



G+ Mass Production Tool For GPL329xx 使用手册

V1.0.1 – 2011 年 8 月 12 日



Important Notice

Generalplus Technology reserves the right to change this documentation without prior notice. Information provided by Generalplus Technology is believed to be accurate and reliable. However, Generalplus Technology makes no warranty for any errors which may appear in this document. Contact Generalplus Technology to obtain the latest version of device specifications before placing your order. No responsibility is assumed by Generalplus Technology for any infringement of patent or other rights of third parties which may result from its use. In addition, Generalplus products are not authorized for use as critical components in life support devices/systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of Generalplus.



目 录

页码

G+ MASS PRODUCTION TOOL FOR GPL329XX.....	1
使用手册.....	1
1 TOOL简介	6
1.1 USB 设备端口映射对话框	6
1.2 主窗口界面	7
1.2.1 脚本配置按钮.....	8
1.2.2 打开按钮.....	8
1.2.3 开始下载按钮.....	8
1.2.4 枚举设备按钮.....	9
1.3 脚本文件配置流程	9
1.4 步骤操作概述.....	12
1.4.1 选择脚本.....	13
1.4.2 使能步骤.....	14
1.4.3 创建步骤.....	14
1.4.4 删除步骤.....	14
1.4.5 调整步骤顺序.....	15
1.5 步骤配置介绍.....	15
1.5.1 读步骤.....	15
1.5.2 写步骤.....	16
1.5.3 代码区与数据区步骤	17
1.5.4 Send Nand Database	18
1.5.5 代码区头文件解析步骤.....	19
1.5.6 引导区头文件解析步骤.....	20
1.5.7 BootHeader烧录步骤.....	21
1.5.8 BootLoader烧录步骤	22
1.5.9 Chip Erase 步骤.....	23
1.5.10 Erase步骤.....	24
1.5.11 获取Nand信息步骤	25
1.5.12 主机端延迟步骤	26
1.5.13 跳转步骤.....	27



1.5.14	Nand Flush 步骤.....	28
1.5.15	寄存器写步骤.....	29
1.5.16	Dram Calibration步骤.....	30
1.5.17	用户自定义命令步骤.....	30
1.5.18	Lua Script.....	32
1.6	下载.....	33



Revision History

Revision	Date	By	Remark
1.0.1	07/18/2011	Phoebe	1. 支援 Lua Script 2. 输入值支援 10 进制和 16 进制 3. 添加 Auto Mode, 通过 Dram Calibration、Send Nand Database、App Header Parsing 步骤, 在 download 过程中使相关内容进行自动调整。
1.0.0	05/12/2011	Eric	1st Version

1 Tool 简介

当用 G+Code packer 包出 binary 文件或 hdb 文件后, 用户就可以呼叫 G+ Mass Production Tool For GPL329xx 来对指定的 memory type 进行烧录. 尽管如此, 用户需要先配置好整个烧录的流程, 即透过此软体进行下载步骤的配置.

所以, 在执行烧录的动作之前, 用户应确保下载脚本文件已正确创建, 原则上, 则文件的默认创建路径是在与可执行档平级的名为 **conf** 的文件夹下. 当然, 此文件的档名由用户自行拟定. 我们会在后续的章节中来更详细的描述此脚本文件的创建过程.

1.1 USB 设备端口映射对话框

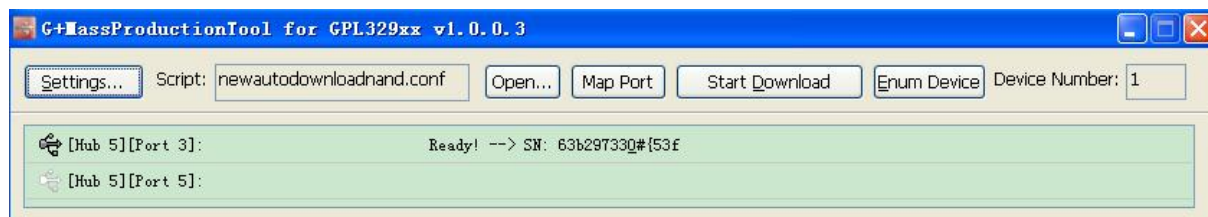
目前, 此 Tool 会透过这个设备端口映射对话框来对 PC 机上所装载的 Generalplus USB 设备进行控管, 假定一开始, Tool 没有检测到 GP USB 设备, 这时用户接入一台 GP USB 设备, Tool 如果枚举到此设备, 就会跳出一个对话框, 来询问用户是否要将此设备显示在 Tool 的主界面上, 同时刷新端口映射表. 若用户选是, 将会执行上述行为, 若选否, 那 Tool 将取消上述操作, 不会将 tool 枚举到的设备放到主界面的设备列表中, 当然端口映射表也不会随之更新. 下面就是端口映射表的所长的样子, 里面有棵树.



用户能透过此树来很清楚的查看到当前接入 PC 的所有设备状况，它可以很明确反映用户，说某个设备挂接在某个集线器的某个端口上，因为 tool 会在每个集线器的每个端口上打印出当前端口所接入的设备信息。有了这些信息，如果用户可以勾选对应的设备选项，表明用户想对此台设备对行烧录，当用户点击 **Set** 按钮后，如果用户选中的设备是能跟此 Tool 沟通的 USB 设备，那么此设备信息将会列在 Tool 的主界面的设备列表中。当然,如果用户选中的设备不是 GP USB 设备，那么，主界面的设备列表也不会将用户在端口映射表选中的设备信息显示出来。

1.2 主窗口界面

当用户配置好端口映射表后，将会返回到主对话框，下面就是主对话框的界面：



有此主窗口界面中，用很多比较简单实用的功能：用户可以透过 **Setting** 按钮来配置下

载脚本；可以点选 **Open** 按钮不用进入脚本配置页面就可以选择所以烧录的脚本，前提是你比较熟悉此脚本所做的事情：至于 **Port map** 按钮，前面已经描述过，当然用户也可以透过此功能来移除当前主界面设备列表中不感兴或不关心的设备，即便此设备有被 **Tool** 枚举到。再有就是透过 **Start Download** 按钮来实现对脚本文件中所指定的步骤进行烧录，至于 **Device Number** 标签,是标示当前设备列表中的设备数目，也就是你真正想烧录的设备数量，而非真正 **Tool** 所枚举到的设备个数,当然，大多情况下是一样的，需要根据用户自己需要而定。

1.2.1 脚本配置按钮

正如大家所知道的，用户可以透过点选 **port map** 按钮来改变设备列表中烧录设备的数目，那么，用户可以同样透过点选 **Setting** 按钮来实现在脚本文件的配置，此文件主要用于配置下载的步骤，以及每个步骤当中所需要的相关参数，诸如要烧录的 **bin** 文件，烧录的存储器型别，及设备起始地址等等...具体详细参数,我们会在稍后每个步骤的配置过程中再另加描述...这里只要知道，用户可以透过它来选择当前所要烧录的 **CPU** 型别，及此种型别下所配置的烧录流程。

1.2.2 打开按钮

一般而言，如果用户不需要对脚本配置文件频繁是做修改的话，那么用户可以透过此按钮，选择用户自己期望的下载脚本。这样，对于工程开发人员而言，就会节约大量时间，从而给此类用户提供更多便利，毕竟有了此按钮,用户就不必每次切换下载脚本时,还要跑到 **Setting** 对话框页面，再来选择自己想要下载的脚本文件。

1.2.3 开始下载按钮

此按钮主要是给使用者提供 **code** 的下载功能，同时也是这个 **Tool** 比较重要的一个环节，最主要是它能支援多台设备并行下载的功能，就是说使用者在配置好下载脚本

后, 按下此按钮, 将会触发一个下载的事件, Tool 在感知此事件后, 会对设备列表中的所有处于有效就绪状态的设备烧录, 让它们同时都去做同一件事, 即按照同一份脚本配置文件对指定的流程来进行执行. Tool 会将脚本配置文件中的每个步骤都执行完毕. 而主界面也会及时反映每台设备烧录状况, 烧录过程中, 使用者随时可以查看每台设备烧录的日志信息, 以及在 UI 直观看到每台设备所完成的进度及相应的状态信息, 假定下载完毕, 会显示当前此设备完成进度为 100% 及状态文本 **Download finished**, 倘若其中某台或某几台设备在烧录中执行到某个步骤发生了错误, 那么 Tool 会将此错误信息打印在设备列表中, 告知使用者哪个集线器下哪个端口下的那台设备烧录失败, 其失败原因会显示在设备列表中, 同时会将此信息在日志档中记录一份. 当然, 不同的状态信息, Tool 会以不同的颜色加以区分, 让使用者一目了然.

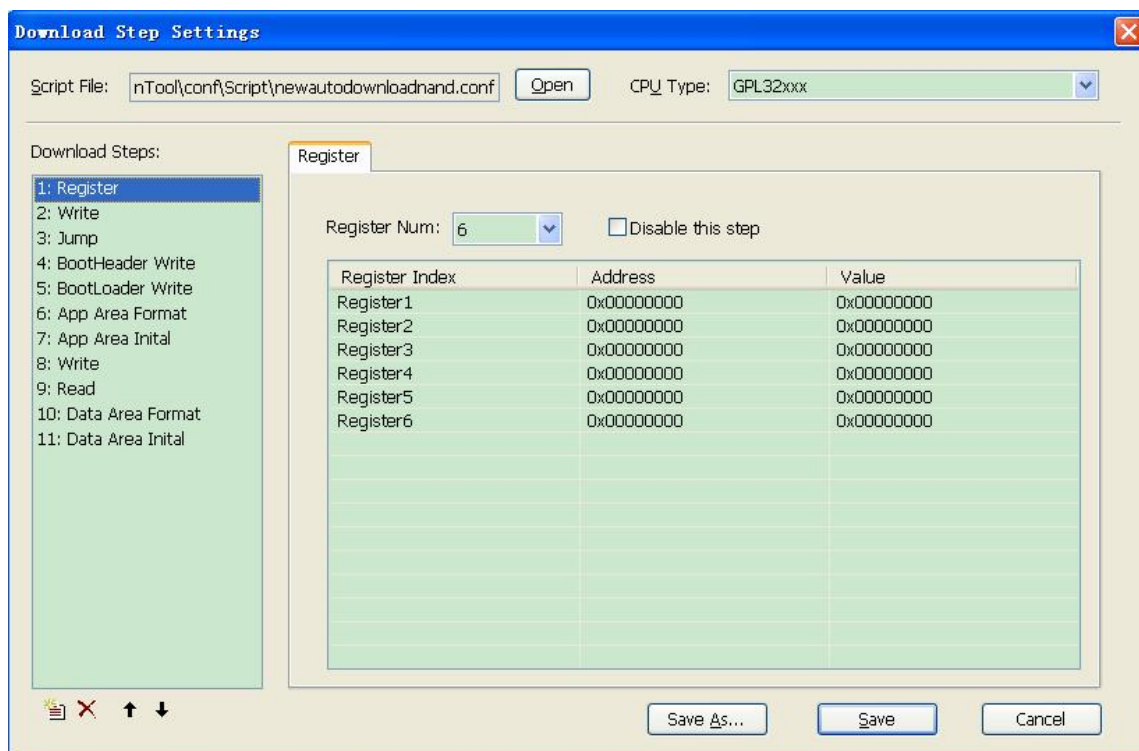
1.2.4 枚举设备按钮

这个按钮主要是提供设备枚举的功能, 其实在大多数情况下, 这个功能不是很必需的, 那是因为 Tool 内部有做自动侦测并连接设备的功能. 所以即使用户不点击此按钮, Tool 也一样会识别到 GP USB 设备, 只是说, 有些时候, PC 透过操作系统检测到了我们接入的设备, 而 Tool 的自动检测机制并没有察觉到此设备, 或者说我们不确定 Tool 本身自动连接功能是否有生效, 这时, 用户就可以尝试去点击此按钮, 来检查当前接入的 USB 设备到底有没有被 Tool 认出. 若有认出, 它会将此设备信息显示在设备列表中, 让用户看到; 若跳出 "No Device be Found." 的消息框, 那表示说 Tool 还没有识别到此台设备.

