红石数字电路

Minecraft中的红石电路是一套图灵完备的系统，因此可以用于搭建从简单的逻辑门到复杂的可编程计算机在内的所有数字电路。

感谢MC社区的红石知识分享：

·英文Minecraft wiki：<https://minecraft.wiki/w/Redstone_circuits>

目录

[1. 红石数字电路基础 Basis of Redstone Digital Circuit 1](#_Toc159931424)

[1.1. 游戏刻和红石刻 Game Tick and Redstone Tick 1](#_Toc159931425)

[1.2. 基本逻辑门 Logic Gate 2](#_Toc159931426)

[1.3. 触发器 Flip-Flop 2](#_Toc159931427)

[1.3.1. D触发器 3](#_Toc159931432)

[1.3.2. T触发器 3](#_Toc159931433)

[2. 进阶技术 Advanced Technology 3](#_Toc159931434)

[3. 小作品：红石计算机 Redstone Computer 4](#_Toc159931435)

[3.1. 总线 Bus 4](#_Toc159931436)

[3.2. 存储器 4](#_Toc159931437)

[1.3.3. 八位寄存器 4](#_Toc159931438)

[3.3. 输出设备 5](#_Toc159931439)

[2.2.1. 七段数码管 5](#_Toc159931443)

# 红石数字电路基础 Basis of Redstone Digital Circuit

成为红石数字电路大师不只需要过硬的数字电路知识，也需要了解Minecraft中基本红石元器件（方块）的特性以及与红石相关的游戏机制。

## 游戏刻和红石刻 Game Tick and Redstone Tick

·MCwiki：<https://minecraft.fandom.com/wiki/Tick>

刻Tick是Minecraft中方块更新的频率，默认为20Hz，即0.05s一刻。红石电路的工作原理本质上就是大量相关联的方块更新。

红石刻Redstone Tick是社区创造的词汇，1红石刻=2刻=0.1s。大多数红石元件在工作时会产生至少1红石刻的传输延迟，大规模红石电路中成百上千个红石元件的延迟叠加在一起会导致明显的响应滞后。

