# KF32 gcc编译器覆盖率功能配置说明

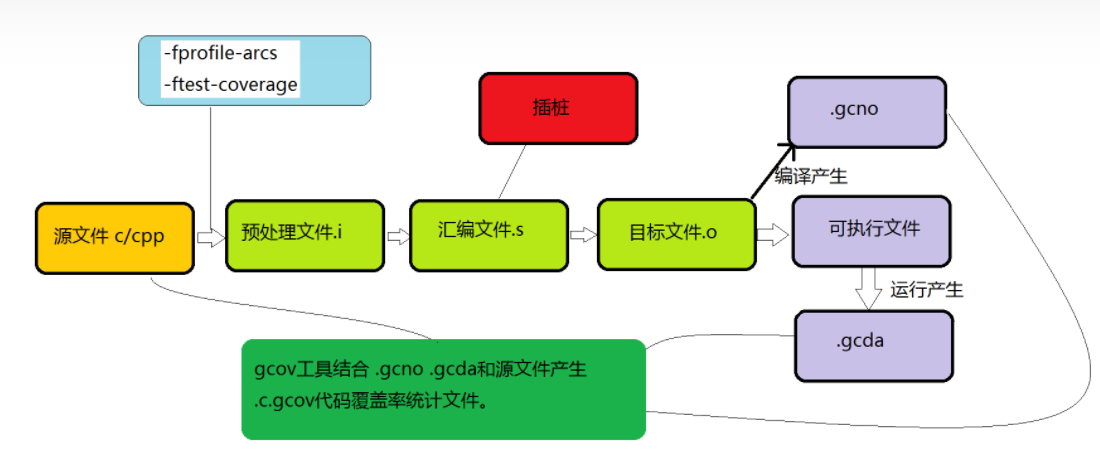
# 编译器选项使能

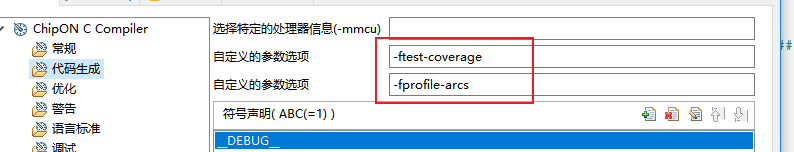
**插桩**是汇编一级的插桩，每个桩点插入N条汇编语句，直接插入.s文件中。在程序执行的过程中，这些桩代码负责收集程序的执行信息。

-fprofile-arcs 选项会让GCC为每个源文件生成同名的.gcda文件【运行输出】

-ftest-coverage 选项会让GCC为每个源文件生成同名的.gcno文件【编译输出】

编译器编译文件时输出文件对应的.gcno文件。程序运行后，通过内部添加的代码【插桩】实现其运行轨迹的信息输出，从而形成.gcda文件。随后gcov程序读取.gcno及.gcda并基于源文件输出文件内容调用的文本文件。





注1：该覆盖率分析将统计函数的使用情况，因此将建立基于文件和函数的信息表，该信息表的结果部分为代码的flash空间，部分为数据的ram空间。同时该模式的函数输出将插入若干的服务指令，即调用次数的累加执行。应用信息表的关联存在从而使可优化不输出的程序被关联，最终分配对应的资源空间。因此，该模式下代码大小与变量大小将明显的增加，你需要预留足够的空间来保存这些内容。若体积不足以完成统计时，可考虑部分文件库文件化或部分独立配置编译器选项，即不开启该选项。

注2：针对插桩的功能实现，需要选择对应的支持工具链版本。

# 插桩方法库设计与实现

设计匹配编译器版本的libgcov.a库，注该方法根据嵌入式控制器的特点进行了代码的适当处理。【基于gnu的工具链，默认情况下，其提供libgcc.a和libgcov.a的库，ChipON的实现将libgcc.a的方法更新进与芯片系列绑定的libcrtvx.a库文件中，如IDE项目环境变量${CHIP\_KERNEL\_LIB}的选择，ChipON历史版本未提供libgcov.a支持库】

