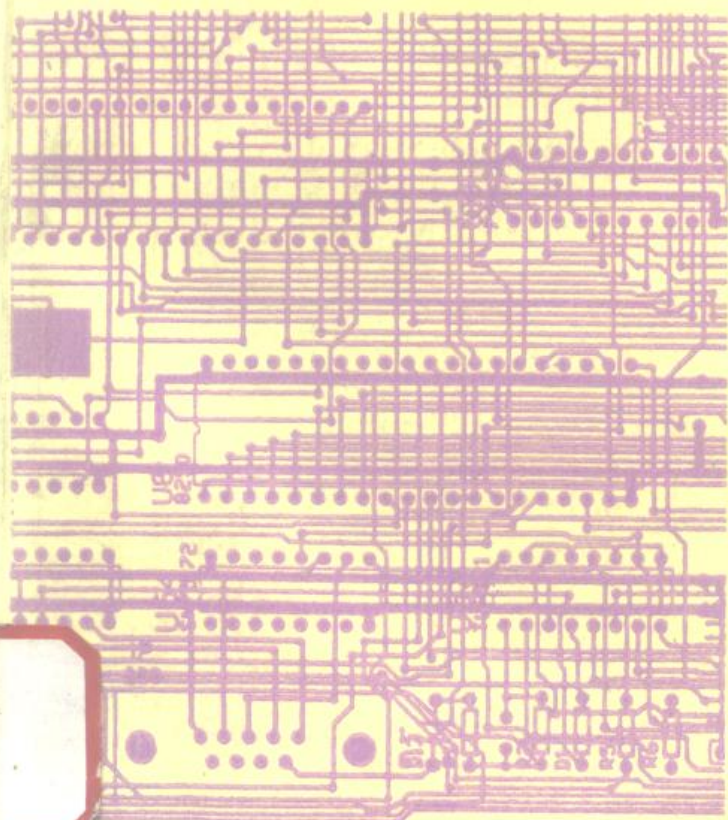
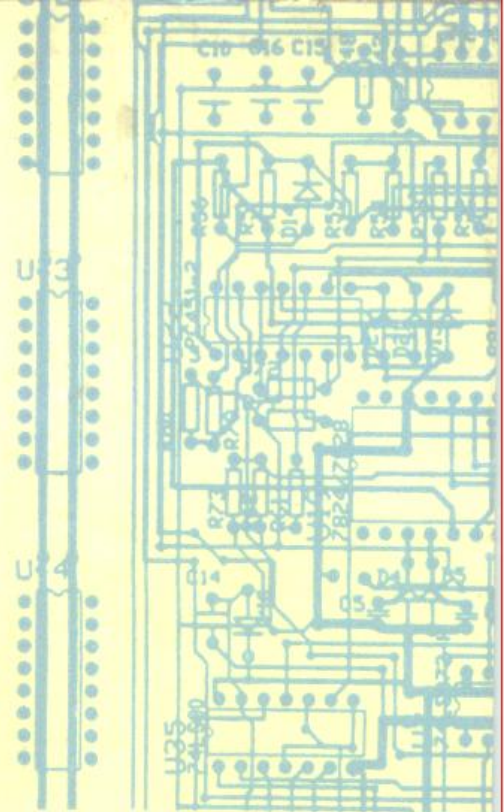


