# ARM 嵌入式系统开发综述

## ARM 开发工程师入门宝典

# 前 言

嵌入式系统通常是以具体应用为中心 ,以处理器为核心且面向实际应用的软硬件系统 , 其硬件是整个嵌入式系统运行的基础和平台 , 提供了软件运行所需的物理平台和通信接口 ; 而嵌入式系统的软件一般包括操作系统和应用软件 , 它们是整个系统的控制核心 , 提供人机交互的信息等。所以 , 嵌入式系统的开发通常包括硬件和软件两部分的开发，硬件部分主要包括选择合适的 MCU 或者 SOC 器件、存储器类型、通讯接口及 I/O、电源及其他的辅助设备等；软件部分主要涉及 OS porting 和应用程序的开发等，与此同时，软件中断调试和实时调试、代码的优化、可移植性/可重用以及软件固化等也是嵌入式软件开发的关键。

嵌入式系统开发的每一个环节都可以独立地展开进行详细的阐述 ,而本文的出发点主要是为嵌入式开发的初学者者提供一个流程参考。因为对于初学者在面对一个嵌入式开发项目的时候，往往面临着诸多困难，如选择什么样的开发平台？什么样的器件类型？在进行编译时怎样实现代码优化？开发工具该如何选择和使用？在进行程序调试时应该注意那些问题以及选择什么样的嵌入式 OS 等等。希望通过本文，能帮助初学者了解有关 ARM 嵌入式系统开发流程。

# 目 录

# 1 嵌入式开发平台

通常嵌入式开发的平台主要包括基于 SoC 或 MCU 开发板，板上提供常用的外设、接口和其他功能模块，开发者一般根据自己的应用需要选择适合自己板级开发平台。在这样的平台上开发者可以进行硬件的扩展，操作系统移植和应用软件的开发、调试及固化，并最终形成自己的产品推向市场。但是基于该平台的软件开发工作往往需要等到硬件平台完成后才能开展，这显然不利于缩短 TTM（Time to Market），同时调试的过程也是需要反复迭代和修改设计的过程，因此硬件方案的变动在所难免。因此在系统方案没有最终定型前，急于搭建硬件平台不仅费时费力，而且也会造成系统开发成本的提高。因此在进行方案设计的时候，利用 CPU 或者其他外设的模型进行早期的评估是非常必要的。

## 1.1 ARM 的开发平台：

- ARMulator 仿真平台

这是一套最基础的 ARM 指令集仿真器，内嵌于 ADS 和 RVDS 中，是每一位 ARM 开发者的很好的起点。ARMulator 可以模拟执行开发人员编写的 C 或汇编程序，支持源代码调试，帮助开发者确定代码编写的正确性。另一方面，ARMulator 还能大致统计出，诸如：代码执行周期数，Cache 命中率，存储器访问等利于我们优化代码的信息。但 ARMulator 是基于 CPU 的模拟，缺点在于比较难于模拟整个芯片系统的行为。

- RealView Integrator-CP 平台

http://www.arm.com/products/DevTools/IntegratorFamily.html

RealView Integrator-CP 平台（RealView Integrator Compact Platform）可以整合 Core Module。Core Module FPGA 还整合了 ARM PrimeCell 系列周边器件和内存控制器，包括 LCD，MMC 卡，音频解码，以及客户自己开发 AHB 接口器件。

- Versatile PB/AB 平台

http://www.arm.com/products/DevTools/VersatileFamily.html

Versatile Platform Baseboard（Versatile PB）是一个可以开发软硬件的 PCB 平台，可以用 LogicTile，AnalyzerTile 进行扩展，用来连接用户开发的器件，逻辑分析仪等。而 Versatile Application Baseboard（Versatile AB）主要区别是硬件扩展功能有限，因而主要用来进行软件应用开发。

- Emulation Baseboard（EB）

http://www.arm.com/products/DevTools/EB.html

EB 平台有一块相对大的 FPGA（Xilinx Virtex2 XC2V6000）可以放下用户设计的周边器件，EB 可以通过 CoreTile 和 LogicTile 进行扩展，使用户做原型验证更加方便。

- ESL 虚拟平台

http://www.arm.com/products/DevTools/RealViewCREATE_Family.html

ARM ESL 虚拟平台利用 SystemC 模型构建整个 SoC 系统，可以基于两种模型构建：时钟精确型（CA）和时钟近似型（CX），CA 模型提供了和实际硬件时钟节拍一直的精确度，利用 ESL SoC Designer 工具在 ESL CA 模型构建虚拟

仿真平台上，SoC 硬件工程师利用 ESL 工具提供的强大的诸如 Core 运行状态监视、Bus Profiling、Cache 工作状态和 Memory Mapping 等可视化插件对系统性能观测和分析，定位系统性能的瓶颈，实现硬件的性能优化和功能划分。

此外，对于嵌入式软件开发工程师而言，ESL 虚拟平台带来的最大好处是让软件开发在更早的阶段开展，而不必等到在硬件平台上进行此工作。这样以来软硬件开发工作可以并行提高，缩短产品上市时间，软硬件的协同开发还可以尽早发现系统 bug，降低开发风险和成本。同时该虚拟平台还提供了 ARM 软件开发调试工具接口同步进行软件调试，在 ESL 虚拟平台上实现软硬件的协同仿真，可以实现优化软件的目的。

从图 1 看，传统流程中容易引起反复的环节，而对引入 ESL 的开发流程，可将诸如驱动开发调试等，提前放置到虚拟开发平台上进行，实现系统设计的优化、缩短开发周期等。而且仿真环境所能提供的调试手段，是 FPGA 平台所无法比拟的。

