不是最准确的！不是最准确的！不是最准确的！

直接说操作：跟着文档走就行

J1-125 ，串口为 UART1 的 48...49 引脚，使能串口外设的步骤如图 12-11 所示。

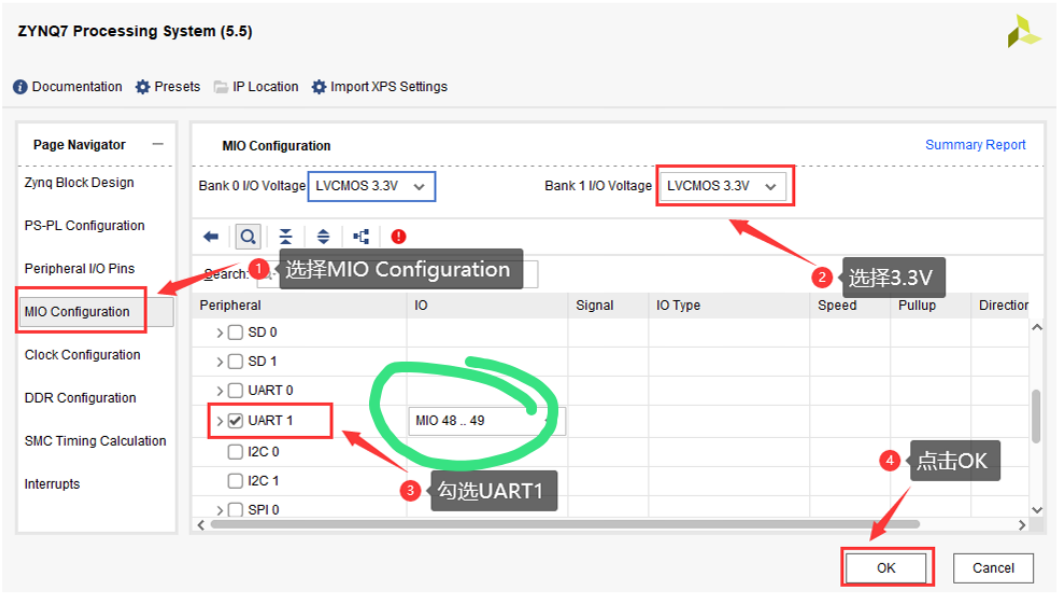
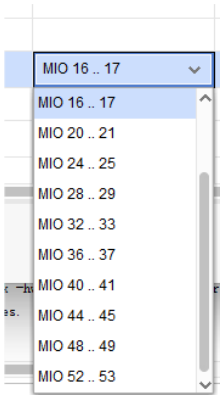


图 12-11 添加串口接口

但是选择的引脚要是自己想用的引脚

例如我要用 MIO16 和 MIO17 这里就选 1617

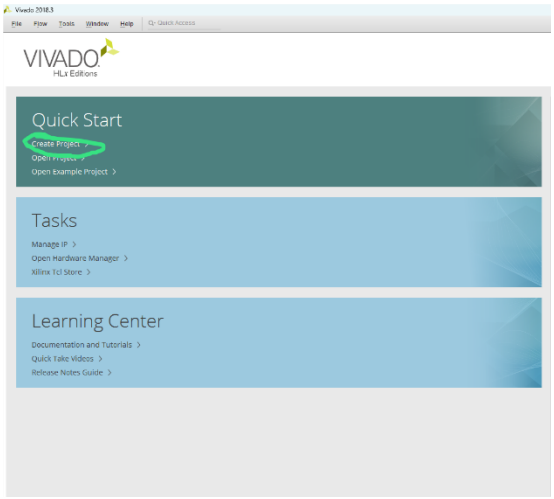


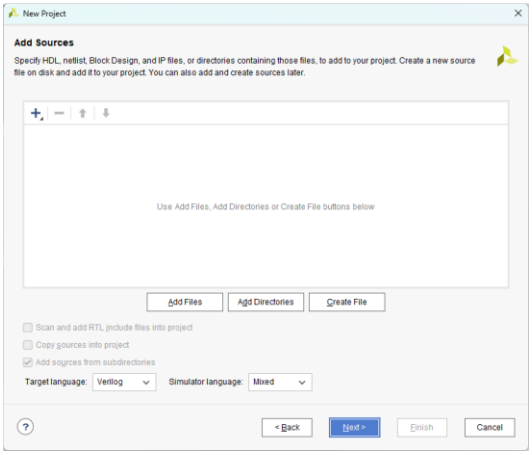
后面的所有操作都不变

但是要注意如果开始选了 1617 要改串口位置需要重新开一个文件在当前文件里面改是没用的

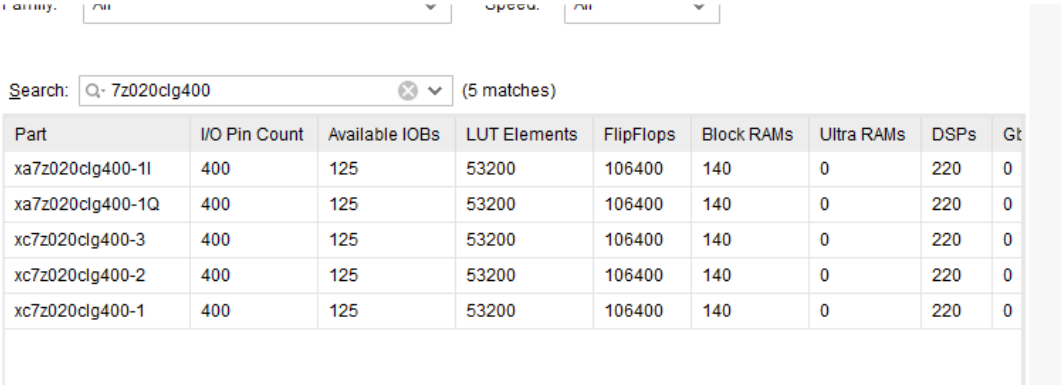
到此结束　后面为超详细的教程

首先打开 vivado　　选择新建工程　然后输入工程名字　一路 next



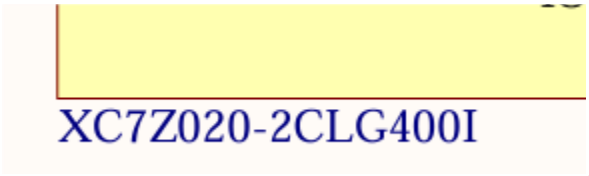


这种有要求输入的位置可以直接 next 跳过



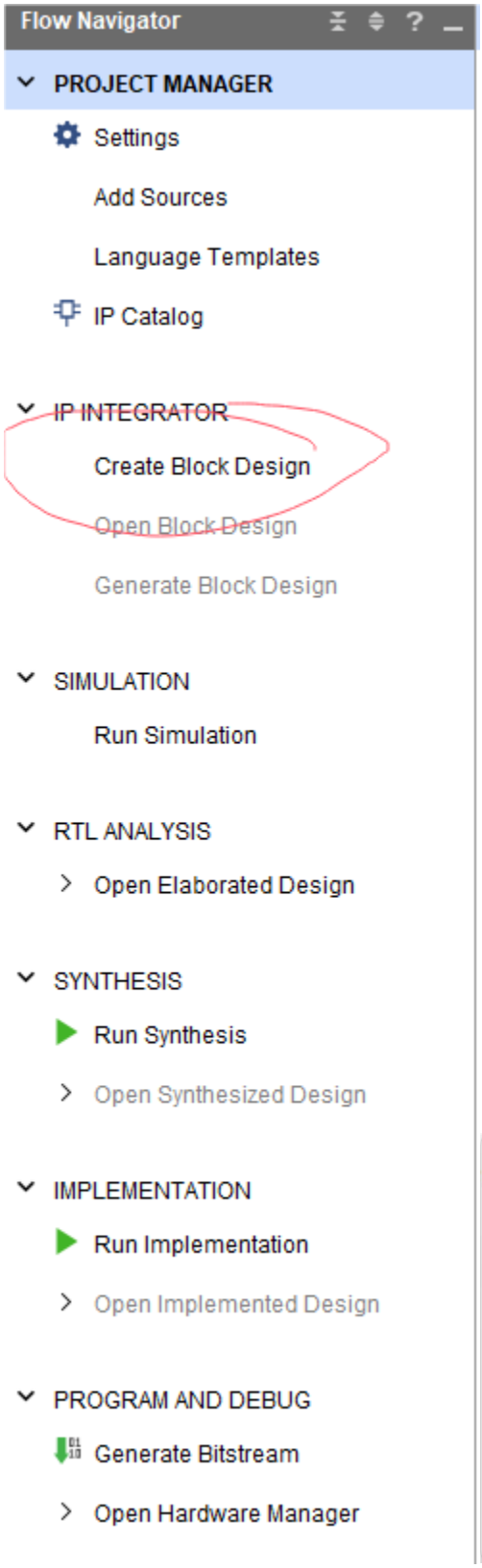
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xa7z020clg400-1I	400	125	53200	106400	140	0	220	0
xa7z020clg400-1Q	400	125	53200	106400	140	0	220	0
xc7z020clg400-3	400	125	53200	106400	140	0	220	0
xc7z020clg400-2	400	125	53200	106400	140	0	220	0
xc7z020clg400-1	400	125	53200	106400	140	0	220	0