1. 可手动将该libgcov.a库存在在标准路径并使用，或存放到项目中使用【若工具标准路径存在，即使项目下存放对应文件，工具链路径下库被优先使用，可差异化名称的使用项目下库】。
2. 在嵌入式单片机中不支持文件概念，需要将写文件功能函数映射为串口的输出，根据项目特点可选择不同的串口输出，文件gcov\_io\_driver.c采用独立的函数名称与libgcov绑定。注:该示例文件的最终串口程序全局生效,如stdio.h中描述，可根据需要做适当调整。
3. 运行结果打印输出，ChipON不引入atexit和exit方法库(设计最小化实现)，根据系统添加条件调用方法\_\_gcov\_exit()即可。即嵌入式应为循环事务的一般不具有终止退出属性。

注：该库未引入malloc|free的动态内存方法。自身占用全局空间和一定的局部变量空间。

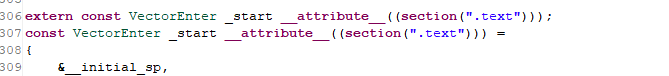
1. ChipON提供示例的串口打印方法，即gcov\_io\_driver.c下的功能函数的数据封装，该输出内容为字符与16进制的混合，提供基于串口16进制接收串口打印信息并提供对应解析程序。

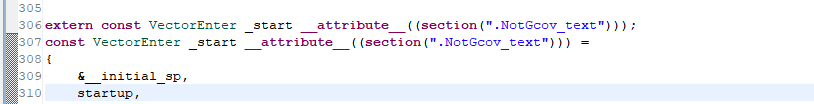
# 项目配置处理

## 启动向量表文件的适配处理

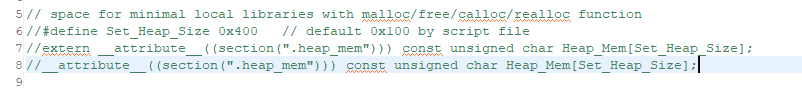
因为插桩程序将在对应的向量表文件中添加若干关联函数，因此默认的脚本将无法确保向量表位于芯片的起始位置的硬件要求。

1. 调整向量表独立标识的段,**如默认.text调整为.NotGcov\_text**

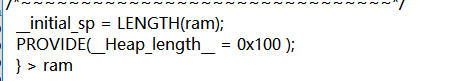




1. **可选**动态申请可分配空间，**向量表文件配置对比如下**

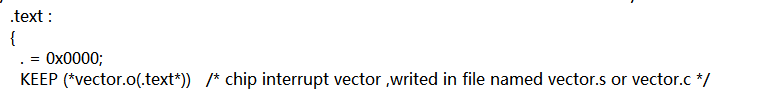


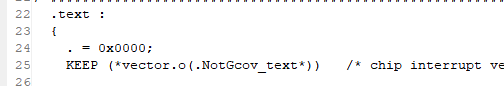
**或在脚本中配置调整，调整对比如下：**



注：该库应不涉及历史文件的读入合并，因此不涉及信息的缓存历史函数信息。未使用malloc方法进行内存申请。

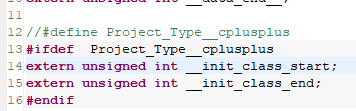
1. **联动脚本段布局实现向量表的优先分配**，**调整对比如下：**

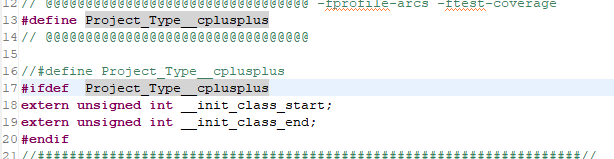




## 启动初始化RAM文件的适配处理

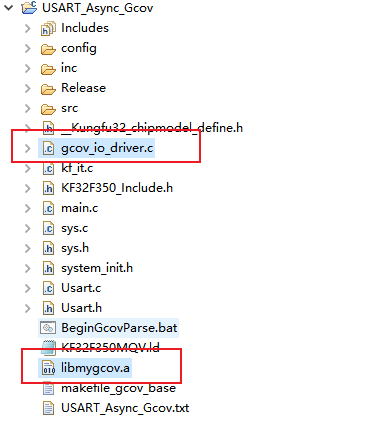
插桩程序建立以文件源文件为单位的信息，和初始化关联。其特性类型C++类的默认构建。因此应初始化过程进行对应指针初始化函数的调用。文件处理变化对比如下：



