ZS110A-loader 示例

发布 ***1.0.0***

2018 年 **11** 月 **02** 日

目录

CHAPTER 1

文档介绍

1.1 文档目的

用 loader 示例工程来介绍 loader 的相关知识。

1.2 术语说明

表 1.1: 术语说明

术语	说明
ZEPHYR	为所有资源受限设备，构建了针对低功耗、小型内存微处理器设备而进行优化的物联网嵌入式小型、可扩展的实时操作系统（RTOS）
Loader	second bootloader，根据不同启动参数加载对应类型的应用程序

1.3 参考文档

- 无

1.4 版本历史

表 1.2: 版本历史

日期	版本	注释	作者
2018-08-22	1.0	初始版本	ZS110A 项目组

CHAPTER 2

Loader 示例

示例 (\samples\loader_binaries) 的目录结构如下：

```
├─keil5: keil 工程目录
│  ├─hello_0
│  ├─hello_1
│  └─loader
└─src: 源码目录
                ├─hello_0
                ├─hello_1
                └─loader
```

从目录结构可以看出本示例共有 3 个 keil 工程，分别如下：

1. loader:

   上电后默认或启动参数 =REBOOT_TYPE_GOTO_APP 时加载 hello_0

   启动参数 =REBOOT_TYPE_GOTO_DTM 时加载 hello_1

2. hello_0：

   每 1s 打印一次 “Hello World 0!”，30s 后执行” sys_pm_reboot(REBOOT_TYPE_GOTO_DTM)” 设置好启动参数后重启。

```
static int time = 0;
while (1) {
        printk("Hello World 0!\n");
        k_sleep(1000);
        time++;
        if (time == 30)
```

(continues on next page)

(续上页)

```
                sys_pm_reboot(REBOOT_TYPE_GOTO_DTM);
}
```

3. hell0_1：

   每 2s 打印一次“Hello World 1!”，30s 后执行”sys_pm_reboot(REBOOT_TYPE_GOTO_APP)”设置好启动参数后重启。

```
static int time = 0;
while (1) {
        printk("Hello World 1!\n");
        k_sleep(2000);
        time += 2;
        if (time == 30)
                sys_pm_reboot(REBOOT_TYPE_GOTO_APP);
}
```

2.1 测试

测试步骤如下：

1. 分别编译及烧写 loader、hello_0、hello_1 三个工程。

2. 打开 pc 端串口工具 (SecureCRT)，串口配置如下：

   - 波特率: 115200
   - 数据位：8
   - 奇偶校验：None
   - 停止位：1
   - 流控：None

3. 按 reset 键后观察串口打印

   开发板上电后，串口每 1s 打印一次“Hello World 0!”，30s 后重启。

   串口每 2s 打印一次“Hello World 1!”，30s 后重启。

   串口每 1s 打印一次“Hello World 0!”，一直重复。

```
-
A
b
M
Hello World 0!
Hello World 0!
Hello World 0!
```

(continues on next page)

(续上页)

```
Hello World 0!
Hello World 0!
Hello World 0!
Hello World 0!
Hello World 0!
Hello World 0!
Hello World 0!
...
...
...
[power] system reboot, type 4608!

-
F
-
D
b
M
Hello World 1!
Hello World 1!
Hello World 1!
Hello World 1!
Hello World 1!
```

CHAPTER 3

Loader 实现简述

3.1 分区表

本示例中，定义了三个分区。分区表的定义如下：

```
__attribute__((section(".partition"))) const struct partition_
↪table part_table_in_nor = {
        .magic = PARTITION_TABLE_MAGIC,
        .part_cnt = 5,
        .part_entry_size = sizeof(struct partition_entry),
        .parts[0] = {
                .name = "fw0_boot",
                .type = BOOT_TYPE,
                .offset = LOADER_A_NOR_ADDR,
        },
        .parts[2] = {
                .name = "fw0_app",
                .type = SYSTEM_TYPE,
                .offset = APP_A_NOR_ADDR,
        },
        .parts[4] = {
                .name = "dtm",
                .type = DTM_TYPE,
                .offset = DTM_NOR_ADDR,
        },
};
```

- loader 对应 parts[0], 分区类型为 BOOT_TYPE。
- hello_0 对应 parts[2], 分区类型为 SYSTEM_TYPE。

- hello_1 对应 parts[4], 分区类型为 DTM_TYPE。

3.2 Loader

loader 中通过判断启动分区类型来选择需要加载的分区（上电后默认为 SYSTEM_TYPE）。

```
if (keycode == KEY_ADFU) {
        /* enter uart adfu */
} else if (keycode == KEY_DTM) {
        /* load dtm */
        ret = boot_app_by_type(DTM_TYPE);
} else {
        /* load application*/
        ret = boot_app_by_type(SYSTEM_TYPE);
}
```

3.3 Hello_0

hello_0 在运行 30s 之后，会调用 sys_pm_reboot(REBOOT_TYPE_GOTO_DTM) 设置启动参数为 DTM_TYPE 后重启

```
static int time = 0;
while (1) {
        printk("Hello World 0!\n");
        k_sleep(1000);
        time++;
        if (time == 30)
                sys_pm_reboot(REBOOT_TYPE_GOTO_DTM);
}
```

3.4 Hello_1

hello_1 在运行 30s 之后，会调用 sys_pm_reboot(REBOOT_TYPE_GOTO_APP) 设置启动参数为 SYSTEM_TYPE 后重启

```
static int time = 0;
while (1) {
        printk("Hello World 1!\n");
        k_sleep(2000);
        time += 2;
        if (time == 30)
                sys_pm_reboot(REBOOT_TYPE_GOTO_APP);
}
```

3.5 其它

1. 本示例中通过调用在应用中调用 sys_pm_reboot 来设置启动加载的分区类型。实际项目中可以在 Loader 中通过检测按键或 gpio 状态来指定加载分区的类型。
2. 更多关于分区、启动的资料可以参考《ZS110A SDK 架构介绍》。

List of Figures

List of Tables