DS_BASIC IP SPEC

Table of Contents

Introduction 2
Feature（silergy internal） 2
Register Definition 2
Register Map 2
Functional Details 3
Block Diagram 3
Module input/output list 3
DS_BASIC function description 6

Introduction

The DS_BASIC module is used to convert input dual daisy chain signals to byte data rx_data[8:0], and convert byte data tx_data[8:0] to dual daisy chain outputs.

Feature

Key features of the DS_BASIC module are:

- Analysis received daisy chain data
- Send daisy chain data
- Direction control

Register Definition

Register Map

Table 1 DS_1BASIC Register Map

Name	Add	D7	D6	D5	D4	D3	D2	D1	D0	Default
COMM_CONF2	0x0003	COMN_TX_DIS	COMS_TX_DIS	STACK_RESPONSE<5:0>						00
CTRL2	0x2003						CMP_BIST_GO	ADD_W_EN	SPI_DIR	

Functional Details

Block Diagram

The following diagram shows the DS_BASIC architecture and internal modules and connections.

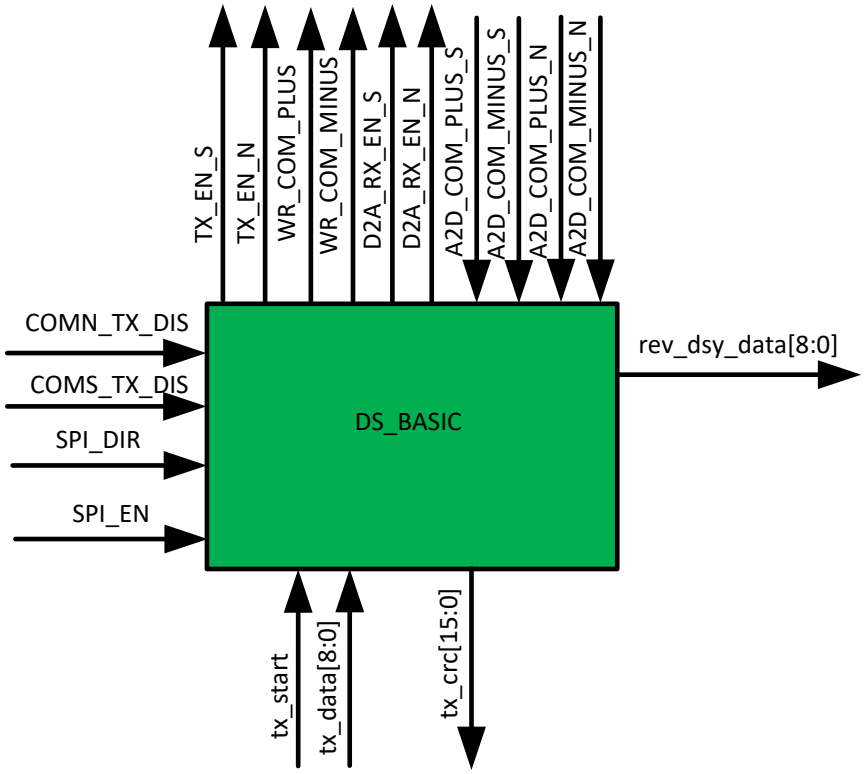


Figure1 DS_BASIC diagram

Module input/output list

Name	Dir	Width	Discirption	duration
rev_dsy_data	O	9	received daisy chain data	Level(32M domain)
neg_rx_en_dsy	O	1	negedge of rx_en_dsy	4 CLK_32M
neg_rx_en_dsy_8M	O	1	negedge of rx_en_s_dsy or rx_en_n_dsy	1 CLK_REG
neg_rx_en_s_dsy	O	1	negedge of rx_en_s_dsy	4 CLK_32M
neg_rx_en_n_dsy	O	1	negedge of rx_en_n_dsy	4 CLK_32M
MISS	O	1		
ORDER	O	1		
SYNCT	O	1		
SYNCD	O	1		
BIT	O	1		
WR_COM_PLUS	O	1	daisy chain output data in positive phase	8 CLK_32M
WR_COM_MINUS	O	1	daisy chain output data in negtive phase	8 CLK_32M

TX_EN_S	O	1	enable daisy chain transmitting on S port	
TX_EN_N	O	1	enable daisy chain transmitting on N port	
D2A_RX_EN_S	O	1	enable daisy chain receiving on S port	
D2A_RX_EN_N	O	1	enable daisy chain receiving on N port	
send_char_end_pos	O	1	mark byte transmitting end time	4 CLK_32M
tx_crc	O	16	crc16 result of tx_one	
rx_en_n	O	1	daisy chai signal is being received on N port	
rx_en_s	O	1	daisy chai signal is being received on S port	
rx_en	O	1	daisy chai signal is being received	
tx_en_32M	O	1	sync send_start with CLK_32M	2 CLK_32M
neg_TX_EN_S	O	1	negedge of TX_EN_S	1 CLK_32M
neg_TX_EN_N	O	1	negedge of TX_EN_N	1 CLK_32M
clr_crc_dsy	O	1	crc clear	3~4 CLK_32M
TX_timeout	O	1	no data to tranmit for a timeout time when TX_EN_X high	
pos_TBYTE_FAST	O	1	fault flag: receiving data is too fast	4 CLK_32M
pos_TBYTE_TO	O	1	fault flag: receiving data is too slow	4 CLK_32M
CLK_32M_SC	I	1	CLK_32M after scan mux	
resetb_CLK	I	1	Asynchronous reset signal(synchronously released)	
rstb_32M_ok_and_sr	I	1	CLK_32M_OK low or soft reset	
SOFT_RSTB_32M	I	1	Soft reset	
rst_spi	I	1	When SPI_EN, reset spi	4 CLK_32M
CLK_REG_SC	I	1	Scan-mux result of 8MHz clock from CLK_32M	8MHz
SLEEP_MODE	I	1	Synchronous result of A2D_SLEEP_1P8 by CLK_256K_SC	Level
A2D_COM_PLUS_S	I	1	Positive input comparator in S port	async
A2D_COM_MINUS_S	I	1	Negative input comparator in S port	async
A2D_COM_PLUS_N	I	1	Positive input comparator in N port	async
A2D_COM_MINUS_N	I	1	Negative input comparator in N port	async
dev_addr_dlv	I	1	Device address identify delivery	Level(8M domain)
dev_addr_dlv_spi	I	1	Device address identify delivery when SPI_EN high	Level(8M domain)
tx_data	I	9	Data to be transmit	CLK_REG domain
state_tx_init	I	1	tx_state is STATE_INIT	1 CLK_REG
state_tx_pec	I	1	tx_state is STATE_PEC	1 CLK_REG
state_rx_init	I	1	state is STATE_INIT	1 CLK_REG
state_rx_bps	I	1	state is STATE_BYPASS	1 CLK_REG
response	I	1	Current device response	Level(8M domain)
pos_response	I	1	Positive edge of response	1 CLK_REG
neg_response	I	1	Negative edge of response	1 CLK_REG
pos_next_rps	I	1	Current device is the next to response	1 CLK_32M

