上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

S800板第三次实验(上)

准备知识的学习

- UART
- NVIC

UART

Universal Asynchronous Receiver / Transmitter

通用异步收发器

UART简介

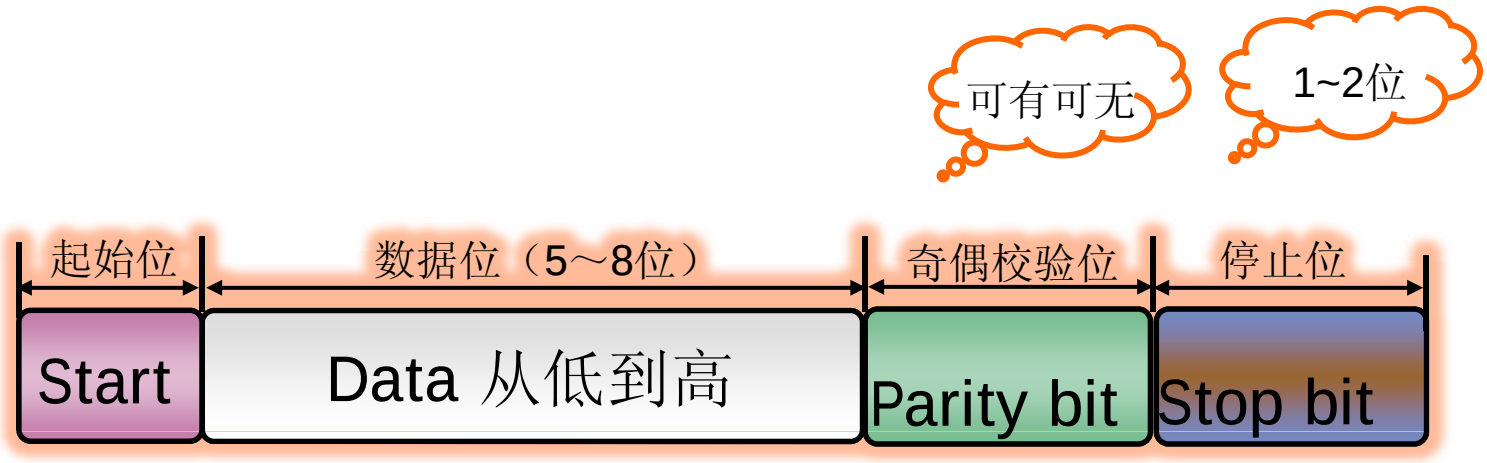
- **UART是一种典型的异步串行通信接口，支持双向通信，可以实现全双工数据传输。**
  - 在早期的通用计算机和PC上，串口几乎是标准配置。而现代PC因为串口的通信速率等硬件特性已经不适合PC的要求，取而代之的是“通用串行通信口”，也就是常说的USB接口。如果要在现代PC上使用串口，必须使用一个USB——串口的硬件转换器，安装相应的驱动后，在PC上会显示一个USB虚拟串口设备。
  - 在嵌入式设计中，因为串口极低的资源消耗、较好的可靠性、简洁的协议以及高度的灵活性，使其非常符合嵌入式设备的应用需求。几乎所有的MCU都把UART作为一个最基本的通信接口，用来实现与其它嵌入式设备或PC的数据通信。

UART简介

- UART是实现设备间异步串行通信的关键模块。
- UART处理数据总线和串口之间的串/并、并/串转换；
- 其过程是：CPU先把准备发送出去的数据写入到UART的数据寄存器端口，再通过FIFO（First Input First Output，先入先出队列）传送到串行发送器，若是没有FIFO，则CPU每次只能写一个数据到UART的数据寄存器端口。