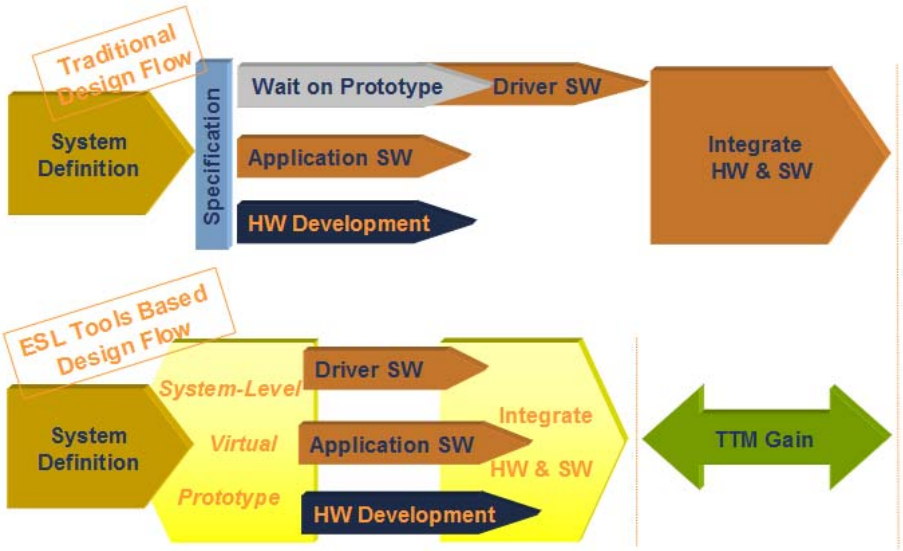


图 1 传统和引入 ESL 工具的 SoC 开发流程

- RTSM

http://www.arm.com/products/DevTools/RealTimeSystemModel1176.html

RTSM（实时系统模型）是对整个芯片系统在指令集层面上的仿真，它能提供快速、准确的指令仿真，以及与 RealView Debugger 的无缝连接。大型应用程序的开发可以使用 RTSM 模拟技术来完成。 RTSM 模拟包括 LCD 显示器、键盘和鼠标等外设的仿真。不到 5s，就可以利用 PC 在 ARM 处理器上对 OS 的启动过程进行模拟，用户可以在 ARM 提供的 RTSM 上进行快速的软件仿真。这是 OEM 在开发软件系统时成本最低的方法。想象一下，芯片公司不用等到芯片生产出来，也不用把缓慢的 FPGA 板交给方案厂商或 OEM；只需要将整个芯片的模型交付，下游厂家就可以尽早尽快地将软件方案开发完毕。最终产品几乎可以从芯片生产出来就准备上市。

## 1.2 器件选型

器件的选择归根结底是为嵌入式系统选择合适的处理器芯片。ARM 处理器是最常见的嵌入式处理器之一，它以低功耗、低成本和高性能而深受业界的青睐。而且 ARM 是目前产业中资源最为广泛的嵌入式处理器，基于广大的 ARM 合作伙伴计划，开发者可以在这个联盟里寻求到各种自己意想不到的帮助。从图 2 给出了常见的 ARM 处理器的架构和支持的操作系统。目前在业内广为人知的 ARM 处理器主要有 ARM7 系列和 ARM9 系列，同时为了关注今后嵌入式系统的发展，也有必要了解一下最新的 ARM11 和 ARM Cotex 系列处理器。

