飞思卡尔单片机快速上手指南

飞思卡尔半导体 IMM FAE 团队

2014 年 9 月 · 审阅稿

飞思卡尔半导体是全球领先的单片机供应商，其单片机产品包含多种内核，有数百个系列。为支持用户使用这些产品，飞思卡尔提供了丰富的网站资源、文档及软硬件工具，另外，我们还有众多的第三方合作伙伴及公共平台的支持。对于不熟悉飞思卡尔产品和网站的初学者来说，了解和使用这些资源这无疑是一个令人望而生畏的浩瀚工程。本指南的目的，就是给初学者提供一个指导，让他们不被这些海量信息淹没；用户根据本指导提供的操作步骤，能迅速找到所需的资源，了解如何使用相关的工具。

在本指南中，我们以飞思卡尔的新一代 Kinetis 单片机 K22 系列为例，介绍了如何获取与之相关的资源，如何对其进行软硬件设计和开发。实际上，这些方法也适用于其它的单片机系列。当然，对于其它有较多不同之处的产品，我们也会继续推出相应的文档，供广大用户参考。

目录

第一章 如何获取技术资料与支持

1.1 概述

当用户使用飞思卡尔单片机芯片时，如何获取芯片的数据手册（Datasheet）、参考设计（Reference Manual）和官方例程等资源呢？另外当用户遇到了技术问题该如何获得帮助和解答呢？这里以 Kinetis 的 K22 系列芯片为例为大家介绍如何解决这些问题。

1.2 如何查找芯片的技术资料

1) 飞思卡尔芯片的资料全都可以在飞思卡尔的官网上免费下载。进入飞思卡尔官网 www.freescale.com ，如果您习惯使用中文，点击右上角 Select language 下的“中文”，选择显示中文，如图 1 所示，用户以后再进入该网站时，它就会自动默认为中文显示。

图 1. 飞思卡尔网站中文语言选择

2) 然后选择“产品”->“微控制器”->“Kinetis ARM ® MCU”，如图 2 所示。

图 2. 选择 Kinetis ARM MCU

3) 在 Kinetis 微控制器/单片机界面中，在左下方选择 K22_120 (120 表示主频为 120M)，如图 3 所示。点击此链接，可进入 K22_120 的产品页面。

图 3. 选择 K22_120

图 4. K22_120 资源集合

4) 如图 4 所示，进入产品页面之后，可以看到这里包含很多资源。在“文档”栏目中能找到芯片的数据手册、参考手册、应用笔记、用户指南等；在“软件和工具”栏目中能找到芯片使用的仿真调试器、评估开发板、软件开发工具等。您可以根据需要进行相关资源的下载。

另外您也可以通过飞思卡尔官方网站的搜索功能进行查找。方法是先进入官网 www.freescale.com, 在右上角的搜索框中输入 K22，然后点击搜索图标或按 Enter 键进行搜索，如图 5 所示。

图 5. 搜索关键字/芯片型号

搜索后的结果如图 6 所示，在这里同样可以获取您需要的相关资源。

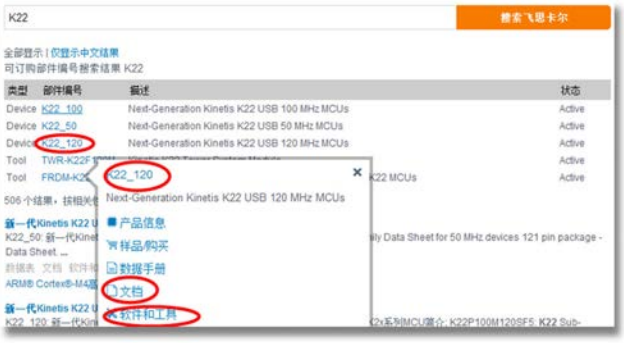


图 6. 搜索结果

另外，在飞思卡尔官网上有一些针对某些应用领域的参考设计，这些参考设计可以帮助您快速进行产品开发。以电表的设计为例，在飞思卡尔官网上点击：应用->工业->智能电网与智能仪表，如图 7 所示。之后在对应页面可以找到相关的参考设计，如图 8 所示。

图 7. 智能电网与智能电表

图 8. 参考设计

1.3 如何得到技术支持

当您在使用飞思卡尔芯片的过程中遇到问题时，您可以通过以下的途径得到支持：TIC、在线论坛和当地 FAE。

1.3.1 TIC (Technical Information Center，技术信息中心)

通过 TIC，您可以提交技术服务请求 SR（支持使用中文），之后会有专业的技术支持工程师为您在线解答。TIC 工程师都是飞思卡尔的资深工程师，他们分布在飞思卡尔全球各地的机构中，专门负责解答客户通过网络提交的问题。每个服务请求完成之后，都会向客户发送满意度调查表。

提交 SR 的具体操作步骤如下：

1) 进入飞思卡尔官网 www.freescale.com，点击支持服务与网络社区->销售与技术支持，如图 9 所示。进入销售与支持页面，点击创建服务请求，如图 10 所示。

图 9. 销售与技术支持

图 10. 创建服务请求

2) 选择技术服务类别和主题，如图 11 所示。

图 11. 技术类别与主题

图 12. 选择器件

3) 选择器件，如图 12 所示。
之所以要将问题进行分类，是为了将问题细化归类，从而有利于我们后台的 TIC 工程师进行问题分拣，进而能找到对所提交问题最有经验的工程师来解答。

4) 录入问题描述，可以添加附件，如图 13 所示。这里您既可以输入中文，也可以输入英文，我们鼓励用户的描述能尽可能的详细和清楚。

图 13. 问题描述

5) 最后提交技术服务请求(需要在飞思卡尔官网注册账号并登陆)。用户可以随时查看 SR（服务请求）的处理状态。正常情况下，我们的 TIC 工程师会在 24 小时内回复到用户注册的邮箱。之后用户就可以直接通过回复邮件的方式直接跟这个工程师沟通，而不需要再去重新提交问题，直到问题解决。如果是新的问题，则需要重新提交一次。 在用户的问题解决完之后，系统会自动发送一份满意度调查表到用户的邮箱，用户可以给本次服务作一个满意度评估。飞思卡尔也会根据评估来提高服务质量。

1.3.2 在线论坛

论坛包括两大类，一是飞思卡尔自己的官方论坛：https://community.freescale.com/welcome。

在这里用户可以先搜索是否有人曾遇到过类似的问题，或者创建自己的问题。提问时可以用中文，但最好用英文，这样就可以有全世界的技术人员进行解答。如图 14 所示。

图 14. Community 搜索与提出问题

还有一类是飞思卡尔第三方中文论坛，主要包括以下三个：

与非网: www.freescaleic.org/bbs/

21ic: http://bbs.21ic.com/iclist-192-1.html

阿莫论坛: www.amobbs.com/forum-9936-1.html

它们的二维码如下：

与非网　　21ic　　阿莫论坛

无论是飞思卡尔自己的论坛还是第三方的论坛，都有飞思卡尔的 FAE 和专门支持论坛的 TIC 工程师支持和回复。这里以与非网为例，可以看到版主均是飞思卡尔的工程师，而且数量很多。

图 15. 第三方论坛的飞思卡尔工程师版主

1.3.3 通过 FAE 得到技术支持

另外您还可以与代理商 FAE 或者飞思卡尔原厂 FAE 进行联系以获取相关的技术支持，建议您先与相应的代理商 FAE 联系。

飞思卡尔目前有六家代理商，分别为 Arrow（艾睿电子）、Avnet（安富利）、WT（文晔科技）、Weikeng (威健国际)、Comtech（科通）和 CEAC（中电器材）。

可以通过点击样品与购买->从分销商处购买，查看代理商的相关信息，如图 16 所示。

图 16. 进入代理商信息页面

进入界面之后，选择国家和城市，就可以查看到相应的代理商联系方式，如图 17 所示。需要注意的是有些代理商现在已经退出了，如 Fortune 和 WPI。这些信息在网站上暂时还没有更新，用户需留意。

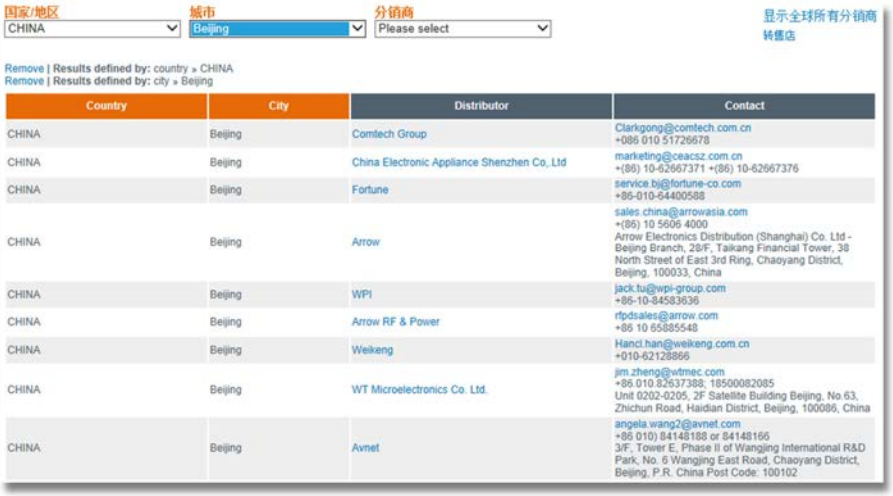


图 17. 代理商信息

1.4 如何查找中文文档

飞思卡尔为中国用户提供了很多中文文档，而且更多的中文文档正在陆续发布。为方便用户，部分使用频率比较高中文文档被整理在一起。用户点击“支持服务与网络社区”->“文档”即可看到，如下图 18 所示。

图 17. 飞思卡尔网站的中文文档

第二章 如何选择产品、申请样片及购买少量芯片和开发工具

2.1 概述

本章将介绍如何确定飞思卡尔单片机的具体型号，如何申请样片、购买芯片及开发工具。本章仍以 Kinetis K22 系列 MCU 为例进行说明。

2.2 芯片选型

飞思卡尔单片机的种类非常多，那么该如何获取合适的芯片呢？飞思卡尔提供了多种不同的方法和工具来帮助用户找到合适的产品。根据对飞思卡尔产品的了解程度和查找目的的不同，用户可以采用不同的方法进行选择。

2.2.1 主页产品查找

如果用户已经知道要查找的产品的基本信息，例如已经知道要找带 USB 功能的 K22 系列的 MCU，但是需要找到一个具体的芯片型号，最常用的方法是在主页进行查找，具体步骤如下：

1) 进入飞思卡尔官网 www.freescale.com，选择“产品”->“微控制器”->“Kinetis ARM ® MCU”，之后就会进入到 Kinetis 微控制器/单片机主页。如图 1 所示。

图 1. Kinetis 微控制器/单片机主页

2) 在主页的左下方 Kinetis 微控制器处，可以进入到每个系列 MCU 的各个子系列，在这里我们选择 K22_100，如图 2 所示。这里 100 表示芯片的主频为 100MHz。

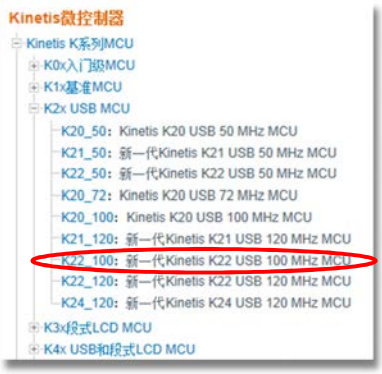


图 2. 选择 K22_100 子系列

3) 进入 K22_100 主页之后，选择购买/参数，就会出现该系列所有芯片的参数信息，如封装形式，管脚数量，内存大小等，可通过这些参数选择最合适的芯片，如图 3 所示。

图 3. K22_100 购买/参数

2.2.2 辅助选型工具

如果用户并不清楚需要查找的芯片基本信息，飞思卡尔还提供了一些辅助的选型工具来帮助用户进行芯片的选型，分别介绍如下。

2.2.2.1 参数选型工具

如果用户知道需要什么内核的产品，则可以通过“参数选型工具”来辅助进行芯片选型。步骤如下：

1) 进入飞思卡尔官网 www.freescale.com ，点击页面上方的参数选型工具。如图 4 所示。

图 4. 参数选型工具

2) 选择微控制器->Kinetis MCU(基于 Cortex-M 内核)，如图 5 所示。

图 5. 选择 Kinetis MCU

3) 在器件类型中，我们选择 K2x USB MCU，如图 6 所示。

图 6. 选择器件类型

4) 之后您可以进行多种操作，以辅助您选取到合适的芯片。比如您可以通过显示/隐藏参数来调整页面的参数显示类型，如图 7 所示。您还可以通过调整参数的范围来进行芯片的选取，如图 8 所示。

图 7. 显示/隐藏参数

图 8. 调整参数范围

2.2.2.2 选型神器 Cross Check

您还可以通过选芯神器 Cross Check(目前最新版本为 V3.0)来帮助您快速找到合适的飞思卡尔芯片，该软件具有以下功能：

1) 根据输入的芯片型号或参数特性来显示飞思卡尔相关器件的预算价，并能够订购样品或购买器件。

2) 根据用户输入的竞争对手的芯片型号，给出 4 款最适合的飞思卡尔器件。

该软件的下载地址为：

http://www.freescale.com/zh-Hans/webapp/sps/site/overview.jsp?nodeId=05D8D7194A&uc=true&lang_cd=zh-Hans

打开该网页后，可以扫描右侧的二维码进行下载手机应用软件，如图 9 所示。

图 9. Cross Check 软件下载

您也可以使用网页版的工具，方法是先登陆飞思卡尔官方网站，在我的账号-安全应用下，可以找到选型神器的应用，包括部件预算价和竞争对手交叉参考两个工具，如图 10 所示。图 11 和 12 是两个工具的使用界面。

图 10. 部件预算价和竞争对手交叉参考

图 11. 部件预算价工具

图 12. 竞争对手交叉参考工具

注：对于竞争对手交叉参考，可以不用登陆就能使用，位置在 Kinetis 微控制器/单片机主页，右侧的辅助工具-交叉参考工具以及下方的设计资源-支持和专业服务-MCU 竞争对手交叉参考。如图 13 所示。或者通过网址 www.freescale.com/crosscheck 进入。

图 13. 竞争对手交叉参考工具

2.2.2.3 解决方案顾问 Solution Advisor

飞思卡尔解决方案顾问是一款基于 Web 的互动式应用向导和动态产品选型工具。如果用户完全不知道要使用哪个芯片，则可以通过这个工具来查找合适的芯片。使用步骤如下：

1) 通过 http://freescale.transim.com/solutionadvisor/LandingPage.aspx 进入到解决方案顾问的主页，如图 14 所示：

图 14. 解决方案顾问主页

2) 选择产品选择器，如图 15 所示。在这里按照提示的先后步骤您可以快速找到最合适的处理器和工具。

图 15. 产品选择器

3) 此工具还提供了电机控制向导，如图 16 所示。它可以帮助您快速找到最合适的电机控制解决方案。

图 16. 电机控制向导

图 17. HMI 向导

4) 您还可以使用 HMI 向导，如图 17 所示。它可以帮助您快速找到最合适的 HMI 解决方案。

2.3 申请免费样片与购买芯片

2.3.1 申请免费样片

1) 在官网点击样品与购买-索要样品，可以进行免费样片的申请，如图 18 所示。

图 18. 免费样片申请

2) 选择微控制器，以 K22 举例，经过检索可以找到 K22 的相关样品，或者您可以直接通过微控制器 -> Kinetis -> K 系列-> K2x -> K22 的方式找到该芯片，检索界面如图 19 所示。

图 19. 检索样品

3) 检索到样片之后，点击黄色的“样品”按钮。如图 20 所示。

图 20. 选择样品

注：我们的样品在一定数量内为免费样品，并且快递费用全免。关于样片具体申请的个数及费用等相关信息，可以在常用问题简答中得到相关解释。如图 21 所示。

图 21. 常见问题解答

2.3.2 如何购买少量芯片

2.3.2.1 在飞思卡尔官网上购买

1) 在飞思卡尔官网上注册。如果您已经是会员，请登录网站。
2) 搜索器件。找到可订购器件的 “直接购买”按钮，该图标位于器件号右侧。如图 22 所示。

图 22. 直接购买图标

3) 点击“直接购买”按钮图标，将产品添加到您的购物车中。当您将产品添加到购物车后，您可以更新您的购物车，继续购物或结帐。如图 23 所示。

图 23. 购物车

目前，我们支持支付宝、VISA 和 Master 卡来进行网上直接在线支付。但是客户需要自己负责清关（手续并不麻烦，但是在海关要求的情况下需要客户自己处理。因为是小批量订单，而且我们不知道客户的具体用途，所以需要客户自己清关）

4) 您在下订单之后，将会收到一封确认电子邮件。您可以对已注册用户主页进行个性化定制，随时跟踪订单情况。 (到货时间视产品供货情况和配送方式而定。)

2.3.2.2 第三方供应商小批量芯片购买

除了飞思卡尔官网之外，客户还可以用过以下与飞思卡尔有官方合作的芯片在线销售公司来购买小批量芯片：

Element14: http://www.element14.com/community/welcome

E 络盟： http://cn.element14.com/

周立功： http://www.zlgmcu.com/

DigiKey: http://www.digikey.com/

Mouser: http://eu.mouser.com/

2.3.3 购买开发板

当您需要购买开发板时，可以在飞思卡尔官网上进行购买，步骤如下：

1) 点击 MCU 界面的软件与工具，在硬件开发工具中查看该系列 MCU 有哪些开发板。这里还以 K22 举例说明，如图 24 和 25 所示。

图 24. 软件与工具

图 25. 硬件开发工具

2) 点击相应的开发板之后，可以进入这个开发板的主页，这里以 FRDM-K22F 为例。可以看到相关的各种资源，在“文档”栏目中能找到开发板的用户指南等；在“下载”栏目中能找到开发板的原理图、Sample code、快速开发包等；选择购买/规格 就可以进行购买。如图 26 所示。

图 26. 购买开发板

另外从飞思卡尔的第三方合作伙伴，如 uCDragon(优龙科技)、Manley(万利电子)、周立功和 lierda(利尔达)，同样可以购买到他们为飞思卡尔芯片定制的开发板。

他们的网址分别为：

优龙科技：http://www.ucdragon.cn

万利电子：http://www.manley.com.cn

周立功：http://www.zlgmcu.com

利尔达：http://www.lierda.com

以优龙科技为例，首先进入官网 http://www.ucdragon.cn，在产品分类中的 ARM 开发板中可以找到飞思卡尔相关的开发板。如图 27 所示。

图 27. uCdragon freescale 开发板

点击相应的开发板之后，可以获得该开发板的信息，通过页面左下方的联系方式，你可以购买到该开发板，如图 28 所示。

图 28. 优龙的销售联系方式

第三章 飞思卡尔单片机的开发环境、开发工具和生态系统

3.1 概述

在根据需求完成选型之后，用户应开始了解相关的开发环境、开发工具和生态系统，评估飞思卡尔提供的工具和资源是否能够满足用户的开发需要。本章节将会介绍集成开发环境（IDE）、评估板（EVM）、调试器（Debugger）、参考代码与设计（Sample code with documents & Existing Reference Design）以及生态系统 (ecosystem)。希望本章节的内容能让用户快速准确地找到相关资源来完成这一过程。

3.2 集成开发环境（IDE）

集成开发环境就是针对可编程器件的集编辑、编译和调试功能于一体的开发调试工具。以 Kinetis 系列 MCU 为例，通用的 IDE 均可支持，例如 IAR、Keil 等。另外飞思卡尔也有自己的 IDE，分别是 Kinetis Development Studio(KDS) 和 CodeWarrior，本节将会主要介绍这两个工具及内嵌在这两个 IDE 中的代码生成工具 Process Expert（处理器专家）。

3.2.1 Kinetis Design Studio (KDS)

KDS 是飞思卡尔在 2014 年刚刚推出的专门用于支持 Kinetis 系列 MCU 的一款集成开发环境软件，能够提供代码编辑到调试的完整功能。KDS 的特点如下：

- 支持目前 Kinetis 全系列的产品，并将不断更新对新产品的支持；
- 开源免费，且不受软件代码大小的限制（提醒：Keil、IAR 等 IDE 均需要收费）；
- 支持在 32/64 位的 Windows 7/8 的操作系统上运行；
- 使用 Elipse 界面风格，并且支持其他可下载的插件，如 Processor Expert；
- 支持 SEGGER J-Link/P&E USB Multilink Universal/CMSIS-DAP/OpenSDA 等调试接口；
- 专门为 Kinetis 系列 MCU 开发，因此具有内核编译器小、响应速度快的优势。

3.2.1.1 KDS 软件下载与安装

目前 KDS 的最新版本是 V1.1.1，以下描述均以该版本为例。软件与相关说明文档的下载路径如下：

软件：*中文首页-> 软件和工具-> 软件中心->IDE- 调试、编译与构建工具-> 微控制器->Kinetis Design Studio-> 下载 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KDS_IDE&fpsp=1&tab=Design_Tools_Tab)*

文档：*中文首页-> 软件和工具-> 软件中心->IDE- 调试、编译与构建工具-> 微控制器->Kinetis Design Studio-> 文档 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KDS_IDE&fpsp=1&tab=Documentation_Tab)*

在 Windows 系统上安装 KDS 很简单，只需双击 KDS-v1.1.0.exe，根据向导即可完成。另外，飞思卡尔还提供了一个工具，叫做 Kinetis 软件开发套件（KSDK），它可以嵌入 KDS 中使用，为用户提供丰富的外设驱动代码、协议栈代码、中间件代码和示例代码。关于 KSDK 的详细介绍，请参见第六章的内容。

在完成 KDS 的安装后，如果需要在 KDS 中嵌入 KSDK，则还要下载和安装 KSDK 软件包，软件与文档的下载路径如下：

软件：*中文首页-> 软件和工具-> 软件中心-> 中间件-> 用于 Kinetis MCU 的软件开发套件-> 下载*
(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK&fpsp=1&tab=Design_Tools_Tab)

文档：*中文首页-> 软件和工具-> 软件中心-> 中间件-> 用于 Kinetis MCU 的软件开发套件-> 文档*
(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK&fpsp=1&tab=Documentation_Tab)

在下载安装 KSDK 后，要嵌入 KDS 中使用要完成以下步骤：

1) 启动 KDS 并在“Help”选项中选择“Intall New Software”

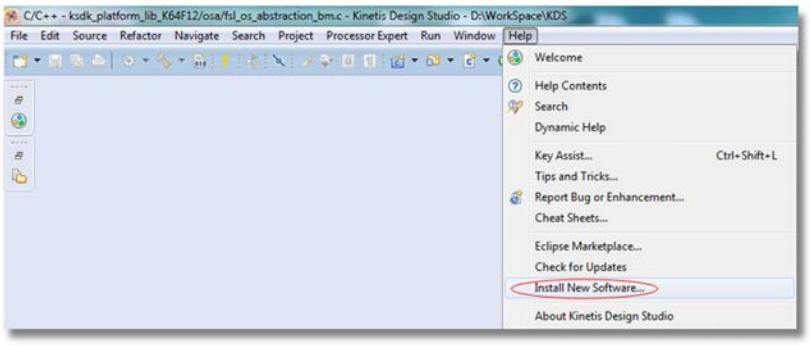


图 1. 在 KDS 中安装 KSDK 步骤一

2) 在“Available Software”的窗口中，点击“Add”并找到 KSDK 安装目录中的.zip 文件，点击 OK。

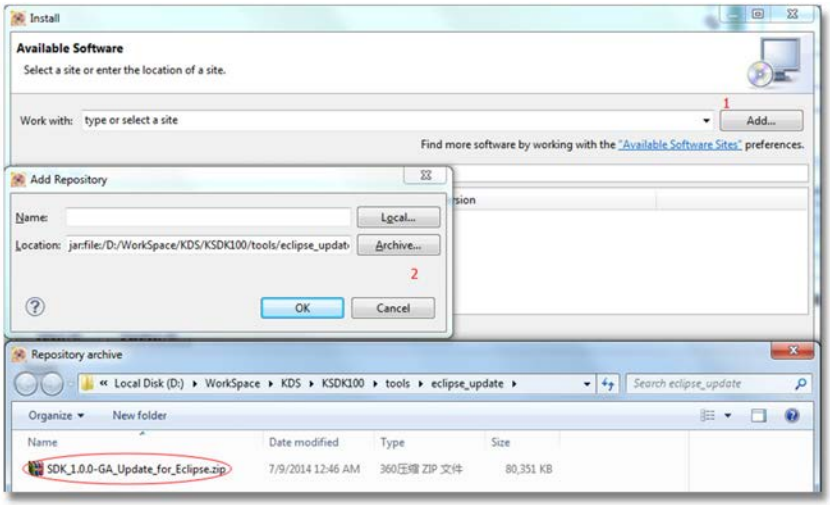


图 2. 在 KDS 中安装 KSDK 步骤二

3) 找到文件位置后，在“the Processor Expert Software category”的项目下显示有“Eclipse Update for KSDK”；在勾选框内打勾，并点击“Next”并完成安装。

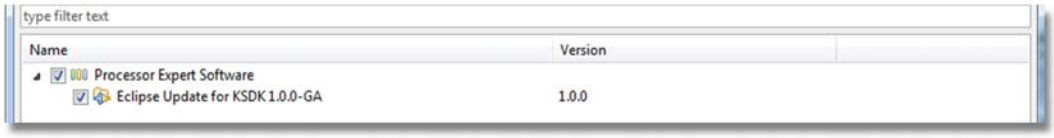


图 3. 在 KDS 中安装 KSDK 步骤三

另外，如果需要在 KDS 中使用 Processor Expert，还需要安装一个更新补丁。该补丁名为 Processor Expert for KDS 1.1-Updata 1，下载路径与 KSDK 相同，安装方法也与安装 KSDK 相同，可参照上述步骤进行操作。

3.2.1.2 在 KDS 中新建工程与导入已有工程

在 KDS 中新建一个工程可参考以下步骤：

1) 在打开 KDS 后，选择新建工程：File->New->Kinetis Design Studio Project。
2) 输入工程名，并在选择工程的存储位置后，点击“Next”。

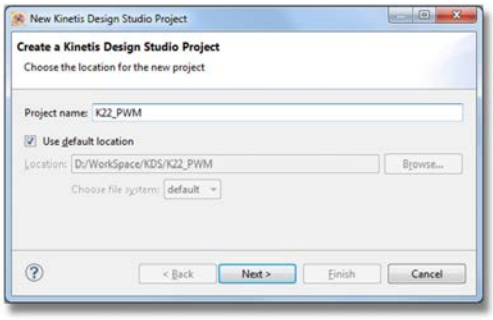


图 4. 在 KDS 中新建工程步骤二

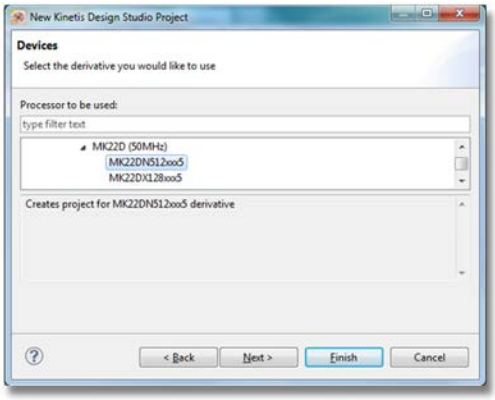


图 5. 在 KDS 中新建工程步骤三

3) 选择需要评估的芯片型，点击“Next”。
4) 在“Rapid Application Development”中，可以在新建工程中加入 Process Expert 和 KSDK 来简化底层驱动代码的编写工作。Processor Expert 已包含在 KDS 中，但加入 KSDK 需要按照 3.2.1.1 节中的步骤手动安装。

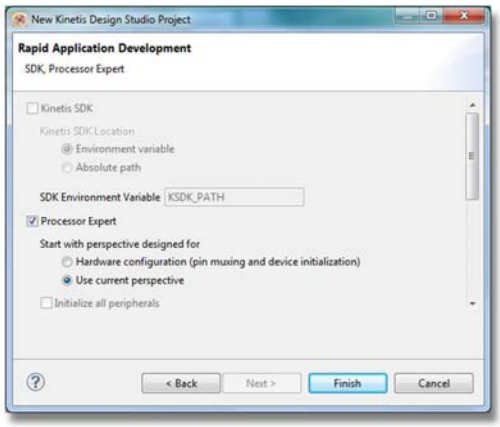


图 6. 在 KDS 中新建工程步骤四

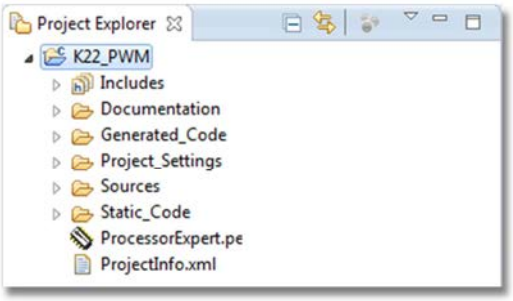


图 7. 在 KDS 中新建工程步骤五

5) 在完成上述配置后，点击“Finish”可以看到如图 7 所示的 Project Explorer 界面。

6) 在工程中，可以通过右击项目名选择 New->Source File 来新建文件，也可以通过配置拖拽功能将已有的源文件、头文件和目录等加入到工程中。

导入已有工程可参考以下步骤：

1) 在打开 KDS 后，选择 File->Import。
2) 在 Import 页中，选择 Existing Projects into Workspace 并点击“Next”。
3) 在 Import Projects 页中，选择 Select root directory 并点击“Browse”来选择已有工程的目标文件夹，选好后点击“OK”。