bypass_end	I	1	Mark the ending time of a bypass byte	1 CLK_REG
rx_dev_addr	I	1	Receive 9'h1C0 when state is STATE_INT or STATE_BYPASS	4 CLK_32M
cnt_rx_byte_num	I	8	Rx byte numer	Level(8M domain)
rd	I	1	Current device in read station	Level(8M domain)
D2A_TOP_DEV	I	1	Current device is fastest from bridge	Level(8M domain)
stack	I	1	Stack operation	Level(8M domain)
COMN_TX_DIS	I	1	N port transmit disable	Level(8M domain)
COMS_TX_DIS	I	1	S port transmit disable	Level(8M domain)
wait_re_clocking	I	14	Wait time before transmitting	CLK_REG domain
tx_start	I	1	Transmitting start	1 CLK_REG
adr_idty_done	I	1	Address identify done	Level(8M domain)
tail_blanking	I	1	Tail blanking time	Level(8M domain)
SPI_EN	I	1	SPI enable	async
SPI_DIR	I	1	Direction configured by i2c_master	Level(8M domain)
SPI_RX_EN	I	1	A byte is received by SPI interface	4 CLK_32M
clr_crc_spi	I	1	At SPI_CSB negedge, clr_crc_spi generate one pulse to set CRC result to default FFFF when SPI_EN high.	4 CLK_32M
RESP	I	1	Response by bridge device	Level(8M domain)
spi_rx_pro	I	1	Spi rx process	4 CLK_32M
neg_rx_en	I	1	Negedge of rx_en	1 CLK_REG
next_rps	I	1	Current device is the next to response	Level(8M domain)
rx_data	I	9	Received data	Level(32M domain)
TONE_TRANS_EN_N	I	1	N port tone transmission enable	Level(CLK_OUT domain)
TONE_TRANS_EN_S	I	1	S port tone transmission enable	Level(CLK_OUT domain)
rx_en_256K	I	1	Daisy chain or spi rx_en	Level(CLK_256K domain)
neg_tx_init	I	1	Pulse after tx_state jump to STATE_INIT from STATE_PEC	1 CLK_REG
STACK_RESPONSE	I	6	Internal time between response bytes	Level(8M domain)
FRAME_DONE	I	9	Received frame done	1 CLK_REG
FR_CRC_FLT	I	1	Frame CRC fault	1 CLK_REG
SCAN_MODE	I	1	Scan mode	level

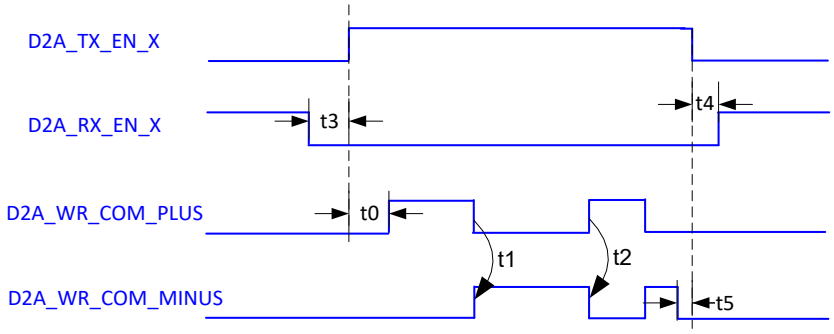
Clock Domain

The clock for DS_BASIC is CLK_32M_SC.

DS_BASIC function description

1 D2A_RX_EN_X and TX_EN_X

Requirements for D2A_RX_EN_X and TX_EN_X by analog part:



① t3>400ns, t4>400ns
② t0>10ns
③ |t1|<=2ns, |t2|<=2ns
④ t5>=0

Figure2 D2A_RX_EN_X and TX_EN_X timing requirement by analog part

1.1 D2A_RX_EN_x:

D2A_RX_EN_S and D2A_RX_EN_N are both initial high(HWR005_DS_BASIC). When D2A_RX_EN_S high, RX in S port is enabled. When D2A_RX_EN_N is high, RX in N port is enabled.

Besides daisy chain receive control, D2A_RX_EN_x also enable SPI receiving. When SPI_EN high, BM20A is used as bridge. If SPI_DIR is high, SPI replace the N port, so SPI_MOSI can be received when D2A_RX_EN_N high; If SPI_DIR is low, SPI replace the S port, so SPI_MOSI can be received when D2A_RX_EN_S high. This feature is delivered by COPY_NXT, which is an output from COMM_CTRL, and is described in IP_SPEC_COMM_CTRL.docx in detail. (HWR006_DS_BASIC, HWR010_DS_BASIC)

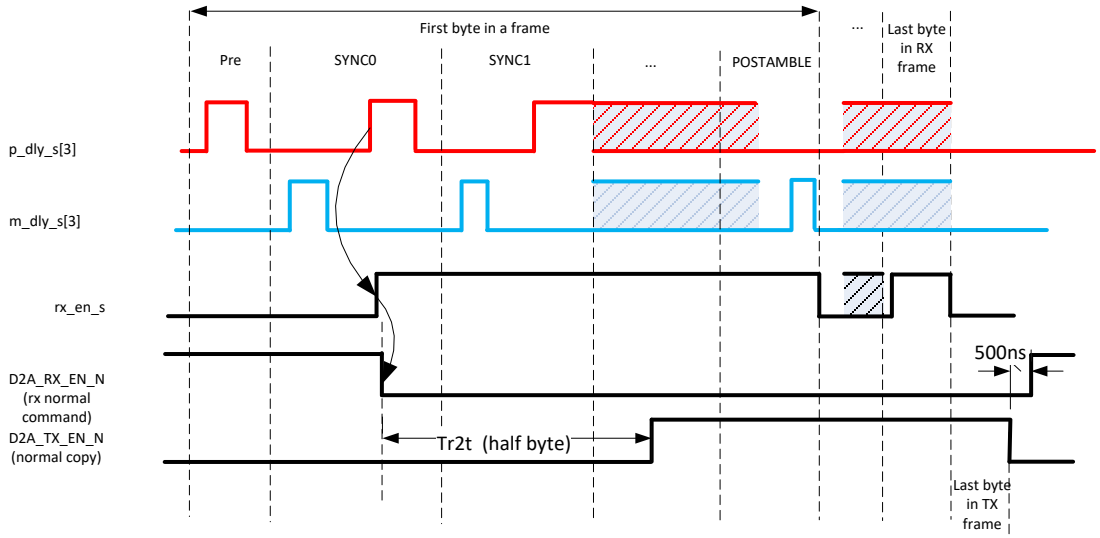


Figure3 D2A_RX_EN_X and D2A_TX_EN_X in normal copy case

1.1.1 D2A_RX_EN_N:

When SPI_EN high and SPI_CLR_DET high, or when SOFT_RSTB low, D2A_RX_EN_N reset to 1.

When no more data is being transmitting, TX_timeout is generated, D2A_RX_EN_N reset to 1.

When data comes from S port(A2D_COM_PLUS_S and A2D_COM_MINUS_S), D2A_RX_EN_N shall change to 0(HWR007_DS_BASIC). When bit receiving timeout, D2A_RX_EN_N shall change to 1.

When SPI replacing S port starts to receive data, which means ① SPI replace S port (SPI_EN and !SPI_DIR), ② SPI start to receive(clr_crc_spi) and ③ not in response procedure(RESP low), D2A_RX_EN_N shall change to 0.

When BM20A is responsing data in S port, which means ①posedge of response ②D2A_TOP_DEV or response to single read command(!stack), and ③ TX_EN_S high, D2A_RX_EN_N shall change to 0. (HWR005_DS_BASIC)

When 500ns after transmitting ends(T2R high), D2A_RX_EN_N changes to !tx_n. (tx_n means response on N port).