UART通信字符帧格式

- UART通信时需保证收发两端的帧格式一致，否则会出现通信错误。



UART常用电路接口说明

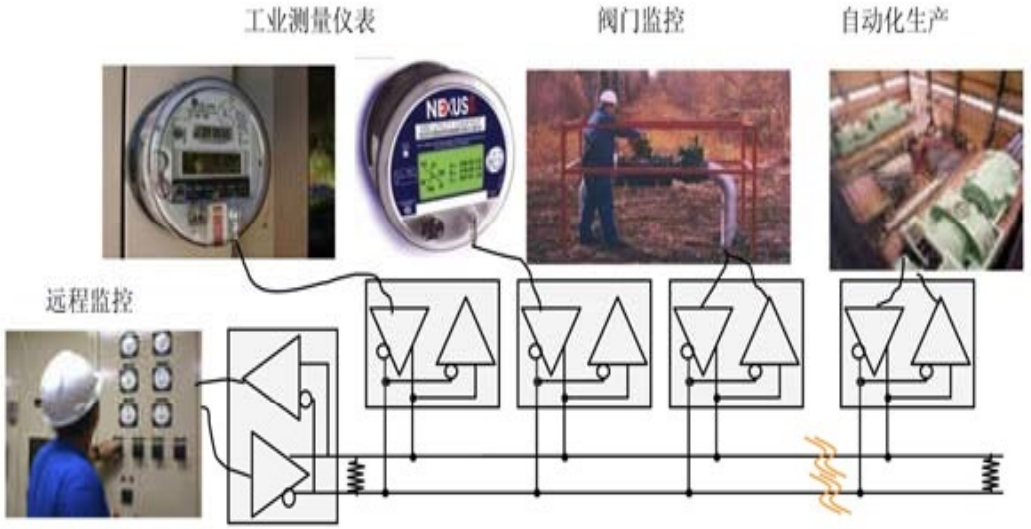
- UART只定义了串行通信的逻辑层，即字符传送的格式与速率，但对于利用于UART进行传输的物理电平接口需要另外定义，因此根据接口电平的不同与控制方式的不同，产生了RS-232与RS-485标准。

RS-232

- RS-232是PC机与设备通信里应用最广泛的一种串行接口。它被定义为一种在低速率串行通讯中增加通讯距离的单端标准，由于其最大通信距离的限制，因此它常用于本地设备之间的通信。
- RS232标准定义逻辑“1”信号相对于地为-3到-15伏，而逻辑“0”相对于地为+3到+15伏。所以，当一个微控制器中的UART相连于PC时，它需要一个RS232驱动器来转换电平。
- 最高传输率20Kb/s
- 最大传输距离约为15m
- 单端传输，共模抑制比低

RS-485

- RS-485是一种常用远距离和多机通信的串行接口。RS-485只是定义电压和阻抗
- 与TTL电平兼容
- 传输距离可达1200m
- 双端传输，共模抑制比高
- 需要RS485驱动器来将单端信号转换为双端信号

