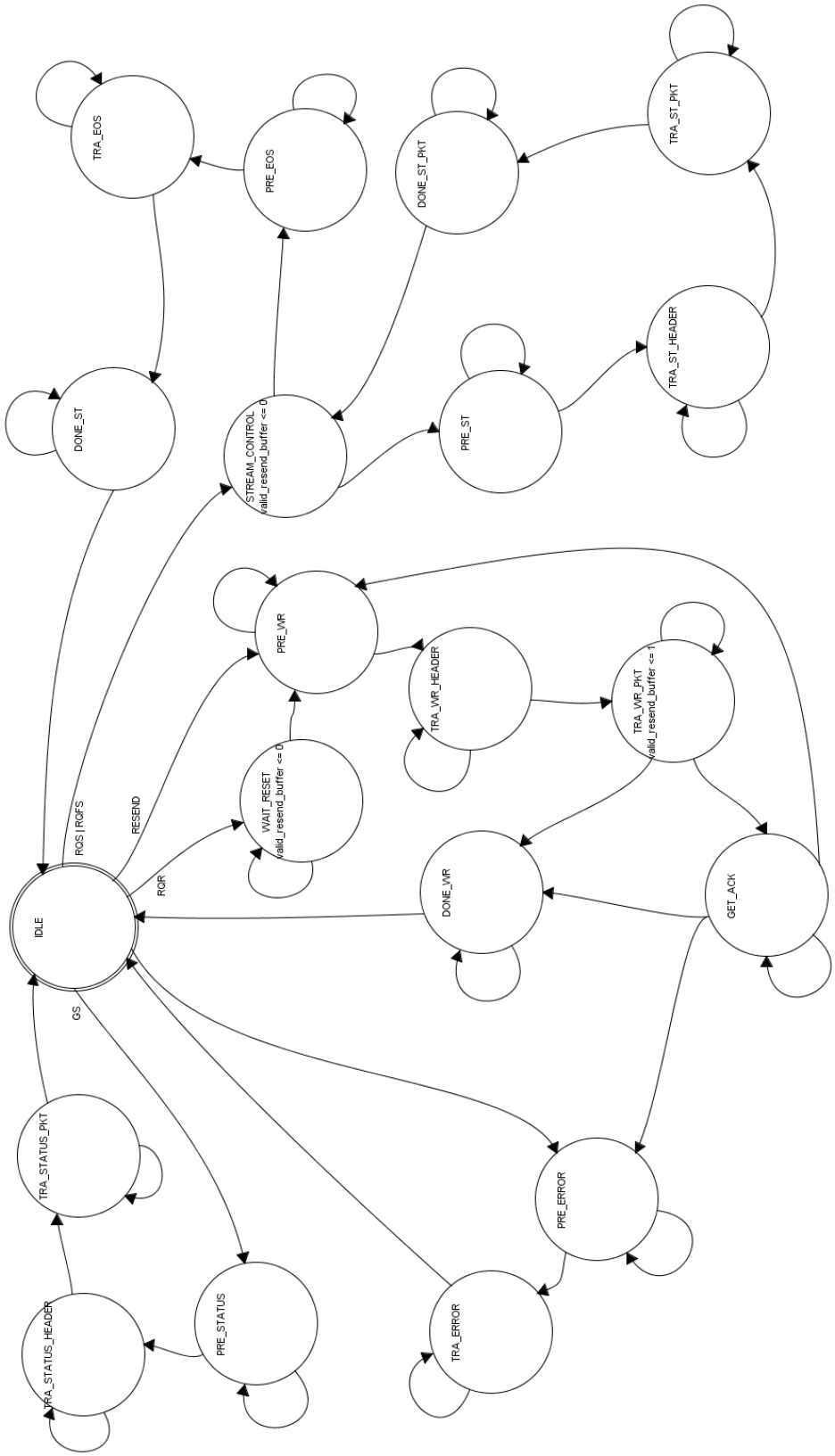
pDTP Client Packets will be interpreted by pDTP RX module. This module will then transmit a client record to the pDTP TX module, which will transmit pDTP Server Packets based on the client orders.



