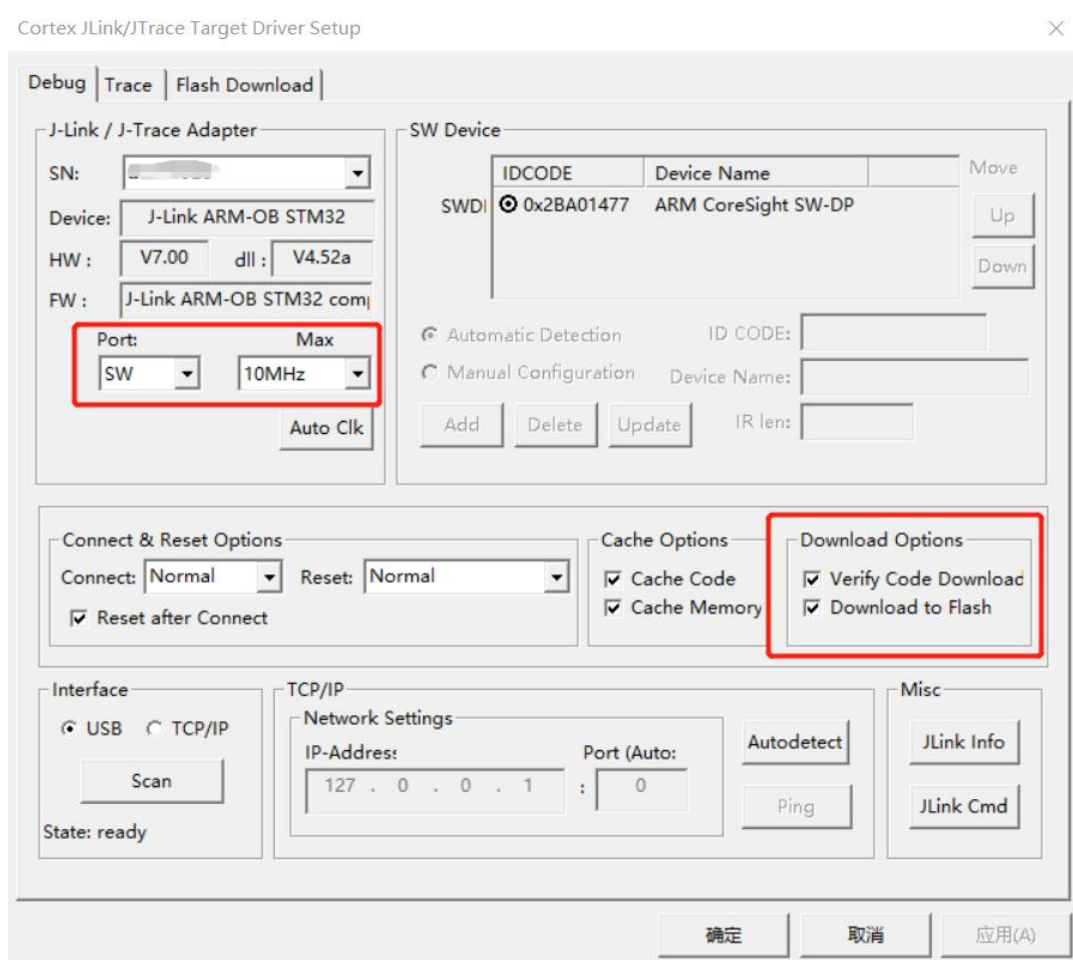
