DCS CAN Bus Interface

ITS WP10 Uprade PRR

Introduction

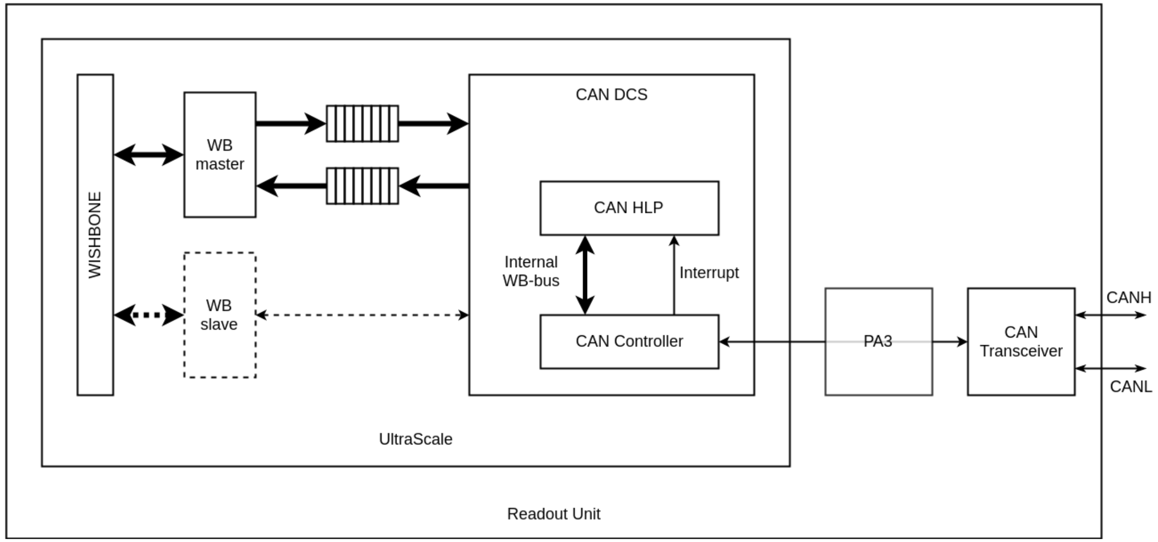
- Readout Unit must be able to communicate with DCS via
  - GBT
  - CAN bus
- CAN bus to be used when GBT is down (e.g. during shutdowns)
- CAN transceiver on RUv1 is wired to PA3
  - CAN controller and HLP engine with wishbone interface should reside in US
  - CAN signals will pass through PA3 via GPIOs on the US
- DCS group has given us freedom to define our own protocol
- OpenCores CAN controller will be used
  - Has been tested in the PA3 firmware
  - Verified that OpenCores controller and external CAN transceiver works

OpenCores CAN Protocol Controller

- https://opencores.org/project,can
- Written in Verilog
- Supports basic CAN and extended CAN
  - 11-bit ID and 29-bit ID
  - ID mask and filtering
- Up to 1Mbps operation
- Transmit/receive/error interrupts
- Wishbone interface (8-bit address and data)
- Register map compatible with Philips SJA1000 CAN Controller IC
  - https://www.nxp.com/docs/en/data-sheet/SJA1000.pdf
- Size: 12k gates (930 flip-flops)

