STM32 Microcontroller Code Flashing via CAN Communication Protocol

Introduction

The aim of this project is to develop a robust system that enables the flashing of code between two STM32 microcontrollers using the CAN (Controller Area Network) communication protocol. This system comprises two key components: the Receiver and the Transceiver, each serving a specific role in achieving the desired functionality.

Project Components

Receiver

Initialization and Bootloader

The Receiver component begins by initializing the CAN communication system and enabling CAN interrupts. It subsequently calls the bootloader, which is responsible for the following tasks:

1. Checking the Option Byte (Data 1): The bootloader checks the Option Byte (Data 1) to determine which application was last in use.

2. Application Selection:

    - If the Option Byte (Data 1) indicates that the last active application was either the bootloader or Application 2, the system prepares the memory location for Application 1. This involves cleaning the memory area previously occupied by any data to ensure a pristine environment for flashing Application 1.
    - In other cases where the Option Byte (Data 1) indicates that Application 1 was last active, the system prepares the memory location for Application 2 by erasing any existing data. This step ensures that Application 2 can be safely flashed without interference.

3. Waiting for CAN Transceiver: The Receiver component waits for the CAN transceiver to send the application code.

4. Data Reception: When the CAN transceiver sends data, it does so in 8-byte chunks. The Receiver receives these 8-byte data packets and starts flashing them into the cleaned memory space designated for the respective application.

5. Completion Flag: Upon receiving the CAN ID 3FF, which serves as an indication that all data has been successfully sent and received, the Receiver sets a flag to signal the completion of the code flashing process.

6. Updating Option Byte: Finally, the Option Byte is updated to specify which application should be launched upon a soft reset. This ensures that the bootloader knows which application to load after a system restart.

Transceiver

The Transceiver component plays a crucial role in this system, as it contains the application code and manages the communication with the Receiver. Its key functionalities include:

1. Initialization: The Transceiver initializes itself, preparing to send data.

2. Data Transmission: It sends data to the Receiver in 8-byte chunks, using CAN ID 300 for each packet. This ensures that the Receiver receives and flashes the code in manageable segments.

3. Completion Notification: The Transceiver sends a final message with CAN ID 3FF, indicating that all data has been successfully transmitted. This message signals to the Receiver that the code flashing process is complete.

Conclusion

In conclusion, this project successfully implements a code flashing system using two STM32 microcontrollers and the CAN communication protocol. The Receiver and Transceiver components work together to ensure that application code can be securely flashed between the devices. With the ability to handle soft resets and maintain application selection via Option Bytes, this system is both reliable and flexible. Future developments may include optimizations for code integrity and additional error handling mechanisms.