

Project Assembly Steps

2.1 Main Block: CAN Controller (`can_controller_std`)

The `can_controller_std` is the heart of the system. It implements the standard CAN protocol (11-bit ID), being responsible for sending and receiving CAN frames on the bus. This block manages communication, organizes data packets, controls message transmission and monitoring, and provides status signals that are used by the rest of the system.

Main Functions:

- Transmit CAN frames with custom data (in this case, the message "HELLO").
- Receive CAN frames from the bus.
- Signal transmission and reception interrupts.
- Work together with a UART controller to display received data.



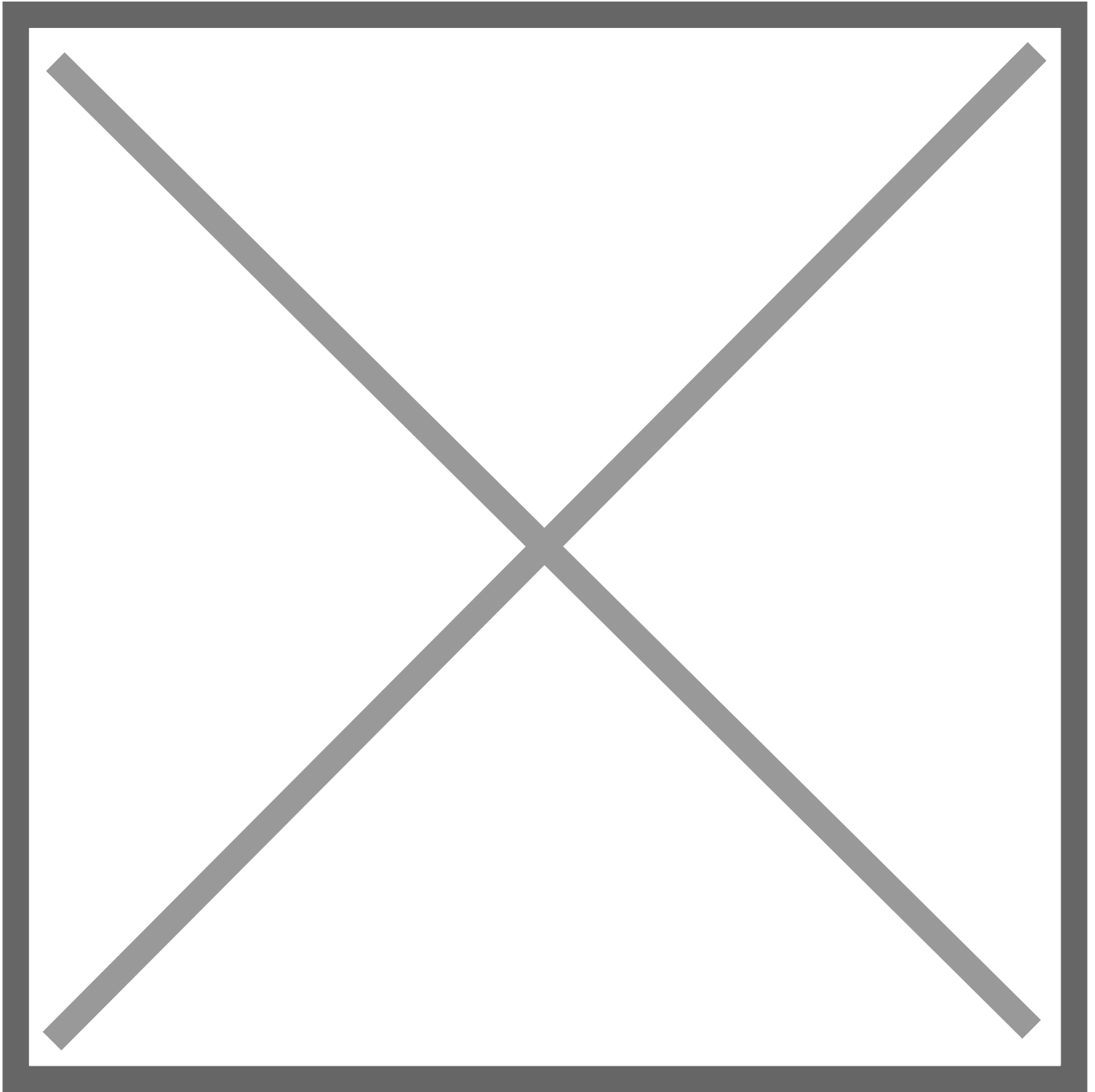
2.2 CAN Controller Block Inputs

To function properly, the `can_controller_std` requires different control, configuration, and data signals. Below, we organize these inputs into three categories: main inputs, configuration parameters, and data for the message to be transmitted.

a) Main Inputs

These inputs control the basic operation and synchronization of the block.

Input	Source	Description
clk_i	Global system clock	Synchronizes all internal operations of the CAN controller.
rst_i	InvertedC (reset)	Active-low reset that initializes the CAN controller when necessary.
rx_i	Physical IO70 pin	Receives data from the CAN bus (CAN RX).
tx_request_i	one_hz_clock (1 Hz pulse)	Requests the CAN controller to transmit a frame every second.
clear_irq_i	InvertedC from key button	Manually clears transmission interrupts, allowing new transmissions.