Planned firmware implementation in UltraScale

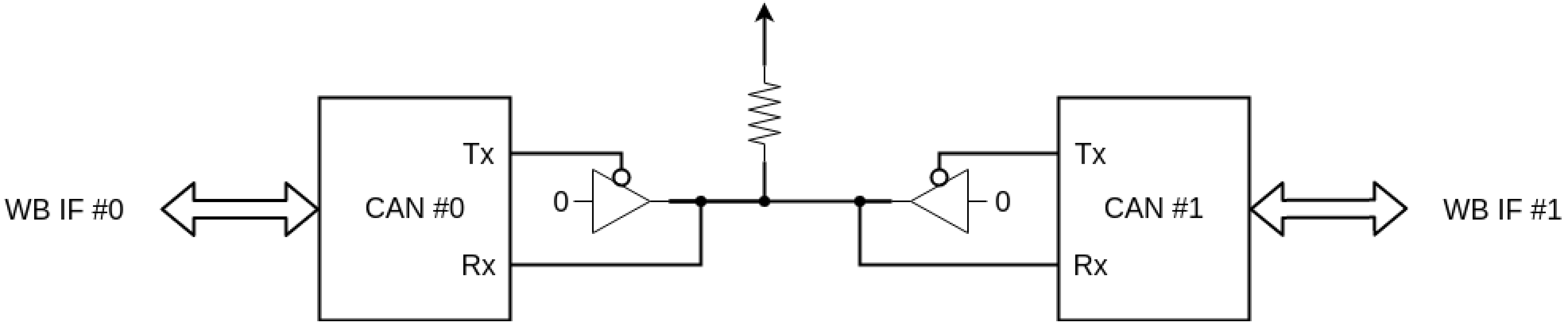
- CAN DCS module based on OpenCores CAN controller
  - Custom "High Level Protocol" (HLP)
    - Simple READ and WRITE commands from DCS
    - READ_RESPONSE and WRITE_RESPONSE provided by Readout Unit
  - Wishbone master
    - DCS commands access wishbone bus via CAN bus



Backup/Reference

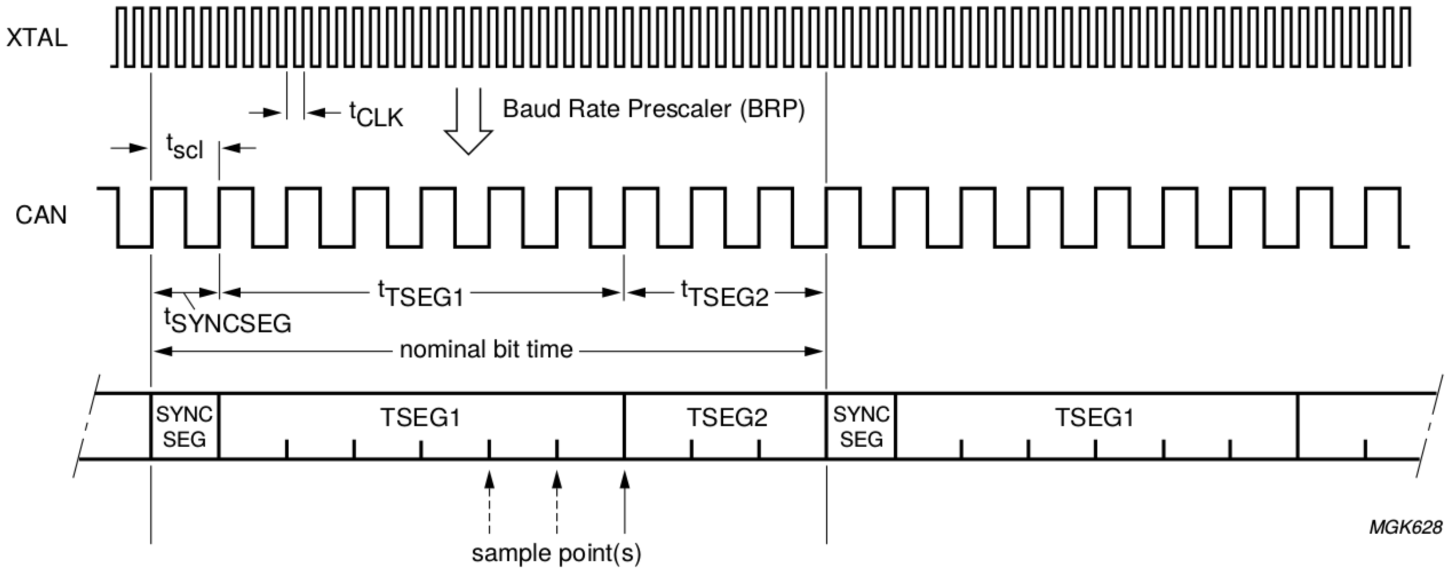
OpenCores CAN Protocol Controller Simulations

- Tested using a simple Bitvis UVVM style testbench
- Two instances of the CAN controller connected
- Two separate WB interfaces to write to each controller
- 40 MHz clock