When received bit length overtime, D2A_RX_EN_N reset to 1.

When sending TONE via N port(TONE_TRANS_EN_N_32M_sync[1] high), D2A_RX_EN_N change to 0(HWR005_DS_BASIC); When sending TONE via N port ends((|TONE_TRANS_EN_N_32M_sync[4:0]) high and TONE_TRANS_EN_N_32M_sync[1] low), D2A_RX_EN_N change to !TONE_TRANS_EN_N_32M_sync[3] & (state_rx_init | state_rx_bps).

1.1.2 D2A_RX_EN_S:

When SPI_EN high and SPI_CLR_DET high, or when SOFT_RSTB low, D2A_RX_EN_S reset to 1.

When no more data is being transmitting, TX_timeout is generated, D2A_RX_EN_S reset to 1.

When data comes from N port(A2D_COM_PLUS_N and A2D_COM_MINUS_N), D2A_RX_EN_S shall change to 0(HWR007_DS_BASIC). When bit receiving timeout, D2A_RX_EN_S shall change to 1.

When SPI replacing N port starts to receive data, which means ①SPI replace N port (SPI_EN and SPI_DIR), ②SPI start to receive(clr_crc_spi, high pulse when SPI_CSB negedge) and ③not in response procedure(RESP low), D2A_RX_EN_S shall change to 0.

When responsing data in N port, which means ①posedge of response ②D2A_TOP_DEV or response to single read command(!stack), and ③ TX_EN_N high, D2A_RX_EN_S shall change to 0. (HWR005_DS_BASIC)

When 500us after transmitting ends (T2R high), D2A_RX_EN_S changes to !tx_s. (tx_s means response on S port).

When received bit length overtime, D2A_RX_EN_S reset to 1.

When sending TONE via S port (TONE_TRANS_EN_S_32M_sync[1] high), D2A_RX_EN_S change to 0(HWR005_DS_BASIC); When sending TONE via S port ends ((|TONE_TRANS_EN_S_32M_sync[4:0]) high and TONE_TRANS_EN_S_32M_sync[1] low), D2A_RX_EN_S change to !TONE_TRANS_EN_S_32M_sync[3] & (state_rx_init | state_rx_bps).

1.2 TX_EN_x:

TX_EN_S and TX_EN_N are both initial low. (HWR005_DS_BASIC)

1.2.1 TX_EN_N:

When SPI replacing N port(SPI_EN & SPI_DIR) and when response to spi_master(RESP) and receiving data from last device(D2A_RX_EN_S), D2A_TX_EN_N is 0. (HWR001_BASIC_CTRL)

When SPI_EN high and SPI_CLR_DET high, or when SOFT_RSTB low, TX_EN_N reset to 0.

When no more data is being transmitting, TX_timeout is generated, TX_EN_N reset to 0.

When receiving data from N port and cnt_bit>=5, TX_EN_N is 0.

For coping to opposite port(HWR002_DS_BASIC):

When data comes from S port (A2D_COM_PLUS_S and A2D_COM_MINUS_S), after T_{r2t}, TX_EN_N shall change to 1(can be disabled by COMN_TX_DIS) (HWR009_DS_BASIC). TX_EN_N shall change to 0 when coping ends. T_{r2t} is half byte time in normal copies; T_{r2t} is 75us after 1st address identify frame byte is received.

When bypass_end, TX_EN_N is 0.

For response:

If Address Identify Command comes from N port, after the frame has been copied to opposite S port, Response shall be in "1st device, 2nd device … top device" order. TX_EN_N shall be high(can be disabled by COMN_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_N shall change to 0 when response ends.

If Stack Read Command comes from N port, after the frame has been copied to opposite S port, Response shall be in "top device, (top-1)st device … 1st device" order. All response frames by other devices shall be copied via N port. After coping, TX_EN_N shall be high(can be disabled by COMN_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_N shall change to 0 when response ends.

If Single Read Command comes from N port, after the frame has been copied to opposite S port, if current device is chosen to response, after coping, TX_EN_N shall be high(can be disabled by COMN_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_N shall change to 0 when response ends. If current device is not chosen to response, TX_EN_N follow copy logic.

For Address identify command, $T_{response}$ shall be typically 24us. For other commands, $T_{response}$ shall be typically 30us.

1.2.2 TX_EN_S :

When SPI replacing S port(SPI_EN & !SPI_DIR) and when response to spi_master(RESP) and receiving data from last device(D2A_RX_EN_S), D2A_TX_EN_S is 0. (HWR001_BASIC_CTRL)

When SPI_EN high and SPI_CLR_DET high, or when SOFT_RSTB low, TX_EN_S reset to 0.

When no more data is being transmitting, TX_timeout is generated, TX_EN_S reset to 0.

When receiving data from S port and cnt_bit>=5, TX_EN_S is 0.

For coping to opposite port(HWR002_DS_BASIC):

When data comes from N port (A2D_COM_PLUS_N and A2D_COM_MINUS_N), after T_{r2t}, TX_EN_S shall change to 1(can be disabled by COMS_TX_DIS) (HWR009_DS_BASIC). TX_EN_S shall change to 0

when coping ends. T_{r2t} is half byte time in normal copies; T_{r2t} is 75us after 1st address identify frame byte is received.

When bypass_end, TX_EN_S is 0.

If Address identify Command comes from S port, after the frame has been copied to opposite N port, Response shall be in "1st device, 2nd device … top device" order. TX_EN_S shall be high(can be disabled by COMS_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_S shall change to 0 when response ends.

If Stack Read Command comes from S port, after the frame has been copied to opposite N port, Response shall be in "top device, (top-1)st device … 1st device" order. All response frames by other devices shall be copied via S port. After coping, TX_EN_S shall be high(can be disabled by COMS_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_S shall change to 0 when response ends.

If Single Read Command comes from S port, after the frame has been copied to opposite N port, if current device is chosen to response, after coping, TX_EN_S shall be high(can be disabled by COMS_TX_DIS) after $T_{reponse}$ (HWR009_DS_BASIC). TX_EN_S shall change to 0 when response ends. If current device is not chosen to response, TX_EN_S follow copy logic.

For Address identify command, $T_{response}$ shall be typically 24us. For other commands, $T_{response}$ shall be typically 30us.

