QuecPython BUS User Guide

LTE Standard Module Series

Version: 1.0.0

Date: 2020-11-10

Status: Preliminary

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit: http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm Or email to
support@quectel.com.

General Notes

Quectel offers the information as a service to its customers. The information provided is based upon customers' requirements. Quectel makes every effort to ensure the quality of the information it makes available. Quectel does not make any warranty as to the information contained herein, and does not accept any liability for any injury, loss or damage of any kind incurred by use of or reliance upon the information. All information supplied herein is subject to change without prior notice.

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

About the Document

History

Revision	Date	Author	Description
-	2020-11-10	Rivern/Kenney	Creation of the document
1.0.0	2020-11-10	Rivern/Kenney	Preliminary

Contents

Figure Index

1 Introduction

Bus refers to a common data channel between computer equipment and equipment to transmit information. The bus is a communication line connecting multiple devices in the computer hardware system. An important feature of it is that it is shared by all devices on the bus and can connect multiple devices in the computer system to the bus. If it is a dedicated signal connection between two devices or devices, it cannot be called a bus.

This document mainly introduce the basic features and approaches of the four peripheral buses, including ADC, UART, SPI and I2C.

2 Basic Features and Approaches

2.1. ADC

ADC basically converts physical variables which are analog in nature to digital signal for processing. They have high conversion efficiency and requires less power. Examples of physical variables include audio signals, temperature, pressure etc. The ADC input pin on QuecPython EVB is shown as follows.

