

# BÁO CÁO THỰC HÀNH BÀI 3

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 20520211 20520219 20520597	Họ và tên Trương Hữu Khang Nguyễn Linh Anh Khoa 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)]

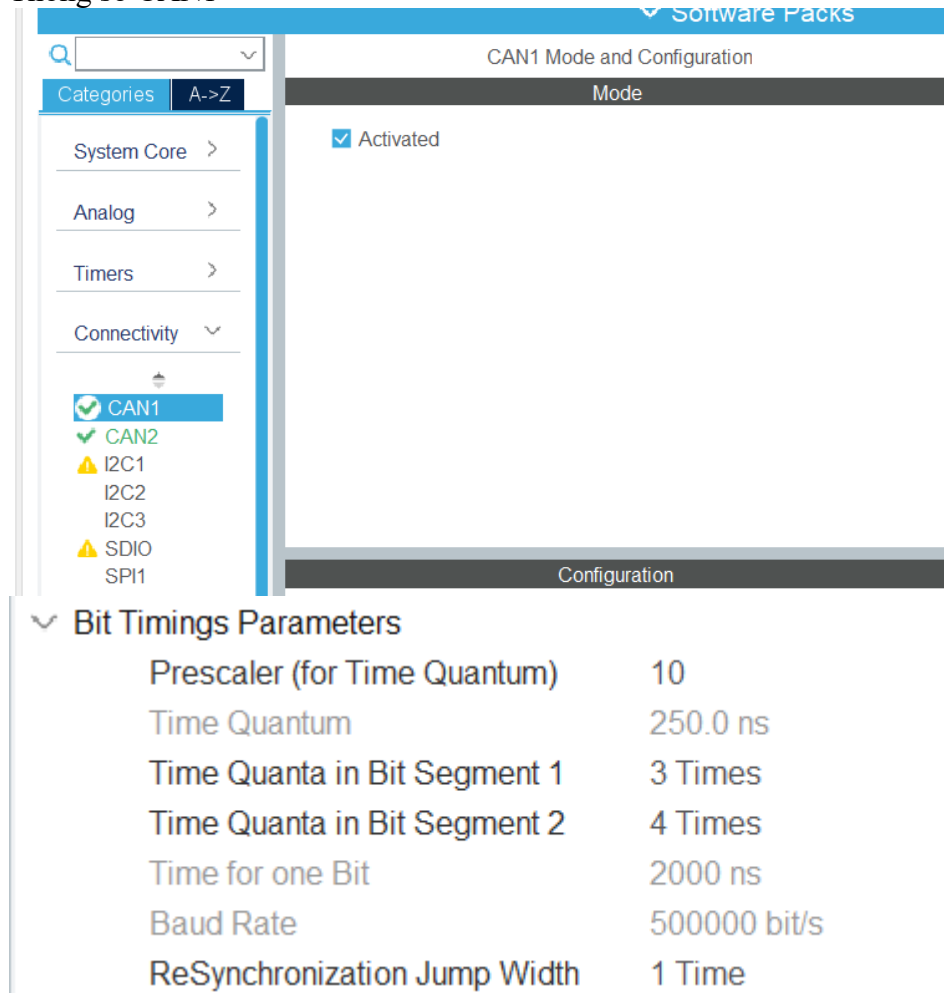
**Mục lục**

Câu 1.	Lập trình giao tiếp CAN đơn giản trên KIT: .....	3
Câu 2.	Viết chương trình để thực hiện lập trình giao tiếp giữa 2 module CAN1 và CAN2: .....	5
Câu 3.	Kiểm tra, theo dõi và nhận diện dữ liệu CAN trên đường truyền .....	9

**Câu 1. Lập trình giao tiếp CAN đơn giản trên KIT:**

Tại phần Connectivity

Thông số CAN1



**Software Packs**

Search:

Categories: A->Z

- System Core >
- Analog >
- Timers >
- Connectivity >
  - ☒ CAN1
  - ☒ CAN2
  - ☐ I2C1
  - ☐ I2C2
  - ☐ I2C3
  - ☐ SDIO
  - ☐ SPI1

