IN1008 利用 IAP 实现 PLC 程序升级

2008/11/25

概述

GUTTA 平台目前对解释型 PLC 和编译型 PLC 都提供了支持。解释型 PLC 不需要得到 PLC 单片机的机器码。GUTTA Ladder Editor 根据用户的 PLC 程序，生成一种可被 PLC 解释系统识别的特定数据。PLC 解释系统根据这些特定数据，分别调用对应的指令操作。而在编译系统下，GUTTA Ladder Editor 需要根据用户的 PLC 程序，得到 PLC 单片机可以直接执行的机器码。这些机器码若需要被执行，就需要用 FLASH 烧写工具通过特殊的硬件烧写到单片的 FLASH。需要特殊硬件，会给 PLC 的最终使用者带来不便。所幸的是，目前绝大多数 FLASH 单片机都提供了在电路编程（或者叫做在系统编程）以及在应用编程。利用单片机的这两个功能，就能简化 FLASH 程序的修改。甚至 PLC 的使用者感觉不出编译型和解释型的区别。

对于目前单片机的 FLASH 的编程能力，可以归纳如下几种：

1. 支持在应用编程（IAP）。也就是说在 FLASH 上运行程序时同时能够对 FLASH 进行擦除、写等操作。当然运行程序的 FLASH 和被操作的 FLASH 不能在同一页上。
2. 支持在电路编程（ICP）。也就是说单片机 IC 不用取下来，直接在电路上，运用几个特殊的引脚就能够进行编程。根据这几个引脚使用上的不同，还可以分为：
   - 同步传输，必须有 CLK 管脚。例如用 SPI 接口。
   - 异步传输，不必有 CLK 管脚。例如用 USART 接口。
3. 支持用户 FLASH 的启动选择。这就意味着用户自己可以编写自己的 Boot Loader 代码。

下面举例说明：

- **NXP 的 LPC2134 单片机**。这个单片机支持：
  - 1：支持在应用编程（IAP）。
  - 2：支持在电路编程（ICP），异步传输，USART 接口。

由于支持 IAP，可以做解释型 PLC。做编译型 PLC 时，若是完全编译（需要更新整个固件），必须使用芯片制造商提供的 ICP 编程工具 LPC2000 Flash Utility。由于这个软件不能嵌入到 GUTTA Ladder Editor 中，使用上不是特别方便。也可以遵循 LPC2000 Flash Utility 的编程协议自己写下载器，这种下载器有开源项目，可供参考。

- **新华龙的 C8051F04x 单片机**。这个单片机支持：
  - 1：支持在应用编程（IAP）。
  - 2：支持在电路编程（ICP），同步传输，JTAG 接口。

由于支持 IAP，可以做解释型 PLC。做编译型 PLC 时，若是完全编译（需要更新整个固件），必须使用 JTAG 接口编程。虽然是在电路编程（IC 不用取下），但 JTAG 不是通用计算机通讯端口，用户必须配备专门的 JTAG 器件才能下载程序，不是特别理想。

- **ATMEL 的 ATMEGA64 单片机**。这个单片机支持：
  - 1：支持在应用编程（IAP）。
  - 2：支持在电路编程（ICP），同步传输，JTAG 接口或者 SPI 接口。
  - 3：支持用户 FLASH 的启动选择。

