1. uartlite\_test应用程序
2. 初始化
   1. GPIO流控

由于板卡GPIO转RS485模块有流控，因此采用GPIO作为流控。

A．实例化XGpio

B．使用XGpio\_Initialize(&Gpio, XPAR\_GPIO\_0\_DEVICE\_ID); 函数作为流控GPIO初始化。

C．使用XGpio\_SetDataDirection(&Gpio, 1, 0xFE);函数初始化GPIO方向为输出。

* 1. UARTLITE串口

1. 实例化XUartlite串口，实例化定时器XScuGic；定时器不断运作，在检测到串口中断后，转到串口中断处理函数（见2）
2. 使用XUartLite\_Initialize(&Uartlite, XPAR\_UARTLITE\_0\_DEVICE\_ID);初始化串口。
3. 使用UartLiteSetupIntrSystem(&IntcInstPtr, &Uartlite, UARTLITE\_IRPT\_INTR);函数设置串口中断。
4. 使能中断XUartLite\_EnableInterrupt(&Uartlite);
5. uartlite串口中断
6. 对于UartLiteSetupIntrSystem(&IntcInstPtr, &Uartlite, UARTLITE\_IRPT\_INTR);函数，需要载入定时器，串口实例，串口ID。

函数内部先初始化XScuGic定时器配置：

XScuGic\_LookupConfig(INTC\_DEVICE\_ID);

XScuGic\_CfgInitialize(IntcInstancePtr, IntcConfig, IntcConfig->CpuBaseAddress);

触发优先级:

XScuGic\_SetPriorityTriggerType(IntcInstancePtr, UartLiteIntrId, 0xA0, 0x3);

连接串口ID，定时器，串口实体，中断处理函数（重要）

XScuGic\_Connect(IntcInstancePtr定时器, UartLiteIntrId串口ID,

(Xil\_ExceptionHandler)Handler中断处理函数,

UartLiteInstPtr串口实体);

使能定时器：

XScuGic\_Enable(IntcInstancePtr, UartLiteIntrId);

*此处未用到异常处理*

异常初始化：

Xil\_ExceptionInit();

载入异常处理函数：

Xil\_ExceptionRegisterHandler(XIL\_EXCEPTION\_ID\_INT,

(Xil\_ExceptionHandler)XScuGic\_InterruptHandler,

IntcInstancePtr);

异常处理使能：

Xil\_ExceptionEnable();

1. 串口中断处理函数

第一步：

通过函数XUartLite\_ReadReg(InstancePtr串口实体->RegBaseAddress, XUL\_STATUS\_REG\_OFFSET);来读取串口实体的状态寄存器值。

第二步：

实验发现，Uartlite接收缓存长16字节，若传输数据超过16字节长，则会触发XUL\_SR\_RX\_FIFO\_FULL中断（同时XUL\_SR\_RX\_FIFO\_VALID\_DATA中断也被触发)。因此，在状态寄存器值== XUL\_SR\_RX\_FIFO\_FULL时，使用XUartLite\_Recv(&Uartlite, ReceivedBufferPtr, MAX\_LEN);函数接收数据，并保持接收状态。

第三步：

最后一次接收的缓存将不触发XUL\_SR\_RX\_FIFO\_FULL中断，因此最后一次使用XUartLite\_Recv(&Uartlite, ReceivedBufferPtr, MAX\_LEN);函数接收数据，并将flag置1，代表不再接收数据。

1. 串口收发状态机

状态机分4个状态，在while(1)循环内运行。分别是：Uartlite串口接收，流控变向等待，Uartlite串口发送，流控再次变向等待。

1. Uartlite串口接收：若上述提到的flag在中断处理函数中被置1，则开始处理数据：将接收的数据指针转到发送数据指针，并置接收计数，flag为0。

否则sleep(1)；

1. 流控变向：变向到发送模式，需要将Gpio的最后一位写1: XGpio\_DiscreteWrite(&Gpio, 1, 0x01);。实验发现，至少需要2ms才能完成软件控制的流控变向。
2. Uartlite串口发送：每间隔2ms发送16字节数据，直到所有接收到的数据被发送完。XUartLite\_Send(&Uartlite, SendBufferPtr+BUFFER\_LEN\*i, BUFFER\_LEN);函数可实现发送数据，但一次只能发送16字节，并且会触发发送中断。
3. 流控变向: 变向到接收模式，需要将Gpio的最后一位写1: XGpio\_DiscreteWrite(&Gpio, 1, 0x00);。
4. dma\_test应用程序
5. 初始化
6. 初始化Gpio，同上；
7. 初始化Uartlite，同上，并初始化中断系统以及中断服务函数，使能中断；
8. 初始化Uart0\_PS，并初始化中断系统以及中断服务函数；