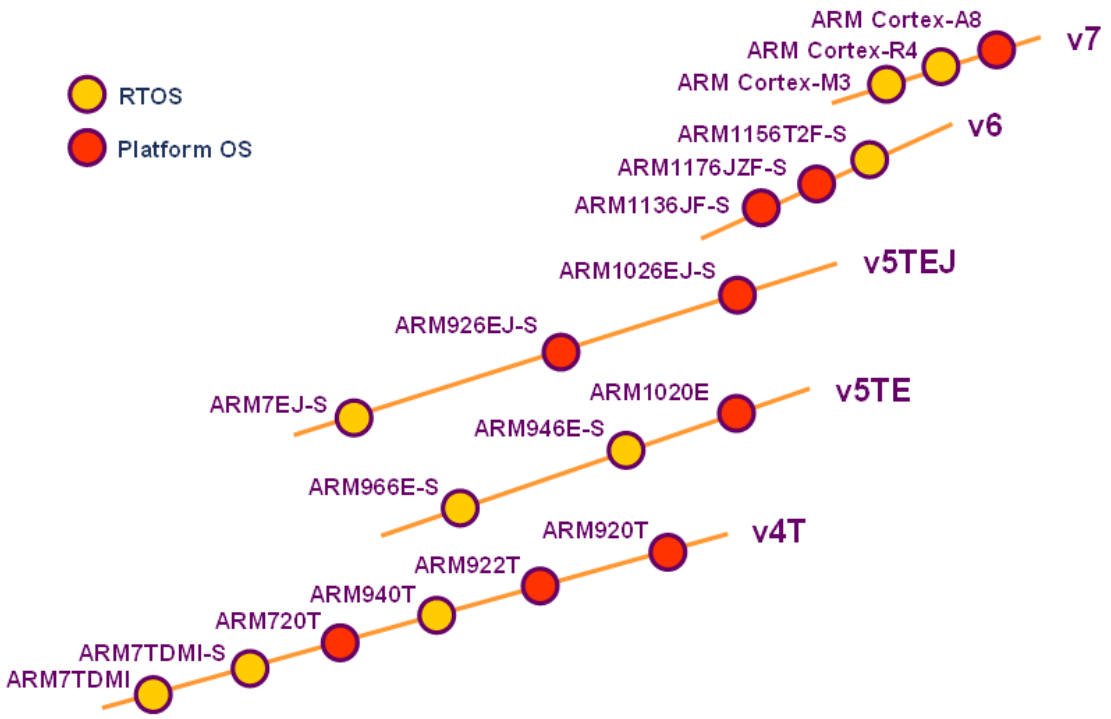


图 2 ARM 体系结构

**ARM7 系列**

ARM7TDMI 是 ARM7 系列中使用最广泛的，它是从最早实现 32 位地址空间编程模式的 ARM6 内核发展而来的，并增加了 64 位乘法指令，支持片上调试、16 位 Thumb 指令集和 EmbeddedICE 观察点硬件。ARM7TDMI 属于 ARM v4 体系结构，采用冯诺伊曼结构，3 级流水处理，平均 0.9DMIPs/Mhz 性能。不过 ARM7TDMI 没有 MMU（Memory Management Unit）和 Cache，所以仅支持那些不需要 MMU 和 Cahce 的小型实时操作系统，如 VxWorks、uC/OS-II 和 uLinux 等 RTOS。其他的 ARM7 系列内核还有 ARM720T 和 ARM7E-S 等。

**ARM9 系列**

ARM9TDMI 相比 ARM7TDMI，将流水级数提高到 5 级从而增加了处理器的

时钟频率，并使用指令和数据存储器分开的哈佛结构以改善 CPI 和提高处理器性能，平均可达 1.1DMIPs/Mhz，但是 ARM9TDMI 仍属于 ARM v4T 体系结构。在 ARM9TDMI 基础上又有 ARM920T、ARM940T 和 ARM922T ,其中 ARM940T 增加了 MPU（Memory Protect Unit）和 Cache；ARM920T 和 ARM922T 加入了 MMU、Cache 和 ETM9( 方便进行 CPU 实时 trace ),从而更好的支持象 Linux 和 WinCE 这样的多线程、多任务操作系统。

**ARM9E 系列**

ARM9E 系列属于 ARM v5TE ,在 ARM9TDMI 的基础上增加了 DSP 扩展指令，是可综合内核，主要有 ARM968E-S、ARM966E-S、ARM946E-S 和 ARM926EJ-S（v5TEJ 指令体系，增加了 Java 指令扩展），其中 ARM926EJ-S 是最具代表性的。通过 DSP 和 Java 的指令扩展，可获得 70%的 DSP 处理能力和 8x 的 Java 处理性能提升。另外分开的指令和数据 Cache 结构进一步提升了软件性能；指令和数据 TCM（Tightly Couple Memory：紧耦合存储器）接口支持零等待访问存储器；双 AMBA AHB 总线接口等。ARM926EJ-S 可达 250Mhz 以上的处理速度，很好地支持 Symbian OS、Linux、Windows CE 和 Palm OS 等主流操作系统。

**ARM11 系列**

ARM11 系列主要有 ARM1136、ARM1156、ARM1176 和 ARM11 MP-Core 等，它们都是 v6 体系结构，相比 v5 系列增加了 SIMD 多媒体指令，获得 1.75x 多媒体处理能力的提升。另外，除了 ARM1136 外，其他的处理器都支持 AMBA 3.0-AXI 总线。ARM11 系列内核最高的处理速度可达 500Mhz 以上（其中 90nm

工艺下，ARM1176 可达到 750Mhz）以及 600DMIPS 的性能，请参考和图 3 相关描述。

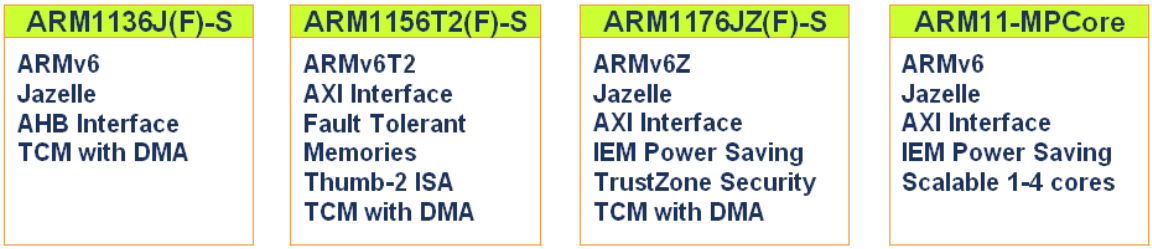


图 3 ARM11 系列内核

基于 ARMv6 架构的 ARM11 系列处理器是根据下一代的消费类电子、无线设备、网络应用和汽车电子产品等需求而制定的。其的媒体处理能力和低功耗特点使它特别适合于无线和消费类电子产品；其高数据吞吐量和高性能的结合非常适合网络处理应用；另外，在实时性能和浮点处理等方面 ARM11 可以满足汽车电子应用的需求。