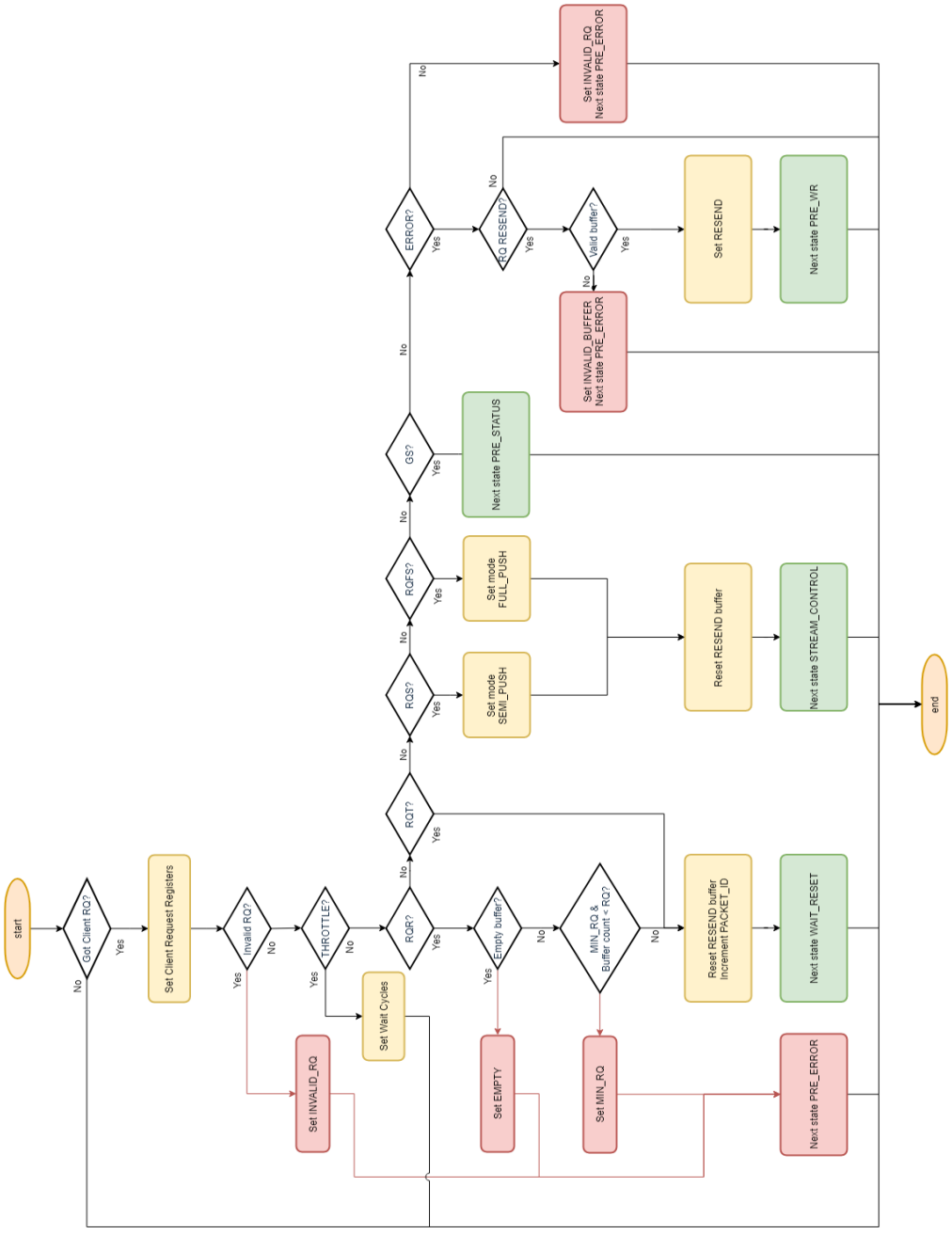
Ports:

- 29070: Loopback Port
- 30000: pDTP Port

6 TX FSM and Flow

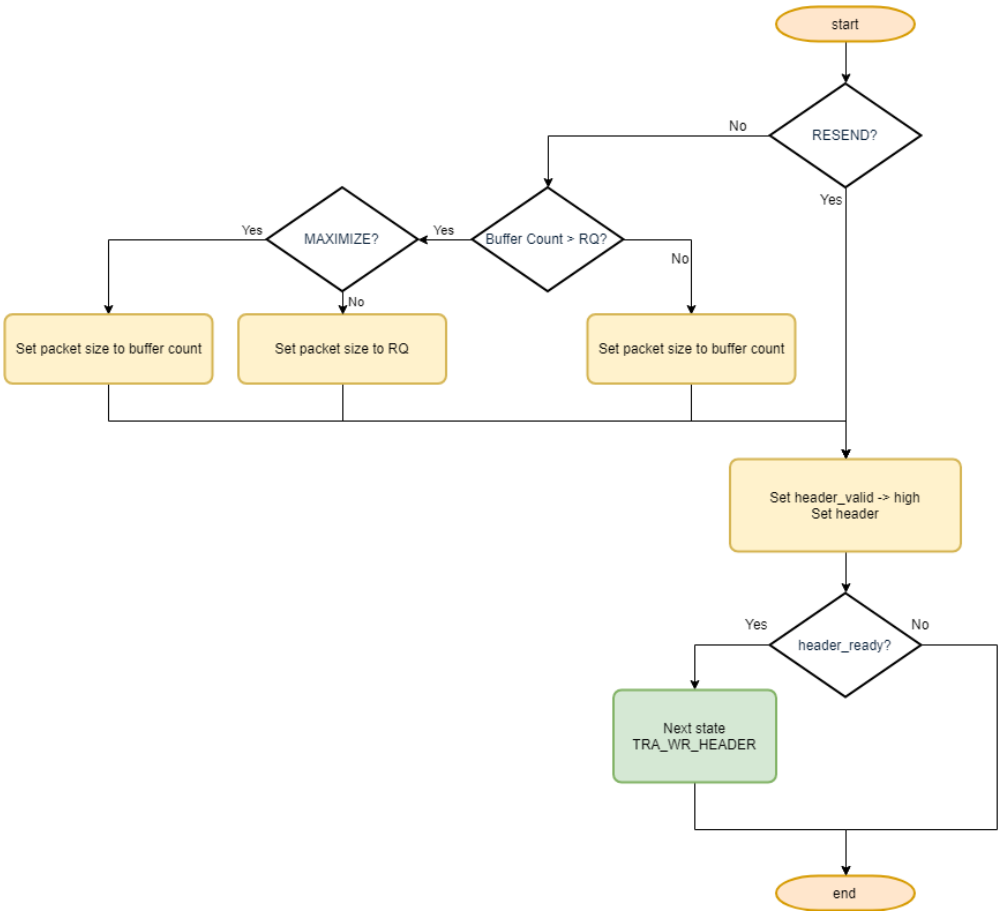


6.1 IDLE Flow


start
Got Client RQ?
Yes
No
Set Client Request Registers
Invalid RQ?
Yes
No
Set INVALID_RQ
THROTTLE?
Yes
No
Set Wait Cycles
RQR?
Yes
No
Empty buffer?
Yes
No
Set EMPTY
MIN_RQ & Buffer count < RQ?
Yes
No
Set MIN_RQ
Reset RESEND buffer Increment PACKET_ID
Next state WAIT_RESET
Next state PRE_ERROR
RQT?
Yes
No
RQS?
Yes
No
Set mode SEMI_PUSH
RQFS?
Yes
No
Set mode FULL_PUSH
Reset RESEND buffer
Next state STREAM_CONTROL
GS?
Yes
No
Next state PRE_STATUS
ERROR?
Yes
No
RQ RESEND?
Yes
No
Valid buffer?
Yes
No
Set INVALID_BUFFER Next state PRE_ERROR
Set RESEND
Next state PRE_WR
Set INVALID_RQ Next state PRE_ERROR
end