**CAN1 Mode and Configuration**

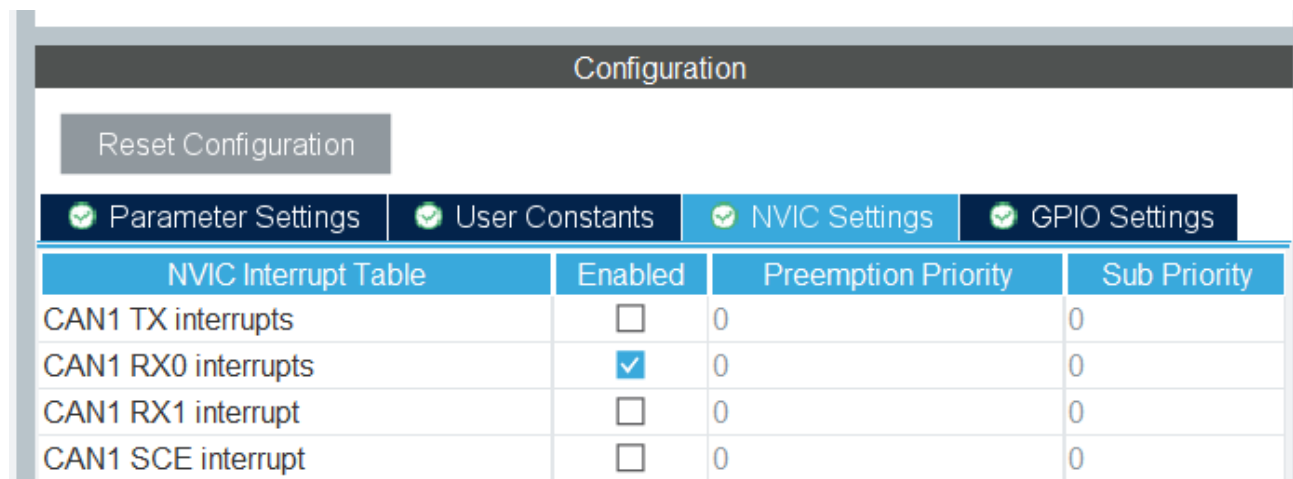
Mode

☒ Activated

Configuration

**Bit Timings Parameters**

Prescaler (for Time Quantum)	10
Time Quantum	250.0 ns
Time Quanta in Bit Segment 1	3 Times
Time Quanta in Bit Segment 2	4 Times
Time for one Bit	2000 ns
Baud Rate	500000 bit/s
ReSynchronization Jump Width	1 Time



**Configuration**

Reset Configuration

☒ Parameter Settings ☒ User Constants ☒ NVIC Settings ☒ GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
CAN1 TX interrupts	<input type="checkbox"/>	0	0
CAN1 RX0 interrupts	<input checked="" type="checkbox"/>	0	0
CAN1 RX1 interrupt	<input type="checkbox"/>	0	0
CAN1 SCE interrupt	<input type="checkbox"/>	0	0

Thông số CAN2

Bit Timings Parameters

Prescaler (for Time Quantum) 10  
Time Quantum 250.0 ns  
Time Quanta in Bit Segment 1 3 Times  
Time Quanta in Bit Segment 2 4 Times  
Time for one Bit 2000 ns  
Baud Rate 500000 bit/s  
ReSynchronization Jump Width. 2 Times

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
CAN2 TX interrupts	<input type="checkbox"/>	0	0
CAN2 RX0 interrupts	<input type="checkbox"/>	0	0
CAN2 RX1 interrupt	<input checked="" type="checkbox"/>	0	0
CAN2 SCE interrupt	<input type="checkbox"/>	0	0

**Câu 2. Viết chương trình để thực hiện lập trình giao tiếp giữa 2 module CAN1 và CAN2:**

Lập trình giao tiếp giữa 2 module CAN1 và CAN2

Khai báo kênh truyền và kênh nhận của từng Node

```
/* USER CODE BEGIN PV */
CAN_TxHeaderTypeDef Node1_TxHeader;
CAN_RxHeaderTypeDef Node1_RxHeader;

CAN_TxHeaderTypeDef Node2_TxHeader;
CAN_RxHeaderTypeDef Node2_RxHeader;
```

Nội dung gửi từ Node1 qua Node2 với nội dung “Hi CE437” với kích thước 8 byte và nội dung gửi từ Node2 qua Node1 với nội dung “Bye Bye!” với kích thước 8 byte.

```
uint32_t TxMsgMailBox;

uint8_t Node1_TX_BUFFER[8] = "Hi CE437";
uint8_t Node1_RX_BUFFER[8];
uint8_t Node2_TX_BUFFER[8] = "Bye Bye!";
uint8_t Node2_RX_BUFFER[8];
/* USER CODE END PV */
```

Hàm HAL\_CAN\_RxFifo0MsgPendingCallback là interrupt khi CAN nhận được dữ liệu vào FIFO0  
Hàm HAL\_CAN\_RxFifo1MsgPendingCallback là interrupt khi CAN nhận được dữ liệu vào FIFO1

```
/* USER CODE BEGIN 0 */
/* USER CODE BEGIN 0 */
void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
    HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &Node1_RxHeader, Node1_RX_BUFFER);
    if (Node1_RxHeader.DLC == 8) {
        printf("Node1 get data: %s\r\n", Node1_RX_BUFFER);
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
    }
}

void HAL_CAN_RxFifo1MsgPendingCallback(CAN_HandleTypeDef *hcan)
{
    HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);
    HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO1, &Node2_RxHeader, Node2_RX_BUFFER);
    if (Node2_RxHeader.DLC == 8) {
        printf("Node2 get data: %s\r\n", Node2_RX_BUFFER);
        HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_RESET);
    }
}
/* USER CODE END 0 */
```

