**ECR6600 SDK入门开发指南**

**修改记录**

| **版本** | **时间** | **说明** |
| --- | --- | --- |
| 1.0 | 2021-3-15 | 初始版本 |
| 2.0 | 2021-6-1 | 增加新增文件编译配置章节、增加常用编译链接选项 |
| 2.1 | 2021-7-23 | 增加系统变量配置说明 |

目 录

[第一篇 概述 5](#_Toc77947557)

[1 引言 5](#_Toc77947558)

[1.1 编写目的 5](#_Toc77947559)

[1.2 文档约定 5](#_Toc77947560)

[2 术语、定义和缩略语 5](#_Toc77947561)

[2.1 术语、定义 5](#_Toc77947562)

[2.2 缩略语 5](#_Toc77947563)

[3 编写依据 5](#_Toc77947564)

[第二篇 开发板硬件介绍 7](#_Toc77947565)

[4 硬件环境 7](#_Toc77947566)

[第三篇 开发环境搭建 9](#_Toc77947567)

[5 系统环境 9](#_Toc77947568)

[5.1 Linux 环境 9](#_Toc77947569)

[5.2 Windows环境 9](#_Toc77947570)

[6 Toolchain 10](#_Toc77947571)

[6.1 编译链工具获取 10](#_Toc77947572)

[6.2 交叉编译链工具解压 10](#_Toc77947573)

[第四篇 SDK编译使用 11](#_Toc77947574)

[7 SDK介绍 11](#_Toc77947575)

[7.1 文件路径定义 11](#_Toc77947576)

[7.2 SDK目录结构 11](#_Toc77947577)

[7.3 编译相关文件说明 12](#_Toc77947578)

[8 SDK编译 14](#_Toc77947579)

[8.1 编译环境配置 14](#_Toc77947580)

[8.1.1 Windows环境编译配置 14](#_Toc77947581)

[8.1.2 Linux环境GCC\_PATH环境变量配置 14](#_Toc77947582)

[8.2 menuconfig选项配置 14](#_Toc77947583)

[8.3 固件编译 16](#_Toc77947584)

[8.4 默认配置选项保存 16](#_Toc77947585)

[8.5 编译命令说明 16](#_Toc77947586)

[8.6 链接脚本 16](#_Toc77947587)

[9 新增文件编译配置 18](#_Toc77947588)

[9.1 新增参考设计目录 18](#_Toc77947589)

[9.1.1 Make.defs路径配置 18](#_Toc77947590)

[9.1.2 参考设计级别配置 18](#_Toc77947591)

[9.2 新增非参考设计目录 19](#_Toc77947592)

[9.2.1 添加drv\_demo.c和drv\_demo.h文件 19](#_Toc77947593)

[9.2.2 修改Kconfig、Make.defs等文件 19](#_Toc77947594)

[9.2.3 编译 20](#_Toc77947595)

[10 常用编译链接选项 21](#_Toc77947596)

[10.1 编译选项 21](#_Toc77947597)

[10.1.1 -I 选项 21](#_Toc77947598)

[10.1.2 -include 选项 21](#_Toc77947599)

[10.1.3 -D选项 21](#_Toc77947600)

[10.1.4 -O选项 21](#_Toc77947601)

[10.1.5 -mcmodel 选项 21](#_Toc77947602)

[10.1.6 -g选项 21](#_Toc77947603)

[10.1.7 -Wxxx/-Wno-xxx选项 21](#_Toc77947604)

[10.1.8 –Werror 21](#_Toc77947605)

[10.2 链接选项 22](#_Toc77947606)

[10.2.1 –nostartfiles选项 22](#_Toc77947607)

[10.2.2 –nostdlib选项 22](#_Toc77947608)

[10.2.3 -T选项 22](#_Toc77947609)

[10.2.4 -L选项 22](#_Toc77947610)

[10.2.5 -l选项 22](#_Toc77947611)

[10.2.6 -Wl,-Map选项 22](#_Toc77947612)

[10.2.7 -Wl,--whole-archive/-Wl,--no-whole-archive 22](#_Toc77947613)