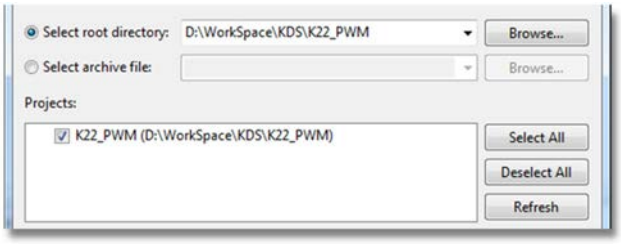


图 8. 在 KDS 中导入已有工程

4) 在 Projects 框中会列举出目标文件夹中所有的工程，在需要导入的工程前的选项框内打勾，并点击“Finish”。

3.2.1.3 编译与调试

在开始编译程序之前，需要预先配置好工程的一些编译参数。右击要编译的 Project 并选择 Properties，在 Proporties for example 页中可以看到工程的所有参数。点击展开“C/C++Build”后在 Settings 中可以看到跟编译有关的参数，默认的参数一般无需修改，如需修改，请参照说明文档。

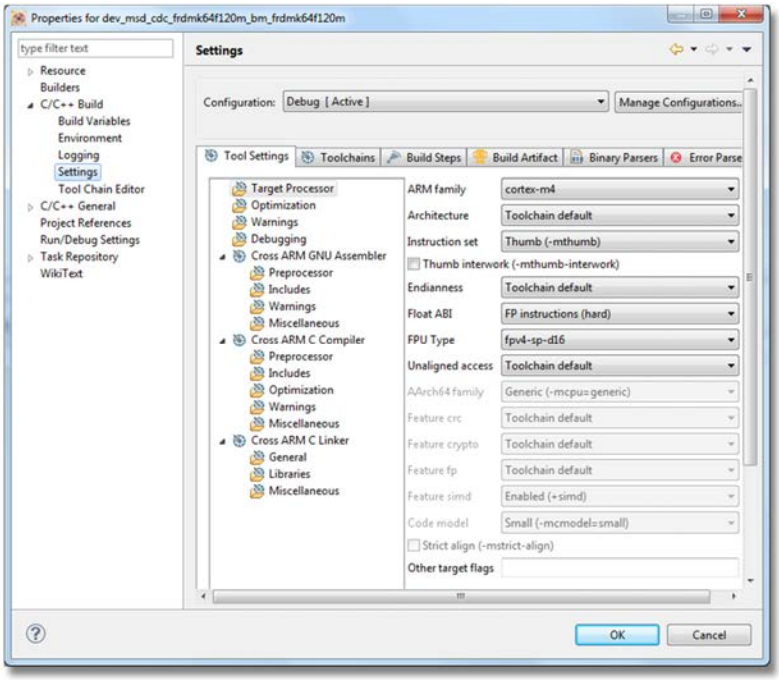


图 9. 在 KDS 中配置编译参数

在点击编译  图标后，在相应的工程文件中看到编译生成的.elf 文件，如下图所示。

图 10. 编译后生成的可下载文件

在开始下载之前，需要先配置相关调试参数，如选择调试接口等。右击需要调试的工程并选择 Debug As->Debug Configurations 进入 Debug 配置页面，如图 11 所示。也可以点击调试按键右侧的下拉箭头来进入此配置页面。在“Main”选项卡中，需保证 C/C++ Application 一栏中所选的.elf 文件为需要调试的工程对应产生的编译文件。对于“Debugger”和“Startup”两栏，根据调试器的不同，需要进行不同的配置，如图 11 所示。这部分内容会在第 3 节中作具体介绍。

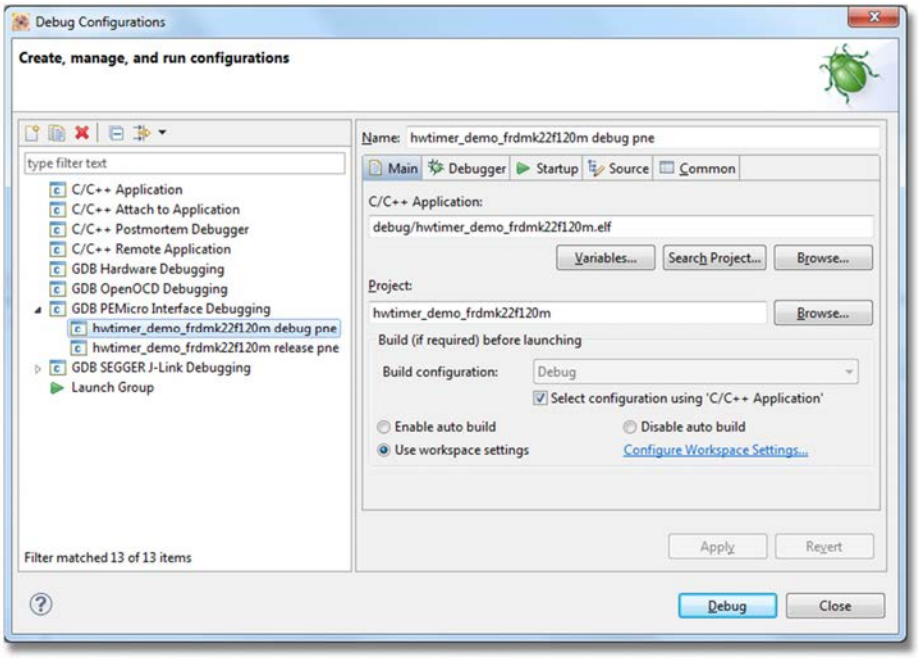


图 11. 在 KDS 中调试配置

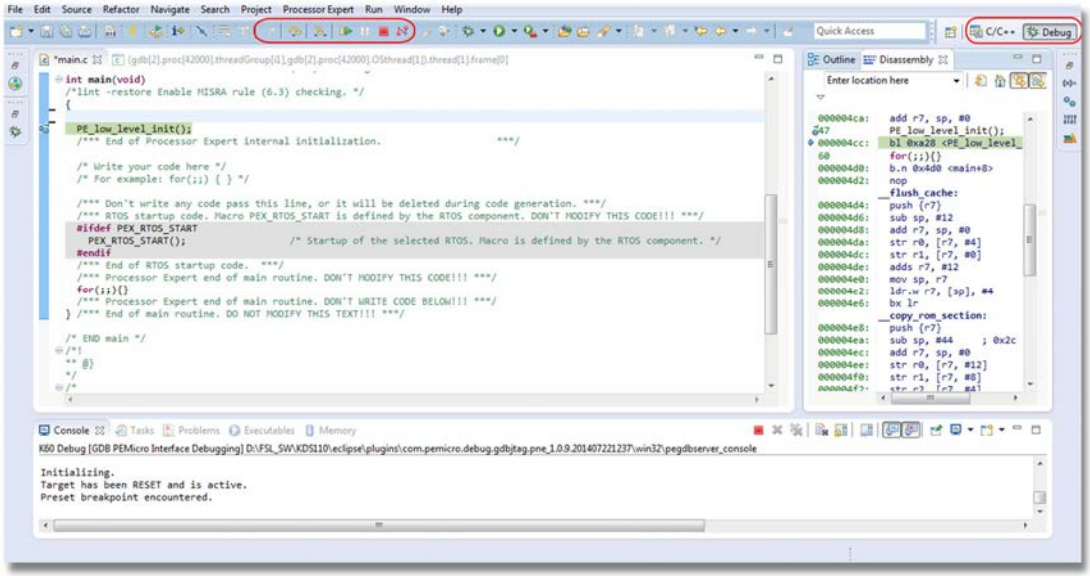


图 12. 在 KDS 中进行调试

配置完成后，将目标板与 PC 机通过 USB 线和相关调试器连接起来，依次点击 Apply 和 Debug 两个按键进入 Debug Perspective，如 12 图所示，可以程序已暂停在 main()函数的初始处。这样就可以实际调试程序了。下表列出了工具条中各按键的主要作用。

表 1. 在 KDS 中调试按键 功能

按键	说明	按键	说明
	开始/继续执行		单步返回函数
	暂停执行		复位
	使能汇编单步执行		断开仿真器
	单步执行		结束调试
	单步进入函数		开始下载与调试

3.2.1.4 下载

在程序调试完成后，需要将调试好的程序下载到 FLASH 中。

点击下载按钮  ，进入 Flash Configurations 界面。配置可参考 2.1.3 小节中的要求相同。在完成配置后，依次选择 Apply 和 Flash，等待程序下载完成后，重新上电或手动复位后程序开始运行。

3.2.2 CodeWarrior 集成开发环境

CodeWarrior（简写为 CW）原是 Metroworks 公司的产品，后来该公司被飞思卡尔收购，因此 CW 也就成为了飞思卡尔自己的 IDE 系统，目前已在业内使用多年。从 10.0 版本以后，CW 开始使用 Eclipse 界面，能够支持不同架构的芯片，包括 ColdFire、ColdFire+、DSC、Kinetis、PowerPC、Qorivva、RS08、S08 和 S12Z 等。

目前针对飞思卡尔单片机的 CW 的最新版本是 CodeWarrior for Microcontrollers v10.6。另外根据客户的用途和需求不同分为评估版、基础版、标准版和专业版，其中评估版是免费的，其支持代码和数据大小也是有限制的。以 Kinetis 系列 MCU 为例，K 系列的限制为 128KB，E/L/M/V 系列的限制为 64KB，如超出限制，则需要申请更高级的版本。

另外需要提醒的是，对于 v10.6 以后的版本，CodeWarrior 不会再支持其后发布的 Kinetis 系列的产品，因此对于使用 Kinetis 新产品的用户，需要将软件平台逐步转移到其他 IDE 上，由于 CodeWarrior 与 KDS 所建立的工程兼容，因此推荐使用 KDS 完成 IDE 平台的切换。

CodeWarrior 软件的安装包和相关文档的下载路径为：

软件：飞思卡尔中文首页->软件和工具->软件中心->IDE-调试、编译与构建工具->微控制器->CodeWarrior for MCUs->下载

(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=CW-MCU10&fpsp=1&tab=Design_Tools_Tab)

文档：飞思卡尔中文首页->软件和工具->软件中心->IDE-调试、编译与构建工具->微控制器

->CodeWarrior for MCUs->文档
(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=CW-MCU10&fpsp=1&tab=Documentation_Tab)

在支持的调试接口、新建工程与导入工程、调试与编译等方面，CodeWarrior 与 KDS 基本相同，此处就不作介绍，具体细节可以参照相关说明文档。

3.2.3 处理器专家（Processor Expert）

Process Expert (简称 PE) 是一款支持 Kinetis、Coldfire 等多种不同内核的 MCU 的快速应用开发软件。该软件以图形化界面的方式快速配置 MCU 的内核及外设，生成相应的初始化及应用代码，省去手动编写底层驱动代码的繁复工作，从而大大提高了软件工程师的工作效率。其生成的代码可以支持 KDS、CodeWarrior、IAR 和 Keil 等多种 IDE；另外，PE 还以嵌入到基于 Eclipse 架构的 IDE 中，例如在 KDS 和 CodeWarrior，可以直接使用嵌入的 PE 功能，PE 所产生的代码将自动放置在相关工程下，然后在此基础上修改主程序就可以进行编译和调试。

KDS 和 CodeWarrior 这两种 IDE 安装后都带有内置的 PE，无需再额外安装。因此其安装过程就不再介绍。

3.2.3.1 新建基于 PE 的工程

下面以 Kinetis K22 为例，创建一个使用 PE 的 KDS 工程。工程所需的具体配置如下：

1) 首先 在 KDS 的选项栏中，点击 File 栏并选择 New->Bareboard Project 来创建一个新的工程。
2) 出现工程向导界面如下，输入工程名并点击 Next。

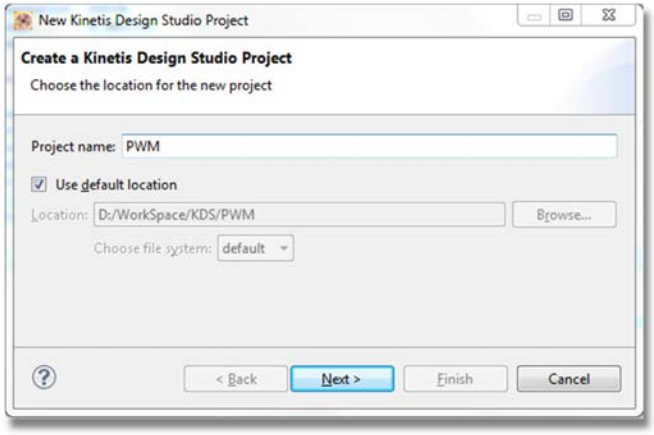


图 13. 工程设置向导

3) 选择芯片类型，此处选择 Kinetis K Series->K2x Family ->K22。点击 Next。
4) 在 Language and Build Tool Options 页中，选择 C 语言并 点击 Next。
5) 在 Rapid Appliation Developmen 页中，在 Rapid Appliation Developmen 组中，选中 Processor Expert 并点击 Finish。至此一个包含 PE 功能的工程已建好，下面开始添加元件来配置所需的外设。
6) 在 Component Library 中，找到相应组件（如 FTM）并右击选择 Add to project 选项。
7) 点击打开添加的组件，其具体配置显示在 Component Inspector 中。
8) 根据所需要求进行以下配置初始化等参数。

9) 在配置完成后，点击图标  产生相应代码。

10) 所产生的代码存放在 Generated_Code 文件夹中。可以看到 FTM 的相关初始化代码已自动完成，点击功能键即可编译运行后。

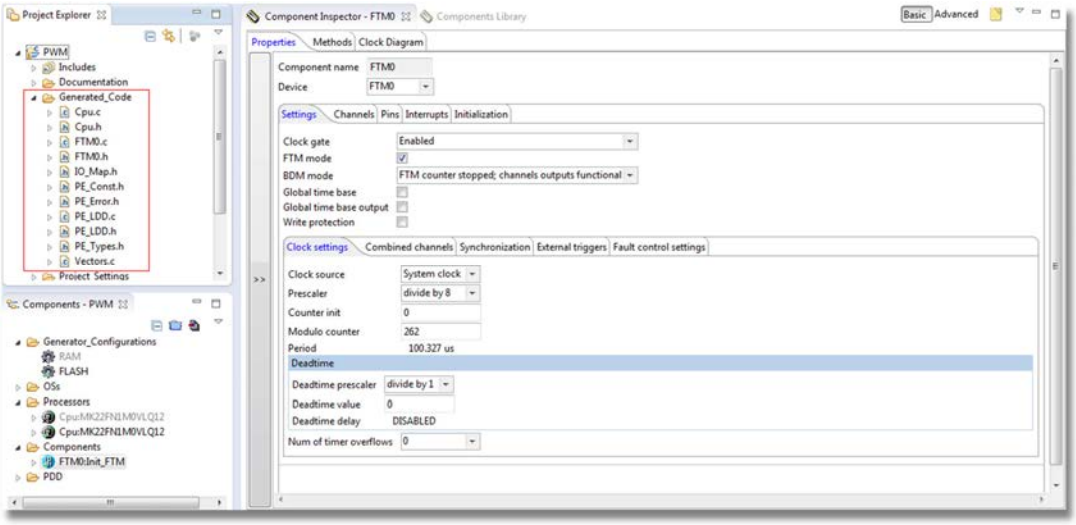


图 14. 在 KDS 中使用 Processor Expert 生成代码

3.3 调试工具（Debugger）

调试工具是开发工具链中的重要一环，方便耐用的开发工具对用户非常重要。接下来就介绍三种常用的调试工具。

3.3.1 OpenSDA 调试功能

OpenSDA 是飞思卡尔自主开发的一种调试平台。在该平台中烧写不同的调试固件，可以使其兼容不同的调试工具（如实现 PEMicro Segger 的 JLink 接口等）。下面介绍如何在 FRDM-K22F 板上使用 OpenSDA 来实现 Jlink 调试。

1) 在 Segger 的官网上下载用于 OpenSDA 的固件库。下载地址为：
   http://www.segger.com/opensda.html
2) 如果 FRDM-K22 板通过 USB 线连接到 PC 机上，请将其拔下。
3) 按住 SW1 键，然后用 USB 线连接 PC 机与板上的 SDA USB 接口。
4) 松开 SW1 键，可见板上一个 LED 灯在有规律地闪烁，且在 PC 机上能看到一个移动存储设备，其盘符为 BOOTLOADER，表示目标板进入了 bootloader 模式。

图 15. OpenSDA 处于 BOOTLOADER 模式下的显示

5) 将步骤 1）中所下载的压缩包 Jlink_OpenSDA_V2_1.zip 解压缩，得到 Jlink_OpenSDA_V2_1.bin 文件。

6) 将上述.bin 文件复制粘贴到 BOOTLOADER 设备中并拔下 USB 线。

7) 重新插上该 USB 线，在设备管理器中可见到如下设备，表示固件烧写成功。

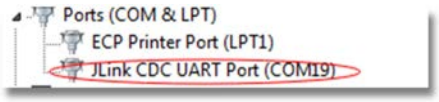


图 16. 固件下载成功后设备管理器中显示

这样在 IDE 中就可以选择以 Jlink 调试下载程序了。另外 OpenSDA 还带有虚拟串口调试功能，串口格式如下。

波特率：115200 bps; 1 位起始位；8 位数据位；无校验位；1 位停止位。

在 KDS 中关于使用 OpenSDA 设置如下图所示。

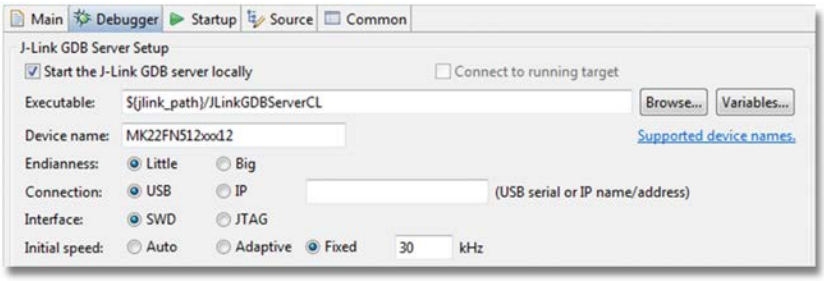


图 17. 在 KDS 中使用 Jlink 调试配置

3.3.2 JLink

Jlink 是 SEGGER 公司为支持仿真 ARM 内核芯片推出的 JTAG 仿真器，支持 Cortex M0/M1/M3/M4 等内核芯片，并与 KDS、CodeWarrior、IAR 和 keil 等多种开发环境无缝连接，使用方便。

Jlink 驱动的下载地址如下：http://www.segger.com/jlink-software.html，该驱动软件不仅提供了程序下载功能，另外 J-Flash 软件还可以不需要 IDE 软件独立地擦除与烧写程序，以及实现查看芯片的 ID、RAM/FLASH 大小、解锁芯片等功能。

下图为 Jlink 的 20 脚的接口图。

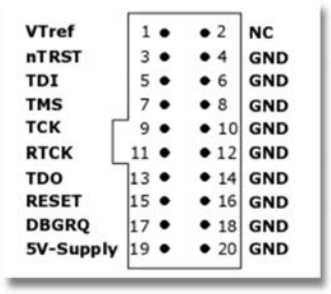


图 18. JLink 的标准 20 芯接口

3.3.3 Multilink

P&E 推出的 USB Multilink Universal（FX）是一款高速一体化开发接口（USB Multilink Universal FX 比 USB Multilink Universal 下载速度更快），它支持 Kinetis、DSC 等多种 MCU。因此其包含多种调试接口以匹配不同种类的 MCU，下图为支持 Kinetis 系列相关的三种调试器。

在使用时，其连接顺序如下:

1) 确保目标板断电，并断开 USB Multilink Universal 与外部的连接。
2) 打开 Multilink 的塑料外壳，选择合适的接口将其与目标板的调试接口连接起来。
3) 将 Multilink 与 PC 机通过 USB 线相连。此时蓝灯亮。
4) 将目标板上电，此时黄灯亮。

在断开连接时，请先将目标板断电。使用时请遵守连接顺序，否则可能到损坏调试器。

3.4 评估板（EVB）

为了缩短用户评估所选芯片以及提供硬件设计参考，飞思卡尔提供了大量的评估板，大部分评估板不仅包含了 MCU 的最小系统、引出的通用 IO 口、提供 JTAG 调试接口、板载调试器 OpenSDA，还针对不同应用的芯片配置了 USB 口，触摸滑条和液晶显示等功能电路和接口。另外，这些外设的驱动程序在官网上也都一应俱全。针对 Kinetis 系列的评估，评估板按结构分为两种，分别是 FRDM 板和 Tower 板（塔式系统板）。一般来说 FRDM 板相对便宜，功能简单，而 Tower 板功能更加丰富。如下图所示。

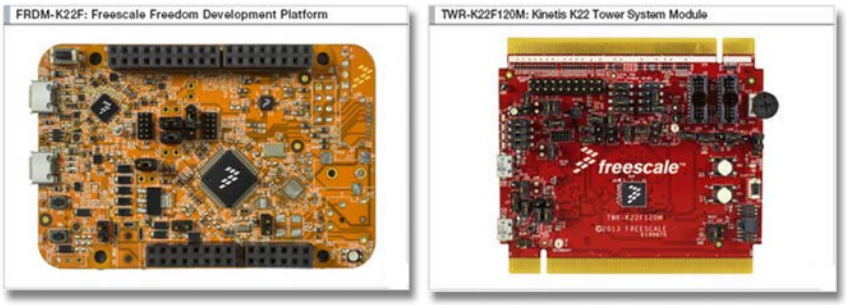


图 19. FRDM 板与 Tower 板

如何根据所选的 MCU 型号找到最适合用于评估的 EVM 板是评估的第一步。下面以 K22 为例，用两种方法来找到所需的评估板。

1) 首先找到介绍 K22 的官网主页，路径为产品->Kinetis ARM@ MCU->K 系列->K2x USB MCU->K22_120MHz。在该主页的右栏中间可以可以看到“推荐软件和工具”，其中可以看到针对 K22 有 FRDM-K22F 和 TWR-K22F120M 两种评估板。

图 20. 搜索合适的评估板的方式一

2) 在飞思卡尔的中文首页中，在“软件和工具”一栏中选择硬件开发工具。在该页面中，可以找到针对 Kinetis 系列的评估，有塔式系统模块化开发平台和飞思卡尔 Freedom 开发平台。这里选择塔式系统模块开发平台。在该页中，选择展开 MCU 与处理器模块->Kinetis MCU 模块，如下图所示。可见所有的 Tower 板都已列举出，在其中总可以找到跟您所选用的型号相同或相近的评估板。

图 21. 搜索合适的评估板的方式二

下面就以 FRDM-K22F 和 TWR–K22F120M 为例，介绍如何使用评估板。

3.4.1 Freedom 开发平台（FRDM 板）

飞思卡尔 Freedom 开发平台是一种小型化、低功率和高性价比的评估和开发系统，十分适合针对 Kinetis MCU 系列器件进行快速应用原型设计和制作演示。关于 FRDM-K22F 的原理图/软件包和说明文档的下载路径如下：

下载路径：*中文首页->产品->Kinetis ARM@MCU->K 系列->K2x USB MCU->K22 120->推荐软件与工具(FRDM-K22)->下载 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=FRDM-K22F&fpsp=1&tab=Design_Tools_Tab)*

文档路径：*中文首页->产品->Kinetis ARM@MCU->K 系列->K2x USB MCU->K22 120->推荐软件与工具(FRDM-K22)->文档 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=FRDM-K22F&fpsp=1&tab=Documentation_Tab)*

K22 的 FRDM 板上基本硬件单元包括：

- 电源电路

板上 K22F 的输入 5V 电源可由直流插座 J23 或用 SDA 的 USB 口提供，输入的 5V 电源经过 LDO 输出 3.3V 供给板上主 MCU 及其他外设。在测量 MCU 功耗时，可将 R62 和 R63 两电阻取下，将 J15 线引出即可测量供给 MCU 的电流值以便测量功耗。

- JTAG 调试电路

板上引出了 SWD 的调试接口，只需通过拿掉 J10 和 J13 的跳线来断开 OpenSDA 与主 MCU 的连接即可。J11 的封装为标准的 10 脚封装（脚间距 0.05”）。

- 晶振电路

MCU 上电后初始化为使用内部时钟源，可以通过修改软件选择外部时钟源，所支持的外部

主晶振的频率范围为 32~40KHz 以及 3~32MHz。该板上的晶振频率为 8MHz。另外板上的 32.768KHz 的晶振用于给 RTC 提供时钟源。

- 复位电路

板上按键 SW2 给主芯片提供复位信号，同时 SW2 也与 OpenSDA 连接，长按 SW2 也可使 OpenSDA 进入 booloader 模式。

3.4.2 塔式系统模块化开发平台（Tower 板）

飞思卡尔塔式系统是一个为 8 位、16 位和 32 位微控制器设计的模块化开发平台，基于该平台，研发人员可通过开速原型技术进行样机研制。通过这个平台，用户可以采用搭积木的方式将各种功能板组合在一起，实现用户需要的各种功能。它为设计者提供了基本的模块单元，以满足微控制器进一步开发的需要。关于 TWR-K22F120M 的原理图/软件包和说明文档的下载路径如下：

下载路径：*中文首页->产品->Kinetis ARM@MCU->K 系列->K2x USB MCU->K22 120->推荐软件与工具(TWR-K22F12M)->下载*

(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=TWR-K22F120M&fpsp=1&tab=Design_Tools_Tab)

文档路径：*中文首页->产品->Kinetis ARM@MCU->K 系列->K2x USB MCU->K22 120->推荐软件与工具(TWR-K22F12M)->文档*

(http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=TWR-K22F120M&fpsp=1&tab=Documentation_Tab)

3.4.3 如何申请 EVB

关于如何得到相关的评估板，有两种方法。

1) 直接联系代理商，向其索要评估板。代理商的具体联系方法请参见 1.3.3 节的介绍。

2) 在飞思卡尔网上订购，如下图所示。

图 22. 获取评估板的方式

3.5 Kinetis Software Development Studio (Kinetis 软件开发套件)

Kinetis SDK 是针对于 Kinetis 系列 MCU 所做的软件开发套件，又称为 KSDK。它由强大的外设驱动代码库，协议栈库与示例代码库等部分组成，能够简化和加快对于 Kinetis 系列 MCU 的应用开发。另外，Kinetis SDK 是免费的工具，而且所有的硬件抽象和外设驱动软件均开放完整源代码。目前最新版本为 1.0.0，支持 K22、K24、K63、K64 和 KV3x 这五种型号的 MCU，随后会不断更新完善，直至能覆盖支持所有的 KInetis 系列 MCU。

3.5.1 KSDK 的下载与安装

以下是 Kinetis SDK 的下载路径及相关文档路径。

下载路径：*中文首页->软件和工具->软件中心->软件开发套件->用于 Kinetis MCU 的软件开发套件->下载 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK&fpsp=1&tab=Design_Tools_Tab)*

文档路径：*中文首页->软件和工具->软件中心->软件开发套件->用于 Kinetis MCU 的软件开发套件->文档 (http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK&fpsp=1&tab=Documentation_Tab)*

下载安装包后，只需根据向导即可完成安装。在安装结束后，可以安装路径中可看到如下文件夹，这些就是 Kinetis SDK 的主要内容。

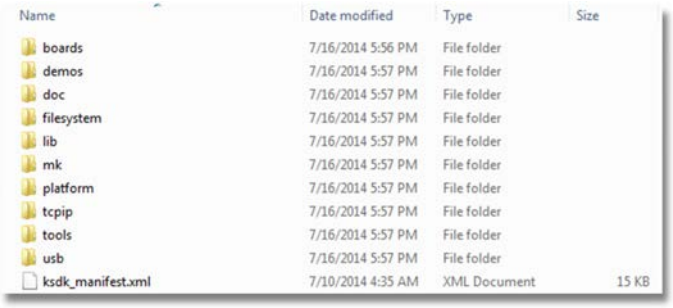


图 23. KSDK 的完整文件

开始使用 SDK 之前，还需要完成以下两个步骤：

1) 在 KDS 中安装 SDK 有关的 eclipse 插件。这部分内容在 2.1.1 节中有介绍，这里不再重复。
2) 设置环境变量。右击“我的电脑”，依次点击“属性”、“高级系统设置”和“环境变量”可以看到如何添加新的环境变量，并按下图所示方式相应设置。

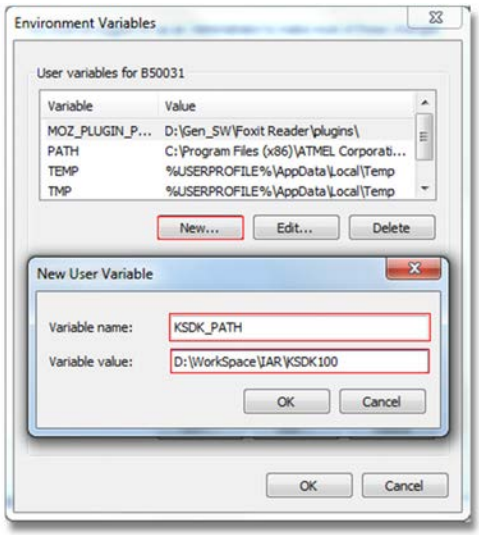


图 24. 设置环境变量

3.5.2 KSDK 的结构介绍

Kinetis SDK 的结构如下所示，它包含了由底层头文件到硬件抽象和外设驱动软件、以及用户应用的完整功能。由图中不同模块的放置位置，可以看出模块之间调用与被调用的关系。下面对各个模块逐一进行介绍。

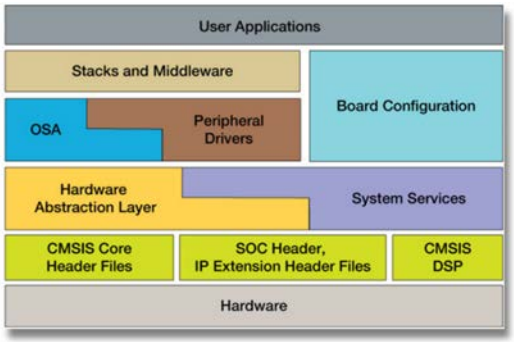


图 25. KSDK 的总体结构

1) CMSIS Core Header Files/SOC Header, IP Extension Header Files/CMSIS DSP

这一层是芯片主要寄存器的地址定义，包含了与 CMSIS 兼容的内核定义头文件和 DSP 运算库。此外，还有针对不同型号芯片的外设定义头文件。所在位置路径为：安装目录->platform->CMSIS。

