LIMA Electronics
QQ : 810654678
Tel: 0755-88365152 销 售
MSN: luoziqiang@hotmail.com
Avago代理分销：光耦、光纤、微波

单片机入门不难------谈 PIC 系列

单片机入门不难------谈 PIC 系列
作者:佚名　来源:不详　录入:Admin　更新时间：2008-7-27 16:27:15　点击数：3
【字体：　　】

单片机入门不难------谈 PIC 系列（转自矿石收音机论坛---崂山）

坛子里的老兄小弟，如果你已经多次想进单片机之门而未入，如果你有很多空闲时间想打发，如果你玩收音机感到疲劳想换个口味，如果。。。
那就听小弟我白话白话，反正我们这些老大难也没有什么重要的事情做。。。。

闲言少许，话说人间事物的学习，不外两类：一类入门易，精通难比如矿石机，再比如练书法谁都会写两下，可写好却不是一朝一夕的事儿。另一类则入门难精通易，比如超外差收音机，学会需要各种放大、振荡、反馈、混频、中频、统调等等，但是一旦你学会，做出一个高质量的收音机，确也不是难事。单片机属于后者，所以无论如何，你得用些时间，学习一些新名词并理解，甚至还得花一些小钱。但是，一旦你掌握，其乐趣也像收音机一样，很有吸引力的。本故事说给一点也不懂计算机，甚至数字电路的引引道。一般入门应该有保障。

改为:　本故事说给懂电工、电子、逻辑电路的老家伙和半老家伙，如果对电子一窍不通人讲那我劝你也别瞎子点灯了，要是边讲基础边讲 pic 这书就不知唱到何年何月啦。

单片机不同于收音机，收音机的任务最终就是听好声音一个，单片机可以做很多事情，要你来安排它的工作。安排它就先的认识它，熟悉它
请看图 1

这个 8 条腿的小螃蟹就是我们的第一顿饭，只要把它吃下去，以后的大餐就好办了。
第 1、8 条腿接电源 +5V　和 地线。头两条腿是螃蟹钳子，好吃的很。 现在剩下了 6 条腿
第 2、3 条腿 使用时外接一个晶振的东西　我们接一个 4 MHz 的。
第 4 条腿是复位脚，是一个信号输入脚。单片机正常运行时接高电平。当有一个低电平脉冲输入到这个脚时单片机就复位。所谓复位就是单片机内部所有的工作部件统统回到规定的状态，程序也复位到头一句上开始逐条运行。例如，你设计的一个报警锁定的 LED 红灯亮后，当需要解除报警时，用一个按钮给这个脚瞬时接地一下，相当于给它一个夫脉冲，系统就复位了，led 灯就熄灭了，程序从头开始。

以上 5 个脚，几乎所有单片机都有，包括世界上最复杂的，和世界比较简单的单片机

LIMA Singapore

功率模块、光耦、芯片电子器件库存回收
QQ: 576147342、135-4412-3655高价回收

-----PIC12CE519

轮到底几条腿啦？奥是第 5 条腿，这条叫单片机的 I/O 脚。就是输入输出脚。你可通过程序动态地控制它作为输入或输出，作为输出时可以程序控制它的输出电平为高 1 或低 0。所以，他的工作状态有四种：输入 0，输入 1，输出 0，输出 1

剩下的两条腿和第 5 脚功能一个样。

原来编辑帖子是有时间限制的阿。也好，俺学乖了，先在记事本上编辑，在复制粘帖。欢迎各位捧场，在此不敢以讲课回答问题的姿态出现，不过各位的问题我尽力而为。

上边我们已经把 8 条腿消化掉了，其实我们要弄明白的也就 3 只腿，我们再简单一些，先整明白两条腿，即 GP0,GP1.这两条腿低级一点的用法，可以控制继电器，LED 灯，高级一些的用法可以进行 I2C 总线，RS232 总线的通信，作为扩展输入可以模拟出来 A/D 转换器（6--7bit），可以测量一个电阻的粗略值。作为输出也可以直接推动扬声器奏出音乐。这是后话暂且不提。

