MCU

AVR-GCC与AVR单片机C语言开发

吴双力 崔 剑 王伯岭 编著

北京航空航天大学出版社

AVR-GCC 与 AVR 单片机 C 语言开发

吴双力　崔　剑　王伯岭　编著

北京航空航天大学出版社

内 容 简 介

本书以 AVR-GCC 的 Windows 版本 WinAVR20040404 为例，介绍使用 AVR-GCC 开发 AVR 单片机的方法。首先介绍 AVR 单片机的特点，使读者对 AVR 单片机有整体的印象。随后简单地介绍了 C 语言的语法和 AVR-GCC 的函数库 avr-libc 的常用库函数及定义。接下来以 ATmega16 单片机为例，介绍了 AVR 单片机常用部件的操作方法。最后，介绍了运行于 AVR 单片机上的实时操作系统 AVRX 的使用方法和 AVR 单片机的调试方法，以供更高层次的读者参考。

本书适合于有一定单片机基础或者了解一定 C 语言知识的单片机爱好者、工程技术人员和大专院校的学生学习 AVR 单片机的 C 语言开发之用。

本书附光盘 1 张，包括书中讲述的 WinAVR 工具套件 AVRStudio，VMLAB 等软件，以及 Unix 平台下开发需要的相关软件和资料。

图书在版编目(CIP)数据

AVR-GCC 与 AVR 单片机 C 语言开发/吴双力等编著.
北京：北京航空航天大学出版社，2004.10
ISBN 7-81077-513-8

Ⅰ.A... Ⅱ.吴... Ⅲ.①单片微型计算机—程序设计②C 语言—程序设计 Ⅳ.①TP368.1②TP312

中国版本图书馆 CIP 数据核字(2004)第 098582 号

AVR-GCC 与 AVR 单片机 C 语言开发

吴双力 崔 剑 王伯岭 编著

责任编辑 王慕冰

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083) 发行部电话：(010)82317024 传真：(010)82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：17 字数：435 千字

2004 年 10 月第 1 版 2004 年 10 月第 1 次印刷 印数：5 000 册

ISBN 7-81077-513-8 定价：28.00 元(含光盘 1 张)

前　言

AVR单片机是ATMEL公司出品的新一代8位单片机，该单片机采用高性能的RISC内核，具有很低的功耗。AVR单片机的内核以时钟振荡器的振荡频率运行，而且绝大部分指令为单周期指令，因此AVR单片机具有非常高的指令执行速度，可达到接近1 MIPS/MHz的性能，运行速度比绝大部分单片机都要高。AVR单片机内部为高级语言进行了优化，用高级语言编写的程序可高效率地生成执行代码。AVR单片机片内集成了大容量的Flash作为程序存储器，可方便地进行改写。AVR单片机在片内集成了EEPROM存储器，可作为数据存储器，避免外接EEPROM存储器的不便。AVR单片机支持ISP功能，部分型号还支持IAP功能，提高了单片机开发的灵活性。AVR单片机很多型号具有可选择的内部振荡器，在要求不高时可代替石英晶体。AVR单片机片内集成了看门狗定时器，可防止程序在运行中跑飞。AVR单片机内部集成了多种外部设备，除了常见的定时器、捕获器、串行接口(UART或USART)，很多型号还集成了TWI(兼容于I^2C接口)、模拟比较器、低电压复位保护、ADC和PWM控制器；在新型号中，ATEML公司还将USB控制器、射频收发电路等集成入AVR单片机。因此，AVR单片机已经不仅仅是一个用于控制的8位单片机，在有些场合甚至可单独组成一个片上系统(SoC)，完成复杂的功能。

AVR单片机进入国内市场以来，从开始大家都不了解到现在很多人开始尝试使用它，除了该单片机本身性能上的一系列优点外，还与该单片机在开发工具上的便利和编译器的良好支持密不可分。到目前为止，AVR单片机上不仅可使用多种开发工具进行开发，而且包括了烧写器、仿真器、调试器、汇编器、高级语言的编译器、集成开发环境等全套的开发工具链。特别值得一提的是，ATMEL公司出品的免费软件AVR Studio可方便地增加外部编译器等工具，构成一个完整而廉价的集成开发环境，显示了ATMEL公司对AVR单片机强大的支持。

用高级语言开发单片机具有一系列的优点，它可以使开发人员专注于算法本身，而不是只关心算法的实现细节。高级语言最接近于自然语言，易于理解和记忆。高级语言可方便地在不同系统中进行移植，源代码可不变或只做少量修改。对于AVR这种为高级语言进行过优化设计的单片机，高级语言生成的代码并不比汇编语言生成的代码多占用许多程序空间，执行速度也不会有大幅度的降低。相反却提高了开发的速度，减少了开发的风险，因此，在很多情况下使用高级语言开发单片机是非常有利的。