2) Hardware Abstraction Layer

这一模块称为硬件抽象层，简称 HAL，主要负责基础模式的设置，可以理解为对底层寄存器的设置，省去了手工配置寄存器的麻烦。针对每个外设 IP，都有相应的 HAL 文件与之对应。同时每个 HAL 文件也只负责与该外设 IP 的配置。因此，HAL 层具有一定底层驱动的抽象意义，但所完成的功能相对简单。所在位置路径为：安装目录->platform->hal。

3) System Services

系统服务模块主要涉及一些常见的系统功能，包括中断管理、时钟管理、功耗管理和定时器管理。这一模块常与 HAL 配合使用，提供给上层的 OSA 和 Perpheral Drivers 模块以完成更多的功能。所在位置路径为：安装目录->platform->system。

4) Peripheral Drivers

这一模块称为外设驱动层，简称 PD。PD 所完成的功能比 HAL 更加丰富，主要通过调用 HAL 和 System Svevices 来实现，通常 PD 可能调用一个以上的 HAL 模块。它们两者是不同的，以 UART 模块举例来说，UART 的 HAL 模块只是完成了字节形式的收发功能，而其 PD 模块则能支持基于中断的数据流传送，或将 DMA 与 UART 配合使用。目前 PD 模块支持了多数的典型用法，但也没有列举完全，如 UART 的 PD 模块不支持智能卡等，这些应用会在以后更新中陆续加入。所在位置路径为：安装目录->platform->drivers。

5) OSA

OSA 的全称为 Operating System Abstraction，操作系统抽象层。OSA 用于设置 SDK 在一些操作系统上运行，当然也支持裸板模式。其包含了操作系统 Kernel 的大多数服务的抽象，这些操作系统包括 MQX、FreeRTOS、µC/OS-II、µC/OS-III。所在位置路径为：安装目录->platform->osa。

6) Stacks and Middleware

这一层包含了一些软件堆栈与协议等，如 USB 协议栈、TCP/IP 协议栈和文件系统等。

7) Board Configration

针对不同的 EVM 板，都有相应的 SDK 例程与之对应。这一模块用于配置不同 EVM 上的管脚复用、外设时钟给定等来实现兼容。安装目录->boards。

另外，KSDK 提供了许多完整例程，可以基于 EVM 板运行，帮助用户学习 KSDK 的结构与功能，所在位置路径为：安装目录->demos。关于如何移植 KSDK 来完成用户应用的开发，请参考第六章的介绍。

3.6 如何获取参考代码和参考设计

在开发应用软件时，用户不仅可能需要针对底层单个外设的驱动，还需要比较复杂功能的软件实现，甚至是一个产品或应用的整体方案。飞思卡尔有许多资源能够帮助用户在不同的开发状态下顺利完成项目。

3.6.1 例程代码（Sample Code）

例程代码对于 MCU 软件开发上手十分重要。下面给出四种找到相关代码例程的方法。

- Kinetis SDK（适合于已涵盖型号的 MCU）

如上述介绍，KSDK 是一种功能强大的软件开发套件，但目前还没有覆盖所有的型号。对于已经涵盖的型号，除了通过前面介绍的路径直接找到外，也可从所涵盖型号的主页下的“软件与工具”选项页中找到。

- Sample Code Package（SC，适合于尚未被 SDK 包含的 MCU）

对于目前还没有被 SDK 涵盖的 MCU 型号，会由包含了针对某个型号的几乎所有模块驱动库的例程代码来支持。相比较 SDK 而言，SC 的结构简单，更易于上手，但功能较弱。另外其一般被放置在对应型号的 Tower 板的主页中，以 K64 为例，路径如下：

下载路径：K64 主页->软件和工具->TWR-K64F120M->下载->软件开发工具

图 26. 在对应 MCU 下的 sample Code

- IDE 安装路径下的 Sample Code（适合于安装 CodeWarrior 等 IDE 的用户）

对于安装了 CodeWarriord 的用户，在其安装目录下也能找到针对不同系列的许多 Sample Code。对于 Kinetis 的 K 系列为例，其路径如下：

CodeWarrior 的安装根目录->MCU->CodeWarrior_Examples->Kinetis_Examples

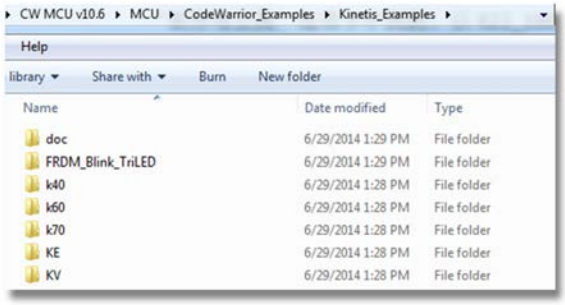


图 27. CodeWarrior 安装目录下的 Sample Code

- 代码片段等

同样在某个型号 MCU 的主页中的软件开发工具中也有相应的实现某个具体功能的代码片段、引导代码等可供参考。例如：

图 28. 代码片段

3.6.2 应用笔记

对于比较复杂功能的软件实现，例如如何在 Sigma-Delata ADC 上实现 FFT、如何使用 DMA 和 GPIO 来模拟 PWM 等，有关实现的功能原理与实现步骤都会以应用笔记的形式记录下来，而且一部分应用笔记还带有示例程序。

要准确快速地找到相关应用笔记，首先应该在所选芯片的文档中查找，对于带有例程的应用笔记，也会有图标显示。

图 29. 搜索应用笔记的方式一

另外，使用文档搜索功能，能够更全面地查找到已有的 AN。下面以 K22 为例给出搜索所有相关 AN 的步骤。

1) 点击首页上的搜索按键

图 30. 搜索应用笔记的方式二步骤 1

2) 点击左边栏中的“文档”。

3) 在“过滤方式”中，依次选择 Kinetis_MCU、K2x_USB_MCU。文档类型中选择应用说明。

图 31. 搜索应用笔记的方式二步骤 2、3

3.6.3 参考设计

参考设计通常是针对某个应用的完整解决方案，包括硬件设计、软件代码和原理说明等。使用“软件和工具”中的高级搜索功能能够帮助您找到最合适的参考设计。下面介绍如何搜索与 Kinetis 的 K 系列相关的 DRM（Design Reference Manual）。

1) 点击飞思卡尔官网主页上的“软件和工具”。在中间一栏的下方找到该功能“高级搜索”。
2) 在软件和工具一栏中选择“参考设计”。
3) 在支持的器件中递进式地选择“微控制器”->“Kinetis MCU”->“K 系列”。
4) 在应用领域中选择 “工业”，如下图所示。

图 32. CodeWarrior 安装目录下的 Sample Code

在选择提交后，就能看到应用 K 系列 MCU 在工业领域中的参考设计。需要用户注意的是，由于飞思卡尔网站上的中文搜索功能不是很完善，建议用户使用英文进行搜索。这样搜索到的结果如图 33 所示：

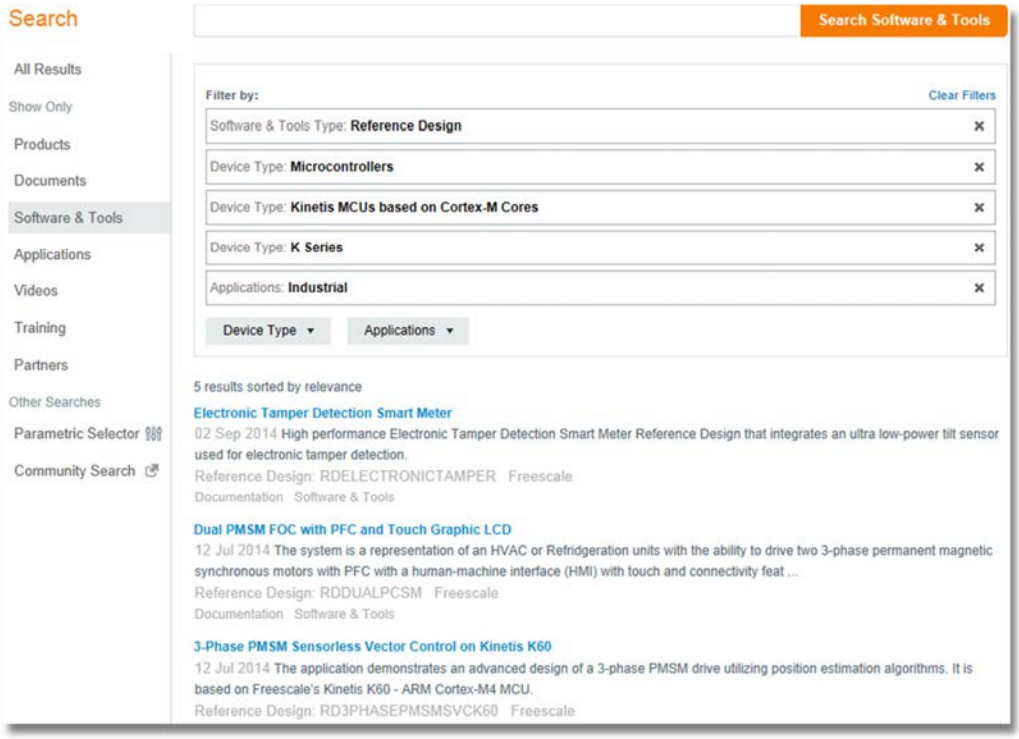


图 33. 参考设计的搜索结果

3.7 飞思卡尔单片机的生态系统（Ecosystem）

飞思卡尔积极与嵌入式企业合作，为方便 MCU 的开发与应用，建立了工具链、操作系统、中间件等不同合作联盟，此外在汽车、通信、工业与消费电子等领域企业合作，为客户提供了强大完善的生态体系。

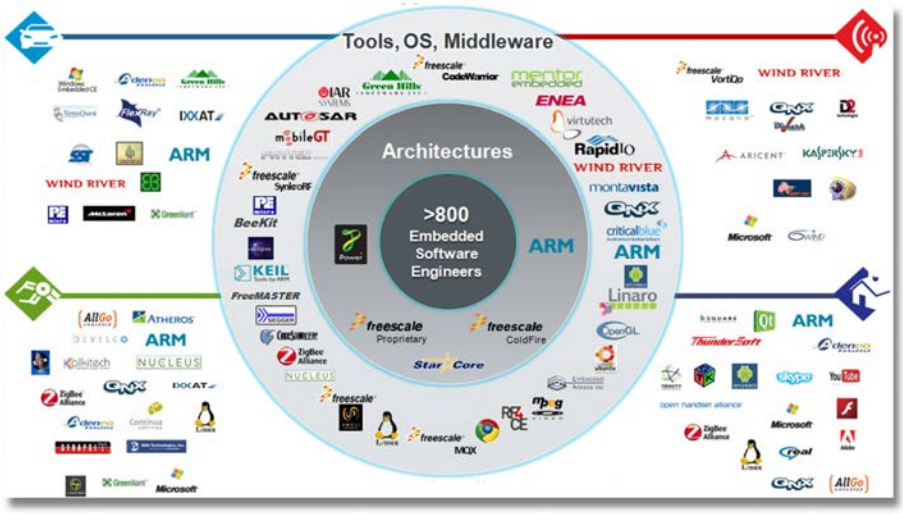


图 34. 飞思卡尔单片机的生态系统

上述不同企业根据其擅长领域的不同，为飞思卡尔提供了诸多支持，有的提供了基于 CAN 和 TCP/IP 的协议栈，有的提供了适用于物联网的 WiFi 技术，还有的提供了车载信息娱乐系统的解决方案。这些技术领先的企业对于飞思卡尔产品广泛深入的支持使得这个生态系统愈发丰富和强大，也使得用户的产品开发更加得心应手。

用户可以在飞思卡尔网站上查询到相关合作伙伴和生态合作系统的信息。如图 35 所示，点击“支持服务与网络社区”中的“Freescale Connect 合作伙伴计划”，就出现了搜索页面。

图 36. 搜索飞思卡尔和合作伙伴和生态系统

第四章 如何阅读飞思卡尔的技术文档

4.1 概述

在嵌入式系统的研发过程中，技术文档的阅读是一个基础。对一款芯片或者一个开发平台从入门到熟悉再到深入的研究，都离不开对其相关文档的查阅。而如今不同的半导体厂家针对其自家的芯片或者开发平台都有其自己一套风格的文档体系结构，这就给一些客户的平台移植或者新手的学习入门都带来了一定程度的障碍和门槛。

本章节以飞思卡尔的 Kinetis K22 系列 MCU 为例，详细介绍了飞思卡尔公司提供的包括 Datasheet、Reference Manual、User Guide、Application Note 以及一些其他相关技术文档的组织结构和阅读方法，提高客户阅读文档的效率，从而进一步帮助缩短客户的研发周期。

4.2 数据手册 Datasheet

与一些其他的半导体厂商不同，飞思卡尔提供的数据手册和参考手册是分开的，即两个独立的技术文档，其中数据手册（Datasheet）可以查阅到芯片相关的电气参数和封装尺寸等信息，而参考手册（Reference Manual）则主要介绍芯片具体的架构、技术细节和寄存器配置等内容，这种结构有效地避免了单一文档的臃肿庞杂，有助于客户清晰地进行芯片选型和电气性能评估。本节主要介绍数据手册（Datasheet）的结构和阅读方法，参考手册（Reference Manual）的内容将在下一小节再做介绍。

4.2.1 Datasheet 的命名规则

Datasheet，即芯片的数据手册，也跟芯片一样有其相应的命名规则。飞思卡尔根据芯片的家族系列、最大管脚数和主频高低对 Datasheet 进行了细分和命名，例如对型号为 MK22FN512VLH12 的芯片，与其对应的 Datasheet 的最新文档为 K22P121M120SF7，其中 K22 为家族系列，P121 为最大 121 管脚（涵盖 64 Pin、100pin），M120 为 120M 主频，SF 为 Sub-Family 的缩写，数字 7 则为文档的版本号（注意：最新的版本号命名规则相较之前版本号做了些改变，加入了根据 Flash 大小的分类）。

4.2.2 Datasheet 的文档结构介绍

Datasheet 的内容包括 Ratings、General、Peripheral Operating Requirements and Behaviors、Dimensions、Pinout 和 Part Identification 等几个部分。其中：

Ratings：即额定值参数，介绍了包括温湿度额定值和芯片的工作电压电流等额定值参数。其中特别需要注意的是，针对本文档芯片 MK22FN512VLH12 的最高焊接温度建议不要超过 260°C，客户在焊接操作和工艺设计的时候需要参考此部分内容。

General：即芯片的通用参数，既介绍了芯片的电压电流工作范围、低功耗模式下的唤醒时间和功耗参数、工作模式下的功耗/频率比和 EMC 特性等非开关量参数，也介绍了芯片内部时钟范围和一些外部接口的时序约束等开关量参数。

Peripheral Operating Requirements and Behaviors：即芯片的外设模块的工作参数，是我们做软硬件开发时最常用的部分，该部分给出了内核模块、时钟模块、内存接口、模拟外设模块和通信接口的工作参数，其中常用到的包括 MCG 和 Oscillator 时钟模块的时钟约束范围（用于倍频或者分频的配置和外部晶振的选择等）、Flash 模块烧写的时间、EzPort/FlexBus 的时序要求、ADC 模块的采样率和有效精度、Comparator 模块的带宽、DAC 的有效精度和 USB/SPI/I2C/UART/I2S 等通信接口模块的工作参数，可以帮助客户最大程度地有效利用芯片内部的功能模块，并做相应的功能实现的预评估和测试。

Dimensions：芯片的封装尺寸，客户在 PCB layout 的过程中如果需要手工画出该芯片 PCB 封装的话，可以根据该部分提供的对应封装的文档序号到飞思卡尔官网搜索并下载，再参照该文档给出的封装尺寸手工画出相应芯片的封装。以 MK22FN512VLH12 为例，其封装为 LQFP-64，Datasheet 给出的对应封装的文档号为 98ASS23234W。

Pinout：管脚分配表，K22F 内部提供了丰富的外设资源，由于管脚数量的限制，其外部管脚一般需要复用成多个外设功能，除了电源管脚、晶振管脚和个别的模拟输入管脚。该管脚复用分配表可以结合 Reference Manual 中的管脚复用模块寄存器配置，根据客户的实际功能需求做相应的复用功能配置。

Part Identification：芯片具体型号的命名规则，以 MK22FN512VLH12 为例，M 表示量产型号，K22 表示 Kinetis 家族 K22 系列，F 表示带浮点运算单元 FPU，N 表示只有 Program Flash 没有 FlexMemory，512 表示 Flash 空间大小为 512 字节，V 表示工业扩展级温度-40~105°C，LH 表示芯片封装为 LQFP-64，12 则表示芯片的主频为 120MHz。

4.3 参考手册 Reference Manual

Reference Manual，即芯片的参考手册，详细介绍了芯片的内核结构、内存映射、时钟分配、电源管理、安全加密、烧写调试、管脚复用以及所有外设模块等的技术细节，是客户开发软件必须了解也是最常需要查阅的技术手册。但是动辄上千页的参考手册，包容了如此多的技术信息，要从里面找到自己想要获取的相关资料对一个新手来说实属不易，因此有必要理清整个参考手册的结构，有针对性的攫取相关信息，掌握阅读 FSL 参考手册的方法。本章节以系统时钟配置、管脚复用和中断管理这三个难点也是重点的部分为例介绍飞思卡尔 Reference Manual 的阅读方法。

4.3.1 “万金油”之第三章 Chip Configuration

在介绍上述三部分的配置之前，需要重点提一下飞思卡尔参考手册非常重要的一章，即第三章 Chip Configuration，这也是飞思卡尔文档的特色之一。几乎其所有的 Reference Manual 中，第三章的地位一直很稳固；作为一个统领全篇的章节，其内容概述了芯片各个模块配置所需要参考的信息和功能架构并给出了一些模块更详细的配置信息和注意事项，所以我们在进行软件编程对某个外设进行配置的时候除了查看该外设相应的章节之外，第三章也是最常需要参考的一章。这里仍然以 MK22FN512VLH12 为例列出第三章中我们常需要参考的一些模块的配置信息：

（1）ARM Cortex-M 内核的 System Tick Timer 的时钟源；

（2）系统中断优先级的配置和中断向量表的分配；

（3）低功耗异步唤醒管理模块（AWIC）的唤醒源；

（4）极低功耗唤醒单元（LLWU）的唤醒源；

（5）DMA 模块的多路请求信号源；

（6）系统内存 Flash 和 SRAM 空间分配和相关注意事项；

（7）FlexBus 信号线的复用管理；

（8）ADC 模块采样通道号及硬件触发信号的分配；

（9）内部模拟比较器输入通道的分配；

（10）Flex Timer 的通道数、硬件触发机制和输入捕捉等参数配置；

（11）USB 模块的工作机制；

（12）SPI 模块发送和接收 FIFO 深度；

（13）UART 模块的中断源分配；

（14）GPIO 高驱动能力管脚的分配。

以对 ADC 模块进行软件配置为例，在 ADC 模块的章节中，ADCx_SC1n 寄存器中 ADCH 位即定义了 ADC 模块对模拟输入采样的通道号，而具体涉及到相应通道号的映射分配则可以在第三章的 ADC Configuration 中找到，如下图 1 所示：

3.7.1.3.1.1 ADC0 Channel Assignment for 64-Pin Package

ADC Channel (SC1n[ADCH])	Channel	Input signal (SC1n[DIFF]= 1)	Input signal (SC1n[DIFF]= 0)
00000	DAD0	ADC0_DP0 and ADC0_DM0[1]	ADC0_DP0[2]
00001	DAD1	Reserved	Reserved
00010	DAD2	Reserved	Reserved
00011	DAD3	ADC0_DP3 and ADC0_DM3[3]	ADC0_DP3[4]
00100[5]	AD4a	Reserved	Reserved
00101[5]	AD5a	Reserved	Reserved
00110[5]	AD6a	Reserved	Reserved
00111[5]	AD7a	Reserved	Reserved
00100[5]	AD4b	Reserved	ADC0_SE4b
00101[5]	AD5b	Reserved	ADC0_SE5b
00110[5]	AD6b	Reserved	ADC0_SE6b

图 1. ADC 通道分配图

4.3.2 系统时钟配置

系统时钟的配置是入门一款芯片平台的敲门砖，不同于以前的 8 位机和 16 位机简单的时钟管理，ARM 平台提供了非常强大而丰富的时钟管理机制，满足其内部各个功能模块的正常运行和低功耗机制的实现，但由此带来的是复杂且繁琐的时钟配置，这让很多新手望而却步只能用拿来主义的方法去使用封装好的函数库，但是出现问题或者根据实际需要进行微调配置的时候就会常常束手无策或者拆西墙补东墙，因此通过参考手册深入地了解 ARM 平台的时钟管理机制之后，用户才能熟练地对系统的时钟进行配置。

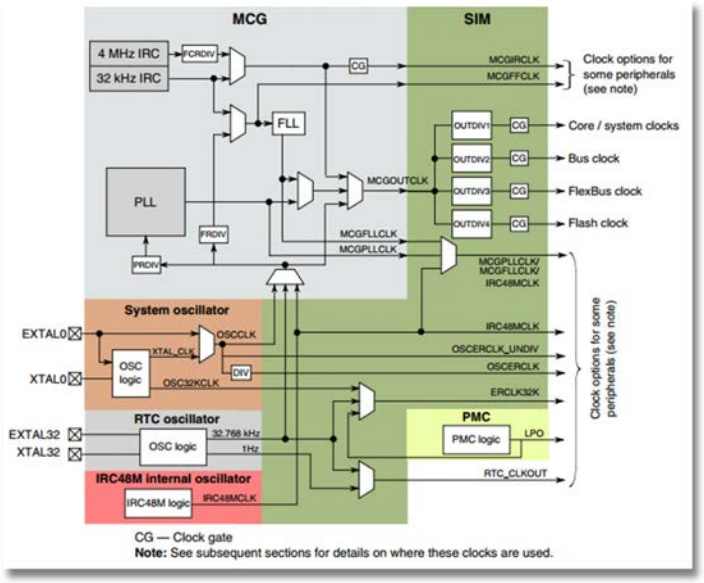


图 2. 系统时钟总体框图

通常的建议是以图文结合的形式去了解和熟悉系统的时钟管理机制。如图 2 所示，将 MK22FN512VLH12 参考手册中“第五章 Clock Distribution”的系统时钟总体框图，拆分开来，就可以清晰的了解其平台的时钟管理机制主要是由 MCG 模块、SIM 模块、System Oscillator 模块，RTC 模块、IRC48M internal oscillator 和 PMC 这六个子模块构成。

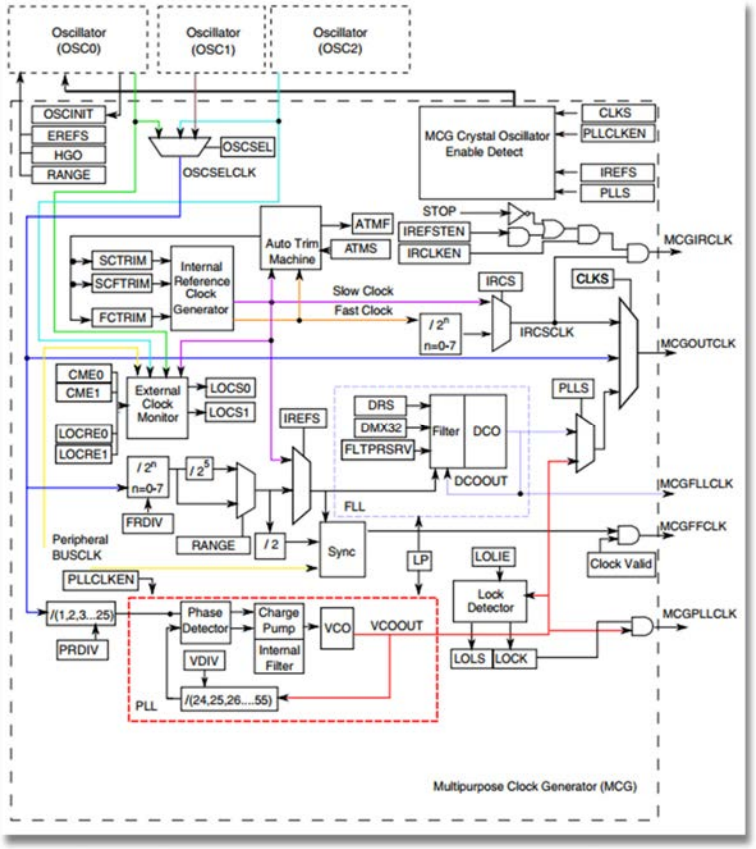


图 3. MCG 模块内部框图

从总体到局部的角度来看，在对总体结构有了全局的认识之后，则可以通过查看上图 1 对应的六个子模块来深入了解其寄存器级的配置实现，以 MCG 子模块为例，可以在与其对应的章节里找到该模块的结构框图如图 3 所示，图中给出了与 System Oscillator 和 SIM 这两个子模块的接口，对 MCG 内部的寄存器级的实现也给出了详细的比特位的配置，而且最重要的是还清晰地描绘出了时钟 Clock 从输入到输出的路径配置，帮助用户理清整个系统时钟管理的结构。

4.3.3 管脚复用

ARM 平台丰富的外设资源与有限的管脚封装之间的矛盾造成了其管脚复用的必然性（除了个别敏感的管脚外），而不同于时钟管理部分的复杂，飞思卡尔对其芯片管脚复用的配置却非常简单。对芯片管脚复用的配置需要结合 Datasheet 中“Pinout”章节的管脚分配图如下图 4，而所需要配置的寄存器可以在“Port Control and Interrupt”这一章中找到，即 PORTx_PCRn（其中 x 表示 A、B、C、D、E、F 等管脚组，而 n 则表示 0~31 之间的管脚号），如下图 5 所示，通过这种 PORT 分类的方式来映射芯片所有管脚。该寄存器中 3 位 MUX 即表示复用功能的选择如下图 6，最多 8 个复用的选项一般足够了。当然，根据所选择的复用外设的需要也可以通过该寄存器使能上拉下拉电阻功能，还有相应的 I/O 外部中断也需要在此寄存器里做相应配置。

121 BGA	100 LQFP	64 LQFP	64 MAP BGA	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	EzPort
E4	1	1	A1	PTE0/ CLKOUT32K	ADC1_ SE4a	ADC1_ SE4a	PTE0/ CLKOUT32K	SPI1_PCS1	UART1_TX			I2C1_SDA	RTC_ CLKOUT	
E3	2	2	B1	PTE1/ LLWU_P0	ADC1_ SE5a	ADC1_ SE5a	PTE1/ LLWU_P0	SPI1_ SOUT	UART1_RX			I2C1_SCL	SPI1_SIN	
E2	3	—	—	PTE2/ LLWU_P1	ADC1_ SE6a	ADC1_ SE6a	PTE2/ LLWU_P1	SPI1_SCK	UART1_ CTS_b					

图 4 Pinout 管脚复用分配图

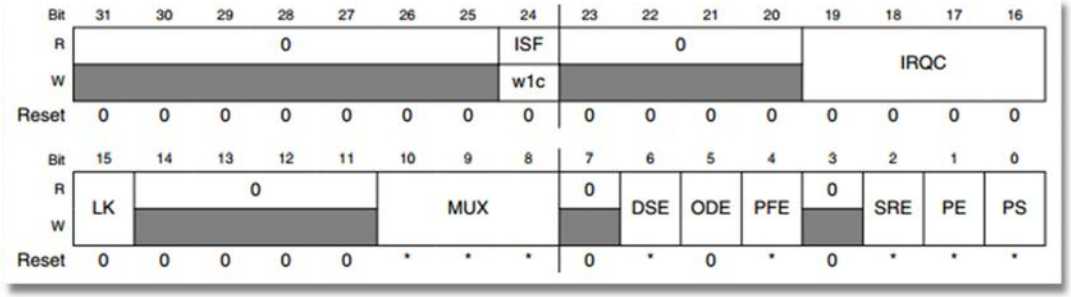


图 5 PORTx_PCRn 寄存器映射

10–8 MUX	Pin Mux Control Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved and may result in configuring the pin for a different pin muxing slot. The corresponding pin is configured in the following pin muxing slot as follows: 000 Pin disabled (analog). 001 Alternative 1 (GPIO). 010 Alternative 2 (chip-specific). 011 Alternative 3 (chip-specific). 100 Alternative 4 (chip-specific). 101 Alternative 5 (chip-specific). 110 Alternative 6 (chip-specific). 111 Alternative 7 (chip-specific).