**ARM Cotex 系列**

Cortex 系列是 ARM 公司目前最新内核系列，属于 v7 架构，主要有 Cortex-A8、Cortex-R4、Cortex-M3 和 Cortex-M1 等处理器，其中 A8 是面向高性能的应用处理器，最高可达 1Ghz 的处理速度，更好的支持多媒体及其他高性能要求，最高可达 2000DMIPS；R4 主要面向嵌入式实时应用领域（Real-Time），7 级流水结构，相对于上代 ARM1156 内核，R4 在性能、功耗和面积（PPA：Performance，Power and Area）取得更好的平衡，>1.5DMIPS/Mhz 和高于 400Mhz 的处理速度。而 M3 主要是面向低成本和高性能的 MCU 应用领域，相比 ARM7TDMI，M3 面积更小，功耗更低，性能更高。Cortex-M3 处理器的核心

是基于哈佛架构的 3 级流水线内核，该内核集成了分支预测，单周期乘法，硬件除法等众多功能强大的特性，使其在 Dhrystone benchmark 上具有出色的表现（1.25 DMIPS/MHz）。根据 Dhrystone benchmark 的测评结果，采用新的 Thumb®-2 指令集架构的 Cortex-M3 处理器，与执行 Thumb 指令的 ARM7TDMI-S®处理器相比，每兆赫的效率提高了 70%，与执行 ARM 指令的 ARM7TDMI-S 处理器相比，效率提高了 35%。

目前已经有 Cortex 系列内嵌的产品问世，如 TI 公司推出的基于 Cortex-A8 内核的 OMAP3430，TI、ST 和 Luminary 也推出了基于 Cortex-M3 内核的低成本高性能 32 位 MCU，更多详情请登陆这些公司的主页查询。

## 2 工具选择

根据开发目标平台的不同，ARM 提供不同的工具解决方案。

### MDK-ARM

RealView Microcontroller Development Kit(MDK) 支持基于 ARM7，ARM9，Cortex-M3 微控制处理器，例如 Atmel，Freescale，Luminary，NXP，OKI，Samsung，Sharp，ST，TI 等厂家的产品。MDK 提供工业标准的编译工具和强大的调试支持。MDK 是专为 MCU 的用户开发嵌入式软件而设计的一套开发工具。包括根据器件定制的调试仿真支持，丰富的项目模版，固件示例以及为内存优化的 RTOS 库。MDK 上手容易，功能强大，适合微控制器应用程序开发。

### RVDS

正如前面所介绍 RVDS 是专为 SOC，FPGA 以及 ASIC 用户开发复杂嵌入式应用程序或者和操作系统平台组件接口而设计的开发工具。RVDS 支持器件设计，支持多核调试，支持基于所有 ARM 和 Cortex 系列 CPU 的程序开发。RVDS 还可以和第三方软件进行很好的连接。

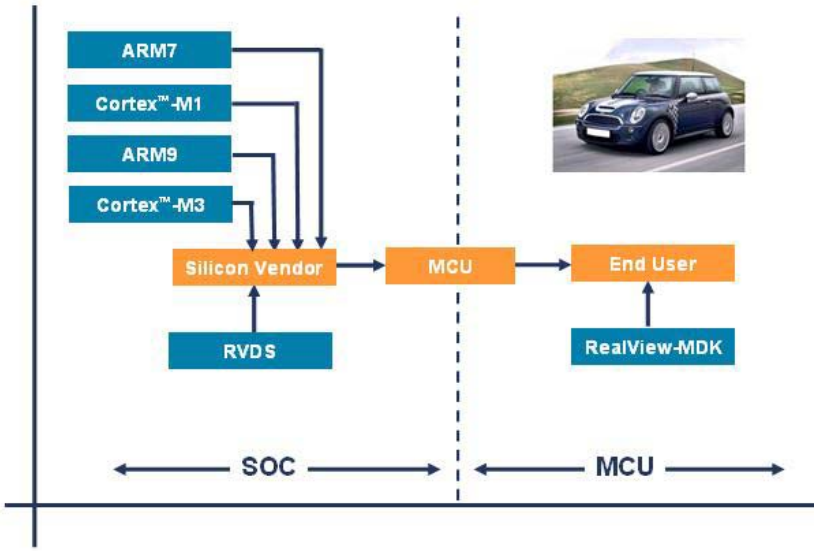


图 4 RVDS 和 RV-MDK

如上图表示：MDK 主要是为终端客户提供价格低廉，功能强大的开发工具。集成了 RealView 编译工具，Keil uVision 开发环境，支持基于 ARM7,ARM9,Cortex-M1,Cortex-M3 产品的仿真，提供非常高效的 RTOS Kernel，除此，提供的 Real-Time 库还有 TCP/IP 网络套件，Flash 文件系统，USB 器件接口，CAN 总线接口等，方便终端用户进行应用开发。因此对于 MDK 用户来说，他们得到的就是可以对 MCU 进行仿真和调试，容易使用又没有冗余的功能，关键是价格实惠，而且用户可以先试用再购买。

对于芯片设计公司以及相关解决方案提供商来说，需要的是更加强大的工具，可以进行多核调试，需要更加先进的调试和分析功能，可以支持多种操作系

统，可以进行 IP 整合开发，可以结合 ESL 工具进行架构评估，系统软硬件划分等，那么选择 RVDS 可以提供完整解决方案。

## 3 编译和连接

ARM RealView 编译工具已经发展了 16 年，一直致力于为客户提供最好的编译器。RVDS 是 ARM 公司继 SDT 与 ADS1.2 之后主推的新一代开发工具，目前最高版本是 3.1。它由 RealView 编译器（RVCT）、RealView 汇编器（armasm）、RealView 连接器（armlinker），以及 RealView 调试器（RVDebugger）三部分组成。

