# AN1384

# 镍氢电池充电器应用程序库

*作者：* *Mihnea Rosu-Hamzescu*
*Microchip Technology Inc.*

## 简介

通过电池充电器应用程序库可以轻松地向便携式应用添加电池充电功能。由于此应用程序库非常紧凑（使用不到2K字的程序空间和不到128字节的RAM），因此适合小型低成本器件，如14引脚的PIC16F616。

硬件要求（对于基本双插槽充电器）：

- 2K字程序空间
- 128字节RAM
- 1个具有自动关断功能的增强型PWM模块（使用16位定时器）
- 1个用于125+ ms计时周期的16位定时器
- 1个比较器（用于PWM逐脉冲限流）
- 1个用于补偿电源变化的带隙型参考电压（或外部ADC参考）
- 1个用于设置限流的CVREF模块（或任意其他DAC类型）
- （最低10位）模数转换器：
  - 2个用于读取电池电压的通道
  - 1个用于读取平均电流值的通道（与比较器输入共用）
  - 2个用于读取温度的通道（可选）；目前，快速充电终止算法基于电压（如果使用温度斜率，则这2个通道是必需的）
- 2个用于驱动电池开关晶体管的具有输出功能的IO端口
- 2个用于LED状态指示的具有输出功能的IO端口（可选）
- 1个用于UART调试的具有输出功能的IO端口（可选）

基本充电器具有一个采用降压转换器配置且带电流反馈的电源。为了对两节电池充电，充电器每秒在两节电池之间切换一次。当电池处于“活动”状态时，其电压和电流读数会更新，然后状态机使用此读数来决定充电过程的下一步。如果使能了调试，读数会发送至PC以进行记录。每节电池获得的平均电流为当前设定值除以插槽数。

## 镍氢电池

镍氢电池已广泛应用于多种高端便携式电子产品，电池性能是这些产品的一个主要考虑因素。移动电话和便携式计算机市场率先采用了镍氢电池。目前，大多数便携式电子产品采用镍氢电池或锂离子电池供电。随着镍氢电池产量的增加，在大多数应用中镍氢电池已取代镍镉电池，不过在一些专业领域中可能有一些例外。

## 镍氢电池与镍镉电池的比较

金属氢化物电池技术本质上是密封镍镉技术的延伸。用吸氢合金替代了基于镉的负极。在相同的重量和体积下，这一替代可增加电池电量，并且可消除重金属毒性问题，而在其他方面，金属氢化物电池与基于镉的产品非常相似。更换设计中的电池类型通常会涉及到一些重要的设计更改。

© 2014 Microchip Technology Inc.

表1： 镍氢电池与镍镉电池的比较

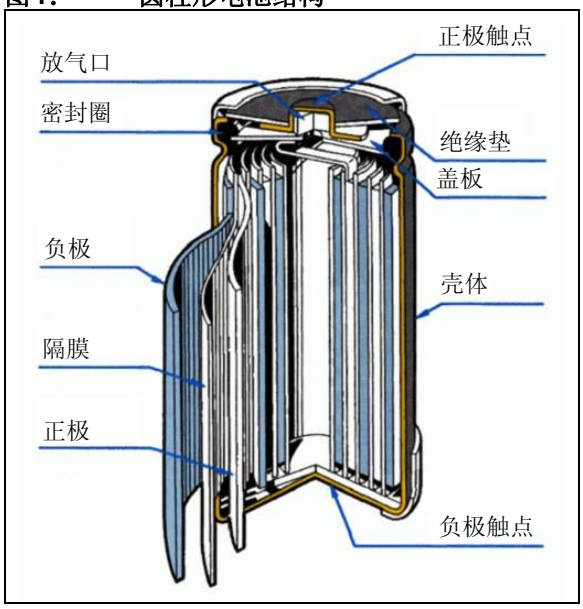
| 特性 | 镍氢电池 | 镍镉电池 |
|---|---|---|
| 标称电压 | 相同，1.25V | |
| 放电容量 | 最多高出40% | |
| 放电曲线和截止电压 | 相等，0.9V | |
| 高速率放电能力 | 本质上相同，镍镉更高 | |
| 充电过程 | 相似；多阶段恒流充电，具有过充控制 | |
| 充电终止 | 通常相似；镍氢电池的转换更微妙，并且有多种推荐的终止协议可选 | 通常相似；镍氢电池充电器应该适用于镍镉电池 |
| 自放电速率 | 镍氢电池更高 | |
| 循环寿命 | 相似，但镍氢电池更取决于应用 | 相似，但抗滥用性能更强 |
| 机械属性 | 相同 | |
| 环境问题 | 镍氢电池减少了环境问题；镍的毒性低 | 镉存在重金属毒性问题 |

## 镍氢电池的原理和结构

### 吸氢合金

金属氢化物电池的负极由特殊的吸氢合金构成。此合金可吸收1000倍于自身体积的氢气，从而形成金属氢化物。此合金还可以反过来释放其存储的气体。不同的制造商开发了多种用于电池制造的吸氢合金。

图1： 圆柱形电池结构




## 镍氢电池的电化学原理

镍氢电池利用了特殊吸氢合金中的吸氢/释放可逆性。电池的正极使用氧化镍，而负极使用吸氢合金。电解液是包含几种类似氢氧化钾（KOH）的物质的碱性溶液。

公式1： 电化学

正极：

$$NiO(OH) + H_2O + e^- \leftrightarrow Ni(OH)_2 + OH^-$$

负极：

$$MH + OH^- \leftrightarrow M + H_2O + e^-$$

总反应：

$$NiO(OH) + MH \leftrightarrow Ni(OH)_2 + M$$

（MH：金属氢化物，M：吸氢合金）
可逆反应的右侧表示电池充电状态，而左侧表示电池放电状态。


© 2014 Microchip Technology Inc.

密封金属氢化物电池的一个非常重要的问题是内部气压的积聚。负极（吸氢合金）的尺寸通常比正极大，以实现密封。在充电过程中，水的电解反应所产生的氢气在负极被吸收。此外，在正极形成的氧气通过隔膜扩散，并在负极将氢气氧化成水而消耗掉。这样在充电/放电过程中形成的所有气体通常均可被再吸收。当然，极端条件可能会导致高压积聚，在这种情况下通过紧急阀门排出气体以避免爆炸。这会对电池造成不可挽回的损坏，轻则永久性地减小容量，重则使电池无法使用。

## 放电性能

镍氢电池的放电特性非常适合现今的便携式电子产品。电压在80%的电池放电时间内稳定在1.2V，并且大容量确保了长时间工作。

## 电池容量

对于产品设计人员来说，最重要的电池参数通常是特定设备使用配置文件下的可用运行时间。虽然在采用最终设计之前确定产品的实际运行时间至关重要，但所有测试均使用额定容量执行。因此设计人员应该了解电池容量是在哪些条件下确定的，以及差异对估计性能有哪些影响。

标准电池额定值（缩写为 C）是指一节新（但有条件）电池以最佳方式充电后在室温下恒流放电时所提供的电量。电池容量与放电电流成反比。这意味着，放电电流越大，电池容量越小。对于金属氢化物电池，额定容量通常记录为5小时内完全耗尽电池电量的放电速率。

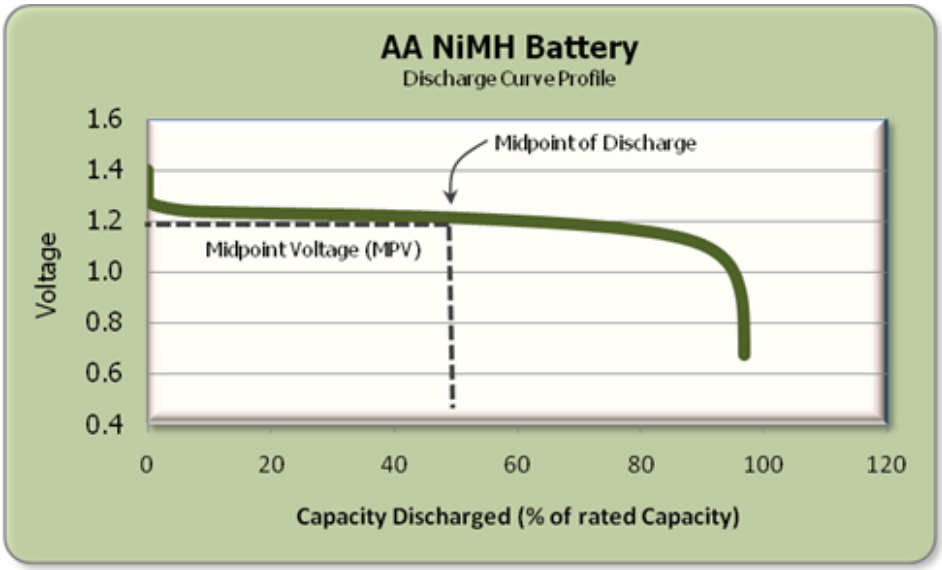
标记值可能反映的是所有电池的平均值，也可能是所有电池的最小值，具体取决于制造商。通常，镍镉电池的额定值基于最小值，而镍氢电池的额定值基于平均值。两种电池之间的差异可能会很大（约10%），具体取决于制造工艺。这就是使用C速率来归一化充电和放电参数会更加方便的原因。这基于这样一个事实：对于一系列尺寸和容量不同的电池，以C为基础进行比较时，性能通常是相同的。

## 电压放电曲线特性

放电曲线主要受温度和放电速率影响。幸运的是，在大多数情况下，电压曲线可保持电子应用所需的稳定状态。

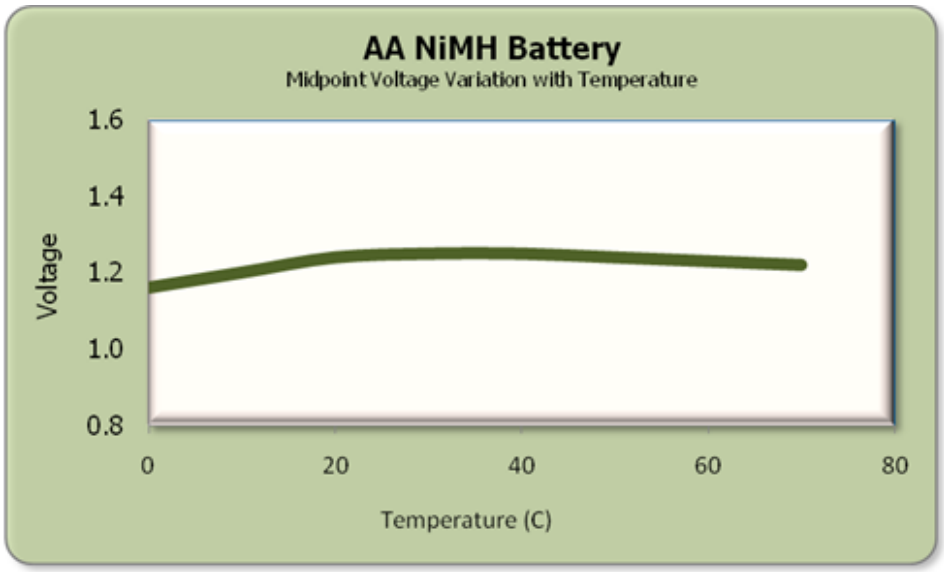
图2显示了电池以5小时速率（C/5速率）放电时的典型放电曲线。开路电压最初非常快速地从约1.4V降至1.2V稳定状态。在电池电量接近耗尽时，呈现出急拐弯的“膝盖”形状，此时电压下降得非常快。

