alpide_control

Version: 1.0

Thursday 6th February, 2020 15:03

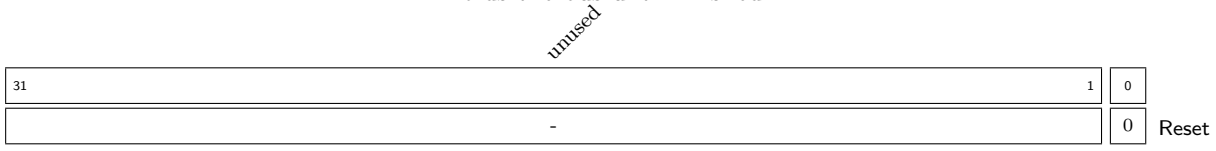
ALPIDE slow control management.

1 Register List

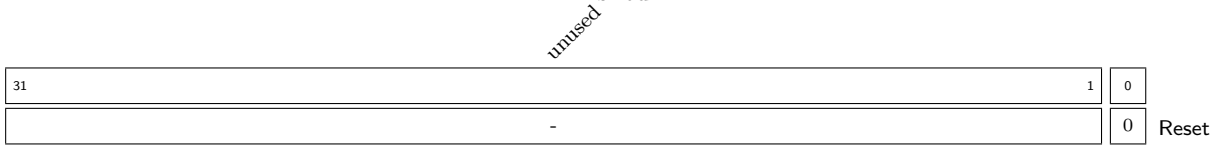
#	Name	Mode	Address	Type	Length	Reset
0	start_write	PULSE/STALL	0x00000000	SL	1	0x0
1	start_read	PULSE/STALL	0x00000001	SL	1	0x0
2	write_ctrl	RW	0x00000002	FIELDS	20	0x0
3	write_address	RW	0x00000003	SLV	16	0x0
4	write_data	RW	0x00000004	SLV	16	0x0
5	control_settings	RW	0x00000005	FIELDS	13	0xA40
6	control_status	RO	0x00000006	FIELDS	1	0x0
7	read_status	RO	0x00000007	FIELDS	12	0x0
8	read_data	RO	0x00000008	SLV	16	0x0
9	reset_counters	PULSE	0x00000009	SL	1	0x0
10	num_trigger_sent	RO	0x0000000A	DEFAULT	32	0x0
11	num_trigger_not_sent	RO	0x0000000B	DEFAULT	32	0x0

2 Registers

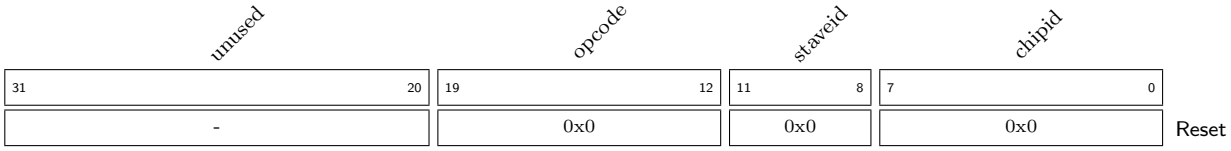
Register 2.1: START_WRITE - PULSE FOR 1 CYCLES - STALL FOR 70 CYCLES (0x00000000)
When written to, start the alpide_control state machine for any write type (all opcodes except read) and holds the bus until finished.



Register 2.2: START_READ - PULSE FOR 1 CYCLES - STALL FOR 116 CYCLES (0x00000001)
When written to, start the alpide_control state machine for the read opcode and holds the bus until finished.



Register 2.3: WRITE_CTRL - RW (0x00000002)
Opcode decides the operation executed. Multicast and multistavecast can be combined to make a multistave multicast, i.e. write to all chips on the pRU.

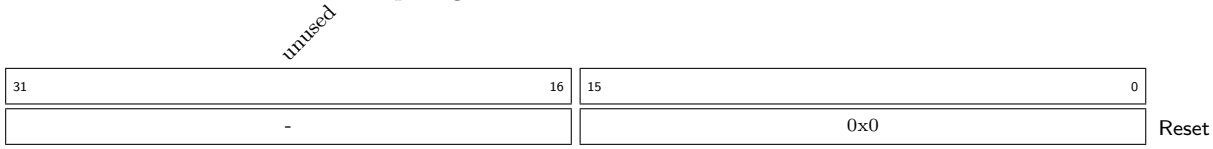


chipid Chip ID. For multicast write, use 0x0F. Multicast writes to all chips on selected. Multicast not allowed for read.

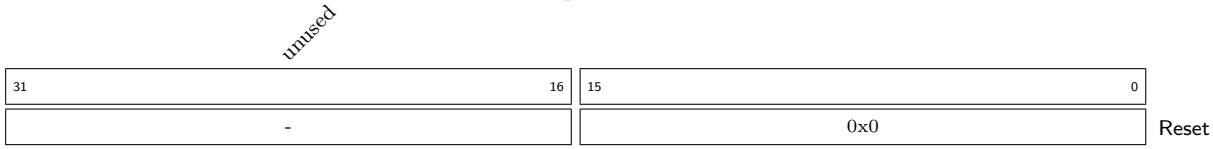
staveid Stave ID. For multistavecast write, use 0xF. Multistavecast writes to all selected chips on all staves. E.g. one can write to CHIP ID # 3 on all staves. Multistavecast not allowed for read.

opcode ALPIDE opcode. E.g. 0x4E for unicast read. See ALPIDE manual p.33.