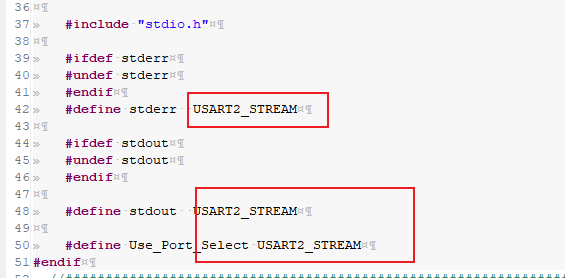


## 引入库与写串口驱动源码到项目

1. 如图将库（若发布工具已具有时不需要项目下存在）与源码文件添加到示例项目如下：



1. 可根据需要在对应的文件下配置底层对应串口



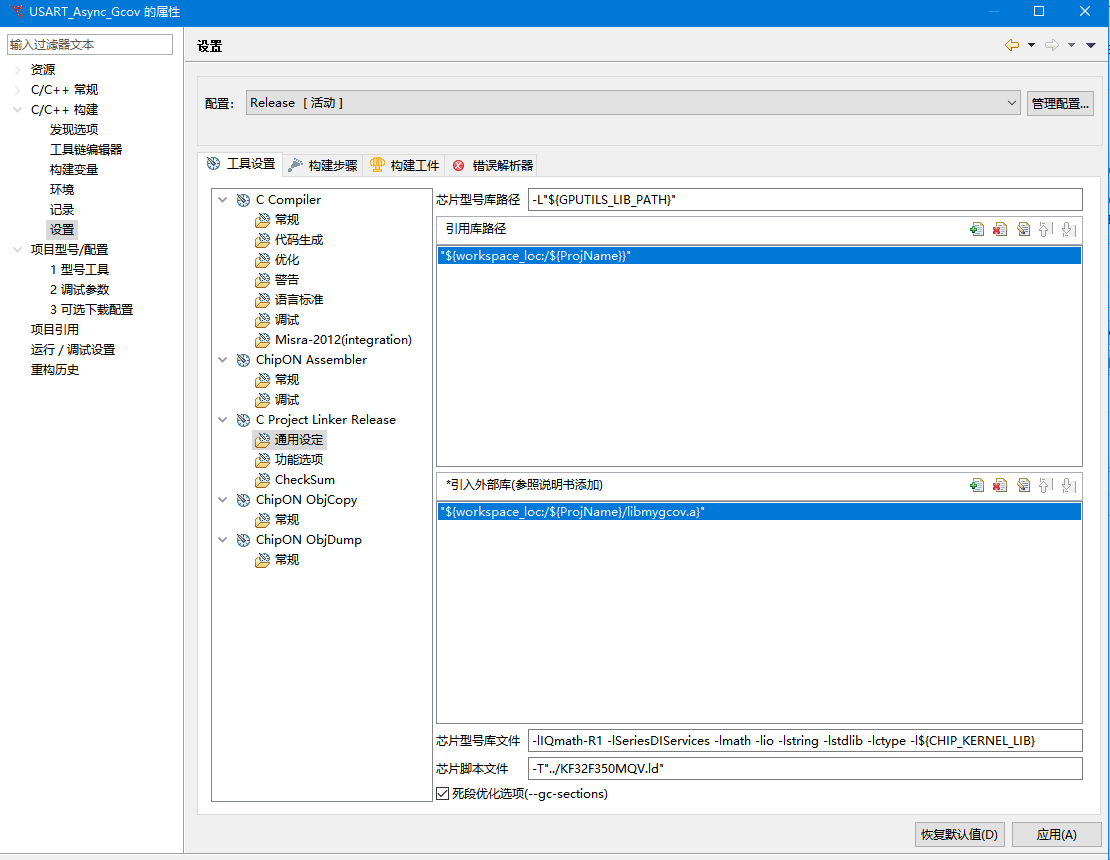
注:应该在项目中配置该串口的功能，如使能和波特率。

1. 可选使用默认示例输入和使用示例文件格式解析

即引入对应的默认脚本和makefile文件。运行脚本实现解析串口数据并输出覆盖率文件信息。具体见示例项目中相关文件示例实现。

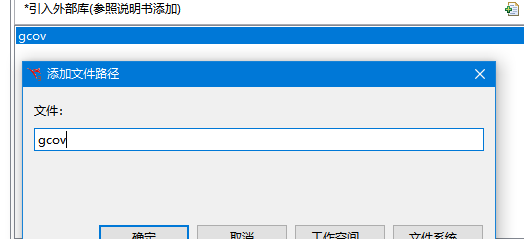
## 配置项目链接的使用该库

1. 使用工作空间选中该库添加到链接库文件中。