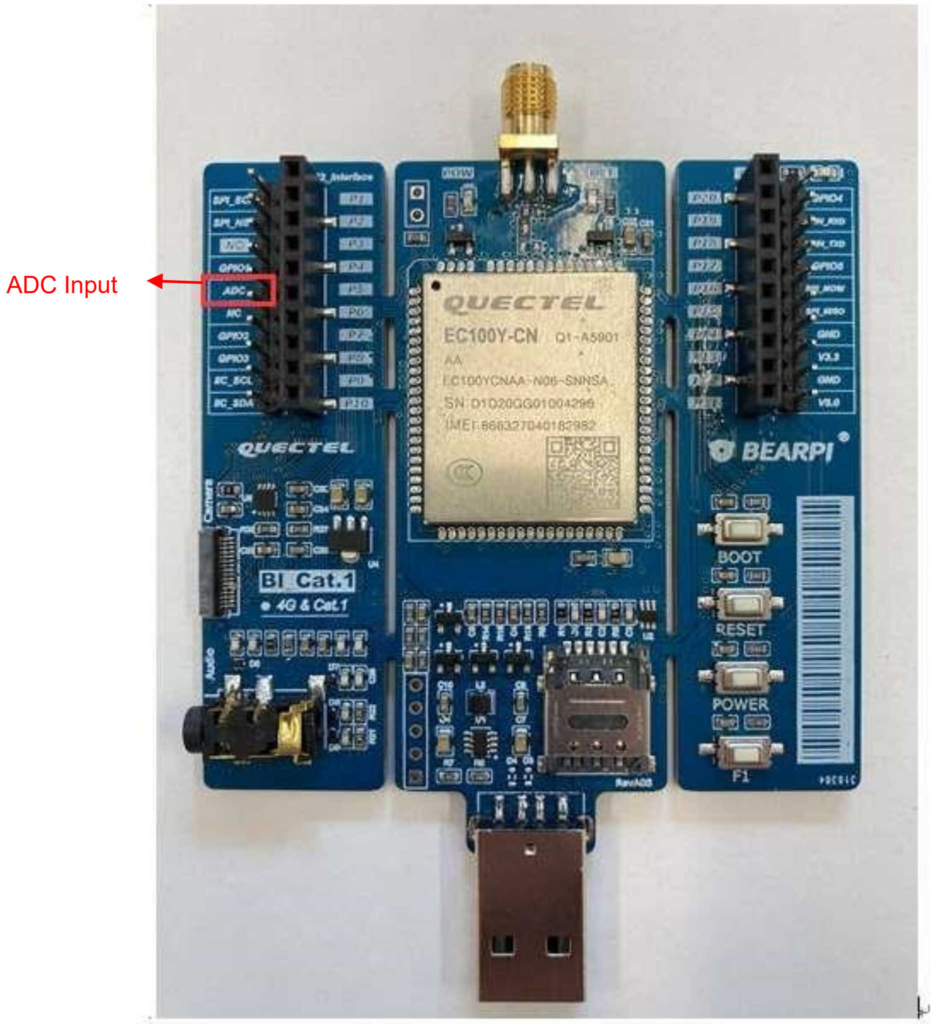


Figure 1: ADC Input Pin

Connect GPIO1 with ADC serial port, and in QuecPython, read the voltage of ADC channel by *misc* module, set the value of PIN by method *Pin.write(value)* of *Pin* class in *machine* module. For more details, refer to *Quectel_QuecPython_Class_Library_Introduction*.

```
from machine import Pin from
misc import ADC
adc = ADC adc.open()
gpio1 = Pin(Pin.GPIO1, Pin.OUT, Pin.PULL_DISABLE, 0)
gpio1.read() # Get the level of GPIO. gpio1.write(1) # Set
GPIO1 output in high-level.  adc.read(ADC.ADC0)
gpio1.write(0) # Set GPIO1 output in low-level.
adc.read(ADC.ADC0)
```

From the command line operation results, it can be seen that the ADC channel voltage changes to 1.8 V.

Figure 2: The Change for ADC Voltage Value

2.2. UART

UART is a serial asynchronous transceiver protocol, which is widely used. UART working principle is to transmit the binary bits of data bit by bit. In UART communication protocol, the high level of the status bit on the signal line represents 1, and the low level represents 0. Of course, when two devices use UART serial communication, the transmission rate and some data bits must be agreed.

The hardware connection is relatively simple, only 3 wires are required. Please make level conversion before connecting if the UART level range of the two devices is inconsistent.

- RX: Receive data. Input to receive the data from another transmitter.
- TX: Receive data. The reverse of RX, this is where the terminal equipment is transmitting serial data, using the same format and protocol that the receiver is expecting.
- GND: Signal Ground. What it does is try to make a common "ground" reference between the equipment that is being connected to compare the voltages for the other signals.