RVDS 对代码密度的提升、代码执行速度的提高，都可以由 ARM 开发工具自动实现，而不需要软件开发人员花费过多的时间手动优化高级语言代码。这是 RVDS 的优势所在。

先前版本中的编译器 armcc，tcc，armcpp，tcpp 已经整合成一个编译器 armcc，可以将标准的 C 或 C++语言源程序编译成 32 位 ARM 指令代码或者 16 位 Thumb 指令代码或者 Thumb-2 指令代码。

编译器输出的 ELF 格式的目标文件，包含调试信息。除此之外，编译器可以输出所生成的汇编语言列表文件。

RVDS 的编译器根据最新的 ARM 架构进行特别的优化，针对每个 ARM 架构都提供最好的代码执行性能，最优的代码密度。可以根据需要选择调试信息级别，以及不同的代码优化方向和优化级别。

RVCT 中 C 和 RogueWaveC++库包括

- 完整 ISO 标准 C 语言库

  标准 C 语言函数集，C 语言库需要的支持函数以及在 Semihosted 执行环境中需要的目标相关的函数。

  ARM C 语言库结构使用户很容易定义目标相关函数，以适应特定的目标环境。

- 浮点函数库使用 ARM 在 IEEE754 标准（二进制浮点算法）上实现的浮点环境。

- RogueWaveC++库

  RogueWaveC++库包含标准 C++函数，编译器需要的支持函数。

各种源文件经过 ARM 编译器编译后生成 ELF 格式的目标文件。这些目标文件和相应的 C/C++运行时库经过 ARM 连接器处理后，生成 ELF 格式映像文件。

ARM 连接器可以去除使用不到的代码段和函数，这样可以减少内存的使用。

ARM 连接器可以将不同的指令代码和数据代码放置到不同的内存地址范围。（http://www.arm.com/support/faqdev/1245.html）

通常在嵌入式系统中，指令和数据代码会固化在非易失性存储器中（ROM 或 Flash），可以从这些地方上电启动。从运行速度方面考虑，部分指令和数据代码会在启动后搬运到易失性存储器(RAM)中，因此连接器可以使用一些方法机制来配置调度。

这种分散装载（scatterloading）的机制可以让把不同的指令和数据分散的放到不同的地址，而且这些地址在系统启动和系统运行可以是不同的映射。

详细的地址分配可以是用参数来指定，或者用一个描述文件来作为连接器的参数。使用描述文件会使维护起来非常简单，而且如果要改变地址分配，不需要把整个项目完全重新来做，只要把项目中需要的目标重新连接即可。

一个 scatterloading 文件的示例：

```
LOAD_FLASH 0x04000000 0x80000          ; 启动地址和长度
{
    EXE_FLASH 0x04000000 0x80000
    {
        init.o (Init, +First)          ;
        * (+RO)                        ;
    }
    32bitRAM 0x0000 0x2000
    {
        vectors.o (Vect, +First)       ;
        int_handler.o (+RO)
    }
    16bitRAM 0x2000 0x80000
    {
        * (+RW,+ZI)                    ;
    }
}
```

- 本文件定义了启动区域和三个执行区域。在大括号外面定义了启动区域（LOAD_FLASH），里面三个定义了执行区域(EXEC_FLASH,32bitRAM,16bitRAM)。
- 为了提高运行速度，异常向量（在 vectors.s）和异常句柄（在 int_handler.c）被重新放置到 32bitRAM 的零地址开始的地方。
- 可以读写的变量被复制到 16bitRAM 的 0x2000 地址开始的地方。
- 零初始化的数据和可读写数据放在 16bitRAM 内。
- 其他不需要搬运的代码只需要还放在 Flash 里就好。

### 3.1 RVCT 的优化级别与优化方向

提到 RVCT 就不能不提 armcc 的四个优化级别和两个编译选项，-O1、-O2、-O3、-O4，以及-Otime、-Ospace。

-Ospace 与-Otime 负责给编译器提供代码优化的大方向，告知编译器编译任务的主要目标是代码密度（-Ospace）还是代码性能（-Otime）。而-O1、-O2、-O3、-O4 则分别代表 4 种逐次递进的不同优化级别。

#### OSpace 还是 OTime？

显然代码密度与代码执行速度在很多情况下是一对矛盾。以下面的代码为例。例 1 中左右两段代码可以完成相同的任务，但是左边的有较高的代码密度，右边的则有较高的执行速度。因为当 expr = 0 时，标志循环结束时，右边的代码可以顺序执行下去；而左边代码必须先跳转至循环体首部判断 expr 的值，随

后再跳转道循环体尾，继续执行下一条指令。

***例 1*** *代码速度与尺寸的对比*

```
while (expr)
{
    body;

}
```

```
if (expr) do
{   do
        { body; }
    while (expr);
}
```

那么我们什么时候使用 Otime 什么时候使用 Ospace 呢？Otime 与 Ospace 需要开发人员根据系统实际需求来决定，最好的情况是在两者之间找到一个合适的平衡点，而不是单纯的追求速度或者代码尺寸的缩小。即，将不同的代码模块根据其特性分别使用不同的编译选项。

此外，RVCT 编译器支持很多非常有用的编译选项，如--no_inline（取消所有代码的内联函数）、--split_ldm（限制 LDM/STM 指令的最大操作寄存器数目）、--split_sections（将每个函数，而不是源文件，作为一个编译单元进行操作）等等。