GCC编译器最初是由Richard Stallman编写并且现在被广泛使用的C编译器，该编译器本来是GNU项目的一个组成部分。由于该编译器是自由软件，任何人都可以对其修改和传播，一部分程序员编写了GCC的函数库avr-libc，并将其移植到AVR单片机上。用于开发AVR单片机的GCC编译器称为AVR-GCC(也称为gcc-avr)。AVR-GCC可自由地获得，而且并不强制收费。目前AVR-GCC可运行在多种主流的操作系统上，包括GNU/Linux，Windows，Mac OS X，Freebsd等，因此使用各种不同系统的开发人员都可以开发AVR单片机。AVR-GCC支持绝大部分AVR单片机，而且受支持的单片机数目正在不断地扩展。可以说，常用的AVR单片机都可以使用AVR-GCC进行开发。AVR-GCC的更新非常迅速，不

断有补丁问世。

笔者在一个偶然的机会接触了 AVR 单片机，其高速、低功耗和开发方便的优点给笔者留下了很深的印象。在随后的使用过程中，积累了一些经验，并且介绍该单片机给身边的人使用。在这个过程中，笔者看到身边的很多人对 AVR 单片机不很了解，甚至有一些误解。他们在学习 AVR 单片机时感到很迷茫，于是觉得有必要写一本完整介绍 AVR 单片机开发的书籍，使更多的人了解 AVR 单片机和 AVR-GCC 软件。在写书的过程中，笔者尽力做到使该书浅显易懂，并对很多初学者会遇到的问题进行了详细讲解，希望以自己的实际经历引导学习 AVR 单片机的初学者少走一些弯路。本书在后半部分给出了大量的实例，这些都是笔者曾经做过的代码。希望初学者能够仔细研究这些代码，因为这对于理解 AVR 单片机的编程方法是十分有益的。

本书在编写过程中得到了广州双龙公司耿德根先生和北京航空航天大学出版社马广云老师的大力支持。另外，杨开老师提供了必要的实验场地及仪器，刘天祥先生提供了一套 USB JTAG 工具，北京航空航天大学电子信息工程学院的同学提出了许多宝贵意见，在此一并表示衷心的感谢。

由于编者水平有限，书中难免有不妥甚至错误之处，希望读者批评指正。另外，也可在网站上直接提出建议和意见。网址是：http://www.foravr.net。

考虑到国内的情况，本书未收集 AVR-GCC 在其他平台包括 Linux，Mac OS X，FreeBSD 系统上的开发方法，关于这方面的内容请读者参考附录给出的信息自行学习。

编　者

2004/08/20

于北京航空航天大学

目　录

第 1 章　AVR 基本知识

1.1　AVR 各系列单片机简介

AVR 单片机是 ATMEL 公司 1997 年推出的精简指令集(RISC)单片机系列。ATMEL 公司通过 AVR 把 RISC 技术带到了 8 位单片机世界，这种全新的结构带来了很多优势。该系列的程序存储器是在片内的 Flash 存储器，可以反复修改上千次。这对新产品开发、产品升级都是很方便的。单片机的指令基本上都是单个晶振周期的，能够达到 1 MIPS/MHz 的性能。该系列单片机针对应用 C 语言编程做了优化；很多型号都是宽电压工作的，同时各种睡眠模式有利于降低系统功耗；再加上内部的振荡器、看门狗、上电复位、A/D 输入、PWM 输出等功能，使其也可以被称为“零外设”的单片机，具有片上系统(SoC，System on Chip)的雏形。因此，AVR 单片机适合于很多领域的应用，表现出卓越的性能。

AVR 单片机家族已经发展成为一个全系列：包括 Tiny AVR，Mega AVR，LCD AVR，USB AVR，DVD AVR，RF AVR，Secure AVR，FPGA AVR 等类别。Tiny AVR 系列的典型芯片如 Tiny11，Tiny12 和 Tiny13 等，该类型单片机的特点是把价格、性能和灵活性很好地结合在一起，典型的应用包括锂电池充电器、冰箱控制和门禁系统等。Mega AVR 系列的典型芯片如 ATmega8，ATmega16 等，该类型单片机的特点是带有具自编程能力的程序存储器，可以通过 SPI、USART 和二线制接口(I^2C)编程，适合于需要远程编程和现场升级的应用领域；同时该类型单片机具有很全的外围设备，适合于多种应用。另外，还有一些增加了面向特殊应用，具有特殊功能的单片机。这些单片机都是在相同的 AVR 的基础上增加了面向应用的特殊功能。例如，LCD AVR 单片机如 ATmega169，集成了 LCD 驱动器，能够驱动 4×25 段的 LCD；USB AVR 单片机如 AT43USB351M，集成了 USB 的物理层和数据链路层的硬件协议，同时由 AVR 核通过编程实现传输层的实现；DVD AVR 单片机如 AT78C1501，其内部通过 AVR 核实现内部数据通道核缓存的控制；RF AVR 单片机如 AT86F401，在 AVR 核的控制下实现开关键控的无线射频数据传输；Secure AVR 单片机如 AT90SC19264RC，是带有 AVR 核的、实现 ISO7816 协议的、用于智能卡的单片机。FPGA AVR 单片机如 AT94K05A，其内部集成有 FPGA。这些类型构成了 AVR 系列单片机的庞大家族，使 AVR 在众多应用领域发挥独特的性能。