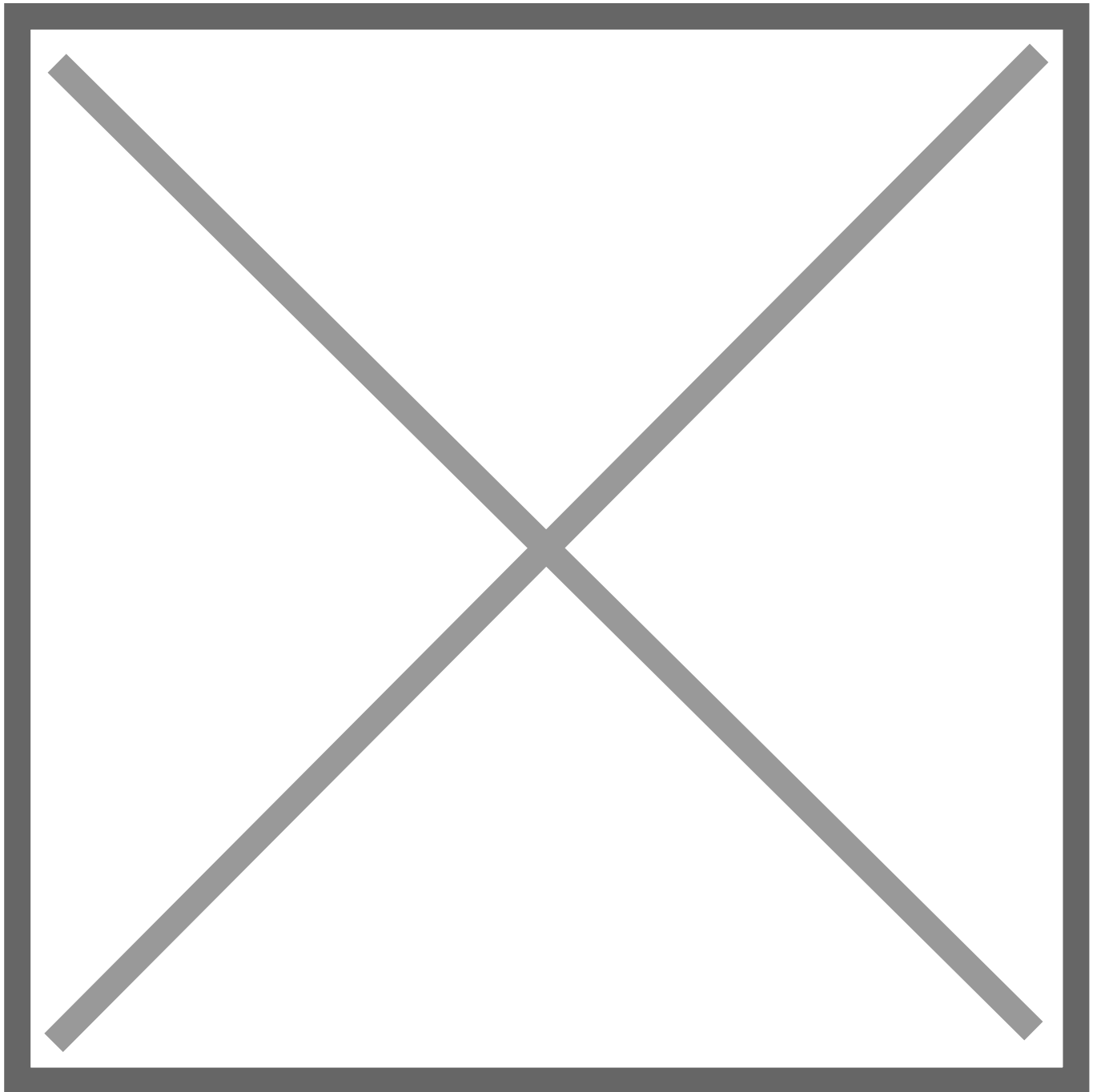


b) Configuration Parameters

These signals configure the bit timing and message filtering of the CAN protocol. They determine the transmission speed and which messages are accepted on the bus.

Parameter	Value	Description
tx_data_0_i	8'd72	H (first byte of the "HELLO" message)

Parameter	Value	Description
time_segment_1_i	4'd15	Defines the first segment of the bit time in the CAN protocol. It controls part of the CAN communication timing, assisting with signal synchronization.
baud_r_presc_i	3'd2	Prescaler for the CAN protocol clock. Defines the timing base which, together with the time segments, determines the CAN transmission speed. Example: 125 Kbps.
sync_jump_width_i	1'b0	Defines the Synchronization Jump Width (SJW). With 0, there is no margin for bit time adjustments in case of small variations.
time_segment_2_i	2'b01	Second time segment in the CAN protocol. Controls the final part of the bit time, important for completing transmission synchronization.
acceptance_code_i	11'h000	Acceptance code for filtering received CAN messages. A value of 000 means any message ID will be accepted (no filtering).
acceptance_mask_i	11'h000	Acceptance mask that defines which bits of acceptance_code_i are relevant. With 000, filtering is disabled and all messages are accepted.