单片机

原理及接口技术

李朝青 编著

北京航空航天大学出版社

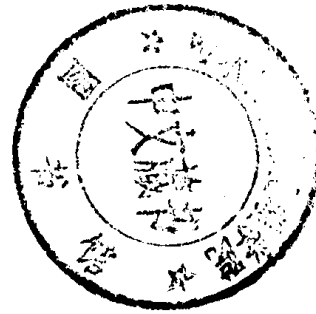


0368.1
L20

078814

单片机原理及接口技术

李朝青 编著



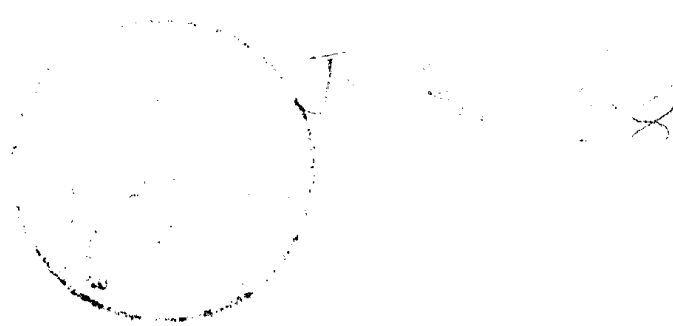
北京航空航天大学出版社

(京)新登字 166 号

内 容 简 介

本书深入浅出地介绍了 8051 单片机、80C552 单片微控制器的原理及应用技术。内容包括:数字逻辑电路和单片机常用外国芯片知识;计算机的数制和码制;微计算机的组成及工作过程;8051、80C552 的结构原理;指令系统及程序设计举例;系统扩展及接口技术;测控系统应用举例;串行通信、分布系统、光纤通信等方面的知识及实例。

本书由浅入深,自成系统,通俗、新颖、丰富、实用,适合自学,亦可作大专、中专教材和从事单片机开发与应用的工程技术人员阅读参考。



单片机原理及接口技术

DANPIANJI YUANLI JI JIEKOU JISHU

作者 李朝青

责任编辑 冯学民

北京航空航天大学出版社出版

新华书店总店科技发行所发行 各地新华书店经销

通县冕子店印刷厂印装

*

787×1092 1/16 印张:27.25 字数:693千字

1994年9月第一版 1994年9月第一次印刷 印数:10100册

ISBN 7-81012-512-5/TP·127 定价: 27.00元

前 言

单片机,亦称单片微控制器(single chip microcontroller),它的诞生是计算机发展史上一个新的里程碑。单片机技术的应用,使许多领域的技术水平和自动化程度得以大大提高。当今世界也正面临着一场以单片机(微电脑)技术为标志的新技术革命,人们渴望迅速走进单片机应用与开发的大门。

计算机的发展有两个大的趋向:一是巨型机,即向超高速、大容量、实时和智能化方向发展;另一个是向微型化(单片微控制器)、低功耗、低价格方向发展。单片机即属于后者。

单片机集成度高、运算速度快、体积小、运行可靠、价格低廉,因此在过程控制、数据采集、机电一体化、智能化仪器仪表、家用电器以及网络技术等方面得到广泛应用。

本书旨在普及单片机的开发与应用技术。作者根据多年从事微机、单片机方面教学、科研、实验的体会,由浅入深地讲述 Intel 8051 单片机和 PHILIPS 新一代 80C51 系列中典型产品 80C552 单片微控制器的原理及应用技术。本书通俗、实用、选题新颖、内容丰富,使没有学过微机原理但具有高中以上文化程度的读者也能比较顺利地阅读此书,并很快达到单片机产品开发的水平。

本书由李朝青主编,第二、四、六、七章由李朝青编写,第三章由郝廷柱编写,第一章由李运编写。范剑敏、沈怡琳、张秋燕、王澍梅、许居衡、曹文嫣等人也参加编写了部分内容。杨秀昆在整理书稿、绘图方面做了大量的工作,在此一并表示感谢。

本书在编写过程中得到北航何立民教授多方面的指导和帮助,并亲自审阅了书稿,在此特别表示感谢。

限于水平,错误难免,敬请读者批评指正。

邮政编码:300191(理工学院电子系)

联系电话:(022)736.7312

编著者

于天津理工学院

1994年2月

目 录

第一章 微型计算机基本知识	(1)
第一节 计算机中数的表示方法及运算	(1)
一、进位计数制	(1)
二、计算机中二进制数的运算	(7)
三、二进制中带符号数的表示方法及运算.....	(10)
四、数的小数点表示法.....	(15)
五、计算机常用编码.....	(17)
第二节 基本数字逻辑电路及实用芯片知识	(21)
一、门电路和逻辑代数.....	(21)
二、组合逻辑电路简介.....	(31)
第三节 时序逻辑电路及芯片知识	(41)
一、触发器.....	(41)
二、寄存器及移位寄存器.....	(48)
三、半导体存储器.....	(50)
第四节 微机的组成及工作过程	(55)
一、微处理器(机)和微机的组成.....	(55)
二、微机的工作过程.....	(59)
三、微计算机系统的概念.....	(65)
四、单片机芯片技术发展现状与展望.....	(65)
五、新一代 80C51 系列单片机	(67)
思考题与习题	(68)
第二章 MCS-51 单片机的结构和原理	(69)
第一节 MCS-51 单片机的结构	(69)
一、MCS-51 单片机的基本组成	(69)
二、MCS-51 单片机内部结构	(70)
第二节 MCS-51 单片机引脚及其功能	(73)
第三节 8051 存储器配置	(75)
一、程序存储器地址空间.....	(76)
二、数据存储器地址空间.....	(77)
第四节 CPU 时序和其他电路	(83)
一、片内振荡器及时钟信号的产生.....	(83)
二、机器周期和指令周期.....	(84)