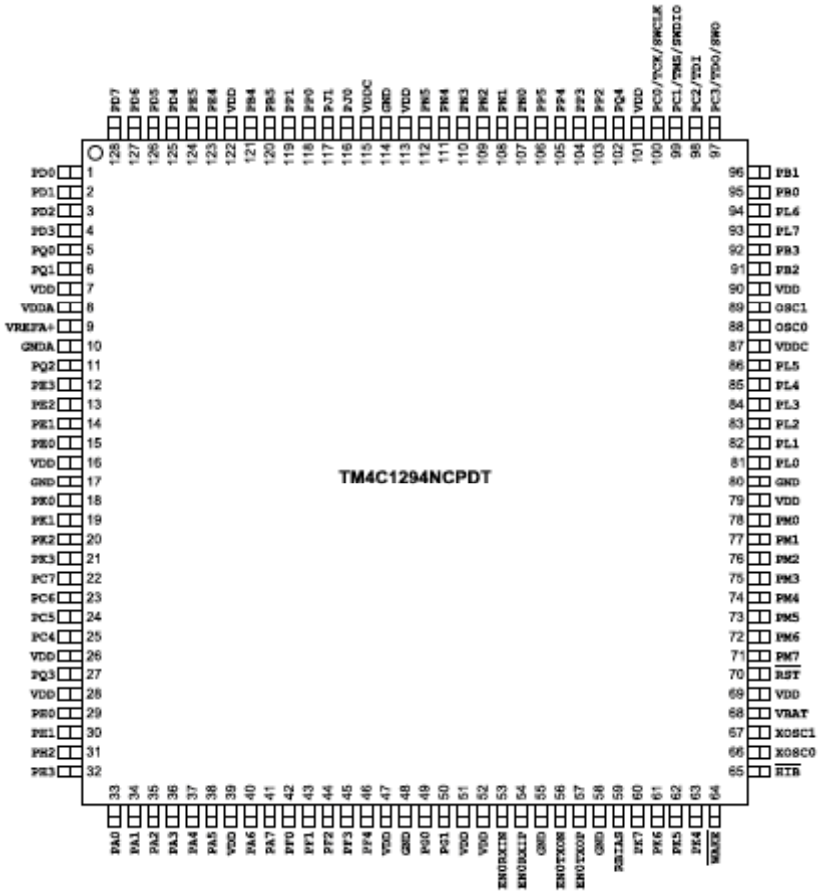


TM4C1294NCPDT的UART

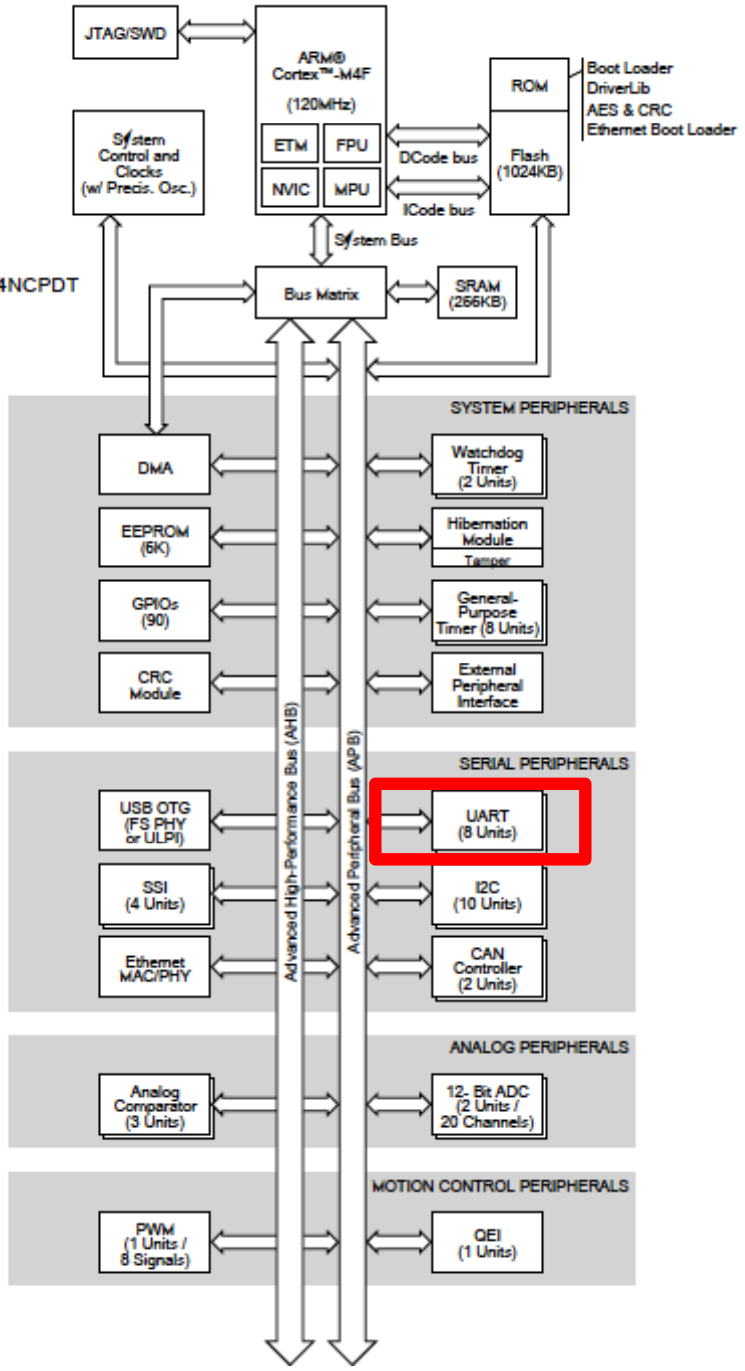
TM4C1294NCPDT微控制器集成了8个标准的UART模块，具有以下特征：

- 可编程的波特率发生器，在常规模式（16 分频）下最高可达到 5Mbps，在高速模式（8分频）下最高可达10Mbps；
- 每路UART就具有相互独立的16×8 发送（TX）FIFO 和接收（RX）FIFO，可降低中断服务对CPU 的占用；
- FIFO 触发深度有如下级别可选：1/8、1/4、1/2、3/4 ，和7/8；
- 含标准的异步通讯位：起始位、停止位、奇偶校验位；
- 线中止(Line-break)的产生与检测；
- 完全可编程的串行接口特性：
  - 可包含5、6、7，或8 个数据位
  - 可产生/检测奇偶校验位，支持偶校验位、奇校验位、粘着校验位，或无校验位
  - 可产生1 或2 个停止位