图 6 管脚复用配置位

结合上述三图，以配置 64 pin LQFP 封装的芯片 MK22FN512VLH12 的 PTE0 和 PTE1 这两个管脚为 UART 功能为例，它们默认的配置为 ADC 的模拟输入端口功能，通过图 4 管脚复用分配图找到 UART1 的复用索引号为 ALT3 选项，则只需要将 PORTE_PCR0 和 PORTE_PCR1 这两个寄存器中 MUX 位写入 3 即可实现 UART1_TX 和 UART1_RX 端口的配置，操作代码如下：

```
PORTE_PCR0 |= 3 << 8;
PORTE_PCR1 |= 3 << 8;
```

4.3.4 中断管理

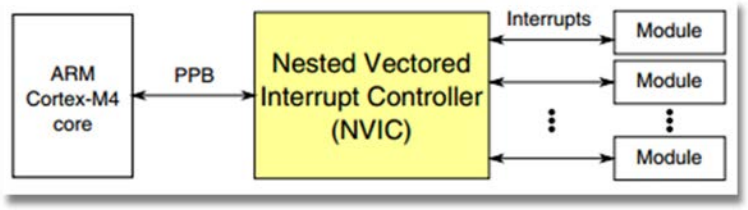


图 7 NVIC 模块框图

ARM 平台的中断管理引入了 NVIC 即嵌套向量中断控制器这个模块，其负责接管芯片运行状态下所有的中断源，是 ARM 内核不可分离的一部分，它与内核的逻辑紧密耦合完成相应的中断处理工作。而且由于其为内核的一部分，因此关于 NVIC 相关寄存器的配置需要结合 ARM 公司的架构技术文档，具体见 4.5.3 节。不过，NVIC 对中断的管理是建立在中断向量表上的，这张表的数量是固定的，但是除了前 16 个内核中断之外，剩余的 IRQ 中断不同的半导体厂家可以根据自己的需要来定制映射不同的外设中断，也就是说各大半导体厂家定义的这个中断向量表可能不一致，飞思卡尔 ARM 平台的中断向量表映射可以在其 Reference Manual 的“第三章 Chip Configuration”中 Core Modules 的 Nested Vectored Interrupt Controller (NVIC) Configuration 小节中找到，用户可以根据这张表的索引号来使能或者禁用相应的 IRQ 中断。

4.4 用户指南 User Guide

为了方便新用户快速入门和熟悉飞思卡尔的产品、开发工具和开发环境，飞思卡尔针对其相关产品提供了相应的用户指导手册，包括部分外设资源模块的编程指导手册和相应的开发板工具使用手册。

4.4.1 Peripheral Quick Reference User Guide—外设模块快速参考用户指南

外设模块用户参考手册详细介绍了芯片常用模块的技术特性，并针对该模块资源的部分功能给出了相应的代码例程作为编程参考，可以加快用户上手时间，提高用户开发的效率，在软件开发时可以结合该用户手册和芯片的 Reference Manual 文档中相应外设模块的章节进行外设资源的配置。飞思卡尔为 ARM 平台 Kinetis 家族的部分系列提供了相应的外设用户参考手册，如下表 1 所示：

表 1 Kinetis 家族外设模块用户参考手册

MCU 系列	文档名称及链接
Kinetis K	KQRUG
Kinetis L	KLQRUG
Kinetis V	KVQRUG

4.4.2 Freedom & TWR Tool User Guide－评估板用户指南

为方便用户评估芯片的性能和熟悉飞思卡尔产品的开发流程，飞思卡尔提供了两套评估板供用户评估和开发测试，其中 Freedom 板为廉价的评估板，自带板载调试器，用户可以直接通过自己的 PC 机快速搭建开发环境并进行测试和评估芯片简单的外设资源，另外塔式系统 Tower 板为可扩展性比较强的评估板，板载资源比较丰富且通过构建塔式系统可以灵活的添加和裁剪外设资源，用户可以进行更高级的功能测试。飞思卡尔为这两套评估板均提供了相应的用户指导手册帮助用户快速入门飞思卡尔产品的开发，以 MK22F 系列为例，Freedom 板的用户手册为 FRDMK22FUG，TWR 板的为 TWRK22F120MUG，它们均可以直接在飞思卡尔官网搜索下载。

4.5 其它的技术文档

除了上述 Datasheet、Reference Manual 和 User Guide 之外，飞思卡尔还提供了其他的丰富的技术文档供用户参考，本章节重点介绍一下用户还经常参考的应用笔记、Errata 勘误表和内核架构的参考文档。

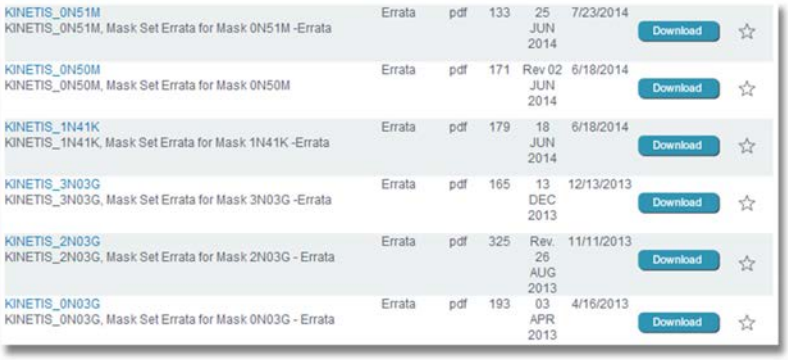
4.5.1 应用笔记 Application Notes

Application Note，即应用笔记，是飞思卡尔内部专家工程师针对飞思卡尔产品的具体应用所撰写的心得笔记。飞思卡尔官网系统提供了大量的中英文应用笔记供客户工程师参考和学习，一些应用笔记也会附上相应的代码例程工客户下载。飞思卡尔应用笔记均以其英文大写字母“AN”开头，后面数字为文档的序列号，用户可以直接在飞思卡尔官网搜索相应的文档序号或者具体应用的关键字进行下载，此外针对某款芯片相关的应用笔记则可以直接在该芯片的产品页中 Document 一栏中找到，如下图所示：

图 8 应用笔记列表

4.5.2 勘误表 Mask Set Errata

芯片的 Errata 勘误表是客户在使用相关芯片进行项目开发时必须要查阅的参考文档，在一款芯片从发布到量产再到最后停产的整个生命周期里，半导体厂商会非常负责任地针对芯片在应用过程中出现的一些功能的局限性或者某些参数的不准确性等问题不定时的发布相应的勘误表，帮助用户在项目开发过程中规避相应的问题和进行风险把控。此外，需要用户注意的是，Errata 勘误表是以 Mask Set 即芯片的掩膜硅版本号进行分类的(芯片 Package 封装上打的标号，比如 2N03G)，所以用户可以根据所使用芯片的版本号在飞思卡尔官网查询相应的 Errata 勘误表文件，同样用户也可以直接到相关芯片的产品页中 Document 一栏中找到该芯片系列所有 Mask Set 的 Errata 勘误表，以 K22F 120M 产品为例，其所有子系列的 Errata 勘误表如下图 9 所示：



KINETIS_0N51M KINETIS_0N51M, Mask Set Errata for Mask 0N51M -Errata	Errata	pdf	133	25 JUN 2014	7/23/2014	Download	☆
KINETIS_0N50M KINETIS_0N50M, Mask Set Errata for Mask 0N50M	Errata	pdf	171	Rev 02 JUN 2014	6/18/2014	Download	☆
KINETIS_1N41K KINETIS_1N41K, Mask Set Errata for Mask 1N41K -Errata	Errata	pdf	179	18 JUN 2014	6/18/2014	Download	☆
KINETIS_3N03G KINETIS_3N03G, Mask Set Errata for Mask 3N03G -Errata	Errata	pdf	165	13 DEC 2013	12/13/2013	Download	☆
KINETIS_2N03G KINETIS_2N03G, Mask Set Errata for Mask 2N03G - Errata	Errata	pdf	325	Rev. 26 AUG 2013	11/11/2013	Download	☆
KINETIS_0N03G KINETIS_0N03G, Mask Set Errata for Mask 0N03G - Errata	Errata	pdf	193	03 APR 2013	4/16/2013	Download	☆

图 9 K22F_120M 勘误表

4.5.3 内核架构参考手册 Architecture Reference Manual

由于飞思卡尔 Kinetis 家族系列的平台是基于 ARM Cortex-M 内核的产品，其内核的知识产权为 ARM 公司所有，所以关于该平台内核架构部分的包括指令集、中断管理和调试组件等详细的技术信息则需要到 ARM 公司官网下载相应内核架构的技术文档作为参考，由于飞思卡尔目前的 Kinetis 产品系列主要为 ARM Cortex-M0+和 Cortex-M4 内核，其在 ARM 官网相应的内核参考文档可以直接搜索得到，其相应的文档链接如下：

（1） ARM Cortex-M4 内核技术参考手册

（http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439d/DDI0439D_cortex_m4_processor_r0p1_trm.pdf）

（2） ARM Cortex-M0+内核技术参考手册

（http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484c/DDI0484C_cortex_m0p_r0p1_trm.pdf）

此外，关于飞思卡尔的自有内核架构包括 DSP56800E/EX 和 Coldfire 等相关内核架构的文档则可以直接在飞思卡尔官网下载。

（1）DSP56800E/DSP56800EX 内核技术参考手册

（http://cache.freescale.com/files/dsp/doc/ref_manual/DSP56800ERM.pdf）

第五章 飞思卡尔单片机硬件设计指南

5.1 概述

在进行嵌入式系统设计时，硬件电路设计的好坏不仅关系到整个系统的功能实现和可靠性，还会对系统软件的复杂程度产生影响。本章节以 Kinetis K22 芯片为对象，介绍采用 Kinetis MCU 进行最小系统硬件设计时，需要了解的一些硬件设计注意事项和设计原则。

在本章节中，除了会介绍电源电路、时钟电路、复位电路、调试电路等 MCU 常见外围硬件设计外，还会针对使用 Kinetis 系列芯片时需要了解的设计做一些探讨，如 NMI 引脚配置、High current driver 引脚分布、IO 输出方式（Open-drain/Pull-Push）支持、RTC、USB、ADC 电路设计和参考电压的选择以及板级 EMC 性能改善等。

5.2 电源电路的设计

5.2.1 K22 的电源引脚分配和功能描述

Kinetis K22 支持 1.71-3.6V 宽电压输入，为了提供稳定的电源，芯片使用多组电源引脚分别为内部电压调节器、IO 引脚驱动、AD 转换电路等供电，并且提供多处电源引出脚，便于用户外接滤波电容，改善系统的电磁兼容性。对于本文档中使用到的 K22 LQFP-64 pin 封装，其电源引脚名称、分布和功能描述如下表 1 所示。

表 1 K22 LQFP-64 pin 封装电源引脚名称、分布和功能描述

引脚名称		功能描述	典型值	引脚号(LQFP-64)
电源输入	VDDx/VSSx[1]	数字电源输入正/负	3.3V、0V	3/4、30/31、48/47
	VDDA/VSSA	AD 模块的电源输入正/负	3.3V、0V	13、16
	VREFH/VREFL	AD 模块参考电压输入高/低	3.3V、0V	14、15
	VBAT	RTC 模块的备用电源输入	3.3V	21
	VREGIN	USB 模块输入参考电压	5V	8
电源输出	VOUT33	USB 模块电源稳压器输出	3.3V	7
	VREF_OUT	内部参考电压输出	1.2V	17

[1]. 文中 VSSx 指的是芯片的地。

5.2.2 MCU 主电源供电引脚

VDDx/VSSx 是芯片的主电源供电引脚，Kinetis K22 LQFP 64 pin 封装的芯片上一共包含 3 对，在电路设计时需要在每对引脚外部分别放置至少一个去耦电容（0.1uF 的陶瓷电容）。并且旁路电容的放置必须尽量靠近 MCU 电源引脚，从而最大限度地缩小 VDD 和 VSS 引脚之间的电容所形成的环路。

在芯片内部，这三对电源的引脚是互相连通的。理论上说，只要将任意一个 VDDx 引脚和任意一个 VSSx 引脚连接到电源上，整个芯片就可以正常工作了。但在实际的设计中，我们建议将三组电源引脚都可靠地连接到电源上，这样对改善芯片内部的电源回路以及提高整个系统的 EMC 性能都更有好处。

5.2.3 模拟外设电源及参考电压引脚

VDDA/VSSA 是芯片内部 ADC、DAC 以及 CMP 等模拟外设的电源输入引脚，为使芯片的模拟外设有稳定的电源，从而得到更好的转换精度，通常需要在靠近 VDDA 引脚的地方并联两个外部稳压电容（10nF 陶瓷电容+1μF 钽电容）。另外，为限制电源中的高频噪声，在设计时，可以将 VDDA 通过电感、磁珠等阻塞元件与数字电源 VDDx 进行隔离。

另外，大部分 Kinetis MCU 还提供了一对模拟参考输入引脚 VREFH/VREFL，用户可以连接一个独立的外部电压基准作为模拟电压的参考（该基准的电压范围为 1.13V~ VDDA）。客户也可以使用片内产生的内部 ADC 参考源，一般为 1.2V。为保证能得到更好的精度，建议使用外部的 3V 电压作为参考。当然，用户也可以直接连接到外部电源引脚 VDDA 上，使用过程中，建议在 VREFH 引脚上连接一个 10nF 和一个 1μF 的电容。对于一些小封装的芯片，可能没有 VREFH 和 VREFL，实际上它们分别在内部被直接连接到 ADC 的供电电源 VDDA 和 VSSA。

5.2.4 RTC 后备电源引脚

VBAT 引脚是 RTC 模块的备用电源输入，用于在系统掉电和低功耗模式下为 RTC 和一个 32 字节寄存器供电，该引脚通常被连接到外部电池 (1.71V < VBAT < 3.6V)。如下图 1 所示，一个简单的电池隔离器由一个共阴极的双肖特基阵列组成，在主系统电源 VDD 关闭时，通过器件 D1 提供备用电池。建议在尽可能靠近 MCU 的地方放置一个 100 nF 旁路电容，以便最大限度地降低电源切换事件的影响。

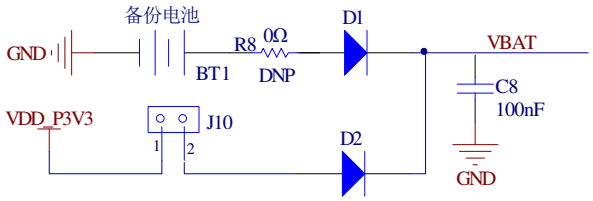


图 1. RTC 后备电源连接

5.2.5 USB 模块电源引脚

VREGIN 是芯片内部 USB 模块的供电引脚，为避免外部瞬态浪涌电压的影响，建议在此引脚的线路上串连一个内阻为 330 欧姆的磁珠，并在靠近该引脚处并联一个 10uF 电容，如果成本允许，考虑在此供电线路靠近 USB 物理接口侧并联一个 TVS 管，作为热插拔和外部浪涌电压的保护电路。

5.2.6 VREF_OUT 和 VOUT33 电压输出引脚

为方便用户使用，Kinetis K22 还有两个电压输出引脚，用户可以通过程序配置分别对外输出一个 1.2V 电压和一个 3.3V 电压。其中，VREF_OUT 引脚是芯片内部 1.2V 基准电压的输出引脚，该电压可以作为芯片内部 ADC，DAC 或者 CMP 作为电压基准，同时也可以作为外部电路的电压源。需要注意的一点是，无论是在内部或者外部使用该 1.2V 电压作为参考，都需要在 VREF_OUT 引脚连接一个 100nF 的负载电容。

VOUT33 引脚是芯片内部 3.3V 电压的输出引脚，该电压是由芯片内部 USB Voltage Regulator 产生，该电压除了可以用来提供给内部的 USB Transceiver 完成 USB 功能外，用户还可通过 VOUT3.3 引脚输出到外部，提供给板上的其它芯片作为供电电源，其最大输出电流为 120mA。并且通常建议在靠近该引脚外部加一个 2.2uF 电容。

5.2.7 电源电路的 PCB 设计建议

电源系统的布线是整个硬件版图最重要的部分，其基本思想是确保 MCU 和其他数字器件通过低阻抗路径接至电源，并且有一些常规可以遵循：

1) 分割不同的电源域，将交流电源与直流电源分开，数字部分与模拟部分分开。实际电路设计中可以选择使用物理隔离和去耦滤波器，对于后者的使用如下图 2 所示，串联隔离元件通常选择小电感或阻抗较小（100 MHz 时约 100Ω）的铁氧体磁珠。去耦电容一般选择 10nF 到 1uF，具有高频成分的一端应选择较低的值。

图 2. 数字和模拟电源隔离示意图

2) 在 MCU 的每个 VDD 电源引脚旁边放置去耦电容，画板时一定要尽量使去耦电容靠近 MCU 电源引脚放置，并尽量缩小 VDD 和 VSS 引脚之间的电容所形成的环路。
3) 合理使用电源层和接地层，接地布线应优先于任何其他布线，电源走线尽量使用较宽走线和较少的层变换。在系统允许的情况下，建议在 MCU 封装的正下方使用一个接地层，以便降低 RF 辐射并提高瞬变抑制性能，提高板级的 EMC 性能（更多关于电源硬件设计和 EMC 的知识请参看 AN2764）。

5.3 时钟电路

5.3.1 K22 的时钟源

为满足用户不同的应用，飞思卡尔 Kinetis MCU 提供多个时钟源选择，如图 3 所示是 K22 的系统时钟框图，它包括多种时钟源，可以通过 MCG 模块灵活地配置使用。其中内部时钟源的优点是成本低，不易受外部影响，而外部时钟源的优点在于能产生非常精确的主时钟。

1) 内部时钟源：
   - 在芯片内部集成了一个 4M Hz 快速时钟和一个 32K 慢速时钟；
   - 一个 1K Hz 固定输出的 LPO，可以工作在任何的低功耗模式下；
   - 内部 IRC48M，主要用于 USB 等对时钟精度要求比较高的应用；
2) 外部时钟源：
   - 在 EXTAL0 和 XTAL0 之间可以外接振荡器或者在 EXTAL0 引脚直接连接时钟源，频率范围为从 32.768Khz 到 32Mhz；
   - 在 EXTAL32 和 XTAL32 之间可以外接 32768Hz 无源振荡器或者有源时钟作为 RTC 或者 MCU 系统的时钟源；

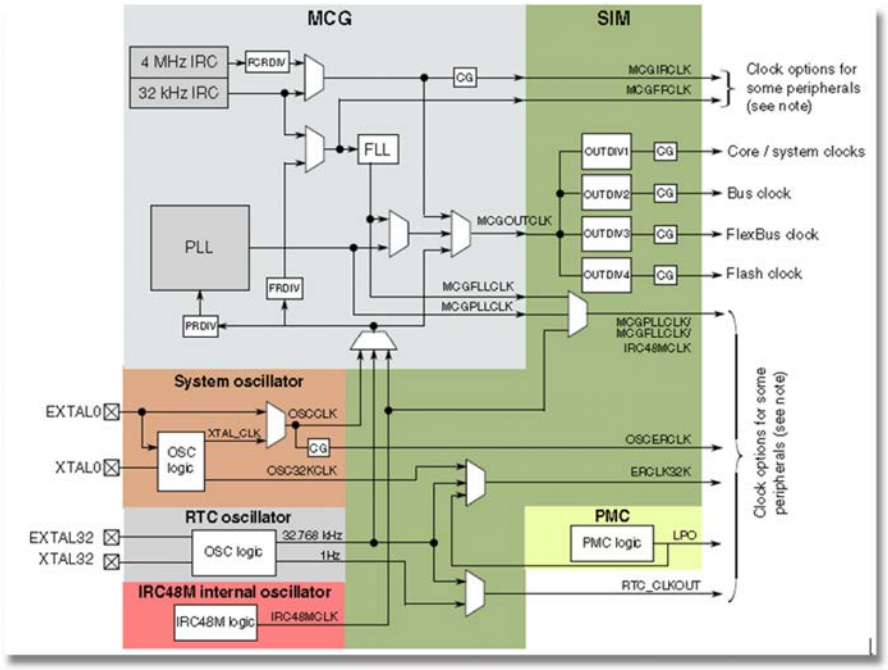


图 3. K22 的系统时钟框图

5.3.2 MCG 的外部时钟振荡器电路设计

常用的外部时钟振荡器有两种类型：机械谐振式，如石英晶体和陶瓷谐振器；被动式 RC（电阻-电容）振荡器。关于该振荡器的元件选用详见特定器件的 Datasheet 手册，此处不再赘述。本节主要针对使用外部晶体振荡器的时钟电路设计展开探讨。

5.3.2.1 外接晶振的电路设计

根据应用的不同，Kinetis 连接外部晶振时可以采用 3 种电路设计，如下表 1 所示。

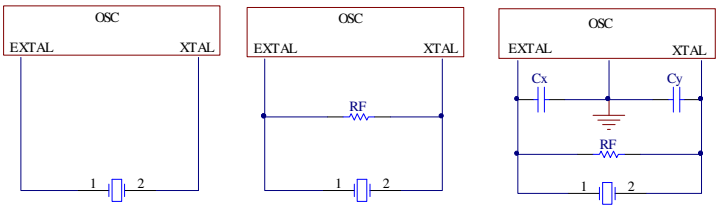
表 1. Kinetis 使用不同外部晶体外部电路方案选择

晶振和应用类型	建议电路
Low-frequency(32 kHz), Low Power	方案 1
Low-frequency(32 kHz), High Gain	方案 2/方案 3
High-frequency(3-32 MHz), Low Power	方案 1/方案 3
High-frequency(3-32 MHz), High Gain	方案 2/方案 3

具体的电路连接分别如下图 4 所示，可以看到，在方案 1 和方案 2 中不需要外部谐振电容，原因是在芯片的内部集成了谐振电容，用户可以通过 CR[SCxP]位来灵活配置内部谐振电容的大小，其最大值可以达到 30 pF。而当选用的晶振所需要的匹配电容大于 30pF 时，建议采用方案 3 的晶振连接，否则建议使用方案 1 和方案 2，从而简化电路设计。

需要指出的是，尽管方案 3 可以在多种模式下使用，但是在低功耗模式使用时，由于芯片振荡器内部默认是集成反馈电阻的，所以一定不要外部再加 R_F。对于 C_X 和 C_Y，负载电容的容值需要根据晶振的要求进行选择。

对于 RTC 外部 32K 晶振输入，由于芯片内部包含反馈电阻和谐振电容，只需似方案 1 在 EXTAL32 和 XTAL32 引脚外接 32K 晶振即可，无需外接电容和电阻，从而可大大简化电路设计。



方案 1　　方案 2　　方案 3

图 4. 使用外部晶体的电路连接方案

5.3.2.2　外接有源振荡器的电路设计

如果采用外部有源时钟，其电路设计就会更加的简单，如图 5 所示。对于 K22，其外部有源时钟的最大值为 50MHz。

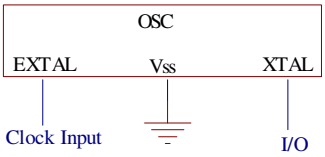


图 5. 使用外部有源振荡器的时钟电路设计

5.3.2.3　晶振电路 PCB 设计建议

Kinetis MCU 的引脚布局考虑到 PCB 布板的需要，通常将 EXTAL 和 XTAL 引脚置于 BGA 封装的外部焊盘环上或 QFP 封装的拐角上，从而为在顶层上靠近 MCU 处放置晶体或谐振器的布线提供空间。通常在时钟引脚的旁边还会有地（VSS）引脚，方便用户的接地布线。以下是一些通用的规则：

1) 晶体和负载电容需要尽可能近地靠近 MCU 的引脚，以减小输出失真和启动稳定时间；
2) 晶体正下方的层上不得有任何种类的信号布线；
3) 合理选用偏置电阻和负载电容，涉及到 EMC 易感性的系统中，应该选用可以使振荡器输入引脚上信号的振幅比较大的那种振荡器配置，但带来后果可能是功耗会比较大；
4) 晶体及其负载元件周围应放置一个防护环，防止安装层上的相邻信号发生串扰。此防护环可以从晶体引脚相邻的 VSS 引脚起始，如下图 6 所示分别是推荐的 BGA 封装和 LQFP 封装的晶体 Layout 布局；

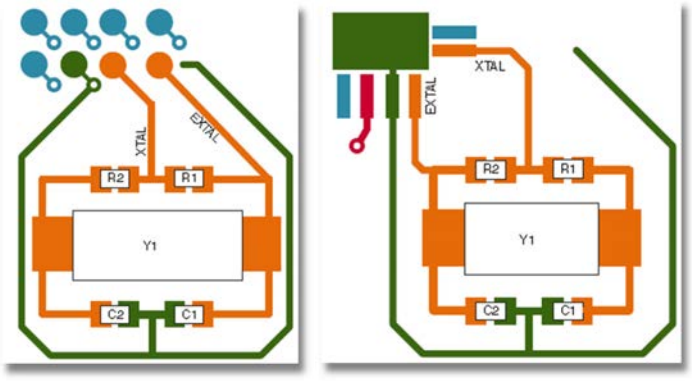


图 6. BGA 和 LQFP 封装建议的晶体 Layout 布局

5.4 复位电路设计

在基于 MCU 的系统中，除振荡器输入之外的最敏感的输入信号就是复位和中断请求输入。RESET_b 引脚是芯片的复位引脚，该引脚默认是使能复位功能的（可通过软件配置为 GPIO 等其它功能），并且低电平有效。Kinetis MCU 的复位引脚是双向引脚，作为输入端，将其拉低可以使芯片复位；作为输出引脚，上电复位期间会有低脉冲输出，表示芯片复位完成。

建议的电路如下图 7 所示，通过上拉电阻把该引脚拉至高电平，同时靠近 MCU 放置一个 100 nF 电容，以实现瞬态保护避免误复位。值得一提的是，RESET_b 引脚还有一个可配置数字滤波器，用以在上电之后抑制该引脚上的潜在噪声（相应配置位位于 RCM_RPFC 寄存器）。如果使用该滤波器的话，上述的上拉电阻和电容可能变得不需要，但在高电气噪声环境中，仍然建议使用外部滤波。

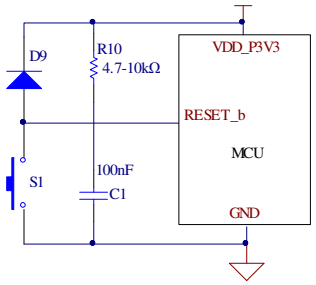


图 7. 典型复位电路设计

5.5 调试接口

Kinetis MCU 同时支持 JTAG 接口和 SWD 接口进行编程调试。除电源外，JTAG 接口需要用到至少 5 个 PIN 脚（JTAG_TCLK/ JTAG_TDI/ JTAG_TDO/ JTAG_TMS/ JTAG_RST），其电路连接如图 8 所示，而串行线调试接口(SWD)最少只需要 3 个 Pin 脚（SWD_CLK/SWD_DIO/SWD_RST），其电路连接如图 9 所示。两者相比，SWD 在高速模式下更稳定，使用引脚更少，所以在使用过程中建议使用 SWD 接口。

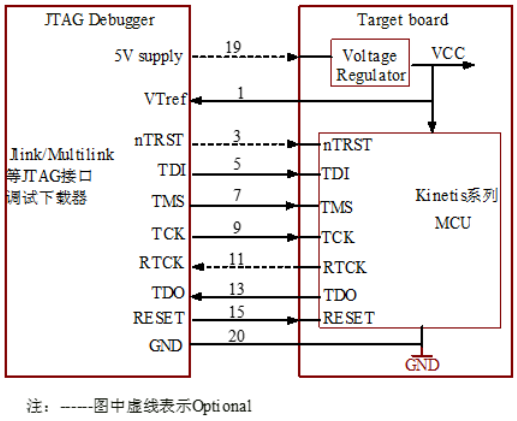


图 8. Kinetis MCU JTAG 接口连接示意图

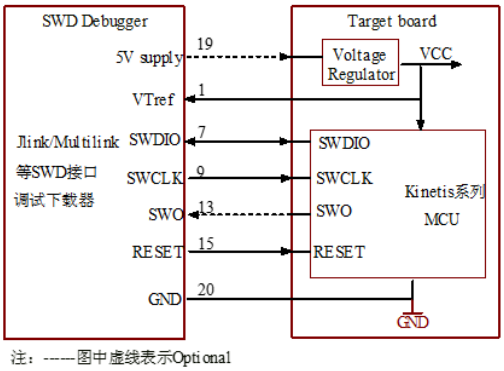


图 9. Kinetis MCU SWD 接口连接示意图

5.6 ADC 模拟输入

模拟采样是 MCU 的一个常用功能，Kinetis 大部分 MCU 系列提供 16 位的 SAR ADC，最多支持 4 个独立 ADC 模块以及 48 个通道（单端，在不考虑引脚复用的情况下），本文档中使用的 MK22FN512VLH12 包含两个独立 ADC 模块，支持 14 个单端通道和 2 个差分通道。

对模拟输入而言，前端滤波电路的设计非常重要，除了考虑模拟信号的截止频率，还需要考虑源阻抗和采样时间，尤其是对于高分辨率模数转换。常规思想是：快速采样时间与慢速采样时间相比，要求更小的电容值和输入阻抗。高分辨率输入与低分辨率输入相比，要求的电容值和输入阻抗可能更小。一般而言，并联电容值范围是 10 pF（高速转换）至 1μF（低速转换），串联电阻的范围是数百欧姆到 10 kΩ。除了合理设计外部滤波电路，模拟电路在 PCB 布线时还有很多需要注意的地方：1. AD 通道的布线要尽量短；2.将数字部分和模拟部分的电源分开，并分区覆铜，保证模拟地和数字地只在一个点结合，而且这个点要求远离干扰，有时会选用一个磁珠连接；3. 走线周围避免放置高噪声元器件，在模拟通道外围使用模拟地进行隔离；4.要想得到特别准确的采样结果，一般推荐在输入端加一些 Buffer/跟随运放。

5.7 USB 电路设计

Kinetis K22 内部 USB 模块的系统框图如下图 10 所示，它集成了一个支持 Dual-role USB OTG-capable (On-The-Go) 的控制器模块、一个 Transceiver、一个 3.3 V regulator 以及一个 USB device charger 检测模块，支持 USB 2.0 规格的 full-speed (FS) Device 模式和 FS/LS host 模式。

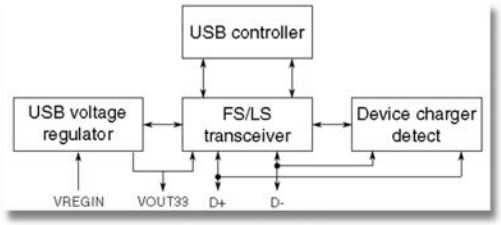


图 10. K22 内部 USB 模块系统框图

Kinetis USB 模块的完善大大简化了 USB 的接口电路设计，但是用户还是需要根据使用模式的不同进行一些外部电路的设计。如下图 11 是 USB 在 Device 和 Host 模式下的推荐电路，其中 Option 1 和 Option 2 是 MCU 工作在 Host 模式下，实现对外部 USB Device 提供 5V 电压的电路图。Option 1 结构简单，成本低，Option 2 方便程序控制 MCU 在 Host 和 Device 模式进行切换，