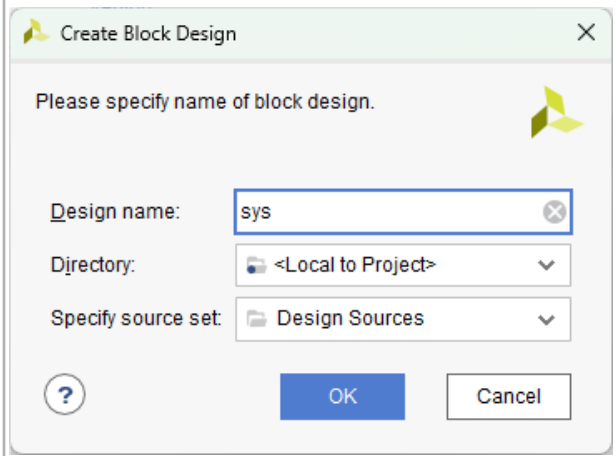
这里我用的 BX71 内核是 7Z020　建议选 clg400-1　（我看芯片手册上是 1）



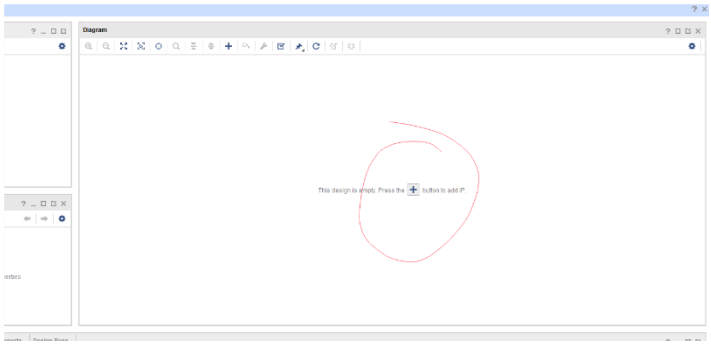
点完 finish 新建就完成了

然后是 vivado 的完成

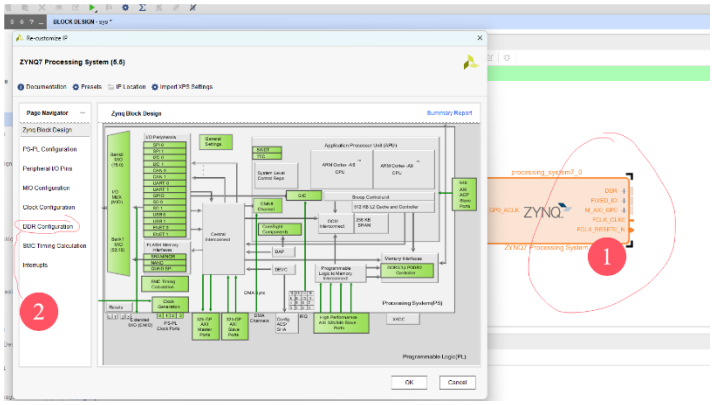




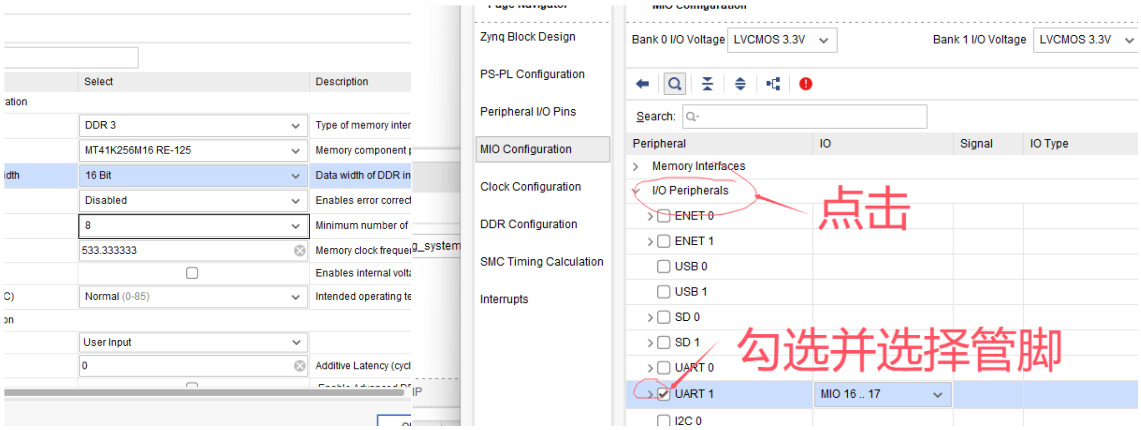
名字随意取 我这里 sys 点 ok



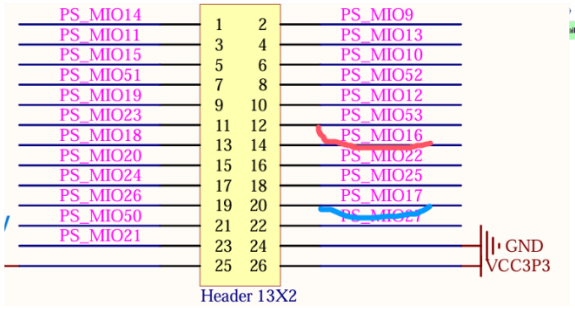
这个步骤因该是添加 ip 核 点击加号查找 zynq 直接添加就行

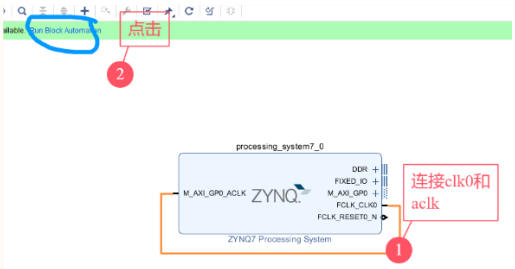


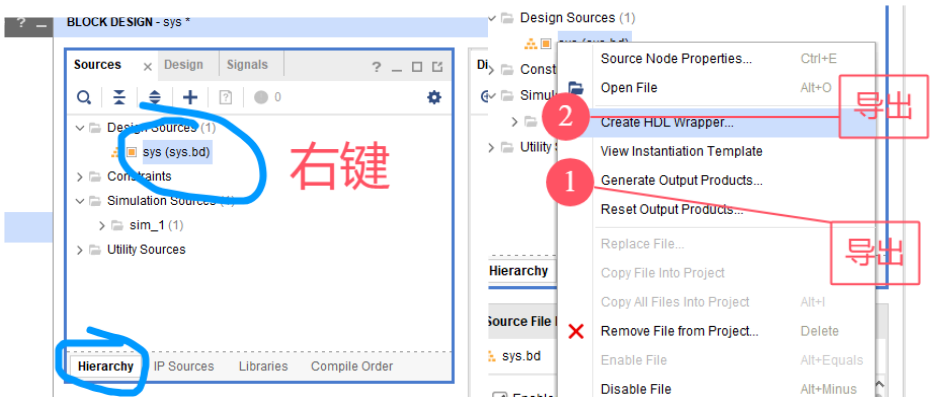
双击核心 打开 ddr 的配置 我这里是 BX71 所以 ddr 选尾号 re125 宽度选 16bit