尽管 AVR 系列单片机型号繁多，功能各异，但所有 AVR 单片机都有相同的存储器结构和指令集，因此各系列 AVR 单片机之间的代码移植是很方便的。

不同系列单片机分别具有配置不同的 SRAM、EEPROM、外部 SRAM 的接口、A/D 转换器、硬件乘法器、UART 及 USART 等外围设备。如果把 Tiny AVR 或 Mega AVR 的所有外围设备全都去除，那么所有系列的 AVR 单片机将只剩下 AVR 核，而所有 AVR 单片机全都有相同的 AVR 核。外围设备是在 AVR 核的控制下工作的，数据的处理运算也是由 AVR 核进行的，因此了解它的工作原理，将会对应用有很大帮助。在这里不对各种外围设备进行详细

的介绍，只简单介绍一下AVR核。外围设备的使用会在后续章节逐一介绍，并附上程序。更详细的原理可以在AVR单片机的数据手册上得到。

AVR单片机非常突出的特点和其优越性在很大程度上与它选择了全新的RISC结构有关。AVR核的核心是快速访问的RISC寄存器文件，包括32×8位通用工作寄存器。在一个时钟周期内，AVR核可把寄存器文件中的任意两个寄存器送给ALU(算术逻辑单元)，完成所期望的操作，并把结果写到任意一个寄存器。ALU支持寄存器之间、寄存器与常数之间的算术和逻辑运算。单寄存器的操作也是ALU执行的。这一点与传统的单片机有很大区别。传统的单片机一般只有一个累加器，所有算术操作都要通过这个累加器进行，这往往成为整个系统的瓶颈。

图1.1是一个AVR MCU的结构框图。从图中可以看出，AVR采用Harvard(哈佛)结构——数据存储空间和程序存储空间是分开的。程序空间通过一级流水线进行访问。一个指令正在执行的同时，下一个指令已经提前从程序存储器中取出。