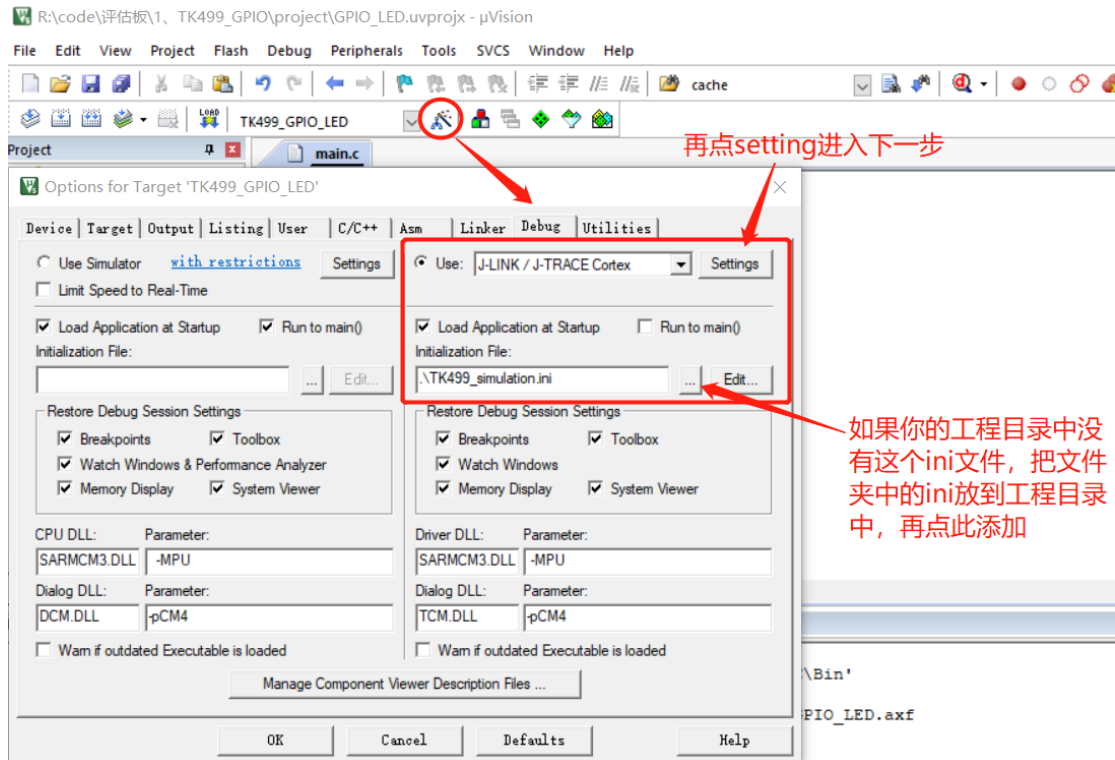
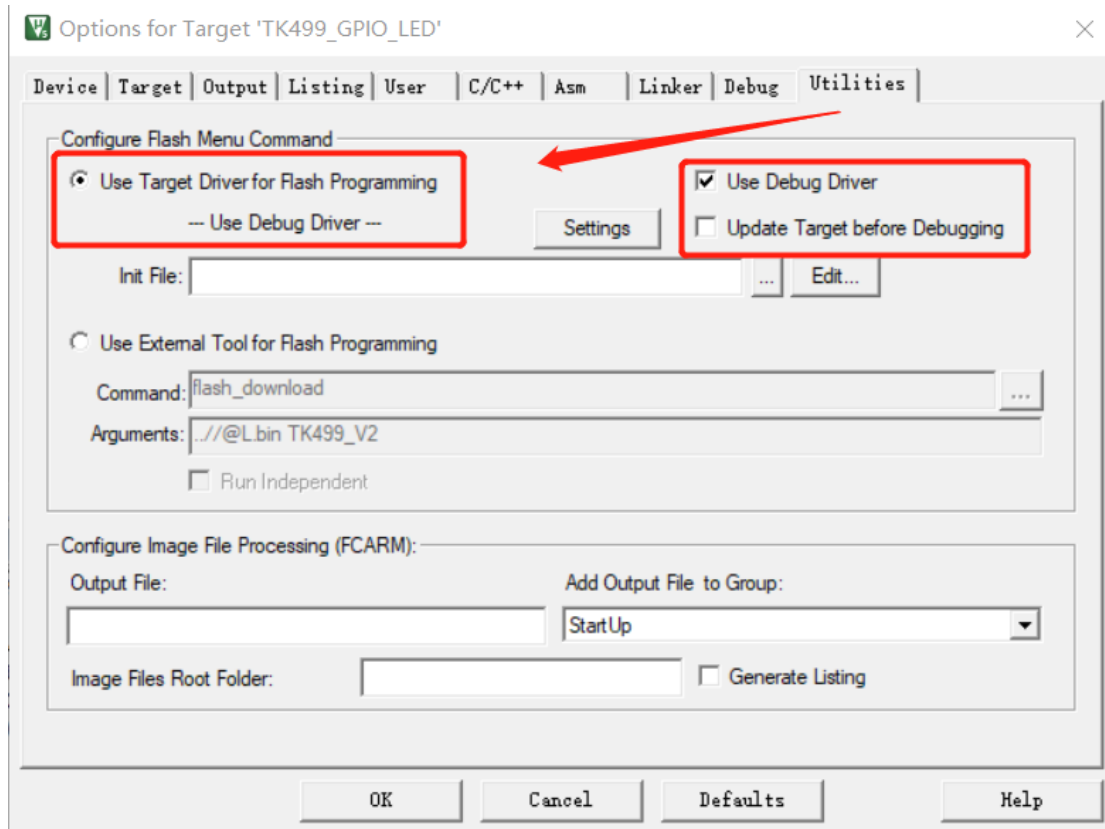