编译器的所有这一切都可以严格根据开发者的要求，帮助开发人员得到系统真正需要的优化过了的代码。

## O3 还是 O2？

老的开发工具，如 ADS1.2 中，只有 3 种递进的代码优化级别，对应 3 种编译选项，即-O0（Minimum optimization）、-O1（Restricted optimization）、-O2（High optimization ）。

使用-O0 编译选项时，RVCT 编译器只对代码作最基本的优化操作，编译结束后用户得到的代码与用户手写源代码之间的差距很小 ,这种特性的主要作用是为了方便用户在程序开发阶段的调试工作 , 避免由于优化而产生的调试屏障。此外 , 很多资深软件工程师偏向于手写优化代码 , 在这种情况下 , 由于代码已经被优化过，可以使用-O0 编译选项减少 RVCT 的工作量，节省编译链接的时间。

-O1 与-O2 则分别是相对于-O0 更加高级别的编译优化选项，前者提供有限的优化；后者则会对代码进行较大程度的优化改进操作。

RVDS 中新增加了-O3（Maximum optimization）编译选项，它可以最大程度的发挥 RVCT 编译器的优势，将代码编译成最优。O3 与 O2 都是较高级别的编译优化选项，但-O3 相比较于-O2，主要优势有以下几点。当用户使用-O3 选项时：

––编译器会自动对代码进行高阶标量优化。所谓的高阶标量优化就是编译器对根据代码特点，针对循环、指针等进行高阶优化。

––编译器会把尽可能多函数的编译为内联（inline）函数；

––Multifile compilation 功能被自动使能。

**对于循环与指针的高阶优化（High-level scalar optimizations）**

当编译选项为-O3 –Otime 时，RVCT 会根据代码的具体情况，针对循环、

指针等部分作高阶优化工作：循环解开（Loop unrolling）、融合（fusion）、位置调整（interchange）、指针优化等等。以例 2 的函数为例。例 2 是一段简单的 C 循环函数，在循环中含有数组指针调用。

***例 2***

***CodeA***

```
void increment(int *restrict b,    int *restrict c)
{        int i;
          for (i = 0; i < 100; i++)
       {
            c[i] = b[i] + 1;
       }
}
```

***CodeB***

```
void increment(int *b, int *c)
{
int i;
int *pb, *pc;
int b3, b4;
pb = b - 1;
pc = c - 1;
b3 = pb[1];
```

```
for (i = (100 / 2); i != 0; i--)
    {
        b4 = *(pb += 2);
        pc[1] = b3 + 1;
        b3 = pb[1];
        *(pc += 2) = b4 + 1;
    }
}
```

仔细观察可以发现，CodeA 与 CodeB 可以完成同样的功能，即将数组 b 的每个成员加 1 赋值给数组 c 对应成员。但是 CodeB 与 CodeA 相比，有较高的执行速度。主要体现在以下几点：

––循环 100 次变成了循环 50 次（loop unrolling），减少了跳转次数；

––数组变成了指针，减少每次计算数组偏移量的指令；

––微调了不同代码操作的执行顺序，减少了流水线 stall 的情况；

––循环从 + + 循环变成了––循环。这样可以使用 ARM 指令的条件位，为每次循环减少了一条判断指令。

很多程序员就是这样，通过这种手写不同的 C 代码，再实现相同任务的情况下，提高了代码执行效率。

在 RVDS 中，使用-O3 –Otime 编译选项，RVCT 会自动帮助程序员进行这些高阶标量优化，即，RVCT 会直接将 CodeA 优化成以前由 CodeB 才能得到的汇编代码。虽然优化之后函数的代码尺寸大于原先的函数，但是执行速度却有大

大的提高，

经过统计，使用 EEMBC benchmarking， -O3 编译选项编译得到的最终代码平均性能相对于-O1 可以有 10%的提升，而总体代码尺寸只增加了 1%。

### 3.2 Multifile compilation

按照传统的编译方式，我们先把各个 C 或 C++文件单独编译成.obj 文件，再将这些目标文件链接在一起。考虑到虽然在编译单独的 C 或 C++文件时，编译器会充分发挥它的优化特性；但此时，编译器无法关注到大量的 C 或 C++文件接口之间可以优化的部分。所以在传统的编译结果里，还有许多优化的余地。如何才能让编译器同时关注和编译所有的源代码呢？

Multifile compilation 是 RVDS 一个较新的特性，它可以帮助开发人员将所有的源文件作为一个 compilation unit 进行编译，并最终生成一个大的目标文件( 如图 3 中的 file1.o )。Mutifile compilation 给软件开发人员带来的直接优势有以下几点：

––增加 inline 的可能性。由于 inline 只能发生在一个 compilation unit 中，所以在没有使用 mutifile compilation 时，inline 只能发生在一个源文件范围内。Multifile Compilation 将一个 compilation unit 扩大到了所有源文件的范围上，所以直接增加了 inline 发生的几率。

––增加了基地址与函数间优化的可能性。同 inline 一样，所有的基地址与函数间的优化也必须在一个 compilation unit 中，随着 conpilation unit 的扩大这种优化的可能性也增加了。