UART0PSSetup(XUartPs \*UartInstancePtr, u16 UartIntrId)；函数初始化，包含对UART0初始化，设置模式，设置数据格式，设置触发和触发mask。

1. 初始化AXIDMA，XAxiDma\_Setup(XPAR\_AXIDMA\_0\_DEVICE\_ID);函数初始化

包含初始化，去SG模式，MM2S去除中断，使能S2MM中断。

1. uartlite串口中断
2. 对于UartLiteSetupIntrSystem(&IntcInstPtr, &Uartlite, UARTLITE\_IRPT\_INTR);函数，需要载入定时器，串口实例，串口ID。

函数内部先初始化XScuGic定时器配置：

XScuGic\_LookupConfig(INTC\_DEVICE\_ID);

XScuGic\_CfgInitialize(IntcInstancePtr, IntcConfig, IntcConfig->CpuBaseAddress);

触发优先级:

XScuGic\_SetPriorityTriggerType(IntcInstancePtr, UartLiteIntrId, 0xA0, 0x3);

连接串口ID，定时器，串口实体，中断处理函数（重要）

XScuGic\_Connect(IntcInstancePtr定时器, UartLiteIntrId串口ID,

(Xil\_ExceptionHandler)Handler中断处理函数,

UartLiteInstPtr串口实体);

使能定时器：

XScuGic\_Enable(IntcInstancePtr, UartLiteIntrId);

*此处未用到异常处理*

异常初始化：

Xil\_ExceptionInit();

载入异常处理函数：

Xil\_ExceptionRegisterHandler(XIL\_EXCEPTION\_ID\_INT,

(Xil\_ExceptionHandler)XScuGic\_InterruptHandler,

IntcInstancePtr);

异常处理使能：

Xil\_ExceptionEnable();

1. 串口中断处理函数

第一步：

通过函数XUartLite\_ReadReg(InstancePtr串口实体->RegBaseAddress, XUL\_STATUS\_REG\_OFFSET);来读取串口实体的状态寄存器值。

第二步：

实验发现，Uartlite接收缓存长16字节，若传输数据超过16字节长，则会触发XUL\_SR\_RX\_FIFO\_FULL中断（同时XUL\_SR\_RX\_FIFO\_VALID\_DATA中断也被触发)。因此，在状态寄存器值== XUL\_SR\_RX\_FIFO\_FULL时，使用XUartLite\_Recv(&Uartlite, ReceivedBufferPtr, MAX\_LEN);函数接收数据，并保持接收状态。

第三步：

最后一次接收的缓存将不触发XUL\_SR\_RX\_FIFO\_FULL中断，因此最后一次使用XUartLite\_Recv(&Uartlite, ReceivedBufferPtr, MAX\_LEN);函数接收数据，并将flag置1，代表不再接收数据。

1. 数据环状态机

状态机分6个状态，在while(1)循环内运行。分别是：Uartlite串口接收，流控变向为发送等待，Uart0\_PS串口发送PL，AXIDMA接收，Uartlite串口发送，流控变向为接收等待。

1. Uartlite串口接收

每秒检查中断服务函数，若XUL\_SR\_RX\_FIFO\_VALID\_DATA中断触发且XUL\_SR\_RX\_FIFO\_FULL不触发，flag置1。

Flag为1时，存储接收到的Uartlite串口数据，并把暂时存储和flag清零。

1. 流控变向发送

切换Gpio数据为0x01。等待2ms保证切换成功。

1. Uart0\_PS串口发送PL

将Uartlite接收缓存数据通过UartPsSend(&Uart0\_PS, UART0SendBufferPtr, UART0SendByteNum);函数发到Uart0\_PS中，传给PL端。

1. AXIDMA接收PL

PL端接收Uart0\_PS数据（55包头，之后数据长度，之后数据），解析后对发送的数据加1，并转换成AXIDMA发送的数据。

AXIDMA按新包发送数据。

XAxiDma\_SimpleTransfer(&AxiDma,(UINTPTR) Dma\_RecvBufPtr, MAX\_LEN, XAXIDMA\_DEVICE\_TO\_DMA);函数控制接收；

Xil\_DCacheFlushRange((UINTPTR)Dma\_RecvBufPtr, MAX\_LEN);将数据载入DDR；

while (XAxiDma\_Busy(&AxiDma,XAXIDMA\_DEVICE\_TO\_DMA)){

/\* Wait \*/

}

等待数据接收完毕。

1. Uartlite串口发送

由于Uartlite buffer大小测试是16字节，因此按照16字节长度分开发送整段AXIDMA接收的数据，每次发送间间隔2ms，直到发送完成。

1. 流控变向接收

为下一次接收做准备，将Gpio置0x00。