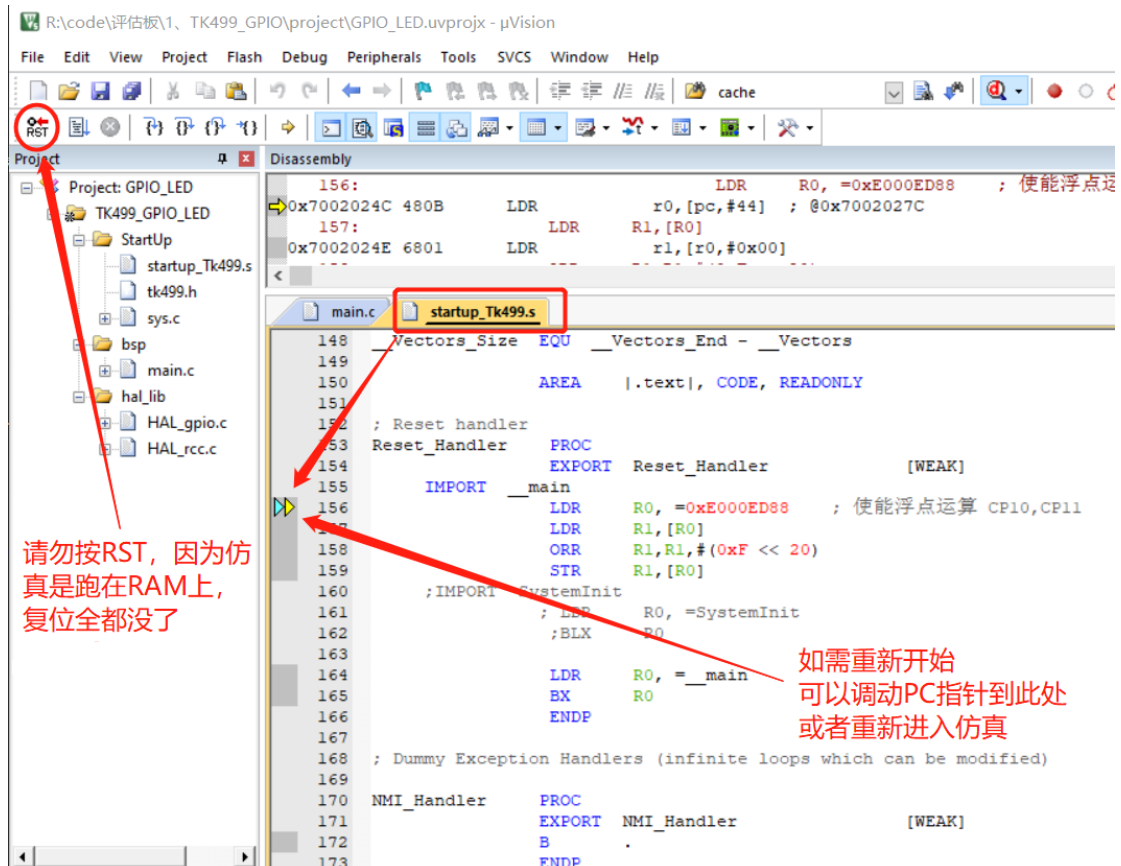