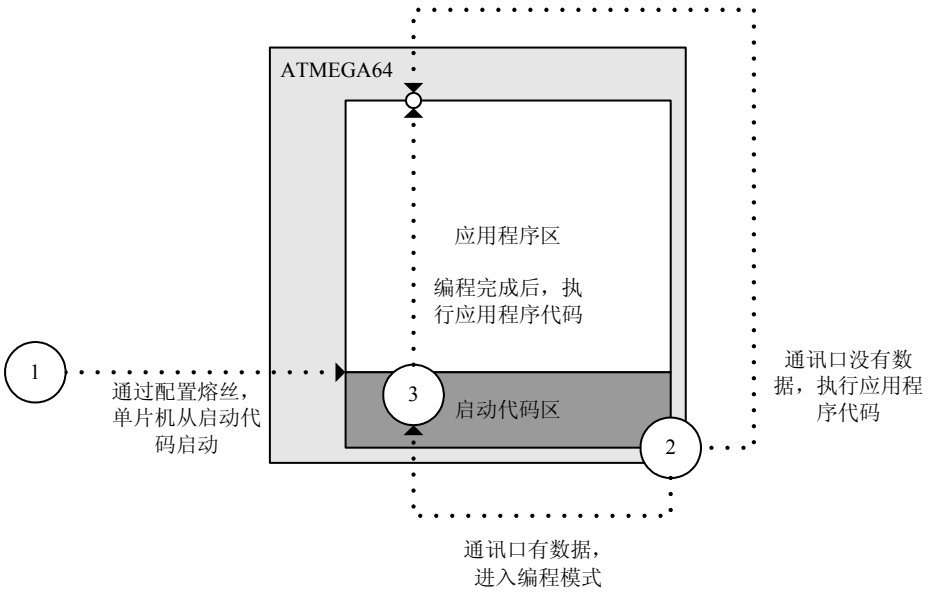
由于支持 IAP，可以做解释型 PLC。做编译型 PLC 时，若是完全编译（需要更新整个固件），必须使用 JTAG 接口编程或者 SPI 编程接口。这两种接口都不是通用计算机通讯端口。JTAG 需要特殊硬件，SPI 虽然可以用计算机并口模拟，但目前大部分计算机没有并口。ATMEGA64 支持用户 FLASH 的启动选择，这就意味着用户自己可以编写自己的 Boot Loader

代码。这正是本文详细介绍的内容。

实现

原理

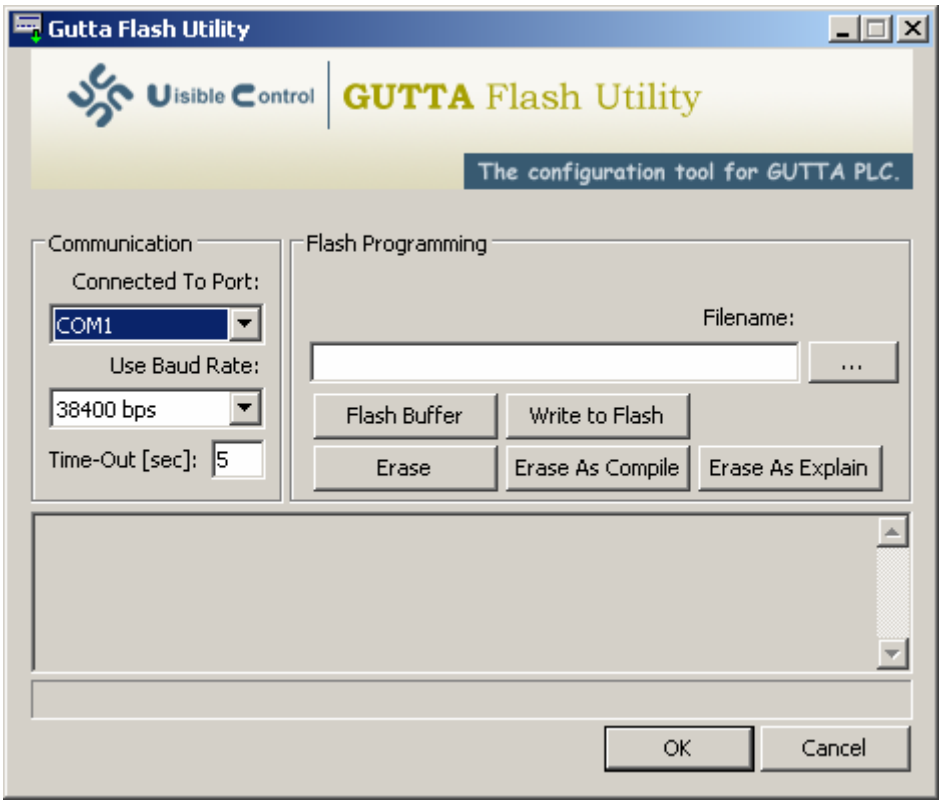
ATMEGA64 单片机 Boot Loader 工作原理：



通过配置 ATMEGA64 的熔丝位，可以将单片机的复位地址修改为 62K 的位置。这样单片机的 FLASH 就分为了两个区。一个是 62K 以下的应用程序区，大小为 62K。一个是 62K 到 64K 的启动代码区，大小为 2K。当片机复位后，先运行启动代码区的代码，启动代码先判断通讯口是否有数据。延时一段时间如果没有发现有效连接，转跳到应用程序区。由于这个延时时间很短，您很难察觉到启动代码区代码的执行。若通讯口有连接，说明有编程请求，单片机进入编程模式。这个时候启动代码通过和计算机通讯，一页一页的对应用程序区的 FLASH 进行编程。当所有 FLASH 数据页编程结束后，启动代码程序转跳到应用程序区。

