I want to try transmitt voice data over CAN BUS line between two STM32F373R8 and I found this tutorial on youtube: [Tutorial Cube MX -13- CAN on STM32F4 Discovery - YouTube]

So I made a simple example of it. I tried to send one byte data from one board to another.

This is the sender's code:

main.c file:

```
/* Private variables ---------------------------------------------------------*/
CAN_HandleTypeDef hcan;
CanTxMsgTypeDef TxM;
CanRxMsgTypeDef RxM;
CAN_FilterConfTypeDef sFilterConfig;
/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
uint8_t i, RxData;
int tester_1=0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CAN_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
/*
void CAN_TX(uint8_t data)
{
//
CAN->sTxMailBox[0].TDLR = data;
CAN->sTxMailBox[0].TIR |= 1;
}
uint8_t CAN_RX(void)
{
//
while(!CAN->RF0R&3);
uint8_t RxD = (CAN->sFIFOMailBox[0].RDLR)&0xFF;
CAN->RF0R |= 1<<5;
return RxD;
}
*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

int main(void)
{

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration----------------------------------------------------------*/
```

```
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_CAN_Init();

/* USER CODE BEGIN 2 */
hcan.pTxMsg=&TxM;
hcan.pRxMsg=&RxM;
sFilterConfig.FilterNumber = 0;
sFilterConfig.FilterMode = CAN_FILTERMODE_IDLIST;
sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
sFilterConfig.FilterIdHigh = 0x245<<5; //Receive only 0x245
//sFilterConfig.FilterIdHigh = 0x244<<5;
sFilterConfig.FilterIdLow = 0;
sFilterConfig.FilterMaskIdHigh = 0;
sFilterConfig.FilterMaskIdLow = 0;
sFilterConfig.FilterFIFOAssignment = 0;
sFilterConfig.FilterActivation = ENABLE;
sFilterConfig.BankNumber = 14;
HAL_CAN_ConfigFilter(&hcan, &sFilterConfig);
HAL_CAN_Receive_IT(&hcan, CAN_FIFO0);
hcan.pTxMsg->StdId = 0x244; // Set STID 0x244
//hcan.pTxMsg->StdId = 0x245;
hcan.pTxMsg->RTR = CAN_RTR_DATA;
hcan.pTxMsg->IDE = CAN_ID_STD;
hcan.pTxMsg->DLC = 1;

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
hcan.pTxMsg->Data[0] = 0x01;
HAL_CAN_Transmit(&hcan, 1);
HAL_Delay(5);
tester_1++;
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_0);
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */

}
```

```
/** System Clock Configuration
*/
void SystemClock_Config(void)
{

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Initializes the CPU, AHB and APB busses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure the Systick interrupt time
*/
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the Systick
*/
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* CAN init function */
static void MX_CAN_Init(void)
{

hcan.Instance = CAN;
hcan.Init.Prescaler = 16;
hcan.Init.Mode = CAN_MODE_NORMAL;
hcan.Init.SJW = CAN_SJW_1TQ;
hcan.Init.BS1 = CAN_BS1_3TQ;
hcan.Init.BS2 = CAN_BS2_5TQ;
hcan.Init.TTCM = DISABLE;
hcan.Init.ABOM = DISABLE;
hcan.Init.AWUM = DISABLE;
hcan.Init.NART = DISABLE;
```

```
hcan.Init.RFLM = DISABLE;
hcan.Init.TXFP = DISABLE;
if (HAL_CAN_Init(&hcan) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/** Configure pins as
* Analog
* Input
* Output
* EVENT_OUT
* EXTI
*/
static void MX_GPIO_Init(void)
{

GPIO_InitTypeDef GPIO_InitStruct;

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOF_CLK_ENABLE();
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, LED1_Pin|LED2_Pin|LED3_Pin|LED4_Pin
|LED5_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : LED1_Pin LED2_Pin LED3_Pin LED4_Pin
LED5_Pin */
GPIO_InitStruct.Pin = LED1_Pin|LED2_Pin|LED3_Pin|LED4_Pin
|LED5_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @param None
* @retval None
*/
void _Error_Handler(char * file, int line)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the HAL error return state */
while(1)
{
}
/* USER CODE END Error_Handler_Debug */
}
```

```
#ifdef USE_FULL_ASSERT

/**
* @brief Reports the name of the source file and the source line number
* where the assert_param error has occurred.
* @param file: pointer to the source file name
* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t* file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and line number,
ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */

}

#endif

/**
* @}
*/

/**
* @}
*/

/************************ (C) COPYRIGHT STMicroelectronics *****END OF
FILE****/
```

Interruprt Handler:

```
void CAN_RX0_IRQHandler(void)
{
/* USER CODE BEGIN CAN_RX0_IRQn 0 */

/* USER CODE END CAN_RX0_IRQn 0 */
HAL_CAN_IRQHandler(&hcan);
/* USER CODE BEGIN CAN_RX0_IRQn 1 */
tester_1++;
HAL_CAN_Receive_IT(&hcan, CAN_FIFO0);
RxData = hcan.pRxMsg->Data[0];
if(hcan.pRxMsg->Data[0] == 0x01)
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
else if(hcan.pRxMsg->Data[0] == 0x02)
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_2);
else
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_4);
/* USER CODE END CAN_RX0_IRQn 1 */
}
```