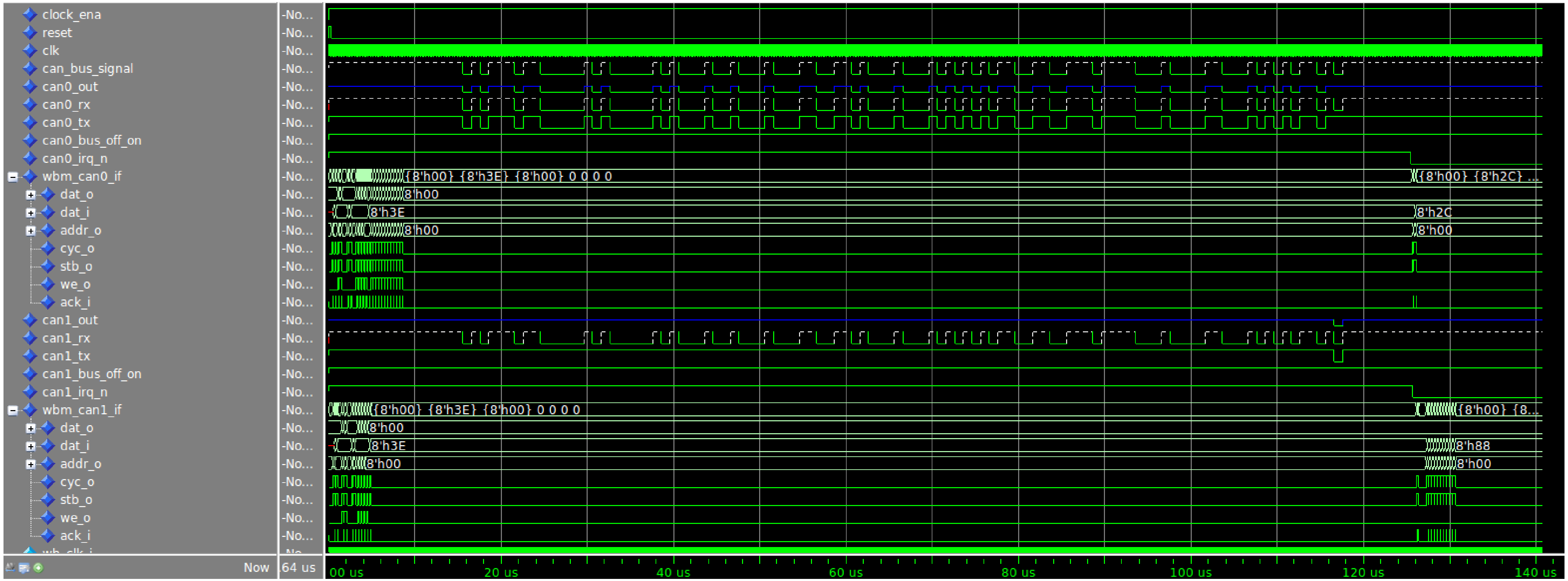
OpenCores CAN Protocol Controller Simulations

- Bit Timing Register (BTR0) configured for 4x baud clock prescale
- t_{SEG1} set to 7 baud clocks, t_{SEG2} set to 3 baud clocks, in BTR1
- $t_{SYNCSEG}$ is always 1 baud clock
- Gives us a bit rate of 40 MHz / (4 * (1+7+3)) = 1Mbps



OpenCores CAN Protocol Controller Simulations

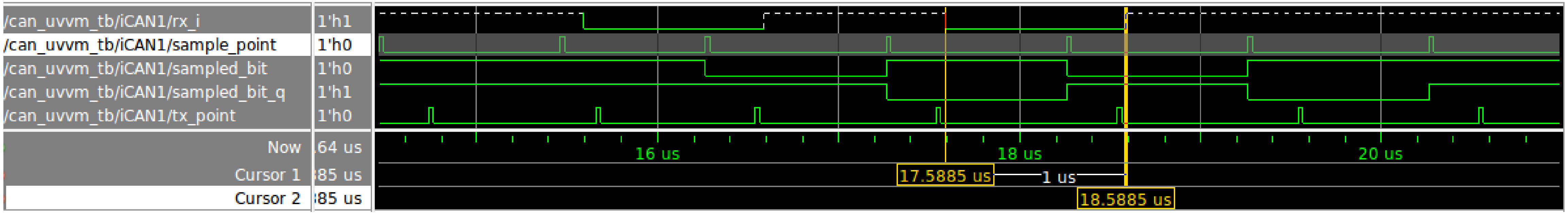
- Simulation waveforms, showing initial WB transactions on both controllers, CAN transmission, and CAN IRQ lines going low after message



