PROJECT REPORT

TRANSFER OF PROCESS DATA USING RS232 AND RS485

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

Objective:

To transfer the process data acquired from a microcontroller to the other microcontroller using Universal Asynchronous Receiver Transmitter by RS232 and RS485 defined electrical voltages.

Hardware Required:

- 2 x Raspberry Pi Pico - Microcontroller Board
- 2 x RS232 to TTL Serial Interface Module
- 2 x MAX485 TTL To RS485 Serial Interface Module
- Null modem cable (DB9 cable)
- 16 x 2 LCD Display
- 10K Potentiometer for adjusting the contrast of the LCD display
- Photoresistor
- MB102 Breadboard Power Supply Module 3.3V/5V
- Breadboards
- Jumper wires

Introduction:

Our project involves reading the voltage across the photoresistor whose resistance is maximum at darkness and and minimum at bright ambient light by connecting across the 3.3V output and one of the ADC channels on one microcontroller and transmitting the data to the other other microcontroller using RS232/RS485. The data received on the other microcontroller is interfaced with a LCD display on which the data is displayed.

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

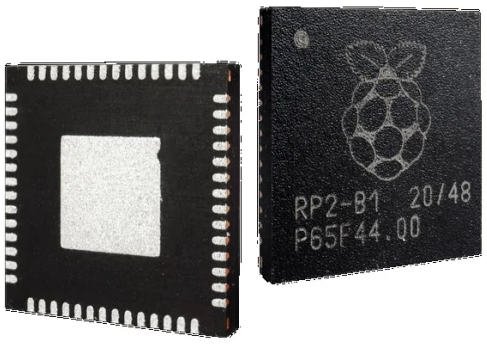
Project Background Topics:

The Raspberry Pi Pico microcontroller board uses the Raspberry Pi RP2040 microcontroller chip.

Microcontroller is a IC chip compact in size, low in cost, employed in embedded systems, performs only a single task with all the required peripherals built-in which is an advantage where footprint of the device and cost of it is an important factor.

Raspberry Pi RP2040 Specifications & Features:

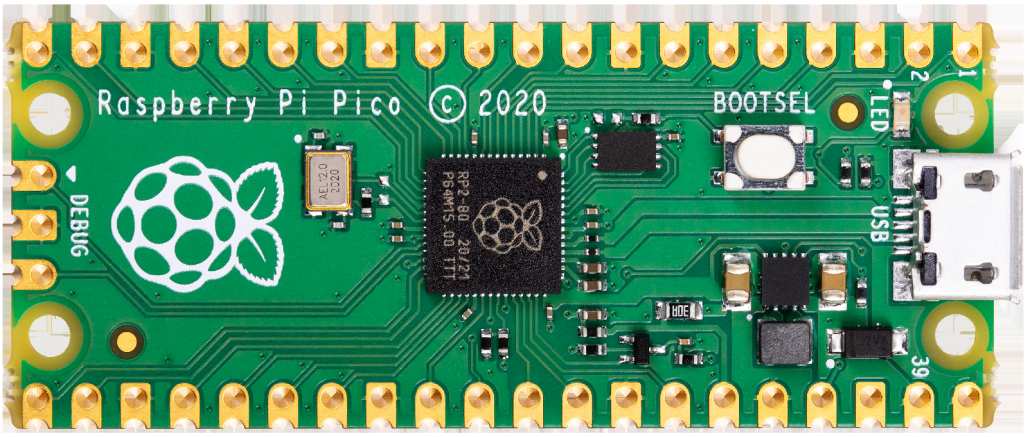
- Dual Cortex ARM M0+ processor cores, up to 133 MHz
- On-chip PLL allows variable core frequency
- 264 kB of embedded SRAM in 6 banks
- External Quad-SPI Flash with eXecute In Place (XIP) and 16kByte on-chip cache
- 30 multifunction GPIO
- 12-bit 500ksps Analogue to Digital Converter (ADC)
- 4 channel ADC, one of which is internally connected to the temperature sensor and 3 channels are available to the user
- 2 × UART, 2 × I2C, 2 × SPI, 16 × PWM channels
- 1 × Timer with 4 alarms, 1 × Real Time Counter
- USB 1.1 Host/Device