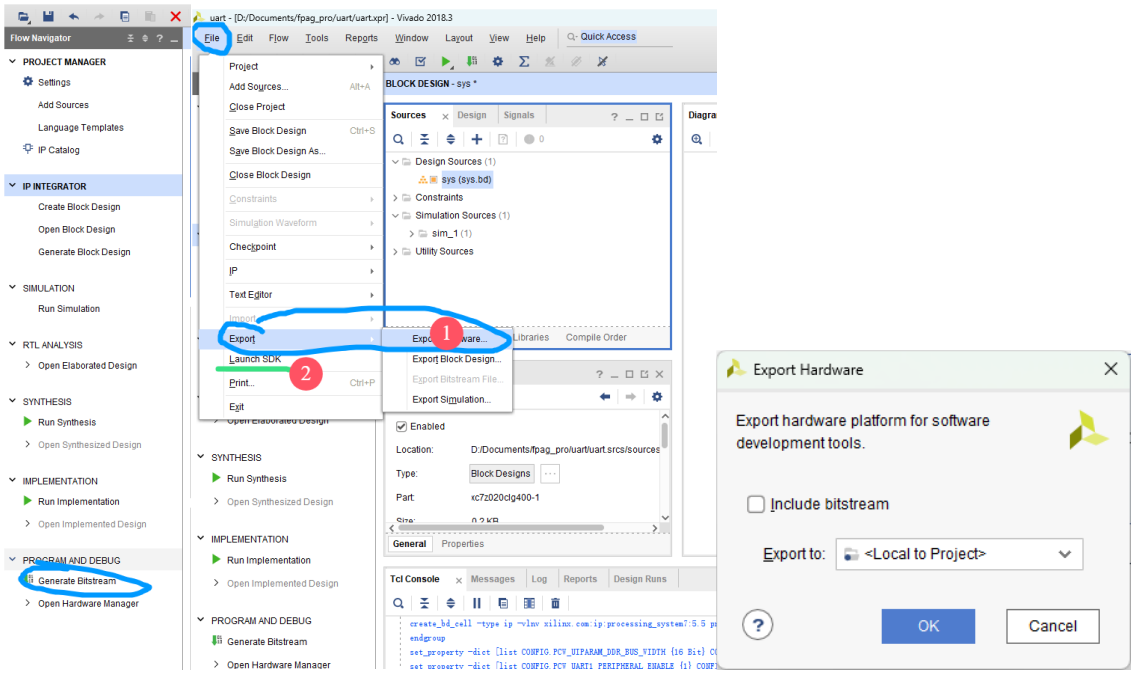
再次双击打开 zynq 核心，选择 mio 配置 uart1 管脚选择自己要用的例如我用 1617 就选 1617