Gán các interrupts

```
/* USER CODE BEGIN 2 */
HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING);
HAL_CAN_ActivateNotification(&hcan2, CAN_IT_RX_FIFO1_MSG_PENDING);
/* USER CODE END 2 */
```

Hàm thêm message vào Tx và cho phép yêu cầu truyền

```
/* USER CODE BEGIN WHILE */
while (1) {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_CAN_AddTxMessage(&hcan1, &Node1_TxHeader, Node1_TX_BUFFER,
                        &TxMsgMailBox);
    HAL_CAN_AddTxMessage(&hcan2, &Node2_TxHeader, Node2_TX_BUFFER,
                        &TxMsgMailBox);
}
/* USER CODE END 3 */
}
```

Ở hàm MX\_CAN1\_Init

Khai báo Header với Trường ID và kích thước 8 byte.

```
static void MX_CAN1_Init(void)
{
    /* USER CODE BEGIN CAN1_Init 0 */
    Node1_TxHeader.IDE = CAN_ID_STD;
    Node1_TxHeader.StdId = 0x555;
    Node1_TxHeader.RTR = CAN_RTR_DATA;
    Node1_TxHeader.DLC = 8;

    static void MX_CAN2_Init(void)
    {
        /* USER CODE BEGIN CAN2_Init 0 */
        Node2_TxHeader.IDE = CAN_ID_STD;
        Node2_TxHeader.StdId = 0x2AA;
        Node2_TxHeader.RTR = CAN_RTR_DATA;
        Node2_TxHeader.DLC = 8;
```

```
hcan1.Instance = CAN1;
hcan1.Init.Prescaler = 10;
hcan1.Init.Mode = CAN_MODE_NORMAL;
hcan1.Init.SyncJumpWidth = CAN_SJW_2TQ;
hcan1.Init.TimeSeg1 = CAN_BS1_3TQ;
hcan1.Init.TimeSeg2 = CAN_BS2_4TQ;
hcan1.Init.TimeTriggeredMode = DISABLE;
hcan1.Init.AutoBusOff = ENABLE;
hcan1.Init.AutoWakeUp = ENABLE;
hcan1.Init.AutoRetransmission = DISABLE;
hcan1.Init.ReceiveFifoLocked = DISABLE;
hcan1.Init.TransmitFifoPriority = DISABLE;
if (HAL_CAN_Init(&hcan1) != HAL_OK) {
    Error_Handler();
}
/* USER CODE BEGIN CAN1_Init 2 */
CAN_FilterTypeDef canfilterconfig;
canfilterconfig.FilterActivation = ENABLE;
canfilterconfig.FilterBank = 18;
canfilterconfig.FilterFIFOAssignment = CAN_FILTER_FIFO0;
canfilterconfig.FilterIdHigh = 0x555 << 5;
canfilterconfig.FilterIdLow = 0;
canfilterconfig.FilterMaskIdHigh = 0x555 << 5;
canfilterconfig.FilterMaskIdLow = 0;
canfilterconfig.FilterMode = CAN_FILTERMODE_IDMASK;
canfilterconfig.FilterScale = CAN_FILTERSCALE_32BIT;
canfilterconfig.SlaveStartFilterBank = 20;

HAL_CAN_ConfigFilter(&hcan1, &canfilterconfig);
```

```
hcan2.Instance = CAN2;
hcan2.Init.Prescaler = 10;
hcan2.Init.Mode = CAN_MODE_NORMAL;
hcan2.Init.SyncJumpWidth = CAN_SJW_2TQ;
hcan2.Init.TimeSeg1 = CAN_BS1_3TQ;
hcan2.Init.TimeSeg2 = CAN_BS2_4TQ;
hcan2.Init.TimeTriggeredMode = DISABLE;
hcan2.Init.AutoBusOff = ENABLE;
hcan2.Init.AutoWakeUp = ENABLE;
hcan2.Init.AutoRetransmission = DISABLE;
hcan2.Init.ReceiveFifoLocked = DISABLE;
hcan2.Init.TransmitFifoPriority = DISABLE;
if (HAL_CAN_Init(&hcan2) != HAL_OK) {
    Error_Handler();
}
/* USER CODE BEGIN CAN2_Init 2 */
CAN_FilterTypeDef canfilterconfig;
canfilterconfig.FilterActivation = ENABLE;
canfilterconfig.FilterBank = 10;
canfilterconfig.FilterFIFOAssignment = CAN_FILTER_FIFO1;
canfilterconfig.FilterIdHigh = 0x2AA << 5;
canfilterconfig.FilterIdLow = 0;
canfilterconfig.FilterMaskIdHigh = 0x2AA << 5;
canfilterconfig.FilterMaskIdLow = 0;
canfilterconfig.FilterMode = CAN_FILTERMODE_IDMASK;
canfilterconfig.FilterScale = CAN_FILTERSCALE_32BIT;
canfilterconfig.SlaveStartFilterBank = 0;

HAL_CAN_ConfigFilter(&hcan2, &canfilterconfig);
```



Câu 3. Kiểm tra, theo dõi và nhận diện dữ liệu CAN trên đường truyền