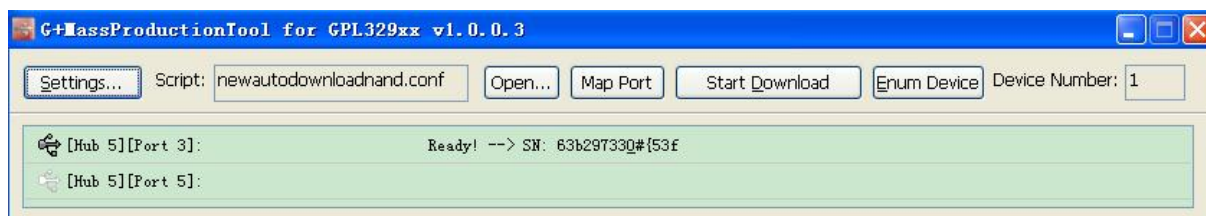
1.3 脚本文件配置流程

在这一节, 我们将简要介绍下脚本文件的配置过程, 在此之前, 先来说下 PC 是如何 Tool 是如何枚举到设备的, 事实上, 我们在前面的章节中就有提到, 对用户而言, 这个将是透明的, 所以从操作上就显得更加简单, 使用者仅仅需要点击 **Enum Device** 按

钮, 既然如此, Tool 主界面窗口中的 **Setting** 又是如何工作的呢? 它主要是对我们的下载脚本文件进行设置的, 当用户点选 **Setting** 按钮后, 将会跟跳出设置下载脚本的画面, 用户透过此页面, 将会配置基于某种 **CPU** 型别的烧录流程文件, 你也可以选择在磁碟上现存的脚本文件, 只需指定相应的路径, Tool 将会自动加载用户指定的脚本文件, 有点想说明下, 理论上, 同一 Tool 的不同发行版本所创建的脚本是相互兼容的, 但是我们目前不建议这样的混用, 特别是如果新旧两个版本时间相差很远, 那么就可能会造成旧的脚本文件在新的发行版本上不能很好的工作, 可能会本次烧录达不到预期效果, 相信随着时间推移, 兼容性的问题将会得到更好的解决. 另外, 这些文件是由需要用户自行配置, 一般由工程人员配置完成, 并经测试, 再将此脚本文件连同每个步骤所用到的烧录 **bin** 档或者是 **hdb** 档, 封成一个包, 由生产方的相关操作人员解包, 进行大量设备的烧录, 以达到量产之功效, 所以对工程开发人员来讲, 掌握下载脚本的正确配置, 无疑是 Tool 所关注的重点, 尽管如此, 工程开发人员在此页面所作的任何配置或对现在设置选项参数所作的修改, 都将会被保存起来, 同时要注意的是, **CPU** 型别的切换, 将直接影响到设备枚举, 因为不同 **CPU** 型别, 所对应的枚举参数是不同的, Tool 会参考这些参数信息, 后续会增加不同 **CPU** 型别, 使这个 Tool 的应用更加广泛. 目前主要是支援 32bit 的 **CPU**. 完成这些配置信息的设置之后, 用户就可以点选 **Save** 按钮, 从而使本次配置生效,



完成了脚本文件的保存，就回到主窗口界面，如果当前设备列表中所接入的设备都处于就绪状态，当用户按下 **Start Download** 按钮后，外于就绪态的设备就会开始依据下载脚本来执行烧录动作。而设备就会由就绪态切换到下载态，而之前的 **Start Download** 按钮会切换到 **Stop** 状态，让用户可以对处于烧录过程中的设备进行控制，要求设备列表中的每台设备终止此次烧录动作。另外，用户也可以敲键盘上的回车键，就相当用鼠标点击 **Start Download** 按钮，具有同等功效。

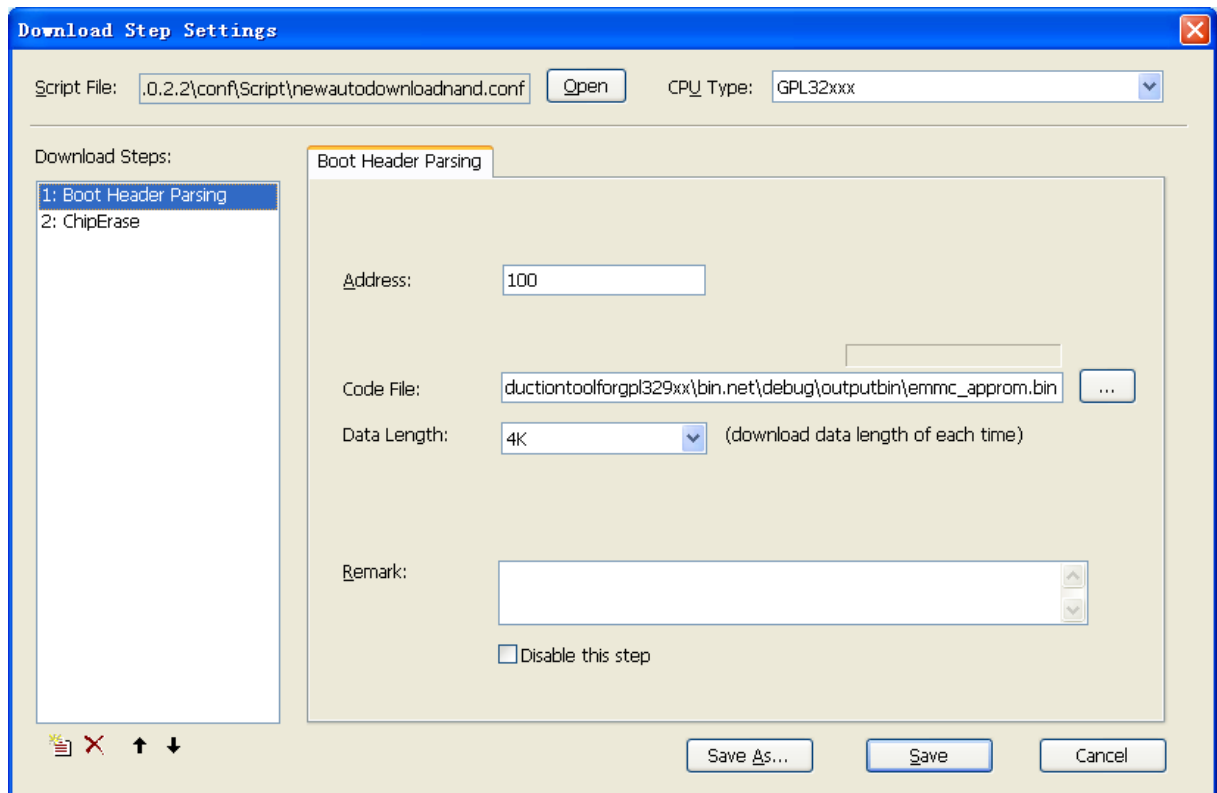


从上面的图中，我们不难发现，此主窗口画面中，有个设备列表，里面装着用户当前在端口映射表中选定的设备配置，显然用户勾选了两台设备，PC 只检测到一台，当

然, 这里只是举例, 来说明此设备列表受控于端口映射表的选项. 此设备列表中, 一个集线器下的那个端口未发现当前配置所支援的设备, 所以也就没有显示任何的设备信息; 而另外枚举到那台, 里面有集线器的索引号, 端口的索引号, 设备当前状态及设备的序列号, Tool 会依据这些集线器的索引号, 端口的索引号, 设备的序列号等资讯在后台为每台设备创建一个日志文件, 这样无论烧录动作是成功完成还是失败而终止, 都会在此日志文件中详细记录. 当然界面上也会同步将烧录状况显示出来.

1.4 步骤操作概述

整个下载脚本文件就是由 CPU 型别及许多步骤组成, 所以工程开发人员若能了解 Tool 对烧录脚本的配置是基于步骤的观念, 那么将会使此下载脚本配置起来显得更加容易, 也更加方便, 因为我们将用户所要完成的一个任务, 分解成许多个步骤来完成, 当所有的步骤都执行完毕, 也就意味着本次用户的请求或下达的任务已成功完成. 下面就对脚本配置中的步骤进行逐一描述;



首先，我们会根据每个步骤的名字来为其提供对应的用户界面，那么在列表框控件中，将会显示用户当前已创建的好的步骤，倘若用户对这些步骤的名字感觉不够直观，那么我们已提供为每个步骤的名字修改的功能，允许用户对其重命名，操作方法是在选中的步骤名称，用鼠标双击，便会出现一个编辑框，用户输入完毕，回车或用鼠标点击其它地方，让此编辑框失去焦点，以告知 **Tool**，当前输入已结束。顺便说一声，之后设备在下载过程中用于记录烧录状况的日志中也将同步更新，会改用使用者自定义的名称加以记录，


1.4.1 选择脚本

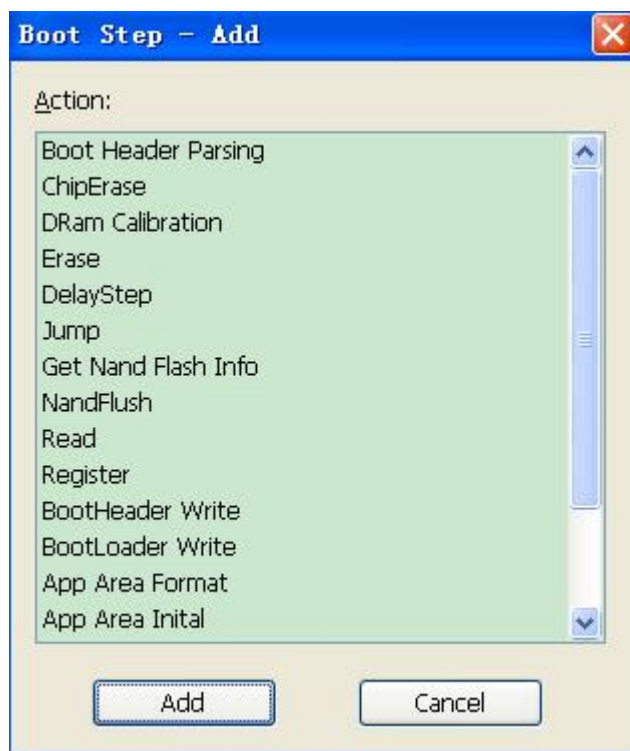
点击 **Open** 按钮，然后选择用户自己想要保存的脚本配置文件。若此文件存在，则表明使用者是想对现有步骤进行修改，否则，**Tool** 会创建一个空的脚本文件。