# 

1. 概述

# 引言

## 编写目的

本文适用于ECR6600芯片SDK客户研发人员。包含ECR6600开发板的基本介绍，使用SDK所需的环境搭建，SDK结构介绍，以及SDK版本编译使用。

## 文档约定

无。

# 术语、定义和缩略语

## 术语、定义

本文使用的专用术语、定义见表2.1。

表 2.1 本文使用的专用术语

| **术语/定义** | **英文** | **说 明** |
| --- | --- | --- |
|  |  |  |
|  |  |  |

## 缩略语

本文使用的专用缩略语见表2.2。

表 2.2 本文使用的专用缩略语

| **缩略语** | **原文** | **中文含义** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# 编写依据

本文涉及的相关文档见表3.1。

表 3.1 涉及的文档

| 文件名称 | 版本号 | 说明 |
| --- | --- | --- |
|  |  |  |
|  |  |  |

1. 开发板硬件介绍

# 硬件环境

ECR6600开发板由USB供电，使用串口线连接USB和图3.1中所示UART0即可完成供电、版本下载、日志输出。

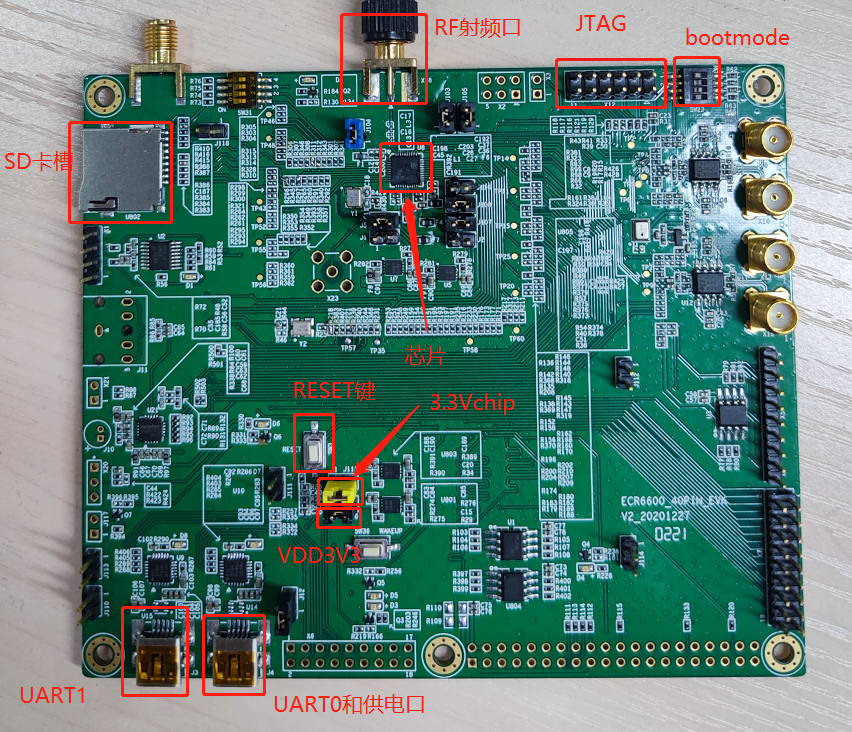


图 3.1 ECR6600开发板

对于bootmode拨码开关，1、2拨码共同控制bootmod0引脚，3、4拨码共同控制bootmode1引脚。具体如图3.2所示。

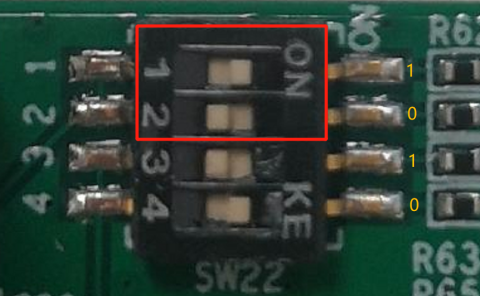


图 3.2 bootmode拨码开关

图中1、2拨码方式表示bootmode0配置为1；3、4拨码开关表示bootmode1配置为0。Bootmode引脚与boot方式的对应关系如表3.1。

表 3.1 menuconfig配置选项说明

| **Boot方式** | **Bootmode1** | **Bootmode0** | **拨码开关拨码方式** |
| --- | --- | --- | --- |
| SPI | 0 | 0 | 2、4拨到ON |
| SDIO | 0 | 1 | 1、4拨到ON |
| UART | 1 | 0 | 2、3拨到ON |
| SPI-FLASH | 1 | 1 | 全部OFF或者1、3拨到ON |