现在要控制使用这两只腿，我这个三脚猫功夫的说书的不得不讲一下软件了，要想讲明白软件又不得不涉及到单片机的内部结构。那位说啦，你可别提这软件和结构了，以前俺就是让它们打败的，现在听到这个心里就打鼓。嘿嘿，不要紧，果真如你所说，那你就不妨跟着我再失败一次， 反正吗多一次失败又不纳税，嘿嘿。不过你也要有思想准备，彻底弄明白是个渐进的过程。

要说这程序和单片机内部结构，还真是老大难，不过蟹黄蟹肉都可都在里面。我现在要是给你说 PIC 单片机是哈佛结构的，51 系列是冯-诺伊曼结构的，恐怕你要立马扎走人了。所以我得用点心思不让你溜号。

好在 PIC 系列的制造商（microchip 微芯公司 美国）理解我等苦衷，全部只有 35 条指令，而且有一些指令我们一般很少使用，常用的也就十几句，用的时候查手册，无需记忆。就算我们两天学习一句，也就两三个月时间，总比到老了还怕它们强啊。废话少说先看下面的两个例语：

```
my_name006:          MOVlw   02h          '常数 2 进入 w
                     MOVwf   GPIO         'W 的数进入 寄存器 GPIO
```

这就是我们编的程序里的两个句子，也叫源程序。有以下特点

每行只能写一句话
每句话由四部分组成：

Lima Electronics Singapore Limited

标号：　操作指令　操作数　　'程序注释

下面我结合例子把这四部分解释一下。

第一部分 my_name006: 叫做标号，它是由字母或数字组成，由冒号结束。标号可有可无，比如第二句就没有标号。
第二部分 MOVlw 叫做操作指令。它是必须有的，不能省略。PIC 系列的单片机共有 35 条指令。
第三部分 02h　叫做操作数。有的指令没有操作数或者操作数是默认的，也不用写。
第四部分是程序注释，必须以单引号开头，主要作用是提醒和备忘。注释也是可有可无。

第二个例句中，省略了标号，当然注释也可以省略。他的指令是 MOVwf， 操作数是 GPIO。操作数不一定是数字，也可能是一个由字母组成的字符串。

知道了语句格式以后，我们下面就学习一些常用语句。我们先把这两个例句弄清楚。

这两句话的作用是把 2 这个常数写入到 GPIO 这个寄存器里。

单片机里有一些部件需要我们使用和操作，都是通过读写寄存器来实现的。每个部件都对应有操控它的寄存器，例如我们要控制使用的管脚 GP0,GP1 这两个管脚对应的寄存器就叫做 GPIO。对 GPIO 寄存器读操作，实际等效察看管脚电平的高低；对 GPIO 寄存器相应的位写 1 操作，实际等校让管脚输出高电平。写 0，输出低电平。

每个寄存器可以储存一个八位的二进制数。这八个位的每个位都有名称，从左向右的名称是：

左端第首位名称叫 D7,
左端第二位名称叫 D6,
左端第三位名称叫 D5,
左端第四位名称叫 D4,
左端第五位名称叫 D3,
左端第六位名称叫 D2,
左端第七位名称叫 D1,
最后一位叫 D0,

而每一个位对应一个管脚的电平，例如当 GPIO 寄存器的 D0 位等于 1 时表示管脚 GP0 的电平是高电平。D0 位等于 0 时表示管脚 GP0 的电平是低电平。常数 2 的八位二进制表示是“00000010” 所以，GPIO 寄存器存放的 8 位 2 进制数的每个位的值以及管脚电平是：

