BÁO CÁO THỰC HÀNH BÀI 4

Môn học: **CHUYÊN ĐỀ THIẾT KẾ HỆ THỐNG NHÚNG 1**- Mã lớp: **CE437.N11**

Giảng viên hướng dẫn thực hành: Phạm Minh Quân

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Thông tin sinh viên** | |  |  | | --- | --- | | MSSV | Họ và tên | | 20520211 | Trương Hữu Khang | | 20520219 | Nguyễn Linh Anh Khoa | | 20520597 | Phan Duy Thông | |
| **Link các tài liệu tham khảo** *(nếu có)* |  |
| **Đánh giá của giảng viên**:  *+ Nhận xét*  *+ Các lỗi trong chương trình*  *+ Gợi ý* |  |

*[Báo cáo chi tiết các thao tác, quy trình sinh viên đã thực hiện trong quá trình làm bài thực hành. Chụp lại hình ảnh màn hình hoặc hình ảnh kết quả chạy trên sản phẩm. Mô tả và giải thích chương trình tương ứng để cho ra kết quả như hình ảnh đã trình bày. Sinh viên xuất ra file .pdf và đặt tên theo cấu trúc: MSSV\_HoTen\_Labx\_Report.pdf (Trong đó: MSSV là mã số sinh viên, HoTen là họ và tên, x trong Labx là chỉ số của bài thực hành tương ứng]*

**MỤC LỤC**

[Câu 1. Viết chương trình thực hiện dịch vụ $22H – Read Data by Identifier để đọc và hiện giá trị ADC qua UART 3](#_Toc120701586)

[Câu 2. Viết chương trình để thực hiện dịch vụ $2EH – Write Data by Identifier để ghi vị trí của Joy Stick xuống ECU và xuất dữ liệu ra UART: 4](#_Toc120701587)

1. Viết chương trình thực hiện dịch vụ $22H – Read Data by Identifier để đọc và hiện giá trị ADC qua UART

Phần chương trình của Tester để gửi yêu cầu $22H đến ECU

uint8\_t uds\_request[2]  = {0x22, 0xF1}; *// uds request package*

*// send request to ECU*

HAL\_CAN\_AddTxMessage(&hcan1, &Node1\_TxHeader, uds\_request, &Node1TxMsgMailBox);

Phần chương trình xử lí của ECU khi nhận được yêu cầu $22H

uint8\_t message[8]; *// store the message*

HAL\_CAN\_GetRxMessage(*hcan*, CAN\_RX\_FIFO1, &Node2\_RxHeader, message); *// get message from FIFO*

if (message[0] == 0x22) { *// check UDS request service ID*

    switch (message[1]) {

case 0xF1:

HAL\_ADC\_Start(&hadc1);

            uds\_response[2] = HAL\_ADC\_GetValue(&hadc1);

            HAL\_ADC\_Stop(&hadc1);

            break;

       default:

            break;

        }

uds\_response[0]    = message[0] + 0x40; *// positive reponse*

uds\_response[1]    = message[1];

Node2\_TxHeader.DLC = sizeof(uds\_response);

HAL\_CAN\_AddTxMessage(*hcan*, &Node2\_TxHeader, uds\_response, &Node2TxMsgMailBox); *// send response*

**Giải thích:**

uds\_request bao gồm 2 byte:

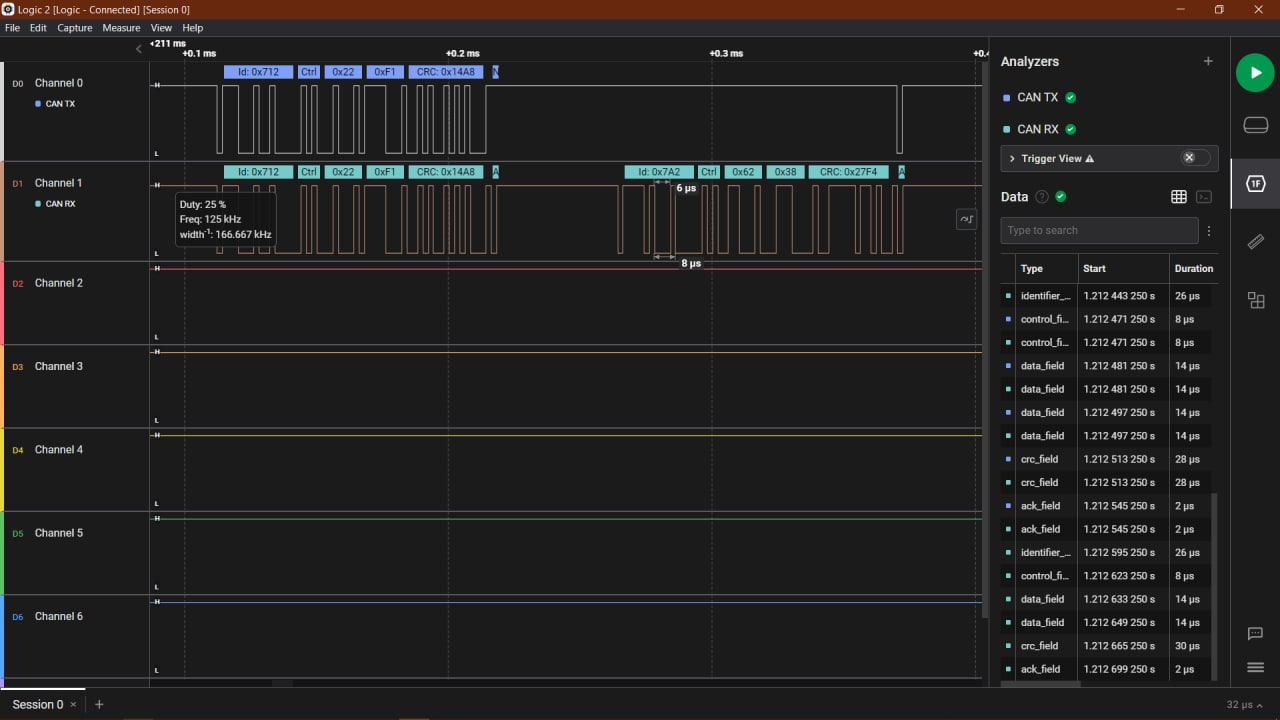
* Byte 1 là Service ID Read Data by Identifier
* Byte 2 là Data Identifier

Gói tin của UDS Protocol sẽ nằm trong phần Payload của một CAN Frame

Ở phía ECU sau khi nhận được gói tin sẽ kiểm tra Service ID của request và kiểm tra Data Identifier để tiến hành đọc dữ liệu

Sau khi đọc dữ liệu hoàn tất tiến hành reponse về Tester với positive reponse

**Tín hiệu đọc được trên đường dây:**



1. Viết chương trình để thực hiện dịch vụ $2EH – Write Data by Identifier để ghi vị trí của Joy Stick xuống ECU và xuất dữ liệu ra UART:

Phần chương trình của Tester

*// khai báo các mảng dữ liệu*

uint8\_t const middle[10] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

uint8\_t const left[10] = {0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA, 0xAA};

uint8\_t const right[10] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

uint8\_t const DataID[2]  = {0xF0, 0x02};

GPIO\_PinState a;

uint8\_t message1[8] = {};

for (;;) {

        HAL\_GPIO\_TogglePin(LD1\_GPIO\_Port, LD1\_Pin);

        a = HAL\_GPIO\_ReadPin(Button\_GPIO\_Port, Button\_Pin);

        if (a == GPIO\_PIN\_SET) {

*// First Frame*

            message1[0] = 0x10;

            message1[1] = 0x0A;

            message1[2] = 0x2E;

            message1[3] = DataID[0];

            message1[4] = DataID[1];

            for (int i = 5; i < 8; i++) {

                message1[i] = middle[i - 5];