--减少了 scatter file 的复杂性。

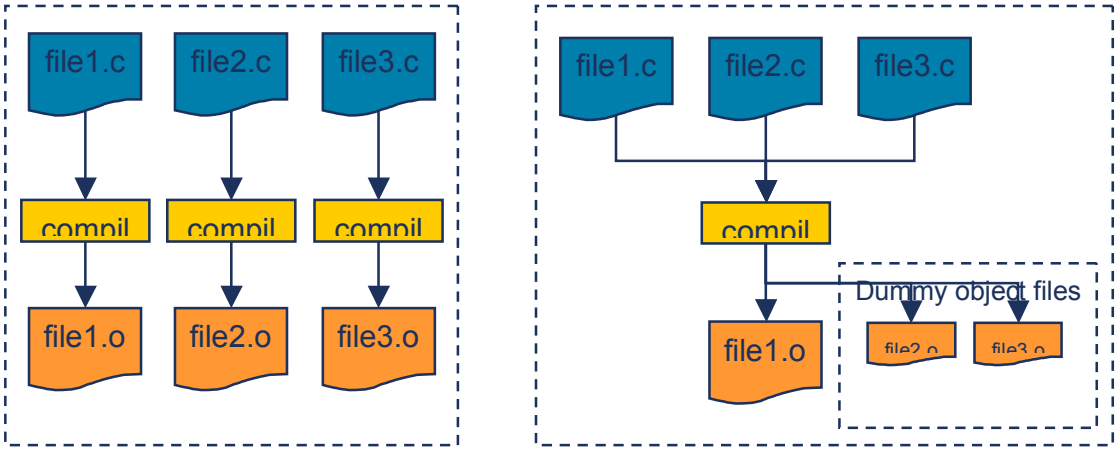


图 3　Mutifile compilation 工作原理

### 3.3 调试

由前面的介绍已经知道 RealView Debugger（RVD） 是 RVDS 的重要组件之一。RealView Debugger 可以更好的帮助客户在复杂 SoC 设计中方便直观的调试软硬件。

- 方便协调的开发软硬件

RVD 可以和 RealView SoC Designer 一起调试，这样软硬件开发人员可以更早的进行合作开发，而且双方都是使用各自熟悉的工具，这样可以有效地缩短开发周期。

- 单核或多核调试

RVD 使用同步机制进行多核调试，使用 RVD 在一个处理器上设置的断点，可以停止整个系统，这样可以观测复杂的多核系统的各种关键状态信息。

- 调试操作系统和中间件

RVD 支持调试业内各种流行的操作系统，正如操作系统一章所述，RVD 可以直观的观测操作系统的执行文本和各种资源。

- 调试跟踪，性能评估

RVD 可以对基于 ARM 处理器的设计进行非插入式的实时地捕捉数据/指令和显示，从而实现调试，跟踪以及性能评估。目前业内其他的调试工具还不能达到有如 RVD 这样出色的性能。RVD 可以对 RealView ESL 对系统模型进行调试，也可以使用 RealView ICE 以及 RealView Trace 对真实的硬件系统进行调试跟踪和性能评估。

- 调试目标设备

无论是开发一个新的软硬件架构，一个操作系统还是一个应用程序，RVD 可以连接到 SoC 模型，指令集仿真模型，实时系统模型或者真实的硬件处理器来帮助完成开发。如此广泛的支持，使得 RVD 在整个开发周期中成为一个不可或缺的得力的开发工具。

## 4 操作系统

随着高端电子消费类产品的广泛普及，实时嵌入式操作系统使用越来越广泛。而基于 ARM 的嵌入式操作系统在各个领域都得到了广泛的应用，利用 ARM 系列产品的强大功能可以完成各种应用程序的开发。

ARM 对操作系统以及系统开发执行环境提供最广泛的选择，客户可以根据需要来选择最适应市场要求的基于 ARM 的嵌入式操作系统。可供选择的嵌入式

操作系统有几十种，使用较多的有 Linux,WinCE,Palm,Symbian 等等。采用 WinCE 更多的是 OEM，以及按需进行特定的嵌入式器件开发的，例如 GPS 导航设备。采用 Palm 操作系统的厂家有联想 三星 索尼，他们的出货量都非常巨大。Symbian 操作系统是先进的全球公开工业标准操作系统，基于 Symbian 操作系统的手机有：BenQ，DoCoMo，Motorola，Nokia，Panasonic，三星，索尼爱立信等。Linux 是源代码开放的操作系统，可以运行在包括 ARM 等多种主流处理器架构上。由于有一大批的工程师在开发开放源代码以及相关开发工具，Linux 可以更方便快捷的进行移植。

以 Linux 为例 ,选择基于 ARM 的 Linux ,可以得到更多的开发源代码的应用，可以利用 ARM 处理器的高性能开发出更广阔的网络和无线应用 ,ARM 的 Jazelle 技术带来 Linux 平台下 Java 程序更好的性能表现。ARM 公司的系列开发工具和开发板，以及各种开发论坛的可利用信息带来更快的产品上市时间。