可靠性更高。而当 MCU 只工作在 Device 模式下，这两个 Option 可以省略。

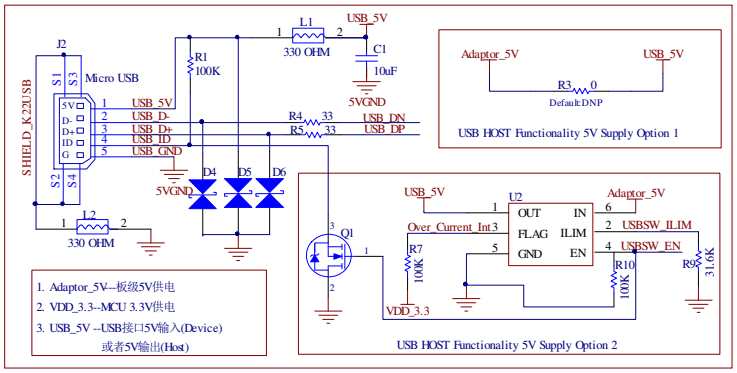


图 11. USB Device and Host 模式下电路设计

5.8 板级 EMC 性能改善与 PCB 布线的注意事项

EMC 性能是作为产品认证的一个重要指标，和板级硬件电路的设计息息相关，EMC 性能取决于许多因素，通常需要考虑的因素包括：连接器、机械组件、高速信号、低速信号、开关和电源域等。

连接器和某些机械组件（开关、继电器等）的放置对最终产品的成型至关重要，所以在开始 PCB 布线之前，必须注意妥善安放各元件并合理划分不同的电源域，如下图 8 所示。必须将低电平模拟信号电路、高速数字信号电路以及噪声电路 （继电器、大电流开关等）分开放置，以将 PCB 子系统间的耦合将到最低。另外，在每个功能区域内分别添加退耦滤波器(DF)、高频率旁路电容(BP)以及提供低通滤波器(LPF)，从而增强系统的整体 EMC 性能。

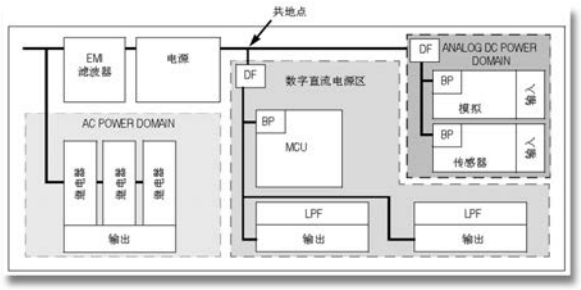


图 12. 板级平面结构布局

用户可在飞思卡尔网站上搜索应用笔记“AN2321：电路板级的电磁兼容设计”作为参考。

5.9 Kinetis MCU 的封装类型和一些特殊引脚说明

5.9.1 封装类型

Kinetis MCU 提供多种不同的封装类型，从 20 脚 1.6x2.0 mm 的 CSP 封装的 KL03 到 256 脚 BGA 封装的 K70，最大程度上满足客户在不同应用场合的需求。关于芯片的封装尺寸信息可以在官网查询，此处以 K22F120M 64-pin 封装为例讲述一下封装尺寸信息的查找方法，步骤如下面的三个截图所示，在该网页中除了可以找到该芯片的封装信息，还可以找到该芯片的环境合规信息、订购信息、工作特性、供货状态等。

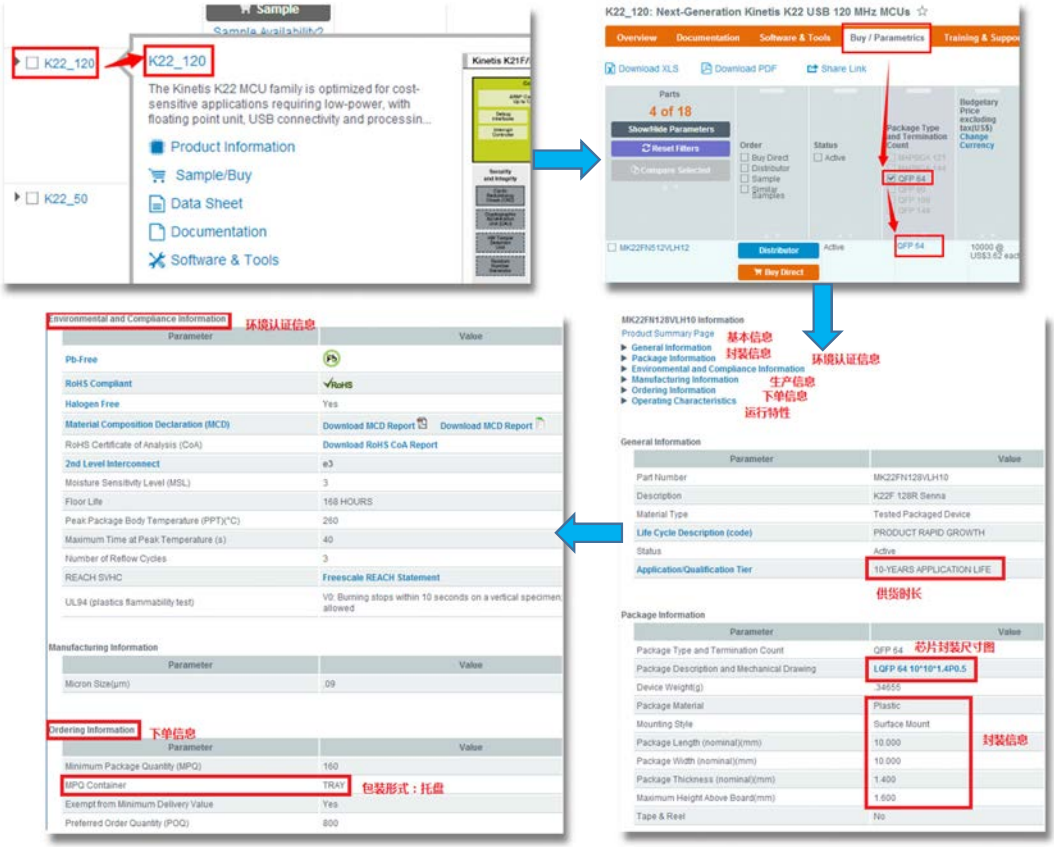


图 13. 封装尺寸信息查找方法截图

5.9.2 NMI 引脚

NMI 引脚是芯片的不可屏蔽中断引脚，芯片默认是使能 NMI 功能的（低电平有效），该引脚内部具有较小的内部上拉电阻，但建议采用外部 4.7 kΩ 至 10 kΩ 上拉电阻。需要特别提出的是，如果该引脚被外部电路拉低，即便该引脚被复用成其它功能(FTM 或者 GPIO)，在下载成程序时，也有可能导致芯片与仿真器连接异常。原因是芯片一上电，在程序还没有执行禁用 NMI 功能时，由于 NMI 引脚被外部拉低，程序就会已经进入 NMI 的中断程序，从而造成程序无法成功下载。此时则需要通过 Flash NVM 配置字节禁用 NMI 功能，它与在程序中禁用 NMI 引脚的区别在于，这种方法会执行用户程序之前生效。

5.9.3 大电流驱动（High Current Driver）引脚

Kinetis E 系列 MCU 部分引脚支持大电流驱动功能，可以直接驱动 LED，每个引脚的最大输出电流能够达到 20mA，总的最大输出电流是 100mA。

5.9.4 LLWU 唤醒引脚

在 Kinetis MCU 低功耗应用中，除了支持芯片内部模块唤醒之外，还支持 GPIO 引脚唤醒，但需要指出的是并不是所有的引脚都支持，所以在电路设计时需要特别注意，具体是哪些引脚支持 LLWU 引脚唤醒功能，可以参照芯片 RM 手册的 Chip Configuration 章节。

5.9.5 未用到的 I/O 及模块的处理

在实际应用中，通常会有一些未用到的微控制器资源，如某些特定的内部模块、晶振引脚、

以及 GPIO 引脚等，为了提高 EMC 性能，需要对这些资源做相应配置。

- 未使用的 I/O 端口通常设置为一个特定的状态，考虑到提高 EMC 性能和低功耗应用，通常设置为输出并外部上拉；
- 未使用的晶振引脚建议配置为 GPIO 功能，避免外部信号的干扰；
- 没有用到的模块应该禁用或者“冻结”。

5.10 设计最小系统硬件电路的 Check List

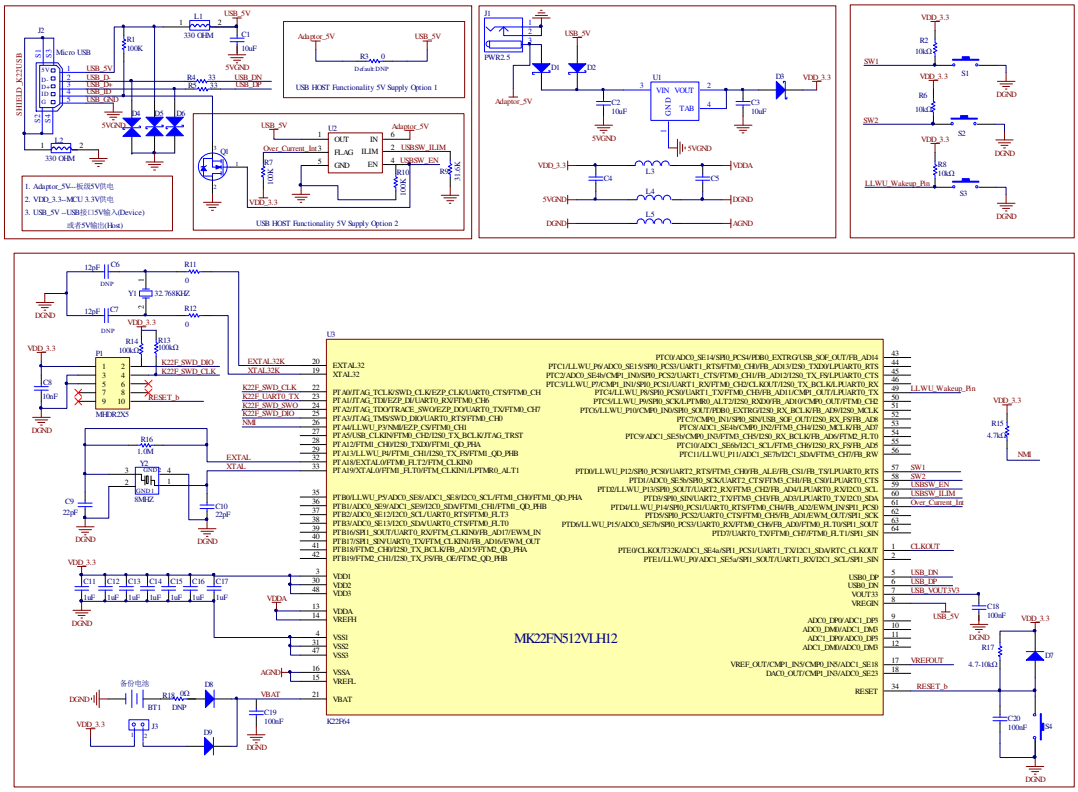
用户设计 K22 的最小系统时，可参看下表进行注意事项的检查。

附表：最小系统设计 Check list

	引脚/信号	注意事项
1	Reset	建议通过电阻把该引脚拉至高电平（4.7k-10k），同时靠近 MCU 放置 100 nF 电容；
2	NMI	引脚内部具有较小的内部上拉电阻，但建议采用外部 4.7 kΩ 至 10 kΩ 上拉电阻，并且 NMI 引脚可以用作低功耗的唤醒引脚；
		如果复用为其它功能，且外部拉低，需要通过 Flash NVM 配置 Disable NMI 功能；
3	XTAL/EXTAL	根据应用场合的不同选择合适的电路，参见“外部晶振设计”章节；
		不使用时建议配置为 GPIO 输出功能；
4	XTAL32/EXTAL32	外部不需要匹配电容和反馈电阻；
5	VDDx/VSSx	推荐每对电源引脚分别放置至少一个去耦电容（0.1uF 的陶瓷电容）。并且旁路电容的放置必须尽量靠近 MCU 电源引脚；
6	LLWU Pin	并不是所有的引脚都支持 LLWU 唤醒功能，具体可查看芯片”Chip configuration”章节；
7	SWD_CLK/ SWD_DIO	尽量缩短线的长度 <= 6 英寸，最大程度避免串扰；使用的仿真连接的 USB 线也不要过长，尽量选用质量好一些的短的 USB 线；
8	LPUART_TX	在 UART transmitter disable 并且芯片进入低功耗状态时，该引脚会出现 tri-state 从而产生漏电流，所以在低功耗应用中建议使能内部上拉功能，或者外部上拉，详细信息参见 Errata: e7986。

5.11 K22 最小系统原理图

附图：K22F 120M LQFP-64pin 最小系统原理图



第六章 飞思卡尔单片机软件开发指南

6.1 概述

前面的章节介绍了飞思卡尔单片机的硬件设计方法，本章将会详细介绍软件开发的各个环节，让开发者能够从零开始，快速完成软件开发流程。

6.2 开发环境设置与开发工具使用

飞思卡尔的 MCU，具备完善的生态系统，支持各种 IDE 和调试工具，并且提供了各式各样便捷的工具，即使是初学者，也能轻松完成。

6.2.1 连接开发板

飞思卡尔提供了种类繁多而且容易使用的开发工具及环境，极易上手，用户可以不用花太多精力在软件开发之外的东西，只要简单的连几根线，即可开始开发工作。

首先要使用的是飞思卡尔的开发板。能见到的主要有两种，一种是功能强大的 Tower 塔式系统，一种是功能比较简单的 Freedom 板。

Tower 塔式系统

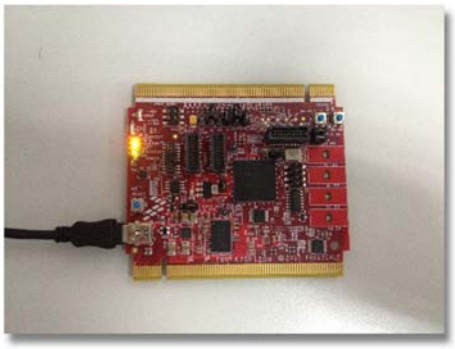


图 1. 单独使用塔式系统主板

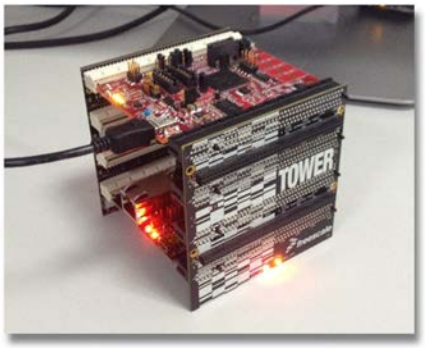


图 2. 完整的塔式系统

Tower 塔式系统的主板能够单独使用，也可以加入到塔式系统中与其他的板子共同使用。Tower 系统主板带有 JTAG 或者 OpenSDA 调试接口，不再需要外部仿真器，只需要一根 USB 线连接开发板和电脑即可实现调试下载功能，如下图 1 及图 2 所示。

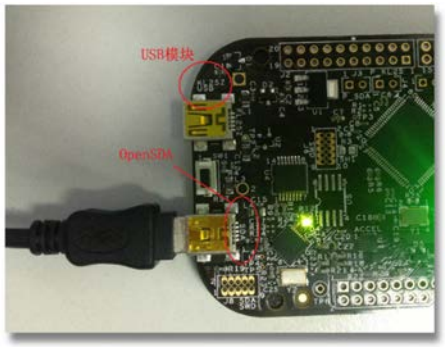


图 3. Freedom 开发板连接

Freedom 板

Freedom 板与 Tower 板相比尺寸要小很多，使用方法也更简单。Freedom 板带有 OpenSDA 调试接口，同样不需要外接仿真器，使用 USB 电缆连接电脑即可。需要注意的是板子上可能会有两个 USB 插口，一个用于 OpenSDA，另一个是芯片本身的 USB 模块接口，在插口附近会印刷有标识，如图 3 所示。

6.2.2 使用 Kinetis Design Studio（KDS）进行软件开发

选择好开发板之后，就可以开始软件的编写和调试了。飞思卡尔单片机支持所有业内主流的集成开发环境，如 IAR、Keil 和 CodeWarrior 等。开发环境的选择主要是根据用户自己的习惯，本文以飞思卡尔最新的 KDS 为例来进行说明。

Kinetis Design Studio，简称 KDS，是飞思卡尔最新推出的针对 Kinetis 系列 MCU 的集成开发环境，可以完成整个软件开发过程中的所有功能，包括编辑、编译、调试下载等，目前最新版本 1.1.1，可通过此链接下载安装软件和文档：

http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=KDS_IDE&tid=vanKDS&uc=true&lang_cd=zh-Hans

KDS 基于 Eclipse 框架，采用 ARM GCC 和 GDB 编译调试环境，同时集成了飞思卡尔的 Processor Expert 和 KSDK 功能，因为与 Codewarrior 一样都是基于 Eclipse 框架，所以使用界面与 CodeWarrior 几乎一样。KDS 专门针对飞思卡尔的 Kinetis MCU 进行了优化，用户体验更好。

打开 KDS 后，首先需要选择一个 workspace，这里推荐在 KDS 的安装目录下新建一个专门的文件夹，命名为 workspace。打开 KDS 后，我们首先新建一个工程，点击 File->New->Kinetis Design Studio Project，如图 4 至图 6 所示。

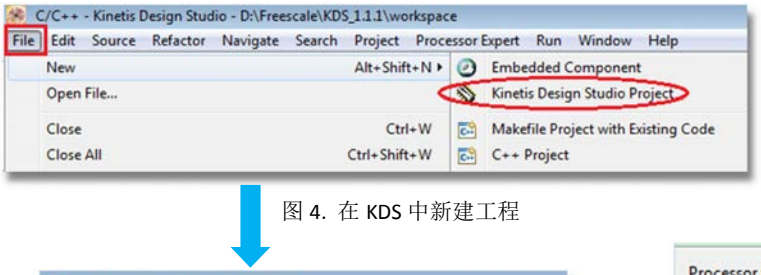


图 4. 在 KDS 中新建工程

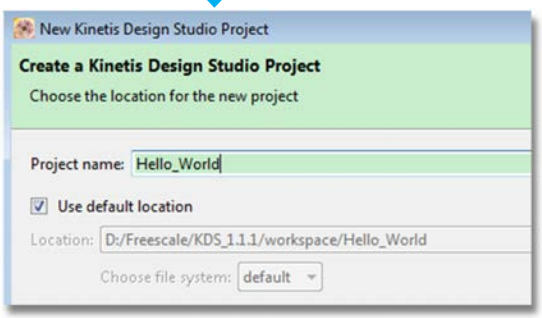


图 5. 输入工程名称

图 6. 选择对应的芯片

最后一步，选择是否需要辅助开发工具，如图 7 所示。

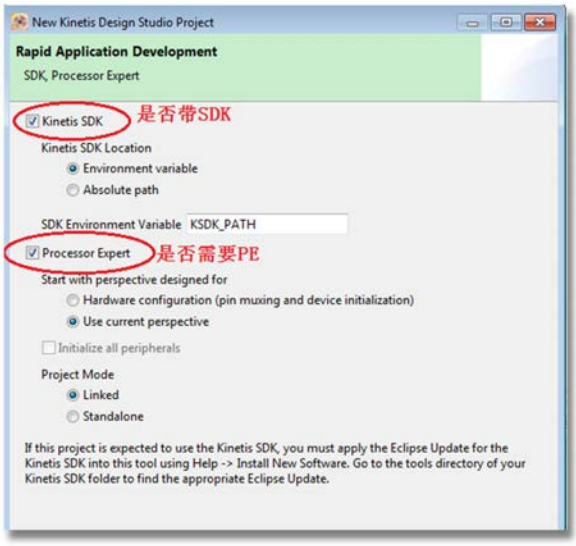


图 7. 选择辅助开发工具

SDK 是飞思卡尔提供的软件开发包，里面提供了各个模块常用的 API（比如使用 UART 发送数据），方便用户开发，在本文第三章中有介绍，具体使用方法在后面的章节中有详细的讲解。如果目标芯片支持 SDK（比如 K22），那么这里可以勾选该选项。

PE（Processor Expert）是飞思卡尔推出的一款图像化配置工具，后面也会详细说明。关于 Project Mode 选项，如果选择 Linked，那么所有的工程都会公用一套静态文件；选择 Standalone 则会单独在各自的工程目录下拷贝一份。

最后一步，点击 Finish 即可完成新工程的创建，KDS 开发环境软件的主界面图 8 所示。

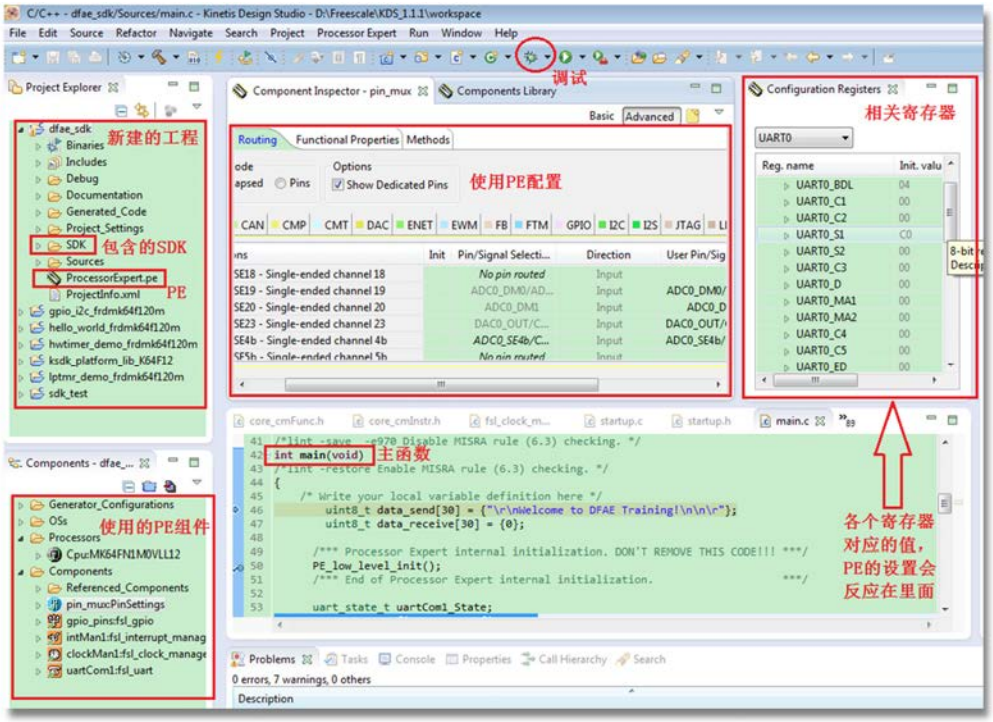


图 8. KDS 的主界面

新工程建立完成后，如果用户已经有自己的代码，也可以把他们添加到刚才新建的工程中，

如图 9 所示。

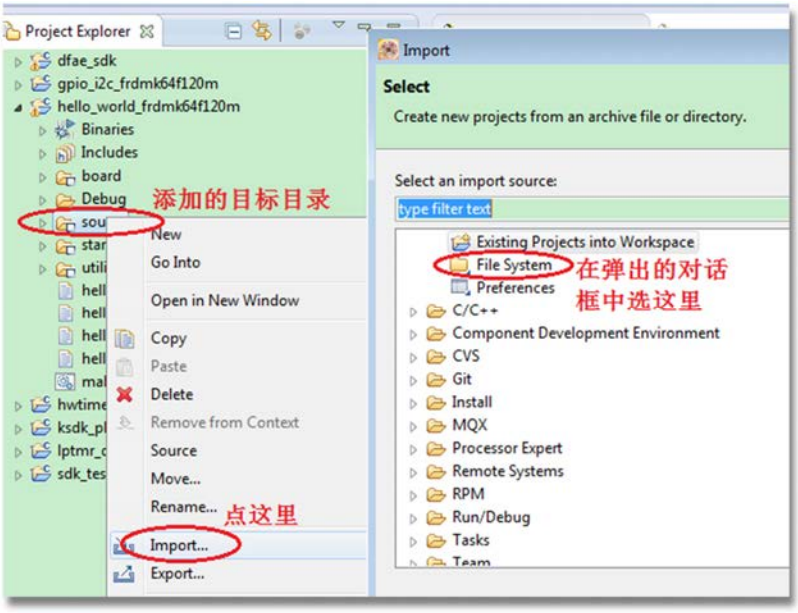


图 9. 添加已有的文件

推荐将用户代码放在 source 目录下面，或者自己新建一个目录，点 next 之后，会弹出一个对话框，如图 10 所示。

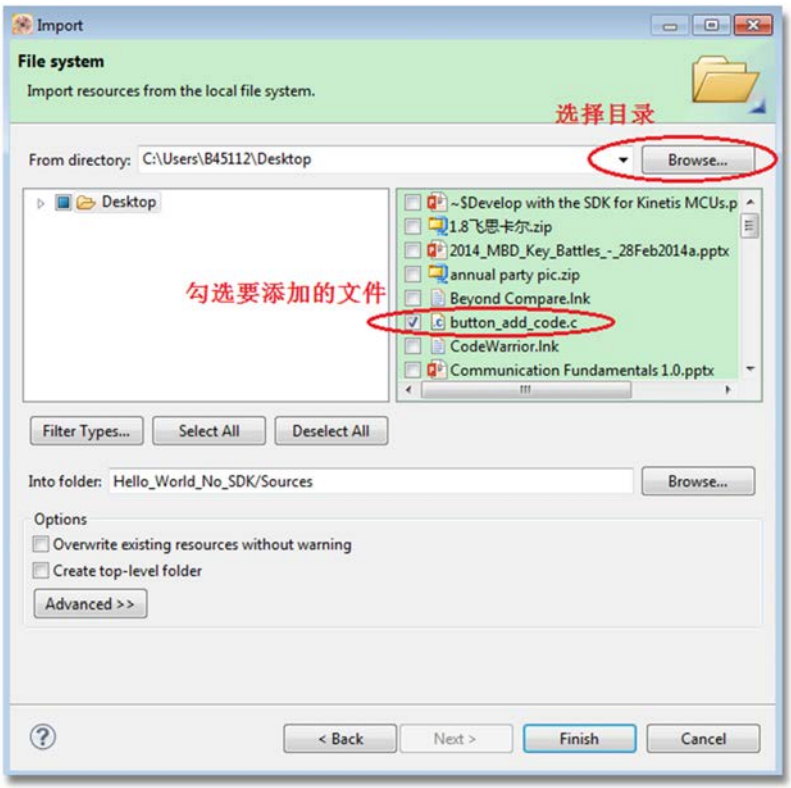


图 10. 添加已有的用户文件

点击“完成”之后，之前选中的所有文件就添加到了工程中。用户也可以通过设置包含路

径调用用户文件。在当前的工程上按右键，选 Properties，在弹出的对话框中按图 11 操作。

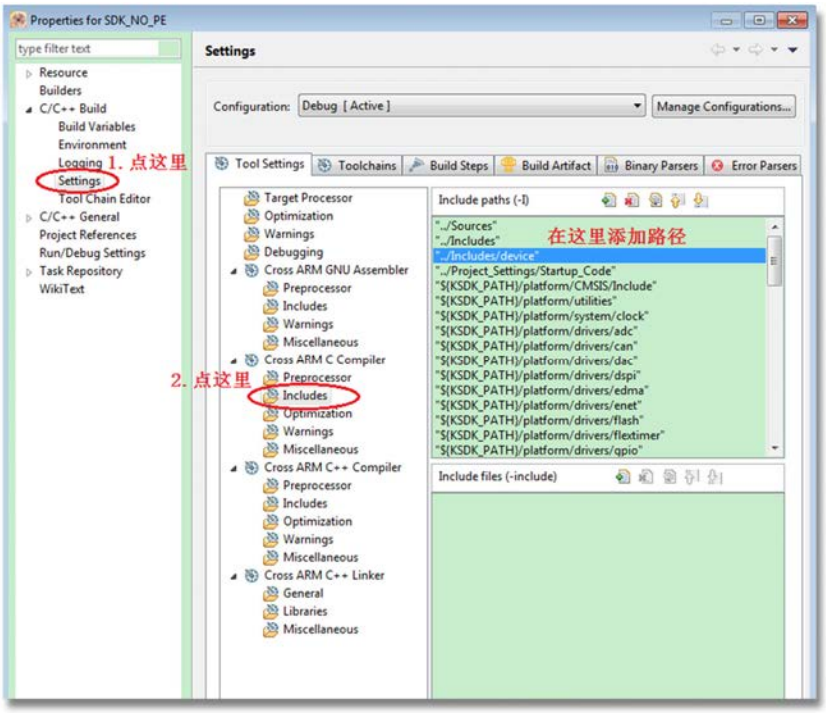


图 11. 包含已有的文件

关于 PE 和 SDK 的使用，以及 MCU 具体模块的编程，后面章节会详细讲述，本小节只说明 KDS 的使用，仅仅在 main 函数中添加最简单的一行赋值代码为例。