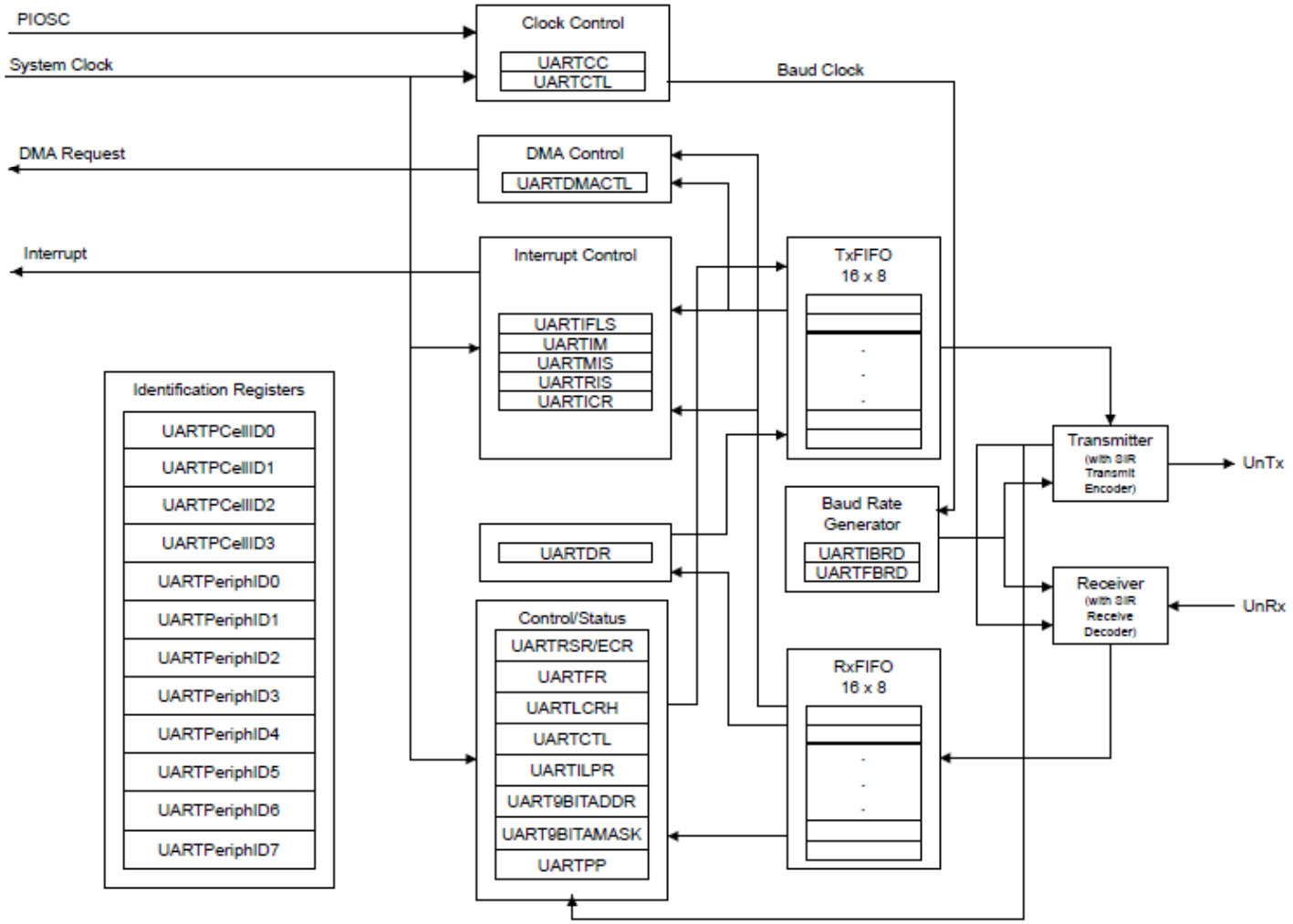
High-Level Block Diagram



TM4C1294NCPDT

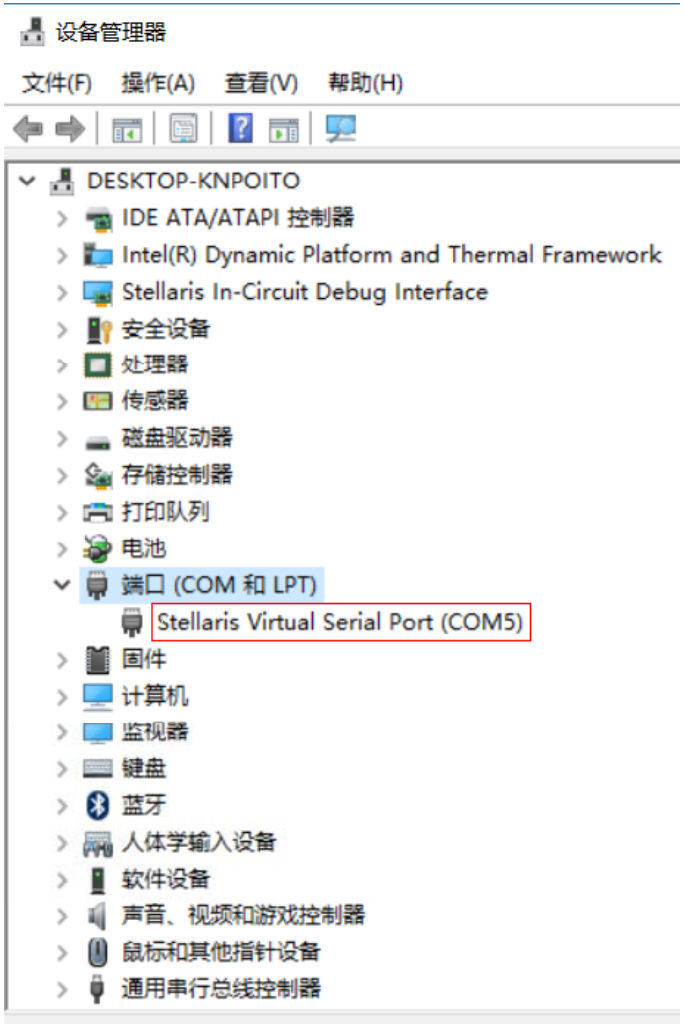


UART控制器结构图


PIOSC
System Clock
DMA Request
Interrupt
Clock Control
UARTCC
UARTCTL
Baud Clock
DMA Control
UARTDMACTL
Interrupt Control
UARTIFLS
UARTIM
UARTMIS
UARTRIS
UARTICR
TxFIFO
16 x 8
Identification Registers
UARTPCellID0
UARTPCellID1
UARTPCellID2
UARTPCellID3
UARTPeriphID0
UARTPeriphID1
UARTPeriphID2
UARTPeriphID3
UARTPeriphID4
UARTPeriphID5
UARTPeriphID6
UARTPeriphID7
UARTDR
Control/Status
UARTRSR/ECR
UARTFR
UARTLCRH
UARTCTL
UARTILPR
UART9BITADDR
UART9BITAMASK
UARTPP
Baud Rate
Generator
UARTIBRD
UARTFBRD
RxFIFO
16 x 8
Transmitter
(with SIR
Transmit
Encoder)
UnTx
Receiver
(with SIR
Receive
Decoder)
UnRx

