LEARN ESP32

UART #1 || Simple Tx Rx

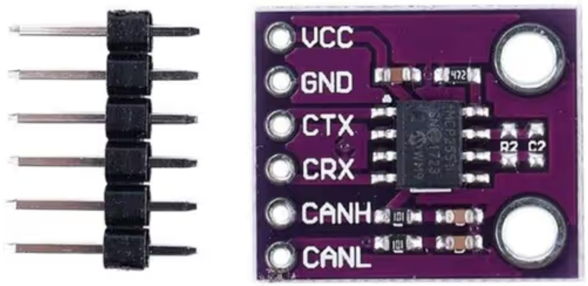
UART #2 || Control LED

I2C #1 || Interface LCD1602

I2C #2 || Read Write MPU6050

CAN Protocol in STM32

This tutorial will cover the Basic Can protocol in STM32. Here we will see how to communicate between two STM32 MCUs using the CAN protocol. Of course we would need 2 can transceivers (at least) to do that, and that's why I am using **MCP2551** can transceivers.



LEARN AVR

AVR #1 || LED Blinking

AVR #2 || Input Button

AVR #3 || UART Send Data

AVR #4 || UART Recv Data

AVR #5 || I2C Send Data

A Little info about the CAN Protocol

AVR #6 || Interface PCF8574

AVR #7 || LCD1602 (I2C)

AVR #8 || Read EEPROM (I2C)

LEARN STM32

Interafce LCD1602 via I2C

Interface LCD1602 (Parallel)

Custom Chars in LCD1602

Interface AIP31068 LCD1602

Interface LCD20x4 (I2C)

Interface SSD1306 Oled

Interface SH1106 Oled

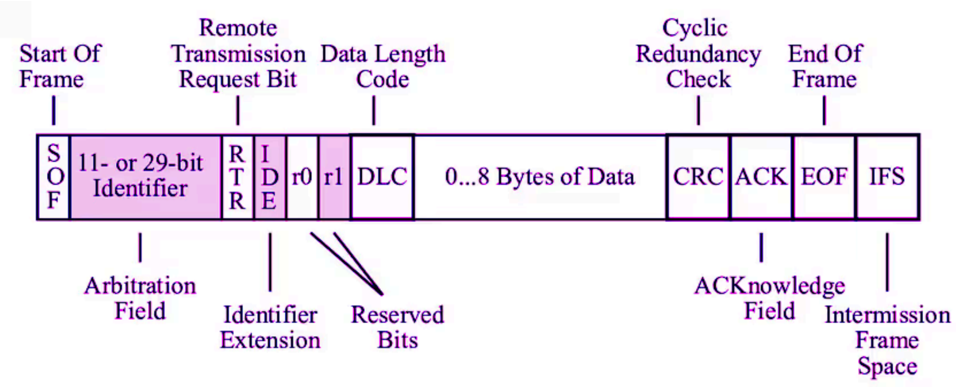
ST7920 GLCD (Parallel)

Interface TFT Display

I am not going to explain every small detail here, instead we will just focus on some important things. For more details about the Protocol, you can google it.

CAN (**Controlled Area Network**) Protocol is a way of communication between different devices, but under certain rules. These rules must be followed when a message is transmitted over the CAN bus. Here we are going to see these rules.

Below is the image showing the Standard CAN Frame.



- Here, **Identifier** is the ID of the transmitting Device. It can be either **11 bits** (Standard ID) or **29 bits** (Extended ID).
- **RTR** (Remote Transmission Request) Specifies if the data is Remote frame or Data frame.
- **IDE** specifies if we are using **Standard** ID or **Extended** ID.

[ST7735 TFT Display (SPI)]
[Interface Dot Matrix Display]
[Cascade Dot Matrix Display]
[Scroll strings on Dot Matrix]
[Timer #1 || PWM Output]
[Timer #2 || PWM Input]
[Timer #4 || Input Capture]
[Timer #5 || Slave Trigger]
[Timer #6 || 3 Phase PWM]
[Timer #7 || Reset Mode]
[Timer #8 || 48 bit Counter]
[Timer #9 || One Pulse Mode]
[Timer #10 || Gated Mode]
[Generate MicroSec Delay]
[Interface DHT11 Sensor]

- **r** is the Reserved bit.
- **DLC** specifies the data length in Bytes.
- **Data Field** is where we send the actual data bytes. It can be upto 8 bytes in size.
- **CRC** is the checksum data byte.
- **ACK** is the acknowledgment bit.