So here I set STID 0x244 and fset filter to receive only with STID: 0x245 messages. (I believe). I set data low:

hcan.pTxMsg->Data[0] = 0x01 so on the receiver side I can notify that I receiver received 0x01 and toggle LED1

(in interrupt handler)

```
if(hcan.pRxMsg->Data[0] == 0x01)
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
```

I changed only filter, ID while loop in receivers code:

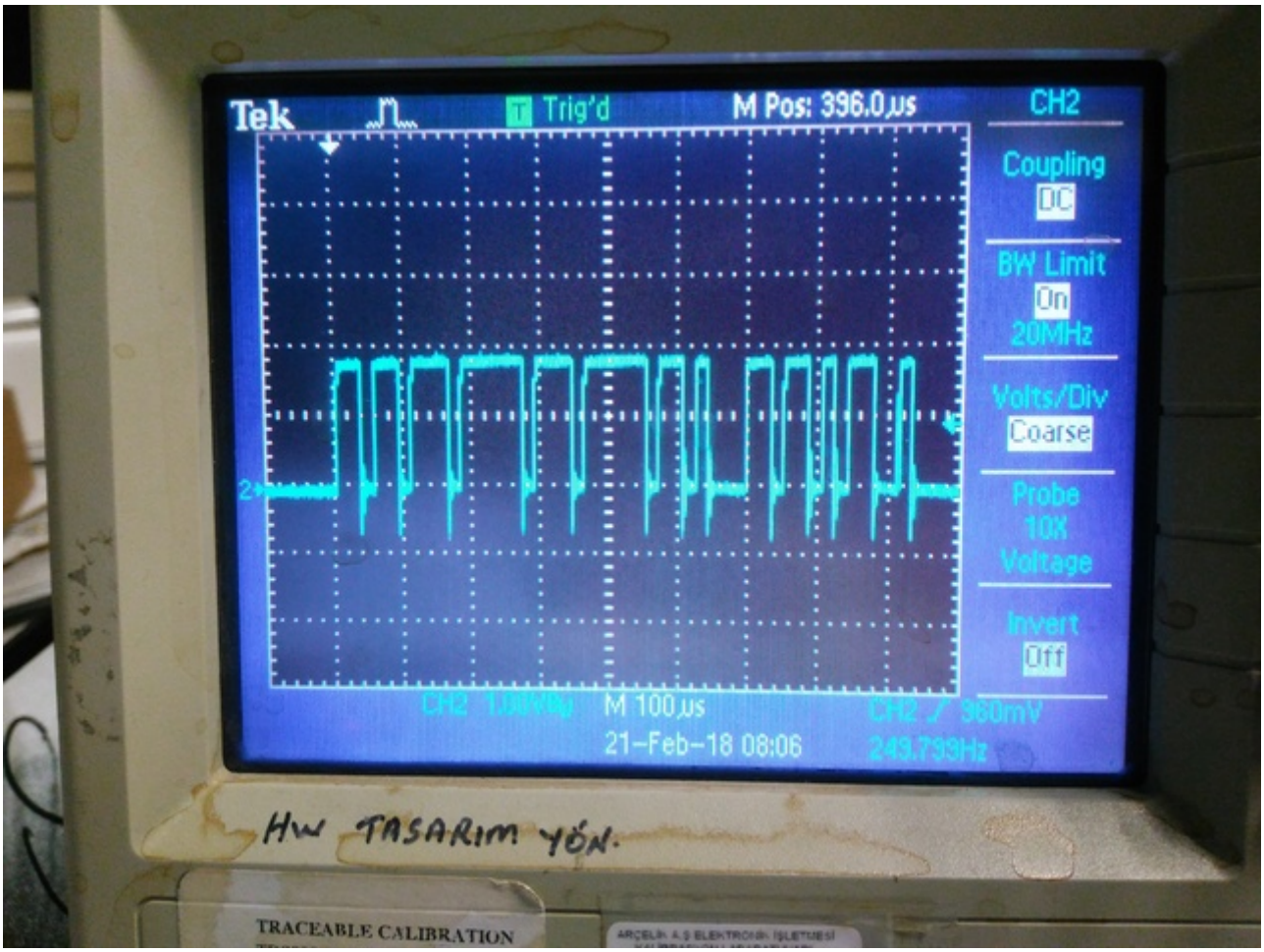
```
hcan.pTxMsg->StdId = 0x245; // instead of 0x244

sFilterConfig.FilterIdHigh = 0x244<<5; // instead of 0x245

{
tester_1++;

HAL_Delay(5);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_0);
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
```

However, this doesn't work. Eventhough I put

HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_0);

in endless while, I can't see LED0 toggling. (It worked in simple GPIO application from earlier so I know there is no hardware problem)

I see that Transmitter board initiates the CANBUS Message as below:



But there is no change on LED's states. In debugging mode (connected to receiver) when I stop the execution, I see that it stuck in HardFault_Handler.

Can direct me to find out where am I doing worng? Thanks in advance.