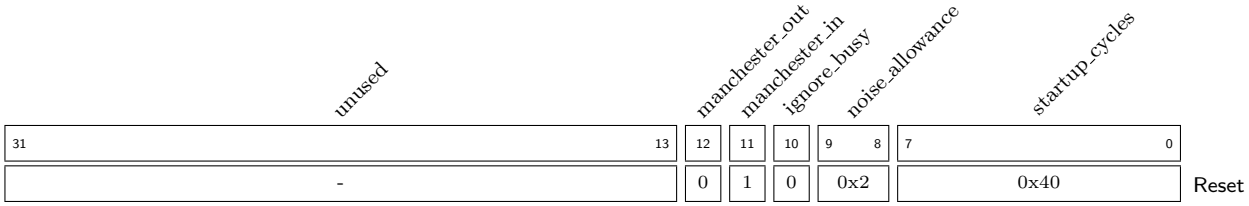
Register 2.4: WRITE_ADDRESS - RW (0x00000003)
The ALPIDE chips register address. Must be written before write_ctrl.



Register 2.5: WRITE_DATA - RW (0x00000004)
Data that is written to ALPIDE register. Must be written before write_ctrl when performing a write operation.



Register 2.6: CONTROL_SETTINGS - RW (0x00000005)
Control module settings.



startup_cycles How many sample clock cycles (240 MHz) to wait before looking for incoming transmission. Dependent on cable length.

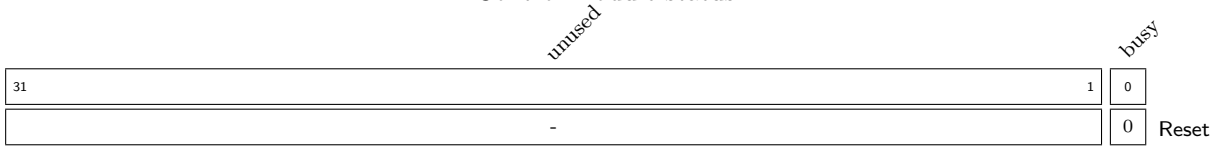
noise_allowance Requirement for the start/stop bit detection. 3 -> All sampled bits in period (strict - may fail with noisy signal). 2 -> Allow for 1 less bit. 1 -> 2 less bits. 0 -> Not allowed (will revert to all bits).

ignore_busy If set, ignores that a chip on the stave is busy, and will transmit trigger anyhow.

manchester_in Assumes the incoming stream is encoded. Manchester off is not yet implemented!

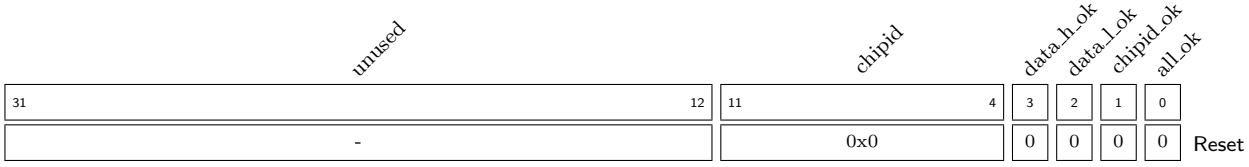
manchester_out NOT IMPLEMENTED! Future feature to enable manchester encoding of outgoing stream.

Register 2.7: CONTROL_STATUS - RO (0x00000006)
Control module status.



busy Indicates whether the control module is busy.

Register 2.8: READ_STATUS - RO (0x00000007)
Read status.



all_ok

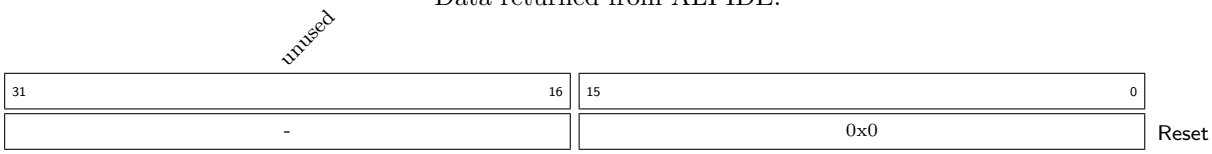
chipid_ok

data_l_ok

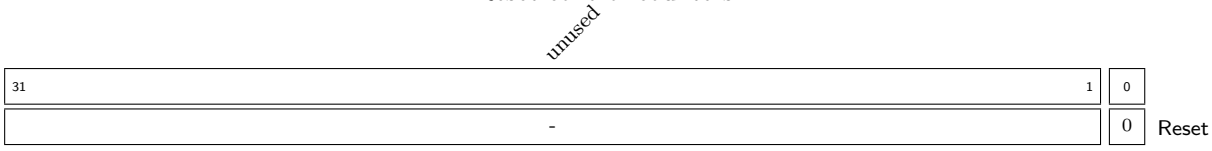
data_h_ok

chipid

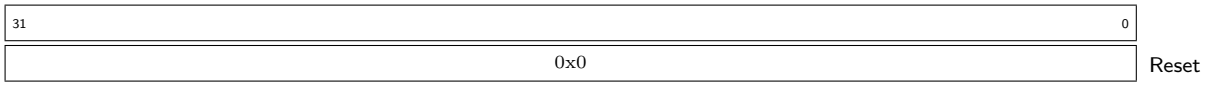
Register 2.9: READ_DATA - RO (0x00000008)
Data returned from ALPIDE.



Register 2.10: RESET_COUNTERS - PULSE FOR 1 CYCLES (0x00000009)
Reset control counters.



Register 2.11: NUM_TRIGGER_SENT - RO (0x0000000A)
Number of triggers transmitted.



Register 2.12: NUM_TRIGGER_NOT_SENT - RO (0x0000000B)
Number of triggers not transmitted because chip is busy.