canola_axi_slave

Address width: 32

Data width: 32

Base address: 0x00000000

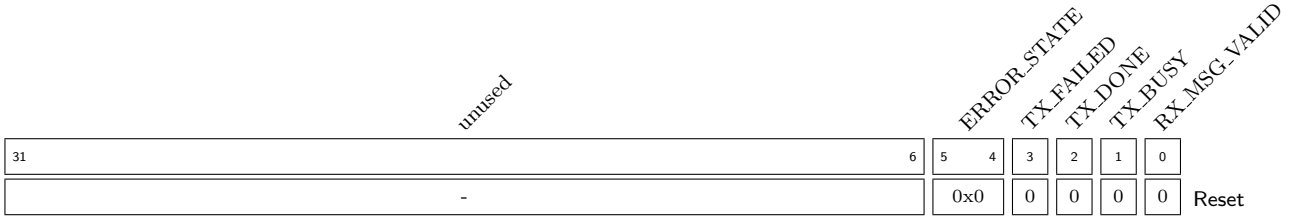
AXI-Lite slave for Canola CAN Controller

1 Register List

#	Name	Mode	Address	Type	Length	Reset
0	STATUS	RO	0x00000000	FIELDS	6	0x0
1	CONTROL	PULSE	0x00000004	FIELDS	1	0x0
2	CONFIG	RW	0x00000008	FIELDS	2	0x0
3	BTL_PROP_SEG	RW	0x00000020	SLV	16	0x7
4	BTL_PHASE_SEG1	RW	0x00000024	SLV	16	0x7
5	BTL_PHASE_SEG2	RW	0x00000028	SLV	16	0x7
6	BTL_SYNC_JUMP_WIDTH	RW	0x0000002C	SLV	2	0x1
7	BTL_TIME_QUANTA_CLOCK_SCALE	RW	0x00000030	SLV	8	0xF
8	TRANSMIT_ERROR_COUNT	RO	0x00000034	SLV	16	0x0
9	RECEIVE_ERROR_COUNT	RO	0x00000038	SLV	16	0x0
10	TX_MSG_SENT_COUNT	RO	0x0000003C	SLV	16	0x0
11	TX_ACK_RECV_COUNT	RO	0x00000040	SLV	16	0x0
12	TX_ARB_LOST_COUNT	RO	0x00000044	SLV	16	0x0
13	TX_ERROR_COUNT	RO	0x00000048	SLV	16	0x0
14	RX_MSG_RECV_COUNT	RO	0x0000004C	SLV	16	0x0
15	RX_CRC_ERROR_COUNT	RO	0x00000050	SLV	16	0x0
16	RX_FORM_ERROR_COUNT	RO	0x00000054	SLV	16	0x0
17	RX_STUFF_ERROR_COUNT	RO	0x00000058	SLV	16	0x0
18	TX_MSG_ID	RW	0x0000005C	FIELDS	31	0x0
19	TX_PAYLOAD_LENGTH	RW	0x00000060	SLV	4	0x0
20	TX_PAYLOAD_0	RW	0x00000064	FIELDS	32	0x0
21	TX_PAYLOAD_1	RW	0x00000068	FIELDS	32	0x0
22	RX_MSG_ID	RO	0x0000006C	FIELDS	31	0x0
23	RX_PAYLOAD_LENGTH	RO	0x00000070	SLV	4	0x0
24	RX_PAYLOAD_0	RO	0x00000074	FIELDS	32	0x0
25	RX_PAYLOAD_1	RO	0x00000078	FIELDS	32	0x0