In this Tutorial, we will see upto the **Data Field only**. The CRC and ACK will be handled by the HAL Library.

Connection and Configuration

The CubeMX Configuration is as shown below.

[Interface DHT22 Sensor](https://controllerstech.com/)

Interface DS18B20 Sensor

Multiple DS18B20 Sensors

Interface ADXL345 I2C

Interface MPU6050 (I2C)

Interface EEPROM (I2C)

Interface Servo Motor

Interface Steeper Motor

Stepper Angle Control

Interface BLDC Motor

Interface Continuous Servo

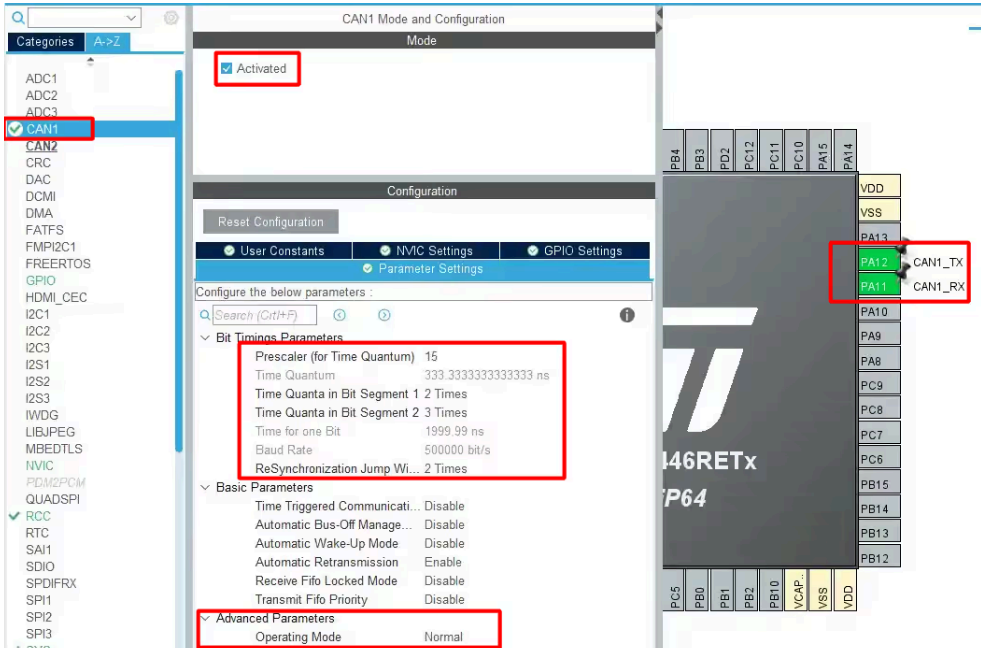
Control RPM of Cont Srvo

How to use SPI Peripheral

Interface SD Card via SPI

Interface SD Card (SDIO)

Internal RTC Real Time Clk



I am using CAN1 for this tutorial.

- Here the **BAUD RATE** is set to 500000 bps. You can try different different combinations for **Prescalar and Time Quanta** to achieve this.
- The Operating Mode is **NORMAL Mode**.
- Pins PA11 and PA12 are configured as **CAN_RX** and **CAN_TX**.

We also need to enable the pull up for the RX pin as shown below.