EK-TM4C1294XL的虚拟UART

- 将EK-TM4C1294XL通过MICRO-USB线与电脑连接后，正常情况（驱动已安装好） PC端会至少看到2个设备：
  - 1个是ICDI在线调试工具
  - 第二个是虚拟串口
- 这个虚拟串口可以用来作为MCU UART 正常通讯或是作为BOOTLOAD使用。
- 在默认跳线时，此虚拟串口与TM4C1294NCPDT CPU的PA0(U0RX)，PA1(U0TX）相连接，即占用了U0的串口
- 任意选择一款PC端串行口通讯软件，在正确配置了控制板的UART口后即可实现PC与控制板的通讯
- 设备管理器截图如右所示



UART初始化配置

- 引脚配置
  - GPIO引脚复用
    - U0RX-PA0
    - U0TX-PA1

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);            //Enable PortA
while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA));
//Wait for the GPIO moduleA ready
GPIOPinConfigure(GPIO_PA0_U0RX);
// Set GPIO A0 and A1 as UART pins.
GPIOPinConfigure(GPIO_PA1_U0TX);
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
```

UART初始化配置

- 波特率及帧格式设置

```
// Configure the UART for 115,200, 8-N-1 operation.
UARTConfigSetExpClk(UART0_BASE, ui32SysClock, 115200,
    (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE
    |UART_CONFIG_PAR_NONE));
```

- FIFO配置（可选）

函数UARTConfigSetExpClk

⌘ 驱动库手册：P568

功能	设置一个UART的参数。
原型	Void **UARTConfigSetExpClk**(uint32_t **ui32Base**, uint32_t **ui32UARTClk**, uint32_t **ui32Baud**, uint32_t **ui32Config**)
参数	**ui32Base** 为UART端口的基地址。 **ui32UARTClk** 为提供给UART模块的时钟频率。 **ui32Baud** 为波特率。 **ui32Config** 为端口数据格式（数据位个数、停止位个数，以及校验位）
说明	该函数配置UART以特定的数据格式运行。 参数**ui32Config**是三个值的逻辑或：数据位的个数、停止位的个数、校验位。UART_CONFIG_WLEN_8, UART_CONFIG_WLEN_7, UART_CONFIG_WLEN_6, and UART_CONFIG_WLEN_5分别确定每字节中8-5位的数据长度。

函数UARTConfigSetExpClk

⌘ 续上页

说明	UART_CONFIG_STOP_ONE和UART_CONFIG_STOP_TWO分别确定停止位是是一位还是两位。UART_CONFIG_PAR_NONE, UART_CONFIG_PAR_EVEN, UART_CONFIG_PAR_ODD, UART_CONFIG_PAR_ONE, and UART_CONFIG_PAR_ZERO 确定校验方式。外设时钟与处理器时钟相同。

UART数据收发原理

- UART通过TX脚发送数据，通过RX脚接收数据，因此可以实现全双工收发。
- 在数据收发的过程中，通常采用阻塞式和非阻塞式两种方式来进行。
- 阻塞式即在发送或接收数据过程中，始终查询状态，占用CPU时间，只有发送或接收结束后才退出。