这里需要注意的是：

* 拨码开关1、2不可同时拨到ON，会造成短路；拨码开关3、4同样不可同时拨到ON。
* 默认状态下，全部拨码开关位于右侧，为spi-flash boot mode模式。
* 烧录时必须处于UART模式；或者是使用支持烧录的UBOOT，按复位按键后1秒内开始烧录。

1. 开发环境搭建

# 系统环境

ECR6600支持Linux和Windows开发环境，具体要求如下。

## Linux 环境

Ubuntu16.4 以上 + 交叉编译链工具

SDIO版本编译需要Linux内核版本4.20以上

为了保证menuconfig编译，还需要执行以下命令安装依赖项：

sudo apt-get install libc6:i386 #64位系统需要安装

sudo apt-get install flex

sudo apt-get install libncurses5-dev

sudo apt-get install bison

sudo apt-get install gperf

sudo apt-get install make

## Windows环境

32位Cygwin + 交叉编译链工具 + Python3

为保证menuconfig编译，必须安装以下工具：binutils、bison、flex、gcc-core、gcc-g++、git、gperf、libncurses-dev、make。版本不做严格要求，较新版本即可，如图4.1所示。

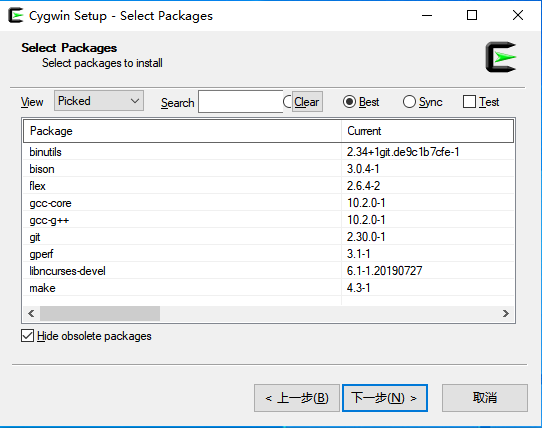


图 4.1 Cygwin Packages

在系统环境变量path中添加cygwin的路径：<your cygwin path>/cygwin/bin。

# Toolchain

## 编译链工具获取

请联系技术支持人员获取。

## 交叉编译链工具解压

* 确定使用的是Windows版本Toolchain，自行解压到合适的目录<your Toolchain path>。Toolchain的配置在SDK的配置文件中进行配置。

在系统环境变量path中添加Toolchain的路径：<your Toolchain path>/bin

* 确定使用的是Linux版本Toolchain。执行以下命令将交叉编译工具链解压到<your Toolchain path>：

# 拷贝该文件至你放置编译链的路径，切换到 Toolchain 压缩包所在目录

$ cd <your nds32le-elf-mculib-v3s.txz path>

# 解压

$ tar xJvf nds32le-elf-mculib-v3s.txz

1. SDK编译使用

# SDK介绍

## 文件路径定义

表 6.1 文件路径说明表

|  |  |  |
| --- | --- | --- |
| **文件路径** | **说明** | **示例** |
| <SDK> | SDK代码的顶层目录 |  |
| <SDK>/<subdir> | SDK代码的二级目录 | <SDK>/drivers |
| <SDK>/<subdir>/……/<subdir> | SDK代码的二级目录下的各级子目录 | <SDK>/drivers/uart |
| <REF\_DESIGN\_DIR> | SDK参考设计目录 | <SDK>/Boards/ecr6600/standalone |

## SDK目录结构

表 6.2 SDK目录结构说明表

|  |  |
| --- | --- |
| **文件路径** | **说明** |
| apps | 通用应用源码，可运行在不同arch和Boards平台 |
| arch | CPU架构相关操作，例如CPU初始化，CPU中断响应控制 |
| Boards | 具体参考设计相关源码 |
| build | 保存固件编译结果 |
| components | SDK提供的基础功能组件 |
| drivers | 外设驱动源码 |
| include | SDK公共头文件 |
| libs | SDK库文件目录 |
| os | 操作系统及适配层源码 |
| PS | PS协议栈默认编译配置项 |
| scripts | 编译开发工具，通用Makefile脚本和config工具源码 |
| dbg\_tools | SDK调试工具 |