Interface DS3231 RTC

Interface 4x4 Keypad

HCSR04 Ultrasonic Sensor

UART Ring Buffer

Multiple UART Ring Buffer

UART DMA IDLE Line

Interface Joystick via ADC

ADC Multiplexer 74HC4051

FreeRTOS #1 || Intro

FreeRTOS #2 || Tasks

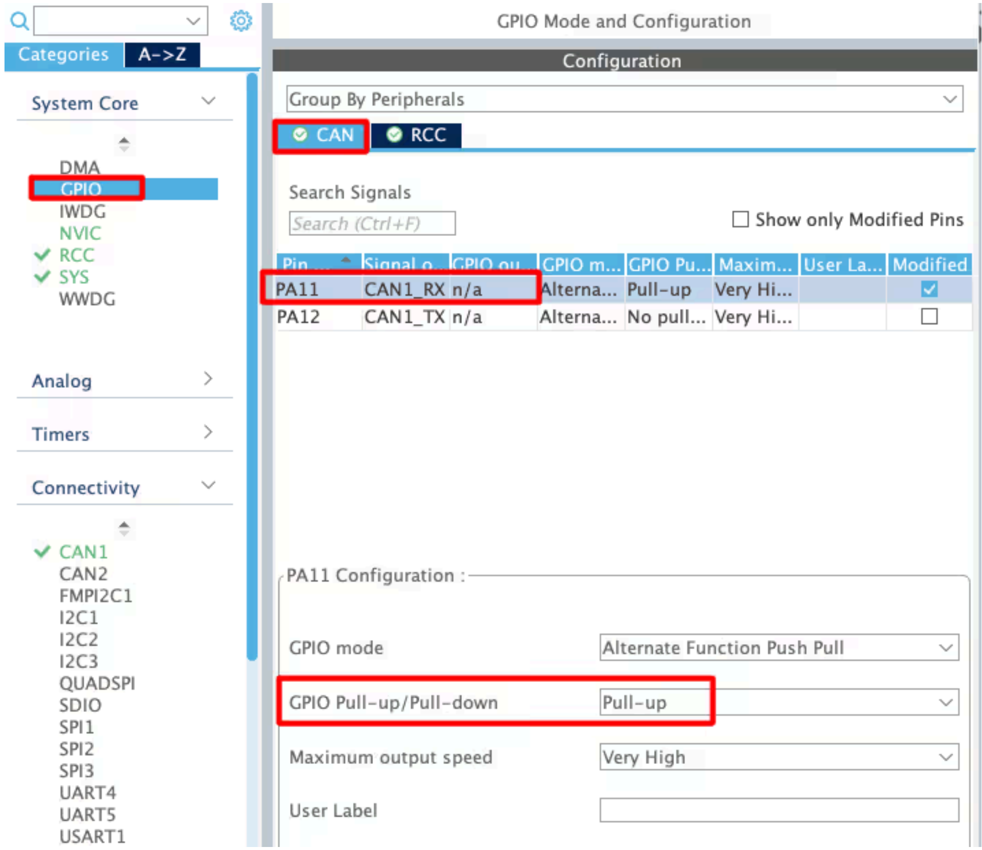
FreeRTOS #3 || Binary Sem

FreeRTOS #4 || Count Sem

FreeRTOS #5 || Queue

FreeRTOS #6 || Mutex

FreeRTOS #7 || Soft Timer



The Connection between F446 and F103 is shown below.