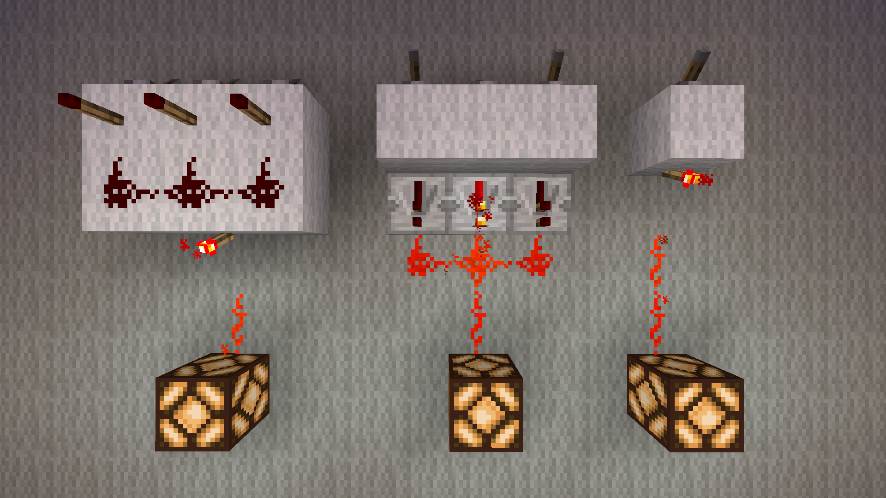
使用命令/tick rate可以改变游戏刻的频率。

注意，本文中的刻指的是**红石刻**。

## 基本逻辑门 Logic Gate

几个输入信号连在一起就实现了或门，红石火把的充能熄灭特性可以实现非门。

下图中从左到右分别是**与门**、**或门**、**非门**的简单实现，与门的实现利用了De Morgan定律，或门使用的红石中继器充当二极管。



稍复杂的组合门，如**与非门**、**或非门**、**异或门**的实现如下，其中异或门巧妙地使用了红石比较器实现。



## 触发器 Flip-Flop

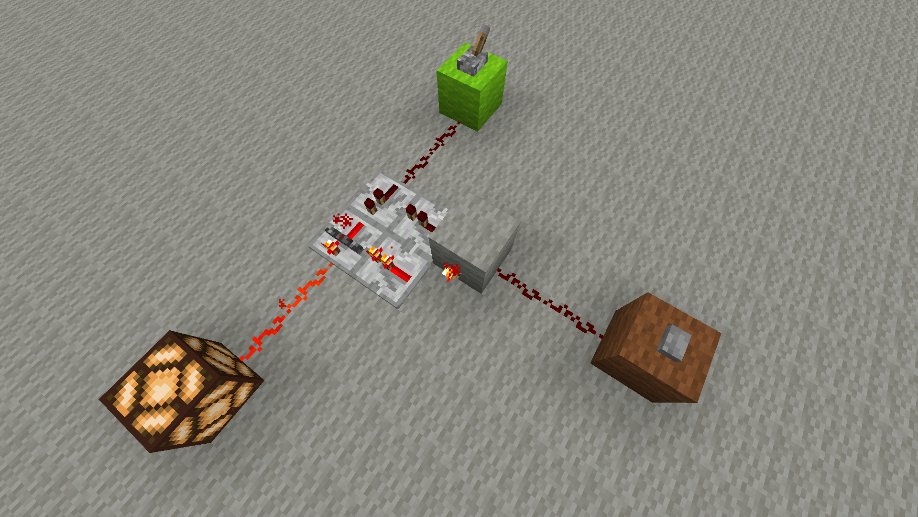
触发器使储存状态成为可能，是实现时序数字电路的基础。在数字电子课程中，介绍了如何使用基本逻辑门实现触发器，而在Minecraft中，中继器的储存特性可以用来制造小巧的触发器。

尽管从原理上，触发器触发状态转换的方式有电平触发Level Trigger和边沿触发Edge Trigger两种，但实用的触发器一般统一使用**上升沿触发Rising Edge Trigger**。



### D触发器

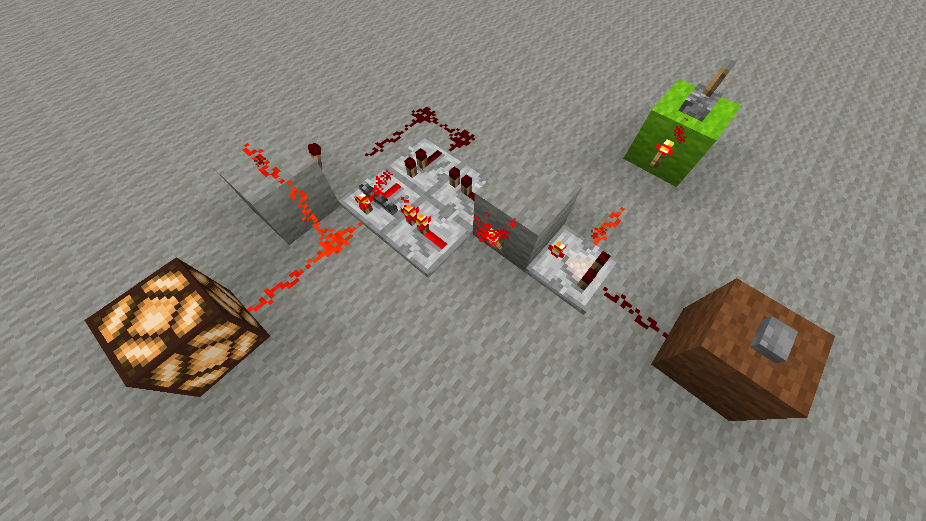
D触发器在触发后，读入并锁存输入值D。下面是一种上升沿触发的D触发器实现。绿色羊毛代表输入D值（D代表Data），红石灯代表输出Q值，棕色羊毛代表时钟信号。