使用

要通过 ATMEGA64 的 Boot Loader 对单片机进行编程，首先要运行编程软件：GUTTA Flash Utility：



- 第一步：

点击 Filename 下面的按钮（…），选择需要下载的文件。如果是在 GUTTA Ladder Editor 中下载弹出的对话框，这一步可以省略。GUTTA Ladder Editor 调用编译器生成 hex 文件后，会自动载入数据，并在信息栏中显示如下文字：

```
Generating source file:  Done.
Compiling source file:  Done.
Parsing hex file:  Done. (File size: 23738)
```

  - 第 1 行表示生成源代码文件完成。
  - 第 2 行表示编译源代码文件完成。
  - 第 3 行表示解析 hex 文件完成。

- 第二步：

在软件的（Connected To Port）中选择计算机串口。并将这个计算机串口连接到开发板的串口上。

- 第三步：

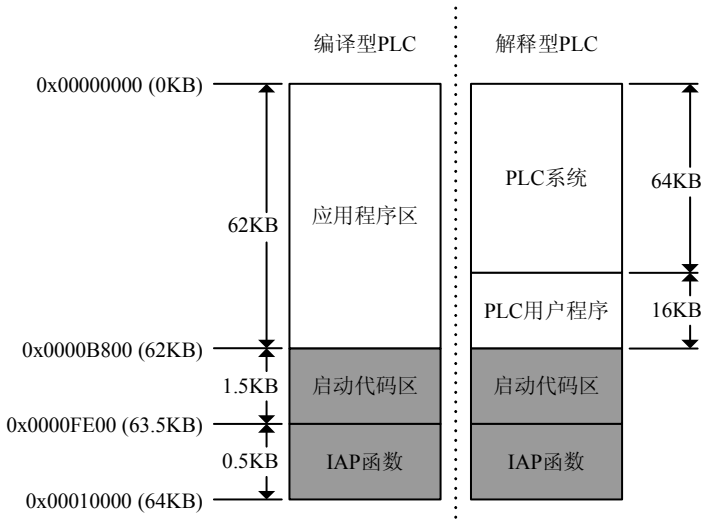
点击按钮（Write to Flash），GUTTA Flash Utlity 会不断的发送握手信号到 ATMEGA64（ATTACH 通讯指令）。这个时候按下开发板的复位按钮。ATMEGA64 复位后，开始运行启动代码程序，与 GUTTA Flash Utlity 连接后便开始了对 ATMEGA64 的编程。信息栏中显示如下文字：

```
Open the hex file:  Done. (File size: 47129)
Excuting command <ATTACH>:  Done.
Excuting command <GETID>:  Done. (ID: "ATMEGA64")
Excuting command <PAGESZ>:  Done. (Page Size: 256)
Excuting command <PAGECNT>:  Done. (Page Count: 248)
Excuting command <CLRP>:  Done.
```

```
Excuting command <DNLDP>:  Done.
Excuting command <LOGOUT>:  Done.
```

- 第 1 行表示打开 hex 文件正确。
- 第 2 行表示连接 ATMEGA64 完成（开发板在指定时间内被复位）。
- 第 3 行表示获得 ID 号完成，被编程的芯片为“ATMEGA64”。
- 第 4 行表示获得 FLASH 程序页大小完成。ATMEGA64 的程序页为大小 256 个字节。
- 第 5 行表示获得 FLASH 程序页数量完成。62K 的用户程序一共有 248 个程序页。
- 第 6 行表示擦除用户程序完成。
- 第 7 行表示下载用户程序完成。
- 第 8 行表示断开 ATMEGA64 完成。，启动代码程序转跳到应用程序区。

AVR ATMEGA64 FLASH 分区



在编译型 PLC 模式下，启动代码通过调用 IAP 函数，更新应用程序区的内容。

在解释型 PLC 模式下，PLC 系统区是固定的。若需要下载 PLC 程序（通过 GUTTA 通讯协议），PLC 系统调用高端地址的 IAP 函数对 PLC 用户程序区进行操作。