**图2：　镍氢电池的典型放电电压曲线**



温度对放电曲线位置的影响很大，而室温的微小变化（±10°C）没有明显影响。大幅度变化（尤其是低温）将降低标称电压。

**图3：　中点电压随温度的变化**




当放电速率低于1C时，对放电曲线形状的影响不大。但是，当速率超过1C时，开始和结束阶段的瞬变将占用较大部分的持续时间。

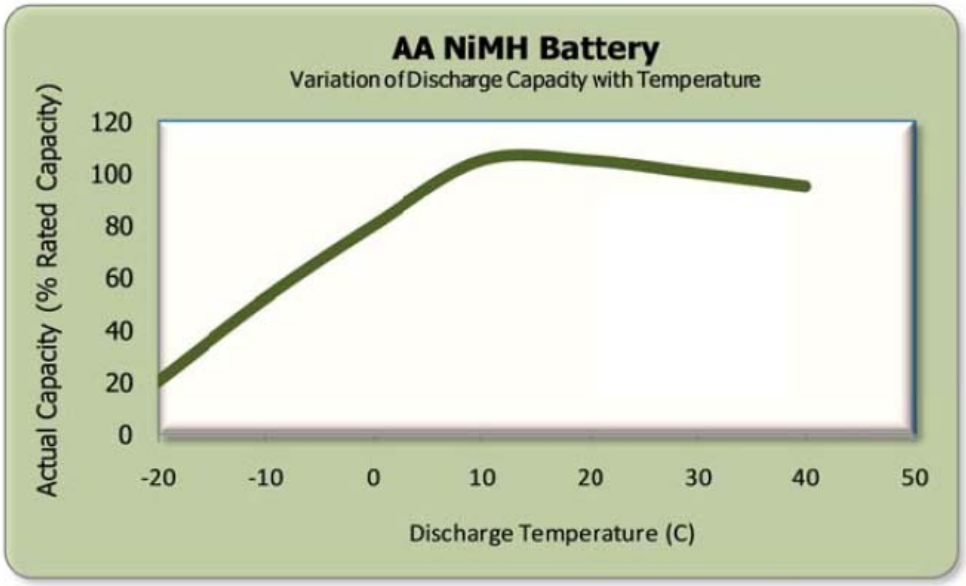
## 放电容量特性

与电压曲线类似，放电期间的容量受温度和放电速率的影响很大。容量还深受电池充电/放电/存储历史的影响。显然，如果电池没有以最佳方式充电或者已存储很长时间，则在放电时将不能放出全部容量。

低温对放电容量的影响很大。例如，金属氢化物电池在0°C下可放出约70%的电量，在-10°C下放出40%，在-20°C下放出20%。高温的影响较小，在50°C下可放出约80%的电量。

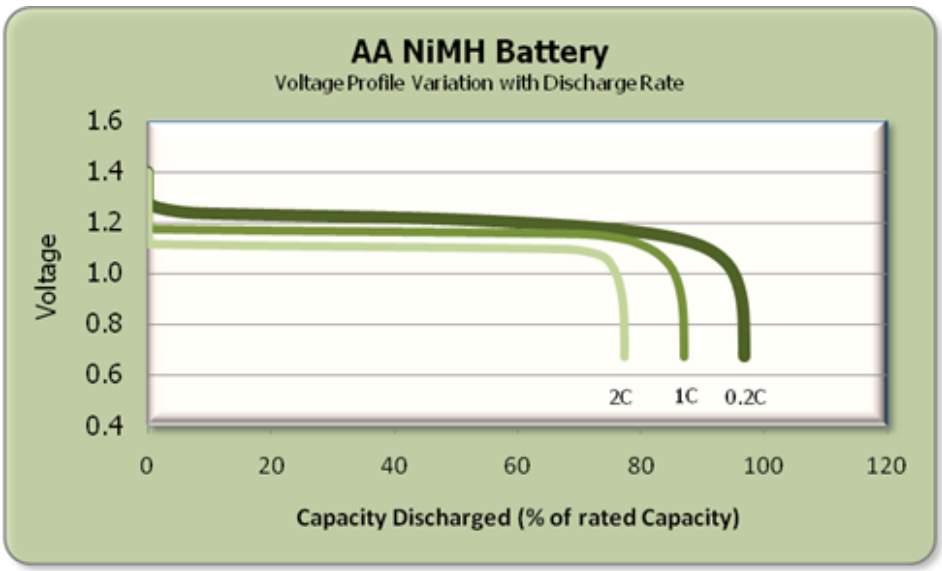

© 2014 Microchip Technology Inc.

图4： 放电容量随温度的变化




放电速率仅在超过1C时才开始显著影响容量。由于在高速率下电压降低明显，因此将导致容量减小，具体取决于放电终止协议的选择。

图5： 电压曲线随放电速率的变化



测量镍氢电池的充电状态是一个真正的挑战，因为正常放电状态下的电压曲线非常平坦。电压检测对于确定电池的充电状态没有任何作用。这对于便携式电子产品至关重要（对于计算机是一个实际问题），因为它们需要使用某种电池电量计来帮助确定何时充电或者保存当前工作。

迄今为止，惟一能合理给出精确结果的充电状态测量方式称为“库伦法”。此方法测量充电和放电期间的电流来计算剩余电量。许多设备中集成的电子元件可执行复杂的电流跟踪和自放电估计。对环境条件进行初始校准和补偿后，可计算剩余电量，精度为5-10%。

在使用镍镉电池的应用中，电压下降或“记忆效应”一直是一个问题。据报告，当镍镉电池按程序进行部分放电时，放电电压曲线中会出现约 150 mV 的电压下降。此问题的严重性有待商榷，但问题的原因在于镉电极的结构，这一点已达成共识。由于镍氢电池使用吸氢合金来替代镉，因此电压下降不再是问题。

## 放电终止

错误的放电终止可能会导致电池极性反转，进而对电池造成不可逆的损坏。强烈建议在完全放电之前将负载从电池上移除。电池在整个放电过程中的典型电压曲线有3个稳定状态。第一个稳定状态在正常放电期间，大约为1.2V。第二个和第三个稳定状态分别由正极放电和负极的剩余电量引起（通常比正极大）。

两个电极反转时，会释放出大量氢气。这将导致气体排出并对电极结构造成不可挽回的损坏。

截止电压（负载断开）通常基于每节电池 0.9V 的电压（1.2V中点电压的75%）。这对于放电速率低于1C的情况行之有效。

但是，使用大电流（1-4C）多少会使中点向较低值移动，并且接近耗尽处的“膝盖”曲线会更加圆滑，这意味着0.9V的截止电压可能有些高。这样会剩余很大一部分电量未使用。因此，在大电流应用中，所用的值最好等于应用工作中点值的75%。

电池的放电终止是一个更复杂的问题。正常的制造工艺会生产出不同容量的电池单元。当这些单元组成电池时，单元容量变化的影响会随着电池中的单元数量的增多而放大。简单地以0.9V/单元的倍数为终止电压可能会导致电池损坏，因为在达到终止电压之前，容量较低的单元会发生明显的极性反转。应该在咨询电池制造商后选择适当的放电电压，尤其是在大电池的情况下。

## 充电特性

镍氢电池能否正确充电是决定其性能的关键因素。成功的充电方案可在对快速、完全充电的需求以及最大限度减少过充（延长寿命的关键因素）的需求之间取得平衡。

相比镍镉电池，镍氢电池对充电条件更为敏感，通常需要对过程进行仔细监视。

成功为金属氢化物电池充电需满足以下几个关键因素：

- 使用三阶段充电策略
- 设计为能检测到更微小的进入过充的迹象
- 使用冗余快速充电终止技术
- 提供安全故障充电终止支持

按照上述准则可对镍氢电池进行快速可靠的充电，同时最大限度地延长电池的充放电寿命。

## 充电期间的电池特性

虽然镍氢电池与镍镉电池的放电性能非常相似，但二者在充电期间的特性存在一些明显的差异。这些差异与每种类型的电池的基本电化学成分有关。镍镉电池在充电时吸热（吸收热量），而镍氢电池在充电时放热（产生热量）。这种差异体现为各个电池类型的压力、电压和温度之间的关系。

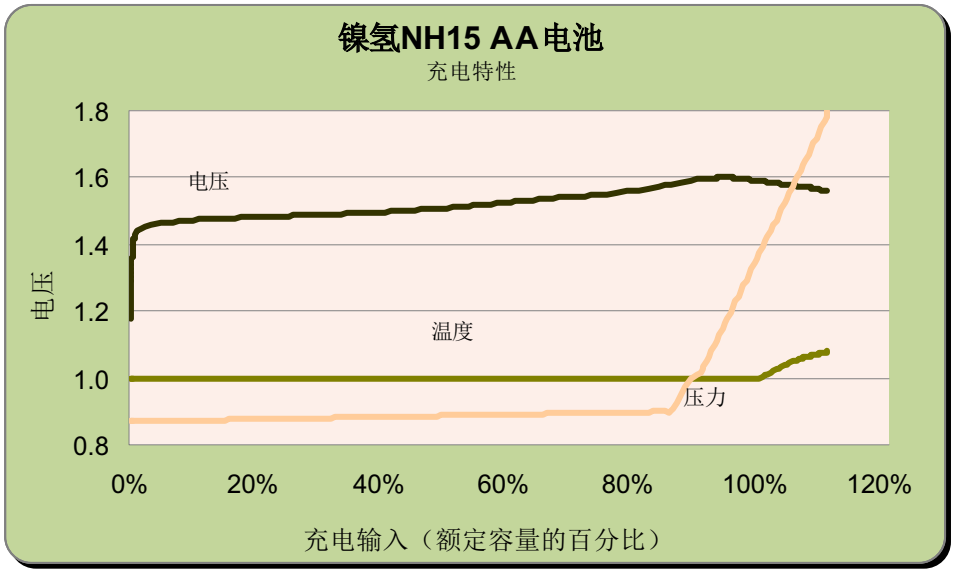
为金属氢化物电池充电时，电压最初急升，之后缓慢升高，直到完全充电。当电池到达过充时，电压达到峰值，之后逐渐降低。

由于金属氢化物电池的充电过程是放热的，因此整个充电过程都在释放热量。当电池到达过充时，大量传入能量转换为热量，使电池温度大幅升高。


© 2014 Microchip Technology Inc.