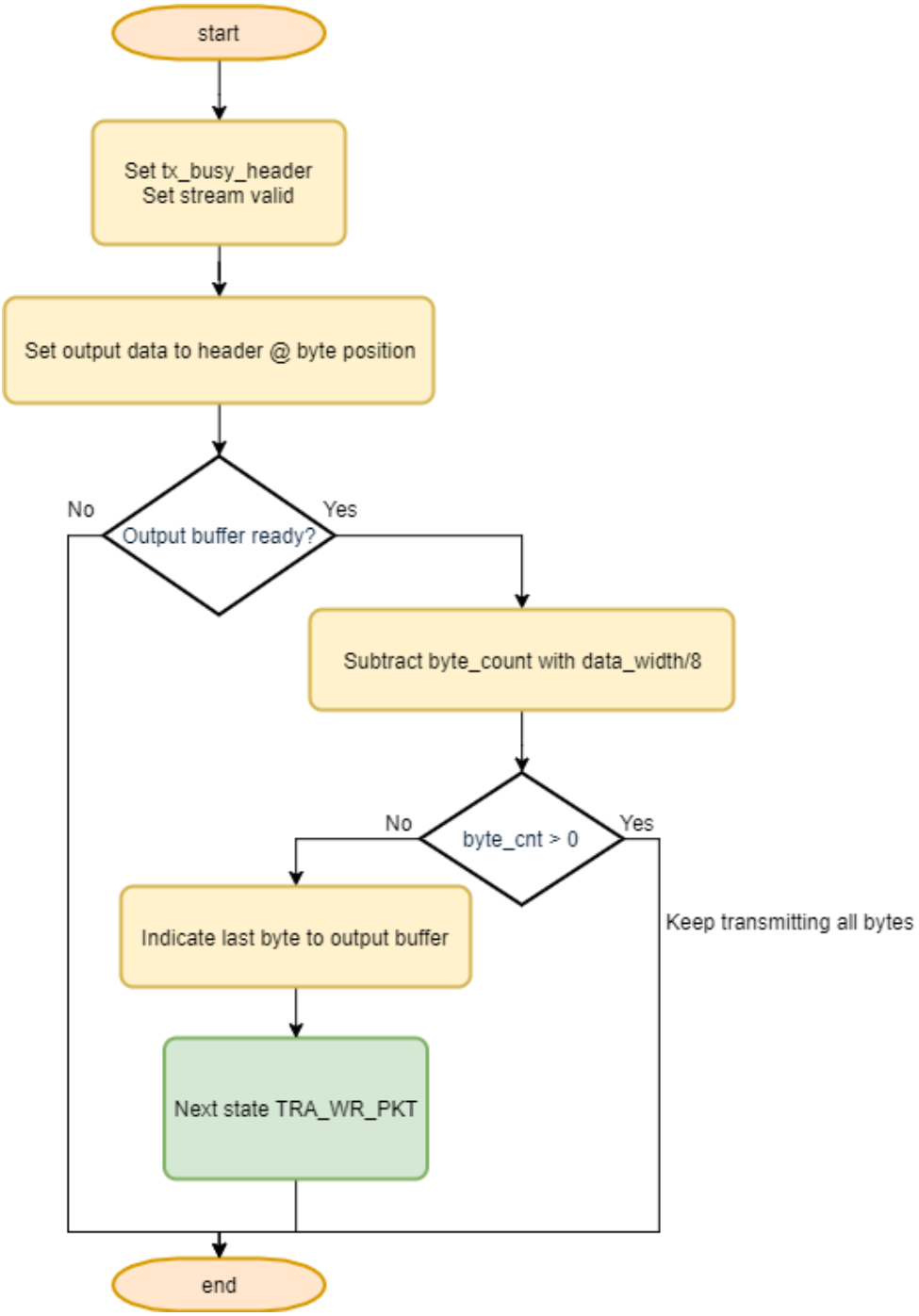
6.2 PRE_WR Flow

PRE_WR is similar to the following states: PRE_ST, PRE_ERROR, PRE_EOS, PRE_STATUS. However, these other states does not have a check for resend functionality. Also, next state can be determined from FSM graph.