OpenCores CAN Protocol Controller Simulations

ALICE ITS UPGRADE

- sample_point marks the point where bits on rx_i are being sampled
- The simulated bit period is 1 us, as it was configured for
- The sampling point is located at 7/10ths of a microsecond into each bit, corresponding to what t_{SEG1} and t_{SEG2} were configured for



OpenCores CAN Protocol Controller Simulations

ALICE ITS UPGRADE

Testbench log

Set up TX buffer on CAN0 with message for CAN1 (ID 0xBB)

Wait for CAN1 to receive message

Verify message contents

```
# UVVM: ID_LOG_HDR                    4940.0 ns  TB seq.            Start a transaction from CAN0 to CAN1
# UVVM: ------------------------------------------------------------------------------------------------------------------------------------------------------
# UVVM: ID_BFM                        5115.0 ns  TB seq.            wb_write(A:x"0A", x"BB") completed. Set TXID1 to xBB
# UVVM: ID_BFM                        5465.0 ns  TB seq.            wb_write(A:x"0B", x"08") completed. Set TXID2 to x08, 8 bytes data
# UVVM: ID_BFM                        5815.0 ns  TB seq.            wb_write(A:x"0C", x"11") completed. Set data1 to x11
# UVVM: ID_BFM                        6165.0 ns  TB seq.            wb_write(A:x"0D", x"22") completed. Set data2 to x22
# UVVM: ID_BFM                        6515.0 ns  TB seq.            wb_write(A:x"0E", x"33") completed. Set data3 to x33
# UVVM: ID_BFM                        6865.0 ns  TB seq.            wb_write(A:x"0F", x"44") completed. Set data4 to x44
# UVVM: ID_BFM                        7215.0 ns  TB seq.            wb_write(A:x"10", x"55") completed. Set data5 to x55
# UVVM: ID_BFM                        7565.0 ns  TB seq.            wb_write(A:x"11", x"66") completed. Set data6 to x66
# UVVM: ID_BFM                        7915.0 ns  TB seq.            wb_write(A:x"12", x"77") completed. Set data7 to x77
# UVVM: ID_BFM                        8265.0 ns  TB seq.            wb_write(A:x"13", x"88") completed. Set data8 to x88
# UVVM: ID_BFM                        8615.0 ns  TB seq.            wb_write(A:x"01", x"01") completed. Request transmission on CAN0
# UVVM:
# UVVM:
# UVVM: ID_LOG_HDR                    8615.0 ns  TB seq.            Wait for CAN1 to receive message
# UVVM: ------------------------------------------------------------------------------------------------------------------------------------------------------
# UVVM:
# UVVM:
# UVVM: ID_LOG_HDR                  125613.5 ns  TB seq.            Got interrupt from CAN1.
# UVVM: ------------------------------------------------------------------------------------------------------------------------------------------------------
# UVVM: ID_BFM                      125790.0 ns  TB seq.            wb_check(A:x"00", x"XX")=> OK, received data = x"3E". Check that CAN0 transmit interrupt was set
# UVVM: ID_BFM                      126140.0 ns  TB seq.            wb_check(A:x"02", x"XX")=> OK, received data = x"2C". Check that CAN0 transmit complete status bit is set
# UVVM: ID_BFM                      126315.0 ns  TB seq.            wb_check(A:x"00", x"XX")=> OK, received data = x"3E". Check that CAN1 receive interrupt was set
# UVVM:
# UVVM:
# UVVM: ID_LOG_HDR                  127315.0 ns  TB seq.            Verify message received by CAN1
# UVVM: ------------------------------------------------------------------------------------------------------------------------------------------------------
# UVVM: ID_BFM                      127490.0 ns  TB seq.            wb_check(A:x"14", x"BB")=> OK, received data = x"BB". Verify received RXID1
# UVVM: ID_BFM                      127840.0 ns  TB seq.            wb_check(A:x"15", x"08")=> OK, received data = x"8". Verify received RXID2, 8 bytes data
# UVVM: ID_BFM                      128190.0 ns  TB seq.            wb_check(A:x"16", x"11")=> OK, received data = x"11". Verify received data byte 1
# UVVM: ID_BFM                      128540.0 ns  TB seq.            wb_check(A:x"17", x"22")=> OK, received data = x"22". Verify received data byte 2
# UVVM: ID_BFM                      128890.0 ns  TB seq.            wb_check(A:x"18", x"33")=> OK, received data = x"33". Verify received data byte 3
# UVVM: ID_BFM                      129240.0 ns  TB seq.            wb_check(A:x"19", x"44")=> OK, received data = x"44". Verify received data byte 4
# UVVM: ID_BFM                      129590.0 ns  TB seq.            wb_check(A:x"1A", x"55")=> OK, received data = x"55". Verify received data byte 5
# UVVM: ID_BFM                      129940.0 ns  TB seq.            wb_check(A:x"1B", x"66")=> OK, received data = x"66". Verify received data byte 6
# UVVM: ID_BFM                      130290.0 ns  TB seq.            wb_check(A:x"1C", x"77")=> OK, received data = x"77". Verify received data byte 7
# UVVM: ID_BFM                      130640.0 ns  TB seq.            wb_check(A:x"1D", x"88")=> OK, received data = x"88". Verify received data byte 8
```