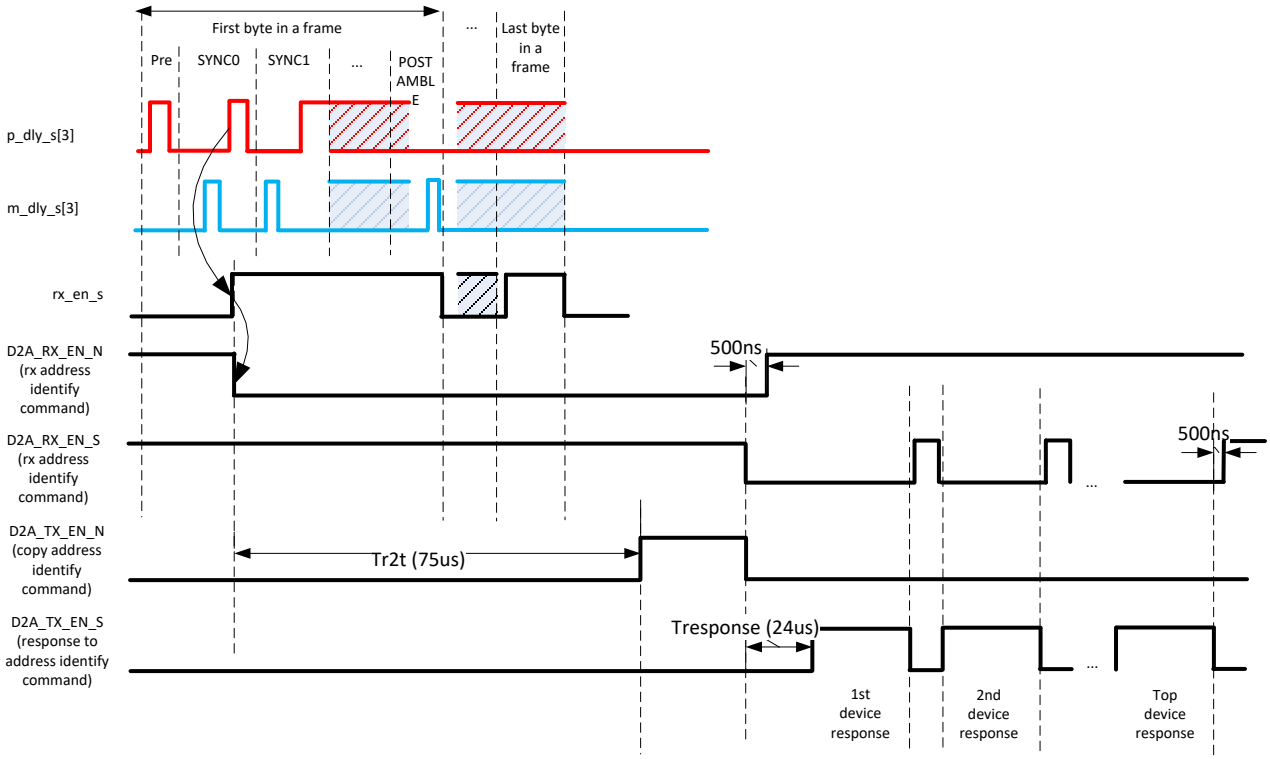


Figure4 D2A_RX_EN_X and D2A_TX_EN_X in address identify case

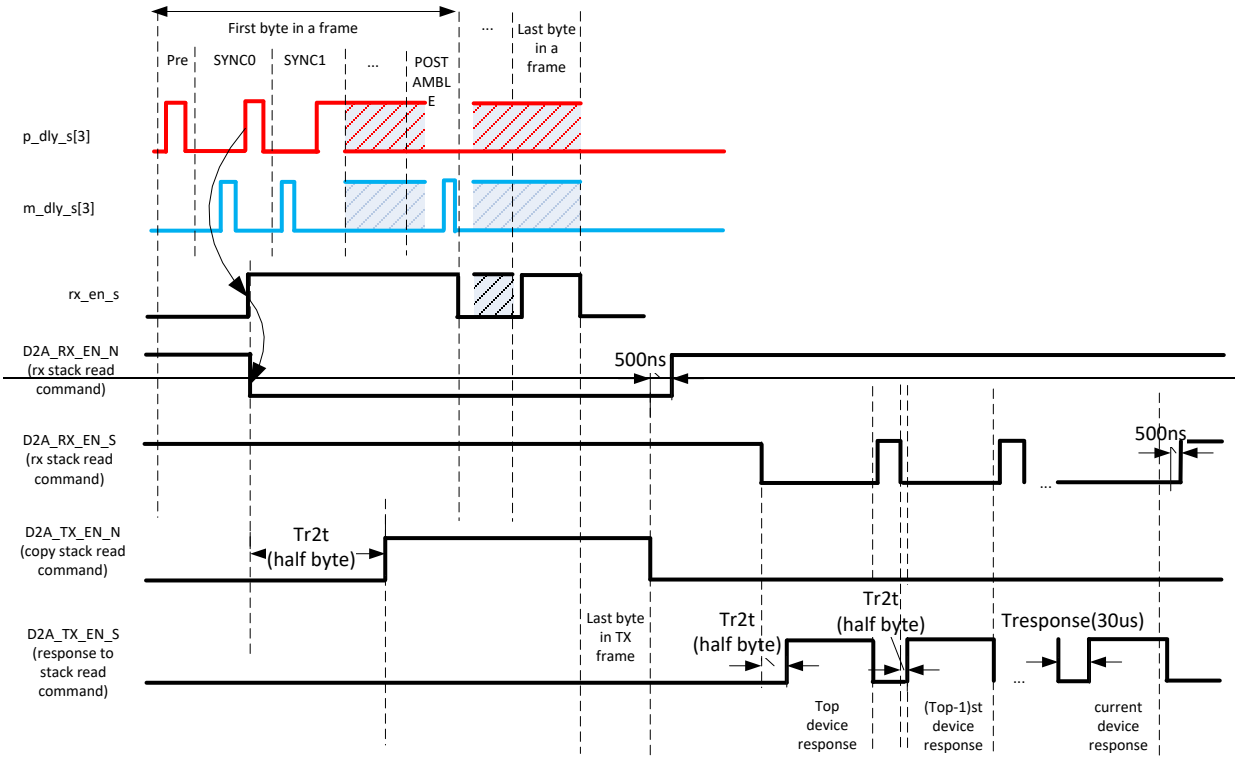


Figure5 D2A_RX_EN_X and D2A_TX_EN_X in stack read case

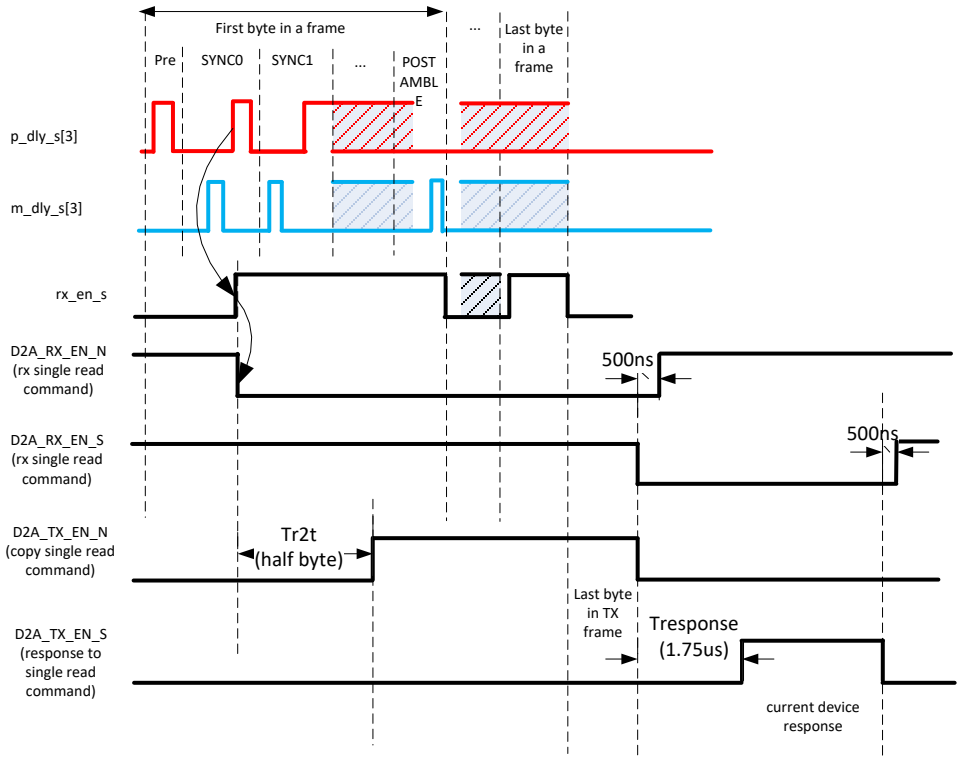


Figure6 D2A_RX_EN_X and D2A_TX_EN_X in single read case

2 Receiving

Ideally, daisy chain signal is shaped like this:

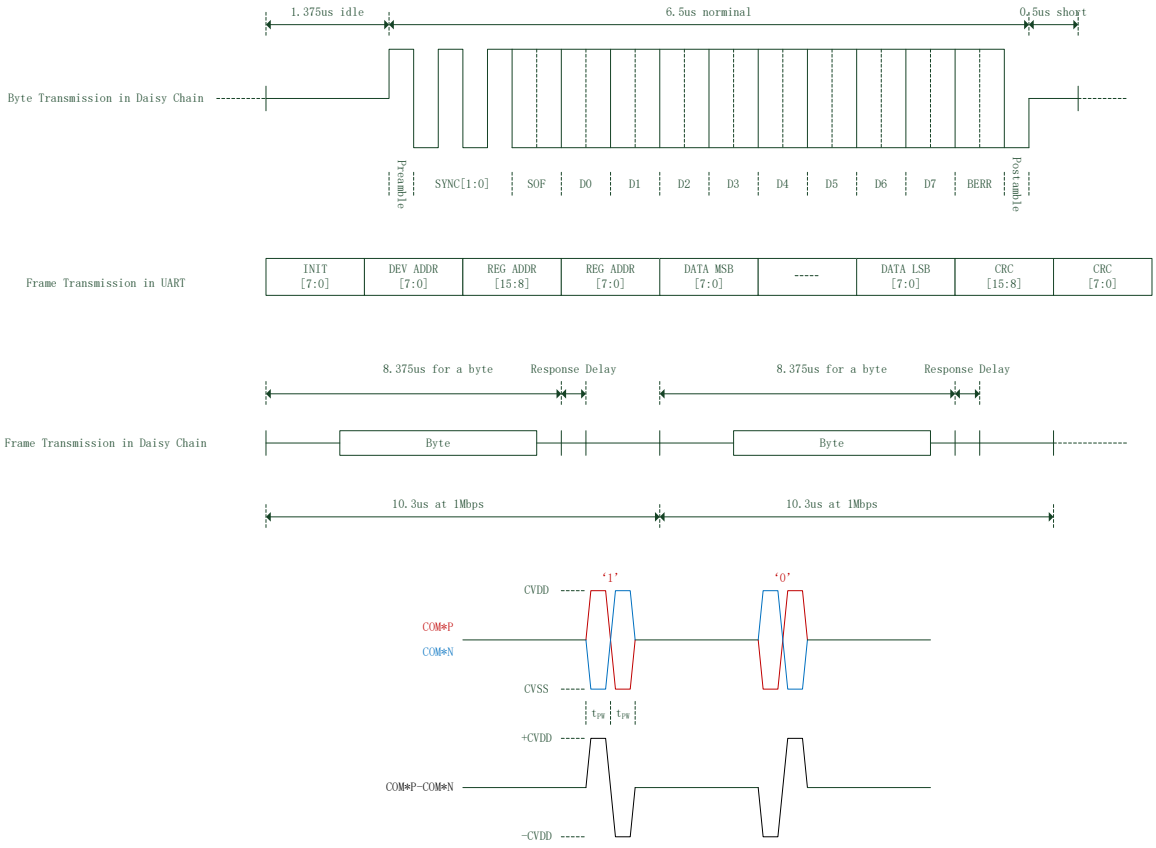


Figure3 ideal daisy chain signal

In fact, daisy chain signal may shaped like the following figure, the middle of each bit is shown in blue dummy lines, it is named “middle line” in the following description.

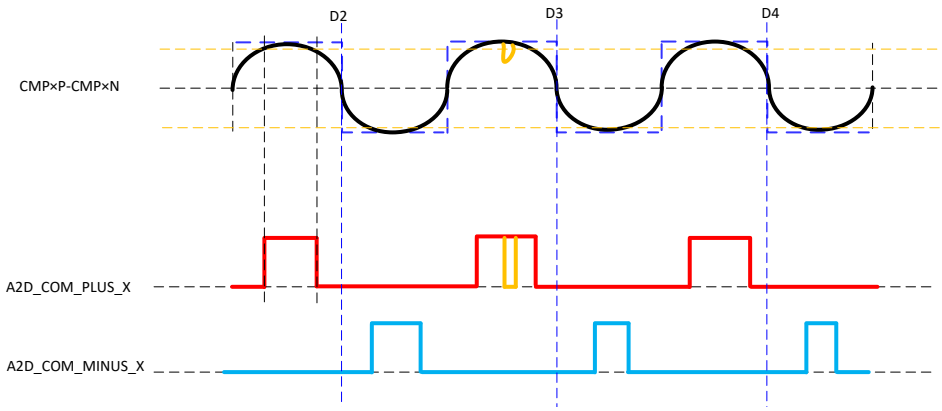


Figure4 real daisy chain signal

Capture received data(HWR001_DS_BASIC, HWR006_COMM_CTRL):

2.1 Filter input daisy chain signals:

Input filter capture A2D_COM_PLUS_X and A2D_COM_MINUS_X every clock(in this module, clock is CLK_32M_SC). Only when 2 success values are the same, the value is regarded as real, and will go on to next operations.

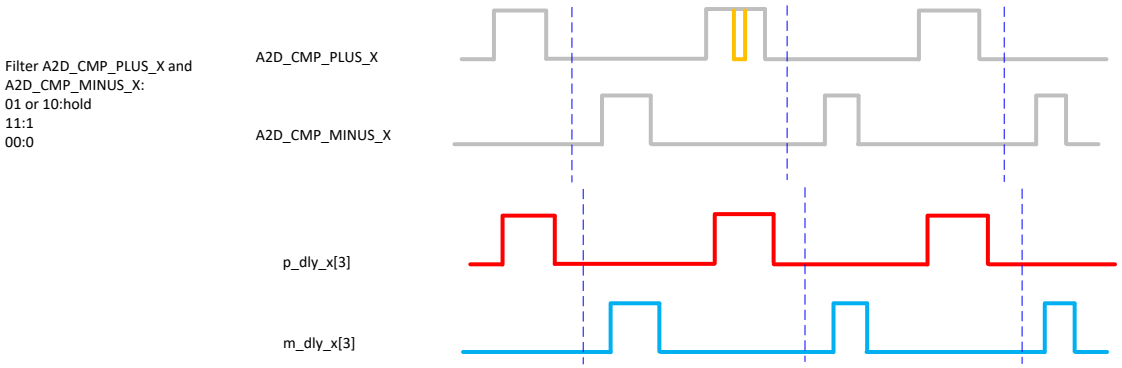


Figure5 input filter

2.2 Bit_length[4:0]

At beginning, bit_length[4:0] shall be updated every byte. As the preamble half bit may disappear, it shall not be calculated.