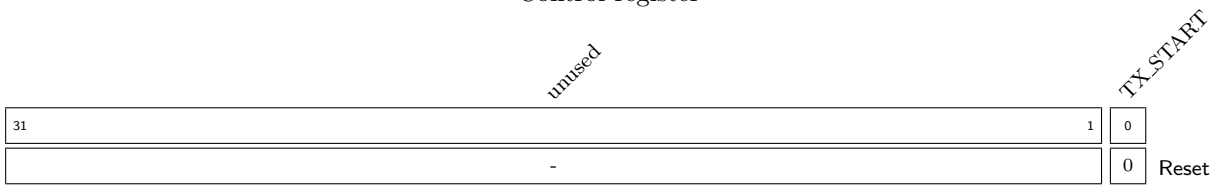
2 Registers

Register 2.1: STATUS - RO (0x00000000)
Status register



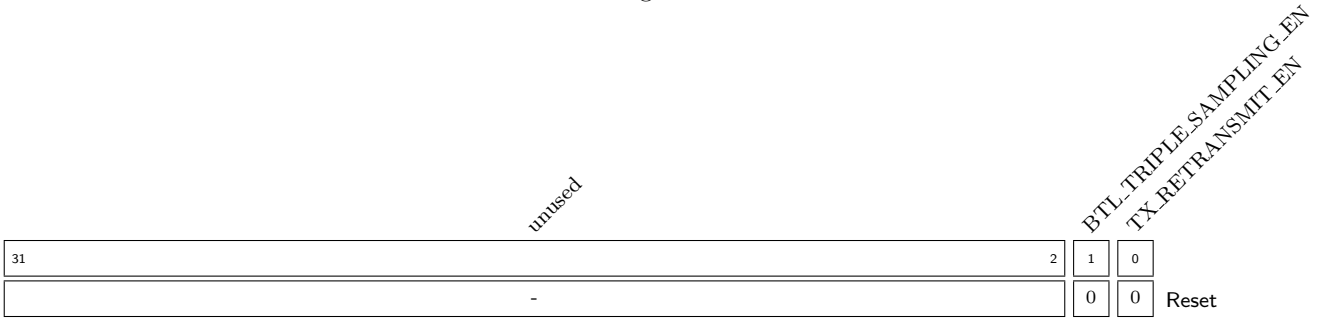
RX_MSG_VALID	Received message is valid
TX_BUSY	Busy transmitting message
TX_DONE	Done transmitting message
TX_FAILED	Transmitting message failed
ERROR_STATE	Error state. b00 = ERROR_ACTIVE, b01 = ERROR_PASSIVE, b1X = BUS_OFF

Register 2.2: CONTROL - PULSE FOR 1 CYCLES - (0x00000004)
Control register



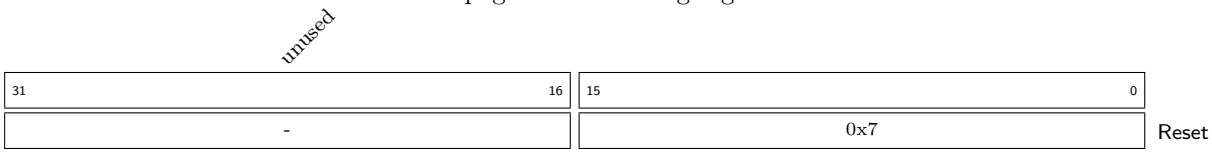
TX_START	Start transmitting message

Register 2.3: CONFIG - RW (0x00000008)
Control register

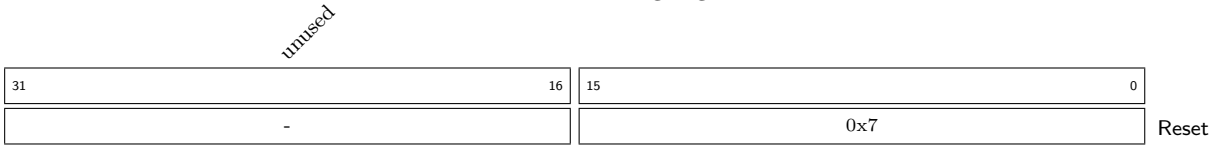


TX_RETRANSMIT_EN	Enable retransmission of messages that failed to send
BTL_TRIPLE_SAMPLING_EN	Enable triple sampling of bits