三、CPU 取指、执行周期时序	(84)
四、访问片外 ROM 的操作时序	(85)
五、访问片外 RAM 的操作时序	(87)
六、复位及复位电路	(88)
第五节 输入/输出端口结构	(90)
一、P0 口	(90)
二、P1 口	(91)
三、P2 口	(92)
四、P3 口	(92)
五、端口的负载能力和接口要求	(93)
第六节 定时器	(94)
一、定时器概述	(94)
二、定时器控制字	(95)
三、定时器工作模式	(97)
第七节 串行接口	(100)
一、串行通信的基本知识	(100)
二、串行接口	(103)
第八节 中断系统	(111)
一、输入/输出方式	(111)
二、中断的概念	(112)
三、8051 中断系统结构及中断控制	(114)
四、中断响应过程及响应时间	(117)
思考题与习题	(118)
第三章 指令系统及程序设计举例	(120)
第一节 指令格式和寻址方式	(120)
一、指令和指令格式	(120)
二、寻址方式	(124)
三、寻址空间及符号注释	(128)
第二节 MCS-51 指令系统	(129)
一、数据传送类指令	(129)
二、算术运算类指令	(134)
三、逻辑操作指令	(138)
四、控制程序转移类指令	(140)
五、位操作类指令	(148)
第三节 MCS-51 汇编语言程序设计举例	(151)
一、简单程序设计举例	(152)
二、分支程序	(153)
三、循环程序	(155)
四、子程序设计举例	(159)

五、代码转换	(161)
六、运算类程序	(165)
思考题与习题	(170)
第四章 单片机系统扩展及接口技术	(172)
第一节 扩展程序存储器	(172)
一、扩展总线	(172)
二、扩展 8K 字节 EPROM	(173)
三、扩展 16K 字节 EPROM	(174)
第二节 扩展数据存储器	(175)
一、常用的数据存储器芯片	(175)
二、8051 扩展 2K 字节 RAM	(180)
三、8031 扩展 32K EPROM 和 32K RAM	(180)
四、8031 扩展 8K 字节 E ² PROM	(181)
五、译码法扩展大容量存储器	(183)
六、外部数据区 RAM 的调试及实验方法	(184)
第三节 并行 I/O 口的直接应用	(187)
一、I/O 口的直接输入/输出	(187)
二、开关电路及驱动电路接口	(190)
三、BCD 码拨盘输入接口	(194)
第四节 可编程并行 I/O 接口器件的扩展技术	(197)
一、扩展 8255A 可编程外围并行接口芯片	(197)
二、扩展 8155 可编程外围并行接口芯片	(205)
三、扩展多片 I/O 口及存储器的实例	(213)
第五节 键盘与显示器接口技术及实验方法	(215)
一、键盘接口与处理程序	(215)
二、LED 显示器接口与显示程序	(223)
三、键盘/LED 显示器与 8155 接口及键盘扫描子程序	(226)
四、串行口控制的键盘/LED 显示器接口电路	(228)
五、液晶显示器(LCD)接口电路	(231)
六、LCD 显示器接口实验方法	(238)
第六节 模/数与数/模转换接口技术	(238)
一、数/模(D/A)转换器接口技术及实验方法	(239)
二、模/数(A/D)转换器接口技术及实验方法	(249)
思考题与习题	(257)
第五章 80C51 系列 80C552 单片微控制器	(259)
第一节 概述	(259)
第二节 80C552 硬件结构	(262)
第三节 存储器组织及特殊功能寄存器	(268)