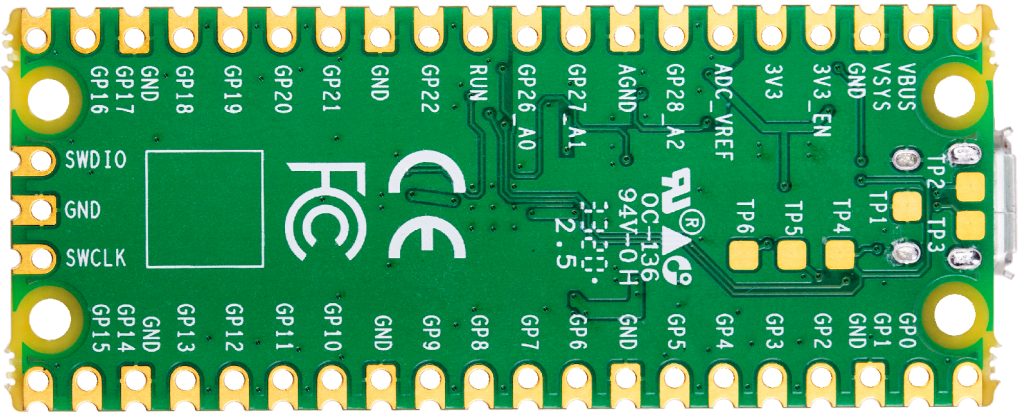


https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

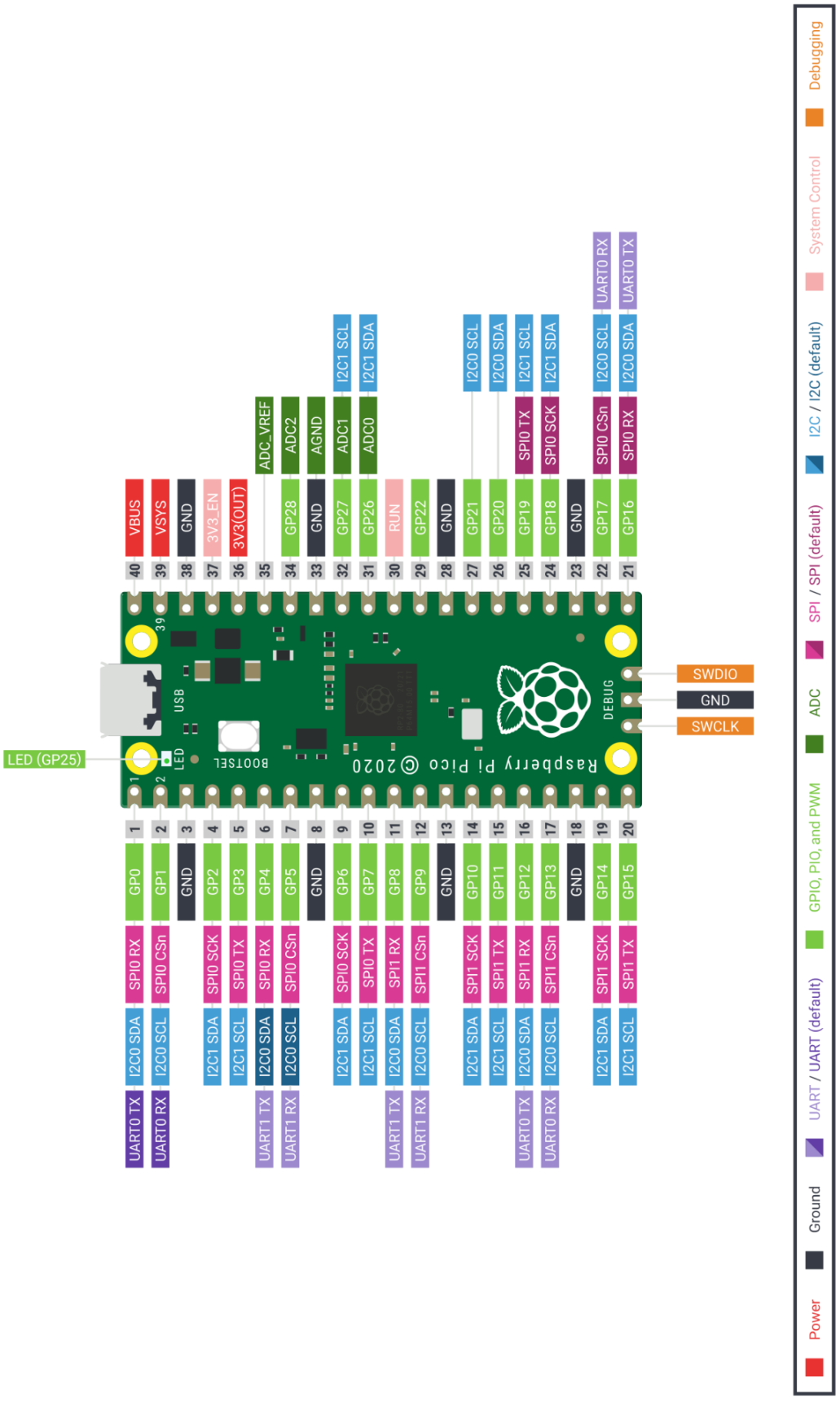
Raspberry Pi Pico - Microcontroller board
Specifications & Features:

- 2MB Flash Memory
- 40 pin board with 26 GPIO among which 23 are digital-only and other 3 are analog capable as ADC input channels
- 3-pin ARM Serial Wire Debug (SWD) port
- Four RP2040 IO are used for internal functions - driving an LED, on-board buck-boost Switched Mode Power Supply (SMPS) power control(generates the required 3.3V to power RP2040 and external circuitry from a wide range of input voltages 1.8V - 5.5V) and sensing the system voltages
- Can be programmed using C/C++ and MicroPython





https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485



https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

RS232:

- RS stands for Recommended Standard
- It is a from of Asynchronous Serial Data Communication
- Point-to-Point Communication
- Defines the voltage levels at which the data transmission occurs
- Uses negative voltage of -3V to -12V(can be up to -25V) for logic 1 and positive voltage of +3V to +13V(can be up to -25V) for logic 0
- Data transmission rate is up to 20 kilobits/second
- Maximum distance of up to 50 feet can be between the DCE and DTE
- Incompatible with TTL voltage levels
- Uses the DB9 cable for the communication
- When communication occurs between DCE - Data Communication Equipment(eg. Modem) and DTE - Data Terminal Equipment(eg. Computer) DB25 and DB9 cable can be used among which DB9 has become the defacto standard
- When communication occurs between two Data Terminal Equipment a Null Modem Cable which is also a 9 pin cable or a Null Modem Adapter along with DB9 cable is required

Lines Drivers & Receivers:

- MC 1488 for conversion of TTL to RS232 voltage level
- MC 1489 for conversion of RS232 to TTL voltage level

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

- MAX 232/MAX 3232 for conversion of TTL to RS232 voltage level and vice versa.

RS485:

- It is a form of Asynchronous Serial Data Communication
- Multi-point interface in which there is one transmitter and up to 32 receivers on the same bus
- Uses differential signaling which eliminates the noise introduced during the transmission of data
- Half-duplex communication when 2 wire form is used and full-duplex communication when 4 wire form is used
- Has two lines A and B respectively where the voltage on the each line is same in magnitude and opposite in polarity
- Since the signal transmission is not with respect to the ground but rather differential signaling is used the noise introduced during the transmission of data can be eliminated when the data is received which largely contributes to the increased distance up to which the data can be sent and received. Which is a major advantage over the RS232
- Data transmission rate is up to 10 megabits/second which is also much higher than the RS232
- Maximum distance of of 4000 feet up to which the data transmission occurs

Lines Drivers & Receivers:

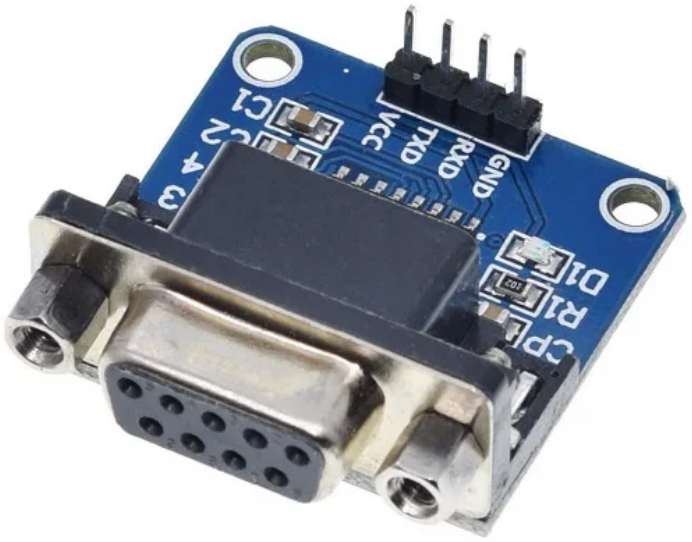
- MAX 485/487 for conversion of TTL to RS485 voltage levels and vice versa

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

RS232 to TTL: MAX 3232 Serial Interface Module:

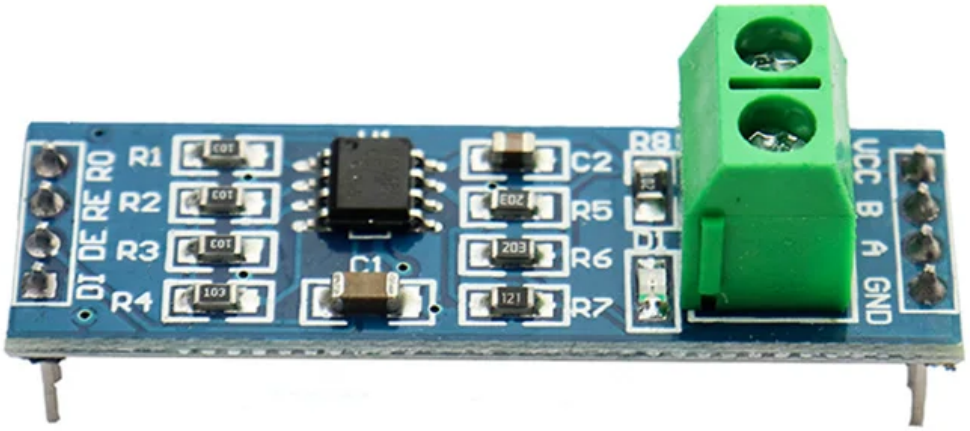
- It uses the IC Chip MAX 3232 for the conversion of TTL to RS232 voltage levels and vice versa
- Operating Voltage: 3.3V - 5.5V
- Operates up to 250 kbit/s



RS485 to TTL: MAX 485 Serial Interface Module:

- It uses the IC Chip MAX 485 for the conversion of TTL to RS485 voltage levels and vice versa
- Operating Voltage: 3.3V - 5.5V
- Operates up to 250 kbits/s

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485



https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

UART:

UART stands for Universal Asynchronous Receiver Transmitter. A UART is a hardware interface device for asynchronous serial communication in which the data format and transmission speeds are configurable. It sends data bits one by one, from the least significant to the most significant, framed by start and stop bits.

Working of our Project:

For the working of the asynchronous serial communication between the two microcontrollers, both the microcontrollers are programmed to use the same baud rate of 9600.

The microcontroller which transmits the data is referred to master and which receives the data is referred to slave throughtout this project report.

The photoresistor is connected across the 3V3 (OUT) which provides the output voltage of 3.3 V and ADC channel 2 of the master. The ADC of the master has the resolution of 12-bit and uses the successive approximation technique for converting the analog value to digital value and the programming environment in which the program executes provides the value as a 16-bit value so we use the conversion factor 3.3 / (2^{16} - 1) which is equal to 3.3 / 65535 to multiply with read value of the ADC to get the actual voltage which is 3.3 V - voltage drop across the photoresistor.

The value is being read by the master for every 2.5 seconds by pausing the execution of the program between reading the values by

the ADC and sent with a precision of up to 2 decimal points as ASCII string with the letter “V” and end of line character appended, eg. “2.95 V\n”. The way of communication between the master and the slave is discussed after discussion how the slave uses the data received from the master.

The slave receives the data from the master approximately for every 2.5 seconds between which it pauses the execution of the program. It differentiates between the individual datum received by identifying the end of line character sent at the end of every value.

The slave is interfaced with a 16 x 2 LCD display module by connecting the RS, RW, EN and 8 data lines with it. Before proceeding to receive the data sent by the master, the slave sents the neccessary commands to the LCD display module to prepare it for displaying the data. After the necessary commands has been sent to the LCD module, the slaves writes the string “Photoresistor” in the first line of LCD and listens if it has received any data from the master and writes it to the second line of the LCD display module.

Communication via RS232:

Both the master and slave are interfaced with a individual RS232 to TTL Serial Interface Modules. The module has four pins, namely V_{CC}, GND, TXD and RXD. The connection for the four pins of the module is same for both the master and slave.

The V_{CC} and GND are connected to a 5V output and GND of a power supply respectively. The module’s TXD and RXD are connected to their respective microcontroller boards to TX and RX respectively.

The two modules are connected via Null Modem cable or a DB9 cable with a Null Modem Adapter through which the data is transmitted from the master to slave. Both the master and the slave uses the hardware UART to send and receive data respectively.

Communication via RS485:

The master and the slave are interfaced with individual RS485 to TTL Serial Interface Modules. The module consists of pins V_{CC}, GND, A, B, DI, RO, DE and RE. The module's pins V_{CC} and GND are connected to the 5V output and GND of the power supply.