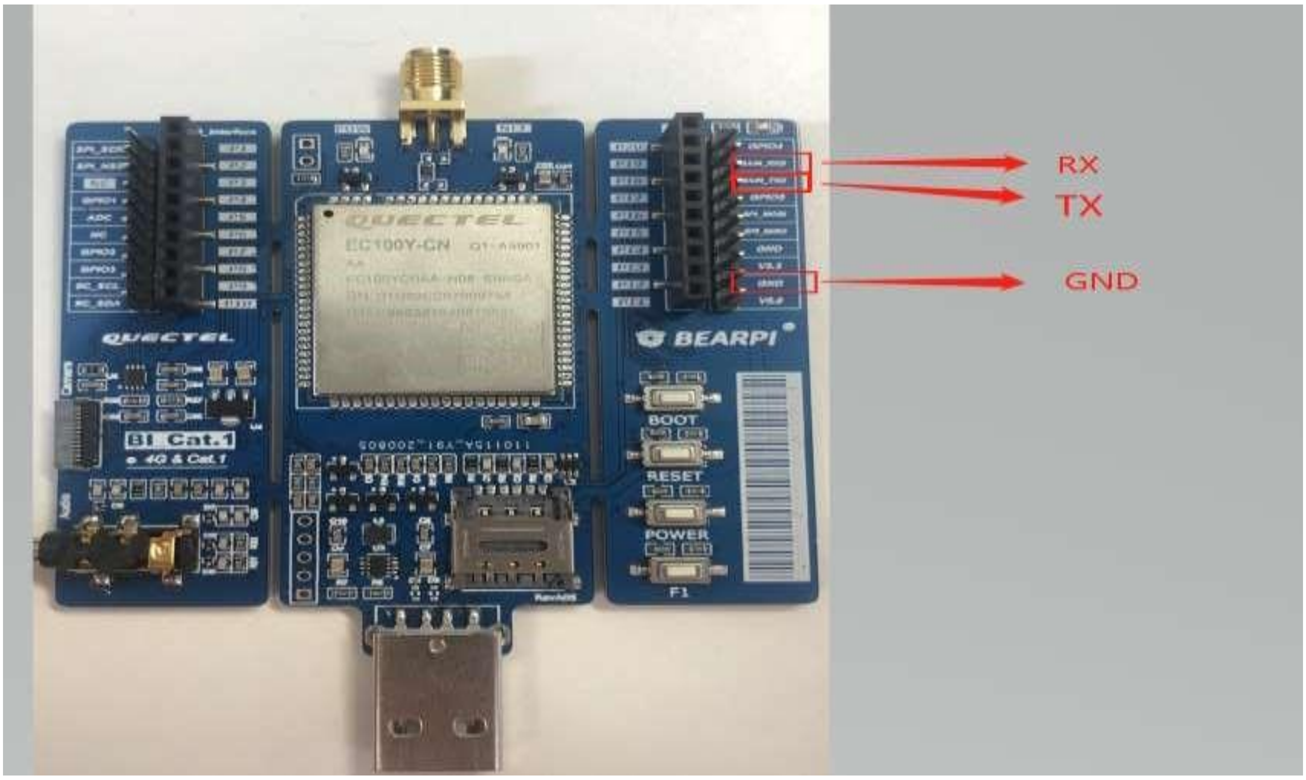


Figure 3: UART Hardware Connection

When using the UART to transmit data, connect the TX with RX on the other, and connect the RX with TX on the other. And in QuecPython, the UART data transmission can be fulfilled with the *UART* class in *machine* module. For more details, refer to *Quectel_QuecPython_Class_Library_Introdcution*.

```
>>>
>>> from machine import UART
>>> uart1 = UART(UART.UART2,9600,8,0,1,0)
>>> uart1.any()
8192
>>> uart1.read(1024)
b'$GPGGA,000102.262,,,,,0,0,,,M,,M,,*4D\r\n$GPGLL,,,,,000102.262,V,N*7F\r\n$GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n$G
00103.262,V,N*7E\r\n$GPGSA,A,1,,,,,,,,,,,,,,,*1E\r\n$GPGSV,1,1,00*79\r\n$GPRMC,000103.262,V,,,,,0.00,0.00,060
GSV,1,1,00*79\r\n$GPRMC,000104.262,V,,,,,0.00,0.00,060180,,,N*41\r\n$GPVTG,0.00,T,,M,0.00,N,0.00,K,N*32\r\n$G
80,,,N*40\r\n$GPVTG,0.00,T,,M,0.00,N,0.00,K,N*32\r\n$GPGGA,000106.262,,,,,0,0,,,M,,M,,*49\r\n$GPGLL,,,,,0001(
GGA,00'
>>>
```

Figure 4: Code Example for UART API

2.3. SPI

SPI is a very common communication protocol used for two-way communication between two devices. A standard SPI bus consists of 4 signals, namely SS, SCK, MOSI, MISO.

- **SS**: When multiple SPI devices are connected to the MCU, the SS of each device is connected to a separate pin of the MCU, while for the other SCK, MOSI, and MISO, multiple devices are connected in parallel to the same bus. And valid in low-level.
- **SCK**: The clock pulses which synchronize data transmission generated by the master. Different devices support different clock frequencies, such as SPI clock frequency of STM32 up to f PCLK / 2.
- **MOSI**: The Master line for sending data to the peripherals.
- **MISO**: The Slave line for sending data to the master.