第四节 并行 I/O 口	(271)
一、P1, P4 和 P5 口结构及功能	(271)
二、对 I/O 口的读写	(271)
三、I/O 带负载能力	(273)
第五节 PWM 及 A/D 转换	(273)
第六节 定时器 T2 和 T3	(277)
一、定时器 T2 和捕捉比较逻辑	(277)
二、监视定时器(看门狗 T3)	(282)
第七节 中断系统	(284)
第八节 I ² C 总线简介	(289)
第六章 单片机在检测及控制系统中的应用	(302)
第一节 单片机测控小系统前向电路——传感器及小信号放大电路	(302)
一、传感器	(302)
二、模拟信号放大及集成运放简介	(303)
三、放大电路实例	(305)
四、增益可编程放大电路	(307)
第二节 数字滤波程序	(308)
一、程序判断滤波	(308)
二、中值滤波	(309)
三、算术平均值滤波	(310)
四、去极值平均滤波	(311)
第三节 软件非线性(补偿)及标度变换	(312)
一、传感器输出特性及检测回路的非线性	(313)
二、查表法	(314)
三、线性插值法	(316)
四、标度变换(工程量变换)	(317)
第四节 数据采集及巡回检测系统	(319)
一、数据采集及显示系统	(319)
二、八路巡回检测系统	(322)
三、80C552 八路巡回检测系统	(324)
第五节 电阻炉温控系统	(326)
一、系统硬件工作分析	(327)
二、软件设计	(328)
第六节 布尔处理的应用举例——单片机控制的自动装箱系统	(335)
第七章 单片机通信技术及分布式系统	(339)
第一节 串行通信基础	(339)
一、串行通信的过程及通信协议	(339)
二、8051 串行口的应用	(341)

第二节 点对点串行异步通信.....	(349)
一、8051 与 8051 之间的通信	(349)
二、8051 与 PC 机之间的通信	(364)
第三节 单片机多机通信.....	(383)
一、主从式 8051-8051 多机通信	(384)
二、外部硬件中断多机通信实例	(392)
第四节 分布式通信系统.....	(394)
一、PC 机与多台 8051 单片机间的通信	(395)
二、采用 RS-422 标准总线的分布式通信系统	(403)
第五节 光纤通信简介.....	(405)
一、光纤通信的特点	(406)
二、光纤通信系统的组成	(406)
三、光纤通信接口	(407)
附录 A MCS-51 指令系统表	(410)
附录 B MCS-51 指令矩阵(汇编/反汇编)表	(413)
附录 C 8086/8088 指令系统综述与查阅表	(413)
参考资料.....	(422)

第一章 微型计算机基础知识

本章是学习和掌握单片机开发与应用技术的基础。首先讲述计算机数制与码制、逻辑代数、数字逻辑电路以及单片机应用系统中常用外围芯片等必备的基础知识；然后结合一个假想模型机讲述微机的结构特点与工作过程。这样，即使读者没有很多计算机理论基础和应用开发经验，也能很快地走入单片机开发与应用的大门。

第一节 计算机中数的表示方法及运算

我们在日常生活中最熟悉的数制是十进制，而在计算机中采用的是二进制。计算机最基本的特点是用电信号来表示二进制信息，这些二进制信息可能是数据、地址、控制命令等。整个计算机系统的工作就是对这些二进制信息进行存储、传送、运算和逻辑判断。但计算机的操作，如用键盘输入程序、数据，或用数码管、CRT 显示器显示信息以及用打印机打印数据等又多用十进制或十六进制数。

在这一节中将讨论数在机器中的表示和运算。关于数的表示法有下列几种：

按进制来分，有十进制数、二进制数、八进制数、十六进制数以及二十进制数；

按数的性质来分，有整数（无符号数、有符号数）和小数（定点数、浮点数）；

按符号来分，有无符号数和带符号数（正数和负数）；

按精度来分，有单精度二进制数和多精度二进制数。

一、进位计数制

（一）十进制数（用 D 表示，一般省略不写）

大约在公元 400 年左右，印度数学家首先发明了用十进制计数，这可能是由于人有十个手指和十个脚指头的缘故吧。约在公元 800 年，阿拉伯人开始使用它，所以又称它为阿拉伯数制，以后传到了欧洲，才被命名为“十进制数制”。

十进制用 0、1、2、3、4、5、6、7、8、9 十个数字来表示数。十进制数制的基数是 10，当计数时，每一位计到十就往上进一位，也就是逢十进一；或者说，上一位的数是下一位的十倍。