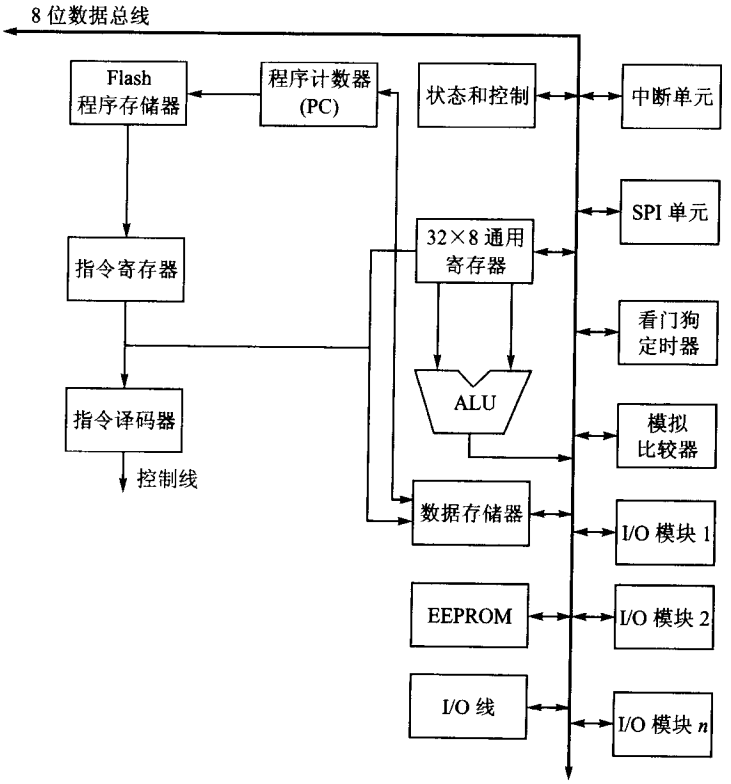

图1.1 AVR MCU体系结构框图

由于算术运算和逻辑运算是真正单周期的，AVR单片机可达到1 MIPS/MHz的性能，这使得设计人员可以在高速处理的同时降低系统的功耗。

第 1 章　AVR 基本知识

1.1　AVR 各系列单片机简介

AVR 单片机是 ATMEL 公司 1997 年推出的精简指令集(RISC)单片机系列。ATMEL 公司通过 AVR 把 RISC 技术带到了 8 位单片机世界，这种全新的结构带来了很多优势。该系列的程序存储器是在片内的 Flash 存储器，可以反复修改上千次。这对新产品开发、产品升级都是很方便的。单片机的指令基本上都是单个晶振周期的，能够达到 1 MIPS/MHz 的性能。该系列单片机针对应用 C 语言编程做了优化；很多型号都是宽电压工作的，同时各种睡眠模式有利于降低系统功耗；再加上内部的振荡器、看门狗、上电复位、A/D 输入、PWM 输出等功能，使其也可以被称为“零外设”的单片机，具有片上系统(SoC, System on Chip)的雏形。因此，AVR 单片机适合于很多领域的应用，表现出卓越的性能。

AVR 单片机家族已经发展成为一个全系列：包括 Tiny AVR，Mega AVR，LCD AVR，USB AVR，DVD AVR，RF AVR，Secure AVR，FPGA AVR 等类别。Tiny AVR 系列的典型芯片如 Tiny11，Tiny12 和 Tiny13 等，该类型单片机的特点是把价格、性能和灵活性很好地结合在一起，典型的应用包括锂电池充电器、冰箱控制和门禁系统等。Mega AVR 系列的典型芯片如 ATmega8，ATmega16 等，该类型单片机的特点是带有具自编程能力的程序存储器，可以通过 SPI、USART 和二线制接口(I^2C)编程，适合于需要远程编程和现场升级的应用领域；同时该类型单片机具有很全的外围设备，适合于多种应用。另外，还有一些增加了面向特殊应用，具有特殊功能的单片机。这些单片机都是在相同的 AVR 的基础上增加了面向应用的特殊功能。例如，LCD AVR 单片机如 ATmega169，集成了 LCD 驱动器，能够驱动 4×25 段的 LCD；USB AVR 单片机如 AT43USB351M，集成了 USB 的物理层和数据链路层的硬件协议，同时由 AVR 核通过编程实现传输层的实现；DVD AVR 单片机如 AT78C1501，其内部通过 AVR 核实现内部数据通道核缓存的控制；RF AVR 单片机如 AT86F401，在 AVR 核的控制下实现开关键控的无线射频数据传输；Secure AVR 单片机如 AT90SC19264RC，是带有 AVR 核的、实现 ISO7816 协议的、用于智能卡的单片机。FPGA AVR 单片机如 AT94K05A，其内部集成有 FPGA。这些类型构成了 AVR 系列单片机的庞大家族，使 AVR 在众多应用领域发挥独特的性能。

尽管 AVR 系列单片机型号繁多，功能各异，但所有 AVR 单片机都有相同的存储器结构和指令集，因此各系列 AVR 单片机之间的代码移植是很方便的。

不同系列单片机分别具有配置不同的 SRAM、EEPROM、外部 SRAM 的接口、A/D 转换器、硬件乘法器、UART 及 USART 等外围设备。如果把 Tiny AVR 或 Mega AVR 的所有外围设备全都去除，那么所有系列的 AVR 单片机将只剩下 AVR 核，而所有 AVR 单片机全都有相同的 AVR 核。外围设备是在 AVR 核的控制下工作的，数据的处理运算也是由 AVR 核进行的，因此了解它的工作原理，将会对应用有很大帮助。在这里不对各种外围设备进行详细

态变量需要驻留于数据存储器空间，并且不能被自动地放进寄存器中。为了有效地解决这个问题，AVR 单片机专门增加了一个有 16 位地址的指令，能够寻址 64 KB 的数据空间。为了访问大容量的存储器，这个指令占用了两个 16 位的字(word)。用这种寻址方式，在访问少量字节时，比用指针更有效；当然对于访问大的存储器区域，还是间接寻址更有效。

下面是一个访问一个字符变量的代码：

```
间接寻址(6 个字节)                    直接寻址(4 个字节)
LDI     R30,LOW(CHCARVAR)             LDS     R16,CHARVAR
LDI     R31,HIGH(CHARVAR)
LD      R16,Z
```