How to use DAC

Low Power Modes

ESP8266 WebServer

Data Exchng with Websvr

Cloud Data Logger

Program FLASH Memory

USB MSC Device

USB MSC Host

USB HID Host

USB CDC Host & Device

USB Virtual COM Port

Emulate STM32 as Mouse

Emulate as Keyboard

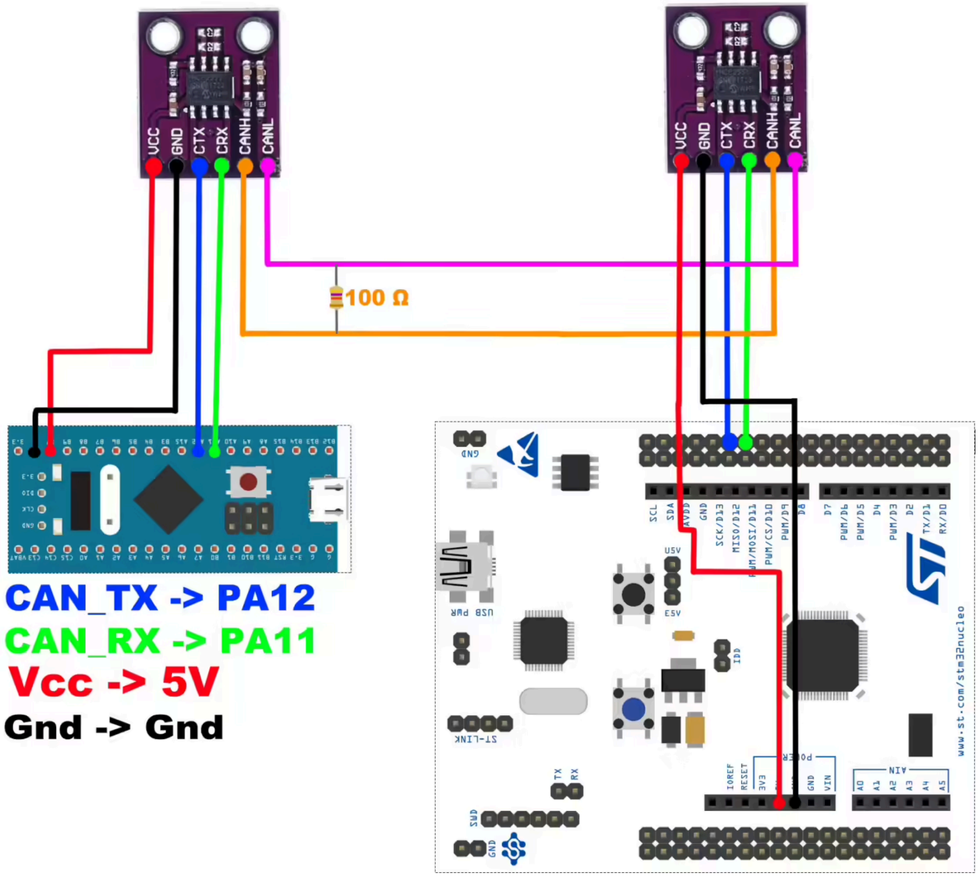
WatchDogs in STM32

Interface BMP180 Sensor

Interface BME280 Sensor

Communicate using HC-12

Register Based #1 || Intro



- Here the **CAN_Tx** and **CAN_Rx** from the Transceivers are connected to **PA12** and **PA11** of the Respective controllers
- **CANH** and **CANL** are connected to each other
- Also there is a **120 ohms Resistance** at the node. This is very important, or else you will not get the data.

How to Modify the CAN Data Frame

To do this, we will define some variables, where we can store the header and the data information.

```
CAN_TxHeaderTypeDef   TxHeader;
uint8_t               TxData[8];
uint32_t              TxMailbox;
```

- Here **TxHeader** will be used to store the header information, like RTR, DLC, etc. This is the type `CAN_TxHeaderTypeDef` .
- **TxData** is used to store the data that we are going to transmit over the CAN bus.
- **TxMailbox** is the mailbox, which will be sent to the CAN bus.

Now we will store the required values in the **TxHeader**, and in the **TxData**.

Using Incremental Encoder
Ethernet #1 || Connection
Ethernet #2 || UDP Server
Ethernet #3 || UDP Client
Ethernet #4 || TCP Server
Ethernet #5 || TCP Client
Ethernet #6 || HTTP Websrvr
Ethernet #6.1 || Websrvr SSI
Ethernet #6.2 || Websrvr CGI
Ethernet #7 || UDP Serv NETCONN
Ethernet #8 || UDP Clnt NETCONN
Ethernet #9 || TCP Serv NETCONN

```
TxHeader.IDE = CAN_ID_STD;
TxHeader.StdId = 0x446;
TxHeader.RTR = CAN_RTR_DATA;
TxHeader.DLC = 2;

TxData[0] = 50;
TxData[1] = 0xAA;
```

- Here **CAN_ID_STD** means that we are using the Standard ID (not extended)
- **0x446** is the Identifier. This is the ID of the Transmitter, and it should be maximum 11 bit wide for the Standard ID.
- **CAN_RTR_DATA** indicates that we are sending a data frame
- **DLC** is the Length of data bytes, and here we are sending 2 data Bytes
- Now we will store the 2 data bytes in the **TxData** array

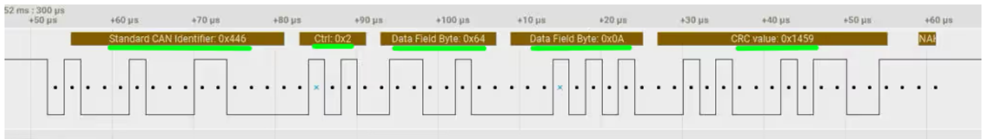
We have the information ready to be transmitted, and now we will finally transmit it on the CAN bus