1.4.2 使能步骤

使能用户当前所选中的步骤，若勾选，则会使该步骤失效，那么实际执行过程中，就会忽略此步骤，而会跳过它执行它后面的步骤。

1.4.3 创建步骤

单击  这个按钮，将会跳出步骤添加的对话框，其画面如下：




此对话框中的列表框控件中会将 Tool 目前支持的所有步骤名称显示出来。

1.4.4 删除步骤

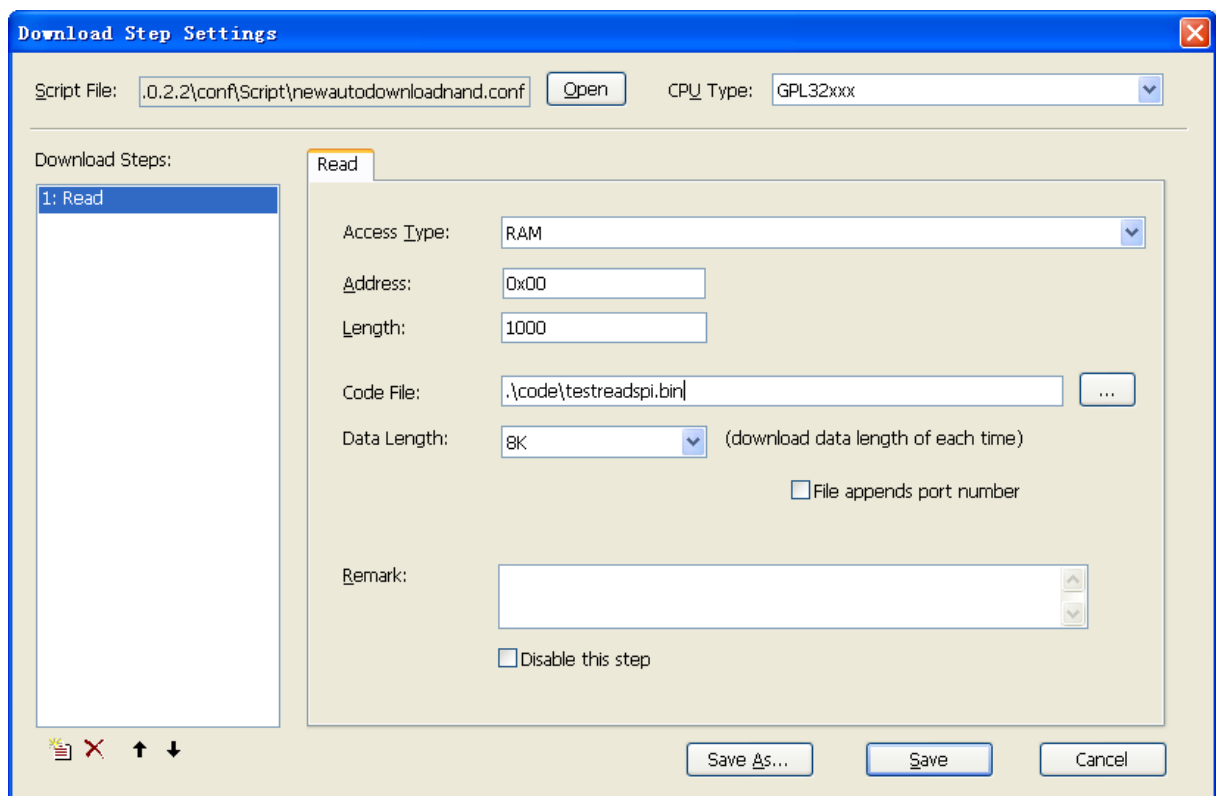
点击  按钮，将会删除用户当前选中的步骤。

1.4.5 调整步骤顺序

单击  按钮, 将会调整用户当前选中步骤在整个步骤列表中的顺序

1.5 步骤配置介绍

1.5.1 读步骤

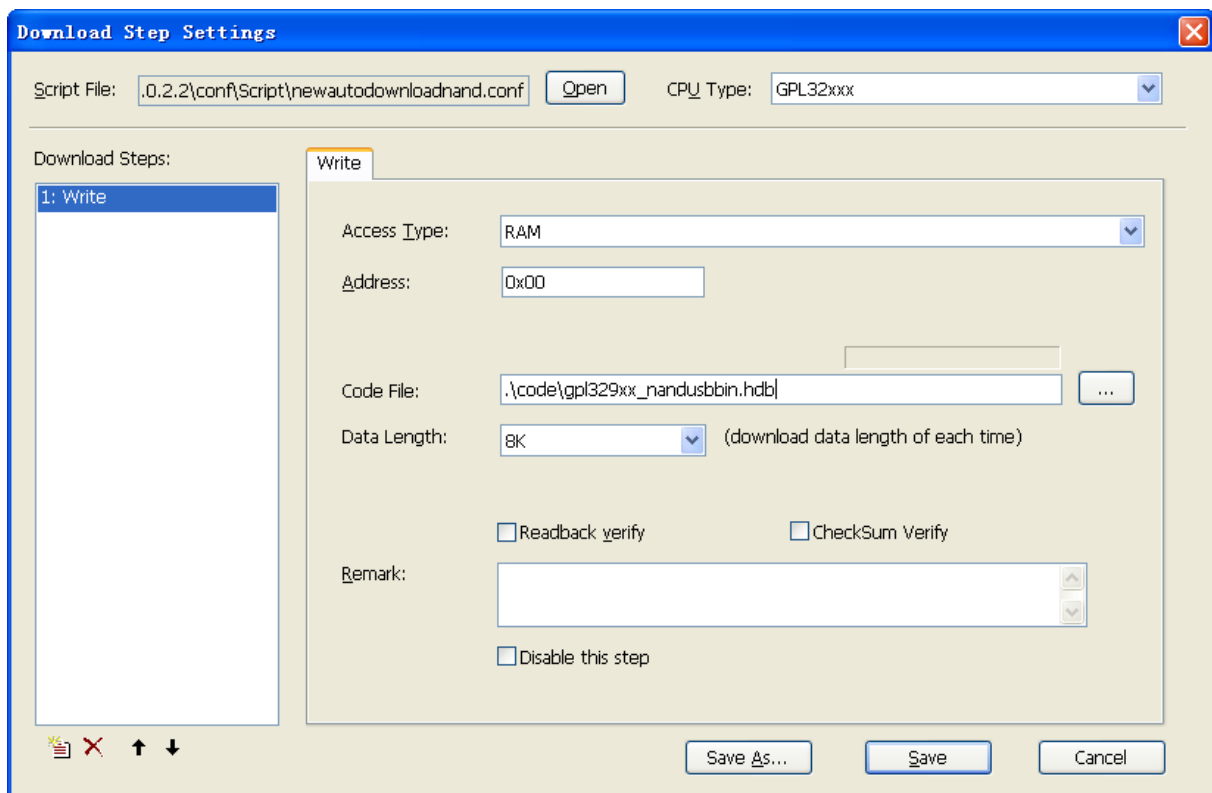


当用户在脚本配置画面左侧的步骤列表中切换到读这个步骤时, 右边将会显示此步骤所需要填入的一些参数, 这些参数主要包括存储器的访问的类型, 目前主要支援 RAM, SPI Flash, Nor Flash, Nand Flash App Area 及 Nand Flash Data Area. 右边的组合框中将会列出以上这几种存储器介质. 下面将说明下每个参数所代表的意义:

- ✓ 存储器访问类型: 在此组合框中的下拉列表, 可以指定用户当前想要对哪种存储器读数据.

- ✓ 地址: 让用户可以指定从存储器的那个位置开始读数据, 地址输入支援 10 进制或 16 进制. 注意, 不同的存储器型别会对应不同的地址单位, 默认为 Byte.
- ✓ 长度: 指定要从选定的存储器的指定地址读取数据的总长度. 注意, 此处所填充的数据是十六进制格式的.
- ✓ 文件: 在磁碟上指定个文件, 用于保存设备上指定的存储器中读取的数据信息
- ✓ 单笔数据长度: 它表明每次找设备指定存储器中读取数据长度, 因为当我们将上面长度设定很大值时, 比如 1MB, 而将此数据长度选为 4KB, 那表示 Tool 会每次找设备要 4KB 的数据, 而总长度是 1MB, 所以 Tool 总共会找设备要 256 次数据, 才能将所有数据读回来.

1.5.2 写步骤

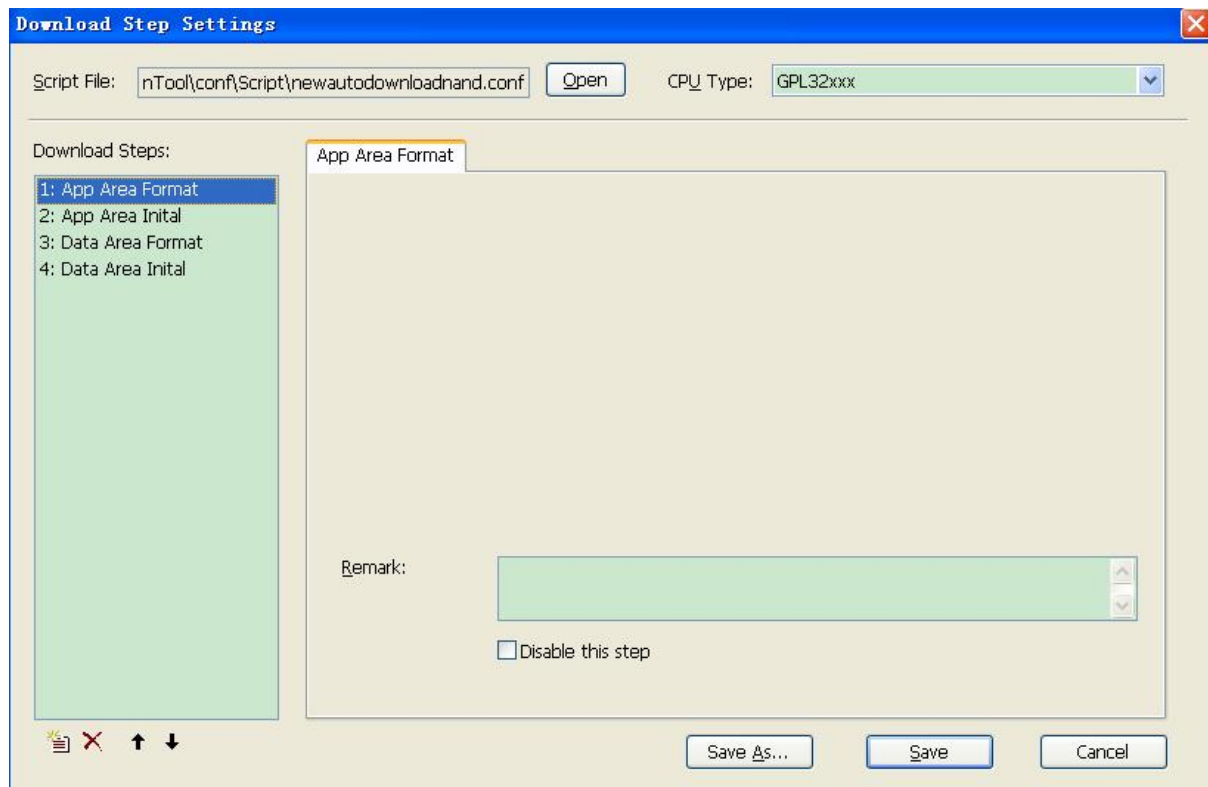


写步骤与读步骤在参数的配置上有很多相似的地方, 比如存储器的选择, 它同样会支援五种类型的存储器. 不同的是, 该步骤由用户指定的烧录文件可是二进制的 bin 文件也可以是由 Code Packer 包出来的 Header Data Block 的 hdb 文件. 如果用户选择烧录的是扩展名为 hdb 的文件, 此页面还会将该文件的所记录的 Tool ID 及 Tool 的版本号

显示出来. 这个主要是想提醒固件开发人员当前所使用的 hdb 档是否是他真正想要烧录的文件.