- 非阻塞式即在执行到发送或接收函数时，发送函数仅仅将数据推送给寄存器，并不保证传送成功，如果能传送则返回TRUE，否则返回FALSE；接收函数仅仅检查接收状态并返回，如果没有收到数据则返回FALSE，如果收到则返回TRUE。

阻塞式编程

- 数据发送
  - 将数据发送到TX FIFO，如果FIFO没有空，一直等待到FIFO有空

    UARTCharPut(UART0_BASE，0x41)；

    UARTCharPut(UART0_BASE，‘A’)；
- 数据接收
  - 检查接收FIFO，直到接收到一个数据才返回

    receive_data = UARTCharGet(UART0_BASE)；

函数UARTCharPut

⌘ 驱动库手册：P565

功能	等待将一个字符发送到指定端口。
原型	Void **UARTCharPut**(uint32_t **ui32Base**, unsigned char **ucData**)
参数	**ui32Base** 为 UART端口的基地址。 **ucData** 为待发送的字符。
说明	该函数将字符**ucData**发送到指定端口的发送FIFO。如果发送FIFO没有空间，则该函数一直等待，直到操作成功。
返回值	无

函数UARTCharGet

⌘ 驱动库手册：P564

功能	等待指定端口的一个字符。
原型	int32_t **UARTCharGet**(uint32_t **ui32Base**)
参数	**ui32Base** 为UART端口的基地址。
说明	该函数从指定端口的接收FIFO获取一个字符。如果FIFO暂时没有字符，则该函数一直等待，直到FIFO不为空。

非阻塞式编程

- 数据发送
  - 将数据发送到TX FIFO，如果FIFO没有空，返回FALSE，否则返回TRUE

tran_status = UARTCharPutNonBlocking(UART0_BASE，0x41)；

- 数据接收
  - int32格式，如果FIFO有数据，则返回数据，如果无则返回-1

receive_data = UARTCharGetNonBlocking (UART0_BASE)；

  - 通常采用如下形式配合使用

if (UartCharsAvail(UART0_BASE)

receive_data = UARTCharGetNonBlocking (UART0_BASE)；

函数UARTCharPutNonBlocking

⌘ 驱动库手册：P565

功能	发送一个字符到指定端口。
原型	Bool **UARTCharPutNonBlocking**(uint32_t **ui32Base**, unsigned char **ucData**)
参数	**ui32Base** 为 UART端口的基地址。 **ucData** 为待发送的字符。
说明	该函数将字符**ucData**写到指定端口的发送FIFO。该函数不阻塞，因此如果没有空间，则返回flase，并且应用必须再试一遍。
返回值	如果字符被成功地放到发送FIFO，返回true；如果发送FIFO没有空间返回false。

函数UARTCharGetNonBlocking

⌘ 驱动库手册：P564

功能	从指定端口接收一个字符。
原型	int32_t **UARTCharGetNonBlocking**(uint32_t **ui32Base**)
参数	**ui32Base** 为UART端口的基地址。
说明	该函数从指定端口的FIFO接收一个字符。
返回值	从指定端口返回所读的字符，返回值类型为int32_t。如果当前接收FIFO中没有字符则返回的值为-1。在试图调用该函数前应先调用函数UARTCharsAvail() 函数。

函数UARTCharsAvail

⌘ 驱动库手册：P566

功能	确定接收FIFO是否有字符。
原型	Bool **UARTCharsAvail**(uint32_t **ui32Base**)
参数	**ui32Base**为UART端口的基地址。
说明	该函数返回一个标志，反映接收FIFO中有没有数据。
返回值	如果接收FIFO中有数据返回**true**；如果接收FIFO中没有数据返回**false**。

UART中断控制

- 开中断