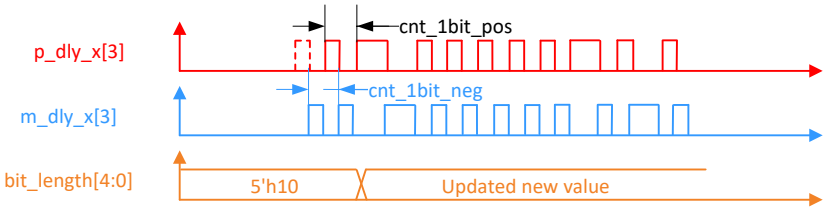


Figure6 bit_length generation

Bit_length[4:0] is the average value of cnt_1bit_pos[4:0] and cnt_1bit_neg[4:0].

2.3 cnt_pre_edge[4:0], cnt_gap_edge[3:0], gap_edge_updated and gap_edge[3:0]

These 4 signals are generated to mark the middle line of each bit. (HWR003_DS_BASIC)

Cnt_pre_edge[4:0] counts the interval time between falling edges of p_dly_x[3] or m_dly_x[3] at middle line time range. Middle line time range is defined as when cnt_pre_edge equals to BIT_LENGTH*(0.75, 1.25). cnt_pre_edge[4:0] start counting at the 2nd falling edge of m_dly_x[3].

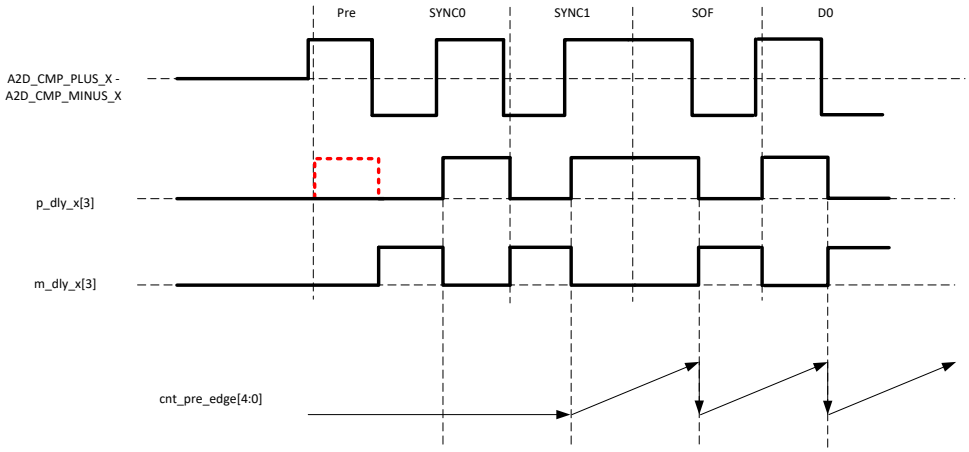


Figure7 cnt_pre_edge start to count

Cnt_gap_edge[3:0] counts the gap time between 2nd half and 1st half of each bit. Cnt_gap_edge[3:0] starts counting at the 1st time cnt_pre_edge[4:0] clears.

Gap_edge_updated is defined to ensure gap_edge update only once per bit. Gap_edge_updated is high when cnt_gap_edge[3:0] reset(2nd half of each bit comes), and is low after bit grap time.

Gap_edge[3:0] keep the maximum cnt_gap_edge[3:0] until next time cnt_pad_edge[3:0] reset to 0.

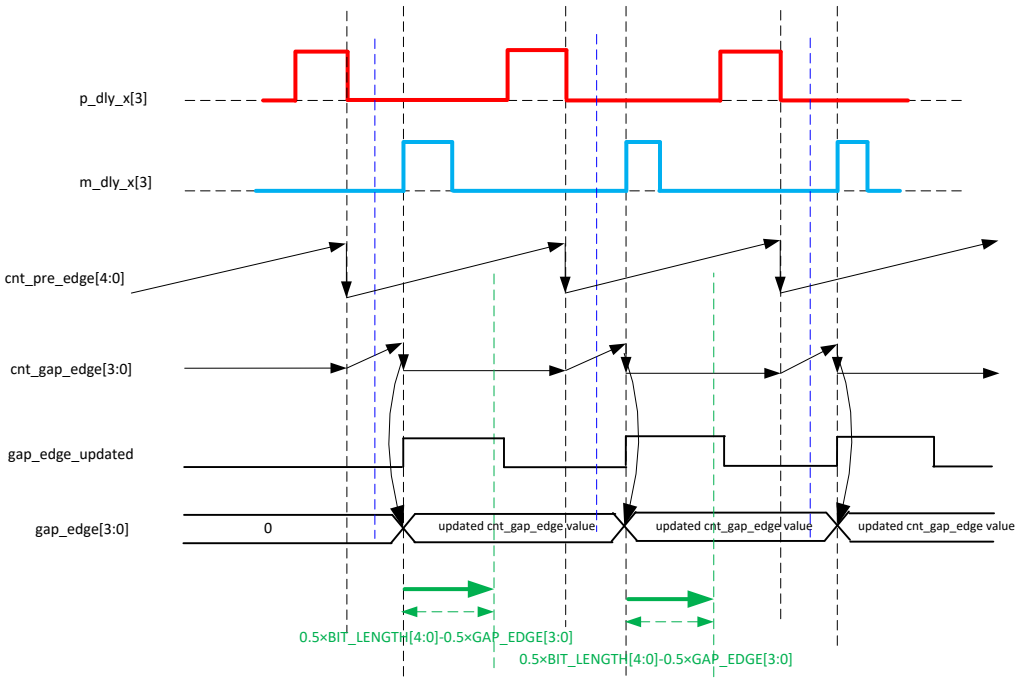


Figure8 cnt_pre_edge, cnt_gap_edge, gap_edge_updated and gap_edge

2.4 buf_h_lst_half[7:0], buf_h_2nd_half[7:0], buf_l_lst_half[7:0], buf_l_2nd_half[7:0]

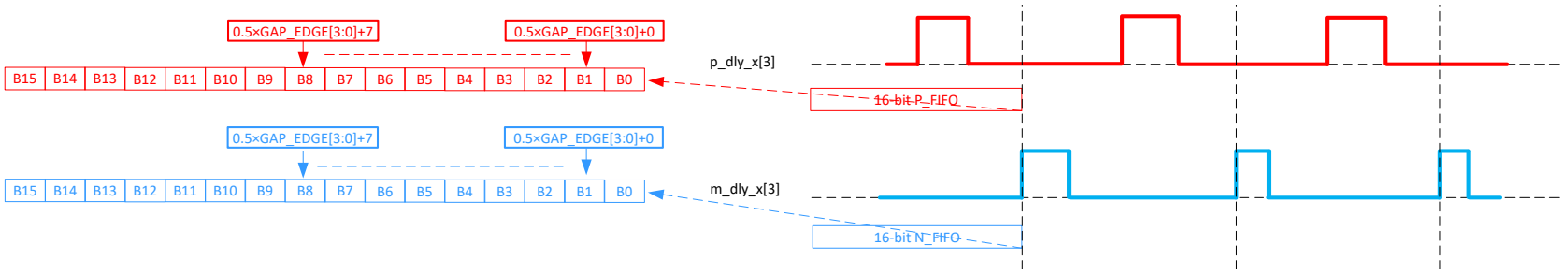


Figure9.1 buf_p[15:0] and buf_m[15:0](first half bit)