1.2.2 零标志位的产生

为了实现条件跳转，AVR 有一些指令用来管理状态寄存器。条件跳转指令根据状态寄存器里的标志决定是否跳转。算术指令控制这些标志位，从而使判断一个数 A 是否大于、等于或小于另一个数 B 成为可能。当这两个数都是 8 位时，是没有问题的，因为所有的标志位都是通过一条指令来设置的；但在 C 语言中更多应用的是 16 位和 32 位数，这时问题就有些棘手了。因为 32 位数的减法要用 4 个连续的 8 位减法指令计算，每次减完，都会设置新的标志位。如果不考虑这个因素，做两个 32 位数的比较时，按通常的零标志位产生方式，得到的结果只能表示最高位是否相等，而不是整个数。AVR 内部把这个问题解决了(从而不需要编译器用更多的代码解决它)，它的零标志位和所有参与运算的寄存器的值有关，而不是最后一组，这样所有的条件跳转在最后一次减法完成后就可以执行了，因为溢出和正标志位都只与最高位有关。

1.2.3 算术运算的调整

- 加法的调整。增加了便于处理 16 位、32 位变量和常数相加减的、带进位的减立即数指令 SBCI，这样 16 位、32 位数和常数运算时压缩了代码，而 8 位数处理时并没有什么变化。
- 和常数的比较。增加了和常数进行比较的指令，这在实际中应用也很频繁。
- 非破坏的比较。两个 8 位数相比较时，可以用比较指令。但如果是 16 位或者 32 位数相比较，可能就得用带进位减法去判断标志位，这种方法的问题是覆盖了要比较的数。解决这个问题的方法是，可以把数字拷贝到另外的区域进行比较，但是这样就会用更多的指令和寄存器。针对这个问题，AVR 增加了带进位的比较，使大于 8 位的数的比较可以非破坏地实现。

通过上面的说明可以看出，AVR 单片机为适于 C 语言应用做了大量优化。它的结构和指令集为用 C 语言进行开发奠定了基础。

图 1.2 是 ATMEL 公司在相同应用下的 C 代码、汇编代码长度的比较结果。由于实际编译器等方面的差别，不能保证这些比较是绝对公平的。实际上对不同的应用，各种 CPU 的表现也是不一样的，有时这个好一些，有时那个有些优势。但是这些应该说能够反映 AVR 在提高 C 代码效率方面做出的努力。这是完成 DES(对称密钥算法)实现加密/解密时的代码量的图示。

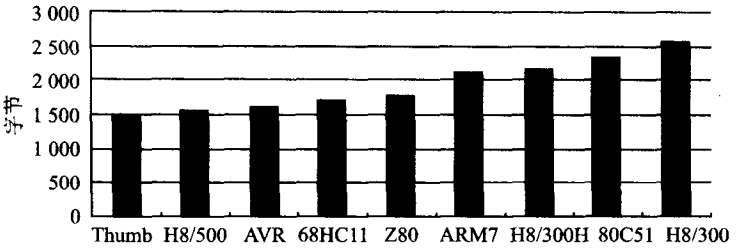

图 1.2　不同单片机编译代码比较 1

图 1.3 是完成 Reed-Solomon 编码/解码时的代码量的图示。

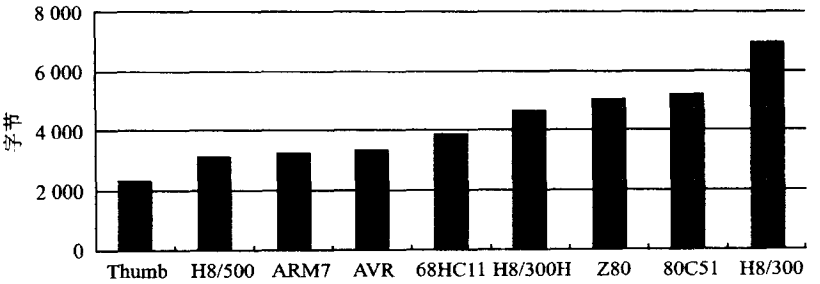

图 1.3　不同单片机编译代码比较 2

1.3　选择合适的 AVR 单片机

AVR 单片机已发展成为一个庞大的家族，包含了很多型号，它们各自的特点不尽相同。Tiny AVR 是一个 AVR 单片机的简化版，Mega AVR 属于功能比较多的类型，其他系列都有面向专门应用的功能。这些在前面也做了简单介绍，更详细的资料请阅读单片机的数据手册。