压力在充电期间也会缓慢增大，但在过充时会大幅升高，因为产生的气体量超出了电池的重新组合能力。如果没有安全口，C 速率（或更高速率）下不受控制的充电会对电池造成物理损坏。

**图6：　　镍氢电池充电特性**




温度对充电效率的影响是镍氢电池与镍镉电池之间的另一个差异所在。镍氢电池的充电接受能力随着温度的升高而降低（从低于20°C的温度开始）。镍镉电池的充电接受能力在室温达到峰值。对于两种电池类型来说，高温下充电接受能力较低是一个严重问题。在靠近热源的位置或者通风有限的隔间中安装电池时，产品设计人员应多加小心。

镍氢电池的充电接受能力还会随着充电速率的加快而提高，因此快速充电器更加高效。

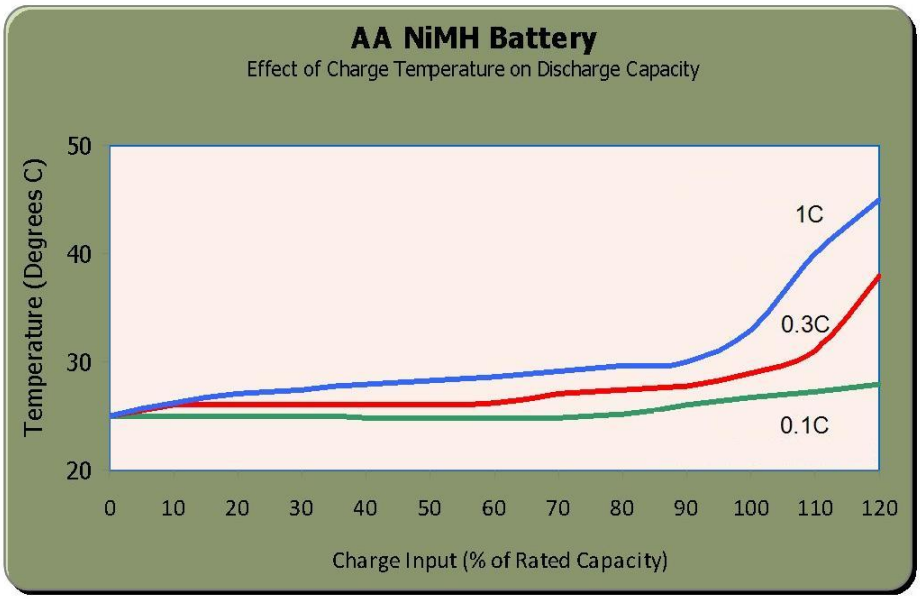
快速可靠地检测到进入过充至关重要，尤其在使用高充电速率的方案中。这是最大限度延长电池寿命的关键。

主要的充电控制方案通常依赖于对电池温度快速上升或电压峰值的检测。

要确定何时终止镍氢电池的快速充电，基于温度的充电控制是最可靠的方式，因此建议基于电压检测的主要控制机制也采用此方式。

© 2014 Microchip Technology Inc.

图7： 充电期间的温度曲线




如今的快速充电器所需的充电速率远高于早期充电器通常使用的0.1至0.3C，因为市场的趋势是缩短充电时间。此外，这些更高的充电速率会使用于检测温度和电压终止条件的斜率变化变得更加明显。要将已放电电池充满，建议使用1C的充电速率。对于使用定时“补充”充电以确保完全充电的充电方案，0.1C速率可在电荷输入与尽量减少过充的不利影响之间取得平衡。之后，保持0.025C（C/40）的充电速率便足以抑制电池自放电。遗憾的是，标准类型电池的自放电在第一天会非常严重（5-10%），但之后每天的自放电少于1%。

如今的充电策略可简单分为两类：两阶段充电器（或慢速充电器）和三阶段充电器（或快速充电器）。

两阶段充电器仅使用一个定时器从初始充电速率切换至保持充电速率。通常，出于成本原因，此类充电器不具有电压或温度检测功能，并且充电速率保持在0.1C以下，以最大限度地降低过压对电池寿命和性能的影响。要确保完全充满电量耗尽的电池，需要16至24小时的充电时间。此方案虽然经济，但并未考虑环境条件或电池的放电程度，因此对于典型的镍氢电池应用，很少建议使用此方案。

三阶段充电器使用快速充电恢复约90%的电量，使用中间定时充电完成充电，然后通过保持充电提供涓流来平衡各单元并补偿自放电。快速充电（电流在1C范围内）通常通过温度检测技术来终止，此技术在过充开始时触发。

中间充电使用0.1C速率，所选的定时持续时间基于电池容量或电池组配置。此阶段使电池无需快速充电至过充区域即可确保完全充电。

对于基于电压的终止，通常跳过中间充电阶段，因为电池需要进入过充区域。

三阶段充电需要使用更复杂的电路，这会增加成本，但可以减少电池过充，从而延长电池寿命。


© 2014 Microchip Technology Inc.

## 电池充电器库

### 多阶段充电

正确处理镍化学电池需要多个充电阶段，尤其在快速充电器中。需要特定机制来检测电池的插入、充电终止状态和移除。此外，深度耗尽的电池需要先涓流充电至定义点，之后再开始快速充电。根据用于快速充电终止的协议，可能需要渐止（top-off）充电阶段，也可能不需要。如果电池在使用之前长期置于充电器中，将使用一个非常小的保持充电电流来抑制电池自放电。

### 可配置参数

充电器库支持多个可配置参数，允许轻松定制充电曲线，而无需对硬件进行任何更改。当然，会存在一些限制。例如，如果降压转换器中使用的电感的最大规定电流为 1.5A，则设置一个对应于 2A 的电流阈值显然是不可行的，或者更糟糕的是导致硬件永久损坏。

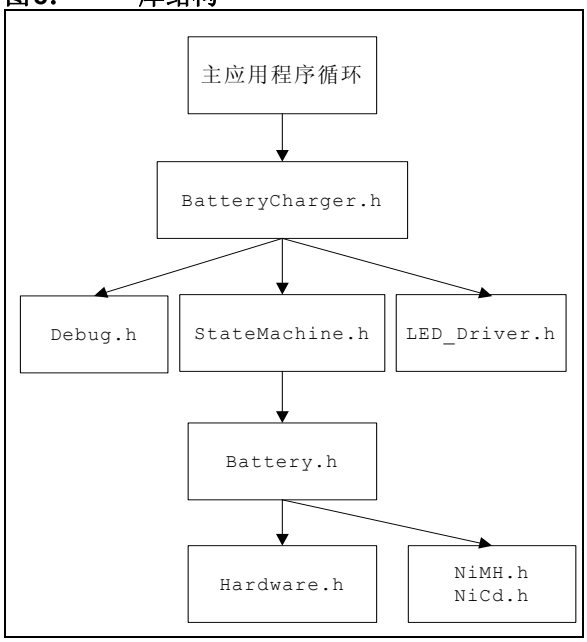
- 充电器功能：
  - 慢速充电器，具有可选保持充电（过时的方法）
  - 快速充电器，具有对耗尽电池的预充电、可选渐止充电以及保持充电
- 电池电压：
  - 插入/移除检测电压
  - 充电开始时的最小和最大浮充电压
  - 充电期间的最大电压
  - 充电电压与浮充电压之间的最大差值，用于检测有缺陷的电池（阻抗测试）
- 电池温度：
  - 充电开始时的最低和最高温度
  - 充电期间的最高温度
- 时间：
  - 慢速充电时间限制
  - 预充电时间限制
  - 快速充电时间限制
  - 渐止时间限制
  - 保持时间限制
- 充电电流（取决于DAC分辨率）：
  - 慢速充电电流
  - 预充电电流
  - 快速充电电流
  - 渐止电流
  - 保持电流
- 充电状态：
  - LED指示

### 库结构

充电器库为树形结构，最简单的使用方式是包含根文件（`BatteryCharger.h`）并将 `DoCharger()` 函数添加到应用程序主循环中。对于不同的硬件平台，还需要在`Hardware.h`文件中进行更改，以便为单片机外设配置寄存器以及与电池、热敏电阻或电流检测放大器连接的AD通道定义正确的值。`NiMH.h`文件包含适用于大多数市售金属氢化物单节电池的充电参数，但用户应至少检查一下这些参数或者咨询电池制造商以获取正确的值。这些参数对于测试来说仍然是一个很好的起点。

使用此库的惟一要求是，必须以125 ms或更短的时间为周期调用主充电器函数。此周期通常使用带预分频器的16 位定时器测量。还将使用 PWM 定时器（在本例中为Timer2）来测量较小的时间间隔。

**图8：　库结构**



## 充电器库文件和函数

### BatteryCharger.h

这是库的根文件，必须包含在用户项目中。它包含几个高阶状态函数，应在用户主应用程序循环中使用这些函数。

`void InitializeCharger(void)`

此函数必须在主应用程序循环之前调用，以确保硬件已正确初始化。外设配置寄存器值在Hardware.h文件中定义，在新平台上使用此库时，应仔细检查这些值。请参见Hardware.h文件。

`unsigned char Get_Charger_State(void)`

将当前充电器状态返回给主应用程序。返回值为下列值之一：

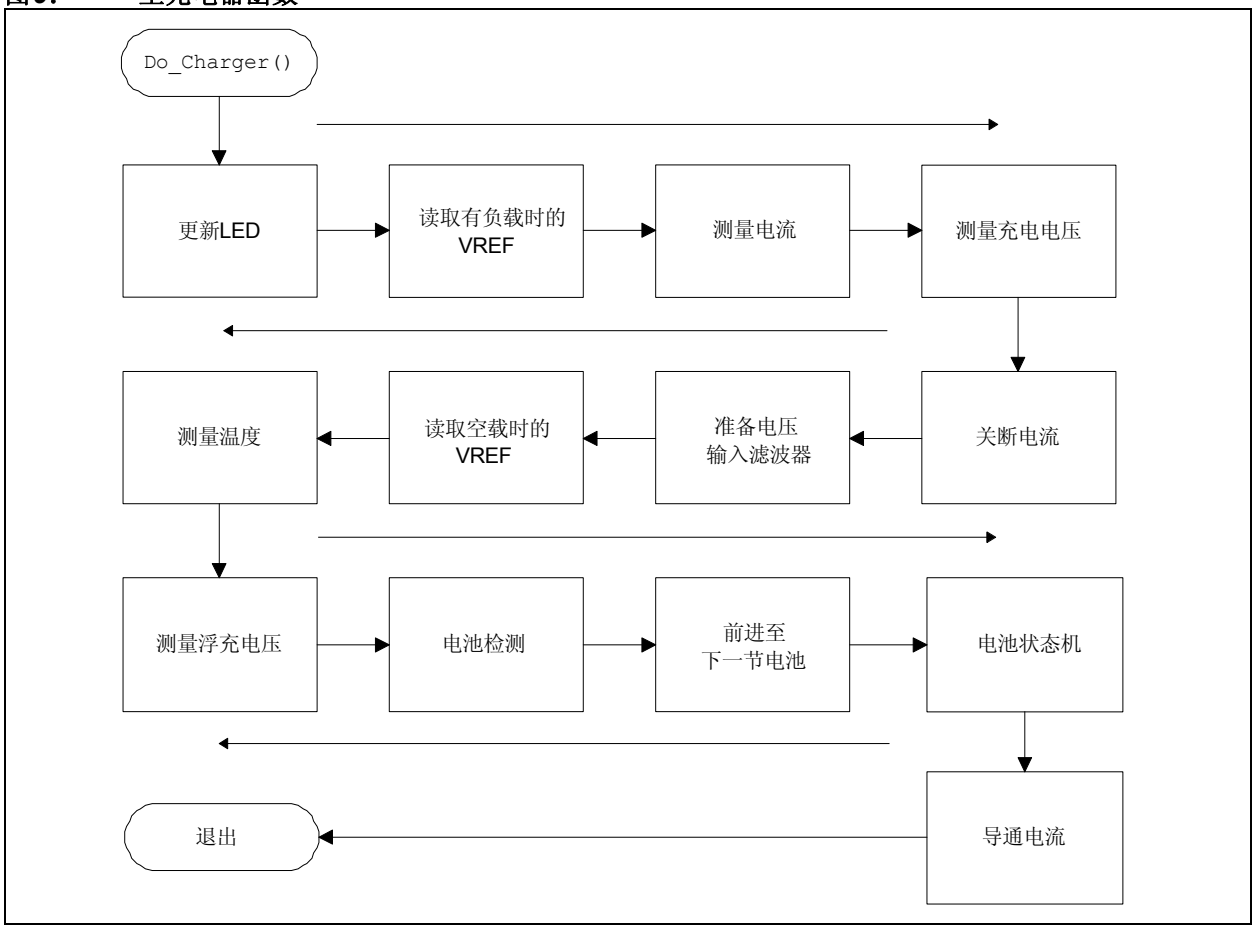
- CHARGER_IDLE——充电器中未插入电池
- CHARGER_WORKING——至少有一节电池插入并且处于充电状态
- CHARGER_DONE—— 所有插入的电池均已充电，可以取出使用
- CHARGER_FAULT—— 至少有一节电池处于故障状态，为安全起见，应对其进行检查

`void Do_Charger(void)`

这是主充电器函数。必须至少每125 ms从主应用程序循环调用一次此函数。此函数可处理所有电压、电流和温度测量、存在检测、电池状态机、LED指示以及发送调试数据。

此函数会在进入和重载时检查定时器是否溢出。默认定时器计时周期为125 ms。

**图9：　主充电器函数**





© 2014 Microchip Technology Inc.

