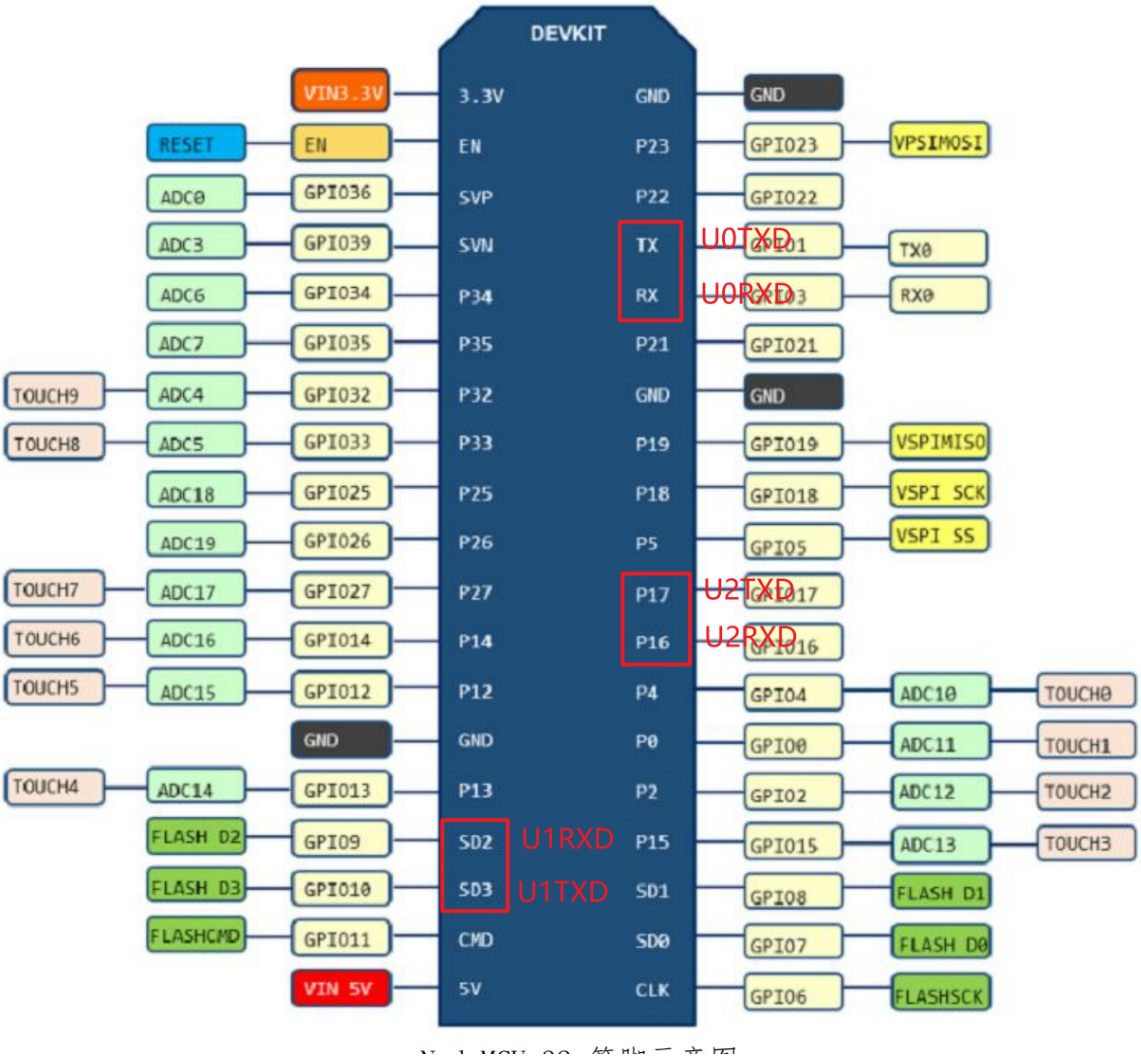
ESP32上的三套串口

在esp32规格说明书中提到了三套串口，分别是UART0，UART1，UART2。这三套串口所对应的引脚如下（U0, U1, U2分别代表UART0, UART1, UART2）：