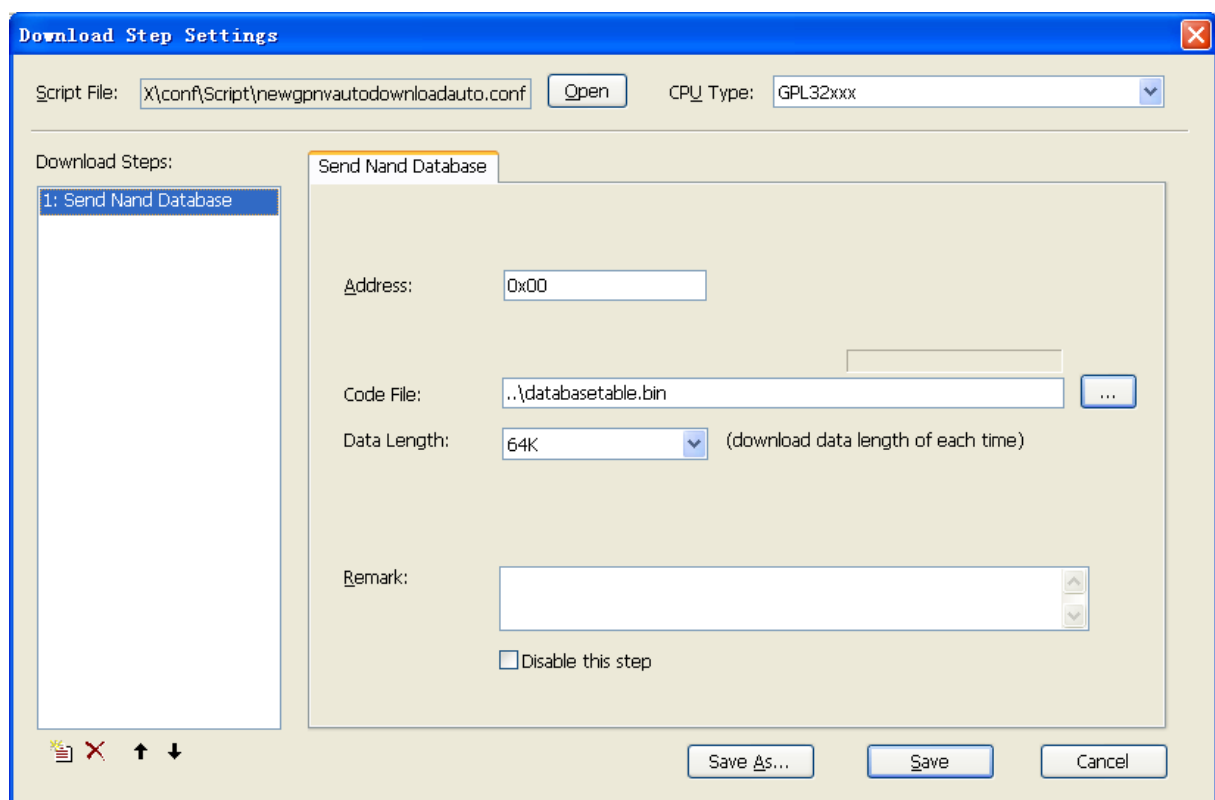
- ✓ 存储器访问类型: 在此组合框中的下拉列表, 可以指定用户当前想要对哪种存储器读数据.
- ✓ 地址: 让用户可以指定从存储器的那个位置开始写数据, 输入值支持 10 进制和 16 进制. 以 0x 开头为 16 进制, 否则为 10 进制. 注意, 不同的存储器型别会对应不同的地址单位, 默认为 Byte. (Byte, Sector, Block)
- ✓ 文件: 在磁碟上选取一个现有的文件, 稍后会将此文件中的内容写到设备上指定的存储器中
- ✓ 单笔数据长度: 它表明每次从 Tool 端发送指定数据长度到设备指定存储器中, 因为当我们将上面长度设定值时, 比如 1MB, 而将此数据长度选为 8KB, 那表示 Tool 每次发送 8KB 的数据到设备指定存储器中, 而总长度是 1MB, 所以 Tool 端总共会向设备发送 128 次数据, 才能完成本次烧录.

1.5.3 代码区与数据区步骤



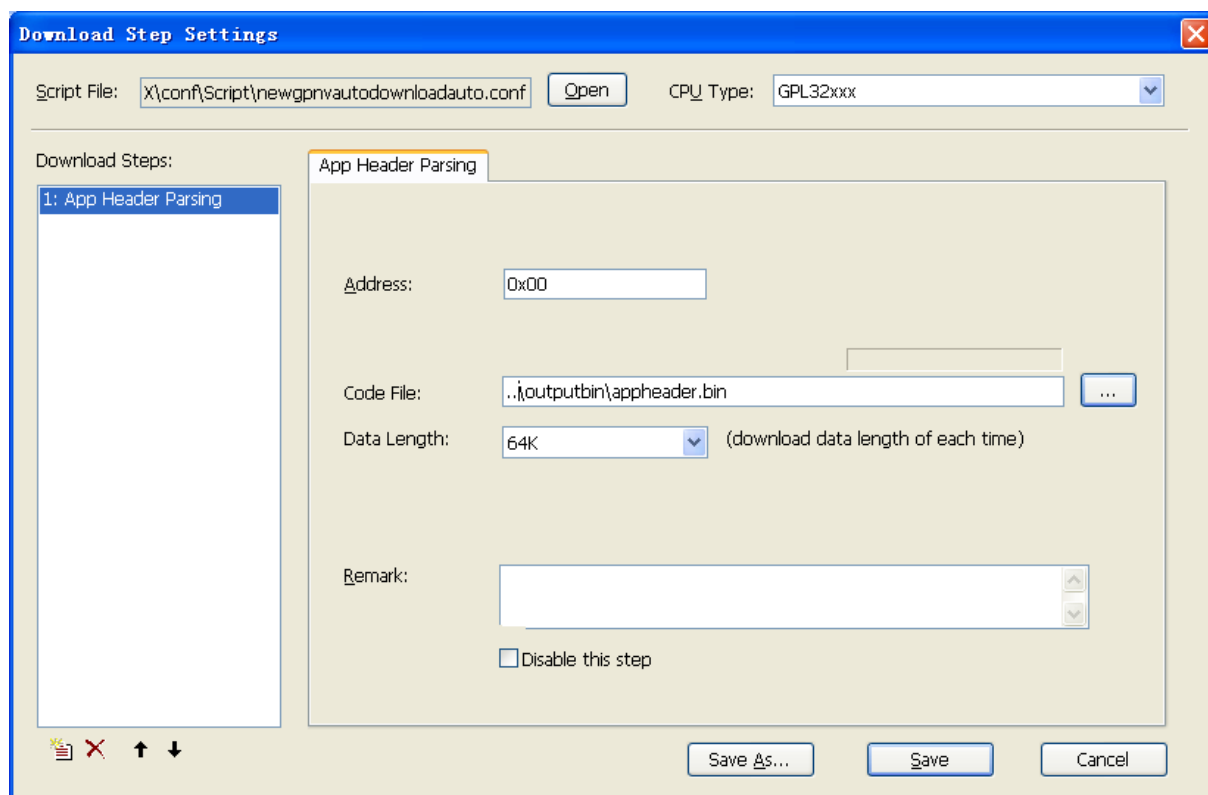
这几个步骤主要是用于对 Nand 这颗存储器中的代码区或者数据区, 进行一个初始化工作, 或者对这些区域进行格式化操作. 这些步骤属于 Step 命令, 这几个 Step 命令有着相同的用户界面, 但是不同的 Step 命令步骤具有不同的涵义. 这些步骤在执行过程中, 常常会对设备进行一个 polling 的动作, 会不停的询问当前的步骤执行的状况, 看是否有成功完成.

1.5.4 Send Nand Database



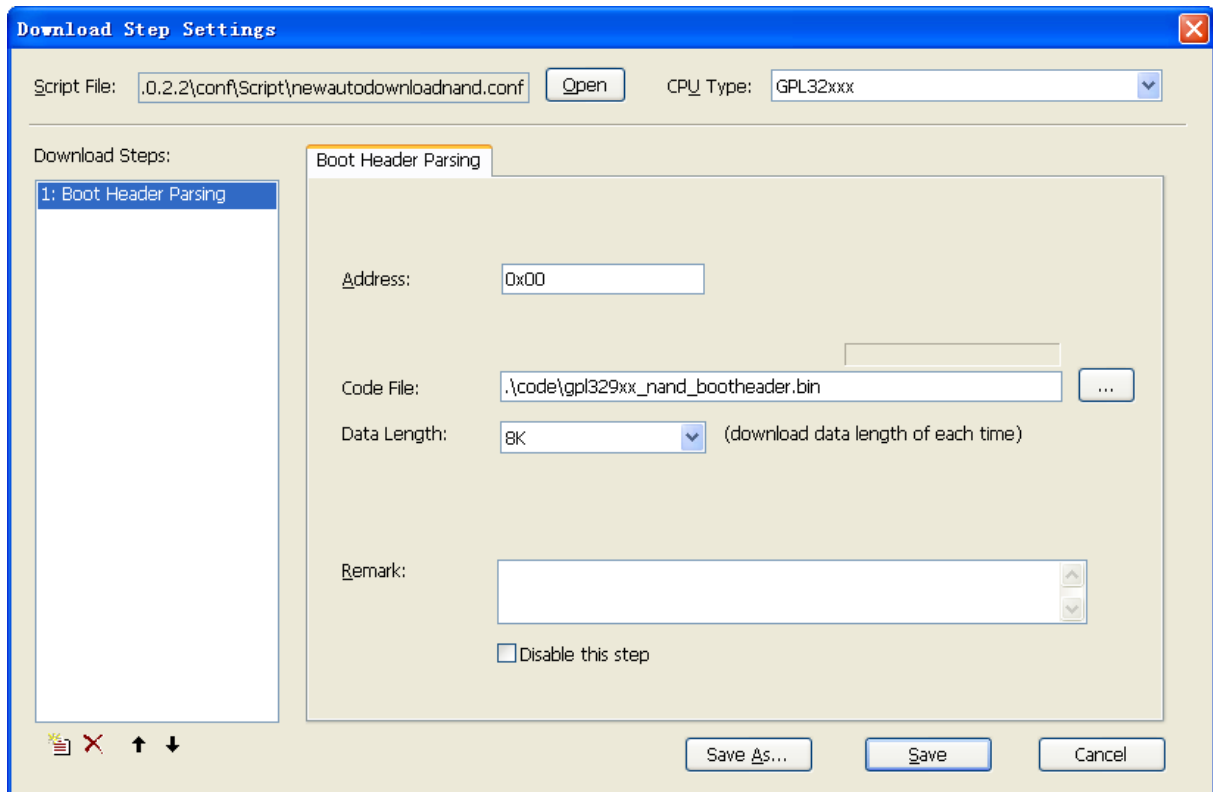
这个步骤主要是透过 0xFF55 这个命令向设备发送一个 DataBaseTable 的二进制 bin 文件, 设备自动判断当前使用的 Nand 型号, 并解析 bin 档, 查询对应 Nand 的型号及相关信息。