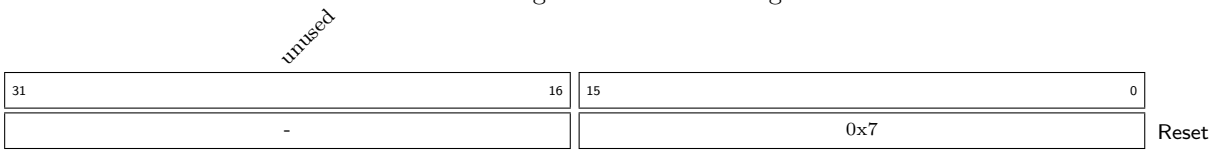
Register 2.4: BTL_PROP_SEG - RW (0x00000020)
Propagation bit timing segment



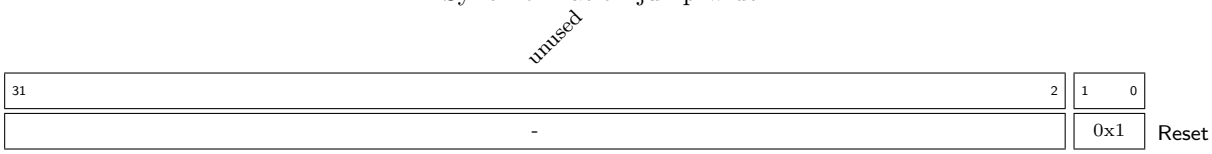
Register 2.5: BTL_PHASE_SEG1 - RW (0x00000024)
Phase 1 bit timing segment



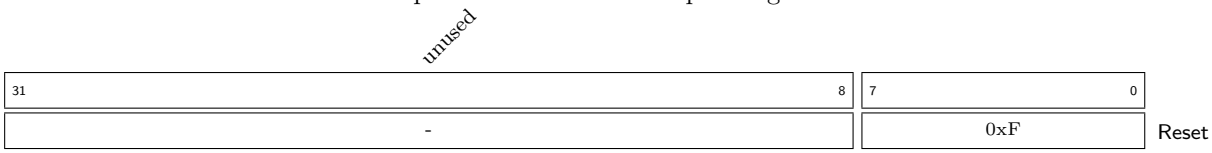
Register 2.6: BTL_PHASE_SEG2 - RW (0x00000028)
Phase segment 2 of bit timing



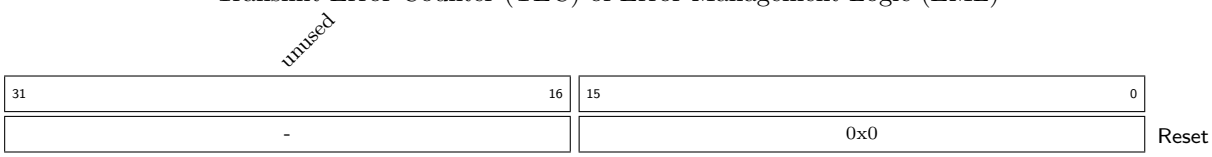
Register 2.7: BTL_SYNC_JUMP_WIDTH - RW (0x0000002C)
Synchronization jump width



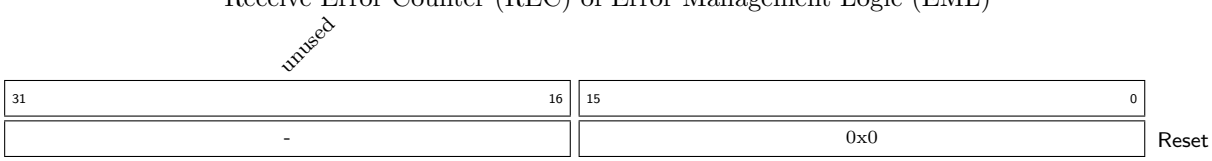
Register 2.8: BTL_TIME_QUANTA_CLOCK_SCALE - RW (0x00000030)
Clock prescale ratio for time quanta generator