表 6.3 Boards目录结构说明表

|  |  |
| --- | --- |
| **文件路径** | **说明** |
| common | 芯片通用代码，例如系统启动流程和链接脚本 |
| <REF\_DESIGN\_DIR> | 具体参考设计源码 |

表 6.4 Boards目录主要文件说明表

|  |  |
| --- | --- |
| **文件路径** | **说明** |
| generated | 编译过程中生成的文件，包括配置文件.config，头文件config.h以及连接脚本ld.script |
| main.c | 参考设计的主入口文件，包括各模块启动流程和主流程源码 |
| defconfig | 参考设计的默认编译配置选项 |

## 编译相关文件说明

工程的全部编译配置由menuconfig完成，最终由makefile编译链接。Menuconfig的全部选项由Kconfig配置实现，是否参与编译由Make.defs指定，最终由makefile确定生成库文件或者镜像文件。全部的文件及说明见表6.5。

表 6.5 工程编译相关文件

|  |  |  |
| --- | --- | --- |
| **文件路径** | **文件名** | **说明** |
| <REF\_DESIGN\_DIR> | Build.bat | cygwin编译启动脚本 |
| <REF\_DESIGN\_DIR> | makefile | Makefile框架编译入口，在< REF\_DESIGN\_DIR >目录下输入make编译命令，最终会调用到顶层主控Makefile |
| <SDK> | Makefile | 顶层主控Makefile，最终编译生成Makefile规则 |
| <SDK> | Make.defs | 获取关键路径变量和包含关键文件 |
| <SDK> | Kconfig | menuconfig顶级选项配置 |
| <SDK>/<subdir> | Makefile | 二级目录库编译规则 |
| <SDK>/<subdir> | Kconfig | 当前目录提供的menuconfig配置选项 |
| <SDK>/<subdir>/…/<subdir> | Make.defs | 指定当前目录中参与编译链接的源文件和头文件 |
| <SDK>/<subdir>/…/<subdir> | Kconfig | 当前目录提供的menuconfig配置选项 |
| <SDK>/scripts | Config.mk | 自定义函数实现 |
| <SDK>/scripts | CMarcoDefine.mk | PS协议栈宏值定义 |
| <SDK>/arch/andes | Toolchain.defs | 定义交叉编译工具路径、前缀、通用CFLAGS\AFLAGS\LDFLAGS、第三方lib库、共享头文件路径以及链接脚本路径 |

表 6.6 工程编译配置相关文件

|  |  |  |
| --- | --- | --- |
| **文件路径** | **文件名** | **说明** |
| <REF\_DESIGN\_DIR> | defconfig | defconfig提供了参考设计的默认编译配置选项，可由此生成.config |
| <SDK>/Boards/ecr6600/standalone/common | ld\_xip.script.S | 连接脚本配置文件 |
| <REF\_DESIGN\_DIR>/generated | .config | menuconfig自动生成参考设计的最终编译配置选项 |
| <REF\_DESIGN\_DIR>/generated | ld\_script | 编译过程中生成的链接脚本，用于最终的链接过程 |