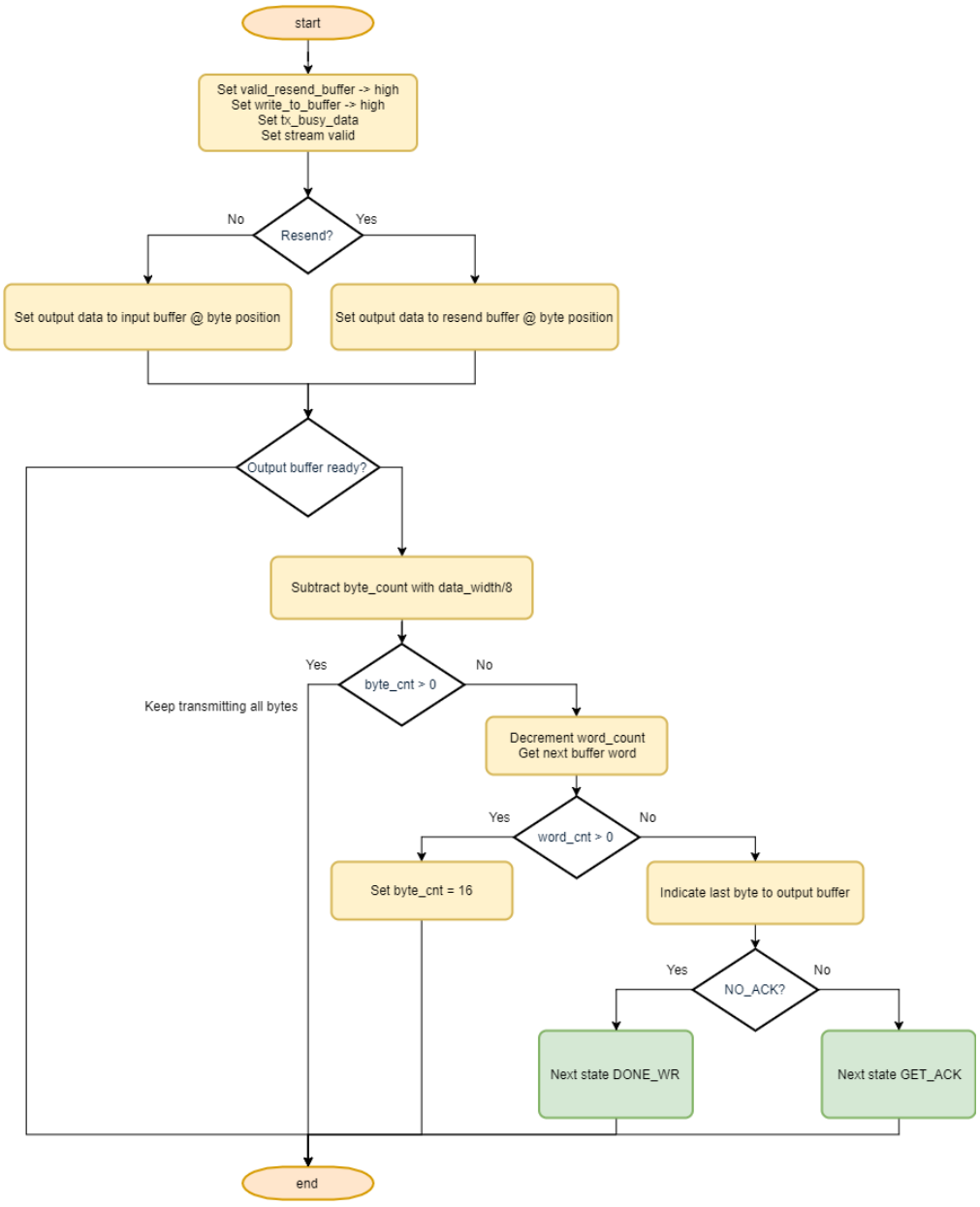
6.3 TRA_WR_HEADER Flow

TRA_WR_HEADER is similar to the following states: TRA_ST_HEADER, TRA_ERROR, TRA_EOS, TRA_STATUS_HEADER. Next state can be determined from FSM graph.



6.4 TRA_WR_PKT Flow

TRA_WR_PKT is similar to the following states: TRA_ST_PKT, TRA_STATUS_PKT. These states are missing the resend functionality. Next state can be determined from FSM graph.



6.5 Flow STREAM_CONTROL