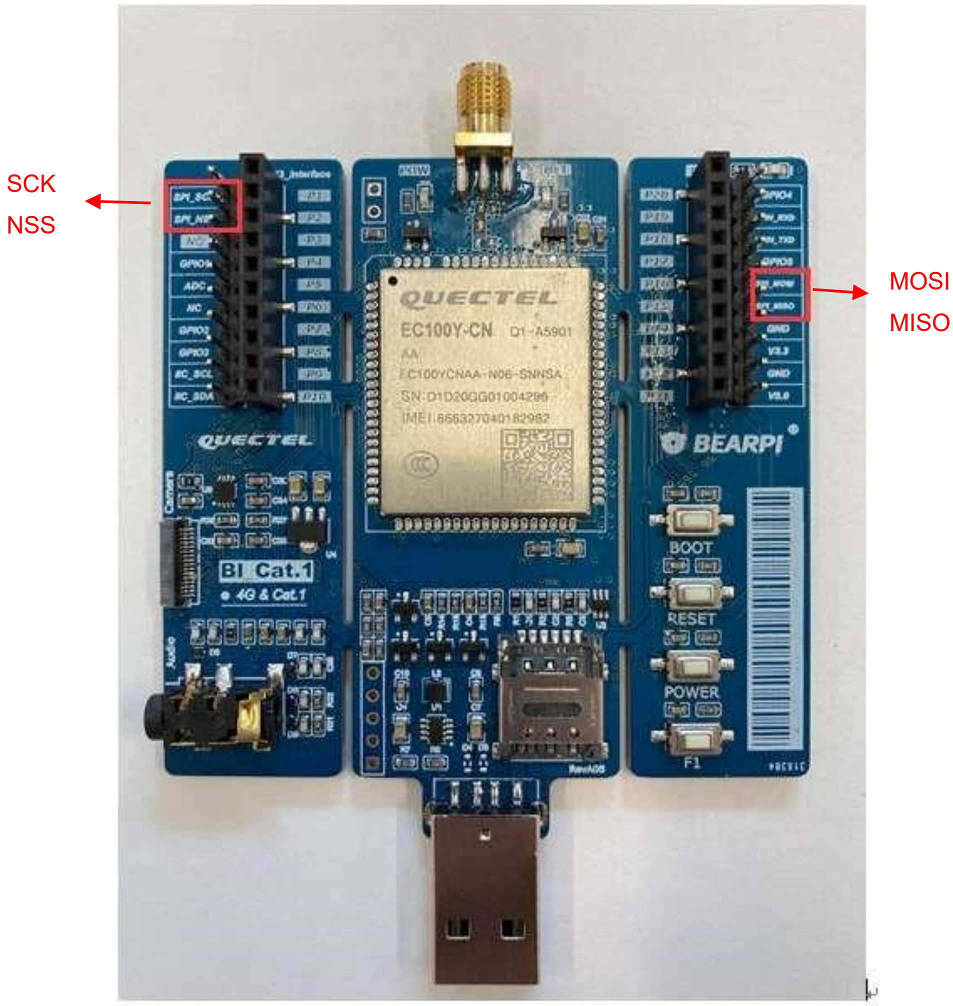


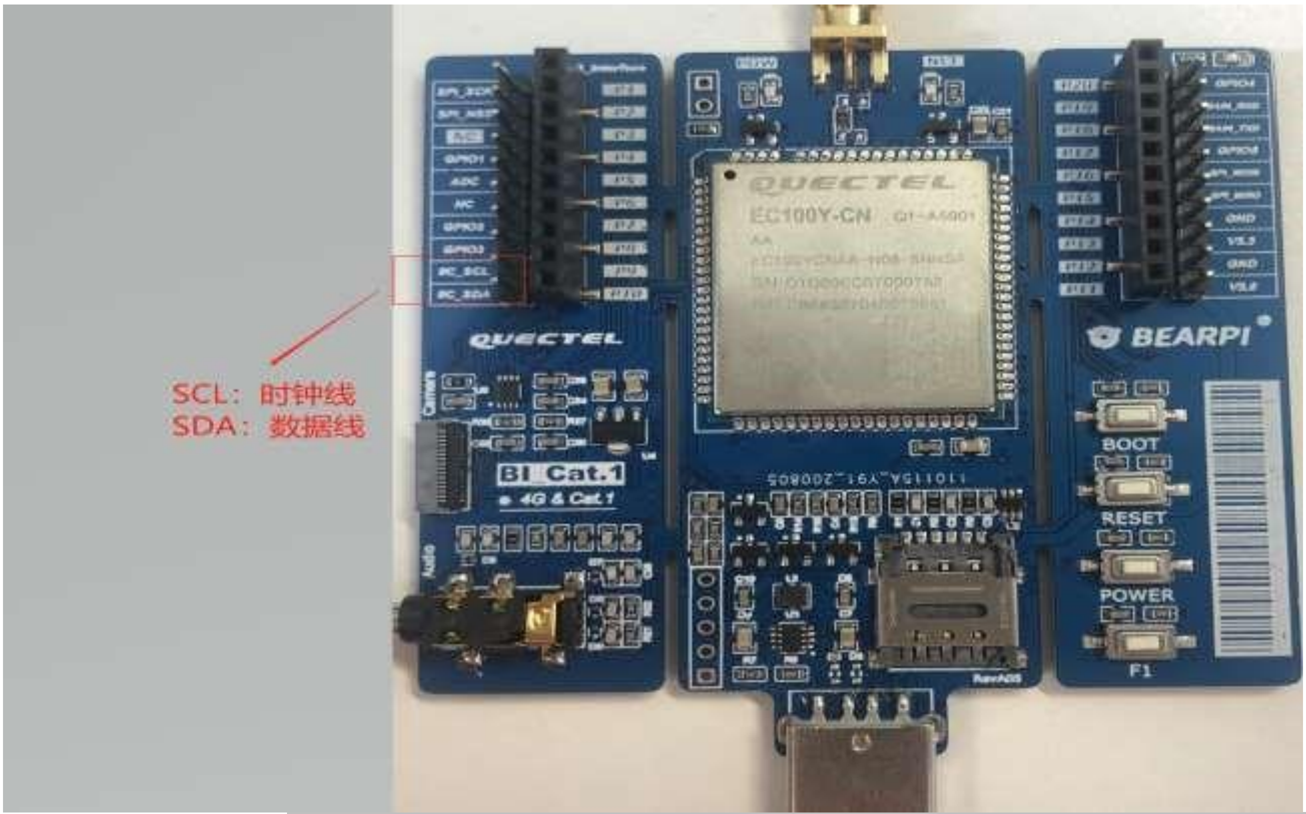
Figure 5: SPI Hardware Connection

2.4. I2C

I2C uses only two wires: SCL (serial clock) and SDA (serial data).

- SCL: The clock signal is transmitted from the master device to the slave device
- SDA: A channel for data transfer between master and slave devices.

