System Documentation

LiDAR-CAN Bridge Module v1.2.1

General Description:

The system will handle data flow from MID-360 LiDAR sensor Ethernet interface to CAN-FD bus by using the STM32F401RBT6 MCU as the central processor and data bridge. This MCU will manage the communication between the W5500 Ethernet controller and the TCAN4550RGY CAN-FD controller through independent high-speed SPI buses. Data from the LiDAR will be received via Ethernet, processed by the MCU, and then re-packaged and transmitted onto the CAN-FD bus. Additionally, the board have internal power management circuits to regulate input power to suitable +12VDC and +3.3VDC levels to use in the system components.

1. Data Flow Description:

The data flow in the board can be described in four main stages:

1.1. Ethernet Data Reception (LiDAR to W5500):

The Livox MID-360 LiDAR sends point cloud and other data over the Ethernet connection. This data, which is typically encapsulated in UDP packets, is received by the W5500 Ethernet Controller. The W5500's primary role is to handle the low-level physical and data link layers of the Ethernet protocol. It’s hardwired TCP/IP stack automatically manages the network protocol headers and places the payload data into its internal TX/RX buffers.

1.2. Data Transfer (W5500 to STM32F401):

The W5500 notifies the STM32F401RBT6 MCU that new data has arrived in its RX buffer by asserting the interrupt line. The MCU then uses the high-speed SPI bus to read the data from the W5500's buffers. The STM32's firmware is responsible for a few key tasks here:

- Reading the UDP payload from the W5500's buffers.
- Parsing the received data to extract relevant information, such as the point cloud data, time stamps, or status messages.
- Implementing any protocol translation or data filtering logic.

1.3. Data Processing and Translation (STM32F401 MCU):

The STM32F401 MCU will process the raw data from the LiDAR and re-format it into CAN-FD frames. This involves:

- Packet Parsing: Extracting specific data fields from the Ethernet packets. The raw LiDAR point cloud data is often too large to fit into a single CAN-FD frame.
- Data Aggregation or Fragmentation: Depending on the data being sent, the MCU may need to aggregate smaller data points into a single CAN-FD frame or, more likely, fragment larger data sets across multiple CAN-FD frames.
- Frame Construction: Building the CAN-FD frames with the appropriate identifier, payload, and control bits as required by the CAN-FD network protocol.

1.4. CAN-FD Data Transmission (STM32F401 to TCAN4550RGY):

Once a CAN-FD frame is constructed in the MCU's memory, the STM32F401 uses its second high-speed SPI bus to write the frame data and control commands to the TCAN4550RGY CAN-FD Controller. The TCAN4550's internal CAN-FD controller takes over from there, managing the CAN-FD protocol, arbitration, and transmission. It then uses its integrated CAN-FD transceiver to convert the digital signals into the analog differential signals required for the physical CAN-FD bus.

2. Proposed Test Strategy:

This board was designed with different features to facilitate testing in PCBA prototypes. General DFM and DFT guidelines were used to ensure that the final system can be actually manufactured, assembled, and tested.

2.1. Hardware Validation:

This phase ensures that the board's power delivery and physical connections are correct before introducing any firmware.

- Visual Inspection: Physically inspect the board for any solder bridges, misaligned components, or missing parts.
- Continuity Check: Use a multimeter in continuity mode to check for unintended shorts, especially between power and ground pins on each IC.
- Voltage Measurement: Apply input power and use a multimeter to measure the output voltage at each regulator's output. Verify the readings match the expected values (3.3V and 12V). Use the on-board test points to measure voltages with higher precision and less risk of shorting.
- LED Indicators: Confirm that the power rail LEDs illuminate correctly (D2=12V OK, D4=3.3V OK).
- Board bring-up: Place the R1 and R2 jumpers to power a single voltage rail at a time and confirm their voltage levels on the complete PDN (Power Delivery Network).

2.2. MCU Bring-Up and Basic Firmware Test:

- MCU Power: Ensure the MCU's power rail is stable and correctly supplied.
- Debugger Connection: Connect the SWD programmer/debugger to the board's SWD interface (J5).

- Basic Firmware: Write a simple "blink" firmware that toggles the USER LED pin (PA2) of the MCU.
- Programming and Test: Program the MCU with this firmware. A successful upload and the blinking USER LED confirm the MCU is powered, clocked, and programmable. This validates the SWD interface and the fundamental MCU operation.

2.3. Peripheral Validation (Modular Approach):

This phase uses basic firmware to test each peripheral independently. Start with one, then add the next.

2.3.1. W5500 Ethernet Controller Test:

- SPI Communication: Write firmware to initialize the W5500 via its dedicated SPI bus. Send simple commands and read back a known register value (e.g., the VERSIONR register) to confirm

the SPI communication is working. Use a logic analyzer or oscilloscope on the SPI test points to debug any issues.
- Link Status: Connect an Ethernet cable from the W5500 to a PC or switch. Write firmware to read the link status register on the W5500. The W5500's own status LEDs should also indicate a successful link.
- Echo Server Test: Implement a simple UDP or TCP echo server in the firmware using the W5500's hardwired stack. Send a packet from a PC to the board and verify that the board echoes the packet back. This confirms the entire Ethernet data path is functional.

2.3.2. TCAN4550RGY CAN-FD Controller Test:
- SPI Communication: Write firmware to initialize the TCAN4550 via its dedicated SPI bus. Read and write to registers to confirm the SPI communication is stable. Again, use a logic analyzer on the SPI test points if needed.
- CAN Loopback Test: The TCAN4550 can be put into a loopback mode where it transmits a message and immediately receives it back internally. Write firmware to configure this mode, send a CAN-FD frame, and check for the received frame in the RX buffer. This verifies the controller's functionality without a physical bus connection.
- Physical Bus Test: Connect the board to a known good CAN-FD node (e.g., a PC with a CAN interface). Write firmware to transmit a simple CAN-FD message. Use a CAN bus analyzer to confirm the message is correctly transmitted and received by the other node. This validates the TCAN4550's transceiver and the physical bus wiring.

2.4. Full System Integration and Data Flow:
Once all subsystems are validated individually, proceed with the final functional test developing a complete firmware for the bridge. This firmware should:

- Initialize both the W5500 and TCAN4550.
- Wait for a command or data packet from the Livox MID-360 over Ethernet.
- Once received, read the data from the W5500 via the MCU SPI 2 Bus.
- Process the data and construct a CAN-FD frame.
- Send the CAN-FD frame to the TCAN4550 via the MCU SPI 1 Bus for transmission on the physical CAN-FD bus.

Finally connect the Livox MID-360, Bridge module, and a CAN bus analyzer to run the complete code and monitor the CAN bus analyzer to confirm that the corresponding CAN-FD frames are correctly received.

Note: The user LED can be used as a status indicator at each stage of this process (e.g., blinking slowly for Ethernet link up, blinking fast for a CAN message transmitted).