# SDK编译

## 编译环境配置

### Windows环境编译配置

打开<REF\_DESIGN\_DIR>/build.bat文件，修改以下语句，将CYGWIN\_PATH设置成cygwin的安装目录。

SET CYGWIN\_PATH=<your cygwin tool path>\bin\

### Linux环境GCC\_PATH环境变量配置

将编译链临时添加到环境变量中

# 临时添加至环境变量中

$ export GCC\_PATH=<your Toolchain path>/nds32le-elf-mculib-v3s/bin/

GCC\_PATH中的<your Toolchain path>为章节5.2 中交叉编译链工具解压后的目录路径。该环境变量末尾的“/”是必须的，不可以省略。

## menuconfig选项配置

ECR6600 SDK的代码编译采用 menuconfig进行管理，进行编译的第一步就是menuconfig编译。执行如下命令：

#进入具体参考设计目录

cd <REF\_DESIGN\_DIR>

# 编译menuconfig

$ make menuconfig

即可看到如图7.1所示界面：

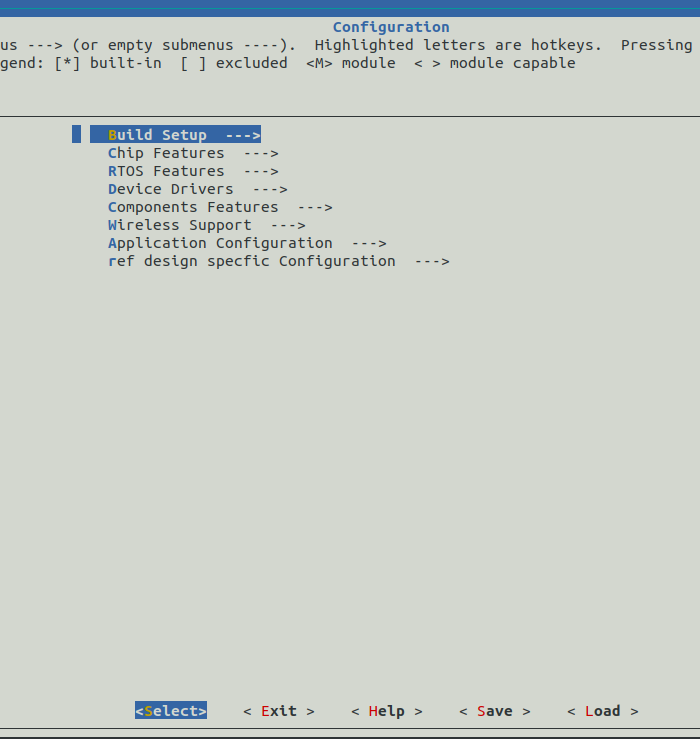


图 7.1 menuconfig页面

以上menuconfig选项的说明见表7.1。

表 7.1 menuconfig配置选项说明

|  |  |
| --- | --- |
| **menuconfig 选项** | **功能描述** |
| Build Setup | 编译配置选项，包括GCC编译选项，链接脚本类型，firmware输出格式等 |
| Chip Features | 当前芯片类型、时钟选项 |
| RTOS Features | 操作系统调试功能选项 |
| Device Drivers | 驱动选项 |
| Components Features | 组件选项 |
| Application Configuration | 应用程序配置 |
| ref design specfic Configuration | 参考设计独有的配置选项以及各参考设计无法兼容的配置选项 |

## 固件编译

完成menuconfig配置后，直接在当前目录开始编译：

# 编译SDK

$ make all

## 默认配置选项保存

完成menuconfig配置，功能验证完成后，如果需要将修改后的配置选项保存成默认配置选项，则执行如下命令：

# 保存默认配置选项

$ make savedefconfig

## 编译命令说明

SDK默认支持的命令如表7.3。

表 7.2 编译命令说明

|  |  |
| --- | --- |
| **命令** | **命令描述** |
| make menuconfig | 编译或启动menuconfig配置 |
| make all | 执行编译,自动解析.config生成config.h,完成编译链接 |
| make clean | 清空config.h和Build目录中的编译链接生成的文件 |
| make defconfig | 恢复默认config配置 |
| make savedefconfig | 将当前config配置保存至默认defconfig |
| make depend | 生成头文件依赖关系 |
| make distclean | 清空所有编译链接生成的文件 |

## 链接脚本

SDK使用的链接脚本命名为ld.script，位于<REF\_DESIGN\_DIR>/ generated/目录下，该链接脚本由<SDK>/Boards/ecr6600/common目录下的ld\_xip.script.S、ld\_ram.script.S等自动生成。ld.script链接脚本主要用于描述如何将输入文件（目标文件、库文件）中的段（text、data、bss等）映射到输出文件（ELF 文件）中，并控制输出文件的存储布局，输出文件的存储布局采用输出段（链接脚本中以“SECTIONS”标记）进行描述。

XIP版本默认链接脚本关键的输出段说明如表 7.4 所示。

表 7.3 链接脚本输出段

|  |  |
| --- | --- |
| **输出段** | **说明** |
| .xip | 1) XIP 段（可读、可执行），位于 FlASH 空间中。  2) 用于存储可执行代码和只读数据。在中断服务程序中执行的代码和访问的数据不能置于该输出段。 |
| .text | 1) SRAM text 段（可读、可执行），位于 SRAM 空间中。  2) 用于存储可执行代码和只读数据。 |
| .data | 1) SRAM data 段（可读、可写），位于 SRAM 空间中。  2) 用于存储初始值非零的全局变量、静态变量等，一般是输入文件的 data 段。 |
| .bss | 1) SRAM bss 段（可读、可写），位于 SRAM 空间中。  2) 用于存储初始值为零的全局变量、静态变量等，一般是输入文件的 bss 段。 |