1.5.5 代码区头文件解析步骤



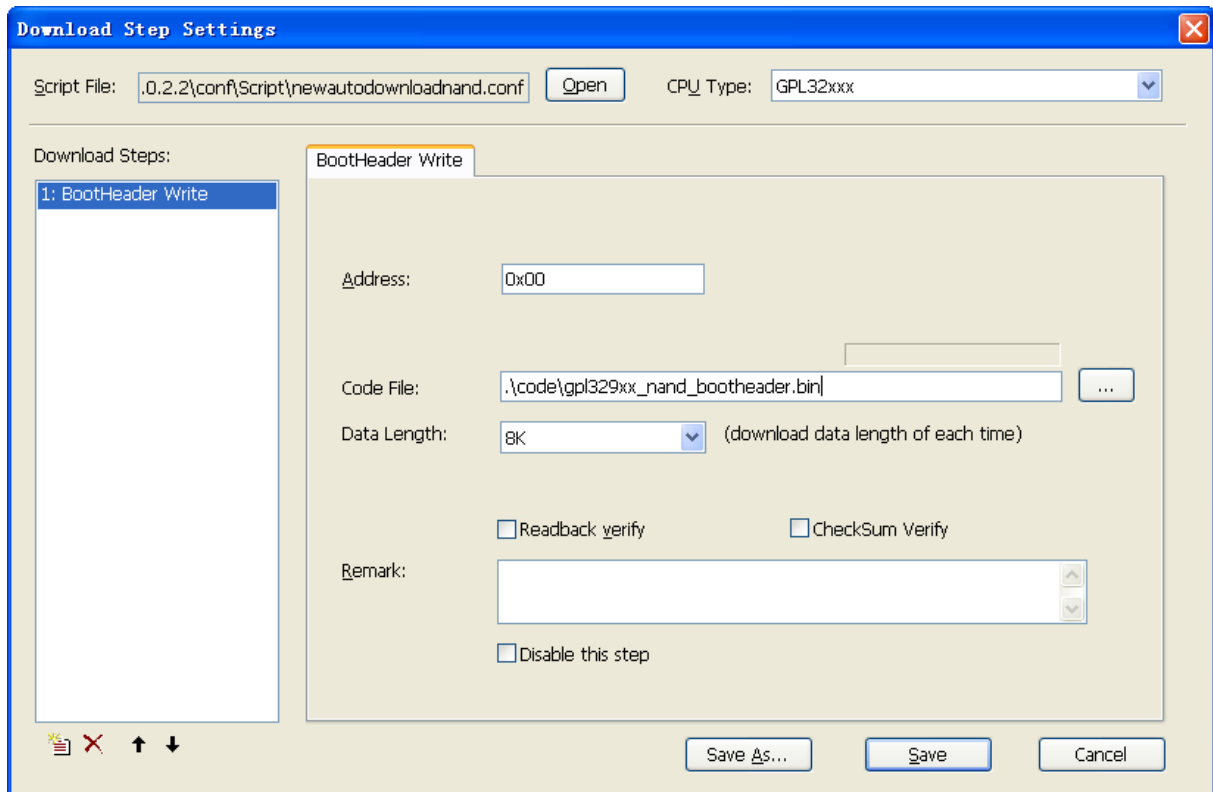
这个步骤主要是透过 0xFF56 这个命令向设备发送一个 AppHeader 的二进制 bin 文件, 设备收到命令后,自动解析 bin 文件,并根据解析出的结果自动调整写 App Area 时相关操作。

1.5.6 引导区头文件解析步骤



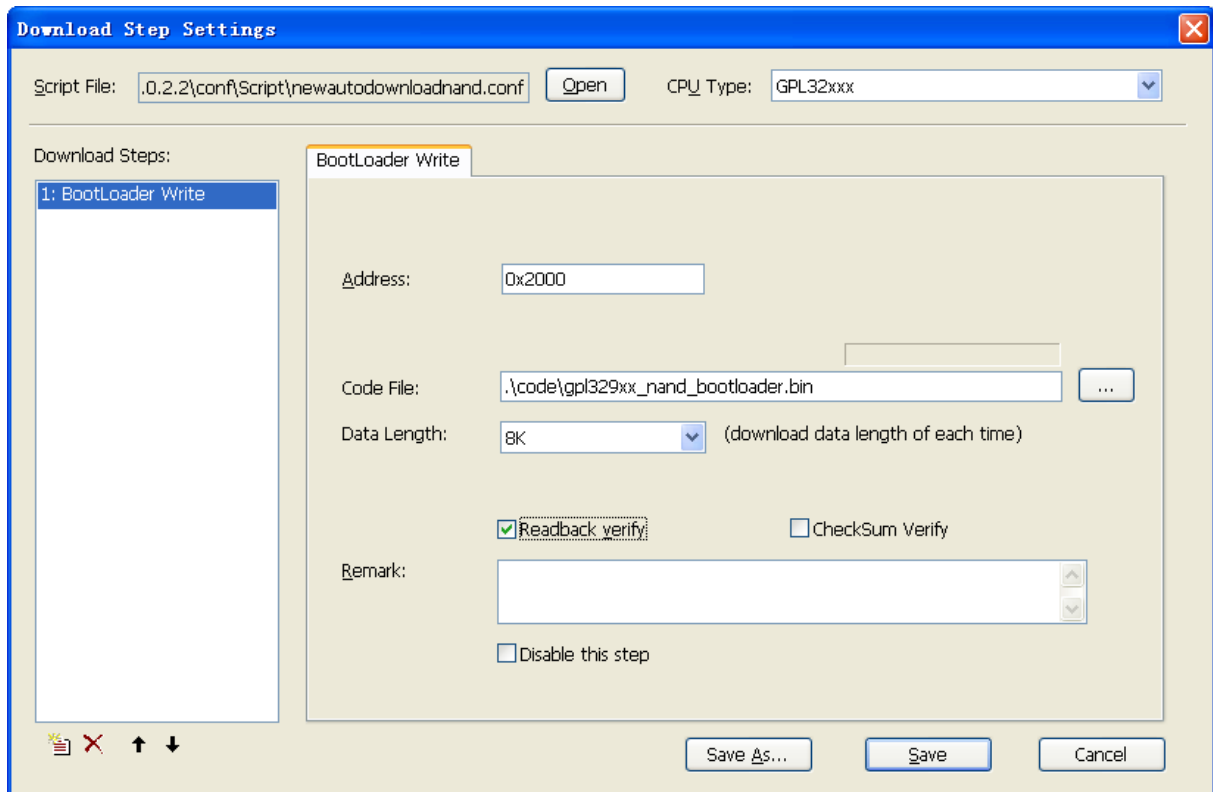
这个步骤主要是透过 0xFF51 这个命令向设备发送一个 BootHeader 的二进制 bin 文件，设备收到命令后，并对接收到 bin 文件进行解析，设备从而知道其引导区的 Block 总数及整个引导区的大小。

1.5.7 BootHeader 烧录步骤



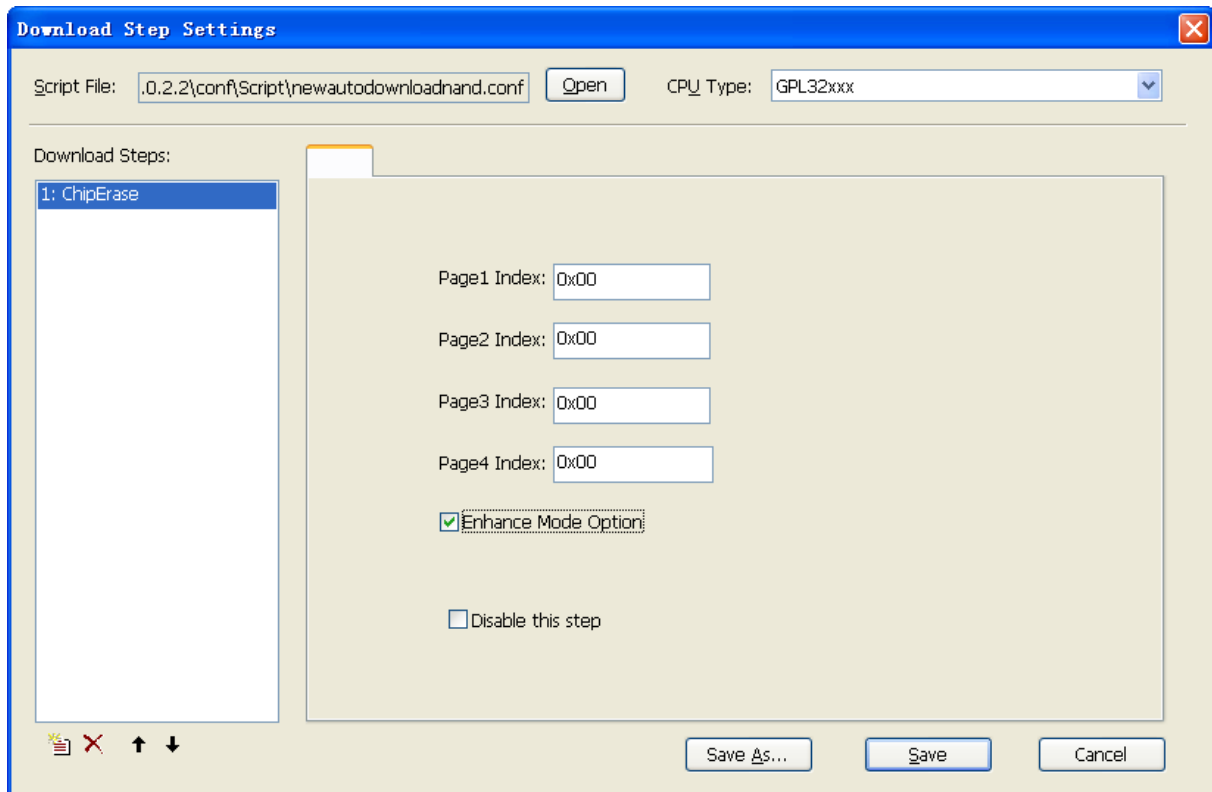
文件进行烧录，在上述的步骤中设备已经得知当前引导区的相关信息，此时这个步骤会将此 **BootHeader** 文件写到设备的引导区中。另外，该步骤还提供了两种写入数据正确性的验证方法：一个是 **Readback** 验证，另一个是 **CheckSum** 验证。两种方法都可以检验当前步骤中的数据文件烧到设备中的正确性与否。客观上给固件开发人员提供了一个错误检查的窗口。因为往往下载成功，并不代表说真正写入到设备中的数据也是正确的，有可能因为地址的原因导致设备 **Boot** 后达不到预期效果或者根本都 **Boot** 不起来，所以以防万一，使用者可以勾选当前步骤所要进行哪种检验机制。另外，此命令也属于 **Step** 命令。