            }

            HAL\_CAN\_AddTxMessage(&hcan1, &Node1\_TxHeader, message1, &Node1TxMsgMailBox);

            message1[0] = 0x21;

            for (int i = 1; i < 8; i++) {

                message1[i] = middle[i + 2];

            }

            HAL\_CAN\_AddTxMessage(&hcan1, &Node1\_TxHeader, message1, &Node1TxMsgMailBox);

        } else {

*// First Frame*

            message1[0] = 0x10;

            message1[1] = 0x0A;

            message1[2] = 0x2E;

            message1[3] = DataID[0];

            message1[4] = DataID[1];

            for (int i = 5; i < 8; i++) {

                message1[i] = right[i - 5];

            }

            HAL\_CAN\_AddTxMessage(&hcan1, &Node1\_TxHeader, message1, &Node1TxMsgMailBox);

            message1[0] = 0x21;

            for (int i = 1; i < 8; i++) {

                message1[i] = right[i + 2];

            }

            HAL\_CAN\_AddTxMessage(&hcan1, &Node1\_TxHeader, message1, &Node1TxMsgMailBox);

        }

        osEventFlagsSet(DoneTransferHandle, ACTIVATE\_FLAG);

        osDelay(1000);

        HAL\_CAN\_GetRxMessage(&hcan1, CAN\_RX\_FIFO0, &Node1\_RxHeader, message1);

        if (message1[0] == 0x2E + 0x40) {

            printf("*%s*\n", "Success");

        }

    }

Phần chương trình của ECU:

uint32\_t fifoLevel;

uint8\_t raw\_message[8] = {};

uint8\_t message2[30]   = {};

uint32\_t datalength;

uint8\_t UDS\_service\_ID;

uint8\_t DataID[2];

uint8\_t reponse[3];

int i;

*/\* Infinite loop \*/*

for (;;) {

osEventFlagsWait(DoneTransferHandle, ACTIVATE\_FLAG, osFlagsWaitAny, osWaitForever);

fifoLevel = HAL\_CAN\_GetRxFifoFillLevel(&hcan2, CAN\_RX\_FIFO1);

if (fifoLevel != 0) {

HAL\_GPIO\_TogglePin(LD2\_GPIO\_Port, LD2\_Pin);

HAL\_CAN\_GetRxMessage(&hcan2, CAN\_RX\_FIFO1, &Node2\_RxHeader, raw\_message);

if (raw\_message[0] & 0b11110000) {

datalength = raw\_message[0] & 0b00001111;

datalength = (datalength << 4) + raw\_message[1];

UDS\_service\_ID = raw\_message[2];

DataID[0]      = raw\_message[3];

DataID[1]      = raw\_message[4];

for (i = 0; i + 5 < 8; i++) {

message2[i] = raw\_message[i + 5];

}

for (i = i; i < datalength; i++) {

HAL\_CAN\_GetRxMessage(&hcan2, CAN\_RX\_FIFO1, &Node2\_RxHeader, raw\_message);

for (int x = 1; x < 8; x++) {

message2[i] = raw\_message[x];

i++;

if (i == datalength) break;

          }

    }

} else {

datalength = raw\_message[0] & 0b00001111;

for (i = 0; i < datalength; i++) {

message2[i] = raw\_message[i + 1];

        }

     }

fifoLevel = 0;

for (int x = 0; x < datalength; x++) {

printf("*%02X* ", message2[x]);

}

printf("\n");

Node2\_TxHeader.DLC = 3;

reponse[0]         = UDS\_service\_ID + 0x40;

reponse[1]         = DataID[0];

reponse[2]         = DataID[1];

HAL\_CAN\_AddTxMessage(&hcan2, &Node2\_TxHeader, reponse, &Node2TxMsgMailBox);