用户除了自己编写程序外，还可以导入现有的工程，这个过程也非常简单，步骤如图 12 至 14 所示。

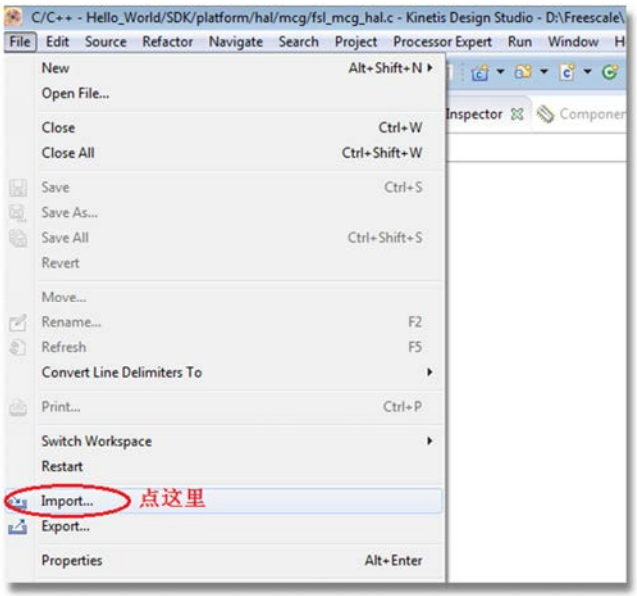


图 12. 导入现有的工程步骤 1

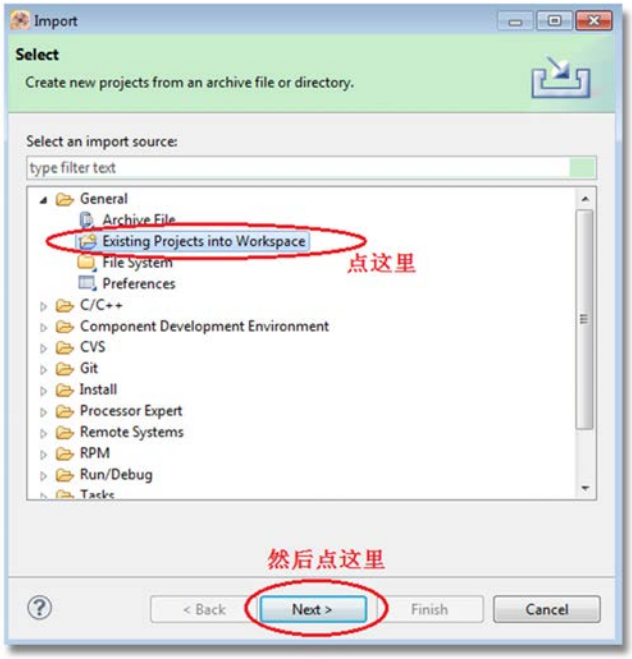


图 13. 导入现有的工程步骤 2

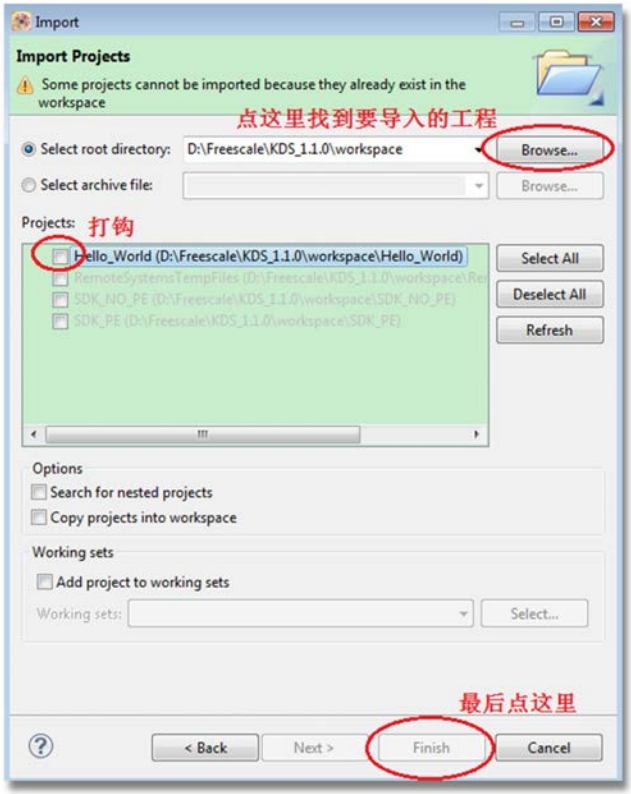


图 14. 导入现有的工程步骤 3

当工程文件都编辑好之后，便可以开始编译调试了。在对应工程上点击鼠标右键，选 Build Project，如图 15 所示。

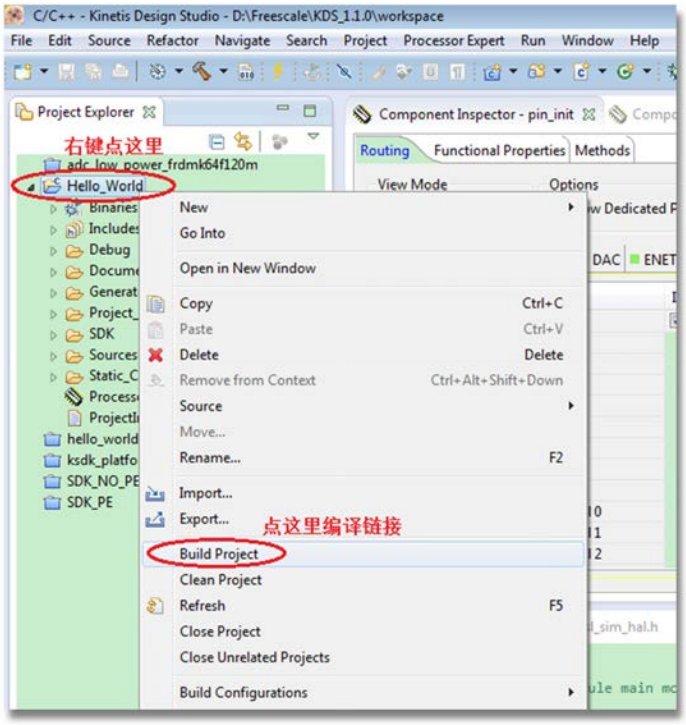


图 15. 编译工程

如果代码文件格式没有问题，编译链接通过后，就可以开始调试了。调试开始前，首先需要设置一下使用的调试器。点击小甲虫图标旁边的下拉箭头，再点击 debug configuration，弹出如下对话框，如图 16 所示，这里我们主要关注三个内容：

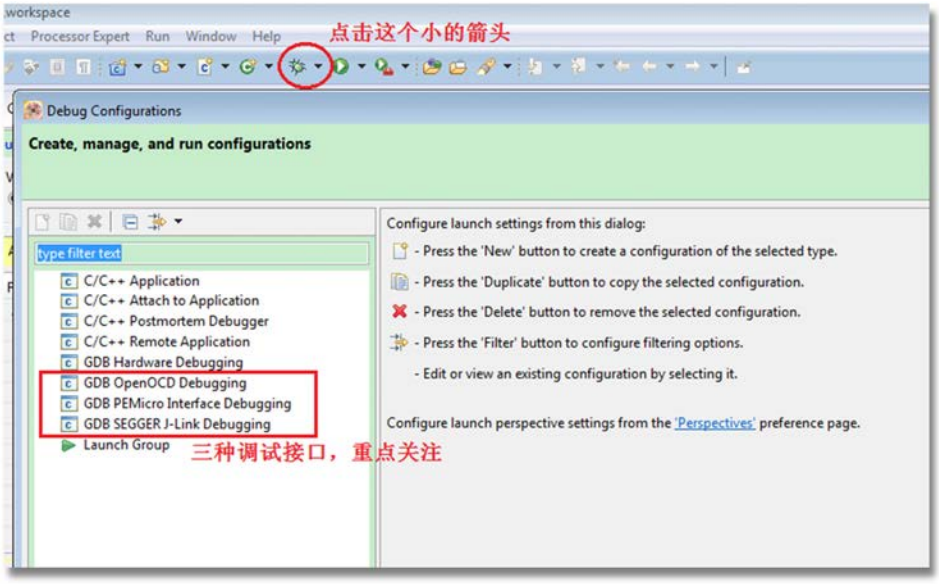


图 16. 调试接口选择

第一，选择调试接口。图 16 中显示的这三种调试接口，对应于三种不同类型的调试工具，飞思卡尔的 Tower 板和 Freedom 板，使用的都是 PEMicro 的调试接口，这里以 PEMicro 调试为例进行配置，剩余的两个调试接口的配置类似。双击调试接口的图标，就会出现具体的配置选项，点击设置，再点击 Debugger 标签，就会出现具体的设置窗口，具体图 17 所示。

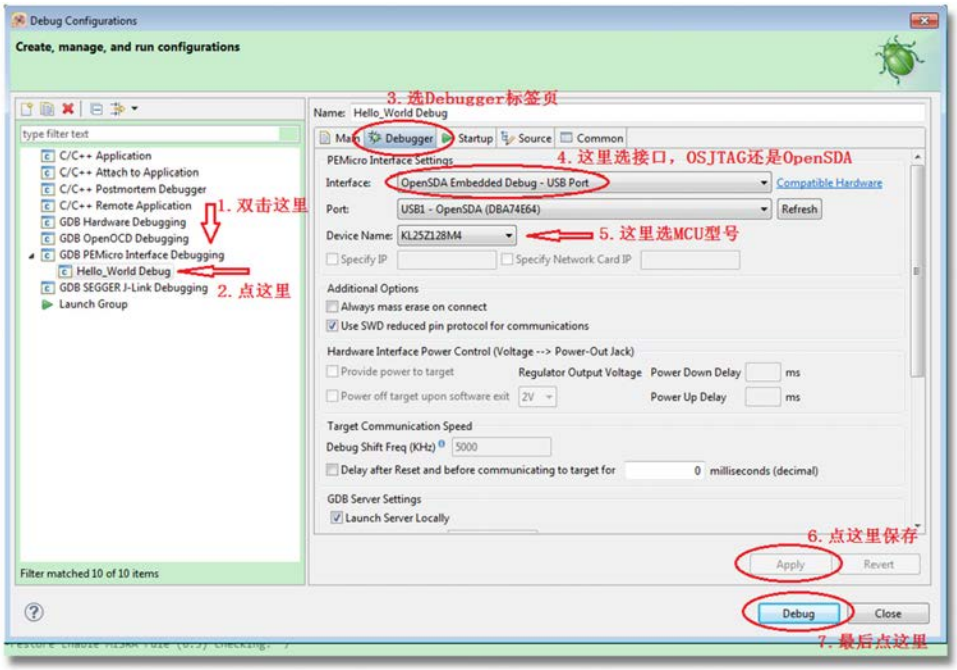


图 17. 配置调试接口信息

最后点击 Debug 以后，就会开始程序的下载和调试。程序下载完成后会弹出调试界面，其中各窗口的主要功能如图 18、19 所示。

图 18. 调试界面中的工具栏和寄存器查看窗口

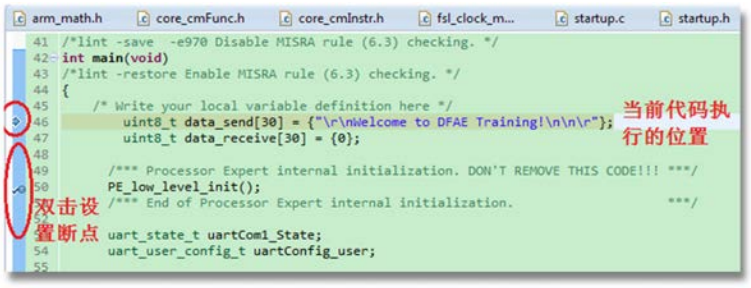


图 19. 调试界面中的代码显示窗口

在调试界面上方的工具栏和窗口中，可以单步执行代码，查看寄存器信息；在调试界面下方，可以设置断点，查看当前代码的执行情况等。

KDS 还有很多其他的功能和使用方法，用户自己可以在使用中体验，也可以参阅相关文档。

6.2.3 PE（Processor Expert，处理器专家）的使用

Processor Expert，简称 PE，是飞思卡尔推出的一款图形化设置工具，通过图形化的简洁方式，可以很方便地对 MCU 的各个模块、功能进行配置，同时自动产生对应的代码。

有了 PE 的帮助，用户可以避免阅读大篇幅的手册，不需要研究每个寄存器位的含义，只要按照提示进行选择，便可完成对 MCU 的配置。

PE 有两种不同的形式，一种就是前面章节提到的，集成在 KDS 或 Codewarrior 里面，直接添加在工程中，可以很方便地调用；另一种是独立的 PE 软件，全称叫 Processor Expert Driver Suite，目前最新版本 10.4，可通过如下链接下载：

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=PE_DRIVER_SUITE

两种方式的使用方法是一致的，考虑到文档前后的一致性，本文以集成在 KDS 中的 PE 的为例，在新建工程的时候勾选 PE，便可在工程中通过 PE 来配置当前的 MCU 了。下面举例说明 PE 的最基本的用法。具体的细节后面会有详细的介绍。

首先，整个 PE 的配置主界面如图 20 所示。

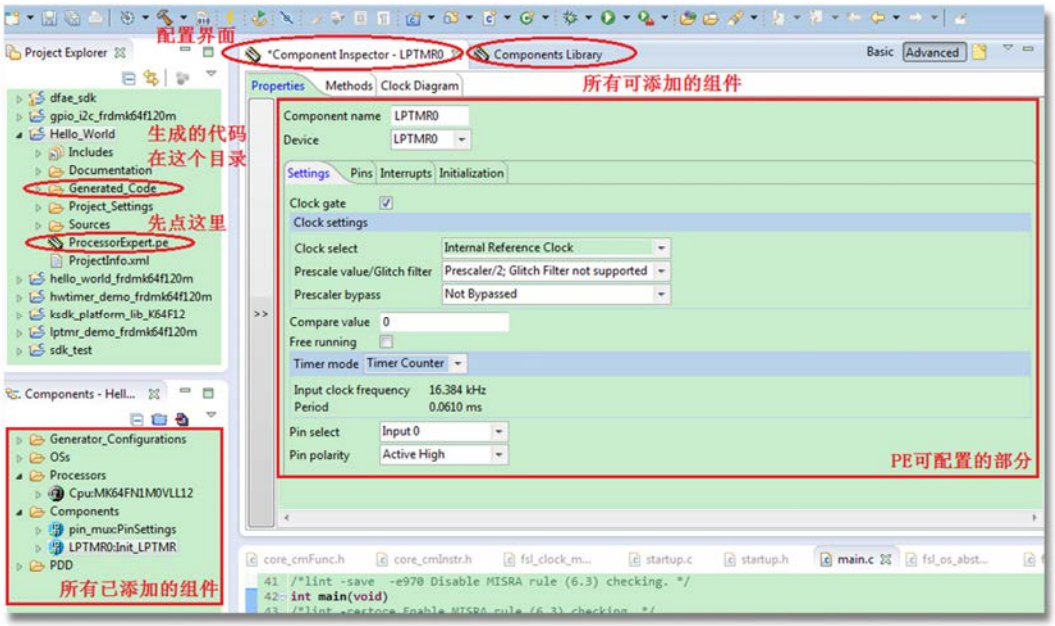


图 20. PE 的配置主界面

在 PE 中，MCU 的某个外设模块或功能模块称为一个 Component（部件）。当用户需要使用哪个模块或功能时，在上图的 Component Library 窗口中添加即可，步骤如图 21 所示。

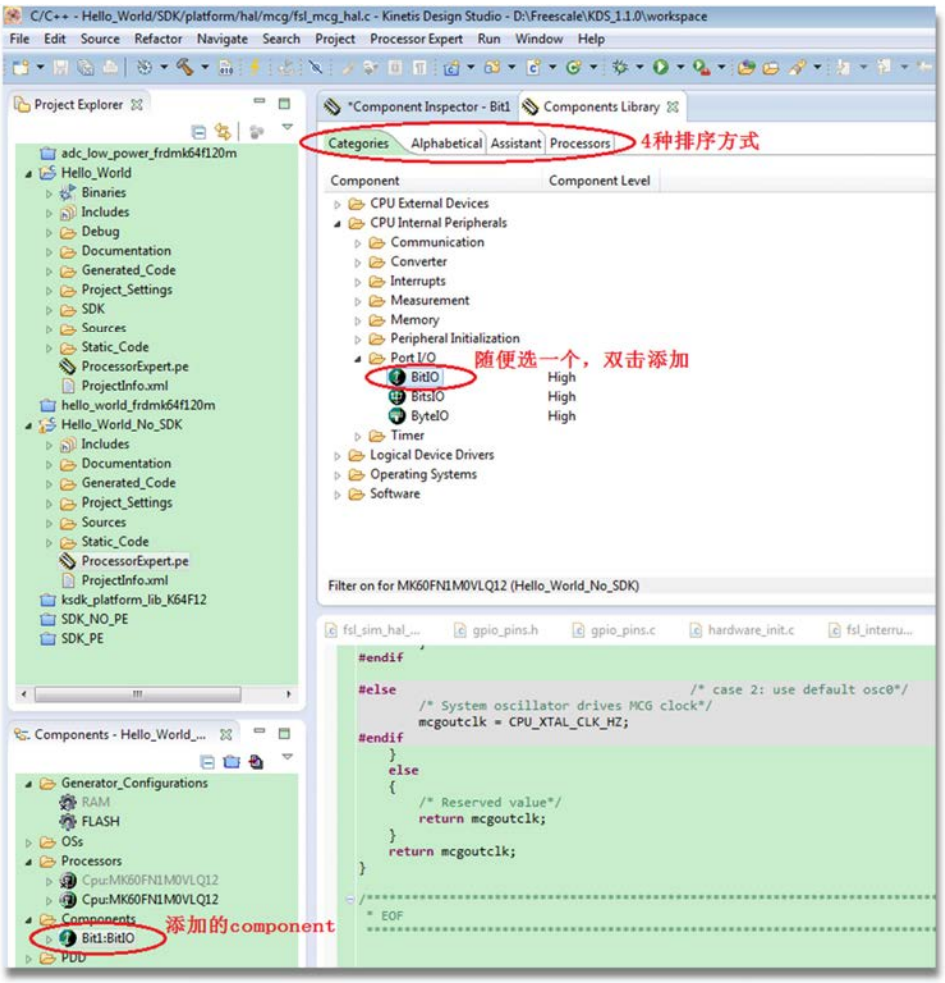


图 21. 添加所需的 component

图中标出了 Components 的 4 种不同的排序和归类方式，用户可以根据自己的习惯使用。其中 Categories 表示按模块功能分类，Alphabetical 表示按首字母排序，推荐使用这两种。

还有一点需要注意，如果添加了 SDK，那么 PE 中的 component 是不一样的，因为 SDK 使用的是不同的模块驱动，如图 22 和 23 所示。

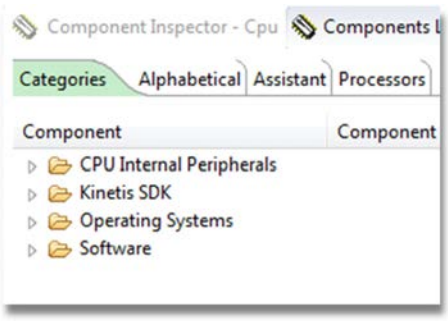


图 22. 有 SDK 时的 Components

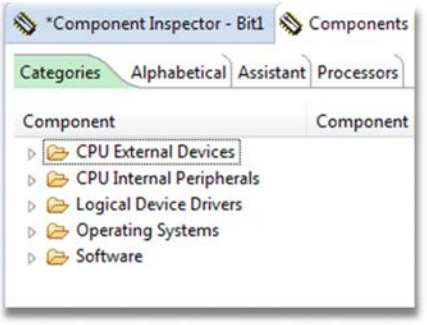


图 23. 没有 SDK 时的 Components

点击所添加的 component，就可以对其进行配置了，本文以 LPTMR 为例来说明如何对 Component 进行配置。

LPTMR 即 Low Power Timer，是一个低功耗定时器，在 MCU 处于低功耗模式下时，仍然能够通过给定的输入时钟源，正常计数并产生中断。LPTMR 内部包含一个自动递增的计数器，并能与给定的值进行比较，产生中断等动作。在 PE 中对 LPTMR 进行配置的步骤如图 24 所示。

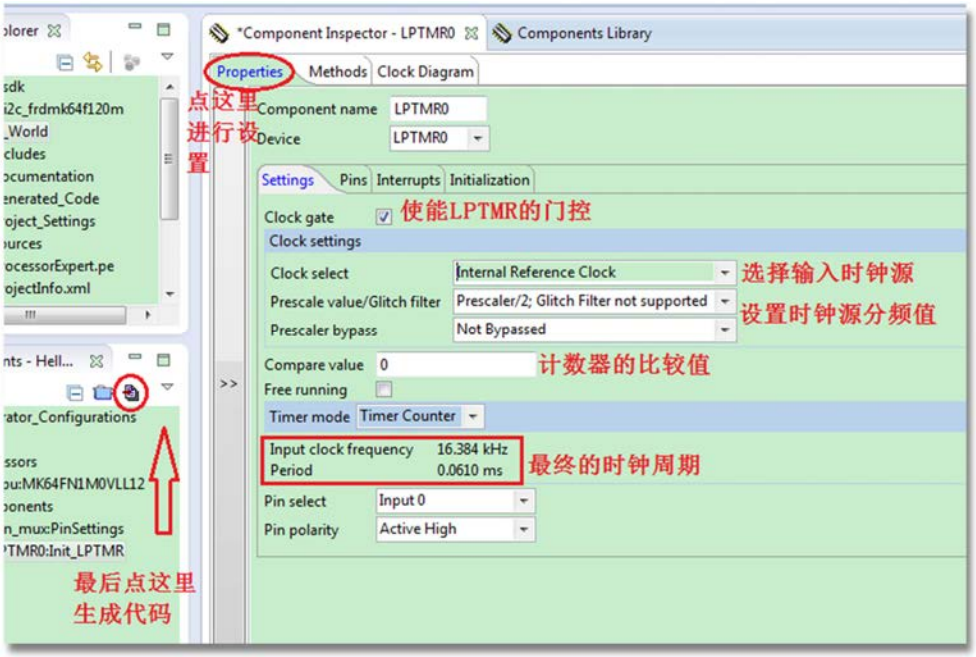


图 24. 在 PE 中对 LPTMR 进行配置

按照上图的步骤，就可以完成对 LPTMR 的配置了。图 24 中那些选择项的内容就是对 LPTMR 进行配置的细节，LPTMR 控制寄存器的每个 bit 位的定义都会体现这里，用户不用再去阅读手册具体的细节，只要根据这个图形化界面的说明进行选择即可。生成代码前，请点击保持按钮保存之前的修改。

这里有两点需要说明。首先是图形化的配置界面虽然简单，但是并不是可以随心所欲的配置，模块本身有自己的一套工作流程，不同的 bit 位之间可能会存在关联，因此有可能出现配置冲突。如果用户是手动输入的代码，则很难发现对这种 bit 位赋值错误；如果是用 PE 进行配置，就会自动提示报错，如图 25 所示。

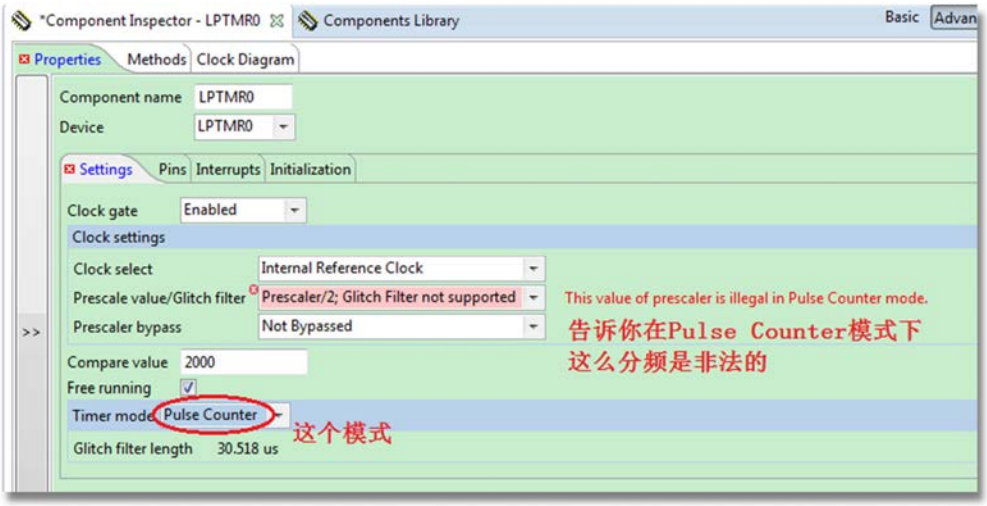


图 25. PE 的出错提示

另外，如果需要开启 LPTMR 中断，则需对其中断进行配置，如图 26 所示，另外还需要编写相应的中断服务程序。在 Generated_Code 目录下，可以找到与配置模块对应的 c 文件，本例使用的模块是 LPTMR0，所以对应的文件是 LPTMR0.c，里面可以看到 PE 生成的中断服务程序原型，如图 27 所示。

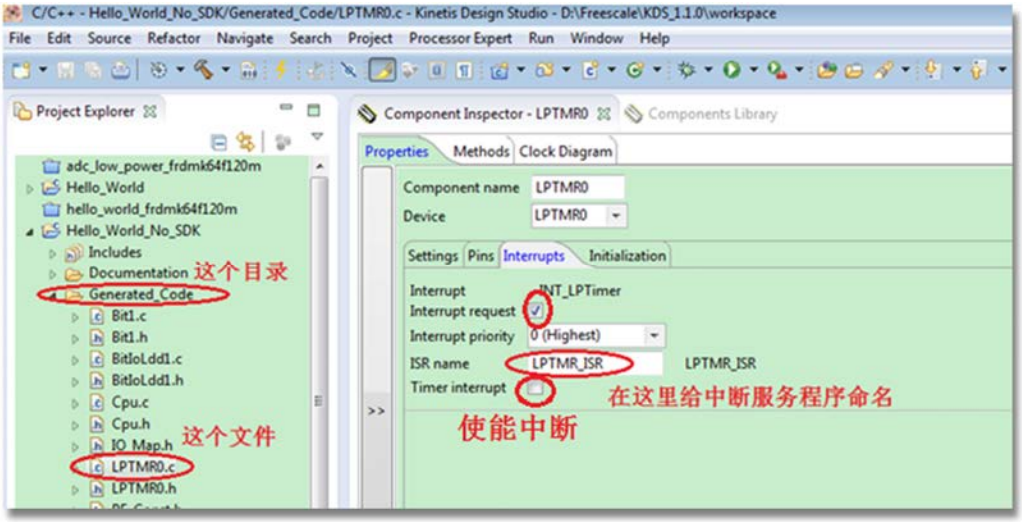


图 26. 在 PE 中开启并设置中断

用户可以在 LPTMR0.c 文件中，加入自己的中断服务程序代码。

图 27. 编写中断服务程序

因为功能不同，不同 component 的配置差别较大，这里就不一一介绍了，但使用起来都是非常简单直观的，用户在可以在使用的时候自己体会。

6.3 基本外设模块编程举例

通过前面的步骤，我们对飞思卡尔 Kinetis MCU 的开发环境、开发工具都有了初步的认识，也熟悉了基本的用法，可以正式开始编写自己的应用代码了。每一个 MCU 都会包括大量的外设模块，但是不同系列 MCU 的相同模块一般具有相同的 IP，因此软件的开发也大同小异，可以很方便的从一个系列，如 K60，移植到另一个系列上， 如 K20。本节会选出几个最具有代表性的模块，并结合参考代码，介绍其编程方法。

在开始本节之前，首先需要介绍一下赋值语句的使用。MCU 的编程，很大程度上就是对寄存器的赋值操作。在飞思卡尔的手册中，每一个寄存器都会有一个名字，同时每个寄存器里的每个 bit 位也会有一个名字，比如 SIM_SCGC5 寄存器，其中有一个 PORTA 位。同时我们对每个 bit 位也会定义一个 mask，比如 SIM_SCGC5_PORTA_MASK，这个 mask 表示对这个寄存器的 PORTA 位的掩膜位。所以如果要对这个 PORTA 位置 1，可以用这样的代码：

```
SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;
```

同样如果对这一位清 0，代码可以这样：

```
SIM_SCGC5 &= ~SIM_SCGC5_PORTA_MASK;
```

这种位操作语句可以方便地对每一个 bit 进行赋值操作。

6.3.1 时钟模块

时钟设置是嵌入式系统的核心，这一节将会详细介绍 Kinetis MCU 的时钟系统，说明怎样具体配置各个模块的时钟。以 Kinetis K22 为例，其时钟系统框图如图 28 所示。

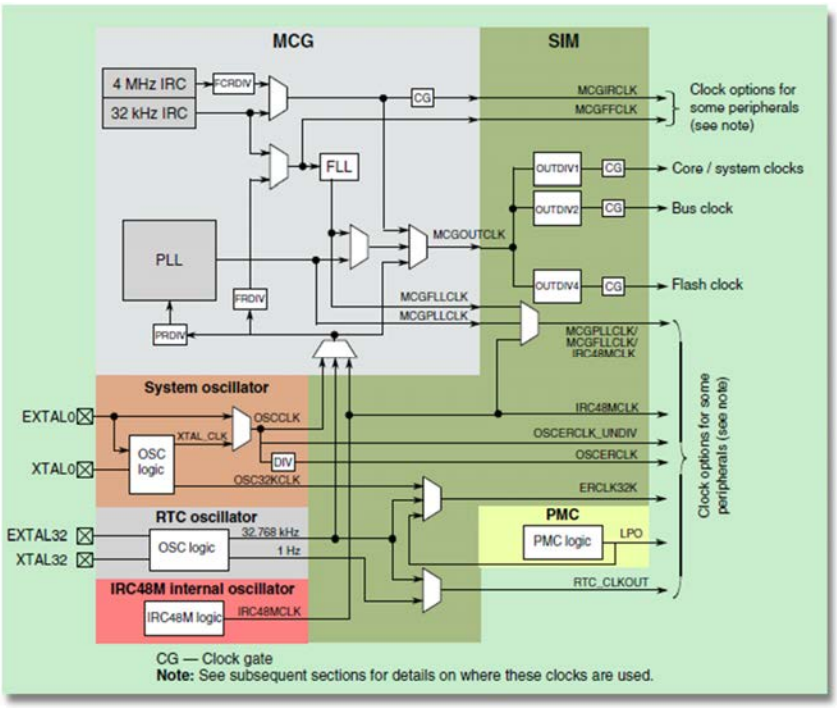


图 28. Kinetis K22 的时钟系统框图

从图中可以看到，K22 包括了 MCG 模块、SIM 模块、振荡器模块以及电源管理模块 PMC。其中振荡器模块主要是提供各种频率的时钟源；电源管理模块提供一个 1KHz 的低功耗时钟；而完成整个系统的时钟配置，最关键的是 MCG 和 SIM 这两个模块，下面分别进行介绍。

6.3.1.1 MCG：Multipurpose Clock Generator 多用途时钟发生器

MCG 模块内部包含锁相环（PLL）和锁频环（FLL），还有两个片内时钟：片内高速时钟（4MHz）和片内低速时钟（32KHz）。用户可以根据需求选择不同的时钟源，配置不同的时钟分频因子，来产生各种不同频率的时钟信号。

根据 PLL/FLL 的使用情况，以及内外时钟的选择，MCG 提供了 9 种不同的工作模式，分别是：

- 锁频环工作片内时钟模式（FEI）
- 锁频环工作片外时钟模式（FEE）
- 锁频环旁路片内时钟模式（FBI）
- 锁频环旁路片外时钟模式（FBE）
- 锁相环工作片外时钟模式（PEE）

