I am using two STM32F103C8 board and trying to interface the CAN protocol.

1) I tried finding examples code for CAN interface in STM32Cube_FW_F1_V1.6.1 but I am unable to find. I am new to this so maybe I am not able to find. So my first question is that is there any example code available?
2) I have made one board as a transmitter, one board as a receiver, I am using CUBE MX and I have taken help from youtube to make this code.

I am checking the status of HAL_CAN_TRANSMIT and HAL_CAN_RECEIVE.

For transmitting it is giving HAL_OK which according to me is transmitting but for receiving end it is giving HAL_TIMEOUT.

This is the code for the transmitter:

```
#include "main.h"
#include "stm32f1xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
CAN_HandleTypeDef hcan;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
CanTxMsgTypeDef TxMsg;
CanRxMsgTypeDef RxMsg;
CAN_FilterConfTypeDef sFilterConfig;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CAN_Init(void);
static void MX_USART1_UART_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 */
uint8_t status_transmit;
/* 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();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  hcan.pTxMsg=&TxMsg;
  hcan.pRxMsg=&RxMsg;


  sFilterConfig.FilterNumber=0;
  sFilterConfig.FilterMode=CAN_FILTERMODE_IDLIST;
  sFilterConfig.FilterScale=CAN_FILTERSCALE_32BIT;
  sFilterConfig.FilterIdHigh=0x245<<5;
  sFilterConfig.FilterIdLow=0;
  sFilterConfig.FilterMaskIdHigh=0;
  sFilterConfig.FilterMaskIdLow=0;
  sFilterConfig.FilterFIFOAssignment=0;
  sFilterConfig.BankNumber=14;

  HAL_CAN_ConfigFilter(&hcan,&sFilterConfig);

  hcan.pTxMsg->StdId=0x244;
  hcan.pTxMsg->RTR=CAN_RTR_DATA;
  hcan.pTxMsg->IDE=CAN_ID_STD;
  hcan.pTxMsg->DLC=1;
  hcan.pTxMsg->Data[0]=0x31;
  /* USER CODE END 2 */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  status_transmit='t';
  HAL_UART_Transmit(&huart1,&status_transmit,1,1);
  while (1)
  {
  if(HAL_CAN_Transmit(&hcan,1)==HAL_OK)
  {
  status_transmit='O';
  HAL_UART_Transmit(&huart1,&status_transmit,1,1);
  }
  else if(HAL_CAN_Transmit(&hcan,1)==HAL_TIMEOUT)
  {
  status_transmit='T';
  HAL_UART_Transmit(&huart1,&status_transmit,1,1);
```

```
}
else if(HAL_CAN_Transmit(&hcan,1)==HAL_ERROR)
{
status_transmit='E';
HAL_UART_Transmit(&huart1,&status_transmit,1,1);
}

HAL_Delay(1000);
/* 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);
}

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

hcan.Instance = CAN1;
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__);
}

}

/* USART1 init function */
static void MX_USART1_UART_Init(void)
{

huart1.Instance = USART1;
huart1.Init.BaudRate = 115200;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/** Pinout Configuration
*/
static void MX_GPIO_Init(void)
{

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

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
```

```
/**
* @brief This function is executed in case of error occurrence.
* @param file: The file name as string.
* @param line: The line in file as a number.
* @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,
tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/**
* @}
*/

/**
* @}
*/

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

This is the code for the receiver

```
#include "main.h"
#include "stm32f1xx_hal.h"

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
CAN_HandleTypeDef hcan;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
```

```
CanTxMsgTypeDef TxMsg;
CanRxMsgTypeDef RxMsg;
CAN_FilterConfTypeDef sFilterConfig;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_CAN_Init(void);
static void MX_USART1_UART_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 */
uint8_t status_receive;
/* 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();
MX_USART1_UART_Init();
/* USER CODE BEGIN 2 */
hcan.pTxMsg=&TxMsg;
hcan.pRxMsg=&RxMsg;


sFilterConfig.FilterNumber=0;
sFilterConfig.FilterMode=CAN_FILTERMODE_IDLIST;
sFilterConfig.FilterScale=CAN_FILTERSCALE_32BIT;
```

```
sFilterConfig.FilterIdHigh=0x244<<5;
sFilterConfig.FilterIdLow=0;
sFilterConfig.FilterMaskIdHigh=0;
sFilterConfig.FilterMaskIdLow=0;
sFilterConfig.FilterFIFOAssignment=0;
sFilterConfig.BankNumber=14;

HAL_CAN_ConfigFilter(&hcan,&sFilterConfig);

hcan.pTxMsg->StdId=0x245;
hcan.pTxMsg->RTR=CAN_RTR_DATA;
hcan.pTxMsg->IDE=CAN_ID_STD;
hcan.pTxMsg->DLC=1;
hcan.pTxMsg->Data[0]=0x31;
/* USER CODE END 2 */
/* Infinite loop */
/* USER CODE BEGIN WHILE */
status_receive='R';
HAL_UART_Transmit(&huart1,&status_receive,1,1);
while (1)
{
if(HAL_CAN_Receive(&hcan,CAN_FIFO0,1000)==HAL_OK)
{
status_receive='O';
HAL_UART_Transmit(&huart1,&status_receive,1,1);
}
else if(HAL_CAN_Receive(&hcan,CAN_FIFO0,1000)==HAL_TIMEOUT)
{
status_receive='T';
HAL_UART_Transmit(&huart1,&status_receive,1,1);
}
else if(HAL_CAN_Receive(&hcan,CAN_FIFO0,1000)==HAL_ERROR)
{
status_receive='E';
HAL_UART_Transmit(&huart1,&status_receive,1,1);
}


/* 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);
}

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

hcan.Instance = CAN1;
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__);
}

}
```

```
/* USART1 init function */
static void MX_USART1_UART_Init(void)
{

huart1.Instance = USART1;
huart1.Init.BaudRate = 115200;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}

}

/** Pinout Configuration
*/
static void MX_GPIO_Init(void)
{

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

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @param file: The file name as string.
* @param line: The line in file as a number.
* @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)
{
```

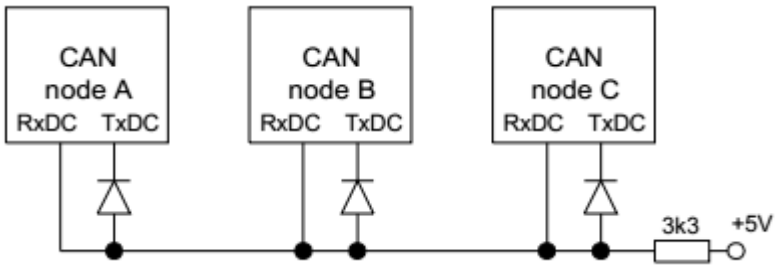
```
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and line number,
tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/**
* @}
*/

/**
* @}
*/

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

The circuit I am using is




Can you please help me out for this?

Thanks