D7 对 应于内部总线管脚的电平　　D7=0　内部总线管脚输出低电平

D6 对应于内部总线管脚的电平　　D6=0　内部总线管脚输出低电平

D5 对应于 GP5 管脚的电平　　D5=0　GP5 管脚输出低电平

D4 对应于 GP4 管脚的电平　　D4=0　GP4 管脚输出低电平

D3 对应于 GP3 管脚的电平　　D3=0　GP3

LIMA Singapore

管脚输出低电平

D2 对应于 GP2 管脚的电平 D2=0 GP2
管脚输出低电平

D1 对应于 GP1 管脚的电平 D1=1 GP1
管脚输出低电平

D0 对应于 GP0 管脚的电平 D0=0 GP0
管脚输出低电平

GP0---GP5 管脚我们可以从上一讲的图 1 硬件中查出所对应的管脚。d7 d6 对应的内部时钟和数据总线我们现在暂且不要管它。以后本事大了在调教它们。在我们的例句中，向 GPIO 寄存器写入了 2，常数 2 的八位二进制表示是“00000010” 因此如果此时 GP0, Gp1 等都已经被定义成输出的话，那么 GP1 输出高电平（接 LED 灯亮），GP0 输出低电平(所接 led 灯熄) 。

截止到现在，你已经学会如何控制管脚的电平高低了。尽管还有一些疑问，比如怎样定义管脚为输出脚（以后会说），我得说如果事先 gp1,gp0 这两个管脚处于输入状态，这两个例句无效，是控制不了电平的。

无论如何，这一会儿，你就学会了两个指令，35 条我看也没啥难的。

'--

再加深一下对寄存器的认识：

要把一个常数存储到，或者说写到一个寄存器中，仅用一条指令是办不到的，必须通过一个特殊的寄存器 W,把数据倒过去. 这就应该使用到两个语句。

MOVlw 02H 指令的意思是把一个常数存入特殊寄存器 W, 这个常数是 3,后面的 H 是表示十六进制
MOVwf GPIO 指令的意思是把特殊寄存器 W 的数值存入寄存器. 这个寄存器的名称是 GPIO

这里涉及到两个概念,常数和寄存器.
常数好说,比如说十进制数 35, 26 但要注意,在单片机系统里我们一般不用十进制,而使用十六进制. 有关数制转换方面的知识，是计算机的基础，必须会熟练地在二进制、十六进制、十进制之间转换，我就不罗索了.
寄存器也叫单片机的内存。
一个寄存器可以存储的数值范围是 0--255,用十六进制表示就是 0---FFH.用二进制表示就是 00000000----11111111.
以后要养成习惯用十六进制表示数.

那么,一个单片机里有多少个这样的寄存器哩，pic12ce512 里面有 1024 个这样的寄存器可以供你使用,为了使用方便生产商已经给它们编上了号码,第一号码是 000H,往下依照次序为 001H,002H.....3FFH.（怎么样，开始用十六进制说事了吧，如果你不熟悉熟制转换赶紧补课来得及）

LIMA Singapore

有了编号就像我们居住的房间有了房间号码,使用就方便的多了.房间号码在邮政行业叫地址,因此我们称这些号码叫做寄存器地址,或称地址数　例如 名称为 GPIO 的寄存器,他的地址,或地址数是　06H 。所以我们的两个例句完全等同于：

```
my_name006:                MOVlw    02h          '常数 2 进入 w
                                    MOVwf    06H          'W 的数进入 寄存器 GPIO
'----------------------------------------
```

有两个寄存器比较特殊,它们没有地址,一个名字叫做 W,　另一个叫做 TRIS. 所以他们两个在存储数据的时候比较快,一个指令就可以解决问题，例如：　　MOVlw　03H　　一条指令就把常数 3 写入到 W 寄存器了。关于 TRIS 寄存器,我们以后用到它再说.
除了他们两个以外的其他所有寄存器，在写入数据时一般都要用两条指令进行。

今天就扯到这里，虽然只有两个指令，但主要目的是要同学们接触一下指令，建立寄存器的概念以及他们同硬件部件的联系。增强学习的信心。能有这些体会，这一节就算过关了。
　　随着以后的深入，你会发现小小单片机里面是一个大世界，兴趣也由此而生。