注：若使用工具资源中已集成的库，仅手动输入名字即可，即gcov。

1. 可以不通过工作空间的选择而直接输入，如mygcov gcov，即基于lib起始的命名a库文件名缩写。



1. 若库存放在其他路径，还应配置引用库搜索路径。

# 工作串口输出示例

如默认驱动源码示例，添加辅助文件断句信息和文件内容。

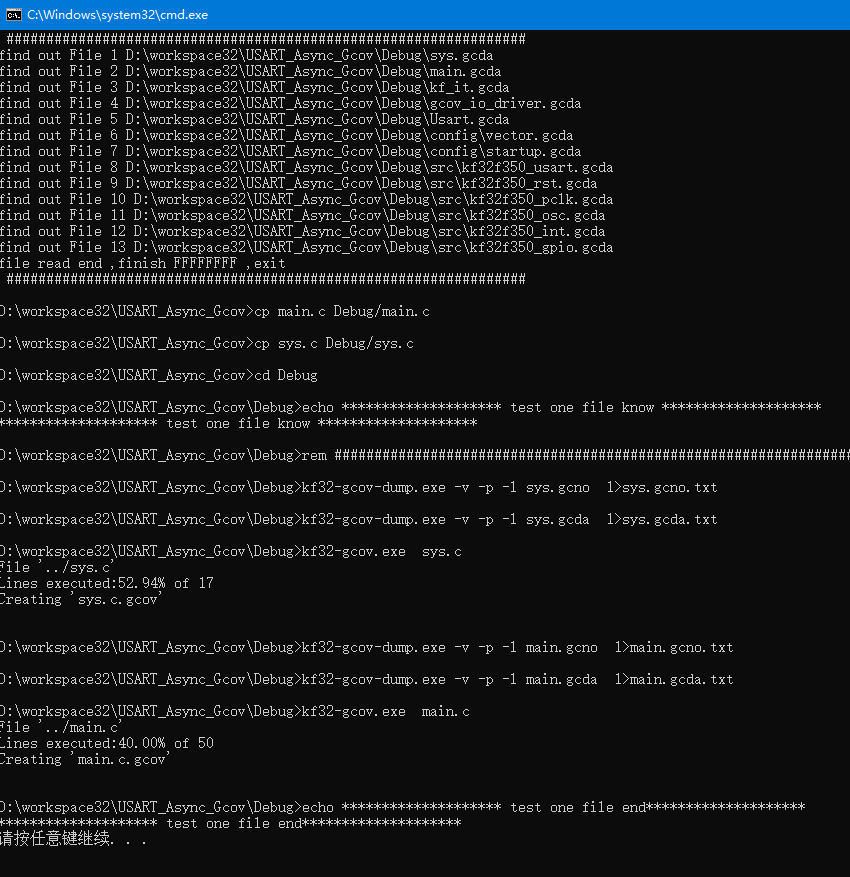


## 特殊处理与要求

其文件输出对应字符串与数据内容，即存在部分可见字符和数据内容。

串口推荐以hex数据格式接收并做自定义上位机的解析与输出。也可以按文本格式接收。但大多数串口上位机文本模式支持不佳，如多个\0内容的接收丢失，显示的结果复制到粘贴过程会失去对应的0数据内容。