如果用 α 表示任何一个十进制数字，那么一个含有 n 位整数、 m 位小数的十进制数的通用表示式是：

$$N = \alpha_{n-1} \times 10^{n-1} + \alpha_{n-2} \times 10^{n-2} + \cdots + \alpha_0 \times 10^0 + \alpha_{-1} \times 10^{-1} + \cdots + \alpha_{-m} \times 10^{-m}$$

或写成:

$$N = \sum_{i=-m}^{n-1} \alpha_i \times 10^i$$

十进制是我们习惯的数制,但不是唯一的数制。比如,还有二进制、八进制、十二进制、十六进制和六十进制等。

(二) 二进制数(用 B 表示)

以 2 为基数的数制叫二进位计数制,计算机中采用的是二进制数。它只包括两个符号,即 0 和 1。在一个二进制数中,前一位的权是后一位的两倍。对于整数,从右往左各位的权是 1, 2, 4, 8, 16, 32, ……; 对于小数,从左往右各位的权是 $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \dots$ 。把十进制表示式中的 10 都换为 2 就得到二进制的表示式:

$N = \alpha_{n-1} \times 2^{n-1} + \alpha_{n-2} \times 2^{n-2} + \dots + \alpha_0 \times 2^0 + \alpha_{-1} \times 2^{-1} + \dots + \alpha_{-(m-1)} \times 2^{-(m-1)} + \alpha_{-m} \times 2^{-m}$
或简写成为:

$$N = \sum_{i=-m}^{n-1} \alpha_i \times 2^i$$

式中的 α_i 是 0 或 1, 具体取值由 N 决定。

例如,二进制数 10101101.1011 B(BINARY)表示的十进制数值是:

$$\begin{aligned} & 1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} \\ & + 1 \times 2^{-3} + 1 \times 2^{-4} = 128 + 32 + 8 + 4 + 1 + 0.5 + 0.125 + 0.0625 = 173.6875 \end{aligned}$$

日常生活中我们习惯于十进位计数制,所以感到二进位计数制很不方便,那么电子计算机为什么还采用二进位计数制呢? 因为目前研究与应用最成熟的是具有两个稳定状态且具有记忆功能的电子电路,使用二进制能够很方便又直观地表示出机器中双稳态电路的两个稳定并可相互变换的物理状态;反过来,一个双稳态电路的 0 或 1 两个状态可以用来表示一位二进制的数,几个电子器件就可以代表一组多位二进制数。二进制数有以下三个特征:

1. 有两个数字符号,即 0 和 1;
2. 不同位置上的数码表示不同的权值;
3. 逢二进一。

(三) 十六进制数(用 H 表示)

尽管用二进制数表示计算机中的信息很方便,但为了便于书写和阅读,我们经常采用十六进制,即在计数时,逢十六进一,这样,书写的长度非常短,且可很方便地将十六进制数转换为二进制或将二进制转换为十六进制。大部分计算机所处理的数据位长都是 4 的整数倍(如 4 位、8 位、16 位、32 位等),所以计算机经常采用十六进制。它有以下三个基本特征:

1. 具有十六个数字符号:0、1、2、3、4、5、6、7、8、9、A、B、C、D、E、F;
2. 逢十六进位;
3. 一位十六进制数可用四位二进制数表示,它们之间存在直接而又唯一对应的关系。

例如:0011B—3H, 1010—AH。这样,一个位数较多的二进制数可以用位数较少的十六进制数来书写,既简单又易于转换。再如:

$$(0101101010110111)_2 = (5AB7)_{16} = 5AB7H(H \text{ 表示十六进制数}),$$

$$(1011100100)_2 = (2E4)_{16} = 2E4H.$$

若十六进制数最高位是 A~F 中的符号之一,应在前边加 0,说明是数字而不是文字,如十六进制数 A7CEH 应写成 0A7CEH。

(四) 不同进制之间的转换

在使用计算机的过程中,经常需要在二进制、十进制和十六进制之间进行相互转换。

1. 二进制→十进制

把二进制数转换为相应的十进制数,只要将二进制中出现 1 的数位权相加即可,整数和小数的位权如图 1.1-1 所示。

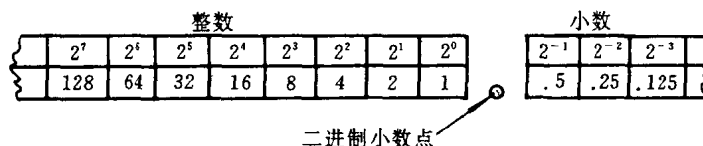


图 1.1-1 整数和小数的位权

例如,把二进制数 1010 转换成相应的十进制数。因为 1010 是整数,也就是说,小数点在该数的右边。最右边的一位称最低位(LSB),它的位权最小,为 $2^0=1$ 。最左边的一位称为最高位(MSB),因为在确定数值时,它表示的位权最大,在这个例子中,它的位权是 $2^3=8$ 。要得到总的数值,需把二进制出现 1 的位权值相加,即 2^3 与 2^1 相加,得到的十进制数是 10。

二进制数	1 0 1 0
位权值	2^3 2^2 2^1 2^0
十进制数	$8+0+2+0=10_{10}$

为了进一步说明这个转换过程,我们再举一例,把二进制数 101101.11 转换成相应的十进制数:

二进制数	1 0 1 1 0 1 1 1
位权值	2^5 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2}
十进制数	$32+0+8+4+0+1+0.5+0.25=45.75$

2. 十进制→二进制

把一个十进制的整数依次除以所需要的底数,就能够转换成不同底数的数。比如:为了把十进制的数转换成相应的二进制数,只要把十进制数依次除以 2 并记下每次所得的余数(余数总是 1 或 0)所得的余数即为相应的二进制数。这种方法称为除 2 取余法。

例如,把十进制数 25 转换成二进制数:

$25 \div 2 = 12$	余数	1 ← LSB
$12 \div 2 = 6$		0
$6 \div 2 = 3$		0
$3 \div 2 = 1$		1
$1 \div 2 = 0$		1 ← MSB

把十进制数除以 2 并记下余数,所得的商再除以 2,并记下余数,然后再把所得的商除以 2,并记下余数,如此不断继续下去,直到商得 0 为止。然后,收集余数,以余数的末位(MSB)开始,

直到余数的第一位(LSB)为止,所以 $25=11001B$ 。注意:余数一定要按颠倒顺序收集,即:第一位余数是最低位,而末位余数是最高位。

要将一个十进制小数转换成不同底数或基数的数时,则应把所需的底数或基数连续不断地乘以该十进制小数,并且记录所得的溢出数(即整数部分)。例如,将十进制数 0.3125 转换成相应的二进制数,则用 2 重复乘:

$$\begin{array}{l} 0.3125 \times 2 = 0.625 = 0.625 \quad \text{溢出} \quad 0 \leftarrow \text{MSB} \\ 0.6250 \times 2 = 1.250 = 0.250 \quad \quad \quad 1 \\ 0.2500 \times 2 = 0.500 = 0.500 \quad \quad \quad 0 \\ 0.5000 \times 2 = 1.000 = 0 \quad \quad \quad 1 \leftarrow \text{LSB} \end{array}$$

这样乘的结果在个位上(即:小数点左边)将得到的 0 或 1,记录下来,就组成相应的二进制小数。这些个位数称作“溢出”。所以,当 0.3125 乘以 2 时,溢出为 0,这个 0 就是相应的二进制小数的最高位;然后 0.625 再乘以 2,得到 1.25 ,溢出为 1,当记录数值时,乘积必须减去 1;而在下一步乘法过程中,仅是 0.25 乘以 2。用同样方法继续下去,直到乘积得 0 为止。但有时结果永不为 0,此时,只要转换到所要求的精度为止即可。将起始溢出位写在二进制小数点以后的第一位(即小数部分的最高位),并继续写到最低位。从最高位到最低位与产生溢出的顺序是一致的。数 $0.3125=0.0101B$ 。

如果十进制数包含整数和小数两部分,则必须将十进制小数点两边的整数和小数部分分开,分别完成相应的转换,然后,再把二进制整数和小数部分组合在一起。例如,将十进制数 14.375 转换成相应的二进制数:

$$\begin{array}{l} 14.375 = 14 + 0.375 \\ 14 \div 2 = 7 \quad \text{余数} \quad 0 \leftarrow \text{LSB} \quad \quad 0.375 \times 2 = 0.75 = 0.75 \quad \text{溢出} \quad 0 \leftarrow \text{MSB} \\ 7 \div 2 = 3 \quad \quad \quad 1 \quad \quad \quad 0.750 \times 2 = 1.50 = 1.50 \quad \quad \quad 1 \\ 3 \div 2 = 1 \quad \quad \quad 1 \quad \quad \quad 0.500 \times 2 = 1.00 = 0 \quad \quad \quad 1 \leftarrow \text{LSB} \\ 1 \div 2 = 0 \quad \quad \quad 1 \leftarrow \text{MSB} \end{array}$$