0 onc

我们上一次讲的两个指令是是如何控制管脚电平的高低。前提是所有管脚已经被定义成输出了（OUT）如果被定义成了输入，则上次的指令虽然也能运行，但运行后丝毫不能改变管脚电平高低，因为此时管脚是输入状态，电平取决于外部输入,指令无法改变。

在 PIC 单片机系列中，改变 I/O 口的输入输出依靠写入寄存器 TRIS 的值，相应位写 0，表示对应管脚被定义成了输出，写 1，就是输入。
现在假如预把 GP1、GP2 管脚定义成输出，其他脚全是输入。那就应该向 TRIS 寄存器写入二进制数 11111001，换算成十六进制就是

F9H.

依照以前我们学到的知识,在 PIC 系列单片机里，本来应该用下列的语句来完成我们的设定：

```
MOVlw　　0F9H　　'常数进 W　以字母开头的常数前面必须加 0
MOVwf　　TRISA　　'把 W 内的数复制到 TRIS
```

实际上 PIC 系列的单片机也都是这么写的,后面加的 A,表示第一个 8 位的口(有的单片机不仅一个口,还有好几个 8 位的 I/O 口如 TRISB　TRISC　TRISD 等等) .

但是,记住了,　　PIC12 系列的单片机必须改写成为:

```
MOVlw　　0F9H　　'常数进 W　以字母开头的常数前面必须加 0
tris　　GPIO　　'把 W 内的数复制到 TRIS　　以后凡见到这个指令一律理解成 MOVwf　　TRISA
```

LIMA Singapore

写法不同,意思是一样的. 这样你就又学了一个指令 TRIS,不过这个指令的实质还是你曾经学过的 MOVwf 只是写法不同罢了.
在 PIC12 系列里 TRIS 作为指令, 在其他系列(PIC16\17\18)里把 TRIS 作为普通寄存器看待.
因为我们现在讲的就是 PIC12CE519,所以我们暂时用

```
tris GPIO
```

这个格式,等以后进入 PIC16C877 我们再写成 MOVwf TRISA , 至于理解按照后者进行.
'－－－－－－－－－－－－－
如果我们要控制 GP1 GP2 管脚的输出电平, 其他管脚作为输入.并且让 GP1 输出低电平,GP2 输出高电平. 完整的程序如下:

```
MOVlw 0F9H '常数进 W
tris GPIO '把 W 内的数复制到 TRIS ，GP1 GP2 为输出，其他为输入
'此行无命令，起到的作用是容易读懂程序
MOVlw 04H '常数 4 的二进制是 00000100 ，GP1=0 GP2=1
MOVwf GPIO 'W 内的数进 GPIO 输出生效，原来定义成输入的脚的电平，不会受该句影响
```

上面已经学会了三条指令，但是 8 位寄存器的概念概念一定要建立起来，程序通过写入寄存器不同的数据

控制管脚作为输入使用还是输出使用，作为输出时是输出高电平还是低电平。
这样的操作又一个特点，就是每次写入数据，同时控制的往往不是一个管脚，而是好几个个.最多一次可

以控制 8 个管脚.在单片机里往往每 8 个脚叫做一个口,如口 A, 口 B,用英文表示就是 GPIO PORTA PORTB PORTC 等.

更多的情况是：某个口内的某一个管脚需要改变电平，其他脚电平不变. 例如我们仅需要 GPIO 口上的 GP1

这个管脚的电平拉高，其他管脚电平不发生变化. 这时候位操作指令为我们提供了方便,假如我们事先已经把 GP1 管脚定义过输出了（方法见前面讲过的）:

```
bcf GPIO,GP1 '注释 GPIO 口上 GP1 管脚电平拉低，我们行话叫清除。
bSf GPIO,GP1 '注释 GPIO 口上 GP1 管脚电平拉高，我们行话叫
```

LIMA Singapore

置位。

怎么样，这样控制某一个管脚的电平就方便多了，你的编程效率大大提高啊.