1.5.8 BootLoader 烧录步骤



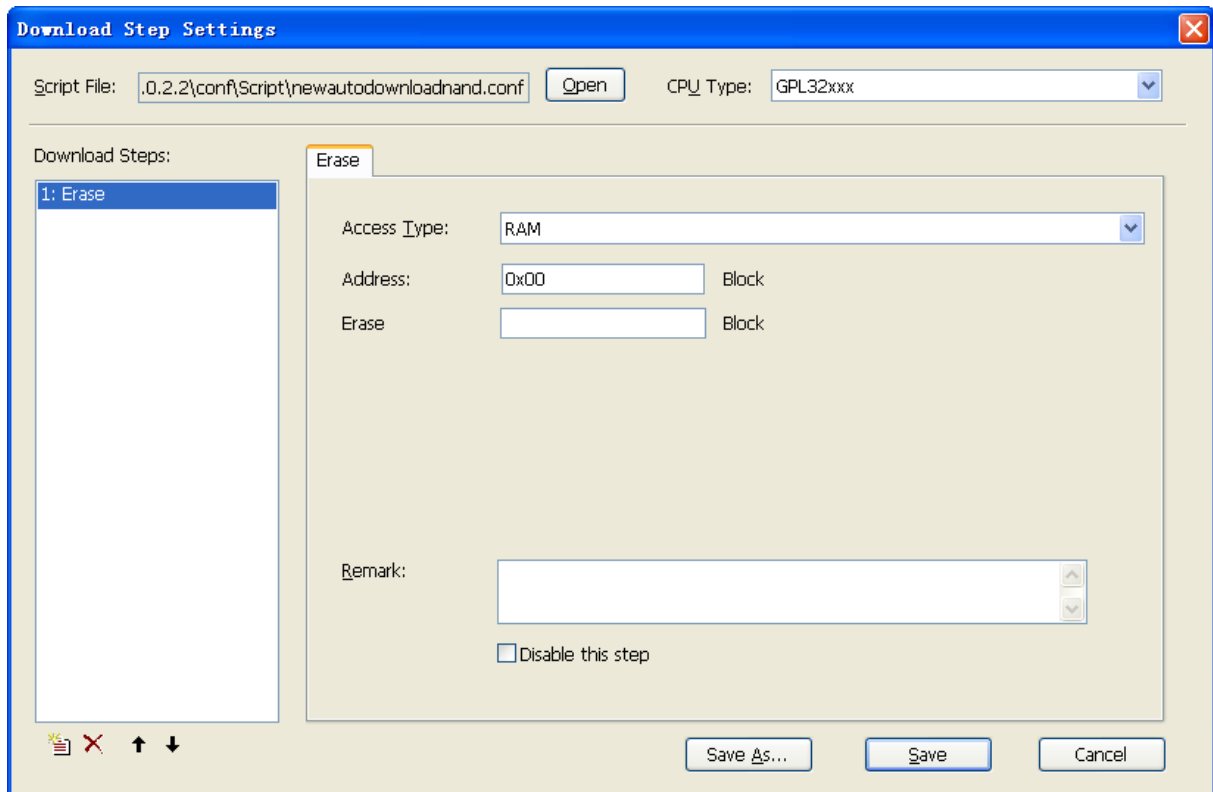
这个步骤与 BootHeader 烧录步骤很相似，这里就不再赘述，该步骤同样也属于 Step 命令。

1.5.9 Chip Erase 步骤



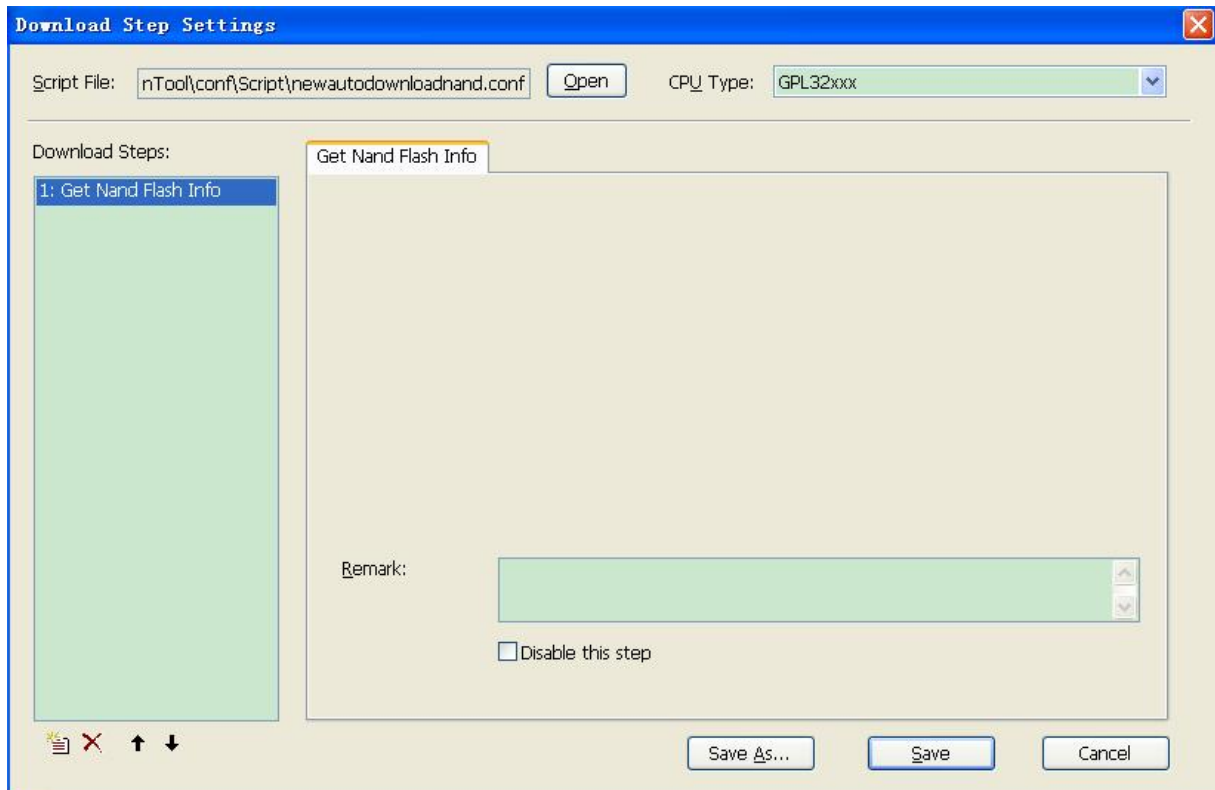
- ✓ 这个步骤主要是对 Nand Flash 的整个 Block 进行擦除动作，默认为普通擦除模式。
- ✓ 加强模式：勾选“Enhance Mode Option”，在 Page index 中输入想要侦测的 Page 数，则会对指定 page 使用加强模式擦除。

1.5.10 Erase 步骤



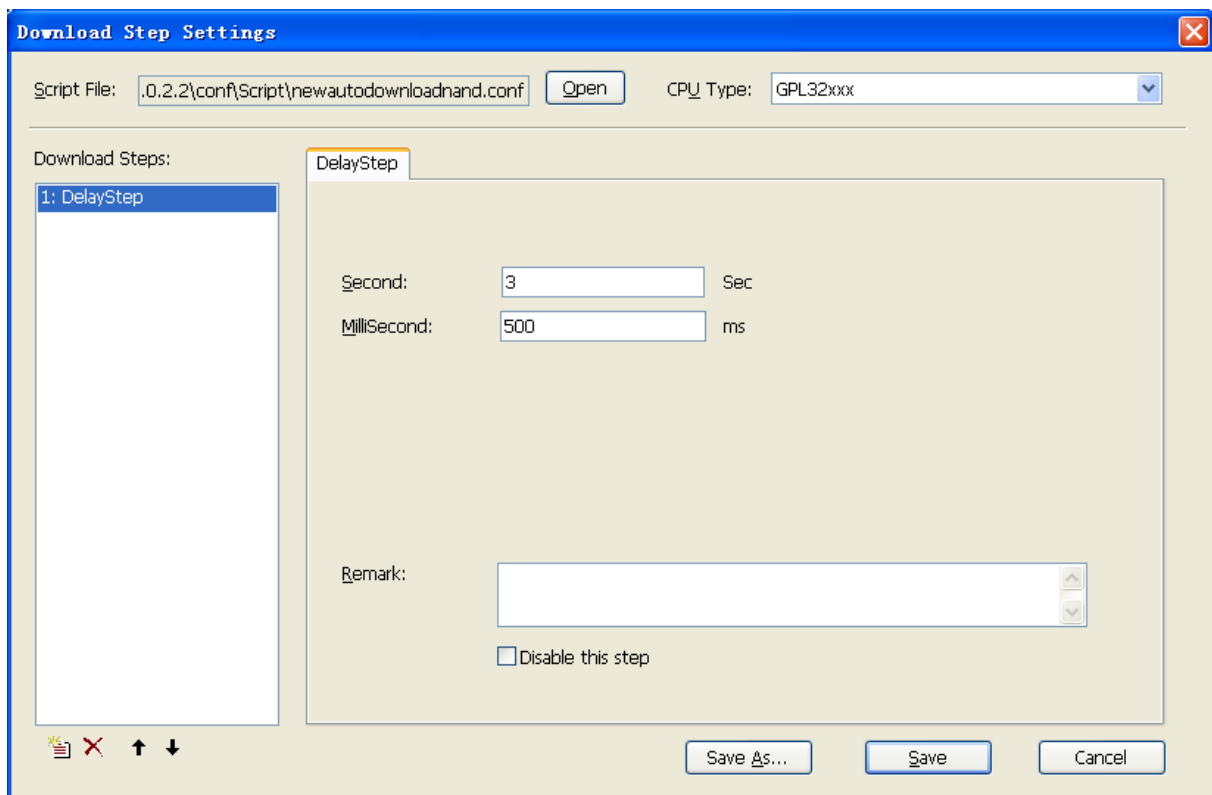
- ✓ 存储器访问类型: 在此组合框中的下拉列表, 可以指定用户当前想要进行擦除的存储器类型.
- ✓ 地址: 指定要擦除的起始 Block.
- ✓ Erase: 指定要擦除的 Block 的大小.

1.5.11 获取 Nand 信息步骤



当主机与设备进行通信时, Tool 就会透过执行此步骤, 来获取 nand 的有关资讯, 换句话说, 我们可以对 Nand 的相关属性进行监视, 并将这些信息用第三方软体进行输出, 如打印在 Uart 串口调试工具中.

