I'm trying to send voice data over CANBUS. I use a microphone to receive voice and get the data and send it over CANBUS (8 Byte data) to other block, synthesize the data in DAC again and output to the speaker. I know and verified that the mic, ADC, DAC and CAN works properly how ever I think somewhere in the timings I messed up.

Here is the senders code:

main.c

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f3xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

CAN_HandleTypeDef hcan;

TIM_HandleTypeDef htim6;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
CanTxMsgTypeDef TxM;
CanRxMsgTypeDef RxM;
CAN_FilterConfTypeDef sFilterConfig;
uint8_t i, RxData;
int tester_1=0;
_Bool conv_Flag;
uint8_t Tx_Audio[8];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_CAN_Init(void);
static void MX_TIM6_Init(void);
static void MX_ADC1_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
* @brief The application entry point.
*
* @retval None
```

```
*/
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_DMA_Init();
MX_CAN_Init();
MX_TIM6_Init();
MX_ADC1_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 // TX
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); // RX
hcan.pTxMsg->StdId = 0x244; // Set STID 0x244 // TX
hcan.pTxMsg->RTR = CAN_RTR_DATA;
hcan.pTxMsg->IDE = CAN_ID_STD;
hcan.pTxMsg->DLC = 8;

//HAL_GPIO_WritePin(GPIOA,GPIO_PIN_2,GPIO_PIN_RESET);
HAL_TIM_Base_Start(&htim6);
HAL_ADC_Start(&hadc1);
HAL_ADC_Start_DMA(&hadc1,(uint32_t*) Tx_Audio, 8);
//HAL_DAC_Start(&hdac1,DAC_CHANNEL_1);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
```

```
while (1)
{

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */

}
/* USER CODE END 3 */

}

/**
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_PeriphCLKInitTypeDef PeriphClkInit;

/**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__);
}

PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_ADC1;
PeriphClkInit.Adc1ClockSelection = RCC_ADC1PCLK2_DIV2;

if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != 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);
}

/* ADC1 init function */
static void MX_ADC1_Init(void)
{

ADC_ChannelConfTypeDef sConfig;

/**Common config
*/
hadc1.Instance = ADC1;
hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
hadc1.Init.ContinuousConvMode = ENABLE;
hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc1.Init.NbrOfConversion = 1;
if (HAL_ADC_Init(&hadc1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**Configure Regular Channel
*/
sConfig.Channel = ADC_CHANNEL_10;
sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SamplingTime = ADC_SAMPLETIME_7CYCLES_5;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

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

hcan.Instance = CAN;
hcan.Init.Prescaler = 1;
hcan.Init.Mode = CAN_MODE_NORMAL;
hcan.Init.SJW = CAN_SJW_1TQ;
hcan.Init.BS1 = CAN_BS1_7TQ;
hcan.Init.BS2 = CAN_BS2_1TQ;
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__);
}

}

/* TIM6 init function */
static void MX_TIM6_Init(void)
{

TIM_MasterConfigTypeDef sMasterConfig;

htim6.Instance = TIM6;
htim6.Init.Prescaler = 16;
htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
htim6.Init.Period = 4095;
htim6.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim6, &sMasterConfig) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/**
* Enable DMA controller clock
*/
static void MX_DMA_Init(void)
{
/* DMA controller clock enable */
__HAL_RCC_DMA1_CLK_ENABLE();

/* DMA interrupt init */
/* DMA1_Channel1_IRQn interrupt configuration */
HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

}

/** 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();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8,
GPIO_PIN_RESET);

/*Configure GPIO pins : PC1 PC6 PC7 PC8 */
GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7|GPIO_PIN_8;
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);

}
```

this is the interrupt handler:

```
void DMA1_Channel1_IRQHandler(void)
{
/* USER CODE BEGIN DMA1_Channel1_IRQn 0 */

/* USER CODE END DMA1_Channel1_IRQn 0 */
HAL_DMA_IRQHandler(&hdma_adc1);
/* USER CODE BEGIN DMA1_Channel1_IRQn 1 */
tester_1++;
for(i = 0; i < 8; i++)
{
//
//hcan.pTxMsg->Data[i] = Tx_Audio[i];
hcan.pTxMsg->Data[i] = (uint8_t) i; // for testing
}
HAL_CAN_Transmit(&hcan, 2);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_3);
/* USER CODE END DMA1_Channel1_IRQn 1 */
}
```

And here is the receiver side:

main.c

```
#include "main.h"
#include "stm32f3xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
CAN_HandleTypeDef hcan;
DAC_HandleTypeDef hdac1;
DMA_HandleTypeDef hdma_dac1_ch1;
```