$14=1110B$
 $0.375=.011B$

将整数与小数部分组合在一起为:

$$14.375 = 1110.011B$$

3. 十进制↔十六进制

十六进制是微处理器、单片机经常使用的另一种数制,如指令机器码都是用十六进制表示的。

表 1.1-1 列出了十进制和十六进制之间的整数关系。表 1.1-2 列出了十进制和十六进制之间的小数关系。

表 1.1-1 十进制、十六进制、二进制整数的关系

十进制	十六进制	二进制	十进制	十六进制	二进制
0	0	0	18	12	10010
1	1	1	19	13	10011
2	2	10	20	14	10100
3	3	11	21	15	10101
4	4	100	22	16	10110
5	5	101	23	17	10111
6	6	110	24	18	11000
7	7	111	25	19	11001
8	8	1000	26	1A	11010
9	9	1001	27	1B	11011
10	A	1010	28	1C	11100
11	B	1011	29	1D	11101
12	C	1100	30	1E	11110
13	D	1101	31	1F	11111
14	E	1110	32	20	100000
15	F	1111	33	21	100001
16	10	10000	34	22	100010
17	11	10001	35	23	100011

表 1.1-2 十进制、十六进制、二进制小数的关系

十进制	十六进制	二进制	十进制	十六进制	二进制
0.00390625	0.01	0.0000001	0.06640625	0.11	0.00010001
0.0078125	0.02	0.0000001	0.0703125	0.12	0.0001001
0.01171875	0.03	0.00000011	0.07421875	0.13	0.00010011
0.015625	0.04	0.000001	0.078125	0.14	0.000101
0.01953125	0.05	0.00000101	0.08203125	0.15	0.00010101
0.0234375	0.06	0.0000011	0.0859375	0.16	0.0001011
0.02734375	0.07	0.00000111	0.08984375	0.17	0.00010111
0.03125	0.08	0.00001	0.09375	0.18	0.00011
0.03515625	0.09	0.00001001	0.09765625	0.19	0.00011001
0.0390625	0.0A	0.0000101	0.1015625	0.1A	0.0001101
0.04296875	0.0B	0.00001011	0.10546875	0.1B	0.00011011
0.046875	0.0C	0.000011	0.109375	0.1C	0.000111
0.05078125	0.0D	0.00001101	0.11328125	0.1D	0.00011101
0.0546875	0.0E	0.0000111	0.1171875	0.1E	0.0001111
0.05859375	0.0F	0.00001111	0.12109375	0.1F	0.00011111
0.0625	0.1	0.0001	0.125	0.2	0.001

(1) 十进制→十六进制

十进制转换成十六进制的方法,象十进制与二进制间的转换方法一样,只是底数是 16。例如,将十进制数 156 转换成十六进制数:

$$156 \div 16 = 9 \quad \text{余数 } 12 = C \leftarrow \text{LSB}$$

$$9 \div 16 = 0 \quad 9 = 9 \leftarrow \text{MSB}$$

将十进制数除以 16,并记下余数,如果余数超过 9,必须把两位数转换成相应的十六进制数(本例中 12=C)。然后商数再除以 16,并记下余数,不断除下去,直到商得 0 为止。将余数由 MSB 写到 LSB,则数 156=9CH(字母 H 用来表示十六进制)。

(2) 十六进制→十进制

反过来,将十六进制数(9C)₁₆转换为十进制数的过程为:

十六进制数: 9 C

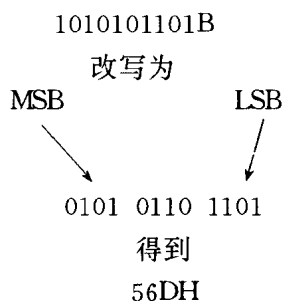
位权值: 16^1 16^0
十进制数: $9 \times 16^1 + 12 \times 16^0 = 156$