[Ethernet #10 || TCP Clnt NETCONN]
[Ethernet #11 || HTTP Serv AJAX P1]
[Ethernet #12 || HTTP Serv AJAX P2]
[How to use the SDRAM]
[Interface GPS Neo 6M]
[Interface RS485 Module]
[Introduction to Modbus]
[Modbus Protocol #1]
[Modbus Protocol #2]
[Modbus Protocol #3]
[Modbus Protocol #3.1]
[Modbus Protocol #4]
[Modbus Protocol #5]
[Modbus Protocol #6]
[Modbus Protocol #7]
[TouchGFX #1 || Intro]

```
if (HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailb
{
   Error_Handler ();
}
```

This can be done by using the function **`HAL_CAN_AddTxMessage`** . It have the following parameters

- **hcan1** is the instance of the CAN, that we are using.
- **TxHeader** is the Header of the message.
- **TxData** is the Data field.
- **TxMailbox** is the mailbox, which will carry the header and data message.



You can see above the data on the TX Line.

- **Identifier is 0x446**, the STD ID of the transmitter
- **Control Field is 0x2**, it contains DLC, RTR, IDE
- **2 Bytes** of data Field

TouchGFX #2 || Text Area

TouchGFX #3 || Send to UI

TouchGFX #4 || Send to UI

TouchGFX #5 || UART to UI

TouchGFX #6 || Multi Screen

TouchGFX #7 || Keyboard

TouchGFX #8 || UI to MCU

STM32 I2C SLAVE PART 1

STM32 I2C SLAVE PART2

STM32 I2C SLAVE PART3

STM32 I2C SLAVE PART4

STM32 I2C SLAVE PART5

STM32 I2C SLAVE PART6

STM32 I2C SLAVE PART7

W25Q #1 || Connection

W25Q #2 || Read

- And at last there is CRC Value, which was added by the HAL

This message is sent to the CAN bus, and now all the CAN devices on this bus will sort of receive this message. I said sort of, because whether to receive the message or not, depends on the **Filter Configuration** for each device.

If the message satisfies the conditions as per the **FILTER**, only then it will be allowed to pass.

Filter Configuration

In order to reduce CPU Load to filter out messages, the STM32 have the Filters built inside the CAN peripheral. Let's Check them out

```
CAN_FilterTypeDef canfilterconfig;

canfilterconfig.FilterActivation = CAN_FILTER_ENABLE;
canfilterconfig.FilterBank = 18;  // which filter bank to
canfilterconfig.FilterFIFOAssignment = CAN_FILTER_FIFO0;
canfilterconfig.FilterIdHigh = 0x446<<5;
```