## 4.1 哪里可以得到 os 软件包 (Open Source and Linux Kernel)

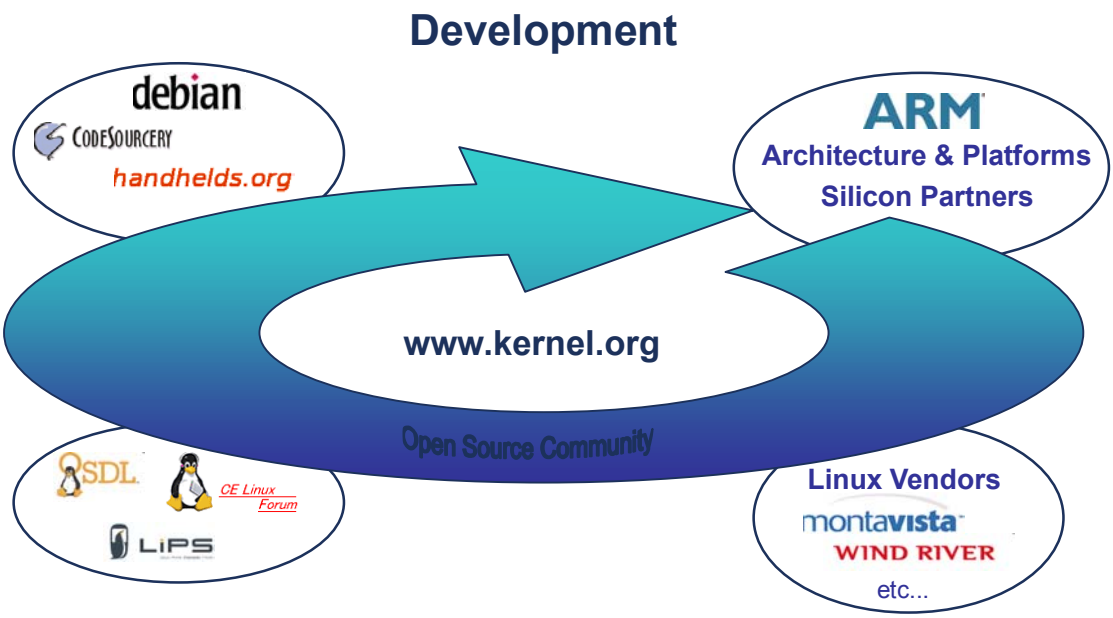


图 5

ARM 网站上可以下载到基于 RealView Itegrator 和 RealView Versatile 平台的 Linux Kernel 镜像文件，补丁以及实用工具。我们都知道 Linux 是需要进行虚拟地址管理的，因此需要处理器整合有 MMU。并不是所有的处理器都整合有 MMU，因此可以在不具备 MMU 的处理器上运行的修改过的 Linux 又叫做 uClinux，uClinux 同样可以从 ARM 网站上下载。

可以以基于 ARM926 的 Linux 开发为例，浏览一下整个开发流程：

需要的相关软件：

- Boot Loader：U-Boot。
- 预编译的 Linux kenel，包括源文件和镜像文件。

- 配置文件。
- 文件系统以及预编译的实用工具和应用软件。

首先可以从 http://www.arm.com/linux/linux_download.html 下载 U-Boot 和 Linux 的镜像文件，相关开发平台有 RealView Integrator-CP 平台，Versatile PB/AB 平台，EB 平台。

## 4.2 安装镜像

可以使用与主板相连的调试器，通过 JTAG 运行控制设备（如 ARM RealView ICE 单元）将映像安装到闪存中。

以 PB926EJ-S 为例，使用从调试器运行主板随附的 BootMonitor.axf 程序来编写映像。Flash 菜单提供一些编写 ELF 或二进制映像的选项。RealView Versatile 系列还提供一个闪存编程实用工具，即“网络闪存实用工具”（NFU）。该程序可以通过以太网连接将映像编写到闪存中。

使用 U-Boot 完成对闪存中的镜像引导。可以通过 U-Boot 完成对 Linux 的一些设置以及将一些环境保存到闪存中。

Linux Kernel 可以从 www.kernel.org 上得到。

## 4.3 交叉编译

主机系统上必需安装 ARM 交叉编译工具链来生成 Linux 内核或应用程序。（系统中需安装 Glibc library 2.3 或更高版本）。

可从 www.codesourcery.com 或 GCC CVS 主存储库下载用于构建

GCC 工具的源文件。该网站还提供讨论组，供用户讨论 ARM GNU 工具的相关技术问题

开发人员可以从 www.codesourcery.com 下载用于构建 GCC 工具的源文件。该网站还提供讨论组，供用户讨论 ARM GNU 工具的相关技术问题。

*arm-elf* 生成与任何操作系统无关的平坦或独立二进制文件。arm-elf 会选择 ELF 支持，其大部分代码与 arm-linux 共享。

*arm-none-linux-gnueabi* 是 Linux 所需的目标，它生成 Linux/ARM 的 ELF 支持。

ARM RealView 开发工具

**总结**

本文概略的介绍了 ARM 公司的开发工具，如何选择您需要的器件，从一个 SoC 设计者或者软件开发者的角度如何选择开发工具，RealView 开发工具中的编译工具、连接工具以及调试工具。简单描述了开发操作系统以及相关应用程序的步骤。

无论是硬件系统开发还是软件开发，调试都是最重要也是需要时间最多的环节，软硬件开发人员的沟通也是缩短产品上市时间的关键。强大的 RealView Debugger 成为不同的开发团队之间沟通最好媒介。RealView 开发工具使得调试环境保持一致，软硬件开发人员更加协调。而后续开发人员或者第三方开发者可以更好继承这一调试环境，包括操作系统以及应用程序库，配以 RealView 系列工具的支持，可以更迅速的完成新的设计。

# 电子工程专辑优秀电子书

- 电子工程师必备手册（上）--GPS 设计全攻略**(HOT)**
- 电子工程师必备手册（下）--运算放大器设计与应用**(HOT)**
- TD-SCDMA 射频系统发射部分性能监测方法**(HOT)**
- LTC 四输出负载点 DC/DC μModule 系统设计要点
- D 类音频放大器设计：概念、原理和方法
- 车用电容传感器的新契机
- LCDTV(欧洲机型)AV 输入设计
- PolySwitch LVR/LVRL 器件有助于家用和专业电器的电机保护
- OFDM 原理
- 视觉人体工程学设计要点