记住：PIC 所有单片机所有寄存器都是可以位操作的，这在 51 的单片机上是不能完全实现的.
不仅如此，PIC 所有单片机所有管脚的单腿驱动输出电流可以高达　25mA，所以如果你驱动一个 5 到 10mA 电流的 LED 发光二极管，根本不用加三极管，串个电阻直接挂在单片机上就得了，这在 51 的单片机上也是不能实现的，要加驱动三极管或驱动芯片.

怎么样，学 PIC 有好处吧.　也别急，好处还有那，且听我慢慢地白话。

一不小心，你已经会 5 个指令了，还有 30 个，加油啊。

继续
单片机的大部分指令，或者说单片机所做的大部分工作，多数在写入或读出寄存器。关于寄存器的初步概念我想我们已经建立起来了,它是一个能够存储 8 位二进制数据(最大 255 = 0FFH) 的单元 每个单元都有它的编号,我们叫做它的地址,或地址编码. 地址编码也是十六进制的.　另外寄存器里的数据掉电就会丢失。

寄存器的英文是 RAM　也要记住.

PIC12CE519 里面共有有 48 个寄存器供我们操作使用，　它们每一个都有固定的地址编码。地址编码并不是连续的号码,而是分成了两段:

第一段:　从 00H 开始, 依次是 01H, 02H, 03H ....0AH, 0BH.....到 1FH 结束.　计 32 个寄存器
第二段:　从 30H 开始, 依次是 31H, 32H, ...到 3FH 结束　　计 16 个寄存器

这种地址不连续编号, 而是要跳过去一段的做法, 对于我们新手来说很是不习惯. 为了让我们容易入门, 我们暂时先不管第二段 RAM, 只当它不存在, 所有程序我们只涉及到第一段连续的 ram 地址. 等我们熟练的掌握好了 ram 的使用,再涉及第二段地址的 RAM, 那时,你就会理解单片机设计者把它们分成两段的苦心了.

为了规范,我们今后一律把 RAM 的分段, 叫做分页. 第一地址段叫 00 页面, 第二地址段叫 01 页面.

例如: 我们学过的 I/O 口电平控制寄存器 GPIO, 它的地址编码是 06H, 属于 00 页面.
'-----------------
所有这 32+16=48 个寄存器除了在地址上分成了两个页面以外,又把它们分成两类：

一类专用寄存器,一类通用寄存器.

LIMA Singapore

所谓专用,就是这个寄存器的功能已经由系统分配好了.例如 地址为 06H 的名称就做 GPIO 寄存器的功能,是它的每个位,都对应到一个 I/O 脚的电平.
另一类 是通用寄存器,你可以理解成它的功能系统没有事先预定,而是由你在编程序的时候随机使用.

pic12ce519 的专用寄存器有 7 个, 位置在我们第 00 叶面的最前面. 这 7 个专用寄存器的地址编码是: 00H，01H, 02H, ----06 H

剩下的所有寄存器包括所有第 01 页面, 全部都是通用寄存器.

例子: 在两个通用寄存器 09H, 0AH 内, 写入常数 FC H

```
    MOVlw       0FCH      '常数进 W
    MOVwf       9H        '复制 W 内的数到通用寄存器 09H
    MOVwf       0AH       '复制 W 内的数到通用寄存器 0AH   由于此时 W 内并没有改变,W 不用再进常数.
'---------------------------------------------
```

下面是 PIC12CE519 的 寄存器 ram 的地址地图:

图最上端的 00 01 表示的是页面号码，或叫页面地址。
左侧 从 00 --- 1FH 是 00 页面， 右侧是 01 页面。

从 00H 到 06H 都已经起好了名称 ，它们是专用寄存器，用处各有不同。以后我们会逐个介绍它们
剩下的都是通用寄存器 或者叫普通寄存器 General Purpose Registers 意思是一般用途的寄存器