W25Q #3 || Erase Sectors
W25Q #4 || Write Pages
W25Q #5 || Update Sectors
W25Q #6 || Write Numbers
W25Q #7 || QSPI MODE
W25Q #8 || External Loader
W25Q #9 || SPI Loader
W25Q #10 || QSPI on H7
W25Q #11 || Xecute In Place
Interface HC-05 BlueTooth
LVGL #1 || Introduction
LVGL #2 || Create UI
LVGL #4 || Send data to UI
LVGL #5 || Keyboard
LVGL #6 || QSPI Flash

```
canfilterconfig.FilterIdLow = 0;
canfilterconfig.FilterMaskIdHigh = 0x446<<5;
canfilterconfig.FilterMaskIdLow = 0x0000;
canfilterconfig.FilterMode = CAN_FILTERMODE_IDMASK;
canfilterconfig.FilterScale = CAN_FILTERSCALE_32BIT;
canfilterconfig.SlaveStartFilterBank = 20;  // how many fi

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

1. **FilterActivation** specifies if we want to enable Filters or not. Obviously we have to enable them
2. **SlaveStartFilterBank** specifies How many Filter Banks do we want to assign to CAN1. Basically the controllers with **dual CAN peripheral** have 28 Filter Banks, which can be distributed between these 2 CAN. Here I am assigning 20 Filter Banks

5. **FilterMode** specifies which type of Filter do we want to use. We have 2 types of filters in STM32. **MASK MODE**, where the Mask register will be used to compare some particular bits in the ID register to the incoming ID. And the **LIST MODE**, where the incoming ID is directly compared with the ID set in the ID Register.
   - I am using MASK Mode here, as it

LVGL #7 || Run from Ext Flash

UART #1 || Send Data

UART #2 || Send Data (DMA)

UART #3 || Recv Data

UART #4 || Recv Data (DMA)

UART #5 || IDLE Line

UART #6 || Half Duplex Mode

UART #7 || 1-wire protocol

UART #8 || LIN PART 1

UART #9 || LIN PART 2

UART #10 || LIN PART 3

FMC LCD #1 || Connection

FMC LCD #2 || Add Touch

FMC LCD #3 || Add LVGL

to the CAN1, and Rest to the CAN 2.

- This parameter is useless for the controllers with **single CAN peripheral**. And these Controllers have 14 Filter Banks (0 to 13)

3. **FilterBank** specifies which Filter Bank do we want to use for the filter Process. Here I have assigned 20 Banks for CAN 1, and I can only choose Out of these 20 Banks. So I am choosing Bank number 18.

- In case of **Single CAN Peripheral**, you can choose any

seems to be more useful

6. **FilterScale** specifies If we want to use one 32 bit Filter Register, or 2 16 bit Filter Registers.

- I am using one 32 Bit Register here.

7. **FilterIdHigh** is the Higher 16 Bits of the ID register. The value set in this register will be compared to the incoming Identifier.

- Here I have decided to only compare the **STD ID** of the incoming message, and that’s why I am shifting the value by 5. The STD ID starts from 5th bit in the ID HIGH Register

Interface TM1637 Display

Clock on TM1637 Display

Interface Passive Buzzer

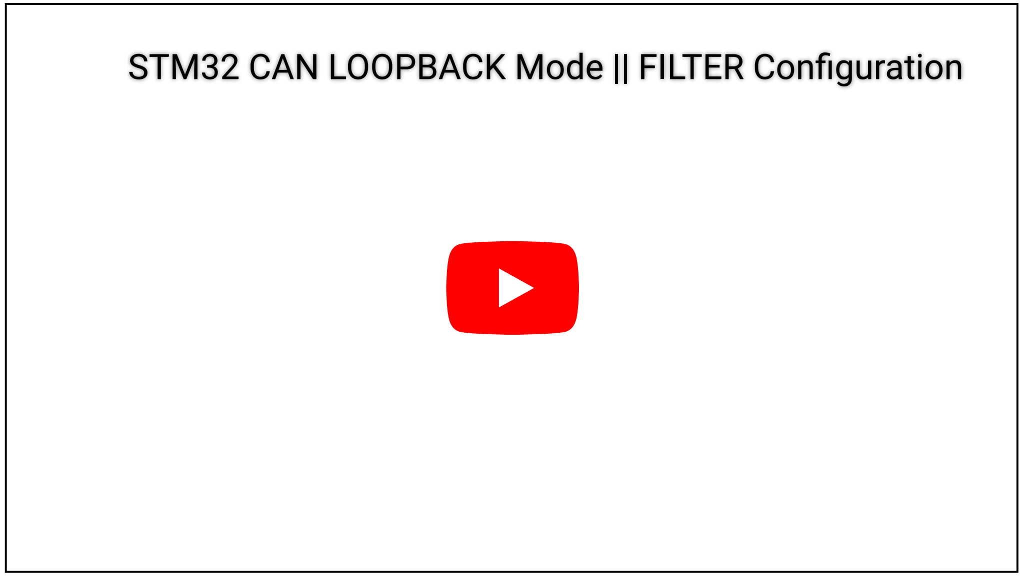
value between 0 to 13

4. **FilterFIFOAssignment** specifies which FIFO are we going to use for the Receive message. Generally we have **2 FIFOs** (FIFO 0, and FIFO 1). I am choosing FIFO 0

8. **FilterMaskIdHigh** is the Higher 16 Bits of the MASK register. The value set in this register will enable the comparison of that particular bit in the ID register to that of the incoming ID.

The Last 2 points might be hard to understand, so I would suggest that you watch the video below. It could be better explained with the working example, and that's shown in the VIDEO.

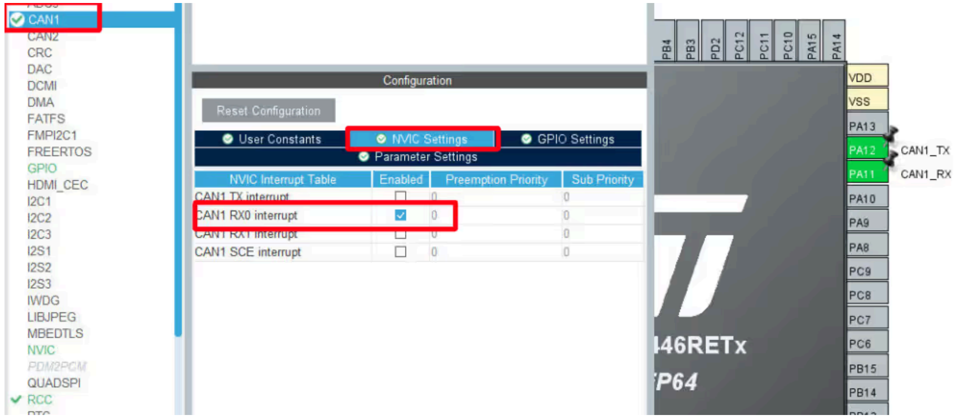
Check out the Video Below



Receiving DATA

We will use the **interrupt for the RX FIFO**, so whenever a message is passed through the Filter an interrupt will be triggered.

First of all We will enable the CAN1 RX0 interrupt in the CubeMX



Now Inside the main Function, we will **Activate the Notification** for the Received message.