充电电流读数和充电电压读数始终是在向电池注入电流时获取的。使用下桥臂电流检测时存在一个小缺点，因为电流检测电阻上的小幅压降会出现在读数中。通过对电流和电压进行适当滤波，可轻松地在固件中消除电流检测电阻的压降。使用上桥臂电流检测可完全消除此问题，但在硬件设计方面要求略高。

浮充电压读数始终在关闭电流源和电池的情况下获取。获取温度读数时最好关断电流源以避免噪声。这对于基于温度斜率的充电终止（需要 0.1 度或更好的分辨率）可能非常重要。

电池存在检测基于浮充电压读数，因此大于几百 mV 的值通常表示有电池插入。之前获取的所有电压、电流和温度读数均供电池状态机使用，并被发送至 PC 进行调试（如果使能了相应选项）。

下一阶段是前进至处于活动状态的电池并开始注入电流。

**例1：　基本用法示例**

```
#include "BatteryCharger.h"
.
.
.
void main(void)
{
    .
    // 用户初始化代码
    .
    InitializeCharger();
    while(1)
    {
        Do_Charger();
        .
        // 用户应用程序代码
        .
    }
}
```

### LED_Driver.h

LED_Driver.h 和 LED_Driver.c 仅包含用于指示电池状态或充电器状态的函数。

void LED_Blink(void)

此函数将根据预定义的方案点亮和熄灭LED。此函数可轻松定制，因为已知每个定时器计数周期会调用一次此函数。在默认的库配置中，每个电池 LED 均具有以下行为：

- 未检测到电池——LED熄灭
- 电池正在充电——LED以0.5 Hz的频率闪烁
- 电池完成充电或处于保持模式——LED点亮
- 电池处于故障状态——LED以2 Hz的频率闪烁

### Debug.h

通常包含一个用于将电池数据以特定格式发送至PC进行调试的函数。基本库使用UART软件将数据发送至PC。

void SendStatus(void)

此函数将每秒发送一个数据包，数据包结构如下：

**表2：　数据包结构**

| 大小（字节） | 说明 |
|---|---|
| 1 | 充电器ID |
| 1 | 电池数 |
| 1 | 数据包序号 |
| 2 | 空载VDD计算值 |
| 1 | 电池A充电状态 |
| 2 | 电池A浮充电压 |
| 2 | 电池A充电电压 |
| 2 | 电池A峰值电压 |
| 2 | 电池A电流 |
| 2 | 电池A温度 |
| 2 | 电池A状态定时器 |
| 1 | 电池A故障代码 |
| 1 | 电池B充电状态 |
| 2 | 电池B浮充电压 |
| 2 | 电池B充电电压 |
| 2 | 电池B峰值电压 |
| 2 | 电池B电流 |
| 2 | 电池B温度 |
| 2 | 电池B状态定时器 |
| 1 | 电池B故障代码 |
| 2 | 电路板热敏电阻B(eta)参数 |
| 2 | 电路板热敏电阻的默认阻值（通常在25°C时） |
| 2 | 0xFFFF数据包结尾同步字段（仅用于USART通信） |

相关的C语言风格的结构可轻松与64字节USB缓冲区配合使用，以实现轻松访问。

**例2：**

```
struct
    {
        unsigned char      ID;
        unsigned char      BatteryCount;
        unsigned char      sequence;
        unsigned int       Vdd;

        unsigned char      B1State;
        unsigned int       B1FloatV;
        unsigned int       B1ChargeV;
        unsigned int       B1PeakV;
        unsigned int       B1Current;
        unsigned int       B1TempC;
        unsigned int       B1StateTimer;
        unsigned char      B1FaultCode;

        unsigned char      B2State;
        unsigned int       B2FloatV;
        unsigned int       B2ChargeV;
        unsigned int       B2PeakV;
        unsigned int       B2Current;
        unsigned int       B2TempC;
        unsigned int       B2StateTimer;
        unsigned char      B2FaultCode;

        unsigned int       ThermistorB;
        unsigned int       ThermistorR0;

    } bat_values;
```

### Uart.h

出于速度和大小考虑，UART软件采用汇编语言编写。函数将适用于 PIC16 器件。用户需要检查并修改 `Uart.as` 中的几个定义，以便使用正确的比特率以及正确的输出引脚。

**例3：**

```
#define TX                 _PORTA,1
#define UARTOscillator     8000000
#define UARTBaudRate       38400
```

在此例中，输出引脚为 RA1，PIC® 器件在 8 MHz（2 MIPS）下运行，传输速率为 38,400 bps。此外，用户必须确保所选引脚具有输出功能（无 OD/OC）且配置为输出。

由于只有一个发送引脚，无法与 PC 进行双向通信，因此在调试数据包结尾添加了2字节同步字段。

如果硬件UART可用，则用户必须添加初始化代码并编写发送函数。

`void UartTx(unsigned char)`

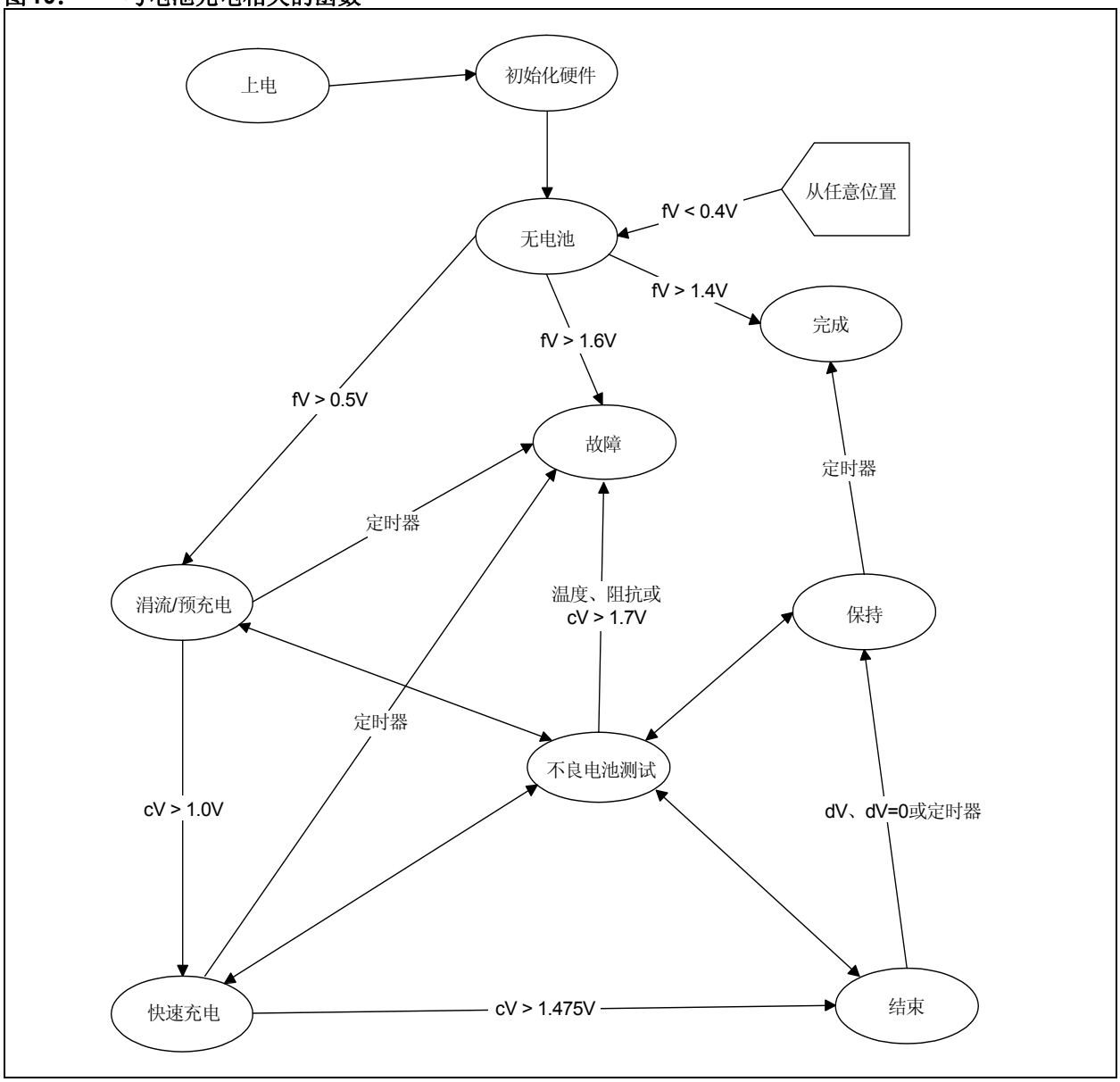
此函数使用预定义的端口速度以串行方式发送单个字符。

### StateMachine.h

包含一个与电池充电状态和各状态间转换规则有关的函数集。


© 2014 Microchip Technology Inc.

图10： 与电池充电相关的函数




void Battery_Detection(void)

此函数负责电池存在检测。某一电池插槽上的浮充电压读数超过0.5V通常表示电池已插入并开始充电。如果浮充电压超过 1.6V，则电池一定是碱性电池或某种其他错误类型的可充电电池，插槽会进入故障状态。如果电压仅超过 1.4V，则直接变为完成状态，因为电池要么是已正常充电的镍类电池，要么是部分充电的碱性电池。温度低于0°C或高于45°C时将使电池插槽进入故障状态。

如果已插入电池并且在任意时刻检测到低于 0.4V 的浮充电压，则意味着已从该插槽中取出电池，并且插槽进入无电池模式。

对于电池检测中使用的电压和温度阈值，请参见 Hardware.h。

void Battery_Fault_Check(void)

检查有效插槽中插入的电池是否有缺陷或者类型错误。过热也被视为故障，充电将停止。虽然镍化学电池不像锂离子电池一样有充电不当便可能爆炸的危险，但可能会破裂并泄漏有毒/腐蚀性物质。

此函数中执行三种安全测试：

- 绝对电压故障——如果检测到超过1.7V的电压，将停止充电
- 绝对温度故障——温度超过50°C时将停止充电
- 阻抗故障——如果充电电压与浮充电压之间的差值超出使用充电电流和允许的最大电池阻抗计算出的阈值，将停止充电

温度阈值在Hardware.h中定义，因为它适用于所有电池类型（过热通常是不好的情况）。

**例4：**

```
#define MAX_CHARGE_TEMP
```

电压阈值和阻抗限值在NiMH.h文件中定义。

**例5：**

```
#define Z_TEST_LIMIT
#define INSTANT_CHARGE_CURRENT
#define ALKALINE_DETECTION_VOLTAGE
```

void Battery_Slow_Charge(void)

