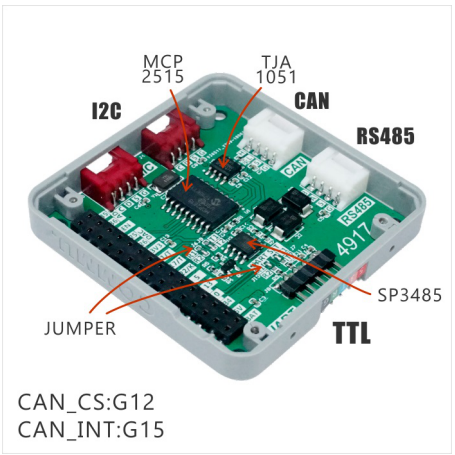
Module COMMU



Description

COMMU is a Muti-Communication-Interface-Converter. Integrated with 2*IIC, 1*TTL, 1*CAN, 1*RS485.

Apparently COMMU has packed with most of series communications.

Default connection: TTL - UART0, RS485 - UART2. Since ESP32 pin map is allowed for re-assign, you can re-assign or re-mapping the TTL or RS485 interface to other pins.

Be care about TTL Interface. It is a UART Interface actually by default. But you can switch it to connect with UART2 after changed those jumpers(J6, J7, J9, J10).

Product Features

- 2x I2C Interface
- 1x CAN Interface
- 1x RS485 Interface
- 1x TTL Interface
- CAN controller: MCP2515-1/SO
- RS485 Transceiver: SP3485EN-L/TR
- Product Size：54.2mm x 54.2mm x 12.8mm
- Product weight：13.5g

Include

- 1x M5Stack COMMU Module

PinMap

CAN	*ESP32 Chip*
CAN_CS	GPIO12
CAN_INT	GPIO15
CAN_SCK	GPIO18

CAN	*ESP32 Chip*
CAN_MISO	GPIO19
CAN_MOSI	GPIO23

I2C Interface	*ESP32 Chip*
IIC_SDA	GPIO21
IIC_SCL	GPIO22

EasyLoader

1.EasyLoader is a simple and fast program burner. Every product page in EasyLoader provides a product-related case program. It can be burned to the master through simple steps, and a series of function verification can be performed.(**Currently EasyLoader is only available for Windows OS**)

2.After downloading the software, double-click to run the application, connect the M5 device to the computer via the data cable, select the port parameters, and click **"Burn"** to start burning.

3.The CP210X (USB driver) needs to be installed before the EasyLoader is burned.

Example

Arduino IDE

CAN communication

These are two COMMU examples for CAN communication, tansmitter and receiver. Press Button A to sent the message, and display the received message on the screen.

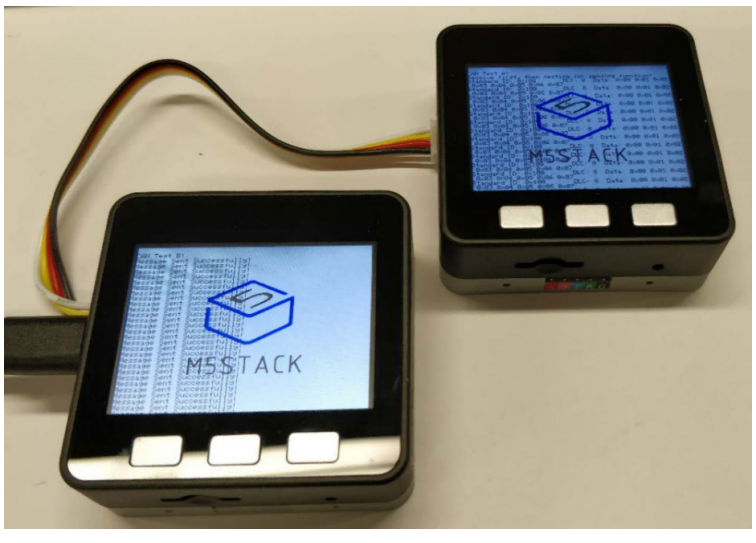
Step 1: Copy MCP_CAN_lib file to `C:\Users\<user_name>\Documents\Arduino\libraries` , **Step 2**: Open project file `commu_can_transmitter.ino` , and commu_can_receiver.ino

Step 3:upload the two project to two M5Cores separatly.

The below code is incomplete(just for usage).

```arduino
/*
    commu_can_transmitter.ino
*/
#include <M5Stack.h>
#include <mcp_can.h>
#include "m5_logo.h"

#define CAN0_INT 15 // Set INT to pin 2
MCP_CAN CAN0(12);   // Set CS to pin 10

// declaration
byte data[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};

// initialization
M5.begin();
CAN0.begin(MCP_ANY, CAN_1000KBPS, MCP_8MHZ);
/* Change to normal mode to allow messages tobe transmitted */
CAN0.setMode(MCP_NORMAL);

  // send data
CAN0.sendMsgBuf(0x100, 0, 8, data);
```

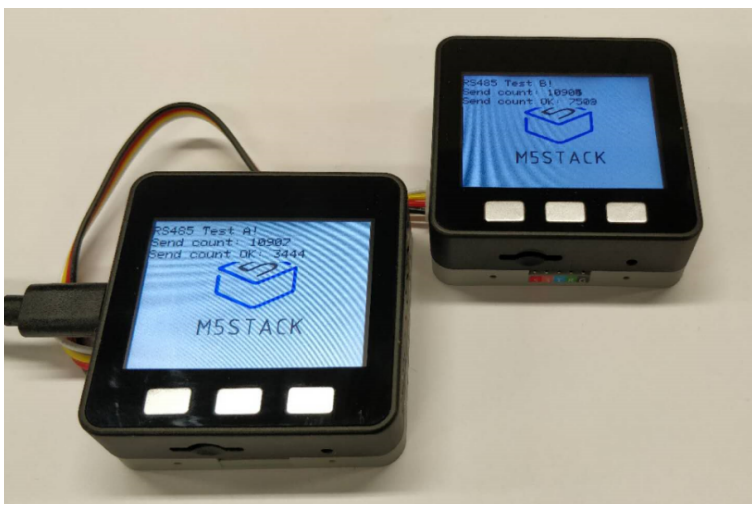
```arduino
/*
    commu_can_receiver.ino
*/
#include <M5Stack.h>
#include <mcp_can.h>
#include "m5_logo.h"

#define CAN0_INT 15 // Set INT to pin 2
MCP_CAN CAN0(12);   // Set CS to pin 10

// declaration
byte data[8] = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07};

// initialization
M5.begin();
/* Initialize MCP2515 running at 16MHz with a baudrate of 500kb/s */
/* and the masks and filters disabled. */
CAN0.begin(MCP_ANY, CAN_1000KBPS, MCP_8MHZ);
/* Set operation mode to normal so theMCP2515 sends acks to received data. */
CAN0.setMode(MCP_NORMAL);
pinMode(CAN0_INT, INPUT);// Configuring pin for /INT input

// read data
CAN0.readMsgBuf(&rxId, &len, rxBuf);
```



RS485 communication

This is a COMMU example for RS485 communication.

Burn example to two M5Cores. Then after pressed Button A, this two cores will send message to each other and receive data.



Schematic