```
IntEnable(INT_UART0);
UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
    //Enable UART0 RX,RT interrupt
IntMasterEnable();
```

函数IntEnable

⌘ 驱动库手册：P352

功能	使能中断。
原型	Void **IntEnable**(uint32_t **ui32Interrupt**)
参数	**ui32Interrupt** 使能一个指定中断。
说明	指定中断在中断控制器中被使能。参数 **ui32Interrupt**必须是 **Peripheral Driver Library User's Guide** 中列出的有效的INT_ 值并且在inc/hw_ints.h头文件中定义过的值。

函数UARTIntEnable

⌘ 驱动库手册：P575

功能	使能单个的UART中断源。
原型	Void **UARTIntEnable**(uint32_t **ui32Base**, uint32_t **ui32IntFlags**)
参数	**ui32Base**为UART端口的基地址。 **ui32IntFlags**为要使能的中断源的位。
说明	该函数使能指定的中断源。只有使能的中断源才被反射到到处理器中断；未使能的中断源对处理器没影响。 参数**ui32IntFlags**为任何下列值的逻辑或： UART_INT_RT： 接收超时中断（接收FIFO已有数据但未满，而后续数据长时间不来） UART_INT_TX：发送中断 UART_INT_RX ：接收中断.....

UART中断控制

- UART的中断清除
  - 当UART发送或接收到一个数据后，会置位相应的中断标志位，如果允许中断，则会进入中断（发送中断或接收中断），使用中断能够减少主程序中消耗的时间，提高效率。
  - 因为UART的中断关联了发送或接收，因此不能象单一中断如SYSTICK等，由CPU在进入中断后自动清除中断标志位，只能由编程人员根据情况清除中断标志位。如果中断标志位不清除，会导致反复的进出中断。

```
int32_t uart0_int_status;
uart0_int_status = UARTIntStatus(UART0_BASE, true);
        // Get the interrrupt status.
UARTIntClear(UART0_BASE, uart0_int_status);
        //Clear the asserted interrupts
```

函数UARTIntStatus

⌘ 驱动库手册：P576

功能	获取当前中断状态。
原型	uint32_t **UARTIntStatus**(uint32_t **ui32Base**, bool **bMasked**)
参数	**ui32Base**为UART端口的基地址。 **bMasked**：**false**表示需要获取原始的中断状态，**true**表示需要获取屏蔽的中断状态。
说明	该函数返回指定UART的中断状态。或者返回原始的中断状态，或者返回允许反射到处理器的中断的状态。
返回值	返回当前中断状态。

函数UARTIntClear

⌘ 驱动库手册：P574

功能	清除UART的中断状态。
原型	Void **UARTIntClear**(uint32_t **ui32Base**, uint32_t **ui32IntFlags**)
参数	**ui32Base**为UART端口的基地址。 **ui32IntFlags**为要清除的中断源的位。
说明	指定的UART中断源被清除，以便不再生效。该函数在中断函数中必须被调用，以避免从中断退出后马上该中断又被触发。

第三次实验（上）

实验目的

- 了解UART 串行通讯的工作原理
- 掌握在PC 端通过串口调试工具与实验板通过UART 通讯的方法
- 掌握UART 的堵塞式与非堵塞式通讯方法

实验内容

- 实验3-1

  例程为UART0 的初始化，实验板在初始化完成后向主机发送“HELLO,WORLD！”字符串。

- 要求：
  - 阅读3-1.c 并理解。
  - 需要使用上位机软件，从ftp的”软件”文件夹下载。
  - SerialPro无需安装，直接打开.exe文件。
  - 需要按照3-1例程中UART的初始化设置正确地配置上位机参数。

Serial Port参数设置

- 串口号应与“设备管理器”中的虚拟串口号一致
- 参数与例程中设置一致：
  设置波特率为115,200 bps，
  校验位为NONE，数据位为8，
  停止位为1
- 然后“打开串口”


```
// Configure the UART for 115,200, 8-N-1 operation.
UARTConfigSetExpClk(UART0_BASE, ui32SysClock, 115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE
|UART_CONFIG_PAR_NONE));
```

实验内容

- 实验3-2
  - 在实验1 的基础上，通过PC 端发送字符串，实验板收到后并原样返回。称为UART ECHO。
  - 阅读3-2.c 并理解。
- 实验3-3
  - 将实验2 改写为非堵塞式方式，即中断方式进行发送与接收。当进行数据接收时，点亮PN1。
  - 阅读3-3.c 并理解。

第三次实验讨论题

1. 实验3-2，if (UARTCharsAvail(UART0_BASE))此行程序的作用。如果没有此行，会导致什么问题？

2. 实验3-3，void UART0_Handler(void)为什么没有在主函数声明？

3. 为什么3-3 的中断中需要读取中断标志并清除，而SYSTICK不需要？