仅当充电器配置为慢速充电器时，才使用此函数。在此模式下没有充电终止协议起作用，充电将在定时器计数结束后停止。故障检测始终有效。定时器值位于NiMH.h文件中：

**例6：**

```
#define CHARGER_OVERNIGHT
#define OVERNIGHT_TIMER
#define CHARGE_VOLTAGE_LIMIT
```

void Battery_Precharge(void)

涓流模式是快速充电器中的第一个状态，因为深度耗尽的电池具有特定的高阻抗。在此模式下使用低速率（约C/10）而非全电流对电池进行充电，直到充电电压超过1.0V。这可防止电池在能够接受更大的充电速率之前过热或发生故障。如果状态定时器在电池电压达到1.0V之前计数结束，则会触发故障状态。

涓流充电状态的相关规定值在NiMH.h中定义：

**例7：**

```
#define TRICKLE_CHARGE_ENABLED
#define TRICKLE_CHARGE_CURRENT
#define TRICKLE_CHARGE_TIME_LIMIT
```

void Battery_Fast_Charge(void)

在此状态下，使用大电流对电池进行充电，直到充电电压超出特定阈值或者定时器计数结束并且电池切换至故障模式为止。平均充电电流取决于电池插槽数（BATTERY_COUNT）和最大电流设置。由于电池插槽每秒切换一次，因此每节电池（如果已插入）每BATTERY_COUNT秒接收1秒的设定电流。显然，两个插槽时的占空比为50%，四个插槽时的占空比为25%。

在快速充电状态下没有充电终止协议起作用，但故障检测有效。如果定时器计数结束时电池充电电压尚未达到结束充电阈值（对于基本库版本，此阈值设置为1.475V），则电池将切换至故障模式。否则，电池将进入结束模式。

void Battery_Finish_Charge(void)

结束充电状态具有两个非常重要的功能。一个是必须完成充电，另一个是必须快速检测到进入过充状态。温度和电压可用于检测电池是否进入过充区域。这是惟一使用充电终止协议的充电状态。

温度斜率是最佳方法，因为其触发非常早，有助于延长电池寿命。如果正确执行此方法，则不会发生过热，也不会产生压力积聚。温度斜率触发值通常介于1.0和2.0°C/分钟之间。缺点是热传感器与电池壳体之间需要具有非常好的热耦合才能获得精确读数。每个电池插槽都需要具有自己的传感器，有时还需要额外的环境传感器。如果传感器不是线性的，则插值代码会占用大量程序空间和存储器。此充电终止触发后，建议以低充电速率（C/10）进行短暂的渐止充电来完成充电。

基于电压的终止协议很容易在硬件中实现，但由于系统噪声和ADC分辨率的原因，在软件中实现可能会比较棘手。强烈建议使用过采样和LP滤波。负ΔV是要查找的第一个迹象。完全充电后，镍化学电池首先会显现电压峰值，然后发生小幅压降，之后电压保持在稳定状态。镍镉电池比较容易检测，因为压降在100 mV左右。镍氢电池的压降要小得多，只有5-10 mV，因此在10位转换器上很难检测到。

检测稳定的充电电压非常简单。电池充电电压应该在整个充电过程中稳定上升，在充电结束时呈现出陡峭的斜坡。如果在定义时间内电压没有升高，则会触发稳定ΔV。如果错过负ΔV，或者电池过旧而无法显示可检测的压降，通常会发生这种情况。


© 2014 Microchip Technology Inc.

这两种触发中的任意一种都将使电池切换至保持模式或结束充电，具体取决于定义的选项。

相关定义可在NiMH.h文件中找到：

**例8：**

```
#define FINISH_CHARGE_CURRENT
#define FINISH_CHARGE_TIME_LIMIT
#define FINISH_CHARGE_TIME_RESULT
#define FINISH_CHARGE_FULL_CURRENT

#define NDV_VOLTAGE_DROP
#define NDV_SAMPLES
#define FDV_SAMPLES
#define PEAK_UPDATE_SAMPLES
```

void Battery_Topoff(void)

仅当在结束模式下成功触发温度斜率之后，电池才切换至此状态。由于温度斜率是早期的过充指示，因此建议使用低速率对电池渐止充电。如果电池寿命比容量更重要，则可跳过此阶段直接进入保持充电或结束充电。

相关定义可在NiMH.h文件中找到：

**例9：**

```
#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_TIME_LIMIT
```

void Battery_Maintenance(void)

在电池保持阶段会注入非常小的电流，仅用于抑制电池自放电，直到用户将电池取出以将其安装到目标设备中。这样做是必需的，因为标准类型电池中的自放电速率在充电后的第一天可达 5-10%。虽然电流非常小，但也不建议将电池长时间置于保持模式。状态定时器计数结束后，电池充电结束。如果未使能保持充电，状态机将跳转至充电结束。

相关定义可在NiMH.h文件中找到：

**例10：**

```
#define MAINTENANCE_CHARGE_ENABLED
#define MAINTENANCE_CHARGE_TIME_LIMIT
```

void Battery_State(void)

此函数用于检查活动的电池状态并调用相应的函数。

void CleanUp_Vars(void)

此特殊函数可在电池进入以下任一非充电状态时清除电池定时器、电压和电流值：NO_BATTERY、故障或完成。此函数是必需的，因为在插入电池时，状态机期望找到清晰的值。

### Battery.h

所有与电池相关的函数、结构和变量均在此库层定义。每节电池都具有自己的数据结构 BAT_STRUCT 和 BAT_STATE_VARS。

BAT_STRUCT包含电池的测量值和计算值：

- 浮充电压——未连接的电池上的电压读数
- 充电电压——正在注入电流的电池上的电压读数
- 峰值电压——当前记录的最大充电电压读数
- 电流 —— 电流检测电阻上的电压读数（电流 - 电压转换）
- 温度——热敏电阻/温度传感器上的电压读数
- PWM占空比——开关电源占空比
- 功率——计算出的电池功率输入

BAT_STATE_VARS 包含电池状态、定时器和去抖动计数器：

- 总充电时间 —— 电池开始充电以来的总时间（以秒为单位）
- 状态定时器——当前充电状态的剩余时间（以秒为单位）
- 峰值更新计数器 —— 新充电电压峰值的剩余计数
- 稳定电压计数器——稳定电压触发的剩余计数
- 负ΔV触发计数器——负ΔV触发的剩余计数

每个插槽的电池充电状态定义如下：

- NO_BATTERY（0x00）——未插入电池
- 故障（0x01）—— 检测到电池有缺陷或类型错误
- 完成（0x02）——电池已充电并等待使用
- 保持（0x04）—— 电池处于保持模式；正注入非常低的电流来抑制自放电
- 涓流（0x08）——将电池涓流充电至1.0V后再进入快速模式

- 快速（0x10）——电池每BATTERY_COUNT秒以最大电流充电一次；充电终止协议不起作用
- 结束（0x20）——电池以最大电流充电完毕；将跟踪峰值电压，并且充电终止协议起作用
- 渐止（0x40）——特定的结束充电终止（如温度斜率）触发后，电池将涓流充电并持续指定的时间

`unsigned char Get_Battery_State(unsigned char n)`

将所选电池（n）的电池状态返回给用户应用程序。

`unsigned long Get_Battery_Time(unsigned char n)`

将所选电池（n）的总充电时间（秒）返回给用户应用程序。

`unsigned int Get_Battery_Volts(unsigned char n)`

将所选电池（n）的浮充电压（mV）返回给用户应用程序。

`void Measure_Current(void)`

此函数用于测量正比较器输入端的电流检测电阻电压。

电流检测电阻值和放大率为已知值，因此很容易通过除法导出电流。

`void Measure_Charge(void)`

测量处于活动状态的电池的充电电压。仅当电池处于活动状态并且正在注入电流时，才能执行此操作。如果电流检测电阻为下桥臂类型，则必须从电池电压中减去压降才能获得正确值。

`void Measure_Float(void)`

测量处于活动状态的电池的浮充电压。要执行此操作，需关断电流和电池顶部的晶体管。还必须对连接了电池与模数转换器输入的 RC 滤波器放电，然后通过电池充电以获得正确值。充电时间取决于RC滤波器时间常数，并且应该至少为此时间常数的5倍（98.5%精度）。

`void Measure_Temp(void)`

测量与处于活动状态的电池连接的温度传感器上的电压。最好在关断电流源的情况下进行测量，以避免读数中掺杂噪声。

`void Switch_Battery(void)`

此函数非常重要，因为它用来切换至下一个电池插槽并检查电池状态，以决定是否应导通电流源。它还会使当前电池的总电池时间递增并使状态定时器递减。

NiMH.h中存在一个特殊定义，此定义可防止电池在达到结束状态后被切换。

**例11：**

```
#define FINISH_CHARGE_FULL_CURRENT
```

此定义实际上会将注入该电池的平均电流增加至设定限值的约90%。终止协议（如负ΔV或温度斜率）需要至少 C/2 的充电速率才能变得可靠，因此在单节电池上使用所有可用电流直至其中一个条件触发，对检测会有很大帮助。

## Hardware.h

这是非常重要的文件，因为所有初始化值均在此处定义。使用不同的硬件平台时，用户必须仔细修改这些值以匹配硬件。

**例12：**

```
#define    NiMH
#define    NiCd
#define    Li_ION
#define    NiZn
```

只能定义正确的电池类型（本例中为镍氢），因此应注释掉其他电池类型。此定义选择了适合电池化学成份的特殊头文件。

**例13：**

```
#define XTAL_FREQ             8000000
#define WORKING_FREQ   XTAL_FREQ / 4
```

必须定义晶振频率以进行正确的时序计算。对于PIC16/18系列，1个指令周期为4个振荡器周期。


© 2014 Microchip Technology Inc.

**例 14：**

```
#define T2_INIT                    0x7C
#define PR2_INIT                   24

#define T2_POSTSCALER              16
#define T2_FLAG_TICK WORKING_FREQ / (PR2_INIT + 1) / T2_POSTSCALER
```

Timer2 用于为开关电源生成 PWM 信号。此外，使用预分频器可以非常简单地生成较短时序，在本例中是PWM 斜坡时序以及用于电池电压读数的充电/放电时序。

**例 15：**

```
#define     PWM_PRECHARGE  (PR2_INIT+1)*4/4
#define     PWM_FAST       (PR2_INIT+1)*4/2
#define     PWM_FINISH     (PR2_INIT+1)*4/2
#define     PWM_TOPOFF     (PR2_INIT+1)*4/4
#define     PWM_MAINT      (PR2_INIT+1)*4/20
#define     PWM_TRICKLE    (PR2_INIT+1)*4/4
#define     PWM_OVERNIGHT  (PR2_INIT+1)*4/4
#define     PWM_OFF         0

#define     PWM_RAMP_STEP   1
```

使用降压转换器配置时，占空比不应超过50%。快速充电和结束充电阶段均具有允许的最大占空比。涓流充电/预充电阶段的占空比仅为最大占空比的¼，因为电流阈值较低。保持模式的占空比非常低，并且在开环不连续模式下工作。