GUTTA Flash Utility 通讯协议

<01H> ATTACH 连接

ATTACH 指令用于连接到目标芯片。GUTTA Flash Utility 在按下（Wirte to Flash）按钮后，会不停的向串口发送 ATTACH 指令。和其他指令不同，GUTTA Flash Utility 不会对芯片响应做等待，而是按照自己固定的时间间隔不停的发送（这个间隔很短），直到接收到正确的响应或者是 ATTACH 尝试时间到（这个时间较长，默认是 5 秒）。您需要在这段时间内复位芯片，使芯片运行 Boot Loader 程序，Boot Loader 程序检测到串口的 ATTACH 指令，就能做出正确的返回。一旦 ATTACH 指令被正确返回，芯片正式进入 Boot Loader 下载模式。发送数据：

功能码	LRC 校验
01H	FFH

返回数据：

功能码	LRC 校验
01H	FFH

<02H> LOGOUT 断开

LOGOUT 指令用于断开目标芯片的连接。Boot Loader 程序一旦检测到串口的 LOGOUT 指令，表示所有的程序下载已经完成。Boot Loader 程序停止串口的通讯同时转跳到应用程序区。

发送数据：

功能码	LRC 校验
02H	FEH

返回数据：

功能码	LRC 校验
02H	FEH

<03H> CLRP 擦除程序

CLRP 指令用于擦除目标芯片的应用程序区的 FLASH。FLASH 一旦被擦除，所有的 FLASH 数据都变成 FFH。

发送数据：

功能码	LRC 校验
03H	FDH

返回数据：

功能码	LRC 校验
03H	FDH

<04H> DNLDP 下载程序

DNLDP 指令用于下载程序。由于 FLASH 中的数据是按页组织的。一条 DNLDP 指令一次下载一个页的数据。因此 DNLDP 指令的实际长度由目标芯片的页大小决定。GUTTA Flash Utility 在下载前需要先通过 PAGESZ 指令查询目标芯片的页大小，从而决定 DNLDP 指令的长度。

发送数据：

功能码	页地址		页数据			LRC 校验
04H						

返回数据：

功能码	LRC 校验

04H	FCH

发送数据的页地址为字（16bit）长度。发送时高字节在前，低字节在后。

<05H> GETID 查询芯片 ID

GETID 指令用于查询目标芯片的识别 ID。ID 以字符串的形式返回。

发送数据：

功能码	LRC 校验
05H	FBH

返回数据：

功能码	芯片 ID 字符串			LRC 校验
05H				

<06H> PAGESZ 查询 FLASH 页大小

PAGESZ 指令用于查询目标芯片的 FLASH 页大小。GUTTA Flash Utility 需要通过 PAGESZ 指令得到页大小从而确定 DNLDP 指令的长度，即一次下载多少应用程序。

发送数据：

功能码	LRC 校验
06H	FAH

返回数据：

功能码	页大小		LRC 校验
06H			

返回数据的页大小为字（16bit）长度。发送时高字节在前，低字节在后。页大小的单位为字节。

<07H> PAGECNT 获得 FLASH 页数量

PAGECNT 指令用于查询目标芯片的 FLASH 页数量。GUTTA Flash Utility 需要通过 PAGESZ 指令得到页数量从而确定最多能够下载多少应用程序数据。