– 锁相环旁路片外时钟模式（PBE）
– 低功耗旁路片内时钟模式（BLPI）
– 低功耗旁路片外时钟模式（BLPE）
– 停止模式（STOP）

锁频环工作片内时钟模式（FEI）是 MCG 模块默认使用模式。MCG 模块的不同工作模式（停止模式除外）之间可以来回自由切换，如图 29 所示。图 29 中标示的不是标准的模式转换图，只是其中一种方式而已。这里，以客户最常用的 PEE 模式工作为例，描述了芯片从上电到最终进入 PEE 模式的转换过程。

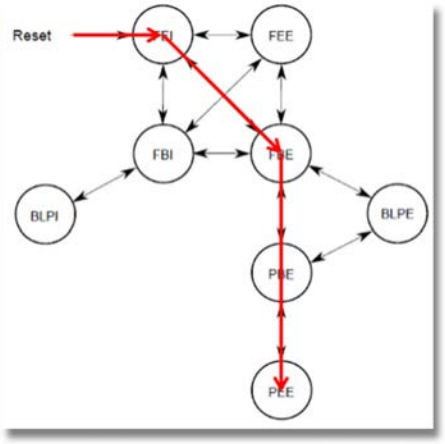


图 29. MCG 工作模式转换图

其中 MCU 上电后的默认模式是 FEI，在启动代码中，会经过图中红线标示的状态，最终进入 PEE 模式。与此对应的时钟分配如图 30 所示。

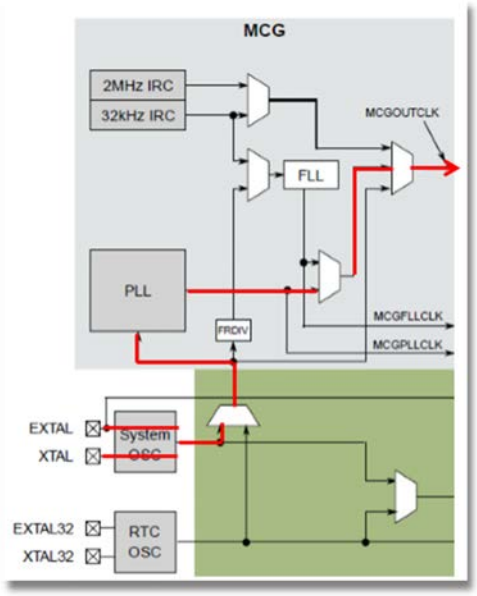


图 30. PEE 模式下的时钟分配

MCG 的编程，核心就是对各个分频因子的配置，以及 PLL/FLL 模块的设置。对 MCG 模块编程，当然可以采用读手册，写寄存器的方式来进行，但是 MCG 模块功能相对较复杂，寄存器很多，手册阅读也较吃力，这么开发软件的话，效率不高，这里推荐采用 PE 工具（Processor Expert）

来生成对应的软件代码，本文与前面的一样，也采用 KDS 环境。

打开 KDS 后，在左侧点 Processor Expert，就会出现时钟设置页面。因为对于 MCU 来说，时钟的设置是必需的，所以不需要像另外的模块一样来添加 Component。配置时钟可按图 31 到 34 所示的步骤进行。

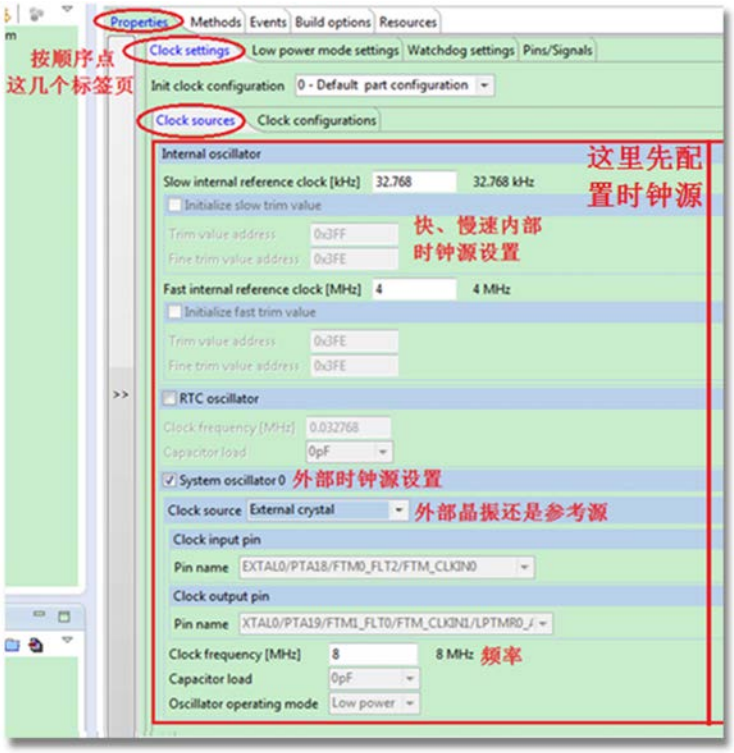


图 31. 设置系统的时钟

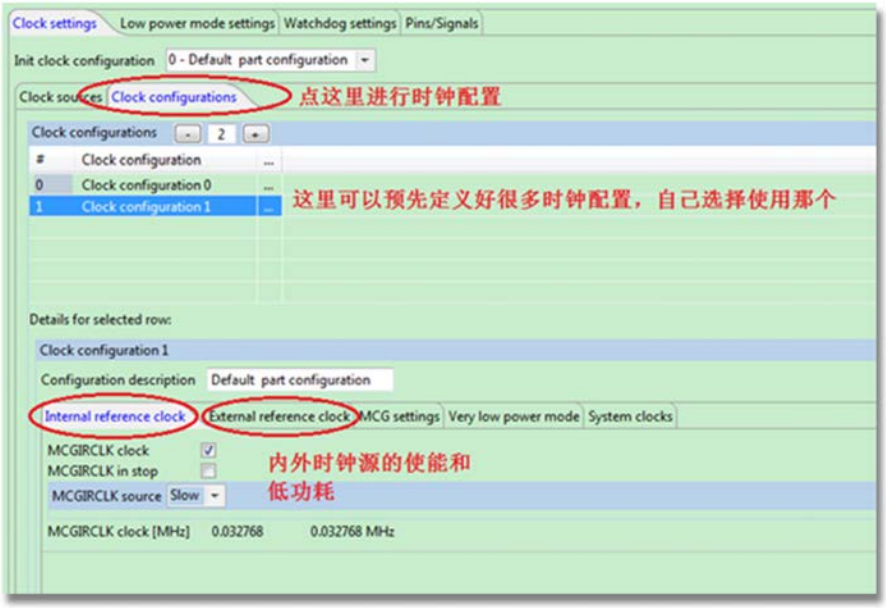


图 32. 对时钟源进行配置

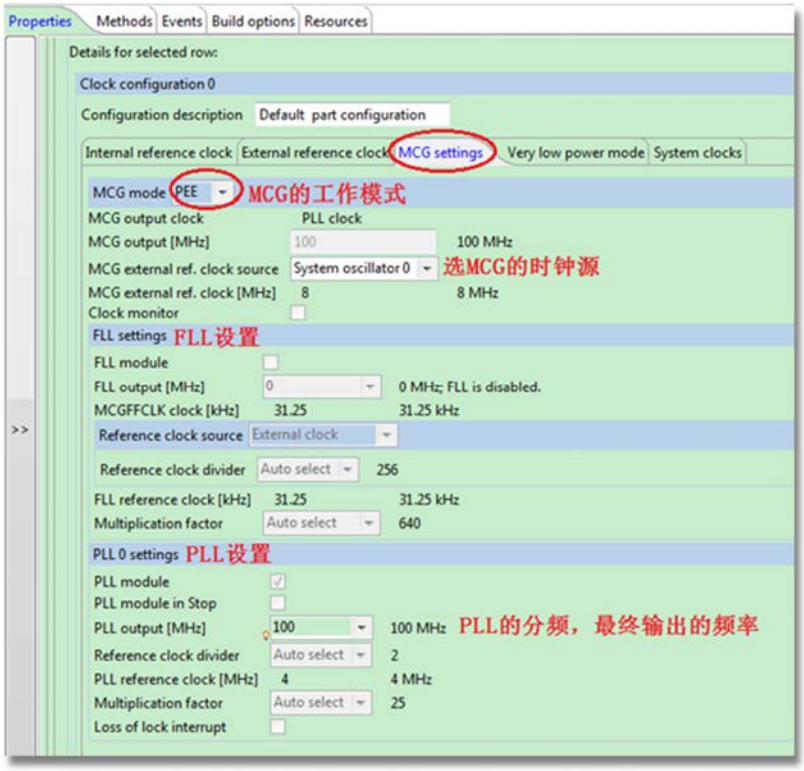


图 33. MCG 的设置

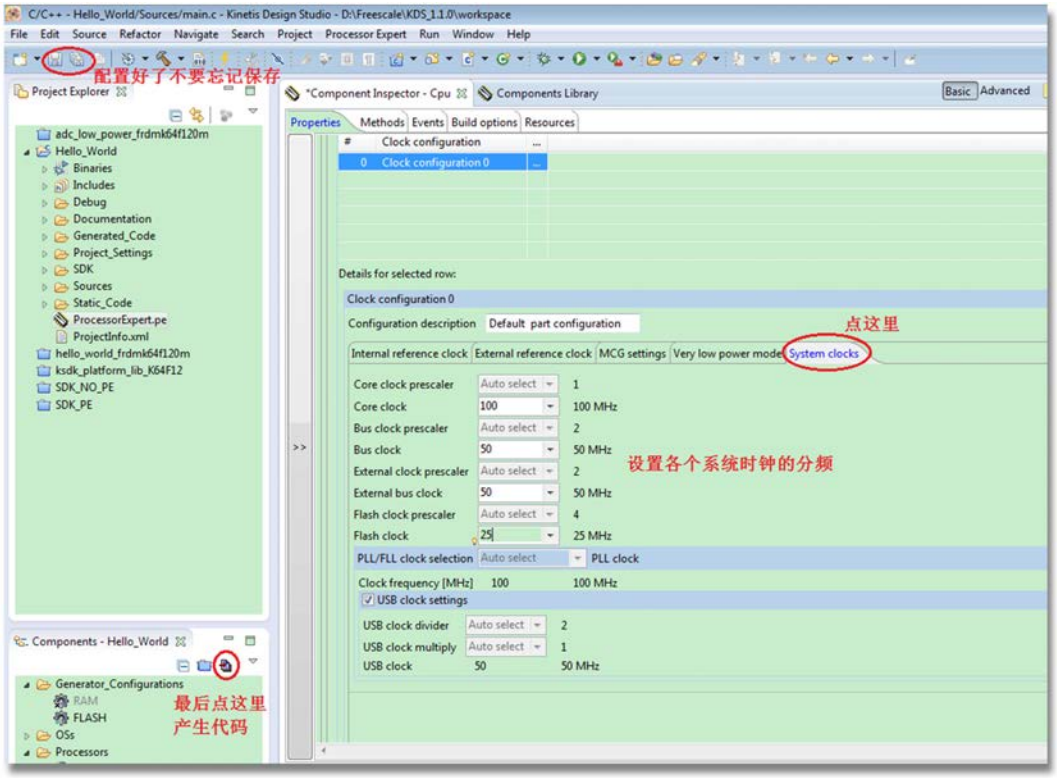


图 34. MCU 时钟设置设置

按照上面的步骤，便可以完成整个系统的时钟设置。前面的步骤中，有一些选项是灰色不

可操作状态，这和当前 MCG 模式有关，本例中选择的是 PEE 模式，因此不能对 FLL 模块进行设置，此时有关 FLL 的设置选项就是灰色的。

6.3.1.2 SIM：System Integration Module 系统集成模块

SIM 模块控制芯片内核时钟、总线时钟、外部总线时钟、FLASH 存储器时钟、USB 模块时钟等系统时钟，配置基于 MCG 输出基准时钟的分频系数。SIM 模块还用于控制外设时钟是否选通，打开使用模块时钟，关闭未使用模块时钟，这些时钟的门控，有利于降低芯片功耗。

但是对于 SIM 的编程却非常的简单，只要关注这样两组寄存器即可：SIM_SCGCx 和 SIM_CLKDIVx（x 表示 1-7 的数字，不同封装的芯片略有差异）。

①SIM_SCGC（System Clock Gating Control 系统时钟门控）：这一组寄存器控制外设模块时钟的开关。我们以 SCGC4 寄存器下面的 UART0 位举例说明，手册中是这样描述的：

10 UART0	UART0 Clock Gate Control This bit controls the clock gate to the UART0 module. 0 Clock disabled 1 Clock enabled

图 35. SCGC4 寄存器中 UART0 位的定义

如果要开启或关闭 UART0 模块的时钟，只需用最简单的赋值操作对这个位进行操作即可。对于其他模块的时钟开关，只要选定相对应的控制位，所有的位操作都是相同的。

②SIM_CLKDIV（System Clock Divider 系统时钟分频）：这一组寄存器控制 MCG 输出时钟后的分频因子，分配给内核、总线、Flash 等。以 CLKDIV1 寄存器中的 OUTDIV1 位为例，手册中是这样描述的：

31–28 OUTDIV1	Clock 1 output divider value This field sets the divide value for the core/system clock from MCGOUTCLK. At the end of reset, it is loaded with either 0000 or 0111 depending on FTF_FOPT[LPBOOT]. 0000 Divide-by-1. 0001 Divide-by-2. 0010 Divide-by-3. 0011 Divide-by-4. 0100 Divide-by-5. 0101 Divide-by-6.

图 36. OUTDIV1 位的定义

可以看到这一位定义了 MCGOUTCLK 给内核时钟的分频，再参考一下前面的 K22 时钟分配图，直接赋值，也同样方便。如果使用 Processor Expert 工具，所有对这些控制位的赋值操作都可以通过图形化界面的选择来实现。

6.3.2 GPIO

GPIO 是所有 MCU 软件最常使用的功能，对 I/O 口的编程本身并不复杂，只需要配置少数几个寄存器。

首先需要进行 PinMux 的设置。芯片在设计的时候，会把很多个功能集成在同一个物理管脚上，用户在使用的时候，需要进行设置。还是以 K22 为例，在手册的 Signal Multiplexing and Signal Descriptions 章节，会有一个表格详细地说明每个管脚的复用功能，如图 37 所示。

Pinout

121 BGA	100 LQFP	64 LQFP	64 MAP BGA	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	EzPort
E4	1	1	A1	PTE0/ CLKOUT32K	ADC1_SE4a	ADC1_SE4a	PTE0/ CLKOUT32K	SPI1_PCS1	UART1_TX			I2C1_SDA	RTC_ CLKOUT	
E3	2	2	B1	PTE1/ LLWU_P0	ADC1_SE5a	ADC1_SE5a	PTE1/ LLWU_P0	SPI1_SOUT	UART1_RX			I2C1_SCL	SPI1_SIN	
E2	3	–	–	PTE2/ LLWU_P1	ADC1_SE6a	ADC1_SE6a	PTE2/ LLWU_P1	SPI1_SCK	UART1_ CTS_b					
F4	4	–	–	PTE3	ADC1_SE7a	ADC1_SE7a	PTE3	SPI1_SIN	UART1_ RTS_b				SPI1_SOUT	
H7	5	–	–	PTE4/ LLWU_P2	DISABLED		PTE4/ LLWU_P2	SPI1_PCS0	LPUART0_ TX					
G4	6	–	–	PTE5	DISABLED		PTE5	SPI1_PCS2	LPUART0_ RX			FTM3_CH0		
F3	7	–	–	PTE6	DISABLED		PTE6	SPI1_PCS3	LPUART0_ CTS_b	I2S0_MCLK		FTM3_CH1	USB_SOF_ OUT	
E6	8	3	C5	VDD	VDD	VDD								

图 37. 芯片的引脚复用

这个表格中，横向的 ALT0-ALT7 就是表示该管脚可以用来实现哪些功能，我们以 PTE3（管脚标号 4）为例，Default 是 ADC1_SE7a，说明上电后，默认为 ADC 采样功能；从表格中看到，这个管脚还可以用作 GPIO（PTE3）、SPI1 的 SIN 或 SOUT、UART1 的 RTS_b 等。

对 PinMux 的设置在 Port Control 寄存器中。Port Control 是很多个寄存器组，命名方式为 PORTx_PCRn，其中 x 表示字母 ABCDEF，对应 PTA-PTE，表示端口 A 到端口 E；n 是数字，表示每一个 Port 包含的实际管脚。比如上文说的 PTE3，就对应于 PORTE_PCR3 寄存器，查手册，可以看到这个寄存器各 bit 的定义如下，

图 38. PTE3 的端口控制寄存器

下面详细说明其中各个控制或状态位的含义。

ISF（Interrupt Status Flag）：该管脚作为 GPIO 时，对应的中断标志位。

IRQC（Interrupt Configuration）：配置 GPIO 中断类型，比如上升/下降沿触发，电平 0/1 触发，触发使用 DMA 还是 CPU 中断。

LK（Lock Register）：设置此寄存器的 0～15 位是否锁定。锁定后，这些位不能再修改，直到系统再次复位。

MUX（Pin Mux Control）：对应管脚的不同功能，总共 3 位来确定该管脚的功能是 ALT0-ALT7 中的哪一个。

DSE（Drive Strength Enable）：设置该 IO 管脚是否可输出大电流。

ODE（Open Drain Enable）：配置管脚是否开漏输出，0 表示推挽输出，1 表示开漏输出。

PE（Pull Enable）：配置是否使能内部上下拉电阻。

PS（Pull Select）：0 表示下拉，1 表示上拉。

只要使用简单的赋值语句，给这些寄存器赋对应的值即可。本节重点讨论 GPIO 功能使用，当把管脚配置成 GPIO 后，便可以对 IO 口进行操作。

GPIO 寄存器根据端口 A 到 E，分为 GPIOx。在实际应用中，主要使用下面三组寄存器：

GPIOx_PDDR：设置该 IO 口的方向，0 表示输入，1 表示输出。

GPIOx_PDIR：该管脚配置为输入时，这一位表示管脚上的输入逻辑。

GPIOx_PDOR：该管脚配置为输出时，输出的逻辑，0 为低电平，1 为高电平。

这些寄存器的每一个 bit 对应于一个 IO 管脚，比如 GPIOA_PDDR[2]就对应 PTA2 管脚，对这些寄存器直接操作，就可以配置 PTA2 管脚的状态。

最后还有一点需要补充，用户在实际使用时，如果有部分管脚没有使用，建议配置为输出，并且输出低电平并接地。这样处理对 EMC 性能最好，当然如果要求不高的话，不接地悬空也是可以的。

6.3.3 ADC 模块

ADC 转换也是用户常用的模块，这一小节主要介绍一下飞思卡尔 Kinetis MCU 中 ADC 模块。很多 Kinetis MCU 片上集成了一个 16 位精度的逐次逼近 ADC，支持单端和差分输入，此外还有硬件平均、校准等其他功能，能够满足基本的工业应用的需要。

首先是 ADC 模块的系统框图，第一眼看上去有些复杂，但是用户不用担心，按照红线圈起来的几个重点部分划分，其实很容易理解。

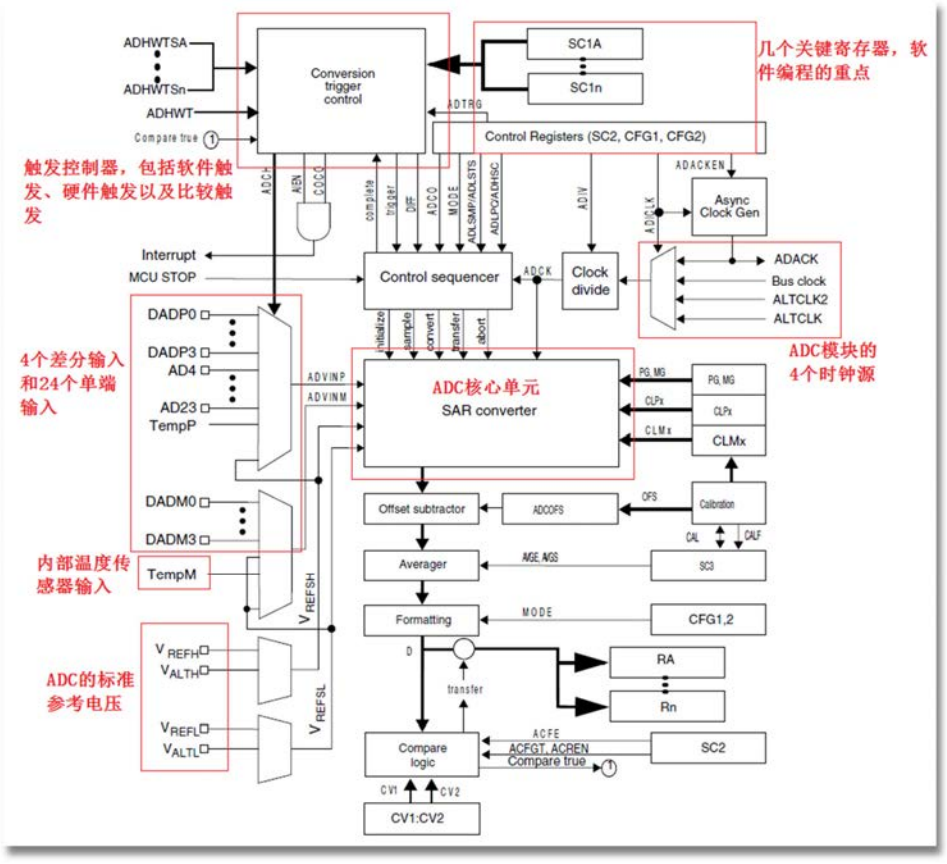


图 39. ADC 模块框图

ADC 模块本身比较复杂，涉及的功能也很多，由于篇幅限制这里不能一一详细讲解，本文的重点是几个关键的部分，让读者能够快速上手，利用 ADC 模块实现最基本的功能。

首先需要配置 PinMux，把用来进行 ADC 采样的管脚，配置成对应的 ADC 功能。

其次需要知道 ADC 的触发模式，即在什么条件下触发 ADC 采样。Kinetis 的 ADC 模块有两种触发方式，一种是硬件触发，MCU 内部的其他模块（比如比较器、Timer 等）和 ADC 有内部互联，满足指定条件就可无需 CPU 干预，直接触发 ADC 采样（比如 Timer 达到一个计数值）。另外一种是软件触发，只要对某一个寄存器写操作，就可以触发采样。对于触发方式的选择，可配置 ADC_SC2 寄存器的 ADTRG 位。

如果要使用硬件触发方式，首先需要配置触发源，相应的控制位在 SIM 模块的 SOPT 寄存器中，在 K22 中是 SIM_SOPT7 寄存器，里面有 ADC1TRGSEL 和 ADC0TRGSEL 位，对应于两个独立的 ADC 模块，ADC0 和 ADC1，如图 40 所示。

3–0 ADC0TRGSEL	ADC0 trigger select Selects the ADC0 trigger source when alternative triggers are functional in stop and VLPS modes. . 0000 PDB external trigger pin input (PDB0_EXTRG) 0001 High speed comparator 0 output 0010 High speed comparator 1 output 0011 Reserved 0100 PIT trigger 0 0101 PIT trigger 1 0110 PIT trigger 2 0111 PIT trigger 3 1000 FTM0 trigger 1001 FTM1 trigger 1010 FTM2 trigger 1011 FTM3 trigger 1100 RTC alarm 1101 RTC seconds 1110 Low-power timer (LPTMR) trigger 1111 Reserved

图 40. ADC 触发源选择

对于触发源的配置，需要设置 ADC_SC1 寄存器。有几个触发源，就会有几个 SC1 寄存器，从 SC1A 到 SC1n，它们一一对应的。用户可以参考 ADC 模块框图最上面的部分。SC1 寄存器包含 ADC 转换功能的控制位，以及相应的标志位，主要包括：

COCO：转换完成标志，为 1 说明当前转换已经完成。

AIEN：中断使能。

DIFF：是否启用差分模式，0 表示单端输入，1 表示差分输入。

ADCH[4:0] 输入通道选择，选择使用哪一个引脚来采样数据，具体看手册引脚定义，该配置与差分模式也有关系。

而对于软件触发，只要 SC1A 寄存器的 ADCH 位不是全 1，那么对 SC1A 寄存器的任意写操作，就可以触发一次 ADC 采样了。

软硬件触发的选择，取决于 ADC_SC2 寄存器的 ADTRG 位。

在开始 AD 转换之前，除了触发方式和触发源的选择，还有一些其他的寄存器需要配置，主要是下面几个：

CFG1 寄存器，其中包括：

ADIV[6:5]：时钟分频，分别对应输入时钟的 1/2/4/8 分频。

ADICLK[1:0]：ADC 模块输入时钟选择，可以在总线时钟、异步时钟等时钟源中选择一个作为 ADC 模块的时钟输入。

ADLPC：低功耗设置，为 1 表示启用低功耗，牺牲采样率换取功耗。

ADLSMP：长/短采样时间设置，1 表示长采样时间。

CFG2 寄存器，其中包括:

ADHSC：高速配置，如果输入时钟源频率很高，置 1 该位可获得更快的总转换时间。

ADLSTS[1:0]：长采样时间，当前面提到的 ADLSMP=1，即选择了长采样时间时，该位设

置具体添加几个时钟周期。

SC2 寄存器：

ADTRG：软硬件触发设置，1 表示硬件触发，0 表示软件触发。

DMAEN：DMA 使能。

SC3 寄存器：

AVGE：硬件平均，置 1 表示启用硬件平均。

CAL：置 1 开始校准，在校准过程中会保持为 1，完成后自动清零。

总而言之，这些配置都是 ADC 的一些功能设置，能够利用采样时间，换取更好的精度或者功耗性能，具体怎么设置，需要根据实际需求。

除了上面这些功能以外，ADC 还有很多其他的功能，比如比较触发、DMA 搬运数据等，本文不再细讲，感兴趣的用户，可以自行查阅手册。

配置好了 ADC 模块，并且满足触发条件，ADC 就会开始工作。当 SC1n 寄存器中的 COCO 位置 1 后，说明采样结束。此时的结果，会存放在 ADC_Rn 寄存器中，n 也表示触发源的个数，Rn 寄存器的数量与 SC1n 一样，比如 SC1B 的结果存放在 RB 中。

ADC 的详细主要功能和使用方法可以查阅手册，也可以参考飞思卡尔官方的示例代码。

当然用户也可以通过 PE 图形化的操作界面来配置 ADC 模块，这里就不再赘述。

6.3.4 UART 模块

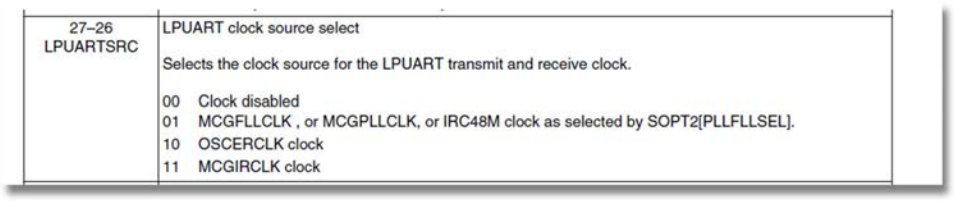
Kinetis 的 UART 模块功能非常强大，支持硬件流量控制、红外、ISO7816、地址匹配等很多功能，作为面向初学者的起步文档，本文仍然着力于 UART 的基本功能，即数据的收发，让开发者能够快速上手，实现 UART 的基本功能。

Kinetis MCU 一般有多个 UART 模块，这些模块可能会有一些差别，以 100MHz 的 K22 为例，如表 1 所示。需要注意的是，不同系列的 MCU，如 120M 的 K22，其模块硬件配置是有所不同的，用户还需仔细查阅用户手册的说明。

表 1. UART 模块的差别

UART 模块	是否支持 ISO7816	FIFO 深度	模块时钟	最大波特率	低功耗支持
UART0	是	8	内核时钟，最大 100MHz	6.25Mbps	否
UART1	否	1	内核时钟最，大 100MHz	6.25Mbps	否
UART2	否	1	外设时钟最大，50MHz	3.13Mbps	否
LPUART	否	无 FIFO	多时钟可选	6.25Mbps	是

一些 UART 模块只能选取总线时钟，另一部可以自由配置，对于 UART 模块的时钟配置，需要看 SIM 这一章节，对应于 SOPT 寄存器。以 K22 为例，在 SOPT2 中，时钟的选择如图 41 所示。



27–26 LPUARTSRC	LPUART clock source select Selects the clock source for the LPUART transmit and receive clock. 00 Clock disabled 01 MCGFLLCLK , or MCGPLLCLK, or IRC48M clock as selected by SOPT2[PLLFLLSEL]. 10 OSCERCLK clock 11 MCGIRCLK clock

图 41. UART 时钟源配置

可以分别选取内部、外部参考时钟，FLL\PLL 时钟作为 UART 模块的时钟。不同时钟源会影响到波特率的设置，同时在低功耗模式下，要确保时钟源没有关闭。

要使用 UART 模块完成最基本的数据收发，首先需要对 UART 进行初始化，配置通信的基本

参数，几个关键点如下，

1. **波特率：**

   UART_BDH 和 UART_BDL 寄存器的 SBR 位，共 12-bit 表示一个波特率。

   UART_C4 寄存器，里面的 BRFA 位，5 位表示对应的 32 分频。

   UART 模块的时钟：参见 SIM_SOPT2 寄存器的 LPUARTSRC 位。

   波特率 = 模块时钟频率/(16 * (SBR[12:0] + BRFD))