由于大多数电源不能很好地响应负载阶跃，因此每次电池电流导通或关断时，占空比都会以特定步长上升或下降。

**例 16：**

```
#define     FTIMER_1SEC    (unsigned long)    T2_FLAG_TICK * 1000 / 1000
#define     FTIMER_1MSEC   (unsigned long)    T2_FLAG_TICK * 1 / 1000
#define     FTIMER_5MSEC   (unsigned long)    T2_FLAG_TICK * 5 / 1000
#define     FTIMER_10MSEC  (unsigned long)    T2_FLAG_TICK * 10 / 1000
#define     FTIMER_50MSEC  (unsigned long)    T2_FLAG_TICK * 50 / 1000
#define     FTIMER_60MSEC  (unsigned long)    T2_FLAG_TICK * 60 / 1000
#define     FTIMER_100MSEC (unsigned long)    T2_FLAG_TICK * 100 / 1000
```

后分频PWM时钟用于测量较短的时间间隔。

**例 17：**

```
#define            T1_INIT                0x31
#define            TIMER_OVERFLOW         TMR1IF

#define            ONE_SECOND             4
#define            LED_TIME               (ONE_SECOND / 4)
#define            TIMER_250_MSEC         WORKING_FREQ / 8 / ONE_SECOND
```

Timer1 用于为主应用程序生成 250 ms 的定时器计时周期。这是在 2 MIPS 下可生成的最长时间间隔，其中预分频比为8（62500 * 8 * 4 = 2000000）。由于显而易见的精度原因，应避免使用非整数除法。

**例18：**

```
#define    TRIS_INPUT     1
#define    TRIS_OUTPUT0

#define    PORTA_INIT     0b11101111
#define    PORTC_INIT     0b11110111
#define    TRISA_INIT     0b11011100
#define    TRISC_INIT     0b11001110

#define    ANSEL_INIT     0xEC

#define    PIE1_INIT      0x00
#define    INTCON_INIT    0x00

#define    CCP1CON_INIT   0x0F
#define    PWM1CON_INIT   0x80
#define    ECCPAS_INIT    0x15

#define    OPTION_INIT    0xF9

#define    VRCON_INIT     0xE0
#define    VRCON_TRICKLE  0xE1
#define    VRCON_FAST     0xE5
#define    VRCON_FINISH   0xE5

#define    CM1CON0_INIT   0x96
#define    CM2CON0_INIT   0x00
#define    CM2CON1_INIT   0x00

#define    ADCON0_MASK    0x81
#define    ADCON1_INIT    0x50
```

这些寄存器初始化在`InitializeHardware()`函数中使用。应修改这些寄存器初始化以匹配用户应用硬件。

**例19：**

```
#define LED_A          RC4
#define LED_B          RA5

#define BAT_A_SW       RA0
#define BAT_B_SW       RC0

#define BAT_ON         0
#define BAT_OFF        1

#define LED_OFF        0
#define LED_ON         1

#define TRUE           1
#define FALSE          0

#define BAT_A_TRIS     TRISA4
#define BAT_B_TRIS     TRISC3

#define BAT_A_VOLT     RA4
#define BAT_B_VOLT     RC3
```

与端口分配和方向相关的定义。

**例20：**

```
#define BAT_A_CHANNEL      0x03
#define BAT_B_CHANNEL      0x07
#define TEMP_A_CHANNEL     0x02
#define TEMP_B_CHANNEL     0x05
#define CURRENT_CHANNEL    0x06

#define CVREF              0x0C
#define VREF_06            0x0D
#define VREF_12            0x0E
```

对于每个电池插槽，都需要一个电压读数ADC通道和一个温度读数ADC通道。电流读数还需要一个额外的通道（与比较器输入复用）。内部参考具有单独的ADC通道。

**例21：**

```
#define ADC_CHARGE_DELAY              20

#define VREF1200                      1185
#define VREF600                       600

#define ADC_SAMPLES                   64
#define ADC_RESULT_SHIFT              6
#define ADC_OVERSAMPLE_MAX            65535
#define ADC_MAX                       1023
#define FILTER_BITS                   6
#define FILTER_MAX                    1023
#define FILTER_SAMPLES                1

#define CHARGE_VOLTAGE_OVERSAMPLING   64
#define CHARGE_VOLTAGE_SAMPLES        1
#define CHARGE_VOLTAGE_FILTER_BITS    6

#define FLOAT_VOLTAGE_SAMPLES         64
#define FLOAT_VOLTAGE_FILTER_BITS     0

#define CURRENT_OVERSAMPLING          64
#define CURRENT_FILTER_BITS           6
```

由于开关电源会使测量值存在大量噪声，因此需要使用较长的采样时间、过采样以及滤波。在10位ADC上，可使用最多支持64个采样的LP滤波器来获得16位值。强烈建议在使用负ΔV终止时对充电电压应用此滤波器。

**例22：**

```
#define SHUNT_AMPLIFICATION    10
#define SHUNT_RESISTANCE       50 //单位为mΩ
#define HIGH_SIDE_SENSING
```

电流检测电阻值和电压放大率决定了电流限值。如果电流检测的类型不是“上桥臂”，则必须从充电电压中减去电流检测电阻压降才能获得正确值。


© 2014 Microchip Technology Inc.

**例23：**

```
#define THERMISTOR_NTC

#define NTC_0C                (unsigned int)0xC353
#define NTC_25C               (unsigned int)0x7FFF
#define NTC_45C               (unsigned int)0x4E3B
#define NTC_50C               (unsigned int)0x43BC

#define TEMP_0C               ADC_OVERSAMPLE_MAX - NTC_0C
#define TEMP_25C              ADC_OVERSAMPLE_MAX - NTC_25C
#define TEMP_45C              ADC_OVERSAMPLE_MAX - NTC_45C
#define TEMP_50C              ADC_OVERSAMPLE_MAX - NTC_50C

#define MAX_CHARGE_TEMP       TEMP_50C
#define MAX_START_CHARGE_TEMP TEMP_50C

#define MIN_CHARGE_TEMP       TEMP_0C
#define MIN_START_CHARGE_TEMP TEMP_0C
```

示例电路板具有NTC型热敏电阻，温度点是使用器件数据手册计算的 16 位电压读数。为避免在含有“if (temperature < NTC_50C)”之类的语句（此语句实际上是检查温度是否大于给定值，而不是检查温度是否小于给定值）时发生混淆，用0xFFFF减去所有值。将NTC热敏电阻置于分压器的上桥臂上同样适用。

`unsigned int Measure_ADC(unsigned char channel, unsigned char samples)`

此函数用于获取所选ADC通道上给定的采样数，并返回总数。对于10位ADC，最大采样数为64。

`void InitializeHardware(void)`

使用*Hardware.h*文件中定义的值初始化外设。

`void Set_Current(unsigned int current)`

此函数用于设置开关电源的电流限值。在使用二次PWM或高分辨率DAC生成模拟电压的系统上，此函数可计算特定 mA 输入值所需的电压。示例电路板使用步长为200 mV的内部CVREF。在本例中，此函数用于接受PWM占空比值并相应地设置参考电压。

`void Bat_Switches_Off(void)`

关断所有电池补充充电开关。在正确测量电池浮充电压之前需要执行此操作。

**NiMH.h**

此头文件包含所有与镍氢化学电池相关的充电设置。与充电时间、终止协议以及特定故障条件有关的所有内容均在此处定义。

```
#define CHARGER_OVERNIGHT
#define CHARGER_FAST
```

充电器库可在以上其中一种模式下工作。涓流（通宵）充电器使用小电流持续12-16小时对电池充电。没有充电终止协议起作用。快速充电使用三个充电阶段，并且使用大得多的电流（通常介于C/2和C之间）。这是推荐大多数电池使用的充电方法。

**例24：**

```
#define CHG_TERM_NEG_DELTA_V_ENABLE
#define CHG_TERM_DELTA_TEMP_ENABLE
#define CHG_TERM_FLAT_DELTA_V_ENABLE

#define CHG_TERM_MIN_I_ENABLE
#define CHG_TERM_FLAT_DELTA_I_ENABLE
```

可在此处使能充电终止协议。负ΔV和稳定ΔV应一起变化。Δ 温度通常是非常可靠的终止协议，但也建议使能基于电压的协议，以防前者不触发。同时至少应使稳定ΔV有效。通常，首先触发的是温度，然后是负ΔV，最后是稳定ΔV。

基于电流的终止协议不适用于镍氢电池和镍镉电池。这类协议专门用于CC/CV类型的电池，如锂离子电池、镍锌电池或铅酸电池。

**例25：**

```
#define NDV_SAMPLES           15          //1个采样 = BATTERY_COUNT秒
#define MIN_I_SAMPLES         15
#define FDV_SAMPLES           15*60
#define FDI_SAMPLES           15*60
#define PEAK_UPDATE_SAMPLES   5
#define DT_TEMP_RISE_RATE     10          //每分钟的温升，以1/10C为单位
#define NDV_VOLTAGE_DROP      5           //NDV所需压降，以mV为单位
```

负 ΔV 需要充电电压峰值跟踪。如果当前电压高于最近记录的连续几次采样电压（上面定义），则更新峰值电压。类似地，如果充电电压比一定数量的采样的峰值电压低5 mV，则负ΔV将触发。如果一定数量的采样中没有新的峰值，则稳定ΔV将触发。

如果每分钟的温升至少为1.0°C，则Δ温度将触发。

由于每节电池的状态每BATTERY_COUNT秒刷新一次，因此很容易计算出条件触发所需的时间。

**例26：**

```
#define Z_TEST_LIMIT              200 //mΩ
#define INSTANT_CHARGE_CURRENT    800 //mA
#define CHARGE_MINUS_FLOAT_LIMIT  (unsigned long) INSTANT_CHARGE_CURRENT * Z_TEST_LIMIT / 1000
```

为了快速检测到充电器中插入损坏的镍氢电池或碱性电池的情况，执行阻抗测试非常重要。阻抗限值以 mΩ 为单位，瞬时最大充电电流限值以mA为单位。由于电压更易于读取，因此可推导出充电电压与浮充电压之间的最大允许差值。

**例27：**

```
#define MAX_BATTERY_CAPACITY_AAA      1000  //mAh
#define MAX_BATTERY_CAPACITY_AA       3000  //mAh

#define CHARGE_EFFICIENCY             125   //使电池恢复至100%充电级别需要125%功率

#define FINISH_CHARGE_TIME_FRACTION   33    //结束充电的最长时间为快速充电时间的33%

#define FAST_CHARGE_TIME_NOLOSS       (unsigned long) MAX_BATTERY_CAPACITY_AAA * 60 * 60 /
                                      INSTANT_CHARGE_CURRENT
```

充电时间应基于电池在特定充电速率下的充电接受能力以及最大电池容量。电流越高，充电接受能力越强。慢速充电器以C/10速率充满一节电池大约需要14-16小时。另一方面，快速充电器以 C 速率充满一节电池仅需要65-70分钟。

最大容量、充电效率和电流限值用于计算完全充电所需的最长时间。这有助于在所有充电终止协议均失效的情况下最大限度地缩短过充时间。结束充电时间定义为总充电时间的一部分。


© 2014 Microchip Technology Inc.