The A and B are the differential voltage lines which need to be connected to the common bus lines A and B respectively of upto 32 devices where there is one transmitter/master and one or more receivers/slave devices.

The DE is Driver Input Enable signal which is set to logic level 1 in the transmitter and reset to logic level 0 in the receiver, similarly the RE is Receiver Output Enable signal but it is low enabled signal so in the receiver it needs to reset to logic level 0 and set to logic level 1 in the transmitter. Typically the signals DE and RE are short together since they require logic level 1 in the transmitter side and logic level 0 in the receiver side.

The DI is Driver Input signal which needs to be connected to the TX pin of the UART and RO is Receiver Output signal which needs to be connected to the RX pin of the UART. The connections of DI and RO are same between the slave/master and the module.

The master sets the DE and RE to logic level 1 and transmits data via the UART to the module and the slave resets the DE and RE to logic level 0 and receives the data via the UART from the module.

The data transmissions occurs between the modules by connecting the A line of master serial module and B line of master serial module to A line of slave serial module and B line of slave serial module respectively.

```
'''
Master Microcontroller Program
'''

from machine import Pin, ADC, UART
from time import sleep_ms

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))
adc = ADC(2)
conversion_factor = 3.3 / 65535
DE_RE = Pin(2, Pin.OUT) # only used for RS485

DE_RE.value(1)
while True:
    reading = adc.read_u16() * conversion_factor
    data = "{:.2f}".format(reading) + ' V\n'
    print(data)
    uart.write(bytes(data, 'utf-8'))
    sleep_ms(2500)
```

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

```
'''
Slave Microcontroller Program
'''

from machine import UART, Pin
from time import sleep_ms

uart = UART(0, baudrate=9600, tx=Pin(16), rx=Pin(17))
commands = [0x38,0x0E,0x01,0x06,0x80]
str1 = "Photoresistor"

RS = Pin(0, Pin.OUT)
RW = Pin(1, Pin.OUT)
EN = Pin(2, Pin.OUT)

DE_RE = Pin(3, Pin.OUT)
```

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

```
data_pins = [Pin(6, Pin.OUT), Pin(7, Pin.OUT),
             Pin(8, Pin.OUT), Pin(9, Pin.OUT),
             Pin(10, Pin.OUT), Pin(11, Pin.OUT),
             Pin(12, Pin.OUT), Pin(13, Pin.OUT)]




DE_RE.value(0) # used only for RS485


def send_cmd(cmd):
    byte = list(map(int, reversed(bin(cmd)[2:])))
    byte = byte + (8 - len(byte)) * [0]
    for i in range(8):
        data_pins[i].value(byte[i])
    RS.value(0)
    RW.value(0)
    EN.value(1)
    sleep_ms(1)
    EN.value(0)


def send_data(data):
    byte = list(map(int, reversed((bin(ord(data)))[2:])))
    byte = byte + (8 - len(byte)) * [0]
```

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

```
    for i in range(8):
        data_pins[i].value(byte[i])
    RS.value(1)
    RW.value(0)
    EN.value(1)
    sleep_ms(1)
    EN.value(0)




for i in commands:
    send_cmd(i)
    sleep_ms(2)


for i in str1:
    send_data(i)
    sleep_ms(2)


while True:
    if uart.any() > 0:
        data = uart.readline()
    else:
        continue
```

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

```
    send_cmd(0xC0)

    data_str = data.decode('utf-8')


    for i in data_str[:-1]:

        send_data(i)

        sleep_ms(2)


    sleep_ms(2500)
```

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485

Application of our Project:

Our project can be used for automatic switching of street lights and switching of lights in houses. This can be achieved by calibrating the voltage and the light intensity at which the lights need to be turned ON and OFF. Since the microprocessor we used is powerful because of having dual cores and has more amount of RAM it can also be used for Machine Learning Applications like by interfacing it with a camera module to monitor the traffic on the roads by placing it on the junction of the roads and sending the corresponding traffic lights to be glown to other traffic light poles placed on the multiple sides on the roads via RS232 or RS485, typically by RS485 because of its elimination of noise since it uses differential signaling which will be very useful in noise prone area like outdoor highways/roads.

Result:

Our project has successfully achieved our objective of transferring the process data acquired from a microcontroller to the other microcontroller using Universal Asynchronous Receiver Transmitter by RS232 and RS485 defined electrical voltages.

https://github.com/pdmnhn/Raspberry-Pi-Pico-RS232-RS485