### T触发器

Minecraft中有不少红石元件本身就带有T触发特性，如发射器、投掷器、铜灯

上面提到的D触发器稍作修改也可以得到T触发器，输入为0时，触发器保持输出电平不变，输入为1时，则切换输出逻辑电平。



# 进阶技术 Advanced Technology

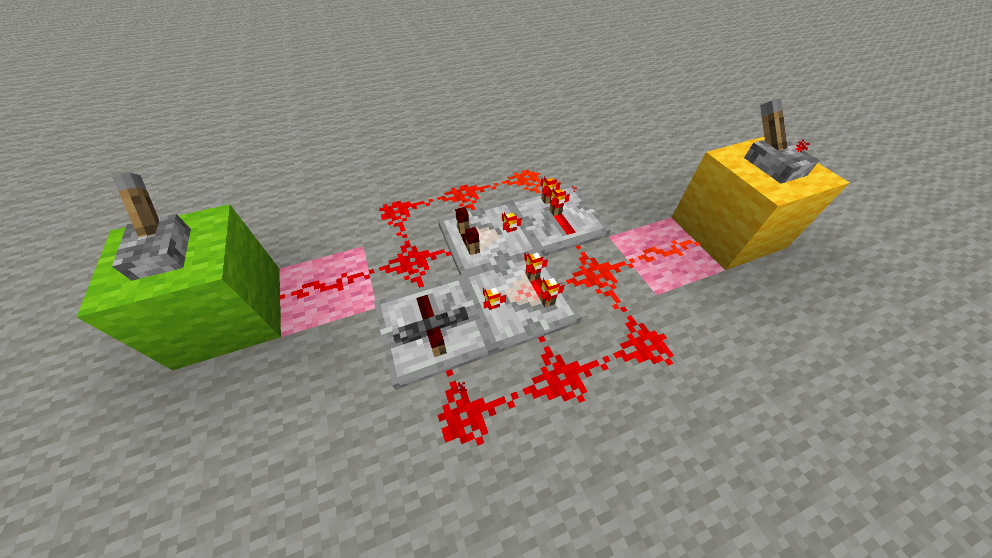
# 小作品：红石计算机 Redstone Computer

这里设计一个红石计算机，数据字长暂定为八位，可以实现简单的整数

## 总线 Bus

总线是计算机各部件之间的信息高速公路，在Minecraft中实现总线的难点在于**双向中继器**。红石粉的信号传输存在衰减，需要使用红石中继器，但两个头尾相接的红石中继器会使高电平信号在两个中继器之间闭环。

·一种双向中继的设计方案：（图中比较器要调成作差模式，粉色羊毛上的红石粉不能换成充能方块）



图中黄色羊毛上拉杆打开，正向比较器后端信号比侧端强，封锁反向中继器；反向比较器后端信号比侧端弱，不会封锁正向中继器。

## 存储器

### 八位寄存器

·维基百科：<https://en.wikipedia.org/wiki/Processor_register>

真实计算机中的寄存器是CPU的内部器件，用于储存CPU计算过程中的数据或指令。要实现一个寄存器，必须要实现其读写的控制。

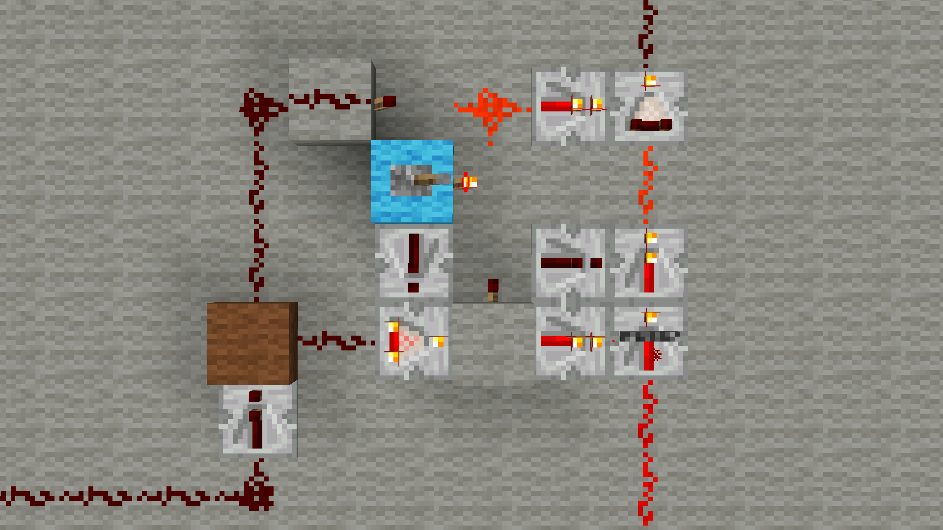
寄存器的写入需要保证数据的稳定，选用上升沿触发的D触发器实现，读出则不需要上升沿触发，使用比较器控制，高电平触发即可。