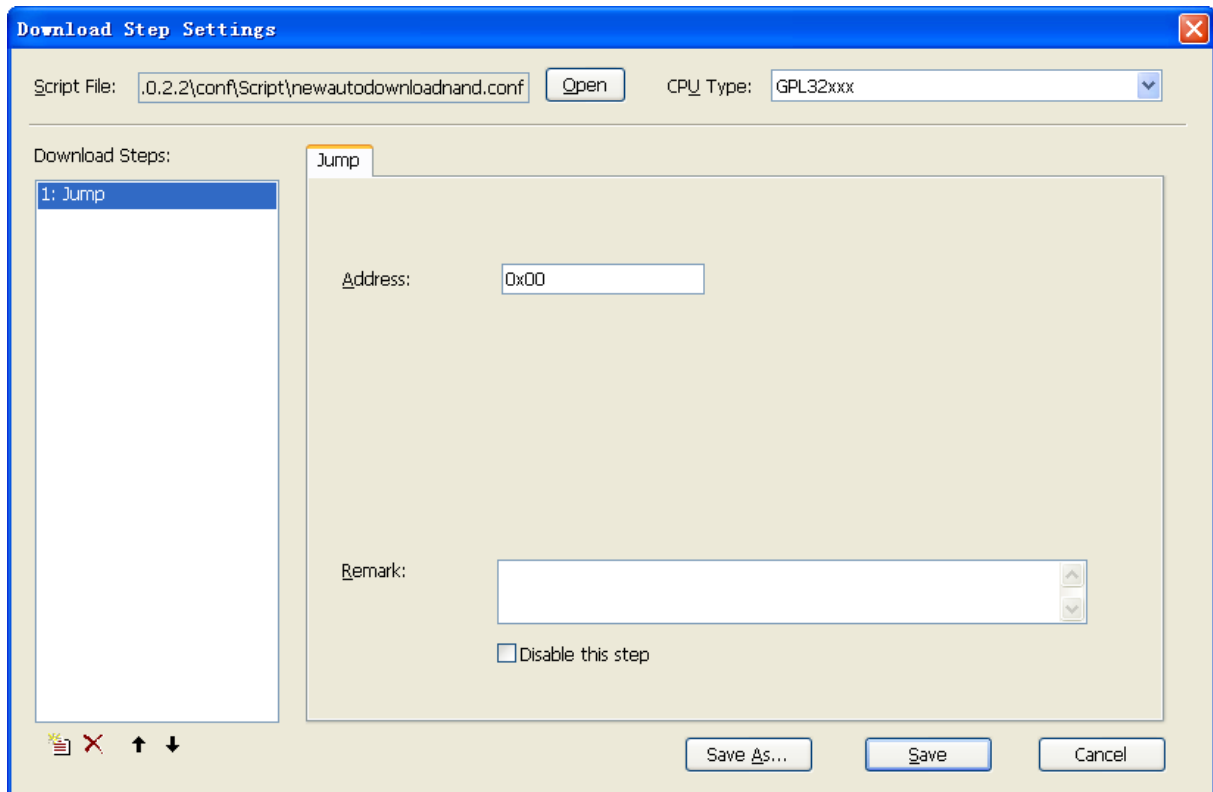
1.5.12 主机端延迟步骤



这个命令主要是用于 PC 端，并不会真正与 USB 设备进行沟通。

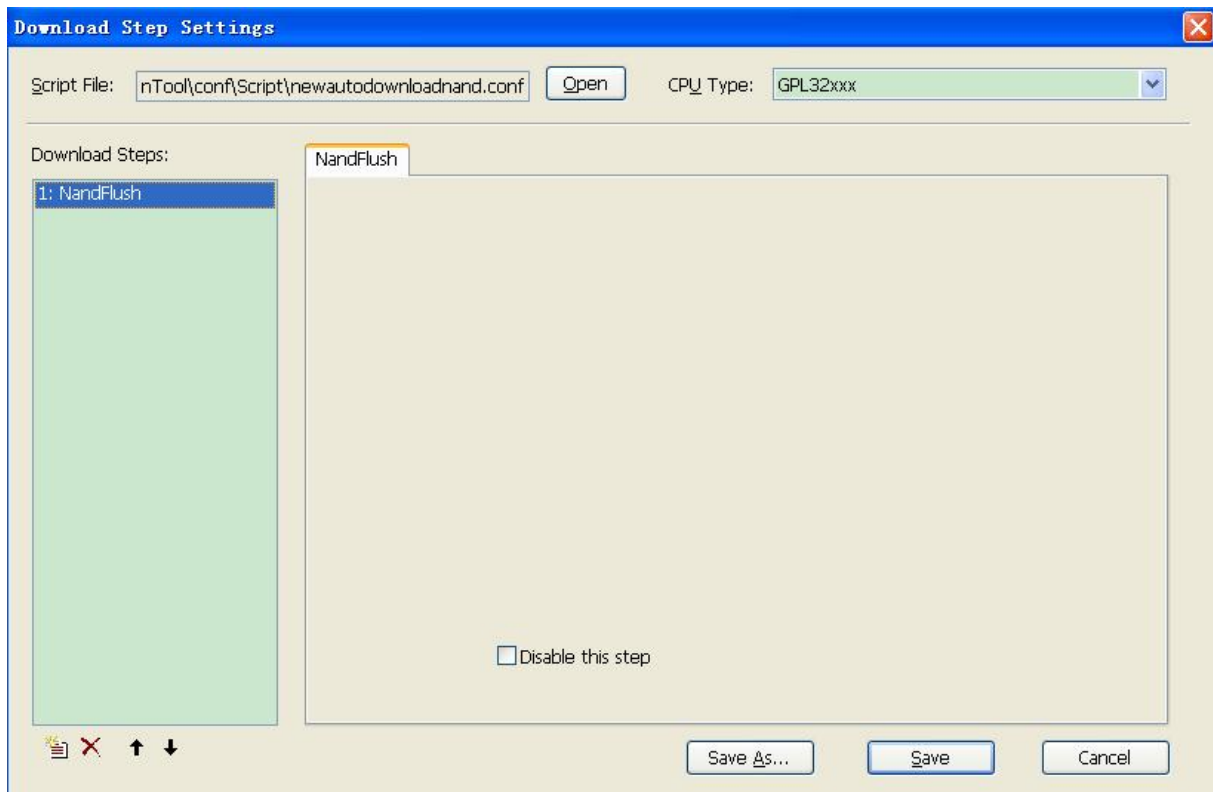
- ✓ 指定使用者想要延迟的时间。如上，延迟时间为 $3 \times 1000 + 500 = 3500$ ms。

1.5.13 跳转步骤



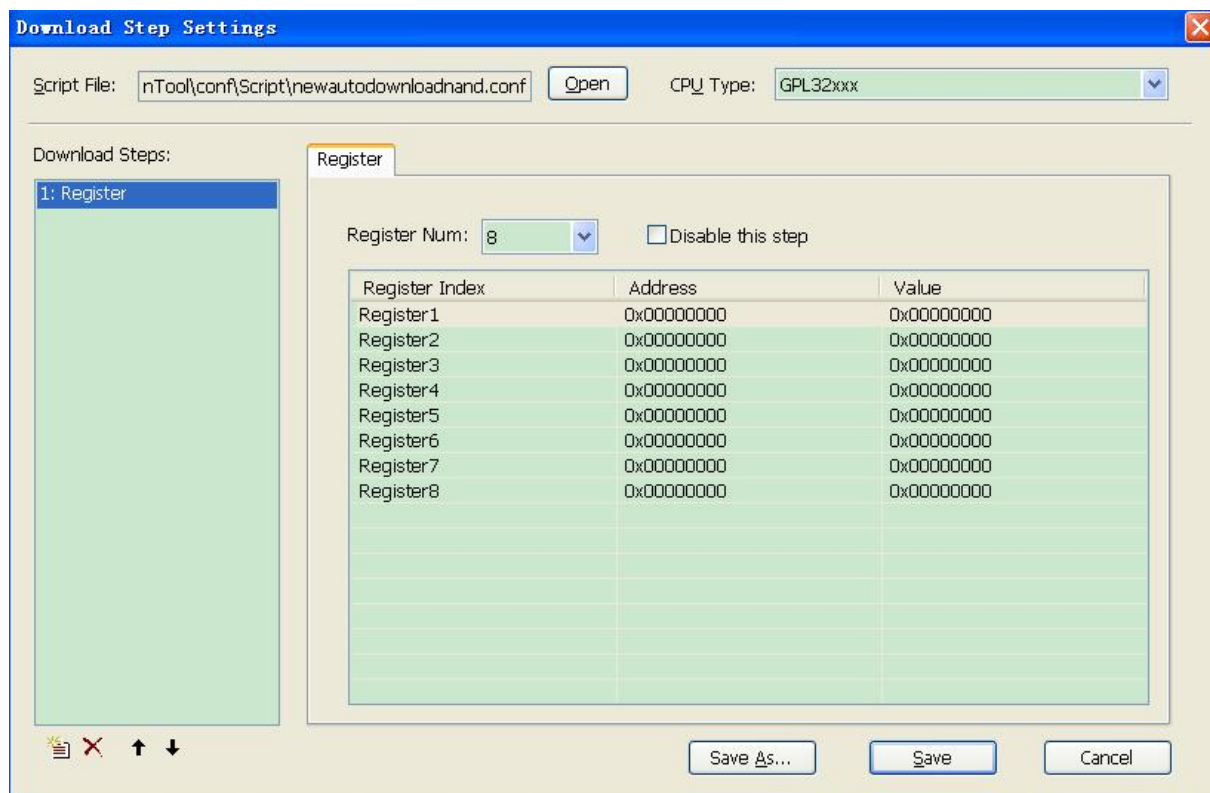
这个步骤主要是将设备的 PC 指针设定到指定的地址,使其跳转到这个内存地址,一般它是 iRam 或 SDRam 中的某个地址.

1.5.14 Nand Flush 步骤



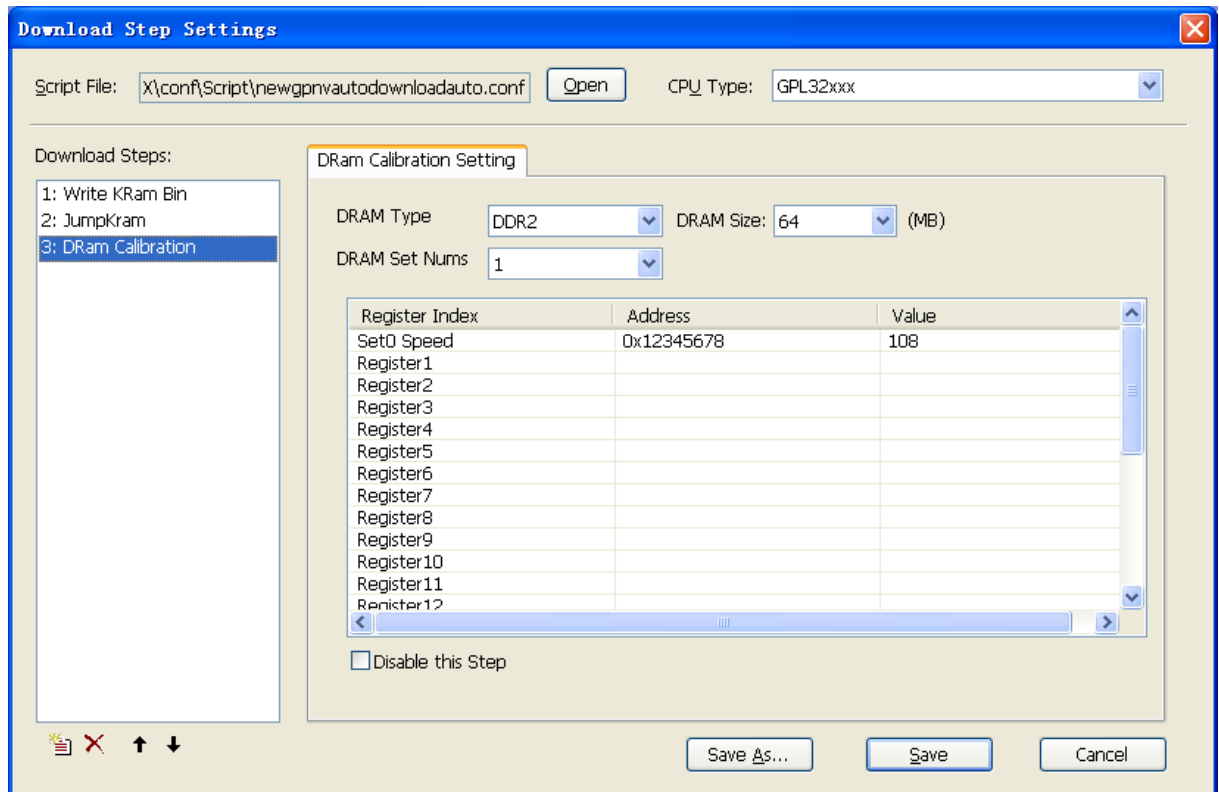
这个步骤主要是用于将缓冲区的数据刷新，使其对数据写入 Nand Flash 中。

1.5.15 寄存器写步骤



这个步骤主要是用于向指定的寄存器写入用户设定的值,Tool 会提供 31 组配置供用户设置,这也是目前支持的最大写入寄存器组数. 另外,我们提供了另一个步骤名称,其名字为读寄存器步骤. 当然,它与写寄存器步骤有很多相似之处,两个步骤均使用同一大套用户界面,惟一不同是,读寄存器步骤中的 **Value** 这一列是只读的,对使用者来说是不可编辑,因为它的值将是透过命令找设备要数据,并将读回的寄存器的数值在此显示出来.另外,当设备成功执行完此步骤,便可以在日志文件查看这些读回来的寄存器资讯.