一般来说，选择单片机要考虑以下几方面，当然这都是在系统功能确定的情况下才能确定。

- I/O 个数。都需要控制哪些外围设备？需要多少个 I/O？
- 程序空间的大小。程序会很庞大吗？需要多少程序空间？这个问题很少有人能够在真正编程之前估计出来，但可以提醒大家的是，如果系统会得到数学运算库，特别是浮点库的完全支持（如要用到可以处理浮点数格式化输出的 printf()函数），那么就要注意选一个 8K 以上程序空间的单片机。
- RAM 的大小。是否要缓存处理大量的数据呢？一般来说，AVR 系列的许多型号的 RAM 比一般的单片机已经大多了，但是若要处理的数据很多，还是要小心地选择。后面也会简单介绍应用于 AVR 的实时操作系统（RTOS），若在应用中使用它，就要为它留出相应的 RAM 资源。
- EEPROM 的大小。有数据需要掉电时也保存吗？它的大小是多少？这些就决定了是否需要 EEPROM，要多大的 EEPROM。
- 外围设备。比如 A/D、通信接口（包括 SPI，USART 等）、PWM 输出、实时时钟（RTC）等。是否需要这些功能？若需要，则对其要求如何？

这些都会影响用户的选择。

还有好多因素，比如封装，是要更小的体积吗？又如功耗，多大的功耗可以接受？再如芯片的工作电压，它的电压范围和用户的系统电压一致吗？当然还有速度，足够快吗？如果使用操作系统，那么在选型时就要和所选的操作系统结合起来，给它留出相应的资源。问题实在是太多了，在这么多因素当中做选择和折中，实在不是容易的事情，有时或许会因为忽略个别问题而犯错误，比如选了8K程序空间的单片机，但是编出来的程序却超出了这个范围，笔者就碰到过。不过，犯了错不要紧，及时改正就行了，幸好AVR系列给予了这种机会。它们的内核是兼容的，指令也基本是兼容的，如果用C语言开发程序，则从一个型号单片机到另一个型号单片机的移植是简单而迅速的。笔者曾经用2个小时的时间把程序从AT90S8515移植到ATmaga8515，因为后者可以低电压工作。

为实现相应的功能并且满足预算的要求，开发者应选择合适的AVR单片机。对于一些特定的工程项目，开发者只须选择工程所需功能的单片机，而不必花费更多的预算在不必要的功能上。当然，对于个人学习使用AVR单片机而言，应该做一个折中。在功能和预算允许的范围内，还要考虑开发的进度，因此各种单片机的模拟调试工具是否廉价和容易获得，是否有合适的可做开发前期验证的开发板都要提前考虑。因为技术开发都是需要调试才能成功的，调试改进的过程往往决定了开发的速度。

表1.1给出一个AVR单片机的选型列表，供大家选择时参考。

1.4 选择合适的编程语言

目前，AVR单片机的开发主要用到两种语言：汇编语言和C语言。以前学习单片机都要先学习汇编语言，因为大家都认为单片机程序很小，用汇编语言简单并且效率高。不过情况慢慢改变了，一方面单片机的技术发展很快，程序空间、数据空间现在增加到很大，CPU也因为AVR的出现从CISC到了RISC，现在的单片机也有了16位、32位的产品。代码空间和程序效率变得不那么紧要了。另一方面，单片机的应用更是从简单的控制扩展到了各个领域，比如有些应用要求单片机内实现TCP/IP的协议栈，这若要用汇编语言实现，实在是勉为其难。

一般认为，单片机的开发应尽量使用C语言。C语言的优势主要表现在以下几方面：

- 开发迅速。用C语言开发无须记住大量的汇编指令。
- 维护方便。产品也不能总不改进，程序不可能是写一次就完成了，还需要不断地改进和维护它。C语言的可读性和模块化使维护更加方便。
- 便于合作开发。C语言可模块化，便于合作开发。后面还会介绍，采用操作系统的开发方式会更方便，因为可把系统分为几个任务，每个任务之间只有很少的联系，可把任务分给不同的人写。
- 方便移植。由于C语言是高级语言，与平台相关的代码不多，因此在不同的平台之间移植很容易，而汇编语言却是各个不同的。

汇编语言的直接和高效也是不可否认的，幸好大部分C编译器都支持内嵌汇编代码，这样就可以把它们的优势结合在一起。C语言程序的代码效率可能比不上汇编语言的程序，但是如前所述，AVR单片机已专门针对C语言应用在硬件和指令集上做了调整，提高了C语言的代码效率。