根据Micro Python提供的文档，如果想要在Micro Python中创建UART0, UART1, UART2的对象，则应该添加分别添加如下的代码：

	UART0	UART1	UART2
tx	1	10	17
rx	3	9	16

```
# UART0
import machine
uart = machine.UART(1, baudrate=115200, tx = 1, rx = 3)
```

```
# UART1
import machine
uart = machine.UART(1, baudrate=115200, tx = 10, rx = 9)
```

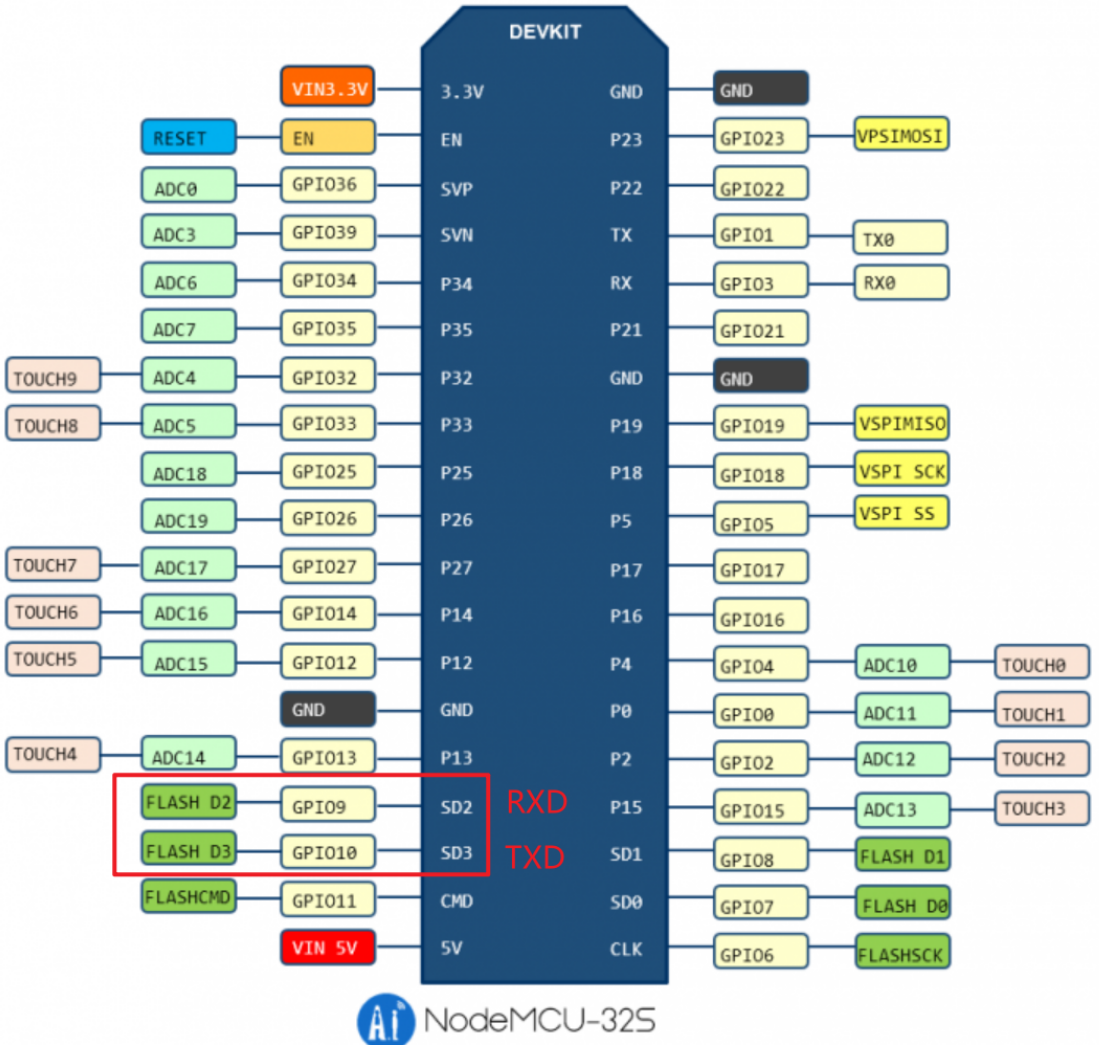
```
# UART2
import machine
uart = machine.UART(1, baudrate=115200, tx = 17, rx = 16)
```