```
TIM_HandleTypeDef htim6;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
CanTxMsgTypeDef TxM;
CanRxMsgTypeDef RxM;
CAN_FilterConfTypeDef sFilterConfig;
uint8_t i, RxData;
int tester_1=0;
const uint16_t voiceFreq[32] = {0x800,0x98f,0xb0f,0xc71,0xda7,
0xea6,0xf63,0xfd8,0xfff,0xfd8,0xf63,0xea6,0xda7,0xc71,
0xb0f,0x98f,0x800,0x670,0x4f0,0x38e,0x258,0x159,0x9c,0x27,
0x0,0x27,0x9c,0x159,0x258,0x38e,0x4f0,0x670};
uint8_t Rx_Audio[8];

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_CAN_Init(void);
static void MX_DAC1_Init(void);
static void MX_TIM6_Init(void);
static void MX_NVIC_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
* @brief The application entry point.
*
* @retval None
*/
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_DMA_Init();
MX_CAN_Init();
MX_DAC1_Init();
MX_TIM6_Init();

/* Initialize interrupts */
MX_NVIC_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 // TX
sFilterConfig.FilterIdHigh = 0x244<<5; //Receive only 0x244 // RX
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); // RX
//hcan.pTxMsg->StdId = 0x244; // Set STID 0x244 // TX
hcan.pTxMsg->StdId = 0x245; // Set STID 0x245 // RX
hcan.pTxMsg->RTR = CAN_RTR_DATA;
hcan.pTxMsg->IDE = CAN_ID_STD;
hcan.pTxMsg->DLC = 1;

HAL_GPIO_WritePin(GPIOA,GPIO_PIN_2,GPIO_PIN_RESET);
HAL_TIM_Base_Start(&htim6);
HAL_DAC_Start(&hdac1,DAC_CHANNEL_1);
/* USER CODE END 2 */

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

/* USER CODE BEGIN 3 */

}

/* USER CODE END 3 */

}

/**
* @brief System Clock Configuration
* @retval None
*/
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);
}

/**
* @brief NVIC Configuration.
* @retval None
*/
static void MX_NVIC_Init(void)
{
/* CAN_RX0_IRQn interrupt configuration */
HAL_NVIC_SetPriority(CAN_RX0_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(CAN_RX0_IRQn);
}

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

hcan.Instance = CAN;
hcan.Init.Prescaler = 1;
hcan.Init.Mode = CAN_MODE_NORMAL;
hcan.Init.SJW = CAN_SJW_1TQ;
```

```
hcan.Init.BS1 = CAN_BS1_7TQ;
hcan.Init.BS2 = CAN_BS2_1TQ;
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__);
}

}

/* DAC1 init function */
static void MX_DAC1_Init(void)
{

DAC_ChannelConfTypeDef sConfig;

/**DAC Initialization
*/
hdac1.Instance = DAC1;
if (HAL_DAC_Init(&hdac1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

/**DAC channel OUT1 config
*/
sConfig.DAC_Trigger = DAC_TRIGGER_T6_TRGO;
sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
if (HAL_DAC_ConfigChannel(&hdac1, &sConfig, DAC_CHANNEL_1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/* TIM6 init function */
static void MX_TIM6_Init(void)
{

TIM_MasterConfigTypeDef sMasterConfig;

htim6.Instance = TIM6;
htim6.Init.Prescaler = 16;
htim6.Init.CounterMode = TIM_COUNTERMODE_UP;
htim6.Init.Period = 4095;
htim6.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
if (HAL_TIM_Base_Init(&htim6) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

sMasterConfig.MasterOutputTrigger = TIM_TRGO_UPDATE;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim6, &sMasterConfig) != HAL_OK)
```

```
{
_Error_Handler(__FILE__, __LINE__);
}

}

/**
* Enable DMA controller clock
*/
static void MX_DMA_Init(void)
{
/* DMA controller clock enable */
__HAL_RCC_DMA1_CLK_ENABLE();

/* DMA interrupt init */
/* DMA1_Channel3_IRQn interrupt configuration */
HAL_NVIC_SetPriority(DMA1_Channel3_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(DMA1_Channel3_IRQn);

}

/** 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();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7
|GPIO_PIN_8, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_2, GPIO_PIN_RESET);

/*Configure GPIO pins : PC0 PC1 PC6 PC7
PC8 */
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_6|GPIO_PIN_7
|GPIO_PIN_8;
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);