表 1.1 AVR 单片机型号列表

		Flash/KB	掩膜ROM/KB	EEPROM/字节	RAM/字节	I/O 引脚	SPI	UART	TWI	硬件乘法器	JTAG 接口	8 位定时器	16 位定时器	10 位A/D 通道数	在线编程(I)自编程(S)	工作电压/V	时钟频率/MHz	USB 速率	HUB 端口	功能端点	LCD 驱动器	LED 驱动器	加密引擎	封装(引脚数)
TINY AVR	ATtiny11	1	—	—	—	6	—	—	—	—	—	1	—	—	—	2.7~5.5	0~6	—	—	—	—	—	—	8 DIP,SOIC
	ATtiny12	1	—	64	—	6	—	—	—	—	—	1	—	—	I	1.8~5.5	0~8	—	—	—	—	—	—	8 DIP,SOIC
	ATtiny13	1	—	64	64	6	—	—	—	—	—	1	—	4	I	1.8~5.5	0~16	—	—	—	—	—	—	8 DIP,SOIC
	ATtiny15L	1	—	64	—	6	—	—	—	—	—	2	—	4	I	2.7~5.5	1.6	—	—	—	—	—	—	8 DIP,SOIC
	ATtiny26	2	—	128	128	16	—	—	—	—	—	2	—	11	I	2.7~5.5	0~16	—	—	—	—	—	—	20 DIP,SOIC,32 MLF
	ATtiny28	2	—	—	—	20	—	—	—	—	—	1	—	—	—	1.8~5.5	0~4	—	—	—	—	—	—	28 DIP,32 TQFP, MLF
	ATtiny2313	2	—	128	128	18	—	1	—	—	—	1	1	—		1.8~5.5	0~16	—	—	—	—	—	—	20 DIP,SOIC,32 MLF
	AT90S1200	1	—	64	—	15	—	—	—	—	—	1	—	—		2.7~6.0	0~12	—	—	—	—	—	—	20 DIP,SOIC,SSOP
	AT90S2313	2	—	128	128	15	—	1	—	—	—	1	1	—		2.7~6.0	0~10	—	—	—	—	—	—	20 DIP,SOIC
	AT90S2323	2	—	128	128	3	—	—	—	—	—	1	—	—		2.7~6.0	0~10	—	—	—	—	—	—	8 DIP,SOIC
	AT90S2343	2	—	128	128	4	—	—	—	—	—	1	—	—		2.7~6.0	0~10	—	—	—	—	—	—	8 DIP,SOIC
MEGA AVR	ATmega48	4	—	256	512	23	2	1	1	Y	—	2	1	8	S	1.8~5.5	0~16	—	—	—	—	—	—	28 DIP,32 TQFP,MLF
	ATmega8	8	—	512	1K	23	1	1	1	Y	—	2	1	8	S	2.7~5.5	0~16	—	—	—	—	—	—	28 DIP,32TQFP,MLF
	ATmega88	8	—	512	1K	23	2	1	1	Y	—	2	1	8	S	1.8~5.5	0~16	—	—	—	—	—	—	28 DIP,32TQFP,MLF
	ATmega8515	8	—	512	512	35	1	1	1	Y	—	1	1	—	S	2.7~5.5	0~16	—	—	—	—	—	—	40 DIP,44 TQFP,MLF,PLCC
	ATmega8535	8	—	512	512	32	1	1	1	Y	—	2	1	8	S	2.7~5.5	0~16	—	—	—	—	—	—	40 DIP,44 TQFP,MLF,PLCC
	ATmega16	16	—	512	1K	32	1	1	1	Y	—	2	1	8	S	2.7~5.5	0~16	—	—	—	—	—	—	40 DIP,44 TQFP,MLF
	ATmega162	16	—	512	1K	35	1	2	1	Y	Y	2	1	—	S	1.8~5.5	0~16	—	—	—	—	—	—	40 DIP,44 TQFP,MLF
	ATmega168	16	—	512	1K	23	2	1	1	Y	Y	2	1	8	S	1.8~5.5	0~16	—	—	—	—	—	—	28 DIP,32 TQFP,MLF
	ATmega32	32	—	1K	2K	32	1	1	1	Y	Y	2	1	8	S	2.7~5.5	0~16	—	—	—	—	—	—	40 DIP,44 TQFP,MLF
	ATmega64	64	—	2K	4K	53	1	2	1	Y	Y	2	2	8	S	2.7~5.5	0~16	—	—	—	—	—	—	64 TQFP,MLF
	ATmega128	128	—	4K	4K	53	1	2	1	Y	Y	2	2	8	S	2.7~5.5	0~16	—	—	—	—	—	—	64 TQFP,MLF
	ATmega256	256	—	4K	8K	53	1	2	1	Y	Y	2	2	8	S	2.7~5.5	0~16	—	—	—	—	—	—	64 TQFP,MLF
LCD AVR	ATmega169	16	—	512	1K	54	1	1	1	Y	Y	2	1	8	S	1.8~5.5	0~16	—	—	—	Y	—	—	64 TQFP,MLF
	ATmega329	32	—	1K	2K	54	1	1	1	Y	Y	2	1	8	S	1.8~5.5	0~16	—	—	—	Y	—	—	64 TQFP,MLF
USB AVR	AT43USB320A	—	—	—	—	512	32	1	1	—	—	—	1	1	—	—	4.5~5.5	0~12	Full	4	3	—	—	—	100 LQFP
	AT43USB325E	—	—	—	16K	512	43	—	—	—	—	—	1	1	—	—	4.5~5.5	0~12	Full	2	4	—	4	—	64 LQFP
	AT43USB325M	—	—	—	—	512	43	—	—	—	—	—	1	1	—	—	4.5~5.5	0~12	Full	2	4	—	4	—	64 LQFP