**例28：**

```
#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_TIME_LIMIT          60*60   //60分钟
#define MAINTENANCE_CHARGE_ENABLED
#define MAINTENANCE_CHARGE_TIME_LIMIT     360*60  //6小时

#define TRICKLE_CHARGE_ENABLED
#define TRICKLE_CHARGE_CURRENT            100
#define TRICKLE_CHARGE_TIME_LIMIT         6*60    //6分钟预充电时间

#define FAST_CHARGE_CURRENT               INSTANT_CHARGE_CURRENT / BATTERY_COUNT
#define FAST_CHARGE_TIME_LIMIT            (unsigned long) FAST_CHARGE_TIME_NOLOSS *
                                          CHARGE_EFFICIENCY / 100

#define FINISH_CHARGE_CURRENT             INSTANT_CHARGE_CURRENT
#define FINISH_CHARGE_TIME_LIMIT          (unsigned long) FAST_CHARGE_TIME_LIMIT *
                                          FINISH_CHARGE_TIME_FRACTION / 100

#define FINISH_CHARGE_TIME_RESULT         1       //1为良好；0为故障
#define FINISH_CHARGE_FULL_CURRENT

#define TOPOFF_CHARGE_ENABLED
#define TOPOFF_CHARGE_CURRENT             100
#define TOPOFF_CHARGE_TIME                60*60   //60分钟

#define OVERNIGHT_CHARGE_ENABLED
#define OVERNIGHT_CHARGE_CURRENT          100
#define OVERNIGHT_CHARGE_TIME             900*60  //15小时
```

对于每个使能的充电状态，都必须指定或计算出倒计时定时器。结束充电还有一个特殊选项，用于在定时器计数结束时仍未触发任何充电器终止协议的情况下将电池切换至故障模式。

**例29：**

```
#define BATTERY_REMOVAL_VOLTAGE       400
#define BATTERY_INSERTION_VOLTAGE     500
#define PRECHARGE_TERMINATION_VOLTAGE 1000
#define ALKALINE_DETECTION_VOLTAGE    1600
#define CHARGE_VOLTAGE_LIMIT          1700
#define FINISH_CHARGE_START_VOLTAGE   1470
#define FULL_BATTERY_VOLTAGE          1400
```

所有电压阈值均以 mV 为单位进行定义，但在内部会转换为16位电压读数（标称参考电压为5V）。

BATTERY_REMOVAL_VOLTAGE和BATTERY_INSERTION_VOLTAGE为电池存在阈值。

PRECHARGE_TERMINATION_VOLTAGE 是停止预充电并切换至快速充电的阈值。

ALKALINE_DETECTION_VOLTAGE 是使新检测到的电池直接进入故障模式的浮充电压阈值。

CHARGE_VOLTAGE_LIMIT是进入故障模式前所允许的最大充电电压。

FINISH_CHARGE_START_VOLTAGE 是用于切换至结束充电的充电电压阈值。

FULL_BATTERY_VOLTAGE是使新插入的电池直接进入完成状态且不进行充电（电池已满）的浮充电压。

## 配置库

为给定硬件平台配置库时需要做的第一件事是确认要求。本文的简介中详细列出了各种要求。所选器件需要带自动关断功能的ECCP外设、比较器、带多个通道的10位ADC以及通用IO引脚。建议使用内部或外部参考电压。

下一步是将Hardware.h中的配置寄存器初始化值设置为正确值。所有值均为定义，并在InitializeHardware()函数中使用。

首先需要定义将使用充电器的电池类型。默认为镍氢电池。目前不支持CC-CV充电（锂离子、铅酸或镍锌）。

**例30：**

```
#define NiMH
```

某些时序参数（如PWM频率或定时器计时周期预装载值）的自动计算需要定义内部运行频率和指令速度。

**例31：**

```
#define XTAL_FREQ   12000000
#define WORKING_FREQXTAL_FREQ / 4

#define T2_INIT     0x7C
#define PWM_FREQ    120000
#define PR2_INIT ((WORKING_FREQ / PWM_FREQ)-1)
```

每个充电状态的PWM占空比基于周期值进行计算。降压转换器的占空比不应超过50%。

通过使能PWM定时器的后分频器，使PWM定时器可轻松用于充电器应用中的时序。

**例32：**

```
#define T2_POSTSCALER 16
#define T2_FLAG_TICK WORKING_FREQ / (PR2_INIT + 1) / T2_POSTSCALER
```

为避免电源电压发生突变，导通或关断电流时PWM上会有一个斜坡。每次PWM定时器溢出时，都会将当前PWM值加上或减去（导通或关断）定义的值。不要忘记为定时器和后分频器值设置正确的配置寄存器值，以进行其他计算。

**例33：**

```
#define PWM_RAMP_STEP1
```

主16位定时器默认设置为100 ms，所有其他主充电器循环值均根据该值得出。Timer1预装载值也基于该值计算。LED指示提供两种简单的闪烁方案：0.5 Hz慢速闪烁和可配置的快速闪烁。

**例34：**

```
#define ONE_SECOND      10
#define LED_TIME        (ONE_SECOND / 2)
#define T1_PRESCALER    8
```

根据电池电压ADC输入端的RC滤波器中元件的值，使用PWM定时器设置充电和放电时间。对于100 kΩ/100 nF，RC常数为10 ms。

```
#define BAT_FILT_DISCHARGE_TIME FTIMER_1MSEC
#define BAT_FILT_CHARGE_TIME FTIMER_50MSEC
```

设置配置寄存器非常简单。但仍有几点需要注意。

首先，PWM必须配置为在达到最大电流时自动关断每个脉冲，并在下一周期重启。关断信号是与电流检测电阻和参考电压连接的比较器的输出。在本例中，参考电压为CVREF。

在确认电流环正常工作之前，切勿以最大占空比或CVREF最大值启动PWM。


© 2014 Microchip Technology Inc.

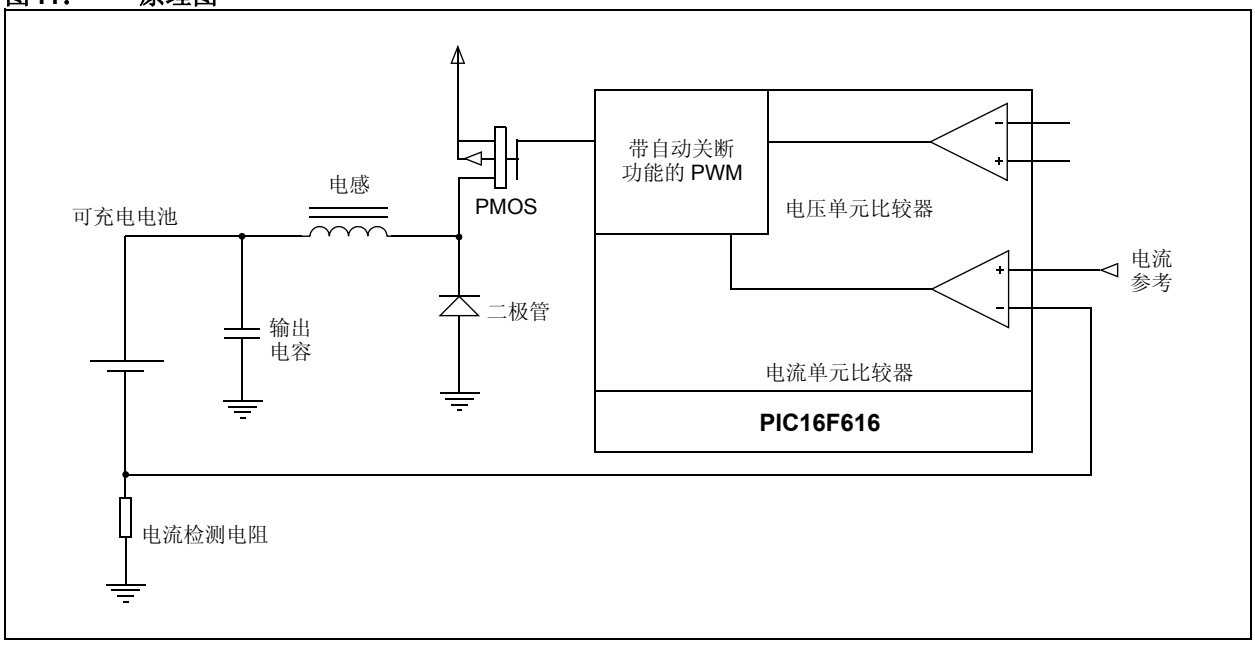
```
#define CCP1CON_INIT       0x0F
#define PWM1CON_INIT       0x80
#define ECCPAS_INIT        0x15
#define PSTRCON_INIT       0x02

#define CM1CON0_INIT       0x9D
#define CM2CON0_INIT       0x00
#define CM2CON1_INIT       0x00

#define REFCON0_INIT       0x90
#define REFCON1_INIT       0xE8
#define REFCON2_INIT       0x00

#define REFCON_TRICKLE     0x04
#define REFCON_FAST        0x11
#define REFCON_FINISH      0x11
```

**图11：　原理图**




第二个问题与ADC参考有关。VDD通常不够精确或不够稳定，需要某种参考电压。如果使用外部参考，则无需进一步补偿。但是，如果需要准确的VDD值，仍必须在规定的通道上读取参考，同时以VDD为参考推导出正确的电源值。

如果参考在内部或者因为任何原因不能用作ADC的VREF，则每次导通或关断电流时都必须读取该参考。由于参考应该在较大的VDD范围内保持稳定，因此很容易推导出正确的VDD值并根据参考值补偿其他ADC读数。

**例35：**

```
#define ADCON0_MASK    0x01
#define ADCON1_INIT    0x04
#define ADCON1_VDD     0x00
#define ADCON2_INIT    0x85
```

必须正确定义电池电压、温度和当前VREF的ADC通道。外部 VREF 通常是标记为 VREF+ 的通道。其余定义取决于使用的参考类型。

**例36：**

```
#define BAT_A_CHANNEL      0x07
#define BAT_B_CHANNEL      0x08
#define TEMP_A_CHANNEL     0x09
#define TEMP_B_CHANNEL     0x09
#define CURRENT_CHANNEL    0x05

#define FVREF_CHANNEL      0x0E
#define DAC_CHANNEL        0x0F
#define VREF_EXT_CHANNEL   0x04
```

当ADC使用VDD作为参考时，应取消注释 #COMPENSATE_VDD 定义，并且所有读数都将归一化为5.000V的标称VDD。此换算的精度取决于参考电压的精度。

其余寄存器主要是端口、三态和模拟功能设置。

**例37：**

```
#define PORTA_INIT     0b00000000
#define PORTC_INIT     0b10110111
#define TRISA_INIT     0b11111111
#define TRISB_INIT     0b00001111
#define TRISC_INIT     0b11001011

#define ANSEL_INIT     0b10110000
#define ANSELH_INIT    0b00000011

#define PIE1_INIT      0x00
#define INTCON_INIT    0x00

#define TXSTA_INIT     0b10100100
#define RCSTA_INIT     0b10000000
#define BAUDCON_INIT   0b00001000
#define SPBRGH_INIT    0
#define SPBRG_INIT     77

#define WDTCON_INIT    0x00
```

限流环的电流检测电阻通常较小，在50-100 mΩ范围内。必须定义该值，以供其他与瞬时电流和电池阻抗有关的计算使用。电流检测电阻上的电压在馈入比较器之前会经过运算放大器放大。

**例38：**

```
#define    SHUNT_AMPLIFICATION    10
#define    SHUNT_RESISTANCE       56
```

如果电流检测电阻为上桥臂型，则取消注释此行。否则保留注释，因为会从电池充电电压中减去电流检测电阻的压降。

**例39：**

```
#define HIGH_SIDE_SENSING
```

如果电路板配有用于温度关断的热敏电阻，则此定义为类型定义。取消 NTC 的注释并保留 PTC 的注释。NTC 热敏电阻在较高温度下的阻抗较低，因此某些位置的代码可能看起来不合规则。

例如，如果直接使用 NTC 读数，“`if(temperature < TEMP_40C)`”将在温度高于40°C时触发，而不是在温度低于40°C时触发。

用最大 ADC 值减去所有 NTC 热敏电阻读数，以便使阻抗看起来随温度的升高而增大。

**例40：**

```
#define THERMISTOR_NTC
```

通常，热敏电阻仅用于检测环境条件。如果温度低于0°C或高于40°C，充电器完全不会开始充电（并进入故障模式）。此外，如果充电时温度高于50°C，则将产生故障条件。

在热敏电阻与电池之间使用紧密的热耦合（每节电池需要一个热敏电阻），可检测充电周期最后一部分中的陡峭斜坡，并借此在发生过充之前终止充电。这是一项非常好的充电终止技术，但代价是需要更复杂的机械设置才能在传感器与电池之间实现紧密的热接触。需要实现可提供0.1°C分辨率的温度计算功能。

**例41：**

```
#define MIN_START_CHARGE_TEMP     TEMP_0C
#define MIN_CHARGE_TEMP           TEMP_0C
#define MAX_START_CHARGE_TEMP     TEMP_40C
#define MAX_CHARGE_TEMP           TEMP_50C
```

与电池相关的定义存储在`Ni-MH.h`文件中。检查并修改这些值非常重要，因此这些值决定了每种电池状态下充电定时器的计算值。特定于电池化学成份的阈值和所用的充电电流也在此文件中，必须全面检查。

首先需要配置的是充电器行为。可选择快速充电或慢速/通宵充电。如果同时定义这两种行为，则快速充电优先。

**例42：**

```
#define CHARGER_OVERNIGHT
#define CHARGER_FAST
```


© 2014 Microchip Technology Inc.