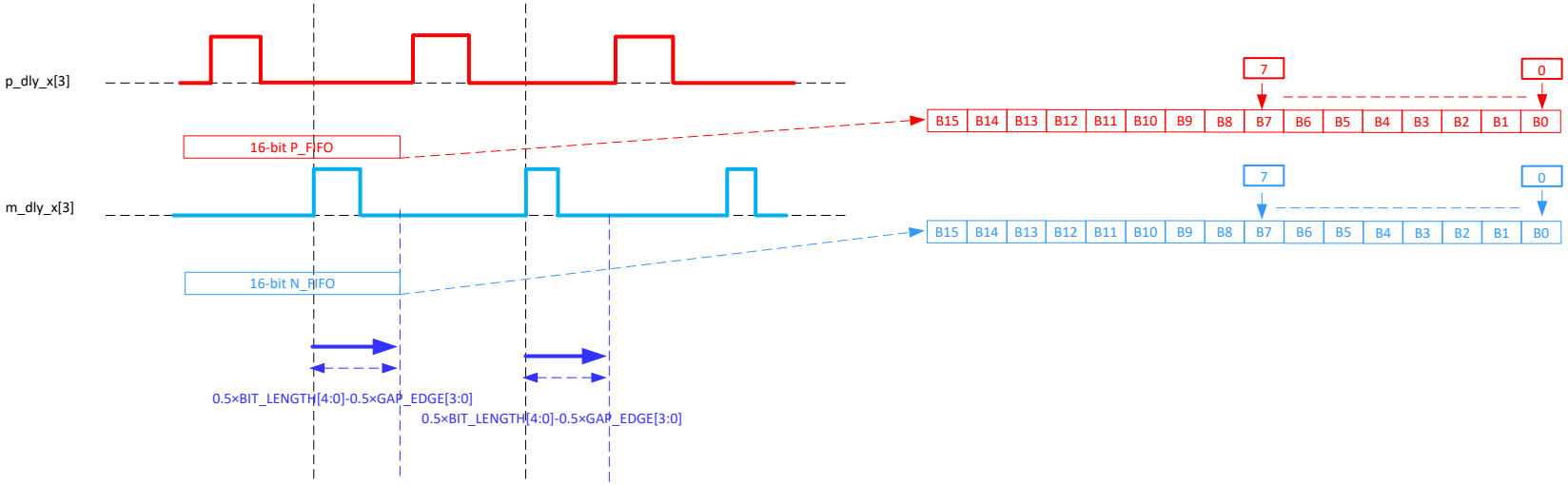


Figure9.2 buf_p[15:0] and buf_m[15:0](second half bit)

These 4 signals are generated to vote for the received bit(HWR004_DS_BASIC):

2.4.1 buf_p[15:0] and buf_m[15:0]

During a byte receiving time(rx_en_s_dsy or rx_en_n_dsy), buf_p[15:0] and buf_m[15:0] are updated every CLK_32M_SC.

In detail, when rx_en_s_dsy, buf_p[15:0] is updated every CLK_32M_SC by p_dly_s[3],

buf_m[15:0] is updated every CLK_32M_SC by m_dly_s[3];

when rx_en_n_dsy, buf_p[15:0] is updated every CLK_32M_SC by p_dly_n[3],

buf_m[15:0] is updated every CLK_32M_SC by m_dly_n[3].

2.4.2 buf_diff[15:0]

buf_diff[15:0] is the xor-ed result of buf_p[15:0] and buf_m[15:0]. Only complementary values can be used.

2.4.3 buf_h_lst_half[7:0] and buf_l_lst_half[7:0]

As the first half bit is distinguished (cnt_gap_edge[3:0]/2) clocks later than real middle bit time, buf_h_lst_half[7:0] is the higher cnt_gap_edge[3:1] bits of complementary value of buf_p[15:0]; buf_l_lst_half[7:0] is the higher cnt_gap_edge[3:1] bits of complementary value of ~buf_p[15:0];

2.4.5 buf_h_2nd_half[7:0] and buf_l_2nd_half[7:0]

Second half bit is distinguished ((BIT_LENGTH[4:0]-GAP_EDGE[3:0])/2) clocks later than the beginning time of 2nd half bit.

buf_h_2nd_half[7:0] is complementary value of buf_p[7:0], and buf_l_2nd_half[7:0] is complementary value of ~buf_p[7:0].

2.5 Cnt_hh[3:0] and cnt_ll[3:0]

Cnt_hh[3:0] count the A2D_COM_PLUS_x high and A2D_COM_MINUS_x low time during the 1st half bit time, and count the A2D_COM_MINUS_x high and A2D_COM_PLUS_x low time during the 2nd half bit time.

Cnt_ll[3:0] count the A2D_COM_MINUS_x high and A2D_COM_PLUS_x low time during the 1st half bit time, and count the A2D_COM_PLUS_x high and A2D_COM_MINUS_x low time during the 2nd half bit time.

2.6 Grap_moment[2:0]

Grap_moment[2:0] is defined to mark the time to grap bit value. Grap_moment[0] is high when gap of cnt_pre_edge and gap_edge equals to (bit_length – gap_edge)/2. High pulse is delivered to grap_moment[1], grap_moment[2] one by one.

Grap_moment[1] is the real grap time for p_bit. So it's also assigned as grap_pulse.

2.7 P_bit

P_bit is the received bit data.

When grap_moment[1] is high, if cnt_hh[3:0] > cnt_ll[3:0], p_bit is high;

if cnt_h[3:0] <= cnt_l[3:0], p_bit is low.

2.8 Cnt_bit[4:0]

Cnt_bit[4:0] counts the number of bits in each byte.

2.9 Rev_dsy_data[8:0]

When (cnt_bit[4:0] == 5'h3) and grap_pulse, p_bit means sof bit. Except for sof bit, following bits are in LSB-first order. Rev_dsy_data[8] is sof bit, rev_dsy_data[7:0] are MSB-first received data.

2.10 Rx_en_x_dsy_m, rx_en_x_dsy, rx_en_x, rx_en and neg_rx_en_dsy

Rx_en_x_dsy_m and rx_en_x_dsy are defined to differ daisy chain signals from tone.

Rx_en_x_dsy_m is high only between the 1st and 2nd rising edge of m_dly_x[3]. If no 2nd rising edge is detected, rx_en_x_dsy_m will clear to 0 when timeout (no new m_dly_x[3]'s rising edge is detected for 63 CLK_32M clocks).

Rx_en_x_dsy is high when p_dly_x[3]'s rising edge when rx_en_x_dsy_m high. Normally, rx_en_x_dsy is low when a byte is completely received. If tone signals are received, rx_en_x_dsy will clear to 0 when timeout(no new p_dly_x[3]'s posedge is detected for 32 CLK_32M clocks).

Rx_en_x considers the SPI ports and rx_en_x_dsy.

If SPI replace S port(SPI_EN & !SPI_DIR), rx_en_s is 0, else rx_en_s is rx_en_s_dsy.

If SPI replace N port(SPI_EN & SPI_DIR), rx_en_n is 0, else rx_en_n is rx_en_n_dsy.

Rx_en is the or-ed result of rx_en_s_dsy and rx_en_n_dsy. All data are only captured when rx_en high(HWR012_DS_BASIC).

Neg_rx_en_dsy is the negedge of rx_en.