# 新增文件编译配置

## 新增参考设计目录

新增参考设计New\_Ref\_Design时，需要在<SDK>/Boards/ecr6600目录下新建New\_Ref\_Design目录，必须包含如表8.1的文件：

表 8.1 参考设计目录文件

|  |  |
| --- | --- |
| **文件名** | **说明** |
| main.c | 实现main函数 |
| Make\_defs | 指定参考设计目录下参与编译链接源文件和头文件路径 |
| Makefile | 参考设计编译入口，直接从其他参考设计拷贝一份即可，无需重新实现 |
| Kconfig | 参考设计专用menuconfig配置选项或各参考设计不兼容的menuconfig配置选项，若没有相关配置选项，创建空文件即可 |
| Compile.mk | 参考设计专用的编译链接选项 |
| defconfig | 参考设计默认menuconfig配置选项 |

### Make.defs路径配置

新增Make.defs后，需要配置VPATH和INCPATHS变量，需要将变量中的路径配置为参考设计目录路径，格式如下：

VPATH += :ecr6600/New\_Ref\_Design

INCPATHS += ecr6600/New\_Ref\_Design/include

### 参考设计级别配置

#### GCC编译和链接选项

所有参考设计公共的GCC编译和链接选项位于<SDK>/arch/andes/Toolchain.defs文件中，针对参考设计单独需要的GCC编译和链接选项，可以在Compile.mk文件中定义，具体定义格式如表8.2：

表 8.2 Compile.mk配置说明

|  |  |
| --- | --- |
| **GCC宏变量** | **说明** |
| CFLAGS+= | 定义参考设计级别的C源码GCC编译选项 |
| AFLAGS+= | 定义参考设计级别的汇编源码的GCC编译选项 |
| LDFLAGS+= | 定义参考设计级别的GCC链接选项 |

#### menuconfig配置选项

针对参考设计单独需要或者各参考设计无法兼容的menuconfig配置选项，可以在<REF\_DESIGN\_DIR>/Kconfig文件中配置。

## 新增非参考设计目录

新增非参考设计目录的主要修改有：添加\*.c\\*.h文件，添加Kconfig控制选项，添加Make.defs编译控制，说明如下：

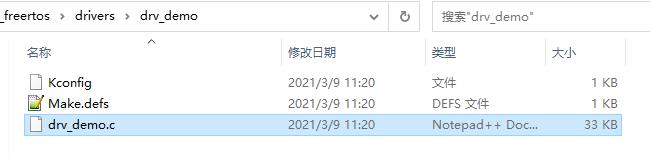
表 8.3 新增文件说明

|  |  |
| --- | --- |
| **文件** | **说明** |
| \*.c | 源文件 |
| \*.h | 头文件 |
| Kconfig | Menuconfig配置文件 |
| Make.defs | 指定新增目录中参与编译链接的源文件和头文件路径 |

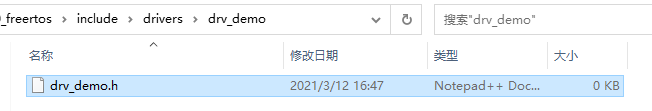
本小节以在<SDK>/drivers/drivers目录下添加一个drv\_demo驱动为例，展示全部要修改的文件。

### 添加drv\_demo.c和drv\_demo.h文件

1. 进入<SDK>/drivers/drivers目录，复制当前目录下的uart文件夹，重名为drv\_demo；
2. 删除drv\_demo文件夹中的uart.c，并添加drv\_demo.c；

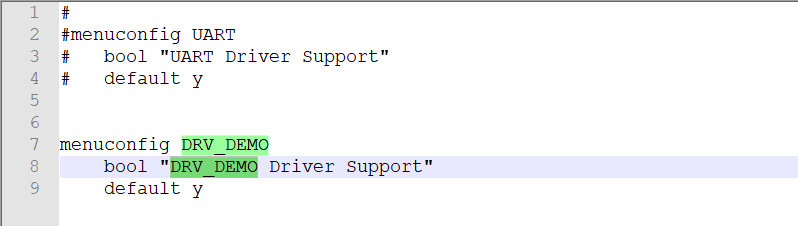


1. 在适当位置添加drv\_demo.h文件，例如进入<SDK>/include/drivers，新建drv\_demo文件夹；
2. 在drv\_demo文件夹中添加drv\_demo.h文件；