·1位寄存器的真值表：

|  |  |  |  |
| --- | --- | --- | --- |
| 选址激活 | R/W读写线 | 允许写入 | 允许读出 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

·1位寄存器的实现：

我相信有体积更小延迟更小的方案。



蓝色羊毛上的拉杆代表R/W读写选择线，打开时寄存器为读模式，允许读出，不允许写入。棕色羊毛代表选址激活线，在读模式下，高电平允许读出；在写模式下，上升沿触发写入数据。

读出、写入均存在3刻延迟。

注意，在一个指令周期内，数据信号到达D触发器必须早于选址信号上升沿，R/W读写选择线的切换也必须早于选址信号上升沿。

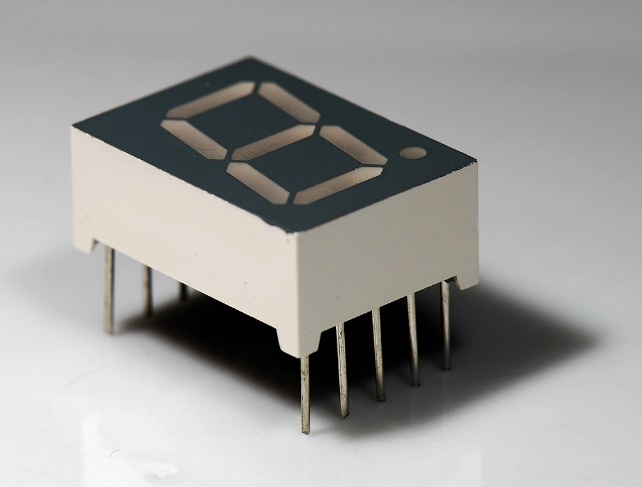
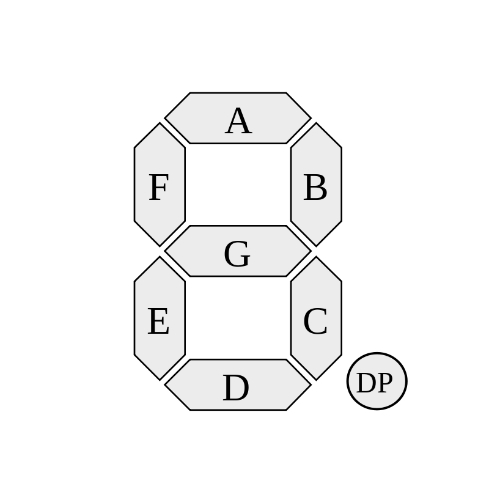
## 输出设备



### 七段数码管

·维基百科：<https://zh.wikipedia.org/wiki/%E4%B8%83%E5%8A%83%E7%AE%A1>

七段数码管或八段数码管（包含小数点）用于显示十六进制数，由于十六进制数包含所有十进制数，也用于表示十进制数。实际使用时，需要一个转换电路将四位二进制数映射到七个通道的亮灭。