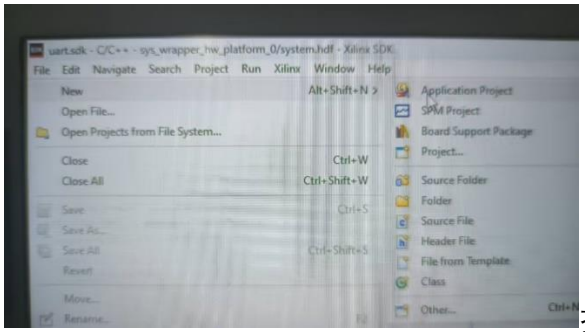


右键 sys 导出数据　编译一遍　本次实验没有用到 pl 端　所以不需要引脚约束

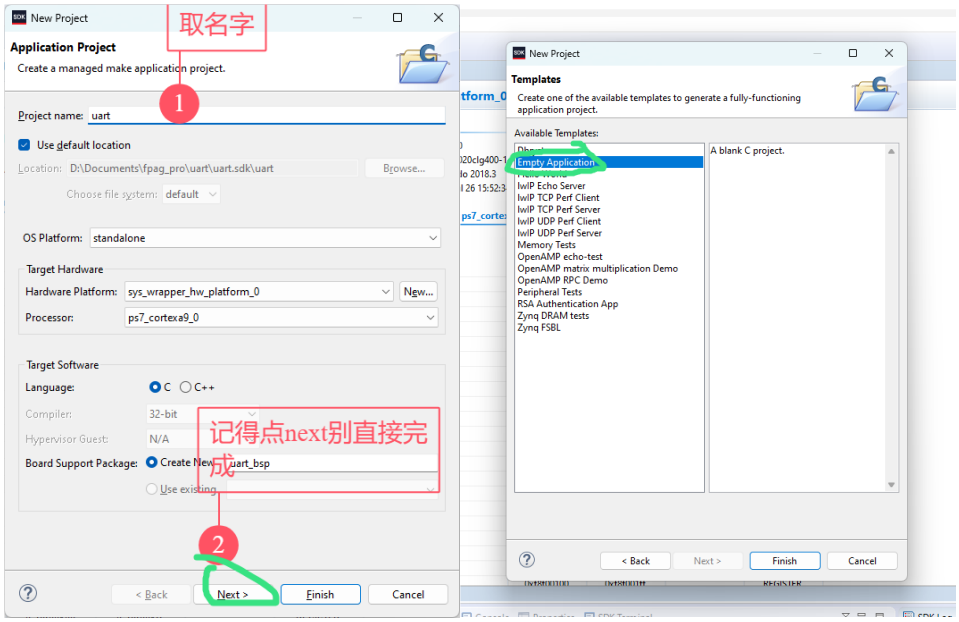


先导出 hardware　然后打开 sdk

这里 include bitstream 可以不勾选 因为我们没用 pl

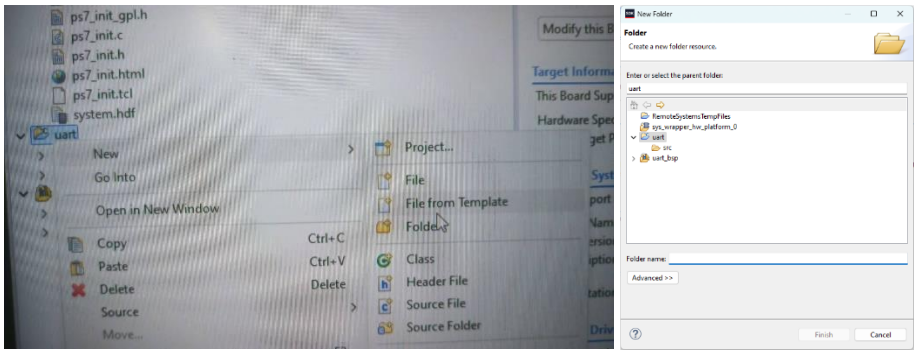


打开 sdk 后点文件 new->app……



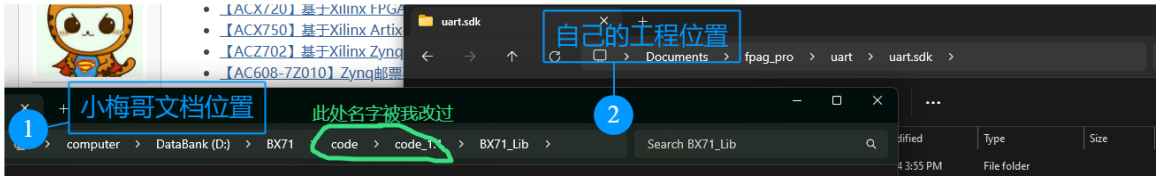
选到 hello world 和空程序都行

然后是文件创建 右键刚刚建的工程名字是自己取得那个 New->folder

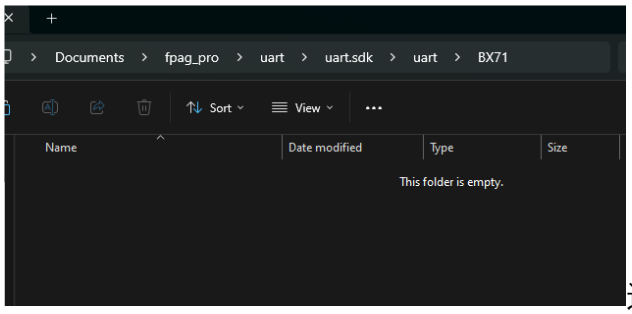


名字按照小梅哥的来   我这里省写了 lib

然后就可以去文件夹里面托文件了

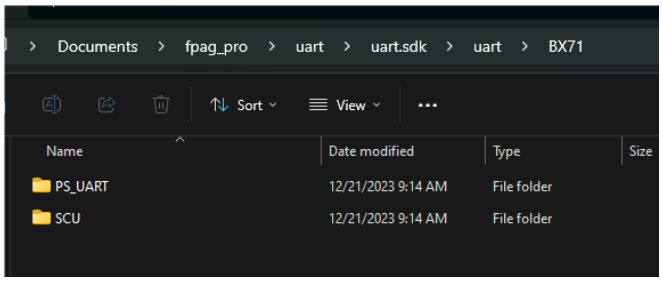


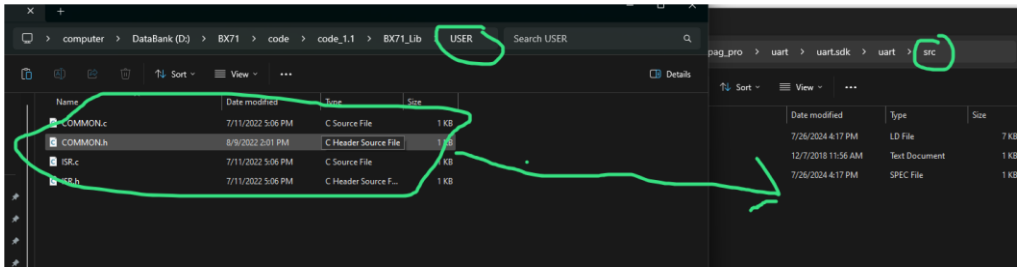
原版名字



这是自己的位置 刚刚创建过 bx71 的

直接打开 把官方文档里面的 PS_UART 和 SCU 复制进自己的工程



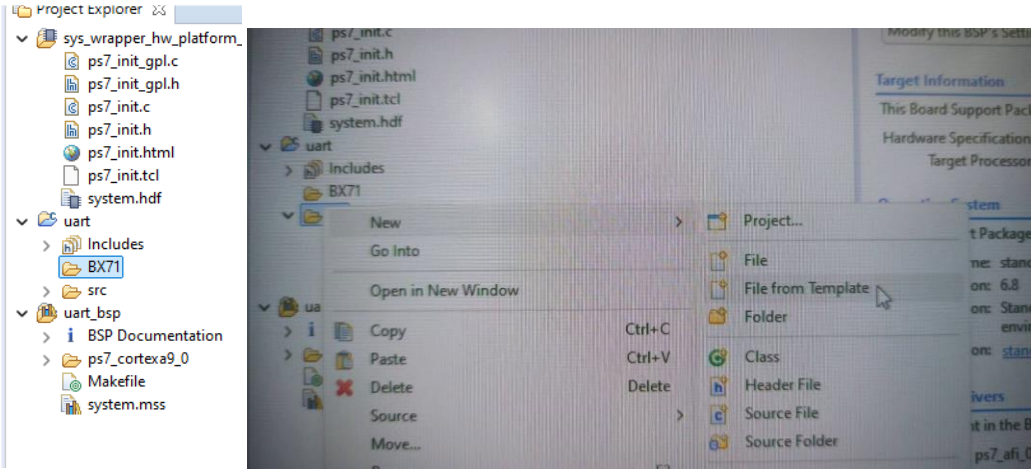


将官方文档 USER 下的文件全部复制到我们工程的 src 的文件夹内

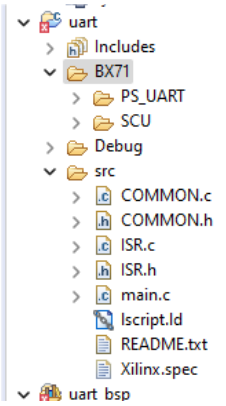
这时我们打开 sdk 发现 bx71 下没显示 怎么办呢

我们添加一个 main.c 文件在 src 下

右键 src new->source file 名字 main.c



这时等一下 系统就能加载出我们 bx71 下的文件了



但是我们发现有报错 先不急 我们一步一步处理

先把这个粘进我们的 main.C 文件 这个要下载小梅哥的文档

03_【裸机教程】基于 C 编程的 Zynq 裸机程序设计与应用教程 v1.0.1.pdf

12.3.2.2编写 main.c

接下来在 src 中新建一个源文件，并命名为 main.c，将本次设计的用户代码粘贴进 main.c 中，代码如下：

```
#include "COMMON.h"

uint8_t Receive_Buffer[10];

int main(void)
{
    uint8_t Data[10];
    uint8_t i;
```

全部粘进 main.c 后我们把这个粘进 common.h

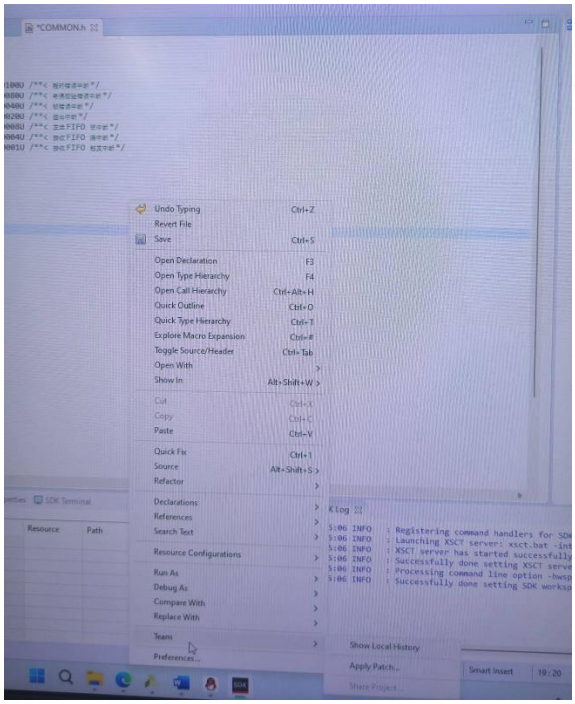
接收超时为 32 字符，启用 7 种中断事件，并链接中断处理函数为 PS_UART1_IRQ_Handler（）。这里，7 种中断事件如下：

```
#define XUARTPS_IXR_TOUT    0x00000100U /**< 超时错误中断 */
#define XUARTPS_IXR_PARITY  0x00000080U /**< 奇偶校验错误中断 */
#define XUARTPS_IXR_FRAMING 0x00000040U /**< 帧错误中断 */
#define XUARTPS_IXR_OVER    0x00000020U /**< 溢出中断 */
#define XUARTPS_IXR_TXEMPTY 0x00000008U /**< 发送 FIFO 空中断 */
#define XUARTPS_IXR_RXFULL  0x00000004U /**< 接收 FIFO 满中断 */
#define XUARTPS_IXR_RXOVR   0x00000001U /**< 接收 FIFO 触发中断 */
```

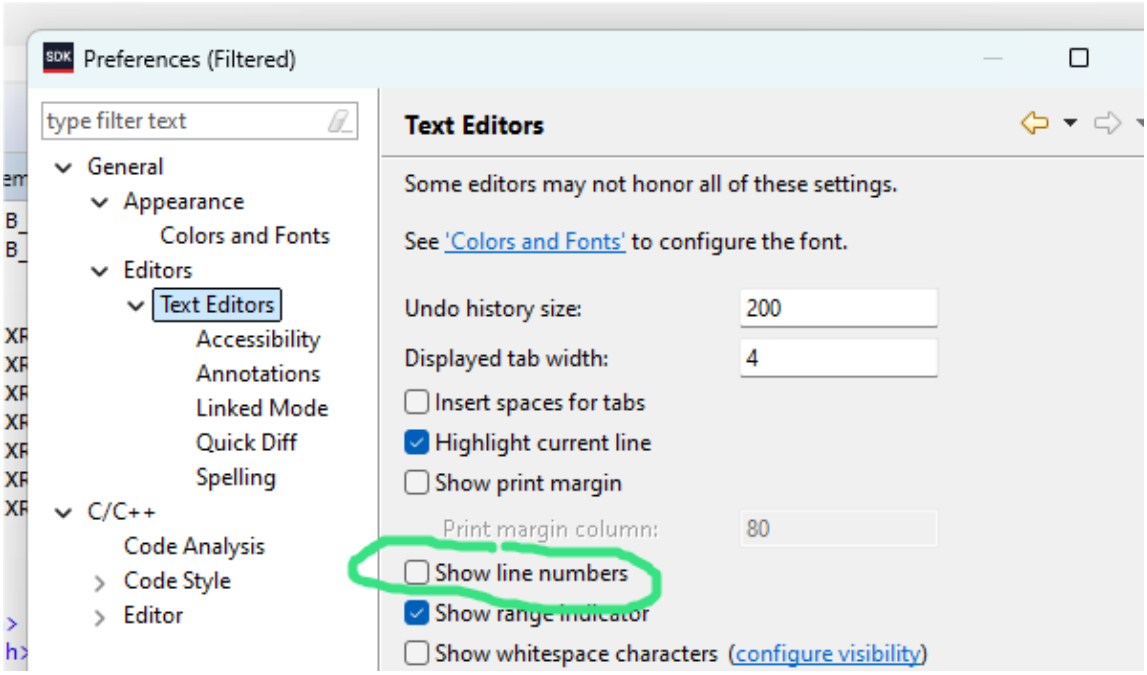
```
#define XUARTPS_IXR_TOUT
0x00000100U /**< 超时错误中断*/
#define XUARTPS_IXR_PARITY  0x00000080U /**< 奇偶校验错误中断*/
#define XUARTPS_IXR_FRAMING 0x00000040U /**< 帧错误中断*/
#define XUARTPS_IXR_OVER
0x00000020U /**< 溢出中断*/
#define XUARTPS_IXR_TXEMPTY 0x00000008U /**< 发送FIFO 空中断*/
#define XUARTPS_IXR_RXFULL  0x00000004U /**< 接收FIFO 满中断*/
#define XUARTPS_IXR_RXOVR   0x00000001U /**< 接收FIFO 触发中断*/
```

