Sumant Kumar Jha

Designing a bootloader for MCU A and MCU B

1. Bootloader Initialization:

MCU A and MCU B should have a bootloader (BL).

MCU A BL interfaces with RS485 with PC. Whereas MCU B BL interfaces with SPI with MCU A.

Therefore, MCU A BL initializes for RS485 and MCU B BL initializes for SPI.

The bootloader in MCU A checks for an update request coming over RS485 and enters the update mode if requested.

2. Update Request Handling:

If an update request is received, it could be

only MCU A: MCU A switches to bootloader mode. Jump to step 3

MCU A + MCU B: MCU A switches to bootloader mode. Jump to step 3

only MCU B: MCU A sends signal to MCU B over SPI to switch to bootloader mode. Jump to step 5

3. Data Transfer to MCU A:

MCU A establishes RS485 communication for data transfer.

Data packets are sent from PC to MCU A containing the firmware update.

MCU A acknowledges the receipt of each packet.

and this process continues.

4. Firmware Update complete MCU A:

MCU A receives firmware packets, validates them, and updates the firmware.

Once the update (no of bytes flashed = totalSize) is complete, MCU A sends an acknowledgment to PC.

5. Preparation for MCU B firmware update

MCU A initializes SPI and check whether update request was received for MCU B also

6. Data Transfer to MCU B:

MCU A establishes SPI communication with MCU B for data transfer.

Data packets are sent from PC->MCU A over RS485 which has propagated further to MCU B over SPI containing the firmware update.

MCU B acknowledges the receipt of each packet to MCU A which in turn sends acknowledgement to PC over RS485.

and this process continues.

7. Firmware update complete MCU B:

MCU B receives firmware packets, validates them, and updates the firmware.

Once the update (no of bytes flashed = totalSize) is complete, MCU B sends an acknowledgment to MCU A which in turn sends acknowledgement to PC.

Design the data structure between MCU A and MCU B using C/C++

```c
// Define a structure for the communication protocol
struct CommunicationPacket {
    uint8_t command;          // Command to indicate the type of packet (e.g., firmware update, acknowledgment)
    uint16_t packetNumber;    // Packet number for tracking and sequencing
    uint16_t dataSize;        // Size of the data payload in bytes
    uint8_t data[SIZE_TO_BE_UPDATED];       // Data payload (adjust the size as needed)
    uint16_t checksum;        // Checksum for error checking
    uint32_t totalSize; // Total Size of the firmware update data in bytes
};

// Define commands for communication
enum Command {
    CMD_REQUEST_UPDATE = 1,   // MCU A requests MCU B to enter update mode
    CMD_ACK_UPDATE_REQUEST,   // MCU B acknowledges the update request
    CMD_DATA_PACKET,          // Data packet containing firmware update
    CMD_UPDATE_COMPLETE,      // MCU B acknowledges the completion of the firmware update
};
```