4. 十六进制↔二进制

(1) 二进制→十六进制

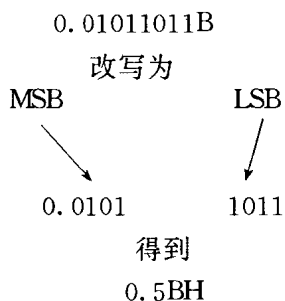
十六进制的每一位都与四位二进制数相对应,要将二进制数转换为十六进制数,首先从低位开始,把数分成4位一组,然后将每4位一组转换成相应的十六进制数。

现将二进制数 10101101101 转换成十六进制数。

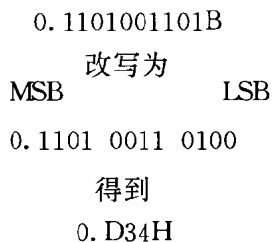


二进制数由 LSB 开始分成 4 位一组,第三组只有 3 位,必须在 MSB 左边增加一个零,而它的二进制数值仍然不变;然后将每 4 位一组转换成相应的十六进制数。注意:给二进制整数加零时,零必须加在 MSB 的左边。

用同样的方法也能将二进制小数转换成十六进制小数,只是二进制各位必须从小数点右边 MSB 开始分成 4 位一组。例如,将二进制小数 0.01011011 转换成十六进制数。



再如,将二进制小数 0.1101001101 转换成相应的十六进制数:



此例需在 LSB 的右边加两个零补足四位。

(2) 十六进制→二进制

十六进制转换成二进制的过程,恰好是上述转换的逆过程,将每位十六进制数直接转换成

③ $1+1=0$ 进位 1

④ $1+1+1=1$ 进位 1

下面我们看一下二进制加法的过程,四位二进制数 1101 与 1101 相加

$$\begin{array}{r} \text{进位:} \quad 1\ 1\ 0\ 1 \\ \text{加数:} \quad 1\ 1\ 0\ 1 \\ \text{被加数:} \quad +\ 1\ 1\ 0\ 1 \\ \hline \text{和:} \quad 1\ 1\ 0\ 1\ 0 \end{array}$$

在第一列(最右列),1加1等于0并向第二列进位1;第二列,0加0再加上第一列的进位1等于1,无进位;第三列,1加1等于0,进位1;第四列,1加1再加上第三列的进位1等于1,进位1,产生第五位1。这样就得到和为11010B。

让我们再看一下8位二进制的加法,10001111B与10110101B相加:

$$\begin{array}{r} \text{进位:} \quad 1\ 0\ 1\ 1\ 1\ 1\ 1\ 1 \\ \text{加数:} \quad 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1 \\ \text{被加数:} \quad +\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ \hline \text{和:} \quad 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0 \end{array}$$

有进位产生 (C_y) 有半进位产生 (AC)

在单片机运算过程中,两个8位二进制数相加的和向高位有进位时,设标记为C_y=1;如无进位,标记为C_y=0。如果低四位向高位有进位,标记为AC=1;如无进位,标记为AC=0。

(二) 二进制减法

二进制减法与十进制减法类似,我们先看一下十进制减法,8303减去5486,差为2817:

$$\begin{array}{r} \text{借位以后的被减数:} \quad 7\ 12\ 9\ 13 \\ \text{被减数:} \quad 8\ 3\ 0\ 3 \\ \text{减数:} \quad -\ 5\ 4\ 8\ 6 \\ \hline \text{差:} \quad 2\ 8\ 1\ 7 \end{array}$$

因为第一列的减数6大于被减数3,所以从被减数的高一位中借一个1,如果此数为0(如本例),则再从高一位不为0的数中借1,该数减少1(本例中从3减至2),被减数中跳过的0值位成为9,如本例,这等于从30中取走1,得结果为29。在十进制中,借位的数值为10,所以现在被减数的值是13,13减6等于7;第二列9减8等于1;第三列中,由于减数大于被减数,所以从高一位中借1,使被减数由2增加到12,12减4等于8;第四列,由于前面的借位,被减数由8减至7,7减5等于2。

当被减数的当前操作位值小于减数的对应位值时,总是从较高一位数中借1,该借位在数值上等于这种数制的基数。因此,在十进制数中借一位等于10,而在二进制数中借一位等于2。

二进制减法规则如下:

- ① $0-0=0$
- ② $1-1=0$
- ③ $1-0=1$
- ④ $0-1=1$ 借位 1