·转换电路的真值表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| IN | A | B | C | D | E | F | G |
| 0000 |  |  |  |  |  |  |  |
| 0001 |  |  |  |  |  |  |  |
| 0010 |  |  |  |  |  |  |  |
| 0011 |  |  |  |  |  |  |  |
| 0100 |  |  |  |  |  |  |  |
| 0101 |  |  |  |  |  |  |  |
| 0110 |  |  |  |  |  |  |  |
| 0111 |  |  |  |  |  |  |  |
| 1000 |  |  |  |  |  |  |  |
| 1001 |  |  |  |  |  |  |  |
| 1010 |  |  |  |  |  |  |  |
| 1011 |  |  |  |  |  |  |  |
| 1100 |  |  |  |  |  |  |  |
| 1101 |  |  |  |  |  |  |  |
| 1110 |  |  |  |  |  |  |  |
| 1111 |  |  |  |  |  |  |  |

观察可以发现，每一段数码管取1的组合比取0的组合多，因此为了简便，原始逻辑式不写成“取1或”而是写成“取0或非”，对应数码管取0的输入组合或起来取非。

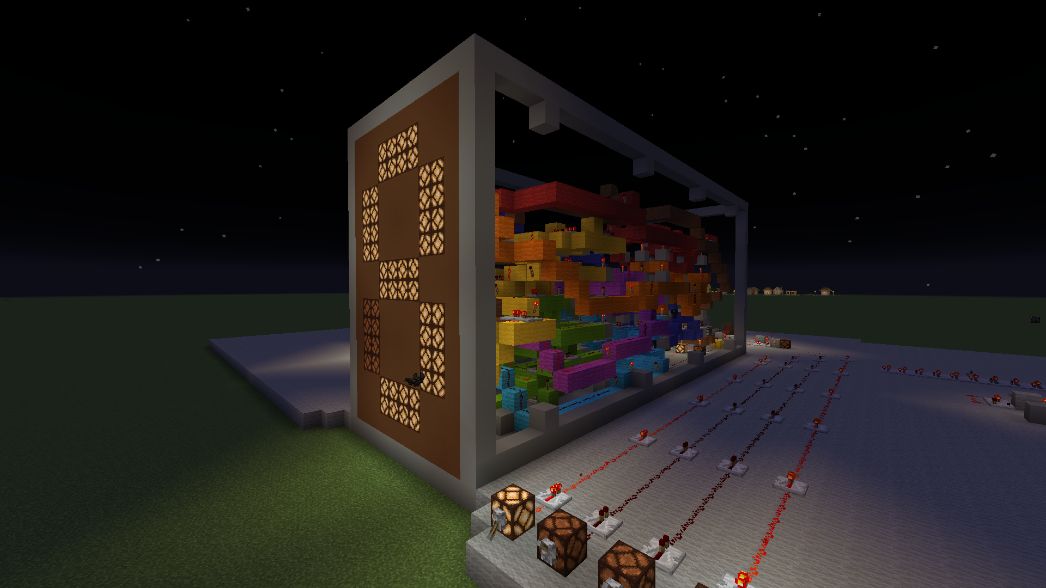
化简目的是使用到的基本逻辑门尽量少。

·逻辑表达式：

这里有个符号复用问题，逻辑表达式里的指的是四位输入IN的值，不是数码管的字母编号。

·成品展示及介绍

注意只有灰色框架以内的才是数码管电路，外围的只是测试电路。



1. 七个灯管的逻辑控制电路：（彩虹色）

利用或门、非门、异或门搭建组合逻辑电路，实现七段灯管的亮灭控制。虽然占据的体积较大，但基本原理很简单。

1. 延时同步触发电路：（棕色）

为了使切换数字时七个灯管可以同时亮灭，用侦测器检测输入信号的变化，延时1.5s后用一个2刻负脉冲同时解除七个灯管通路的锁存。这也意味着整个电路从接受输入到输出数字之间，存在1.5s的延迟。

调试发现触发时延设置短于1.5s会导致数字显示异常。

1. 极限刷新率测试：

可以接受最高1Hz的输入信号，意思是输入信号两次切换状态之间的时间间隔不能短于1s，每次改变输入信号时，二进制的四个通道要同时切换。

输入频率高于1Hz会导致数字显示异常。