/*Configure GPIO pin : PA2 */
GPIO_InitStruct.Pin = GPIO_PIN_2;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}
```

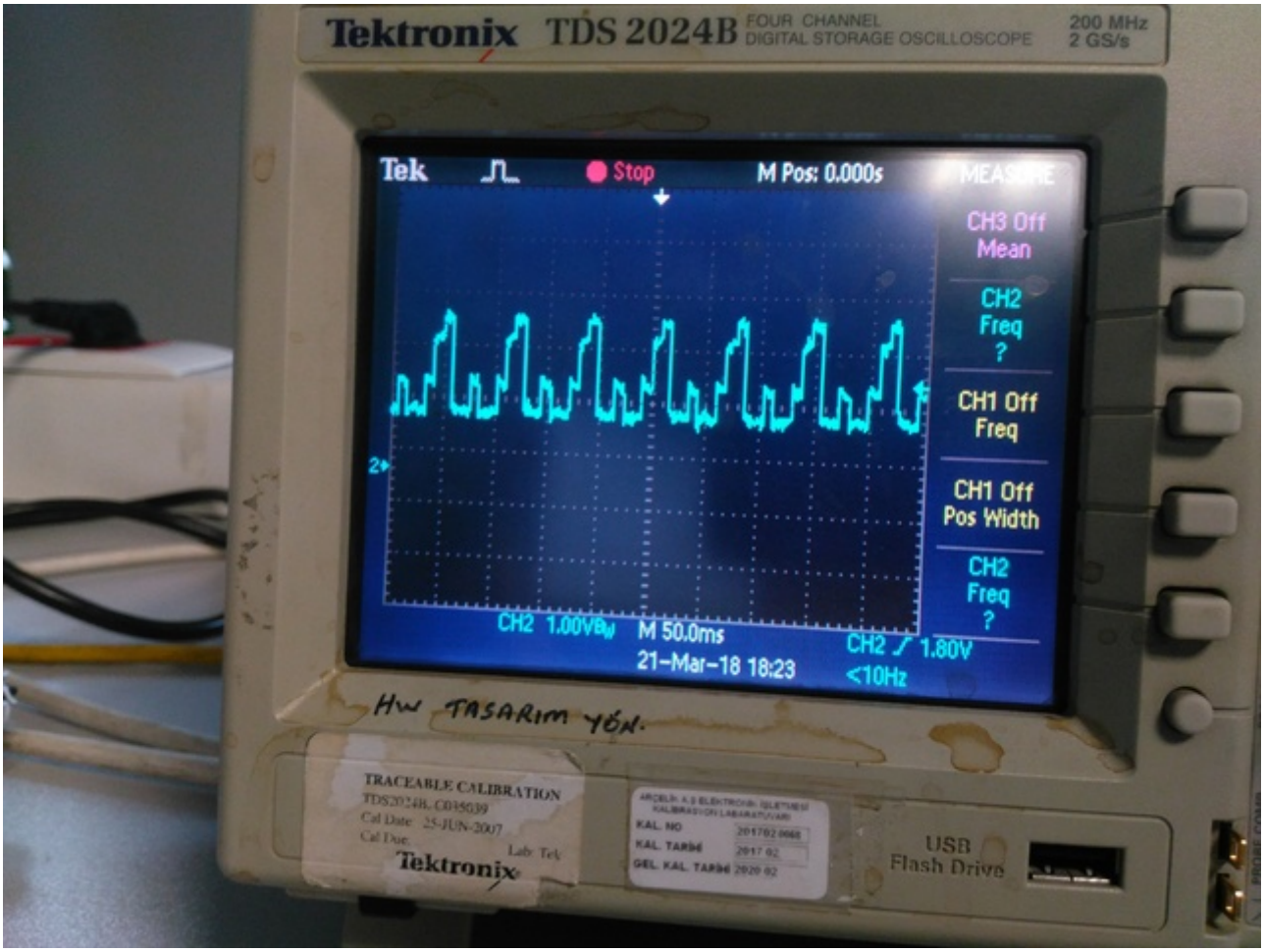
interrupt handler:

```
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 */
HAL_CAN_Receive_IT(&hcan, CAN_FIFO0);
for(j=0; j < 8; j++)
{
//
Rx_Audio[j] = hcan.pRxMsg->Data[j];
}
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, (uint32_t*) Rx_Audio, 8,
DAC_ALIGN_12B_R);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
/* USER CODE END CAN_RX0_IRQn 1 */
}
```

I use DMA->DAC in receiver and DMA->ADC in transmitter. Just for to understand if the problem is in CANBUS, I sen 0x00, 0x01 .... 0x07 as 8 byte data over CANBUS and I saw that CANBUS is clear! However, eventhough Receiver receives the correct data, it can not produce the right output. It receives 0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7...

so the output must have been a triangle wave but this is actually what I get:



And the frequency is wrong. I think there is something messed up in this line in CANBUS RX interrupt handler:

```
HAL_DAC_Start_DMA(&hdac1, DAC_CHANNEL_1, (uint32_t*) Rx_Audio, 8,
DAC_ALIGN_12B_R);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_1);
```

Because, I monitor Rx_Audio and I see that it is correct but the output is wrong. Also I never see GPIOC PIN1 toggles. It never turns on. Where do you think I'm doing wrong?