但是经过实测发现，如果将103的RXD和TXD分别连接到ESP32的**UART0**的TXD和RXD，则即使ESP32在此之前未经过任何处理，此时103板通过串口发送的数据都会直接出现在ESP32的python命令行中，并且ESP32进入了一种近似阻塞的状态，无法输入或执行任何指令，进入了一种单工状态，不建议使用。

ESP32使用UART1连接103

连线

ESP32	103
GND	GND
RXD	TXD
TXD	RXD



ESP32设置

```
import _thread
import machine
import time
getDataFlag = 1 # 用flag来控制线程的开启和关闭
uart = machine.UART(1, baudrate=115200, tx = 10, rx = 9) # 创建串口对象
# 定义接收103数据的线程函数
def getData():
    while getDataFlag:
        if uart.any(): # 如果串口接收到了数据
            data = uart.read() # 读取串口数据
            print("Received data:", data) # 打印

_thread.start_new_thread(getData, ()) # 添加并开启线程
```

经过上述设置之后，ESP32就开启了一个用于接收串口数据的线程。如果此时想要发送数据到串口，则：

```
uart.write("Hello World\n")
```

如果想要停止读取数据，只需：

```
getDataFlag = 0
```

103设置

103的demo程序主要分为如下几个部分：

- 全局变量设置
- 按钮中断
- 接收串口数据中断
- 主函数

```
// 首先把printf定义为向串口输出
#include <stdio.h>
int __io_putchar(int ch){
    uint8_t c=ch;
    HAL_UART_Transmit(&huart1, &c, 1, 100);
    return ch;
}
```

```
// 全局变量设置
uint8_t c; // 用来接收串口数据
uint8_t charArray[1000]; // 用于存放串口数据
int charIndex = 0; // 存放串口数据所用到的index
uint32_t num = 0; // 按下按钮发送的变量，用于检测程序的正确性
```

```
// 按钮中断
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin){
    if(GPIO_Pin == BTN_Pin){
        printf("num is %d\n", num);
    }
}
```

```
// 接收串口数据中断
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart->Instance == USART1)
    {
        // 如果遇到换行，则将结果发送给ESP32
        if (c == '\n')
        {
            num++;
            charArray[charIndex] = '\0';
            charIndex = 0;
            printf("%s\n", (char*)charArray); // 发送数据
        }
        else
        {
            charArray[charIndex++] = c; // 存放字符
        }
        HAL_UART_Receive_IT(&huart1, &c, 1); // 开启下一次接收
    }
}
```

```c
// 主函数
int main()
{
    // 开始接收一个字节
    HAL_UART_Receive_IT(&huart1, &c, 1);
    while(1)
    {
        // LED灯闪烁，主要用于验证程序的正确性
        HAL_UART_Receive_IT(&huart1, &c, 1);
        HAL_Delay(1000);
    }
}
```

注意

- 103不要每接收一个字符就执行发送这种消耗大量时间的动作，会出bug，最好读到某个设定的终结符再进行处理
- UART2没有测试过，猜测和UART1差别不大，但是UART0用不了一点