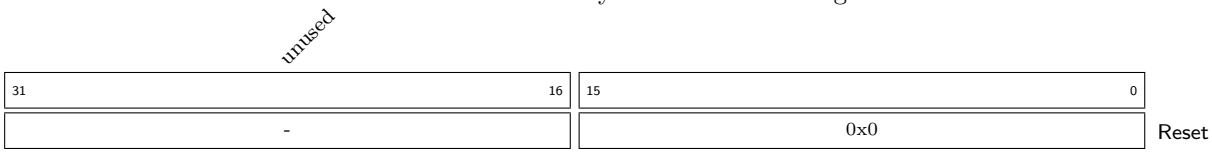
Register 2.9: TRANSMIT_ERROR_COUNT - RO (0x00000034)
Transmit Error Counter (TEC) of Error Management Logic (EML)



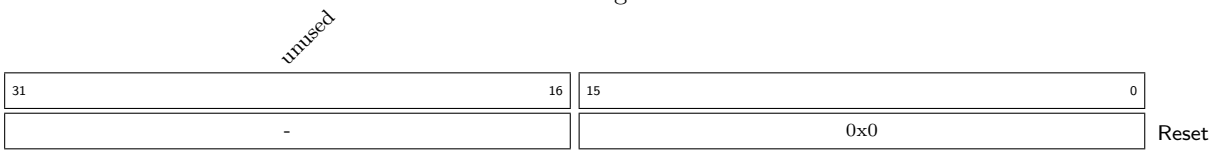
Register 2.10: RECEIVE_ERROR_COUNT - RO (0x00000038)
Receive Error Counter (REC) of Error Management Logic (EML)



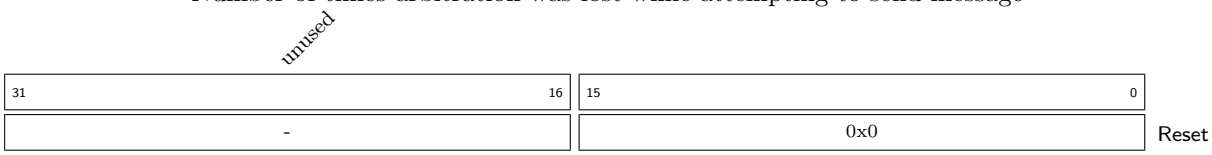
Register 2.11: TX_MSG_SENT_COUNT - RO (0x0000003C)
Number of successfully transmitted messages



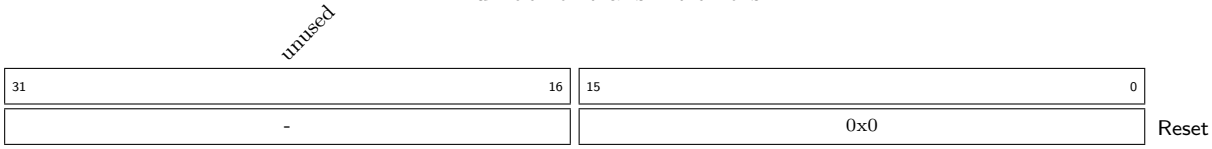
Register 2.12: TX_ACK_RECV_COUNT - RO (0x00000040)
Number of transmitted messages where ACK was received



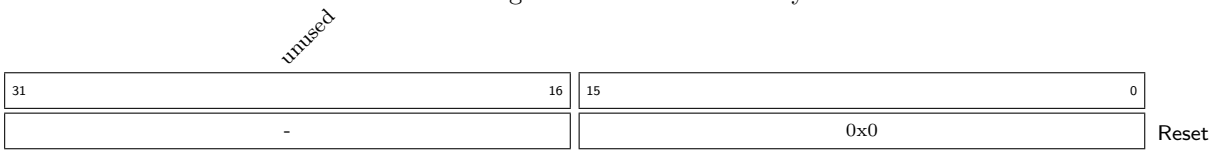
Register 2.13: TX_ARB_LOST_COUNT - RO (0x00000044)
Number of times arbitration was lost while attempting to send message