发送数据：

功能码	LRC 校验
07H	F9H

返回数据：

功能码	页数量		LRC 校验
06H			

返回数据的页数量为字（16bit）长度。发送时高字节在前，低字节在后。

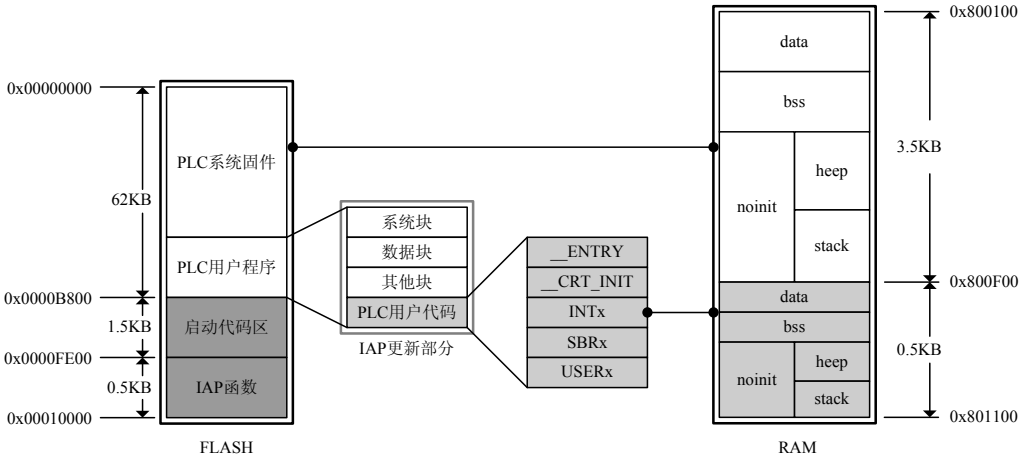
部分编译

对于编译型 PLC，存在两种形式：全编译和部分编译。全编译指 PLC 系统固件和 PLC 用户程序做为一个整体下载到目标硬件上去。部分编译指 PLC 系统固件保持不变，下载程序时只编译 PLC 用户程序部分。然后 GUTTA 编程软件通过和 PLC 系统固件的通讯，利用 IAP 将用 PLC 户程序代码写入指定的 FLASH 地址中。

比较：

1. 全编译固件升级方便，因为每次下载都需要重新编译（或链接）固件，在需要更新 PLC 固件时，只需要更新 GUTTA 软件中的固件库（未链接）就可以了。部分编译由于平时 PLC 固件保持不变，升级固件需要使用特殊的硬件和软件。
2. 全编译下载数据量大，较慢。部分编译下载数据量小，较快。
3. 全编译程序加密困难，由于系统固件每次都需要重新导入，其他厂商只需要复制硬件就能仿制产品。部分编译由于只下载 PLC 程序逻辑部分代码，系统固件依然可以使用单片机的加密措施从而不会被轻易读取。
4. 全编译由于系统固件和用户程序一起链接，用户在用 C 语言编写自定义功能块时，可以方便的调用系统固件的功能函数、方便的引用系统固件使用的全局变量。而部分编译由于系统固件所有的函数和变量都已定位，只能通过直接地址来调用，不是很方便。部分编译固件程序的 FLASH 空间和用户程序的 FLASH 空间完全隔离。同样的，部分编译固件程序的 RAM 空间和用户程序的 RAM 空间也是完全隔离。导致两个部分互相操作不是很方便。

下面给出 CPU-EC20 (AVR,Compile)部分编译的空间结构以及运行流程：



由于部分编译只更新部分代码，PLC 系统固件保持不变，那么我们可以将 PLC 程序的上传下载通讯程序放入 PLC 系统固件中。这样只要单片机支持 IAP 功能，我们就能实现部分编译型 PLC。PLC 用户程序包括 4 部分：系统块、数据块、其他块、PLC 用户代码块。系统块、数据块、其他块都是常数数据；PLC 用户代码块是可执行数据。PLC 系统固件使用前 3.5KB 的 RAM。PLC 用户代码块使用后 0.5KB 的 RAM。这些分区信息必须以参数的形式传递给链接器，使其能够正确的定位代码和变量。PLC 用户代码分为 5 部分：

- __ENTRY：PLC 用户代码的总入口，地址为 PLC 用户程序开始地址加上一个常数偏移（GCC 在链接的时候会插入向量表，故有一个常数偏移）。PLC 用户代码的其他函数都必须通过这个函数来调用。PLC 系统固件只需要知道这个地址即可，免去了其他定位。

- __CRT_INIT：PLC 用户代码的 C 语言运行环境初始化函数。不论是 PLC 的冷启动还是热启动，都必须调用这个函数以初始化用户自定义 C 语言功能块中的所有常数数据。
- INTx：PLC 的中断程序，由 GUTTA 编译系统根据 PLC 程序自动生成。
- SBRx：PLC 的子程序，由 GUTTA 编译系统根据 PLC 程序自动生成。
- USERx：用户自定义 C 语言功能块，由用户自己书写，由 GUTTA 编译系统编译。

附件

Boot Loader 源代码

http://www.visiblecontrol.com/technologies/inindex/in1008/BootLoader.zip

GUTTA Flash Utility 源代码

http://www.visiblecontrol.com/technologies/inindex/in1008/GuttaFlashUtility.zip