

Hello to everyone,

I need to communicate with two STM32F407 via CanBus for an my homework. In order to fully understand what I'm doing, I'm just trying to send data first. As far as I know and reseach, I wrote the following code and I saw the signal with the logic analyzer. But "end of frame" does not appear in CAN Frame. Do you help me with this? I'm sorry my English is bad. I hope I could tell the problem.

Note 1: I used Stm32CubeMx.

Note 2: I will use 2 STM32f407G-DISC1.

Note 3: I tried to set the CAN Baut Rate to 250Kbit / s.

Note 4: As an example I have benefited from this: STM32Cube_FW_F4_V1.15.0\Projects\STM324x9I_EVAL\Examples\CAN\CAN_Networking

Note 5: My project files are in the attached files.

Code:

```

/**
 * *****
 * File Name      : main.c
 * Description    : Main program body
 * *****
 *
 * COPYRIGHT(c) 2017 STMicroelectronics
 *
 * Redistribution and use in source and binary forms, with or without modification,
 * are permitted provided that the following conditions are met:
 * 1. Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 * 3. Neither the name of STMicroelectronics nor the names of its contributors
 * may be used to endorse or promote products derived from this software
 * without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 * *****
 */
/* Includes -----*/
#include "main.h"
#include "stm32f4xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables -----*/
CAN_HandleTypeDef hcan1;

/* USER CODE BEGIN PV */
/* Private variables -----*/
#define BUFFER_SIZE 100

char ErrorHandlerReg [BUFFER_SIZE];
char HAL_CAN_TxCpltCallbackStatusReg [BUFFER_SIZE];

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_CAN1_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----*/
static void CAN1_Config(void);
void HAL_CAN_TxCpltCallback(CAN_HandleTypeDef *hcan);
/* 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();

```

```

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

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_CAN1_Init();

/* USER CODE BEGIN 2 */
CAN1_Config();
__HAL_RCC_CAN1_CLK_ENABLE();

if(HAL_CAN_Receive_IT(&hcan1, CAN_FIFO0) != HAL_OK)
{
    /* Reception Error */
    Error_Handler();
}

/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_13,GPIO_PIN_SET);
    hcan1.pTxMsg->StdId = 0x321; //Specifies the standard identifier ( Min_Data = 0 and Max_Data = 0x7FF)
    hcan1.pTxMsg->ExtId = 0x01; //Specifies the extended identifier. (Min_Data = 0 and Max_Data = 0x1FFFFFFF)
    hcan1.pTxMsg->RTR = 0; //Remote Transmission Request (Data Frame: RTR=0, Remote Frame: RTR=1)
    hcan1.pTxMsg->IDE = CAN_ID_EXT; //Specifies the type of identifier for the message that will be transmitted (CAN_ID_EXT, CAN_ID_STD)
    hcan1.pTxMsg->DLC = 8; //Specifies the length of the frame that will be transmitted (Min_Data = 0 and Max_Data
    hcan1.pTxMsg->Data[0] = 0x58; //X
    hcan1.pTxMsg->Data[1] = 0x48; //H
    hcan1.pTxMsg->Data[2] = 0x41; //A
    hcan1.pTxMsg->Data[3] = 0x4C; //L
    hcan1.pTxMsg->Data[4] = 0x44; //D
    hcan1.pTxMsg->Data[5] = 0x55; //U
    hcan1.pTxMsg->Data[6] = 0x4E; //N
    hcan1.pTxMsg->Data[7] = 0x58; //X
    HAL_CAN_Transmit_IT(&hcan1);

    /* USER CODE END 3 */
}

/** System Clock Configuration
*/
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
    __HAL_RCC_PWR_CLK_ENABLE();

    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSISState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = 16;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 8;
    RCC_OscInitStruct.PLL.PLLN = 168;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 4;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }

    /**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_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
    {
        Error_Handler();
    }

    /**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);
}

```

```

/* CAN1 init function */
static void MX_CAN1_Init(void)
{
    hcan1.Instance = CAN1;
    hcan1.Init.Prescaler = 12;
    hcan1.Init.Mode = CAN_MODE_NORMAL;
    hcan1.Init.SJW = CAN_SJW_1TQ;
    hcan1.Init.BS1 = CAN_BS1_6TQ;
    hcan1.Init.BS2 = CAN_BS2_7TQ;
    hcan1.Init.TTCM = DISABLE;
    hcan1.Init.ABOM = DISABLE;
    hcan1.Init.AWUM = DISABLE;
    hcan1.Init.NART = DISABLE;
    hcan1.Init.RFLM = DISABLE;
    hcan1.Init.TXFP = DISABLE;

    if (HAL_CAN_Init(&hcan1) != HAL_OK)
    {
        Error_Handler();
    }
}

/** 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_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);

    /*Configure GPIO pin : PA0 */
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /*Configure GPIO pins : PD12 PD13 PD14 PD15 */
    GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
}

/* USER CODE BEGIN 4 */

static void CAN1_Config(void)
{
    CAN_FilterConfTypeDef sFilterConfig;
    static CanTxMsgTypeDef TxMessage;
    static CanRxMsgTypeDef RxMessage;

    hcan1.pTxMsg = &TxMessage;
    hcan1.pRxMsg = &RxMessage;

    /*##-2- Configure the CAN Filter #####*/
    sFilterConfig.FilterNumber = 0;
    sFilterConfig.FilterMode = CAN_FILTERMODE_IDMASK;
    sFilterConfig.FilterScale = CAN_FILTERSCALE_32BIT;
    sFilterConfig.FilterIdHigh = 0x0000;
    sFilterConfig.FilterIdLow = 0x0000;
    sFilterConfig.FilterMaskIdHigh = 0x0000;
    sFilterConfig.FilterMaskIdLow = 0x0000;
    sFilterConfig.FilterFIFOAssignment = 0;
    sFilterConfig.FilterActivation = ENABLE;
    sFilterConfig.BankNumber = 14;

    if(HAL_CAN_ConfigFilter(&hcan1, &sFilterConfig) != HAL_OK)
    {
        /* Filter configuration Error */
        Error_Handler();
    }
}

void HAL_CAN_TxCpltCallback(CAN_HandleTypeDef *hcan)
{
    sprintf(HAL_CAN_TxCpltCallbackStatusReg, "HAL_CAN_TxCpltCallbackStatusReg\r\n");
}

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @param None
 * @retval None
 */

```

```

void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,GPIO_PIN_SET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(GPIOD,GPIO_PIN_14,GPIO_PIN_RESET);
    HAL_Delay(100);
    sprintf(ErrorHandlerReg,"!!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!! !!!!");
  }
  /* USER CODE END Error_Handler */
}

#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 *****/

```

Logic Analyzer SS