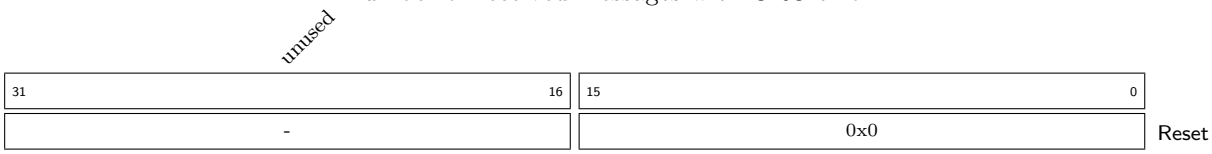
Register 2.14: TX_ERROR_COUNT - RO (0x00000048)
Number of transmit errors



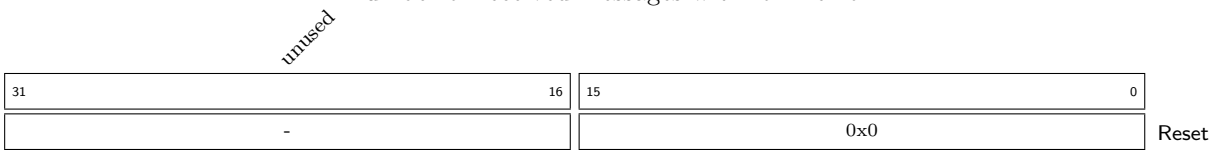
Register 2.15: RX_MSG_RECV_COUNT - RO (0x0000004C)
Number of messages that were successfully received



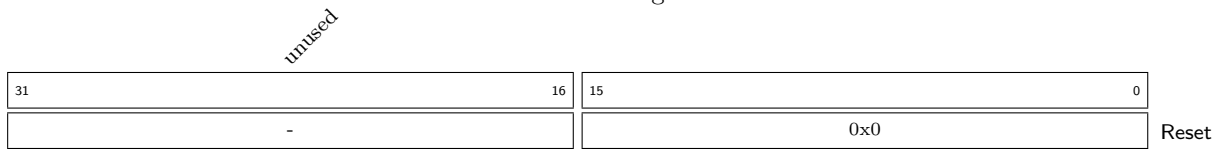
Register 2.16: RX_CRC_ERROR_COUNT - RO (0x00000050)
Number of received messages with CRC error



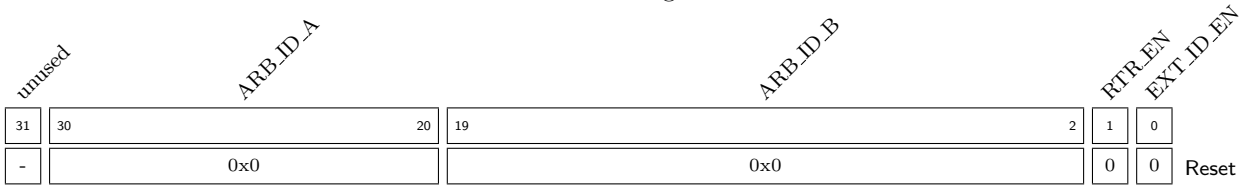
Register 2.17: RX_FORM_ERROR_COUNT - RO (0x00000054)
Number of received messages with form error



Register 2.18: RX_STUFF_ERROR_COUNT - RO (0x00000058)
Number of received messages with stuff error



Register 2.19: TX_MSG_ID - RW (0x0000005C)
Number of received messages with stuff error



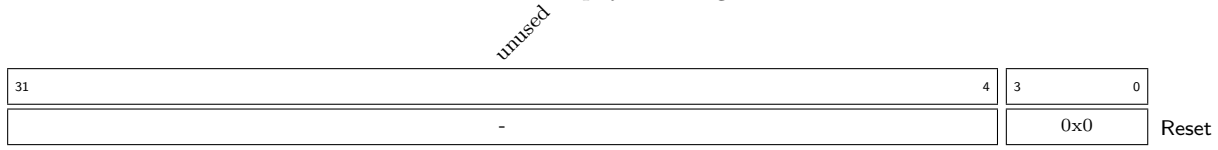
EXT_ID_EN Transmit message with extended ID