地址从 20H 到 2FH 也不是“空洞”，也不是不能访问，只是读写它们的时候等于读写它们左侧对应的 00 页面。这一点我们可能有些迷惑，弄不明白也没有关系，以后随着程序理解的深入，会搞清楚的。

内存图谱，不要求记下来，但是应该有个大体印象，用的时候会察看就可以了。等编程时间一长

就那么几个字节，自然就记住了。

所谓字节是衡量二进制数据长度的一个单位。一个寄存器刚好能记住一个字节的数据。如果你要存储的数

据比较大超过了 255，那就要占 2 个存储器甚至更多。描述的时候通常我们不说这个数值占了多少个寄存器

LIMA Singapore

，而是说这个数据是几个字节的。

字节的英文是 byte　一个二进制数的一位，叫比特 英文 bit　　1 byte 包含 8 bit

继续

下面我们学习一条新指令,叫做空操作指令

```
        nop       '什么事情也不做，但执行这个指令也要消耗掉一点时间。它没有操作数。
                  '不要理解成程序停了，实际上程序仍在正常运行。执行一连串的空操作指令，单片机
                  '白耗费时间，什么活也不干，往往用于延时
```

如果你需要一个很短时间的延时,可以采用一连串的空操作。注意每个 nop 也是占一行，例如：

```
MOVlw     0F9H        '常数进 W
          tris      GPIO        '把 W 内的数复制到 TRIS ，GP1　GP2 为输出，其他为输入
          bsf       GPIO,GP1    '管脚 GP1 输出高电平点亮 LED 灯（如果你已经接上灯的话）
          nop
          nop
          nop
          nop
          nop
          ... .

          bcf       GPIO,GP1    '管脚 GP1 输出低电平关闭 LED 灯
          nop
          nop
          nop
          nop
            ...
```

运行的效果是接在管脚 GP1 上的 LED 灯先亮一段时间，再熄灭一段时间的闪烁。

继续

由于本人一向散漫，随写随发的感觉不错，等到以后再整理不迟。发现错误不能编辑时在下

LIMA Singapore

一贴里纠正

这回再说一个程序转向的语句，goto　　指令，学过 basic　和　c　等语言的对它不陌生。单片机对程序的执行是逐句自上而下进行。当它运行到某个位置，如果你不希望继续运行它下面的语句，而是希望它无条件的强行转到某一句上，就可以使用 goto 语句。

我们还是通过例子来说明 goto 的使用方法。
已知外部晶振的频率为 4 MHz, 设计程序从 pic12ce512 单片机的 GP1 管脚上输出一个方波信号，信号频率固定并计算出频率的值。

```
                MOVlw    0F9H              '常数进 W
                tris     GPIO              '把 W 内的数复制到 TRIS ，GP1 GP2 为输出，其他为输入
myWAVE:         bsf      GPIO,GP1          '管脚 GP1 输出高电平点亮 LED 灯（如果你已经接上灯的话）
                nop
                nop
                nop
                nop
                nop
                nop
                nop

                bcf      GPIO,GP1          '管脚 GP1 输出低电平关闭 LED 灯
                nop
                nop
                nop
                nop
                nop

                goto     myWAVE            'myWAVE 是标号,某行必须有这个标号,否则程序通不过

                nop                        '由于 goto 的存在，以下语句得不到运行
                nop
                nop
```

当程序自上而下运行到 goto 语句时, 不再继续运行它底下的语句, 而是让程序强行转向到标号为 myWAVE 的语句上，并继续运行.
这样一来的结果,程序会永远在标号 myWAVE 的这一句 bsf　　GPIO,GP1　到 goto 之间循环,　打转转.

LIMA Singapore

客观运行的结果是 GP1 管脚电平不停地一会高,一会低, 就输出了方波信号.