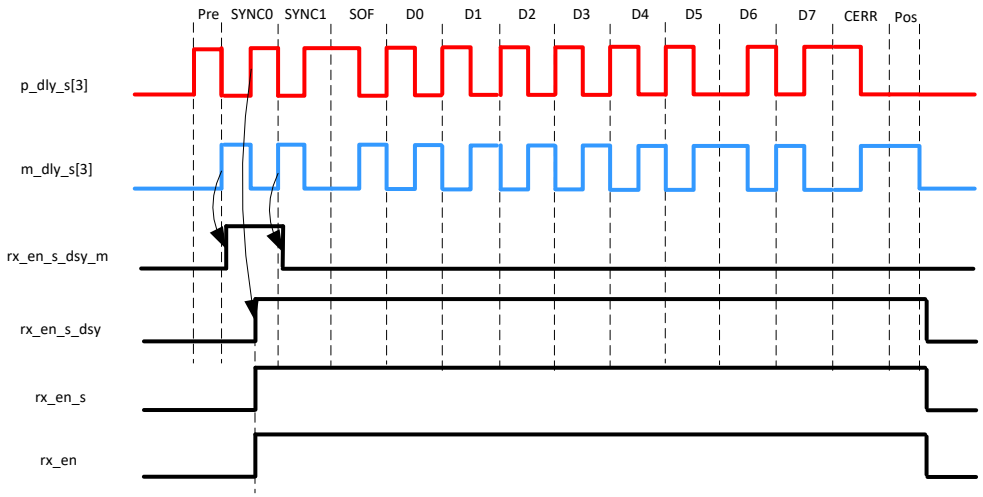


Figure10.1 rx_en_x_dsy_m, rx_en_x_dsy, rx_en_x and rx_en

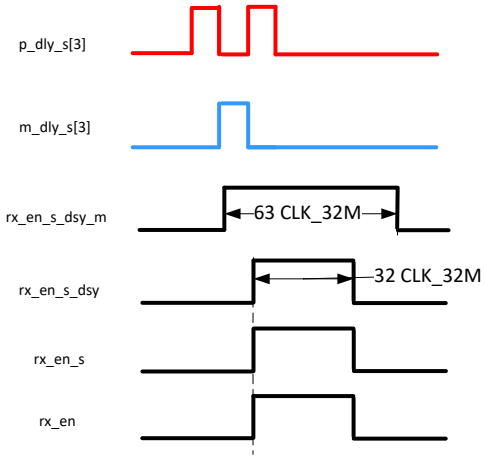


Figure10.2 positive tone disturb

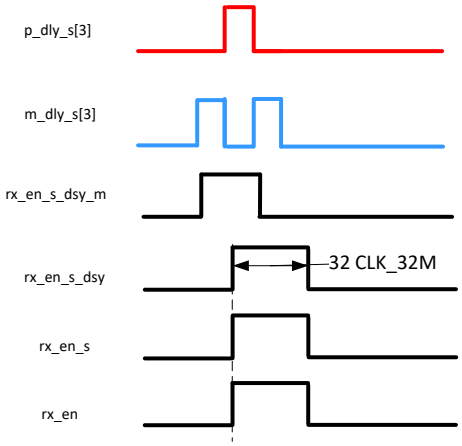


Figure10.3 negative tone disturb

2.11 Tail_blanking and idle_clear(HWR011_DS_BASIC)

Tail_blanking is a mask signal to avoid receiving. It is defined to ignore tail from analog part. After a byte is completely received, within (500+8*STACK_RESPONSE*T_{CLK_32M})ns, no edges of p_dly_x[3] and m_dly_x[3] shall be used for receiving.

Idle_clear is a clear signal for cnt_m_dly_neg[1:0], which counts the number of m_dly_x[3]'s falling edge. When cnt_m_dly_neg[1:0] gets clear, all information about the last byte is totally removed. Idle_clear is high if p_dly_x[3] and m_dly_x[3] and tail_blanking are all 0 for 500ns. Idle_clear is low when p_dly_x[3] or m_dly_x[3] is high.

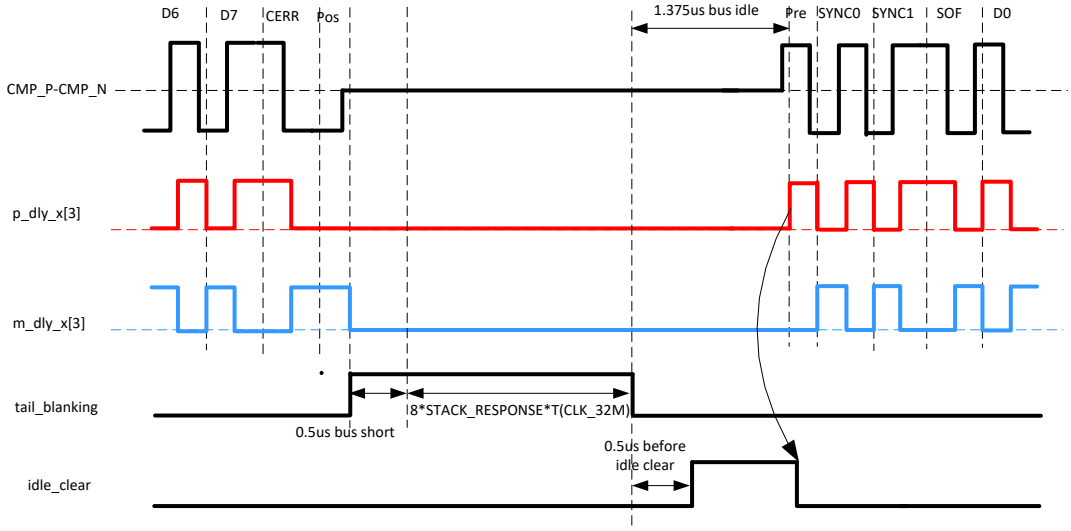


Figure11 tail_blanking and idle_clear

3 Transmitting

After tx_start(from instance u_COMM_CTRL) high, u_DS_BASIC starts to transmit data.

3.1 Send_start and tx_en[1:0]

Rising edge of tx_start lead to signals send_start and tx_en[1:0].

3.2 Cnt_tx_bit[4:0], cnt_tx_bit_num[3:0] and tx_character_ends(HWR006_COMM_CTRL)

When transmitting starts, tx_en[0] high. cnt_tx_bit[4:0] starts to count the main clock number per bit. Cnt_tx_bit_num[3:0] starts to count the number of bits per byte.

When (cnt_tx_bit_num[3:0]>=4'hD), a byte is completely transmitted, tx_character_ends is high

3.1 Tx_one(HWR006_COMM_CTRL)

Tx_one captures data one bit by one bit(according to cnt_tx_bit[4:0] and cnt_tx_bit_num[3:0]) from tx_data[8:0](from instance u_COMM_CTRL). Tx_data[8] is the first bit, others are in LSB-first order. When in 1st half bit(cnt_tx_bit[4:0]<8), tx_one is the inverted value of last bit; when in 2nd half bit, tx_one is the value of current bit.

3.3 WR_COM_PLUS and WR_COM_MINUS(HWR006_COMM_CTRL)

WR_COM_PLUS and WR_COM_MINUS influence the final output D2A_WR_COM_PLUS and D2A_ WR_COM_MINUS. WR_COM_PLUS and WR_COM_MINUS are complementary signals which output tx_one bit by bit.

3.2 Tx_crc[15:0]

Tx_crc[15:0] is result of CRC16 algorithm with polynomial 8005(X16+X15+X2+1), initial CRC 16'hFFFF, using tx_one as input bit stream.