RTR_EN Remote Transmission Request

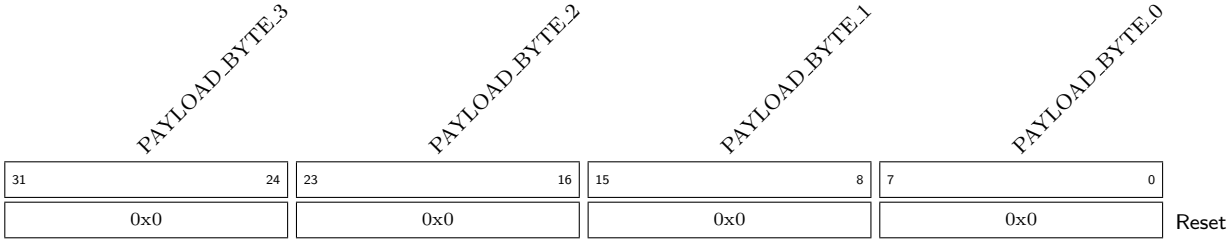
ARB_ID_B Arbitration ID B (extended only)

ARB_ID_A Arbitration ID A

Register 2.20: TX_PAYLOAD_LENGTH - RW (0x00000060)
Transmit payload length



Register 2.21: TX_PAYLOAD_0 - RW (0x00000064)
Tx payload bytes 0 to 3



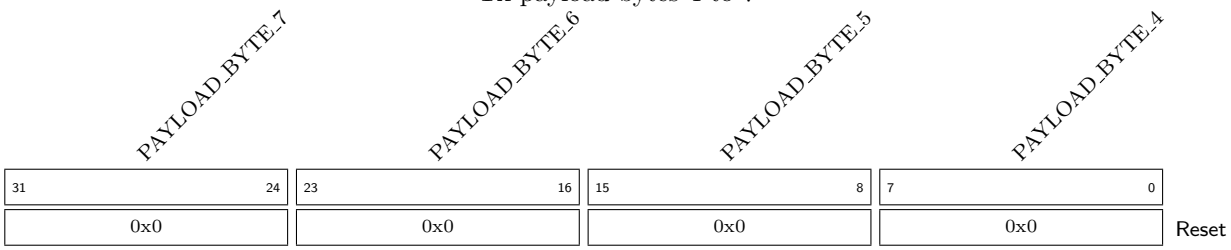
PAYLOAD_BYTE_0 Payload byte 0

PAYLOAD_BYTE_1 Payload byte 1

PAYLOAD_BYTE_2 Payload byte 2

PAYLOAD_BYTE_3 Payload byte 3

Register 2.22: TX_PAYLOAD_1 - RW (0x00000068)
Tx payload bytes 4 to 7



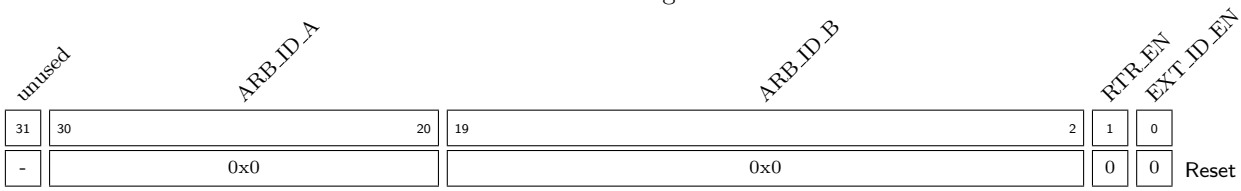
PAYLOAD_BYTE_4 Payload byte 4

PAYLOAD_BYTE_5 Payload byte 5

PAYLOAD_BYTE_6 Payload byte 6

PAYLOAD_BYTE_7 Payload byte 7

Register 2.23: RX_MSG_ID - RO (0x0000006C)
Number of received messages with stuff error



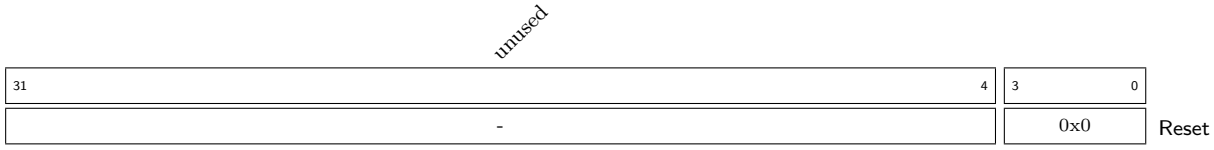
EXT_ID_EN Received message with extended ID