```
if (HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_M
{
    Error_Handler();
}
```

Here we will choose **`CAN_IT_RX_FIFO0_MSG_PENDING`** . This would trigger the interrupt whenever there is some pending message in the RX_FIFO 0. Once the interrupt is triggered, a callback function will be called. In this case, it will be

`HAL_CAN_RxFifo0MsgPendingCallback`

```
CAN_RxHeaderTypeDef   RxHeader;
uint8_t               RxData[8];

void HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hc
{
  if (HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, Rx
  {
    Error_Handler();
  }
  if ((RxHeader.StdId == 0x103))
  {
      datacheck = 1;
  }
}
```

- Here we will **Receive** the message from **RX_FIFO 0**.
- The **message Header** will be stored in the RxHeader, and the **data** will be stored in RxData.
- We can do **further checks**, like if the message was **received from the ID 0x103**, then the **datacheck** flag will be set.
- Later in the while loop we can perform some actions based on this flag

```
if (datacheck)
{
 HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
}
```

For example, If the Flag is set, The **LED** will turn ON.

RESULT

The result here is hard to put in images, so I would suggest that you watch the video for more detailed working.

Check out the Video Below

STM32 CAN Communication || NORMAL Mode

DOWNLOAD SECTION

Info

You can help with the development by DONATING Below.

To download the project, click the DOWNLOAD button.

DOWNLOAD

DONATE

Subscribe

Join the discussion

{} [+]

Hottest comment thread

Most reacted comment

35 COMMENTS

Newest

Guest
Arseny

Guest September 23, 2024 9:46 PM

Why are there two reserved bits (r0, r1) after the ID and before the data length code, if according to the documentation of Bosch there is one reserved bit r0?

1 Reply

Guest
Mohammadreza Mehrdad

Guest December 3, 2023 7:00 PM

Hello,
Thanks for your useful tutorials.

I am going to use Extended CAN of stm32 but I could not find how I can do it. Is it possible you help and tell me which part of the code need to be changed.

Regards,

Mehrdad

0 Reply

View Replies

View Replies (1)

Guest

Alex

Guest November 18, 2023 1:06 AM

Olá amigo, qualo modelo das placas, a menor é a BluePill e a outra é que núcleo?

0 Reply

Guest

Jayaprakash

Guest December 23, 2022 12:16 PM

Hi, I am using stm32f407 I tried your code but there is no data is received,i don't know what I Missed

4 Reply

Guest

Roman

Guest December 1, 2022 4:43 AM

Hey man! Big thanks, your code examples are a great help, especially this one. Video tutorials on youtube are great too. (if only not for the robotic voice-over, but I understand how hard it is to make a proper voice-over)

0 Reply

Guest

kaasi

Guest September 7, 2022 12:21 PM

super ra kaasi

0 Reply

Guest
Annika

Guest September 2, 2022 6:38 PM

Thank you very much for this helpful tutorial. I also watched the vid on youtube for the can one master multiple slaves. Do I understand right: I can only tell a controller to which other controllers (IDs) to listen to, but I cannot tell the master to send a message to a certain slave? So all slaves listen to the master, and the master message passes the filter of all slaves, and I have to pack a “address information” into the data part of the message, which every slave has to check then? I would be very happy if you… Read more »

View Replies

0 Reply View Replies (2)

Guest
Thamanoon Kedwiriyakarn

Guest July 6, 2022 9:13 AM

Thank you very much Sir.

0 Reply

Guest
Fabian

Guest June 27, 2022 2:37 AM

I have to establish CAN bus communication between two stm32f407VET6 boards and it doesn’t work for me, can I directly connect tx to tx and rx to rx? because I understand that the boards have two integrated can, I do not have the MCP2551, I have two MCP2515