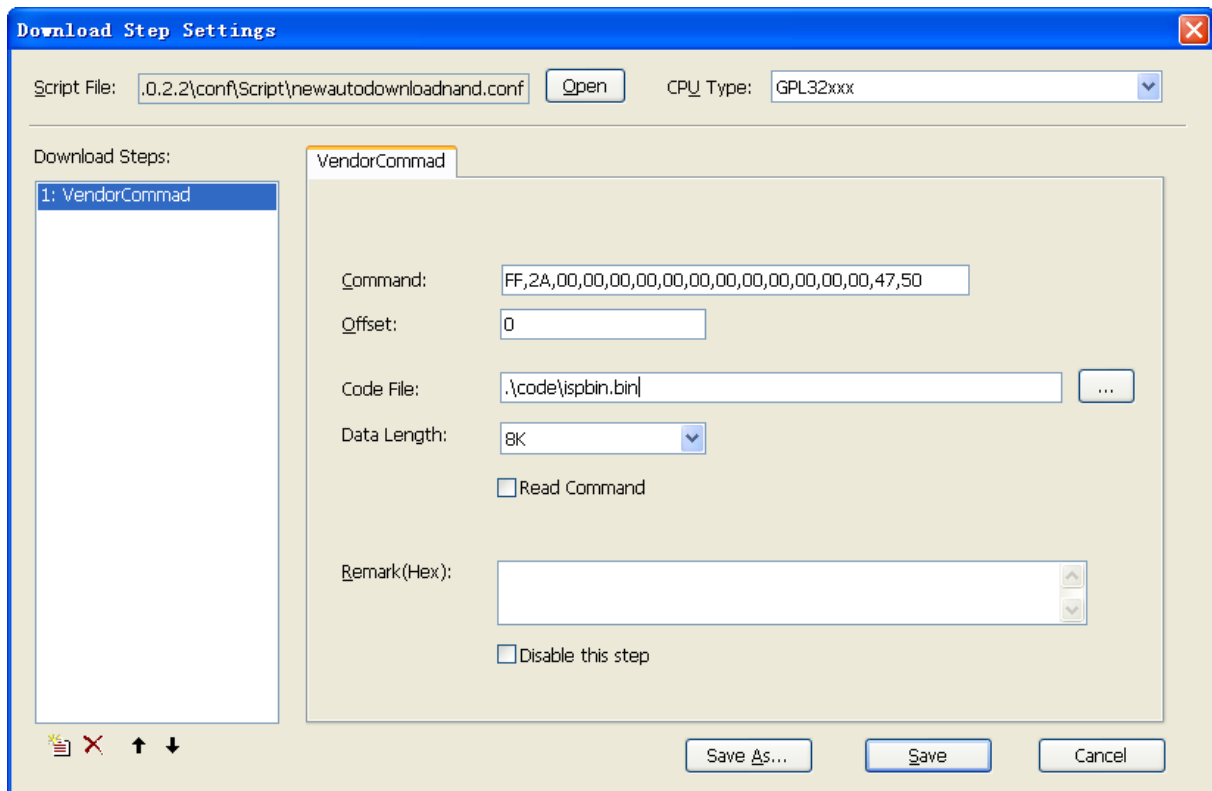
1.5.16 Dram Calibration 步骤



该步骤主要通过烧写 bin 档和 Jump 操作，并发送 DRAM Type、DRAM Size 和 SRAM Set Num 等信息给设备，设置在获取信息并解析后，让 IC 自动进行 Dram 校准操作。Tool 会在校准后读取相关寄存器的值并显示在界面中。

1.5.17 用户自定义命令步骤

用户自定义写命令：



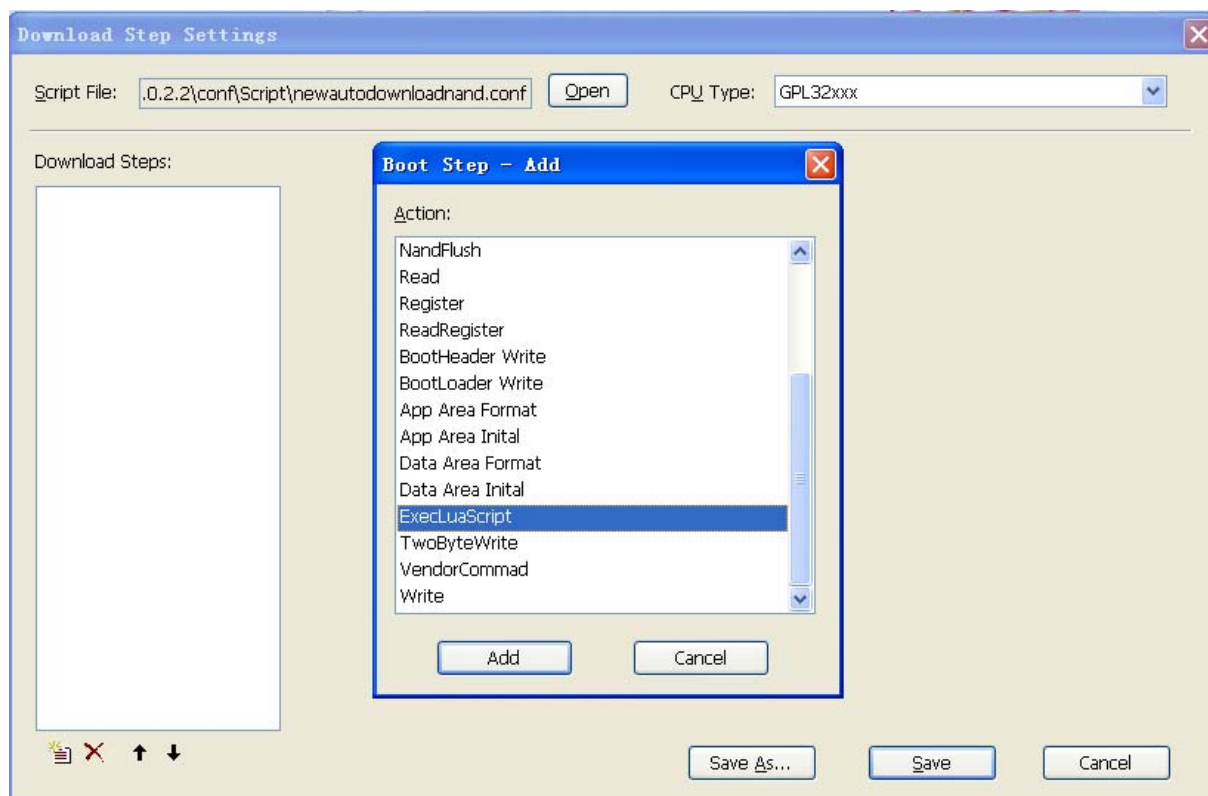
我们提供此步骤，主要是借此方式来适当增加弹性，也是希望最大限度的满足用户的需求。从而大的规范下对此步骤进行了一定程度的客制化。在这里,Tool 给用户提供了自定义写与自定义读两种用户界面相似却又功能不同的命令。一般而言, Tool 中其它步骤都可能透过此步骤来实现。所以一定程度上要求使用对其参数加以了解，尽量发挥此命令的功效;所以下面说下自定义写命令的相关参数及其用法:

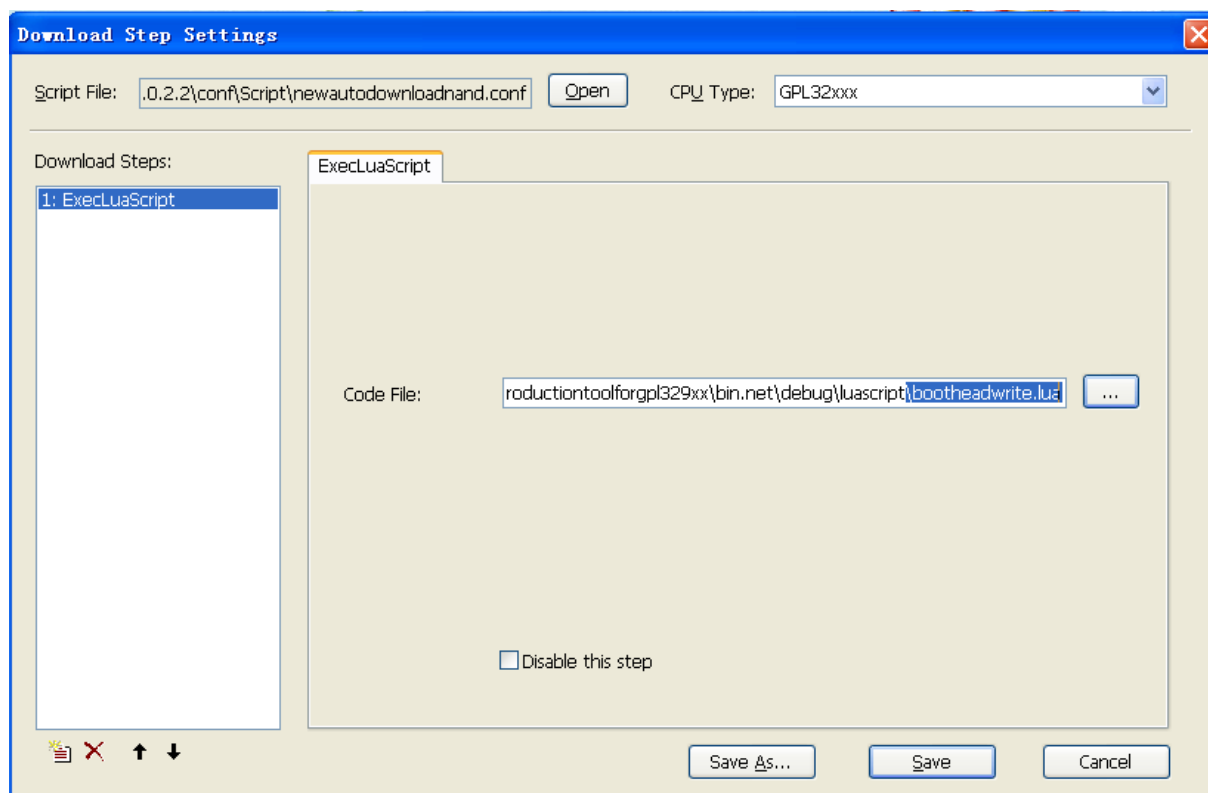
- ✓ 命令协议串: 用户可以在此栏位填入他自己想要送给设备的命令字符串, 这一串字符串将会直接作为 CBW 的 16 个字节的 数据送给设备, 在 Tool 是不会对这 16 个字节进行任何形式的解析, 当然我们假定此命令串的含义可由固件开发人员或其它使用者自行定制。
- ✓ 偏移地址: 设置偏移地址的字节数。
- ✓ 烧录文件: 让用户可以指定将要烧录(或读回)的代码文件, 若此栏位为空, 则表明用户并不想烧录文件, 而是作其它用途; 若不为空, 就说明使用者是要烧录 (或读回到) 此文件。
- ✓ 长度: 指定每一次读或写的数据长度。
- ✓ 读命令选项框: 主要是决定当前命令字符串是用作读还是写, 若勾选此选项, 表明当前配置

是想透过此步骤来从设备读取数据，否则，它就是写入命令，会根据用户设定的参数值,执行写入操作。默认是不勾选的，即写入步骤。

1.5.18 Lua Script

我们提供支援 Lua Script 的功能，在 MP 工具中整合了对 Lua 脚本的解析，用户可以将下载行为写为 Lua 脚本，再将此 Lua 脚本以 Step 形式挂进 MP 工具即可。





1.6 下载

当所有的配置都已设置好，并且设备列表的设备处于就绪态，此时就可以点击“Start Download”按钮，让 Tool 来执行烧录动作。用户也可以随时透过“Stop Download”按钮来取消本次烧录任务。在下载过程中，用户可以观察某个集线器的某个端口所挂接设备的下载进度，还有相应的状态信息。双击此设备所在的行，就会跳出一个对话框页面，此画面记录了该设备下载过程的所有日志信息，并会打印出下载的起始时间及终止时间。这里所提到的终止时间不一定是成功执行任务所花的时间，有可能是烧录途中因某种原因而引发的终止时间。