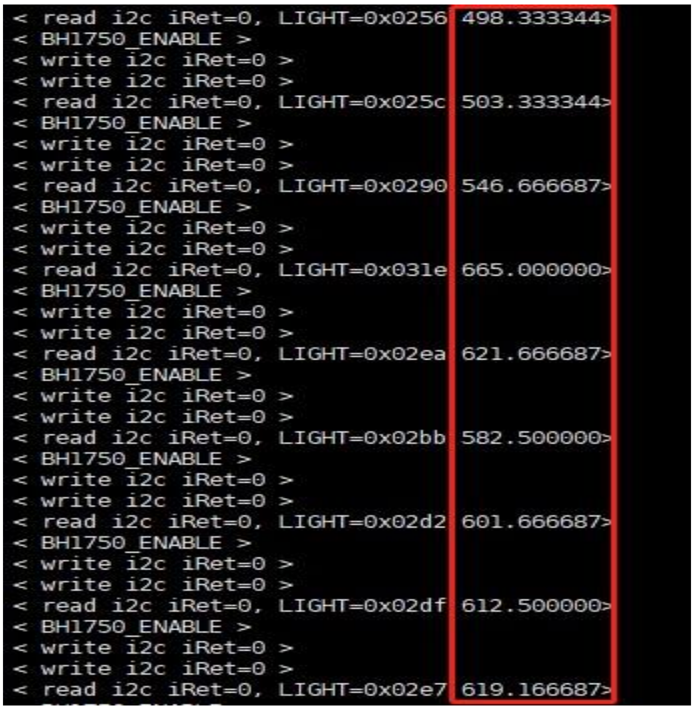
I2C is a serial communication protocol, so data is transferred bit by bit along a single wire (the SDA line). Like SPI, I2C is synchronous, so the output of bits is synchronized to the sampling of bits by a clock signal shared between the master and the slave. The clock signal is always controlled by the master.