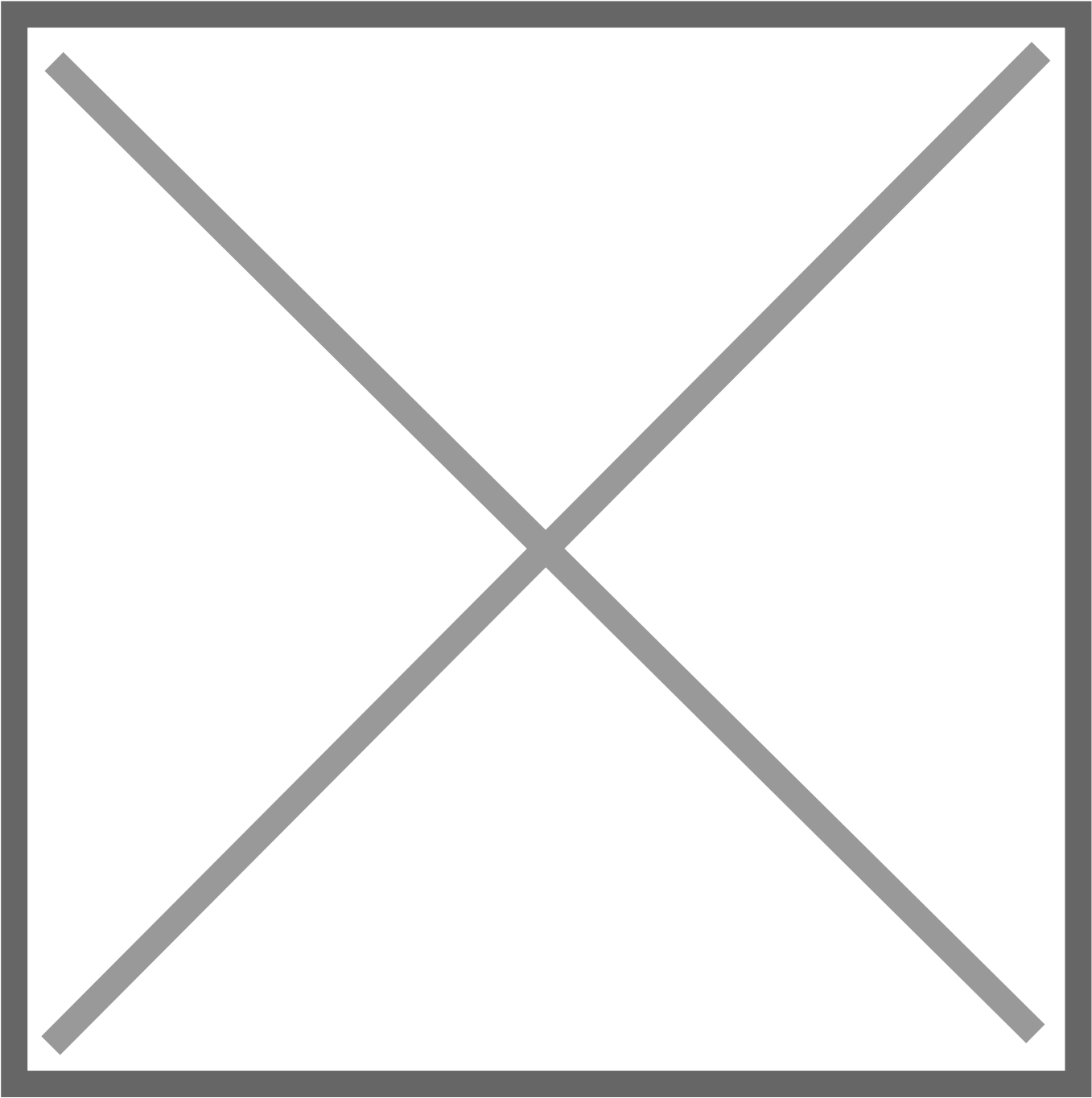


c) Data to Be Transmitted

These signals are responsible for loading the information that composes the message transmitted on the CAN bus. In this project, we want to send the string "HELLO", consisting of 5 characters.

Signal	Value	Description
rtr_i	1'b0	Mensagem de dados normal.
packet_id_i	11'h257	ID da mensagem no barramento CAN.
packet_length_i	4'd5	"HELLO" tem 5 caracteres ➡ 5 bytes.

Signal	Value	Description
tx_data_0_i	72	Letter H (ASCII 72).
tx_data_1_i	69	Letter E (ASCII 69).
tx_data_2_i	76	Letter L (ASCII 76).
tx_data_3_i	76	Letter L (ASCII 76).
tx_data_4_i	79	Letter O (ASCII 79).



2.3 System Outputs

a) Outputs of the can_controller_std Block

The outputs of the can_controller_std block are responsible for:

- Transmitting CAN frames on the bus.
- Indicating the status of operations (message transmission and reception).
- Providing received data to other system modules, such as the UART debug interface.