	AT43USB326	—	16	—	—	512	32	—	—	—	—	—	2	—	—	—	4.5~5.5	0~12	Full	2	3	—	4	—	48 LQFP
	AT43USB351M	—	16	—	—	1K	19	1	—	—	—	—	1	1	12	—	4.5~5.5	—	Low-Full	—	5	—	—	—	48 LQFP
	AT43USB353M	—	24	—	—	1K	15	—	—	—	—	—	1	1	12	—	4.5~5.5	0~24	Full	2	4	—	—	—	48 LQFP
	AT43USB355E	—	—	—	24K	1K	27	1	—	—	—	—	1	1	12	—	4.5~5.5	0~12	Full	2	4	—	—	—	64 LQFP
	AT43USB355M	—	24	—	—	1K	27	1	—	—	—	—	1	1	12	—	4.5~5.5	0~12	Full	—	—	—	—	—	64 LQFP
	AT76C711	—	—	—	—	8K	42	1	2	—	—	—	1	1	—	—	3.0~3.6	0~24	Full	—	6	—	—	—	64 LQFP,BGA
	AT78C1501	—	—	—	—	—	24	—	—	—	—	—	—	—	—	—	3.0~3.6	0~40	—	—	—	—	—	—	208 LQFP
DAD AVR	AT78C1502	—	—	—	—	12K	24	—	1	—	Y	—	—	1	—	—	3.0~3.6	0~40	—	—	—	—	—	—	128 LQFP
RF AVR	AT86RF401	2	—	128	128	6	—	—	—	—	—	Y	—	—	1	—	2.0~5.0	11~19	—	—	—	—	—	—	20 TSSOP
SECURE AVR	AT90SC19236R	—	192	36K	4K	NA	—	—	1	—	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC19264RC	—	192	64K	6K	NA	—	—	1	—	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	Y	Die Module,44 LQFP
	AT90SC25672R	—	256	72K	6K	NA	—	—	1	—	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC320856	8	32	56K	1.5K	NA	—	—	—	—	—	—	1	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC3232CS	32	—	32K	3K	NA	1	—	1	Y	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	Y	Die Module,44 LQFP
	AT90SC4816R	—	48	16K	1.5K	NA	—	—	—	—	—	—	1	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC4816RS	—	48	16K	1.5K	NA	—	—	—	—	—	—	1	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC6404R	—	64	4K	2K	NA	—	—	—	—	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC6432R	—	64	32K	2K	NA	—	—	—	—	—	—	1	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT90SC6464C	64	—	64K	3K	NA	—	—	1	—	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	Y	Die Module,44 LQFP
	AT90SC6464C-USB	64	—	64K	3K	NA	—	—	1	—	—	—	2	—	—	—	2.7~5.5	NA	Full	—	4	—	—	Y	Die Module,44 LQFP
	AT90SC9608RC	—	96	8K	3K	NA	—	—	1	Y	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	Y	Die Module,44 LQFP
	AT90SC9616RC	—	96	16K	3K	NA	—	—	1	Y	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	Y	Die Module,44 LQFP
	AT90SC9636RC	—	96	36K	3K	NA	—	—	1	Y	—	—	2	—	—	—	2.7~5.5	NA	—	—	—	—	—	—	Die Module,44 LQFP
	AT97SC3201	—	32	32K	2K	12	—	—	—	—	—	—	—	—	—	—	3.0~3.6	0~33	—	—	—	—	—	Y	28 TSSOP
FPGA AVR	AT94K05AL	4-16	—	—	4K-16K	96	—	2	1	Y	Y	2	1	—	I	1.6~3.6	0~25	—	—	—	—	—	—	88 PLCC,100/144/208 QFP
	AT94K10AL	20-32	—	—	4K-16K	192	—	2	1	Y	Y	2	1	—	I	1.6~3.6	0~25	—	—	—	—	—	Y	88 PLCC,100/144/208 QFP
	AT94K40AL	20-32	—	—	4K-16K	384	—	2	1	Y	Y	2	1	—	I	1.6~3.6	0~25	—	—	—	—	—	—	88 PLCC,100/144/208 QFP