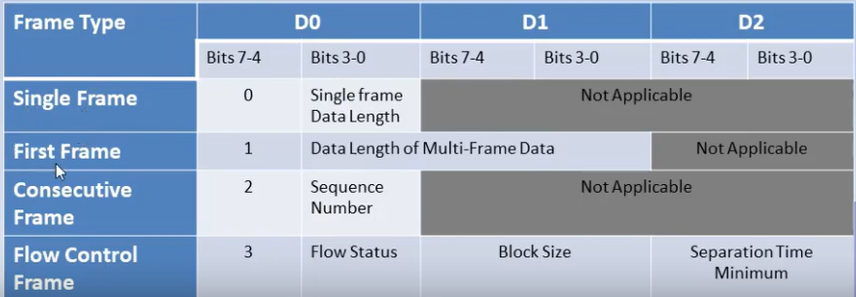
}

}

Giải thích:

Do chương trình sử dụng RTOS để giả lập hoạt động của hai thiết bị nên có sử dụng thêm biến cờ để đồng bộ việc gửi và nhận ở cả hai Thread.

Ở Task của Tester, task sẽ tiến hành đọc giá trị của Joy Stick, sau đó tiến hành quá trình tạo ra các gói tin (do dữ liệu vượt quá 8 byte nên không thể gửi được trên một CAN Frame). Các gói tin này sẽ tuân theo format:

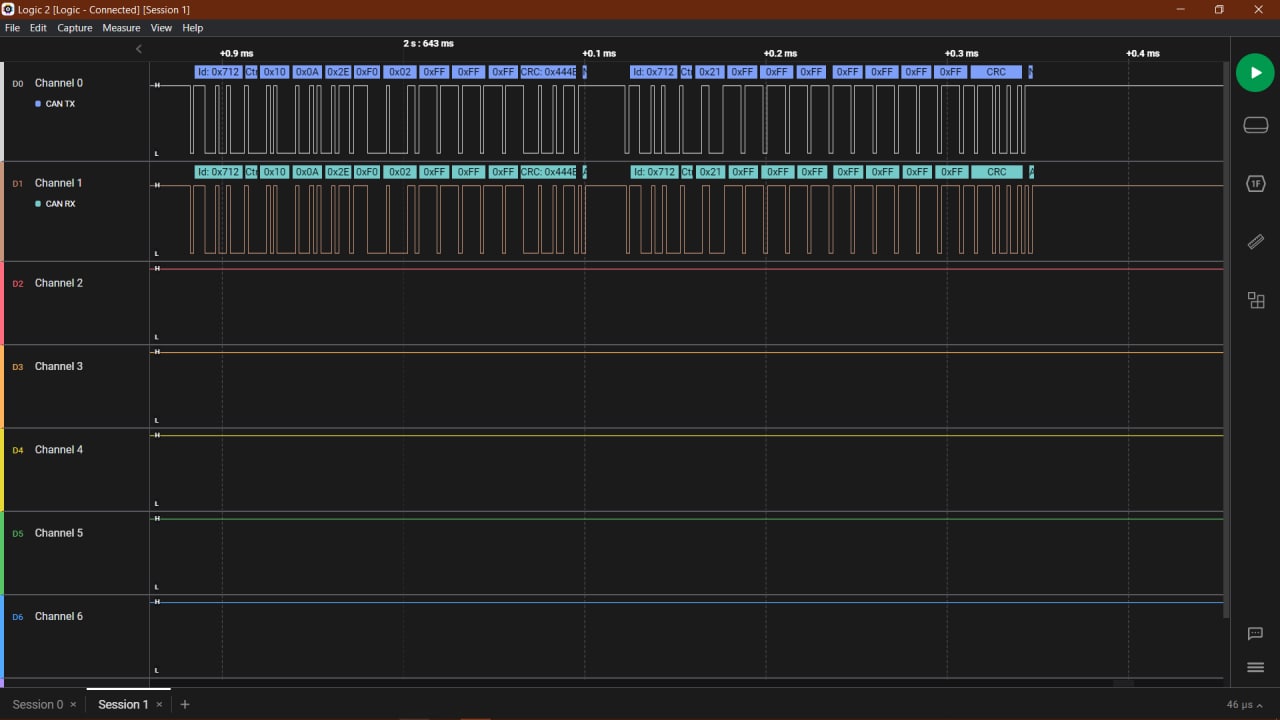


Hình 1. Format của các gói tin CAN

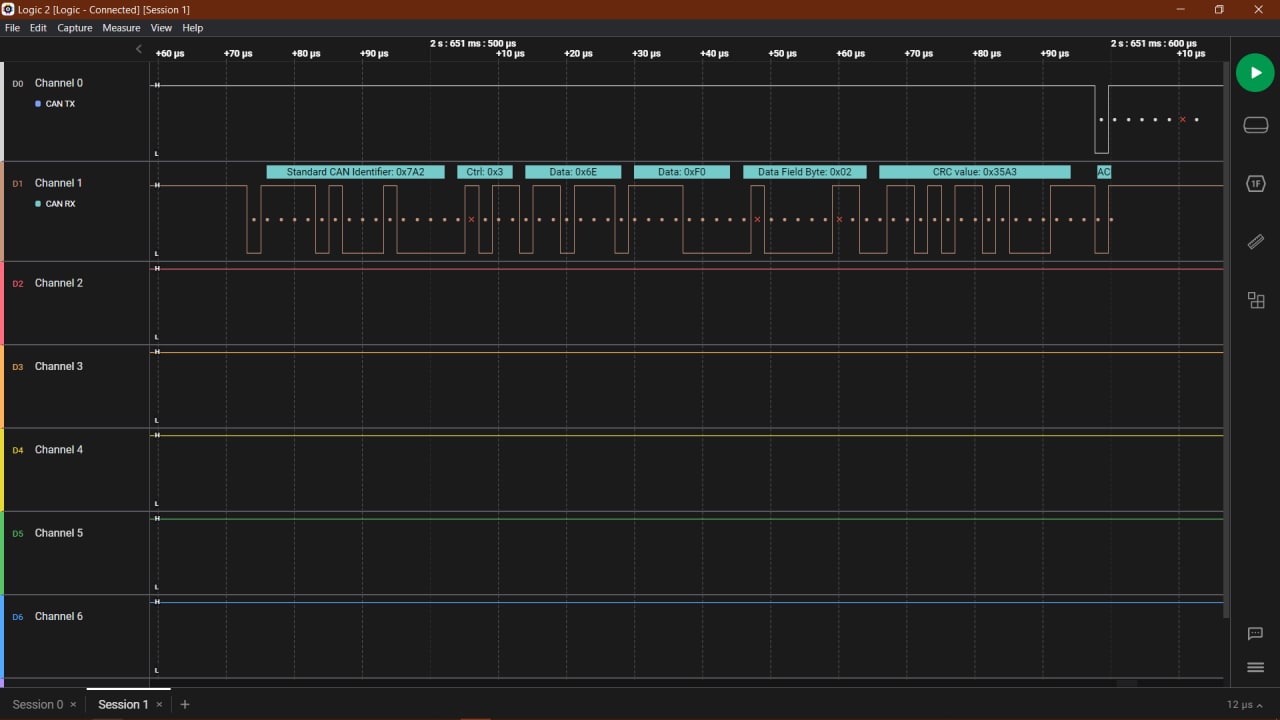
Do không yêu cầu việc phải cấu hình các giá trị cho lớp mạng nên gói tin Flow Control sẽ không được thực hiện.

Ở Task của ECU, sau khi nhận được cờ báo hiệu đã hoàn thành việc chuyển dữ liệu sẽ tiến hành ghép các gói tin lại với nhau. Các giá trị như DataLength, SequenceNumber sẽ được lưu lại để sử dụng cho quá trình ghép gói tin (việc lấy dữ liệu sẽ dựa vào format của gói tin có sẵn). Sau khi việc ghép gói tin hoàn tất ECU sẽ phản hồi về Tester với gói tin Positive Reponse. Tester sau khi nhận được gói tin Positive Reponse sẽ in “Success” ra UART

Các tín hiệu đọc được trên đường dây:



Hình 2. Các gói tin trong nhiều khung truyển CAN



Hình 3. Gói tin Positive Reponse do ECU gửi về cho Tester