这里不在同一排是因为字符串太长了我们缩进一下就行

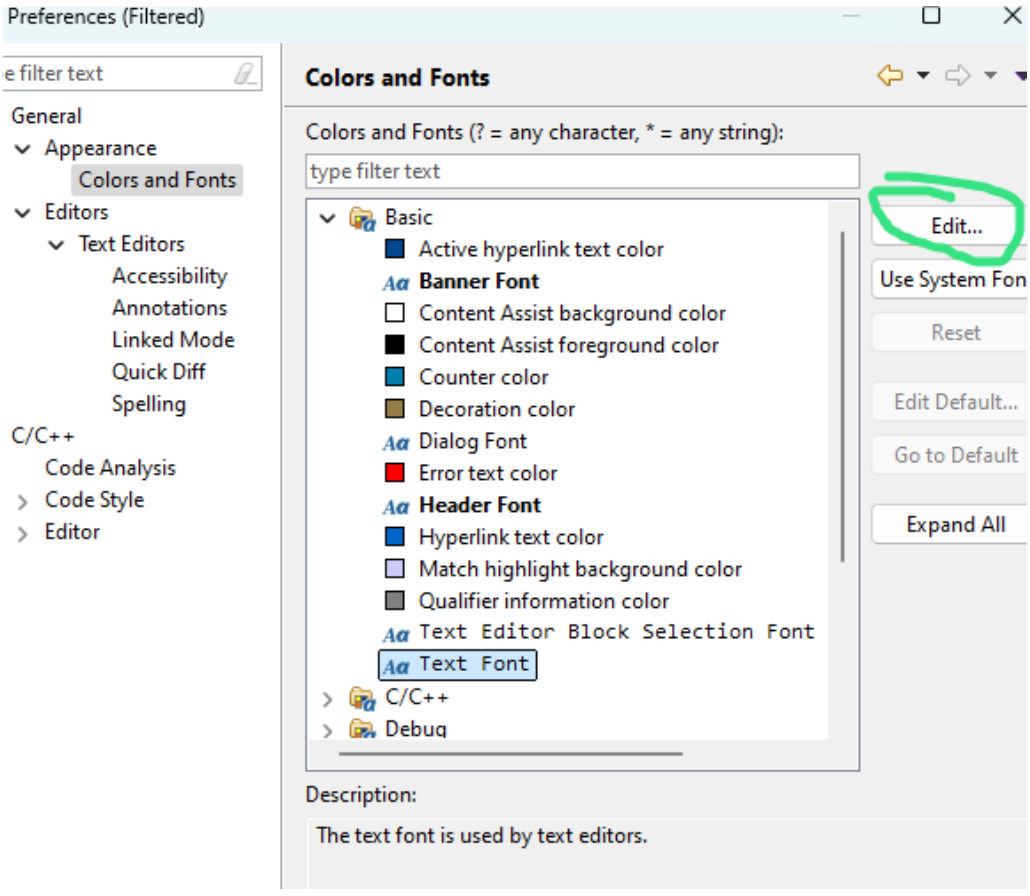
这时字比较小 我们改一下 顺便显示一下行数

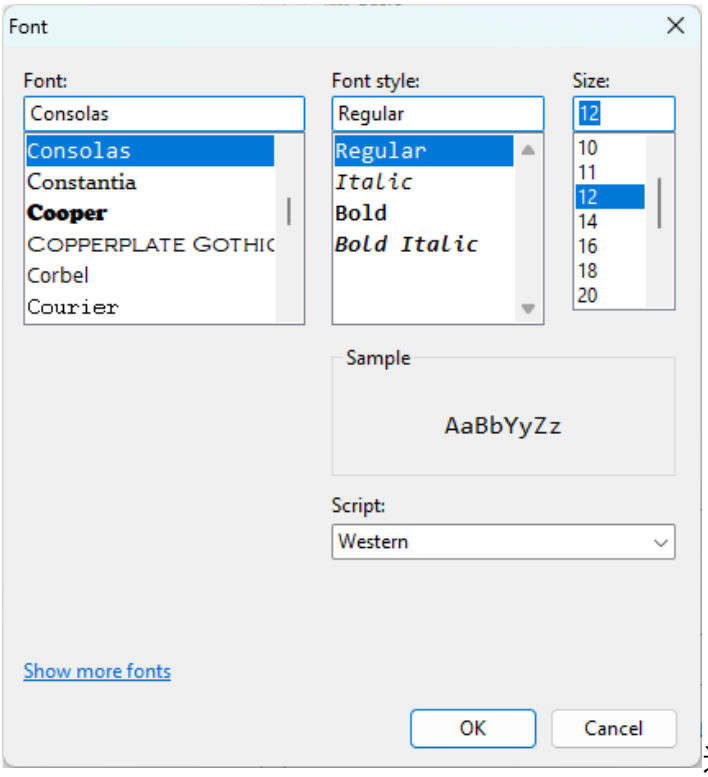


在任意代码界面右键空白处　点击最下方的 prefereces



行数　　字体大小





这里我改的 12

然后我们执行下一步

前面讲到，初始化串口中断时链接串口中断处理函数为 PS_UART1_IRQ_Handler（），但是此时我们还未添加该函数。双击 ISR.c 文件，

在其中添加本次串口中断处理语句，如下：

```
//中断里使用的标志位
volatile int Send_All_Flag = 0; //全部发送标志
volatile int Recv_All_Flag = 0; //全部接收标志
volatile int TimeOut_Flag = 0;  //超时标志
```

在 isr.c 文件中添加这段代码

isr.h 中添加这段

这里共支持 7 种中断事件类型，设计中只使用到其中三种，所有中断事件类型定义如下：

```
#define XUARTPS_EVENT_RECV_DATA          1U /**< 数据接收完成 */
#define XUARTPS_EVENT_RECV_TOUT          2U /**< 接收超时 */
#define XUARTPS_EVENT_SENT_DATA          3U /**< 数据传输完成 */
#define XUARTPS_EVENT_RECV_ERROR         4U /**< 接收错误 */
#define XUARTPS_EVENT_MODEM              5U /**< 模式状态改变 */
#define XUARTPS_EVENT_PARE_FRAME_BRKE    6U /**< 接收奇偶校验、帧、中断错误 */
#define XUARTPS_EVENT_RECV_ORERR         7U /**< 接收溢出 */
```

还是 isr.h

通过中断事件而产生的标志位会作为全局变量，在 main 函数中被使用。因此我们需要使用 extern 对其修饰。打开 ISR.h 文件，使用 extern 修饰定义的标志位，并声明串口中断处理函数，如下：

```
//中断里使用的标志位
extern volatile int Send_All_Flag;  //全部发送标志
extern volatile int Recv_All_Flag;  //全部接收标志
extern volatile int TimeOut_Flag;   //超时标志

void PS_UART1_IRQ_Handler(void *CallBackRef, u32 Event, unsigned int EventData);
```