View Replies

1 Reply View Replies (1)

MrH Guest

Guest June 22, 2022 10:39 AM

Hi, I am using STM32F103C8 and I have tried your code and setup. However, your code is working on “loopback mode” and not working in “Normal Mode”. I don’t know why, it won’t transmit and wont receive message. I believe there is other people commented this problem on your youtube (CAN Normal Mode) video but has yet get an answer, also there are many other facing this issue as well (stack forum or st community) both also do not have answer. If anyone reading this and have solution please share it here. Thank you. Extra Info: If I configure in… Read more »

View Replies

4 Reply View Replies (5)

Ronter Guest

Guest April 2, 2022 5:05 PM

Hello,

ı want to know, which is id transmitted message to me? How can ı lean?

“HAL_CAN_AddTxMessage” methods accept uint8_t data, can ı transmit array?

In my scenario I have a master and 5 slaves. 5 slaves are sending messages to master. And I want to know, which slave sent the message to the master?

Thank you for this post.

Last edited 2 years ago by Ronter

View Replies

0 Reply View Replies (1)

Иван Guest

Guest March 5, 2022 1:19 AM

Нужны stm32 can

0 Reply

Guest
Jaboop

Guest February 18, 2022 12:04 AM

Do you have any good sources on what modifications need to be made to use the FDCAN peripheral on the H7 processors? I can see some signals making it back and forth between the two boards but they analyze as Errors and after two exchanges back and forth the CAN State get stuck in BUSY.

View Replies

0 Reply View Replies (1)

Guest
Lakshminarayana

Guest February 10, 2022 6:43 PM

Hello Thanks for the tutorial it was helpful but i have one problem, i can only receive the data in Blue Pill but the data sent by the Bull pill is not received at F4 controller and i am using CAN BUS 2 for the communication i need as all suggested by you. but still no way i could achieve it.

could you please guide me what wrong i am doing.

Thanks
With regards
Lakshminarayana KS

View Replies

0 Reply View Replies (2)

Guest
velan vs

Guest December 7, 2021 11:21 AM

super bro

0 Reply

Guest
Sharan

Guest September 23, 2021 5:48 PM

Hey, This worked well. Now is it possible to use multiple addresses with a device to transmit data? That is the slave has multiple addresses for a single device, each address has different data values. Now the master accesses the slave with multiple addresses according to the required data values. For example:- The master requires the data"time" in the slave, the master uses the particular address(0x102) for the data "time".If the master requires the data "Brightness" in the slave, the master uses the particular address(0x103) for the data "Brightness".Likewise…. Can you give me a solution to work out this in… Read more »

0 Reply

Guest
tue nguyen

Guest August 30, 2021 9:31 PM

I use CAN1 for F407VG, it works normal but use CAN2 it does not receive data. I dont know why?

Last edited 3 years ago by tue nguyen

View Replies

1 Reply View Replies (2)

Guest
VAN-NHI NGUYEN

Guest August 1, 2021 2:21 PM

Hello, i don’t know what the value 0x443 in STD ID is for, while on the receiver side we check the value 0x103.

View Replies

0 Reply View Replies (1)

Har Guest

Guest June 26, 2021 2:47 PM

Hi Admin,

Thank you so much for your tutorials…!

I am STM32F103 CAN (MCP2561 as CAN transceiver) with 500kbps and CAN BUS ANALYZER to check the CAN msgs.

Here am sending 3 CAN msgs for every 100ms but some times some msgs are missing so that am getting the CAN cycle time as 200ms and I have followed the same procedure as above but some times my CAN msgs are missing.

what I have to do eliminate this error ?

Thanks again

21 Reply