除了显而易见的定时器、电压和温度故障保护机制，还有两个主要的基于电压的触发用于快速充电终止。强烈建议在快速充电时保持负ΔV和稳定ΔV有效。

**例43：**

```
#define CHG_TERM_NEG_DELTA_V_ENABLE
#define CHG_TERM_FLAT_DELTA_V_ENABLE
```

为避免出现与噪声相关的问题，充电终止协议具有在达到0时触发的特殊计数器。例如，如果最近15次采样的充电电压比上一次记录的峰值低至少5 mV，则负ΔV触发。此外，仅当最近5次采样的充电电压高于上一次记录的峰值时，才会更新峰值。

稳定ΔV的触发时间更长，设置为15分钟。如果递减条件在任何时候都不为真，则计数器值将恢复为初始值。

务必记住，电池一次充电一秒，除非使能结束充电全电流选项；下列值实际上乘以了电池插槽数（时间相关）。

**例44：**

```
#define NDV_SAMPLES           15
#define FDV_SAMPLES           15*60
#define PEAK_UPDATE_SAMPLES   5
#define NDV_VOLTAGE_DROP      5
```

阻抗测试的执行方式非常简单。将最大允许阻抗（以mΩ为单位）与瞬时快速充电电流相乘可获得充电电压与浮充电压之间的最大允许差值。更精确的方式是在每次检查阻抗故障条件时都将电流读数与最大阻抗相乘。当然，这种长整型乘法会占用一些额外的存储器。

**例45：**

```
#define Z_TEST_LIMIT              150
#define INSTANT_CHARGE_CURRENT    850
#define CHARGE_MINUS_FLOAT_LIMIT  (unsigned long) INSTANT_CHARGE_CURRENT * Z_TEST_LIMIT / 1000
```

快速充电和结束充电定时器均使用充电器允许的最大电池容量自动进行计算。如果使用的容量较大，将无法完成充电。使用典型效率80%（125%功率）计算实际定时器值。此外，结束充电时间定义为不超过计算出的快速充电时间的33%。

**例46：**

```
#define MAX_BATTERY_CAPACITY_AAA      1000
#define MAX_BATTERY_CAPACITY_AA       3000

#define CHARGE_EFFICIENCY_INV         125
#define FINISH_CHARGE_TIME_FRACTION   33

#define FAST_CHARGE_TIME_NOLOSS       (unsigned long) MAX_BATTERY_CAPACITY_AAA * 60 * 60 /
                                      INSTANT_CHARGE_CURRENT

#define FAST_CHARGE_TIME_LIMIT        (unsigned long) FAST_CHARGE_TIME_NOLOSS *
                                      CHARGE_EFFICIENCY_INV / 100

#define FINISH_CHARGE_TIME_LIMIT      (unsigned long) FAST_CHARGE_TIME_LIMIT *
                                      FINISH_CHARGE_TIME_FRACTION /         100
```

通宵充电时间和渐止（如果适用）充电时间为手动声明。

**例47：**

```
#define TOPOFF_CHARGE_TIME    60*60
#define OVERNIGHT_CHARGE_TIME 900*60
```

一个非常重要的#define，用于选择结束充电是否将所有可用电流输入一节电池，直至一种充电终止协议触发。这对于获取可靠的负 ΔV 波形非常有帮助，但可能会产生额外的热量。首先，检查平均充电电流（瞬时电流 / 电池插槽数）是否高于电池的 C/2 电流。在此例中，瞬时电流为850 mA，电池插槽数为2。在结束充电中不使用全电流对 750 mAh 电池充电，应允许可靠的 –dV 检测。但请记住，较小的充电电流会使–dV 波形进一步延时到过充区域中（即使电池未变热）。

**例 48：**

```
#define FINISH_CHARGE_FULL_CURRENT
```

一组特定于镍氢化学电池（以及更小的AAA电池型号）的电压阈值用于转换充电状态，以检测错误的电池类型和其他故障。可以修改这些电压阈值以满足应用需求，如有疑问，可从电池制造商处获取这些值或与之进行讨论。所有值均以mV为单位。

电池移除和检测非常简单。

**例 49：**

```
#define BATTERY_REMOVAL_VOLTAGE   400
#define BATTERY_INSERTION_VOLTAGE 500
```

深度放电的电池不能进行快速充电，因为高内阻会导致发热。应改为进行涓流充电，直到电压达到1.0V。

**例 50：**

```
#define PRECHARGE_TERMINATION_VOLTAGE 1000
```

通常，如果插入电池时浮充电压非常高，则表示电池类型错误。充电器会将插槽置于故障模式。

**例 51：**

```
#define ALKALINE_DETECTION_VOLTAGE 1600
```

最大充电电压限值为 1.7V。如果使用较高的充电电流，此限值可能会发生变化。

**例 52：**

```
#define CHARGE_VOLTAGE_LIMIT 1700
```

从快速充电到结束充电的转换电压设置为1470 mV。此值也可能随充电电流变化，但不要设置过高，否则电池可能不再改变状态。这种情况并不好，因为快速充电没有任何充电终止协议。

**例 53：**

```
#define FINISH_CHARGE_START_VOLTAGE 1470
```

插入电池时浮充电压较高（但不足以导致故障）通常意味着可充电电池已满或者插入了部分放电的碱性电池。充电器会自动将插槽置于完成模式。不会进行充电。

**例 54：**

```
#define FULL_BATTERY_VOLTAGE 1400
```

已放电的电池具有较高内阻，因此有时建议在电池达到最低充电电压后检查阻抗故障。

**例 55：**

```
#define IMPEDANCE_CHECK_VOLTAGE   1200
```


© 2014 Microchip Technology Inc.

## 结论

Microchip电池充电器库允许用户快速向其应用添加充电功能。其硬件非常简单，并且可使用一个电流源为多节电池充电，因此极具成本效益。虽然没有适用于各类充电电池的神奇配方，而且每个制造商都有自己的一套建议，但该库仍具有高度可定制性，可轻松进行重新配置以匹配各类充电电池。

无论是独立充电器还是更复杂的设备，本电池充电器库都可被证明是便携式应用中很有价值的工具。

注：


© 2014 Microchip Technology Inc.

请注意以下有关 **Microchip** 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

**提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。**

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

**商标**

Microchip 的名称和徽标组合、 Microchip 徽标、 dsPIC、 FlashFlex、 KEELOQ、 KEELOQ 徽标、 MPLAB、 PIC、 PICmicro、 PICSTART、 PIC32 徽标、 rfPIC、 SST、 SST 徽标、 SuperFlash 和 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

FilterLab、 Hampshire、 HI-TECH C、 Linear Active Thermistor、 MTP、 SEEVAL 和 The Embedded Control Solutions Company 均为 Microchip Technology Inc. 在美国的注册商标。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

Analog-for-the-Digital Age、 Application Maestro、 BodyCom、 chipKIT、 chipKIT 徽标、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 dsPICworks、 dsSPEAK、 ECAN、 ECONOMONITOR、 FanSense、 HI-TIDE、 In-Circuit Serial Programming、 ICSP、 Mindi、 MiWi、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 mTouch、 Omniscient Code Generation、 PICC、 PICC-18、 PICDEM、 PICDEM.net、 PICkit、 PICtail、 REAL ICE、 rfLAB、 Select Mode、 SQI、 Serial Quad I/O、 Total Endurance、 TSHARC、 UniWinDriver、 WiperLock、 ZENA 和 Z-Scale 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Inc. 在美国的服务标记。

GestIC 和 ULPP 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2014, Microchip Technology Inc. 版权所有。

ISBN：978-1-63276-077-7

**QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV**

**ISO/TS 16949**

*Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。*

# 全球销售及服务网点

## 美洲

**公司总部 Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

**亚特兰大 Atlanta**
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

**奥斯汀 Austin, TX**
Tel: 1-512-257-3370

**波士顿 Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

**芝加哥 Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

**克里夫兰 Cleveland**
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

**达拉斯 Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

**底特律 Detroit**
Novi, MI
Tel: 1-248-848-4000

**休斯敦 Houston, TX**
Tel: 1-281-894-5983

**印第安纳波利斯 Indianapolis**
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453

**洛杉矶 Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

**纽约 New York, NY**
Tel: 1-631-435-6000

**圣何塞 San Jose, CA**
Tel: 1-408-735-9110

**加拿大多伦多 Toronto**
Tel: 1-905-673-0699
Fax: 1-905-673-6509

## 亚太地区

**亚太总部 Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

**中国 - 北京**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**中国 - 成都**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**中国 - 重庆**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**中国 - 杭州**
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

**中国 - 香港特别行政区**
Tel: 852-2943-5100
Fax: 852-2401-3431

**中国 - 南京**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**中国 - 青岛**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**中国 - 上海**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**中国 - 沈阳**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**中国 - 深圳**
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

**中国 - 武汉**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**中国 - 西安**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**中国 - 厦门**
Tel: 86-592-238-8138
Fax: 86-592-238-8130

**中国 - 珠海**
Tel: 86-756-321-0040
Fax: 86-756-321-0049

## 亚太地区

**台湾地区 - 高雄**
Tel: 886-7-213-7830

**台湾地区 - 台北**
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

**台湾地区 - 新竹**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**澳大利亚 Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**印度 India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**印度 India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**印度 India - Pune**
Tel: 91-20-3019-1500

**日本 Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**日本 Japan - Tokyo**
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

**韩国 Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**韩国 Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

**马来西亚 Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**马来西亚 Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**菲律宾 Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**新加坡 Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**泰国 Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## 欧洲

**奥地利 Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**丹麦 Denmark-Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**法国 France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**德国 Germany - Dusseldorf**
Tel: 49-2129-3766400

**德国 Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**德国 Germany - Pforzheim**
Tel: 49-7231-424750

**意大利 Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**意大利 Italy - Venice**
Tel: 39-049-7625286

**荷兰 Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**波兰 Poland - Warsaw**
Tel: 48-22-3325737

**西班牙 Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**瑞典 Sweden - Stockholm**
Tel: 46-8-5090-4654

**英国 UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820

03/25/14


© 2014 Microchip Technology Inc.