现在换到 common.h　继续添加

```
EventData);
```

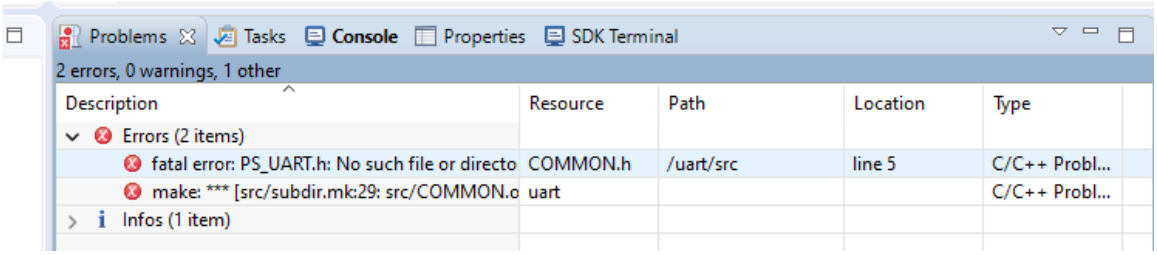
12.3.2.4添加头文件声明

最后我们需要添加本次所使用的 PS_GPIO 库的头文件声明，打开 COMMON.h，在 BX71 头文件下添加以下语句：

```
#include "PS_UART.h"
```

至此，我们便完成了本次 CPU 软件程序设计。接下来，保存设计，软件会自动编译，确认工程没有错误后，便可以开始板级验证了。

这个时候我们 ctrl+s 保存我们修改的文件　　这个 sdk 每保存一次就会编译或者说查错一次　但是我们看见错不要慌　跟着文档一步一步来就好

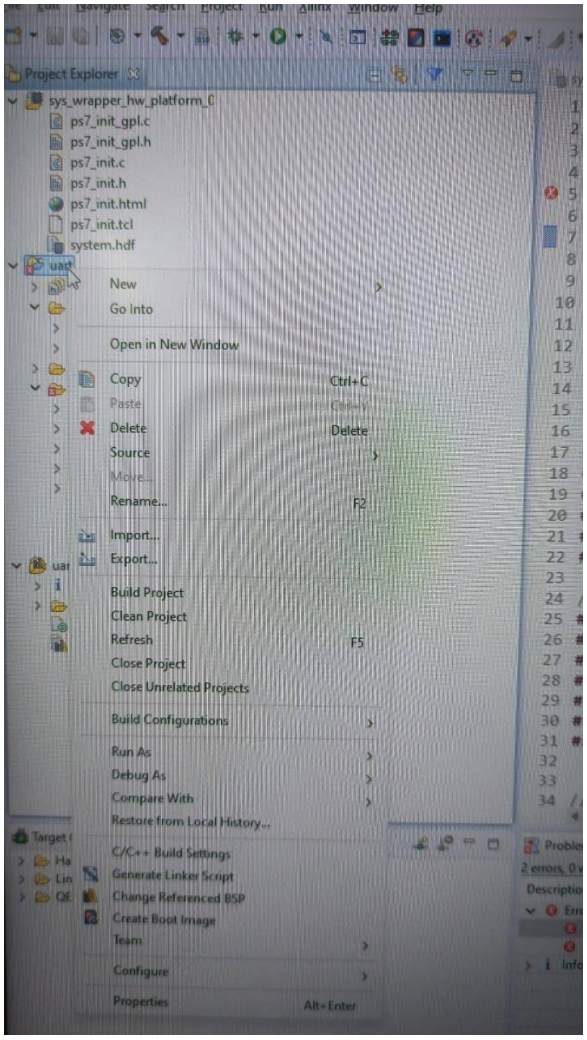


这是我全保存一次后的报错

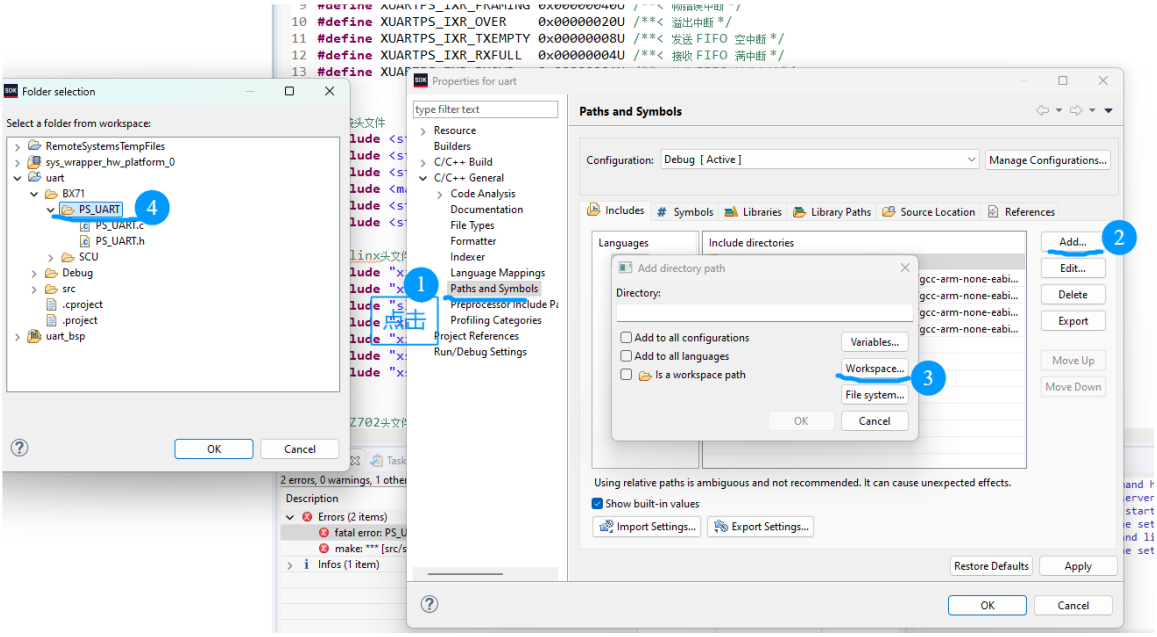
Description Resource Path Location Type

fatal error: PS_UART.h: No such file or directory COMMON.h /uart/src line 5 C/C++ Problem

这个是因为我们没有添加路径

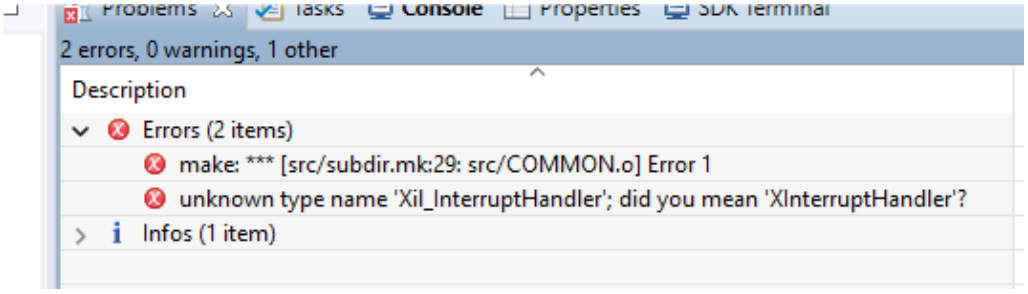


右键我们创建的工程 点击最后的一个 properties



要添加 3 次　PS_UART　SCU　SRC

添加完后我们的报错就变了



Description　Resource　　Path　Location　　Type

unknown type name 'Xil_InterruptHandler'; did you mean 'XInterruptHandler'?
　　PS_UART.h　/uart/BX71/PS_UART line 13 C/C++ Problem