TKM32F499 仿真说明





通过上面三部分，设置就完成了。目前 TK499 仿真运行在 RAM 上，免除对 FLASH 又擦又写，启动速度快。但注意不要点仿真中的 RST，因为一复位，RAM 上的数据都没了。如果你要重新开始，可以调动 PC 指针到 .S 文件中的 _main 处或者重新进入仿真。如图：

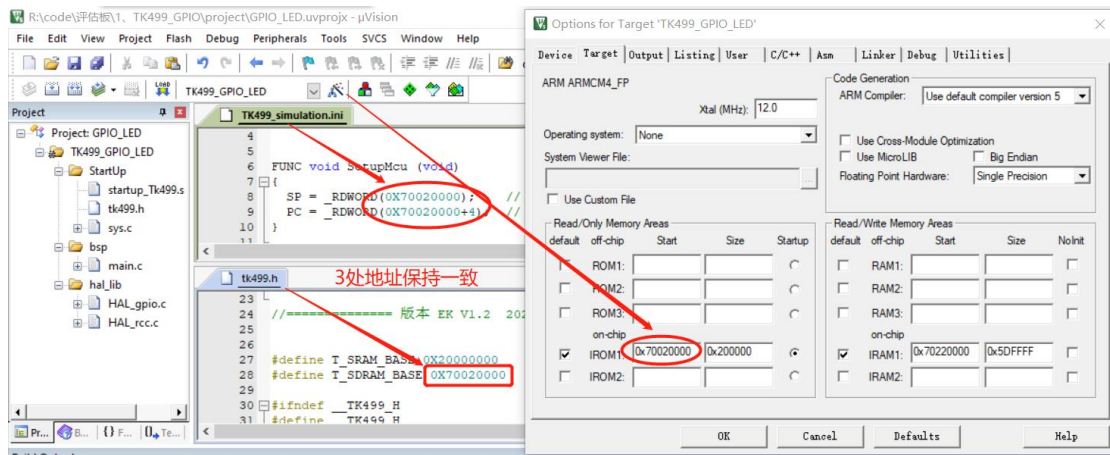


注意事项：

Al_Responder_enable();// 在 main 函数开头如果有这一行，在仿真时要注释掉，抢答器具有乱序发射特性，会令仿真指针跟不上步骤导致跟丢；

官方的 bootloader 通常已经打开 Al_Responder，所以不仿真时不开也可以。但如果你的程序不依赖官方的 bootloader，或者 bootloader 中没打开 Al_Responder，或者干脆不用 bootloader，注意要打开。否则没有了指令抢答功能，程序运行会比较慢。

TK499 的程序存放的起始地址，可以通过 Bootloader 移动，也可以不用 Bootloader。当起始地址发生变化时，记得仿真的起始地址也要相应变化。如下图所示，三处地址保持一致即可。




如果你对提升仿真速度感兴趣，请往下看：

一般仿真都是用于查错，最快的办法还是注释掉暂时不用的代码，这样启动也快，运行也快。特别是图片，仿真时最好不加载，或者存在外部 QSPI FLASH 中调用。如果每启动一次，都要下载 MB 级的图片，累死。

Al_Responder 携带了一个 cache 功能，在芯片运行时，用来平衡 SDRAM 读取比 MCU 的主频慢的问题。Responder 会预读指令，当指令被猜中时，MCU 直接从缓存中快速取指，从而起到加快运行的作用。所以如果遇到 MCU 需要频繁循环取指时，效率会显著提高。当关掉这个时，就会变成效率显著降低了。

所以程序中长延时可以不用 for 循环（类似 for(i=0;i<2000000;i++)），因为往往 MCU 要来回取指几百万次才能延时个零点几秒，同时还要担心会不会被优化掉。如果改用定时器，那么通常就十几个指令搞定。当然短延时无所谓。如果一定要用 for 循环来做长

延时，那也有个小技巧可以用一用。就是在延时后第一行打一个断点，点这个按钮  可以快速运行到断点。原因是：如果不用这个按钮，用单步(step)或者 step over，那么每循环一次，仿真器都要查询一下，效率就很低了。

图片，纯色等海量填充功能，可以选用 DMA 或者 TK80 的自动填充功能，同理也可以加快一些仿真速度。