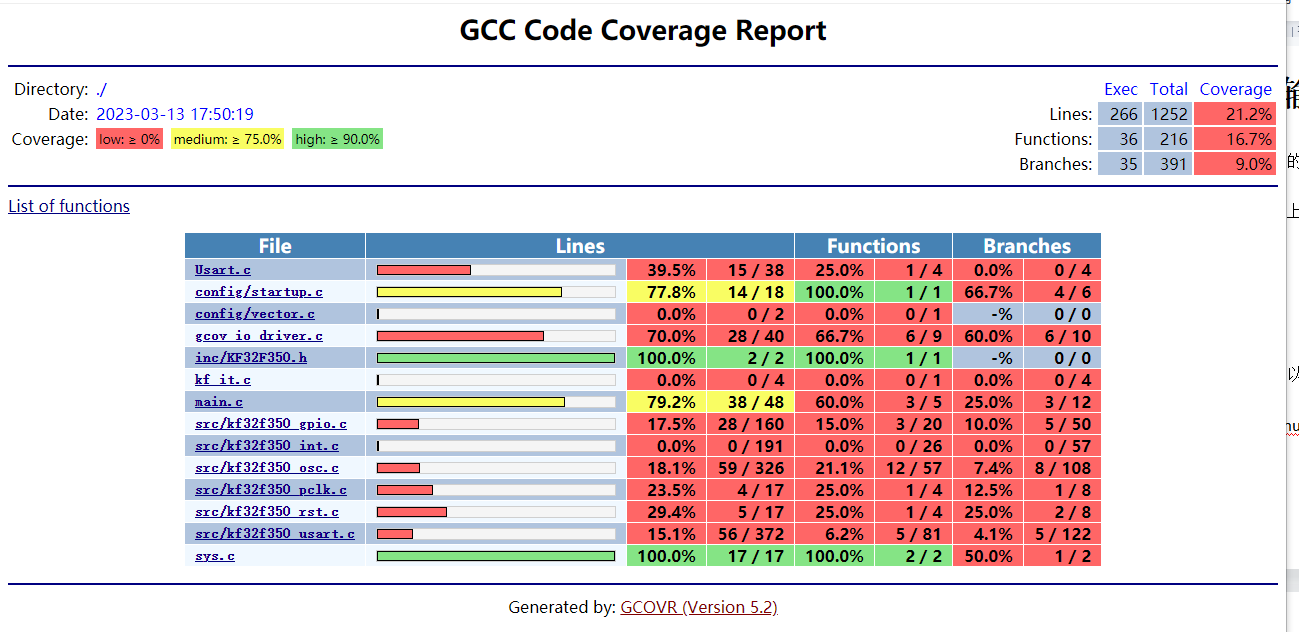
ChipON示例方案中提供格式文本的识别并在源项目相应路径输出识别的文件内容，推荐使用脚本完成该工作，其中该串口文件解析程序位于工具链的common目录下，其区分windows版本和linux版本。示例转换过程如下(windows)：



# 覆盖率文件输出

根据串口的内容输出对应的gcda文件。可使用如kf32-gcov main.gcno 的调用，其检索main.gcda 和main.c生成文件main.c.gcov。

随后可根据需要选用市面上lcov或gcovr等程序输出如html格式的可视化结果。其gcovr示例输出如下：





# 其他

附录中提供windows和linux平台下的示例项目。其将对应的同名项目串口输出文件存放于项目路径下的同名文本文件。提供对应批处理脚本bat或控制台脚本sh。运行脚本将结合该项目的makefile文件从而获取源码的相关路径信息，其中linux下还需要维护脚本配置ChipON工具链的位置。具体见项目下的脚本与模板makefile文件。