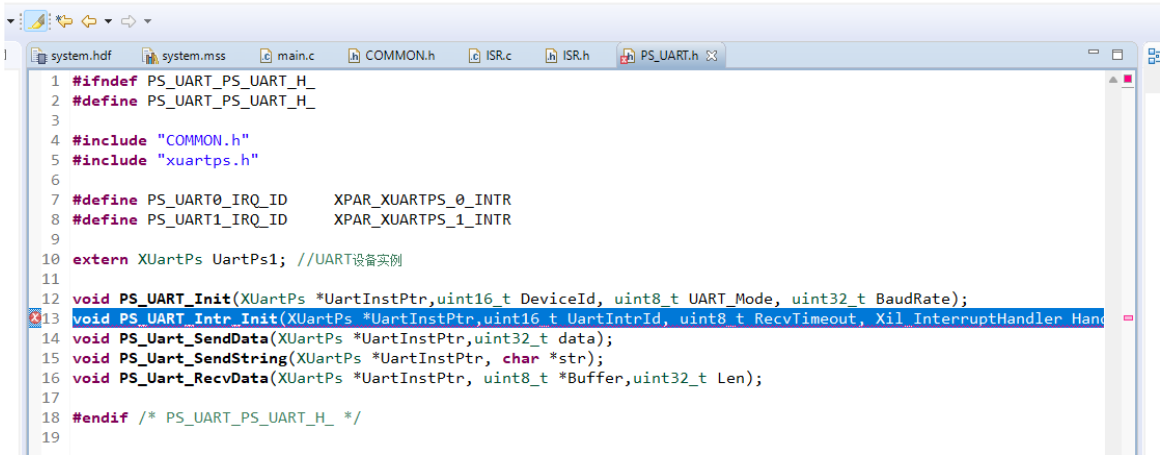
这个表面是说我们没定义　实际上是我们没有引入头文件

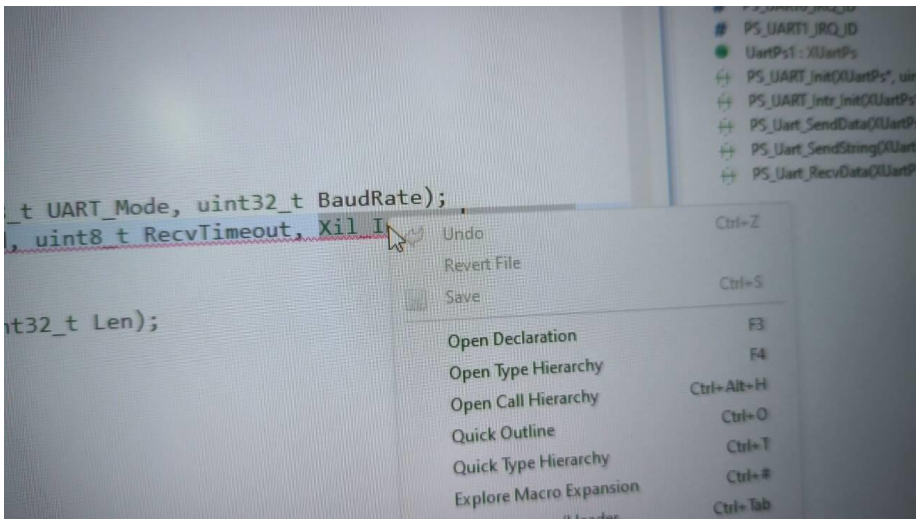
我们双击报错过去看看

可以看到报错的语句

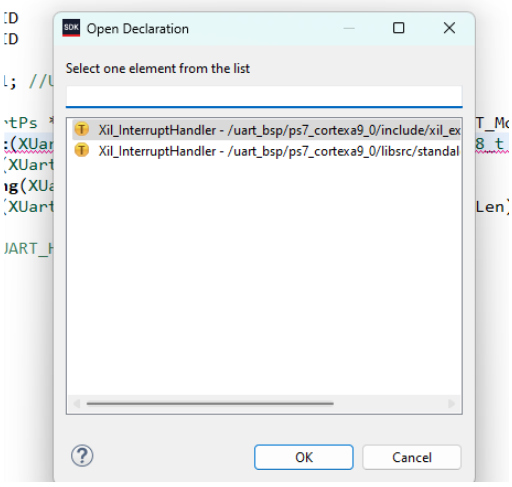
这里的报错说的不是一整句　而是括号内的 xil_interrup……

我们点击这串文字　跳转过去

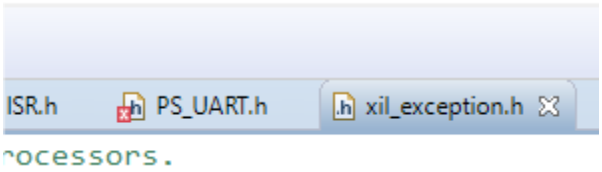




点击第一个 open decl……

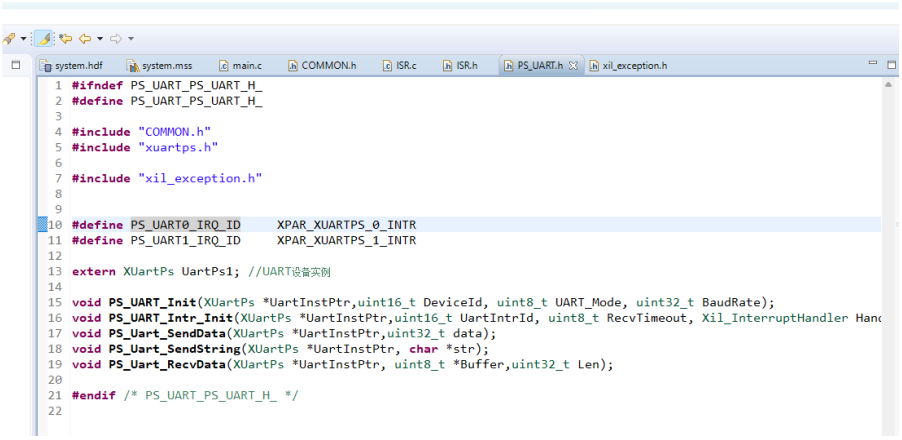


这里点两个好像都是一样的　我一般点第一个



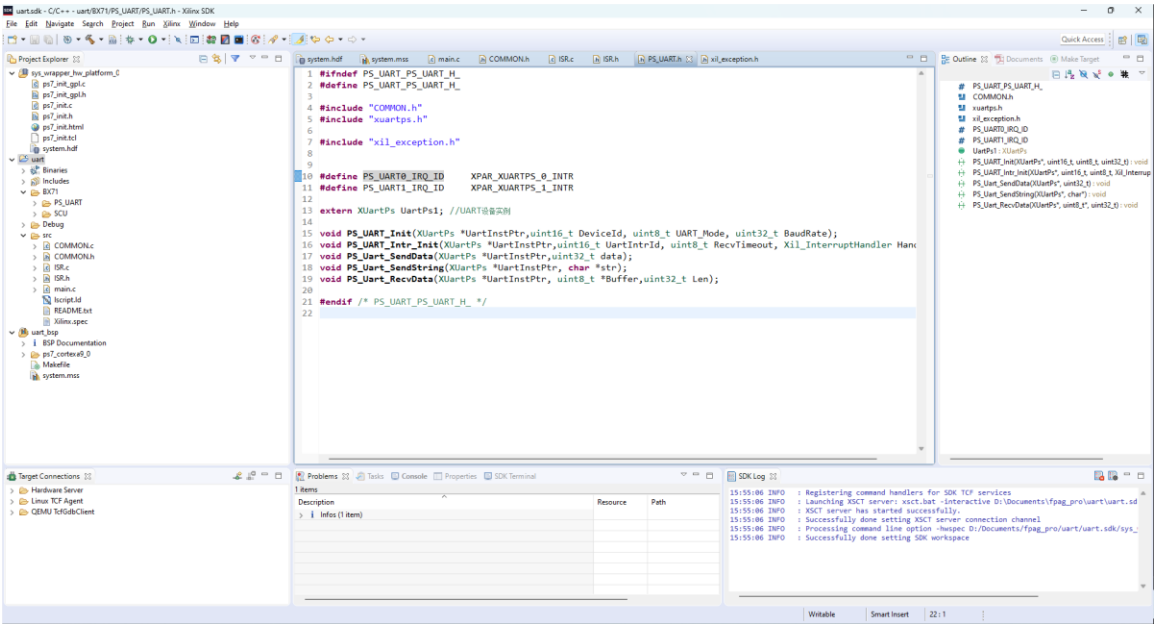
然后我们就能看见这个头文件的名字

我们把它写进 PS_UART.h 就行

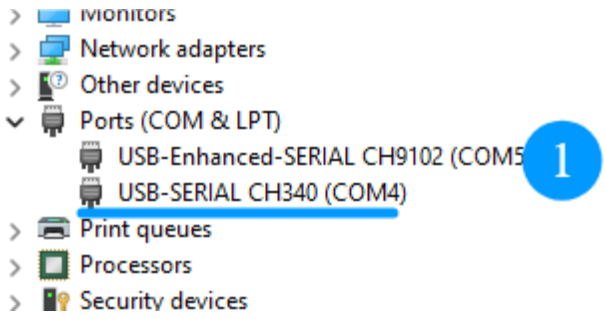


这样又解决掉一个问题

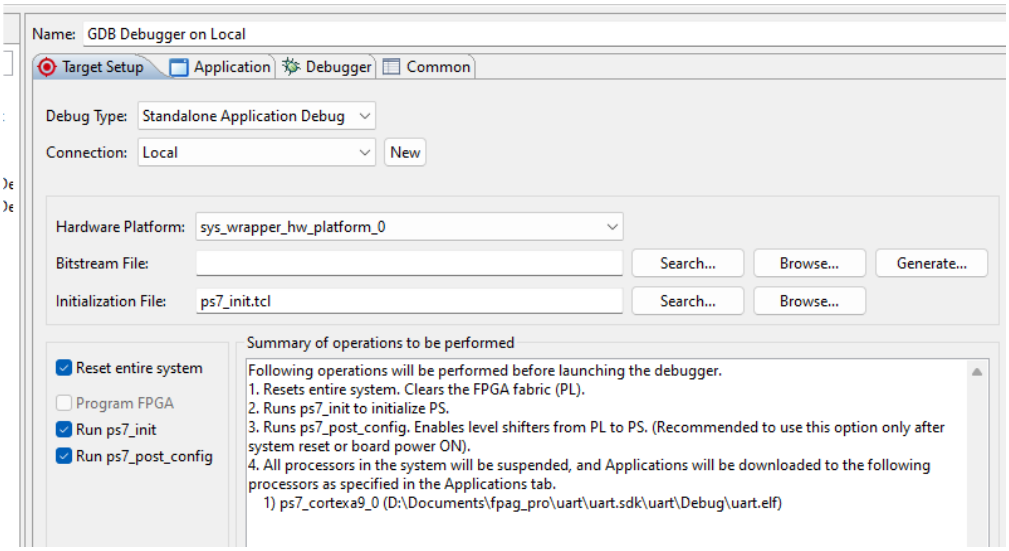
然后就没报错了　我们可以 ctrl+b 验证一下

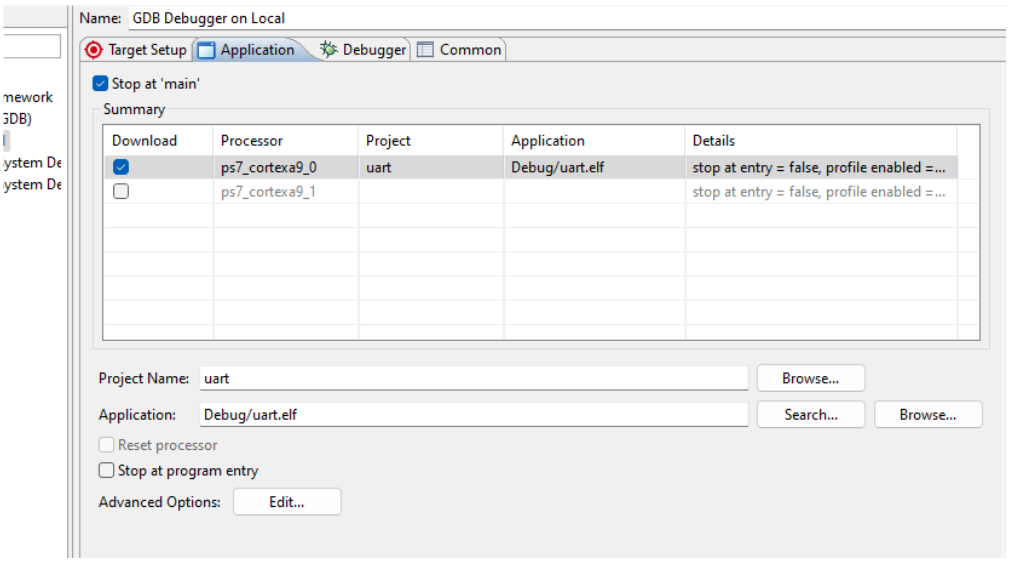


没有问题　可以上电测试了　要注意我们已经用了 1617　所以不能选 fpag 的那个串口了

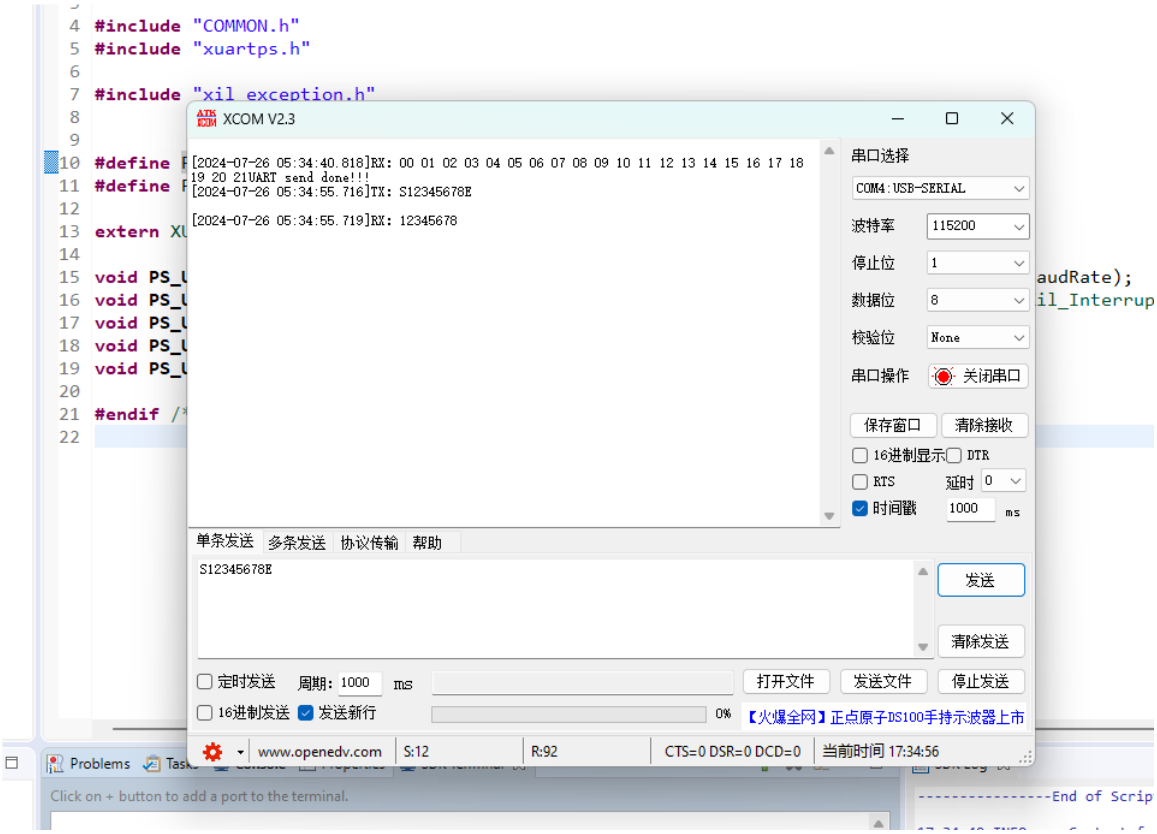


这里需要一个串口调试器来在接到电脑上　才能调试





没有 bit 文件是因为我们没用 pl 这个直接点就行


#include "COMMON.h"
#include "xuartps.h"
#include "xil_exception.h"
#define
#define
extern XU
void PS_U
void PS_U
void PS_U
void PS_U
void PS_U
#endif /*
audRate);
il_Interrup
XCOM V2.3
[2024-07-26 05:34:40.818]RX: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21UART send done!!!
[2024-07-26 05:34:55.716]TX: S12345678E
[2024-07-26 05:34:55.719]RX: 12345678
串口选择
COM4:USB-SERIAL
波特率
115200
停止位
1
数据位
8
校验位
None
串口操作
关闭串口
保存窗口
清除接收
16进制显示
DTR
RTS
延时
0
时间戳
1000
ms
单条发送
多条发送
协议传输
帮助
S12345678E
发送
清除发送
定时发送
周期:
1000
ms
打开文件
发送文件
停止发送
16进制发送
发送新行
0%
【火爆全网】正点原子DS100手持示波器上市
www.openedv.com
S:12
R:92
CTS=0 DSR=0 DCD=0
当前时间 17:34:56
Problems
Click on + button to add a port to the terminal.
----------End of Scrip

到此一切正常