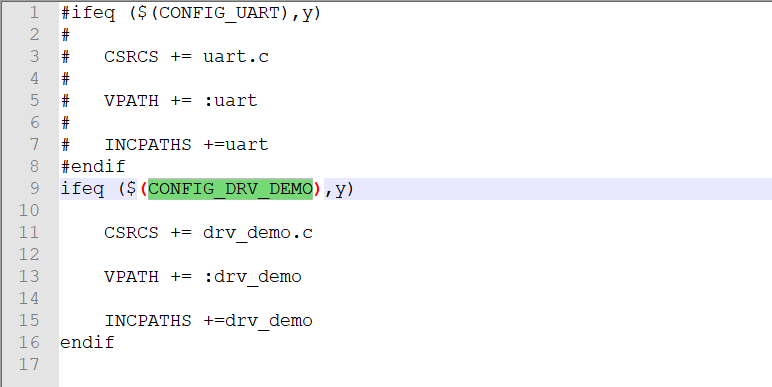
### 修改Kconfig、Make.defs等文件

1. 打开drivers/drv\_demo/Kconfig，按原文件格式修改即可。



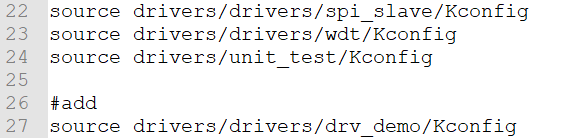
这里尤其值得注意的是，若新增的配置选项不是所有的参考设计都打开，那么将新增配置选项的default默认值设置为n。当参考设计需要该新增配置选项时，手动打开。

1. 打开drivers/drv\_demo/Make.defs，按原文件格式修改即可。



1. 打开drivers/Kconfig，在文件中添加一行：

source drivers/drivers/drv\_demo/Kconfig



### 编译

进入<REF\_DESIGN\_DIR>目录，执行make menuconfig，即可在Device Drivers选项下面看到DRV\_DEMO选项，并且已经被默认选择。执行make all即可完成编译。

# 常用编译链接选项

## 编译选项

### -I 选项

作用：指定头文件目录查找路径

格式：-I 头文件目录路径>

### -include 选项

作用：指定单个头文件路径>

格式：-include <头文件路径>

### -D选项

作用： 定义编译宏值

格式： -D<宏名>；

-D<宏名=整型宏值>；

-D<宏名=’”字符串宏值”’ >

### -O选项

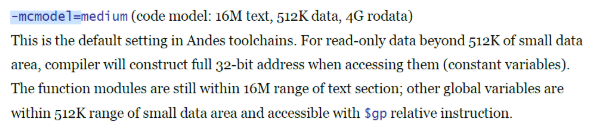
作用：编译优化等级

格式：-O0；-O1；-O2；-O3；-Os

### -mcmodel 选项

作用：主要用于指定函数跳转范围，large表示4G跳转空间，medium表示16M跳转空间。

格式：-mcmodel=large



### -g选项

作用：调试信息选项

格式：-g

### -Wxxx/-Wno-xxx选项

作用：打开/关闭编译warning

格式：-Wxxx/-Wno-xxx

### –Werror

作用：waring提升为编译报错

格式：-Werror

## 链接选项

### –nostartfiles选项

作用：不链接gcc提供的启动文件

格式：-nostartfiles

### –nostdlib选项

作用：不链接gcc默认提供的lib库

格式：-nostdlib

### -T选项

作用：指定连接脚本

格式：-T<连接脚本路径>

### -L选项

作用：指定lib库查找路径

格式：-L<lib库路径>

### -l选项

作用：指定lib库

格式：-T<lib库名>，其中lib库名要去除前缀“lib”和后缀“.a”

### -Wl,-Map选项

作用：指定输出map文件路径

格式：-Wl,-Map=<输出map文件路径>

### -Wl,--whole-archive/-Wl,--no-whole-archive

作用：指定默认参与链接的lib库

格式：-Wl,--whole-archive -l<lib库名1> -l<lib库名2> -Wl,--no-whole-archive