CAN bus testing on RUv1

- AnaGate CAN adapter used for testing (same as DCS group is using).
- Successfully sent/received CAN messages to/from CAN controller in PA3

AnaGate CAN Monitor program

```
AnaGate CAN at 192.168.1.254:5001 ( 1Msg/s) ( 3Msg) View:continuous
08.03.2018 16:10:28.657    682 (0x02aa) :  0xaa  0xbb  0xcc  0xdd  0xee  0xff  0x11  0x22  - Sended to partner
08.03.2018 16:02:29.638   1496 (0x05d8) :  0x11  0x22  0x33  0x44  0x55  0x66  0x77  0x88
08.03.2018 16:02:02.569   1496 (0x05d8) :  0x11  0x22  0x33  0x44  0x55  0x66  0x77  0x88
08.03.2018 15:58:40.707    170 (0x00aa) :  0xaa  0xbb  0xcc  0xdd  0xee  0xff  0x11  0x22  - Sended to partner
08.03.2018 15:56:26.850   1496 (0x05d8) :  0x11  0x22  0x33  0x44  0x55  0x66  0x77  0x88
AnaGate device succesfully connected.
```

54	CAN_RXB_ID1	8212	8	FF	✓
55	CAN_RXB_ID2	8213	8	E8	✓
56	CAN_RXB_DATA1	8214	8	AA	✓
57	CAN_RXB_DATA2	8215	8	BB	✓
58	CAN_RXB_DATA3	8216	8	CC	✓
59	CAN_RXB_DATA4	8217	8	DD	✓
60	CAN_RXB_DATA5	8218	8	EE	✓
61	CAN_RXB_DATA6	8219	8	FF	✓
62	CAN_RXB_DATA7	8220	8	11	✓
63	CAN_RXB_DATA8	8221	8	22	✓

Readout Unit PA3 GUI software (RX buffer registers)

44	CAN_TXB_ID1	8202	8	BB	✓
45	CAN_TXB_ID2	8203	8	8	✓
46	CAN_TXB_DATA1	8204	8	11	✓
47	CAN_TXB_DATA2	8205	8	22	✓
48	CAN_TXB_DATA3	8206	8	33	✓
49	CAN_TXB_DATA4	8207	8	44	✓
50	CAN_TXB_DATA5	8208	8	55	✓
51	CAN_TXB_DATA6	8209	8	66	✓
52	CAN_TXB_DATA7	8210	8	77	✓
53	CAN_TXB_DATA8	8211	8	88	✓

Readout Unit PA3 GUI software (TX buffer registers)