RTR_EN Received Remote Transmission Request (RTR)

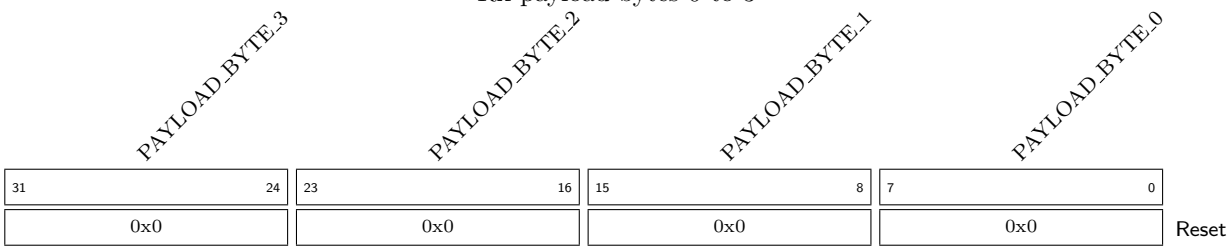
ARB_ID_B Received Arbitration ID B (extended only)

ARB_ID_A Received Arbitration ID A

Register 2.24: RX_PAYLOAD_LENGTH - RO (0x00000070)
Received payload length



Register 2.25: RX_PAYLOAD_0 - RO (0x00000074)
Rx payload bytes 0 to 3



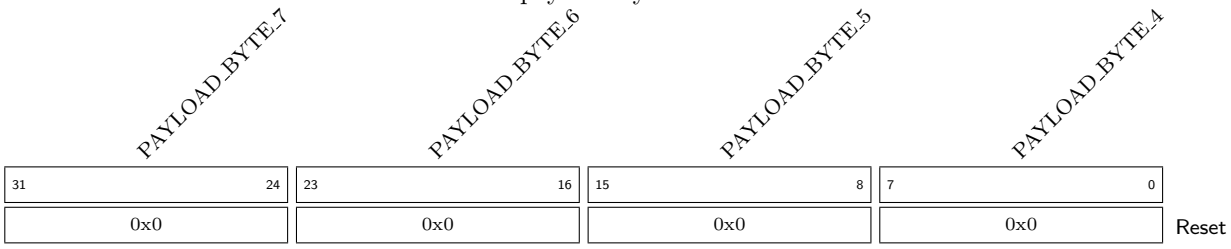
PAYLOAD_BYTE_0 Payload byte 0

PAYLOAD_BYTE_1 Payload byte 1

PAYLOAD_BYTE_2 Payload byte 2

PAYLOAD_BYTE_3 Payload byte 3

Register 2.26: RX_PAYLOAD_1 - RO (0x00000078)
Rx payload bytes 4 to 7



PAYLOAD_BYTE_4 Payload byte 4

PAYLOAD_BYTE_5 Payload byte 5

PAYLOAD_BYTE_6 Payload byte 6

PAYLOAD_BYTE_7 Payload byte 7

3 Example VHDL Register Access

All registers are bundled in records based on their mode. E.g. all RW registers are accessed through the record *bustype_rw_regs*. Access is also dependent on the type of register. All register of type SL, SLV and DEFAULT are all directly accessed by just specifying the mode record signal. E.g. the RW register *reg0* can be assigned a value like this (assuming AXI-bus):

```
axi_rw_regs.reg0 <= (others => '0');
```

Registers of type FIELD cannot be directly accessed without specification of a certain field. This is because the registers are implemented as a record in VHDL (thus a record of records). E.g. if the RO register *reg1* contains the field *field3* it can be accessed like this (assuming AXI-bus):

```
axi_ro_regs.reg1.field3 <= (others => '0');
```