2. **数据帧格式：**

   UART 通信的数据帧格式如下图所示：

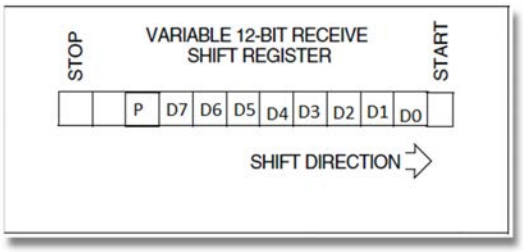


图 42. UART 的帧格式

从图中可以看到，UART 每帧的数据包括 1 个开始位，若干个数据位，紧接着是奇偶校验位，最后是停止位。

用户需根据不同的使用需求，配置如下寄存器：

UART_C1: M 位，选择 8-bit 还是 9-bit 数据位，1 表示 9-bit 数据位。

UART_C1: PE 位，置 1 使能奇偶校验。

UART_C1: PT 位，奇/偶选择，1 表示奇校验，0 表示偶校验。

当这些 UART 基本配置完成后，就可以开始收发数据了。还需要先使能 UART 收发器，这个操作在 UART_C2 寄存器中，TE 置 1 使能发送，RE 置 1 使能接收。

收发器使能完毕后，如果是发送数据，则首先需要等待 UART_S1 寄存器的 TDRE 位置 1，这一位为 1 时表示目前可以发送数据，此时只要将待发数据写入 UART_D 寄存器中即可完成发送。关于 TDRE 位是否置 1，可以采用轮询的方式查询这一位；也可以配置 UART_C2 中的 TIE 位为 1 来使能中断，一旦 TDRE 为 1，会产生一个中断。

接收数据的过程与发送类似。收发器使能完毕后，如果有数据发送过来，UART 模块会自动接收。如果 UART_S1 寄存器的 RDRF 位置 1，说明此时数据已经接收完毕，只要读取 UART_D 寄存器即可获得刚才接收到的数据。关于 RDRF 位是否置 1，也是同样可以采用轮询的方式去查询或者采用中断方式，对应的中断使能位是 UART_C2 中的 RIE。

这里还有一点需要说明，UART_D 寄存器只有 8-bit，如果使用了 9-bit 数据帧，那么第 9 bit 的数据位于 UART_C3 寄存器的 T8 或 R8 位。

UART 的其他功能，比如 FIFO 的使用、接收唤醒、硬件流量控制等，用户可以自行查阅相关手册。在本文后面介绍 SDK 的章节，还包括一个用 SDK 实现 UART 功能的例子，用户可以参考。

6.3.5 低功耗设置

Kinetis 系列 MCU 具备多种低功耗模式，用户可以在功耗和性能之间进行权衡。在说明 MCU 的低功耗模式之前，首先需要介绍一下 ARM 的内核。ARM Cortex M4 的内核，包含了三种不同的工作模式，分别是 Run，Sleep，以及 Deep Sleep。其中 Run 就是常规的运行模式，最大化性能；Sleep 模式，内核会停止工作，但是能响应中断；在 Deep Sleep 模式下，内核会处于静态，

停止响应中断。

Kinetis 在 ARM 本身的模式上，通过对外设的不同控制，扩展出了 13 种不同的模式，主要介绍见下表。

表 2. 低功耗模式说明

运行模式	描述	对应 ARM 内核模式	常用的唤醒方法	典型的电流值
RUN，运行模式	正常的运行模式	Run	\	13.83mA
HSRUN，High Speed Run，高速运行模式	最高频率运行，最大化性能	Run	\	23.6mA
VLPR，Very Low Power Run，低功耗运行	降频运行，内核频率只有 4MHz，关掉低压检测功能；片上的电压调节器只输出刚刚足够的功率	Run	\	0.61mA
WAIT，等待模式	内核进入 sleep 模式，但是外设模块可以正常工作；能够正常响应中断	Sleep	中断	4.4mA
VLPW，Very Low Power Wait，低功耗等待	与 WAIT 模式类似，只是会降频运行，关闭低电压检测	Sleep	中断	0.382mA
STOP，停止模式	芯片会进入静态状态，不响应中断，但可通过一些异步中断唤醒芯片；外设的时钟关掉，绝大多数外设会停止运行	Sleep Deep	中断	0.27mA
VLPS，低功耗停止	与 STOP 类似，只是会降频运行，关闭低压检测。**能够让 ADC 运行和支持 I/O 中断的最低功耗模式。**	Sleep Deep	中断	4.5uA
LLS3，Low Leakage Stop 3，停止 3 模式	内核时钟，系统时钟，总线时钟全部关掉。内部的逻辑、所有的 RAM 状态能够保持。	Sleep Deep	唤醒中断	2.6uA
LLS2，Low Leakage Stop 2，停止 2 模式	与 LLS3 类似，但只保留 SRAM_U 的一部分。**（CPU 内部逻辑保持的最低功耗模式，再往后唤醒相当于 CPU 复位）**	Sleep Deep	唤醒中断	2.4uA
VLLS3	内部所有逻辑掉电，所有的 SRAM 状态能够保持。	Sleep Deep	低功耗唤醒复位	1.9uA
VLLS2	在 VLLS3 的基础上，再关闭部分 SRAM（SRAM_L 的全部和 SRAM_U 的一部分）	Sleep Deep	低功耗唤醒复位	1.7uA
VLLS1	在 VLLS2 的基础上，所有的 SRAM 全部掉电，只保留 32-Byte 的系统 register file 和 32-Byte 的 VBAT register file。	Sleep Deep	低功耗唤醒复位	0.73uA
VLLS0	在 VLLS1 的基础上，再关闭 1kHz 的低功耗时钟。**所有的低功耗模式，I/O 口的状态都能保持**	Sleep Deep	低功耗唤醒复位	0.14uA

这里还需要说明一点，每个低功耗模式下的实际电流，与芯片温度、各个模块的使能情况相关，实际使用会有一些差异，本文只给出一些典型数值，具体的应用情况，请参考芯片技术

手册。

这些低功耗模式之间不能随意切换，具体的转换图如下所示。

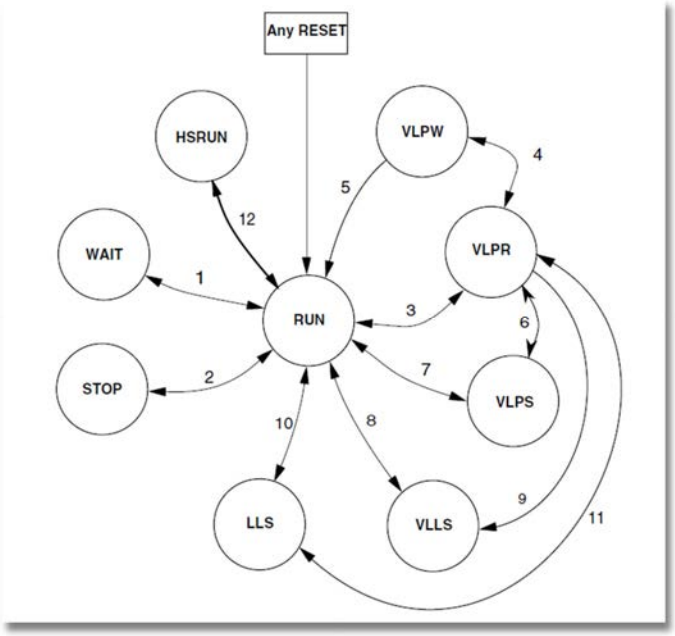


图 43. 低功耗模式状态转换图

低功耗模式进入和退出的具体操作，请参阅参考手册的“System Mode Controller（SMC）”章节，此外还有很多细节上需要注意的地方，比如时钟模式设置，一些模块的状态等。本文建议初次接触飞思卡尔产品的用户在 Freescale 官方的参考代码基础上或者采用 SDK 进行低功耗模式的开发。

6.3.6 USB 编程简介

USB 协议本身比较复杂，详细介绍超出了本文的范畴。本节不会涉及复杂的 USB 功能开发细节，只是简单介绍一下对 Kinetis MCU 的 USB 应用开发，飞思卡尔能够提供哪些资源，以及怎样使用这些资源。

对于 USB 的应用，飞思卡尔提供了一个完整 USB 协议栈，目前最新正式版 4.1.1，提供 HID、HUB、CDC 等所有常见类的支持，全面自持 Device、Host、OTG 三种方式，能很好的与 Kinetis MCU 配合。USB 协议栈的下载链接如下：

http://www.freescale.com/zh-Hans/webapp/sps/site/prod_summary.jsp?code=MEDICALUSB&uc=true&lang_cd=zh-Hans

下载后安装即可。

飞思卡尔的 USB 协议栈提供了非常全面的例程，几乎所有的 USB 类都有若干个例程可以参考，这里强烈建议初学者先打开这些例程看看，跑跑对应的代码，熟悉一下协议栈。这些例程位于“协议栈的安装目录 –>Source-> Device 或 Host 或 OTG ->examples”目录下，如下图所示。

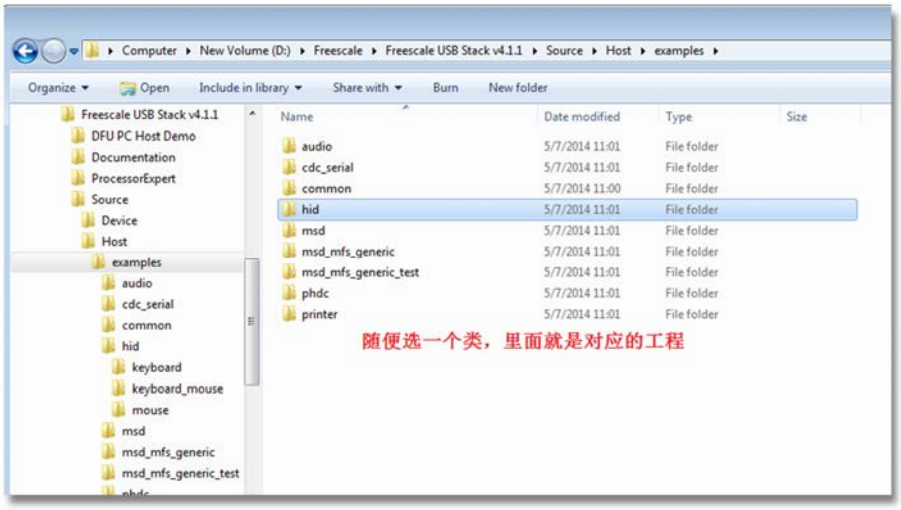


图 44. 打开 USB 协议栈工程

针对 CodeWarrior、IAR、Keil 的例程都是现成的，可直接导入，编译下载即可。

另外，安装目录下的 Documentation 文件夹里面，包含了 USB 协议栈使用的所有相关文档。这里建议先浏览一下 USBHOSTUG、USBOTGUG 和 USBUG 三个文档，分别介绍了 Host、Device、OTG 的使用，以及怎样建立自己的应用程序。关于协议栈的具体细节，限于篇幅，这里就不深入讨论了。

6.4 基于 Kinetis SDK 的开发

SDK（Software Development Kit），中文名为软件开发套件，由强大的外设驱动、协议栈、中间件和示例应用组成，旨在简化和加速基于所有 Kinetis MCU 的应用开发。现在推出的 SDK 是针对 Kinetis MCU 的，所以又称为 KSDK。前面已经提到，SDK 将整个软件进行分层，提供了各种功能丰富的函数，这些函数可直接对寄存器进行操作，用户不必自己去写对寄存器进行操作的代码，所以大大地节省了用户阅读手册的时间。

在使用 SDK 之前，首先需要完成两步的设置，分别是设置环境变量，以及在 KDS 中安装与 SDK 有关的 eclipse 插件。这两个操作的详细步骤，可参考本文档 3.5.1 节的内容。

SDK 本身包含了大量的例程，这些例程涉及到了 Kinetis 各个模块的典型用法，对于初学者，建议先导入例程进行初步的研究，对 SDK 整体上有一个大概的了解，例程存放于“SDK 安装目录\demos”下面，直接导入编译运行即可。

图 45. SDK 的库文件

需要注意的是，在编译例程工程之前，用户应该先编译 SDK 的库。SDK 库位于“SDK 安装目录\lib\ksdk_platform_lib”目录中，库本身也是一个工程，导入后先进行编译，编译结束后，就可以编译例程了。SDK 库的工程名 ksdk_platform_lib，如图 45 所示。

在 SDK 库文件的 platform 目录下，包含了该芯片所有能用到的功能模块。首先是 clock 目录，因为时钟管理由 system services 层提供支持，所以其中还包含了 system 目录，其中含有对时钟进行操作的所有函数，比如初始化、设置分频、MCG 模式切换等，具体每个函数的实现，用户可参看代码。

对比 gpio 目录，可以看到有所不同的是其中包含了 common、drivers、hal 三个目录。其中 drivers 就是外设驱动层，hal 就是硬件抽象层，里面同样包含了对 GPIO 进行操作的代码。

再看一下 lptmr 的工程文件，在其 main 函数中，包含了 lptmr 配置、初始化、开始计数这一系列功能的实现函数，都是调用驱动层的 API 来实现的，如图 46 所示。

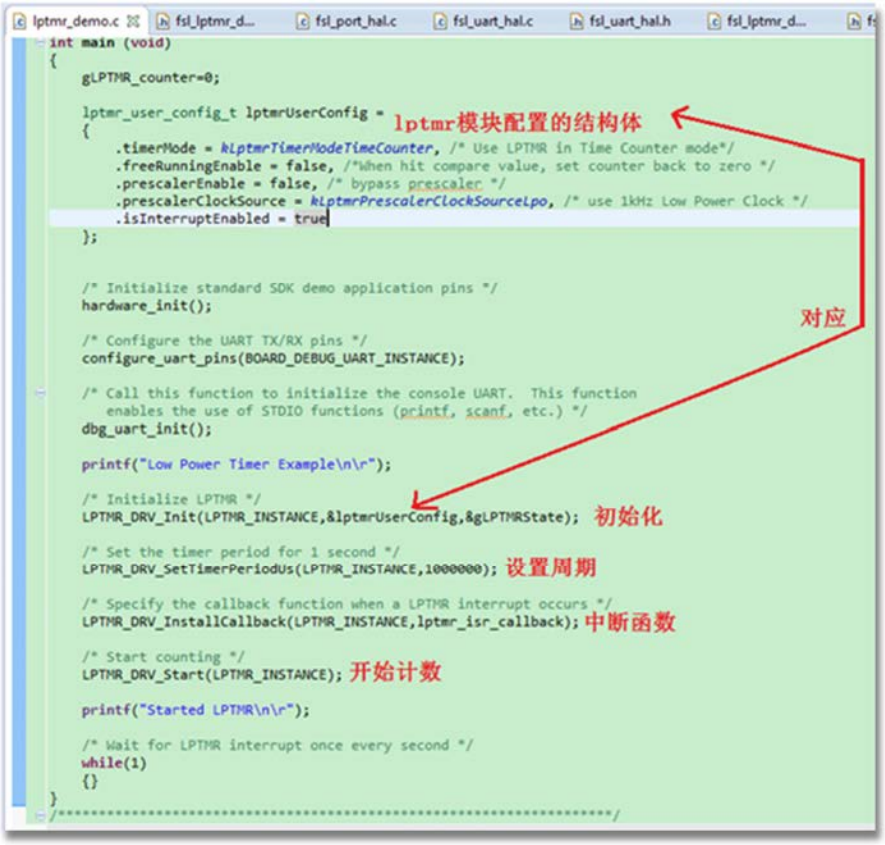
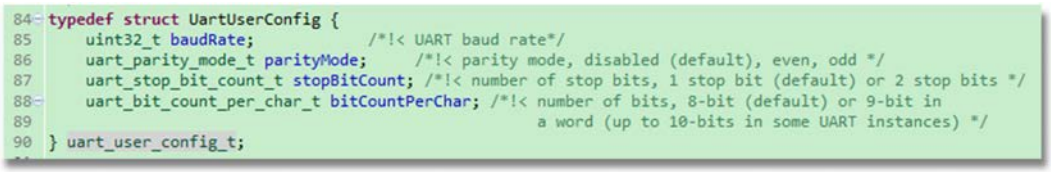


图 46. LPTMR 工程的 Main 函数调用

在 lptmr 的 main 函数中，调用了几个 driver 层的函数，这些 driver 层的函数，本身又会调用 HAL 层的函数，这里体现了 SDK 的分层结构。用户可以在 lptmr 的目录下查看各个函数的具体实现。

如果用户需要自己开发程序，可以根据实际需求，调用 platform 库中所有的 HAL、Driver 层的函数。**建议：不要同时调用不同层的函数**。

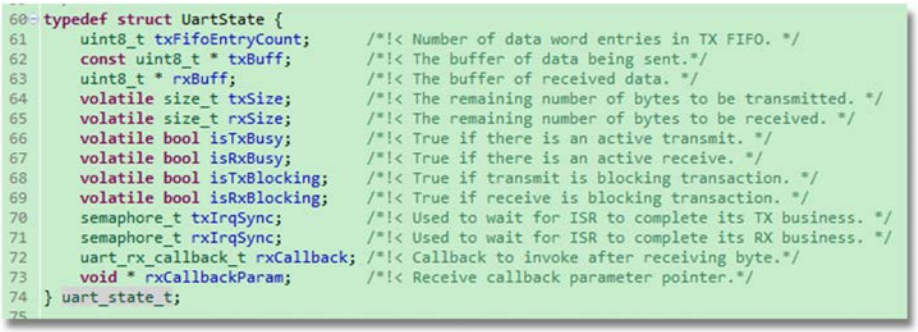
下面以 UART 模块为例，通过调用驱动层函数来实现数据的收发。每个模块驱动的使用，都需要两个结构体变量，这两个结构体，一个完成对模块的配置，一个记录模块的状态，以 UART 为例，配置结构体定义如图 47，里面定义了波特率、奇偶校验、数据帧长度等设置。



```
typedef struct UartUserConfig {
    uint32_t baudRate;           /*!< UART baud rate*/
    uart_parity_mode_t parityMode;     /*!< parity mode, disabled (default), even, odd */
    uart_stop_bit_count_t stopBitCount; /*!< number of stop bits, 1 stop bit (default) or 2 stop bits */
    uart_bit_count_per_char_t bitCountPerChar; /*!< number of bits, 8-bit (default) or 9-bit in
                                                    a word (up to 10-bits in some UART instances) */
} uart_user_config_t;
```

图 47. UART 模块配置结构体定义

状态结构体定义如图 48 所示。



```
typedef struct UartState {
    uint8_t txFifoEntryCount;      /*!< Number of data word entries in TX FIFO. */
    const uint8_t * txBuff;        /*!< The buffer of data being sent.*/
    uint8_t * rxBuff;              /*!< The buffer of received data. */
    volatile size_t txSize;        /*!< The remaining number of bytes to be transmitted. */
    volatile size_t rxSize;        /*!< The remaining number of bytes to be received. */
    volatile bool isTxBusy;        /*!< True if there is an active transmit. */
    volatile bool isRxBusy;        /*!< True if there is an active receive. */
    volatile bool isTxBlocking;    /*!< True if transmit is blocking transaction. */
    volatile bool isRxBlocking;    /*!< True if receive is blocking transaction. */
    semaphore_t txIrqSync;         /*!< Used to wait for ISR to complete its TX business. */
    semaphore_t rxIrqSync;         /*!< Used to wait for ISR to complete its RX business. */
    uart_rx_callback_t rxCallback; /*!< Callback to invoke after receiving byte.*/
    void * rxCallbackParam;        /*!< Receive callback parameter pointer.*/
} uart_state_t;
```

图 48. UART 模块状态结构体定义

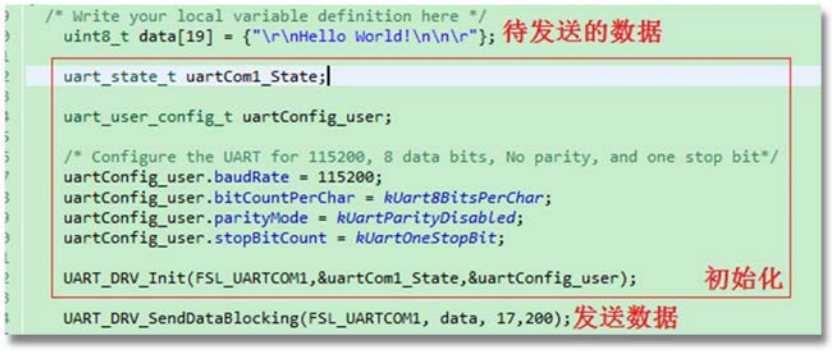
状态结构体中定义了当前 UART 模块运行时候的一些变量。

用户在使用 UART 模块时，需要首先定义这两个数据结构，然后就可以调用 UART 模块的初始化函数了，如图 49 所示。

```
    uart_state_t uartCom1_State; 定义状态结构体

    uart_user_config_t uartConfig_user;定义配置结构体

    /* Configure the UART for 115200, 8 data bits, No parity, and one stop bit*/为配置结构体赋值
    uartConfig_user.baudRate = 115200;
    uartConfig_user.bitCountPerChar = kUart8BitsPerChar;
    uartConfig_user.parityMode = kUartParityDisabled;
    uartConfig_user.stopBitCount = kUartOneStopBit;

    UART_DRV_Init(FSL_UARTCOM1,&uartCom1_State,&uartConfig_user);调用初始化函数
```

图 49. UART 模块初始化

初始化函数有三个参数，第一个参数表示 UART 实体，一个芯片里面可能多个 UART 模块，用一个数字进行区分，比如 0 表示 UART0；第二个和第三个参数就是前面定义的两个结构体。Driver 层会根据这些参数，对 UART 模块进行初始化操作。初始化结束后，就可以开始进行数据的收发了，示例代码如图 50 所示。



```
  /* Write your local variable definition here */
    uint8_t data[19] = {"\r\nHello World!\n\n\r"}; 待发送的数据

    uart_state_t uartCom1_State;

    uart_user_config_t uartConfig_user;

    /* Configure the UART for 115200, 8 data bits, No parity, and one stop bit*/
    uartConfig_user.baudRate = 115200;
    uartConfig_user.bitCountPerChar = kUart8BitsPerChar;
    uartConfig_user.parityMode = kUartParityDisabled;
    uartConfig_user.stopBitCount = kUartOneStopBit;

    UART_DRV_Init(FSL_UARTCOM1,&uartCom1_State,&uartConfig_user);        初始化

    UART_DRV_SendDataBlocking(FSL_UARTCOM1, data, 17,200);发送数据
```

图 50. UART 发送函数示例代码

在示例代码中，定义了一个数组“data[19]”来存放待发送数据，然后直接调用驱动中的发送函数 UART_DRV_SendDataBlocking() 即可。数据的接收与发送类似，只是调用不同的函数。除了收发函数，驱动层还提供了诸如反初始化、获取当前状态、终止发送等诸多函数，函数接口都不复杂，用户可以自行研究。

最后，还要再提一下 SDK 的中断使用。在启动代码中，会有一个 startup_MK64F12.s 文件（不同型号的芯片后面数字会有差别），里面定义了中断向量表，如图 51 所示。

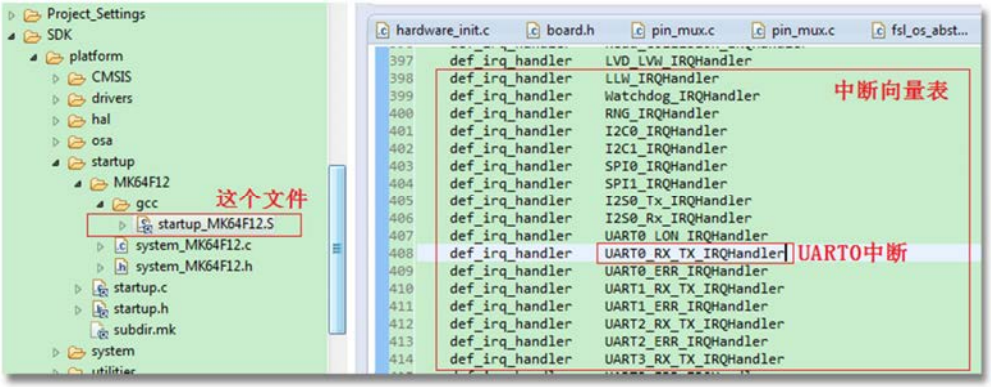


图 51. SDK 的中断向量表

右边一栏中定义的就是每个中断的中断处理函数。在 SDK 中，将中断处理函数分成了两层来调用，图中定义的中断函数会再去调用驱动层提供的函数，清标志位、中断执行代码都是在驱动层完成的，如图 52 所示。

图 52. 中断的两级调用

如果用户不需要驱动层提供的 API，想自己编写中断处理函数，只要进行简单的代码修改即可。

除了 UART 模块，其他模块驱动的调用与 UART 类似，限于篇幅，本文不一一举例。

用户在实际中编写自己的程序时，对某些函数的 API 和用法存在疑问时，可以参考 demos 目录下 SDK 自带的例程，仿照例程使用。SDK 提供的例程基本上能够涵盖所有的模块。

同时，doc 目录下还包含了很多的文档，建议先阅读 user guide，里面有一些最基本的使用说明，以及快速上手指南。更详细的每个函数细节描述，请参考文档“SDK API Reference Manual”。

6.5 怎样移植客户的应用

在很多情况下，开发者已经有了自己的一套应用代码，或者有基于其他 MCU 的代码，那么如何将其移至到 Kinetis 上面的呢？本小节将会从主要的方面给出一些基本的建议，并以 KDS 环境为例。

对于代码的移植，首先需要建立一个新的工程，并且在工程建立好之后，需要添加所有的 C 文件和 h 文件到相应的目录下，这两步较为简单，具体过程也可以参考介绍 KDS 的章节。有一点要注意，在建立工程的时候，一定要选择对应的芯片，同时推荐使用 PE 工具，因为 PE 能提供一套非常完善的初始化代码，减少移植的工作量。源代码文件全部添加完成后，就可以调用里面的函数了。

其次需要正确配置时钟，不同开发板的时钟源是不一样的。配置时钟的代码在 ”CPU.c” 或

者 ”芯片名字.c(例如 MK64XXX.c)” 文件中，里面有对 MCG、SIM_CLKDIV 等寄存器的操作，用户需要根据自己的实际情况来进行配置。当然这一步也可以直接采用 PE 来配置。

接下来需要设置中断向量表，在 KDS 中的 Vectors.c 文件里有一个指针数组，用来定义所有的中断向量的入口，如图 53 所示。

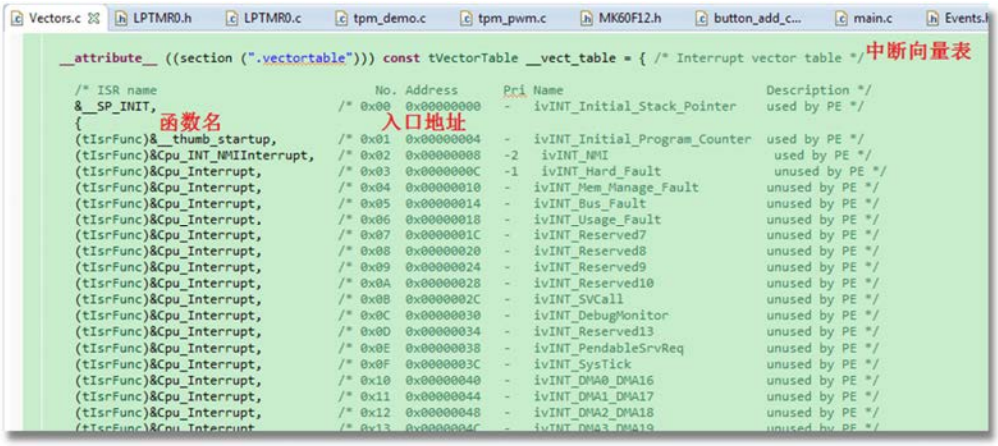


图 53. 修改中断向量表

用户需按照实际需求，将函数名改成实际使用的中断函数名。具体的中断向量号及入口地址，可以查阅芯片手册。

除了上面这些，用户还要按照目标板的需求，修改 PinMux，将管脚配置成对应的功能，具体请参考本文的前一章节，GPIO 部分。

此外，还需要修改 memory map，因为芯片的地址空间可能不同。比如原程序在 1MB Flash 的 MCU 上运行，那么移植目标芯片只有 512KB，就需要注意合理的分配地址空间，确保地址不会溢出，以免程序跑飞。

最后，还需要修改 Linker 文件，来分配存储器空间的使用。在工程目录下，会有一个.ld 文件，一般名字为芯片名（例如 K64FN1Mxxx12.ld），如果之前的代码对地址分配有特殊需求，用户可在这里进行配置，如图 54 所示。

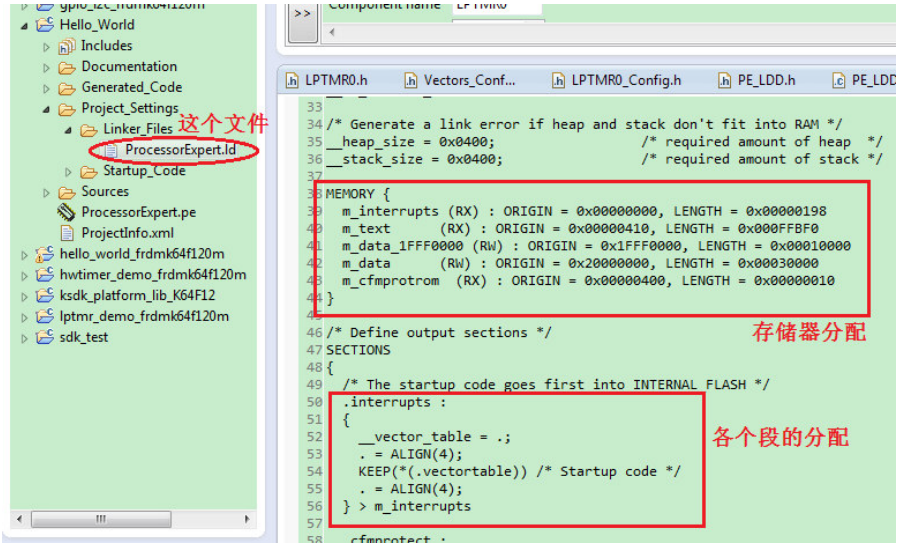


图 54. Linker 文件修改

关于 Linker 文件的具体语法，请参考文档 AN4498。此文档可以在 Freescale 官网搜索到。