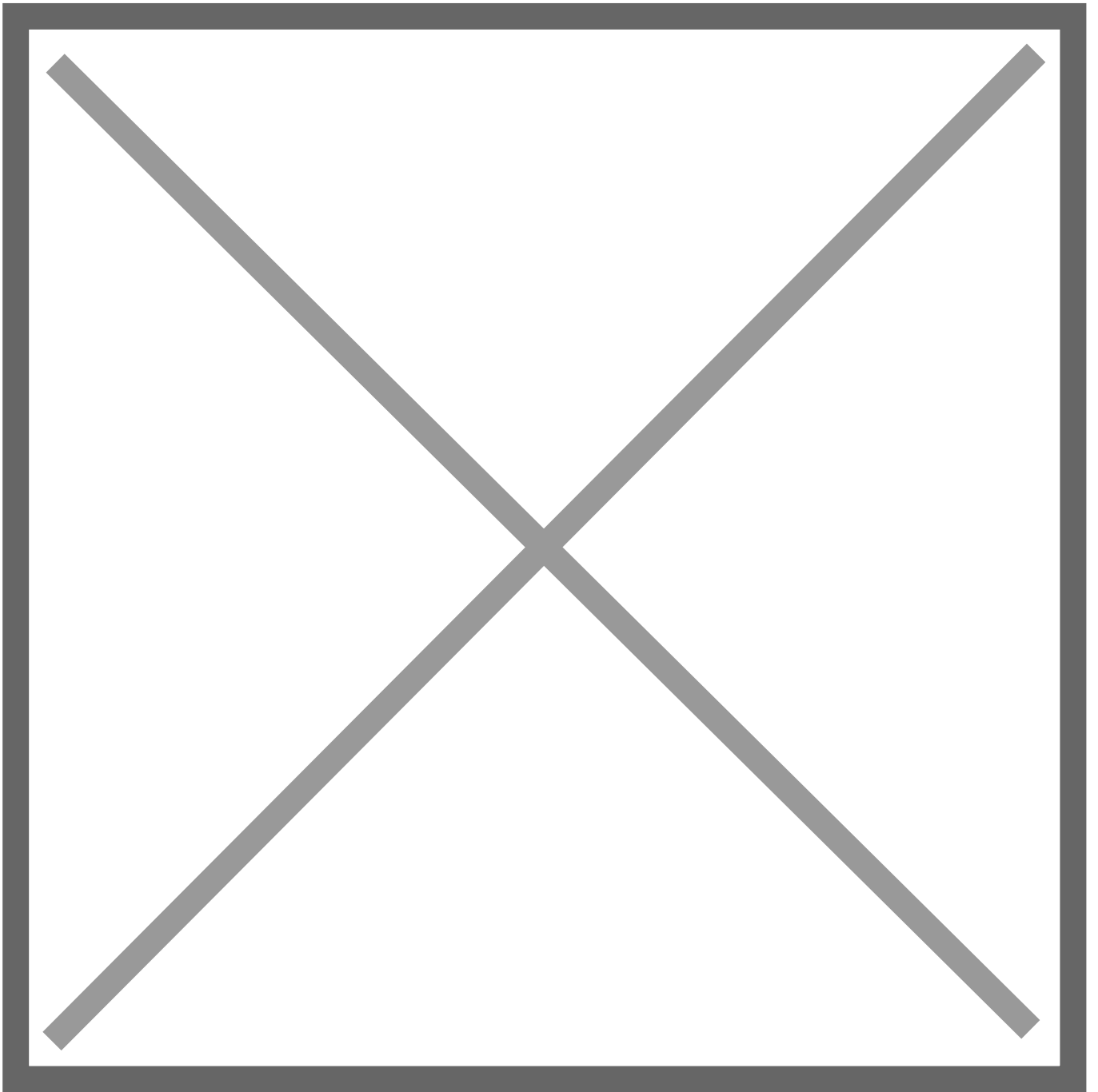
These outputs enable both physical communication via CAN and system monitoring through UART and LEDs.

Output	Destination	Function
tx_o	IO71	Transmits the CAN frame on the bus.
transmit_irq_o	LED0	Indicates that the transmission has started.
transmit_ack_irq_o	LED1	Confirms that the transmission was acknowledged (ACK).
receive_irq_o	uart_can_std_printer	Signals that a CAN frame has been received.

b) Outputs of the uart_can_std_printer Block

In addition to the direct outputs of the can_controller_std, there are components that process this information and provide user feedback through LEDs and UART.

Output	Destination	Function
uartTxPin	uartTx	Sends the received message via UART (ASCII).
txDone	LED5	Indicates that the UART transmission is complete.



And at the end, your project should be something like this:



Revision #1

Created 21 March 2025 12:28:00 by Caroline

Updated 21 March 2025 12:35:27 by Caroline