SCL
SDA

Figure 6: I2C Hardware Connection

The following data takes the connection between I2C and light sensor as an example.



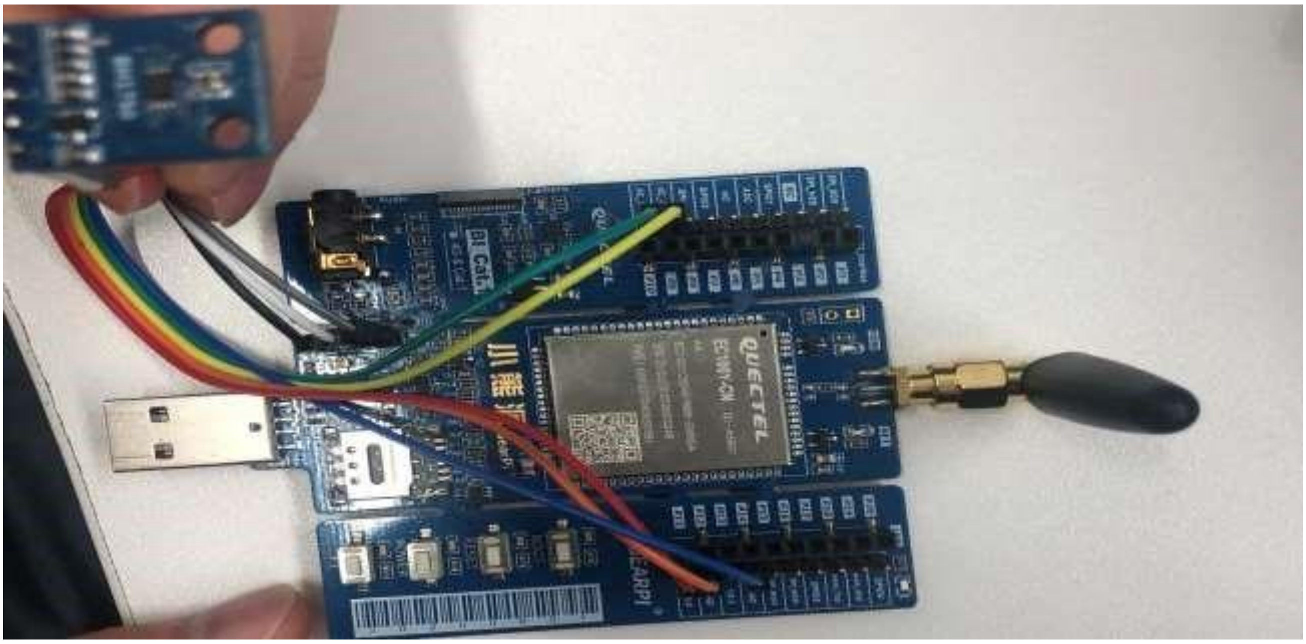


Figure 7: The Connection between the I2C and Light Sensor

3 Terms and Abbreviations

Table 1: Terms and Abbreviations

Abbreviation	Description
ADC	Analog-to-Digital Converter
API	Application Programming Interface
CPU	Central Processing Unit
UART	Universal Asynchronous Receiver/Transmitter
GND	Ground
IC	Integrated Circuit
I2C	Inter-Integrated Circuit
LCD	Liquid Crystal Display
MCU	Microprogrammed Control Unit

MISO	Master In Slave Out
MOSI	Master Out Slave In
RX	Receive
TX	Transmit
SCK	Serial Clock
SCL	Serial Clock Line
SDA	Serial Data Line
SPI	Serial Peripheral Interface
SS	Slave Select