要计算方波的频率,我们必须知道单片机每运行一条指令需要多少时间.这个时间的单位不以通常的秒 毫秒 或微秒作为单位, 而是以"机器周期" 为单位. 以后凡是我们讨论单片机内部的时间问题都要以机器周期作为时间单位. 至于一个机器周期究竟是多少微妙或毫秒, 取决于单片机的品牌和振荡频率频率大小, 等一会我们再用公式计算我们 PIC12CE512 在 4MHz 震荡频率下的机器周期是多少个微妙。

我们先看看我们的程序中 GP1 脚的高电平低电平都是用了多少个机器周期.

PIC 单片机所有指令都是单机器周期的指令,
例外的情况是 goto 语句要用 2 个机器周期 还有一个 call 指令用的时间也不完全是一个机器周期(待后续)
其他品牌的某些单片机可不是这样,一条指令往往要用几个周期……

从 bsf 到 bcf 有 8 个指令,都是单周期指令,所以 GP1 高电平时间长度是 8 个机器周期
从 bcf 到 bsf 有 7 条指令,其中 6 条是单周期指令 1 条双周期指令(goto). 所以 GP1 低电平时间长度也是 8 个机器周期

这样,我们输出方波的周期长度就是 16 个机器周期.

Pic 品牌的机器周期 = 4/振荡频率 （公式）

所以,在我们的例子当中 1 个机器周期=4/4MHz= 1 uS
也就是说,我们的例子中,执行一条指令仅需要 1 微秒的时间.

这样,我们输出的方波周期就是 16 微秒, 频率是 f =1/16 =0.0625 兆赫 =62.5 KHz

如果这个方波的频率比较低,你再接一个扬声器到 GP1 脚上你就可以听到声音了
频率降低到几赫兹的时候, 接一个 led 灯, 就会不停的闪烁.

当然, 频率太低你用的 nop 指令的数目会很多,程序虽简单但是臃肿, 这没有关系,我们主要是在学习程序, 弄清楚道理是目的。
要想使得程序不臃肿我们有的是办法,这就必须再学习新的指令.

如果此前我讲的你基本都弄明白了,那你现在已经抓住单片机入门的门把手了, 还需轻轻的推开.

LIMA Singapore

当你坐在家里吃着月饼，惬意地用电视遥控器选择电视频道，不停地用 +/- 键盘调节电视音量到合适的时候，你可曾想过，此时崂山也许正钻在在温度高达 35 摄氏度以上的树丛里，忍耐蚊子蚂蚁的叮咬，研究用什么样的通信线更好地防止雨水侵蚀和动物的啃咬。

也许你从没有留意你按下的节目频道、音量等这些 标有 + / - 符号的键盘是怎样工作控制大小的。

下面我们学习两个新指令 incf 和 decf ,它们都是对某一个寄存器进行增 1 或减 1 操作，例句中假如我们要操作的寄存器是 09H

```
MOVlw     02H     '常数 2 进入 W
MOVwf     09H     '把 w 内的数 2 复制到 09H 这个寄存器
                          '现在 09H 寄存器内存储的数是 2
incf          09H       '寄存器 09H 内存储的数 增加 1
                          '现在 09H 内存储的数变成 3
decf          09H       '寄存器 09H 内存储的数 减掉 1
                          '现在 09H 内存储的数变成 2

MOVlw     0FFH      '常数 255 进入 W
MOVwf     09H       '把 w 内的数 255 复制到 09H 这个寄存器
                          '现在 09H 寄存器内存储的数是 255

incf      09H             '寄存器 09H 内存储的数 增加 1
                          '现在 09H 内存储的数变成 0

decf      09H             '寄存器 09H 内存储的数 减掉 1
                          '现在 09H 内存储的数又变成 255
```

如果你事先定义好了地址为 09H 的这个寄存器里存储的数字大小，代表电视机节目频道的话，你会很喜欢这两个指令的。并且当节目频道到达最大值 255 或最小值 0 的时候无需担心,寄存器在 0 时减 1 会得 255, 255 状态下增 1 会得 0

至于为什么会这样，学过环形计数器的人不会感到奇怪的。你要是没有学过计数器电路也不要紧，记住一个寄存器的最大存储数值是 255 = 0FFH 就可以了，加减法都会导致它“进位”

当然控制音量时这个程序不能使用，因为它在 0 和 255 之间变化，音量忽大忽小怎们行。

为解决这个问题， 我们必须再学习两条指令 incfsz 和 decfsz

它们与上两个功能基本相同,不同的是: 寄存器增 1 或减 1 操作以后,该指令会自动判定寄存器内的结果是否为零,如果不为零,继续正常执行该指令后面的语句. 但如果结果为零的话,

LIMA Singapore

则程序会 "跳一步" .绕过紧挨着它下面的一条指令，继续执行更下面的语句,举例子说明

假定我们操作的寄存器还是 09H:

```
    MOVlw       0FDH     '常数 253 进入 W
    MOVwf       09H      '把 w  内的数 253  复制到 09H  这个寄存器
                         '现在 09H  寄存器内存储的数是 253

    incfsz      09H     '寄存器 09H 内存储的数  增加 1，结果变成 254     结果不等于 0,故程序继续执行下一指令
    nop                  '该句得到执行（因为上一句寄存器 09H 的计算结果不等于 0）
    incf        09H      '寄存器 09H 内存储的数  增加 1，结果是 255
    incfsz      09H      '寄存器 09H 内存储的数  增加 1，结果变成 0
                         '因为结果等于 0,故程序要跳过下面的一句(不运行下面的一句).
    incf        09H      '由于上一句的存在并结果为 0，该句得不到执行，被忽略
    incf        09H      '程序跳入这一句继续运行  寄存器 09H 内存储的数  增加 1
    nop                  '因此现在  09H 寄存器存储的数是 1
    nop                  '继续运行
          .
          .
          .
          .
          .
```

思考题：设计一段程序代码，当用户连续按下音量减小键后，判定音量寄存器 09H 的存储音量数值，
防止该寄存器的值从 0 变成 255，以免震惊到用户。

```
          .
          .
          .
          .
SMALL_SOUND:   nop        '标号可以任意写的，此前用户一旦按下音量减，就把程
' 序引导到这一句上来
                    decfsz      09H     '寄存器 09H 内存储的数  减 1，如果结果为 0  就跳一步
                    goto        OK      '如果上一句结果不为 0，执行该句后，程序去了 ok 语句
                    MOVlw       01H     '跳到这一步说明寄存器结果是 0
                    MOVwf       09      '强行把  09H 内的数值写成 1，仍然是小音量,这样音量不会被因为  减小而变成 255
```

UMA Singapore

```
OK:                 nop        '继续运行
        .
          .
```

思考题：利用 decfsz 指令设计一段延时代码，使得延时时间可以在 10 个机器周期到 65535 个机器周期之间，
可以通过程序任意控制
在这个例子中，设我们要控制的延时时间大约是 24086 个及其周期，用 16 进制表是就是 5E16 H.
如果用到通用寄存器，请使用 0AH, 0BH

```
yanshi:    MOVlw          5EH          ' 常数 5E 进 W 标号是延时
           MOVwf          0BH     ,    '0B 寄存器数为 5EH
           MOVlw          16H          '常数 16 进 W
           MOVwf          0AH          '0A 寄存器数为 16H

jixu:      decfsz         0AH          '0A 寄存器内的数减 1，如果结果为 0 跳步
           goto           jixu         '结果不为 0，继续

           decfsz         0BH          '0B 寄存器内的数减 1，如果结果为 0 跳步
           goto           jixu         '结果不为 0，继续

           nop            '延时完毕
        .
          .
          .
          .
```

你现在可以只用这几个简单句子完成任意时间的延时程序了。

LIMA Singapore