Nuclei™ N100 系列

处理器内核快速应用手册

版权声明

联系我们

若您有任何疑问，请通过电子邮件 support@nucleisys.com 联系芯来科技。

修订历史

版本号	修订日期	修订的章节	修订的内容
1.0	2019/11/19	N/A	1. 初始版本
2.0	2020/1/22	N/A	1. 修正了部分文字说明

目录

版权声明 **0**

联系我们 **0**

修订历史 **1**

图片清单 **4**

1. 概述 **5**

2. 背景知识——N100 系列处理器内核 **6**

2.1. 指令集介绍 6
2.2. CSR 寄存器 6
2.3. N100 系列内核特权架构介绍 6
2.4. N100 系列内核异常机制 6
2.5. N100 系列内核中断机制 7
2.6. N100 系列内核 TIMER 和 IRQC 介绍 7
2.6.1. TIMER 简介 *7*
2.6.2. IRQC 简介 *7*
2.6.3. 中断屏蔽 *8*
2.6.4. 中断级别、优先级与仲裁 *8*
2.6.5. 中断服务程序 *8*
2.6.6. 中断嵌套 *9*
2.6.7. 中断咬尾 *9*

3. 背景知识——N100 系列内核配套 SOC **10**

3.1. N100 系列内核配套 SoC 框图 10
3.2. N100 系列内核配套 SoC 储存资源 10
3.3. N100 系列内核配套 SoC 外设介绍 11

4. 应用实例解析——DEMO_IRQC **12**

4.1. DEMO_IRQC 简述 12
4.2. DEMO_IRQC 程序代码结构 12
4.3. DEMO_IRQC 程序解析 14
4.3.1. 主程序流程图 *14*
4.3.2. 用户按键中断处理函数流程图 *15*
4.3.3. 快速移植中断应用 *15*

5. DEMO_IRQC 软硬件快速启动 **18**

5.1. 第一步：硬件线缆连接 18
5.2. 第二步：设置下载器在 LINUX 系统中的 USB 权限 19
5.3. 第三步：连接外部中断引脚至拨码开关引脚 19
5.4. 第四步：将 DEMO_IRQC 程序下载至评估板并运行 20

图片清单

图 2-1 中断与内核关系结构图 7
图 3-1 N100 系列内核配套 SoC 结构图 10
图 4-1 主程序流程图 14
图 4-2 按键中断处理函数流程图 15
图 5-1 评估板及下载器线缆连接图 19
图 5-2 连接按键至 MCU_GPIO 图 20
图 5-3 运行 DEMO_IRQC 示例后于主机串口终端上显示信息 21

1. 概述

N100 内核用户将通过本文，快速掌握该内核应用开发的必要知识点。包括：内核及配套 SoC 的背景知识介绍、应用实例解析及其软硬件快速启动方法。

中断应用,是 N100 内核应用中的重点和难点。本文第一个应用实例，将着重讲解 N100 内核的中断使用方法。通过对例程解析，使用户在最短的时间内，熟练掌握中断应用并移植到定制化应用项目的开发中。

为了使用户能对以上介绍的应用实例，进行快速启动和实践操作。本文将基于芯来科技推出的蜂鸟 FPGA 评估板，介绍硬件快速启动方法和应用实例的调试运行流程。

2. 背景知识——N100 系列处理器内核

Nuclei N100 处理器内核（简称 N100 内核）主要面向极低功耗与极小面积的场景而设计，非常适合于替代传统的 8051 内核或者 ARM Cortex-M0 系列内核应用于 IoT 或其他低功耗场景。

详情请参见《Nuclei_N100 系列简明数据手册》了解其详情。

2.1. 指令集介绍

N100 内核遵循标准的 RISC-V 指令集标准。

详情请参见《Nuclei_N100 系列指令架构手册》第 1.1 节和第 1.2 节，了解其详情。

2.2. CSR 寄存器

RISC-V 的架构中定义了一些控制和状态寄存器（Control and Status Register，CSR），用于配置或记录一些运行的状态。CSR 寄存器是处理器核内部的寄存器，使用其专有的 12 位地址编码空间。详情请参见《Nuclei_N100 系列指令架构手册》第 1.3 节了解其详情。

2.3. N100 系列内核特权架构介绍

N100 系列内核支持一个特权模式（Privilege Modes）：机器模式（Machine Mode）。

2.4. N100 系列内核异常机制

异常（Exception）机制，即处理器核在顺序执行程序指令流的过程中突然遇到了异常的事情而中止执行当前的程序，转而去处理该异常。异常发生后，处理器会进入异常服务处理程序。

详情请参见《Nuclei_N100 系列指令架构手册》第 3 章了解其详情。

2.5. N100 系列内核中断机制

中断（Interrupt）机制，即处理器核在顺序执行程序指令流的过程中突然被别的请求打断而中止执行当前的程序，转而去处理别的事情，待其处理完了别的事情，然后重新回到之前程序中断的点继续执行之前的程序指令流。详情请参见《Nuclei_N100 系列指令架构手册》第 4 章了解其详情。

2.6. N100 系列内核 TIMER 和 IRQC 介绍

2.6.1. TIMER 简介

计时器单元（Timer Unit，TIMER），在 N100 系列内核中主要用于产生计时器中断（Timer Interrupt）和软件中断（Software Interrupt）。详情请参见《Nuclei_N100 系列指令架构手册》第 5.1 节了解其详情。

2.6.2. IRQC 简介

N100 系列内核支持在 RISC-V 标准 CLIC 基础上简化而来的“私有中断控制器（Priviate Interrupt Controller，简称 IRQC）”，用于管理所有的中断源。详情请参见《Nuclei_N100 系列指令架构手册》第 4.3 节了解其详情。

IRQC 单元生成一根中断线，发送给处理器内核（作为中断目标），其关系结构如图 2-1 所示。

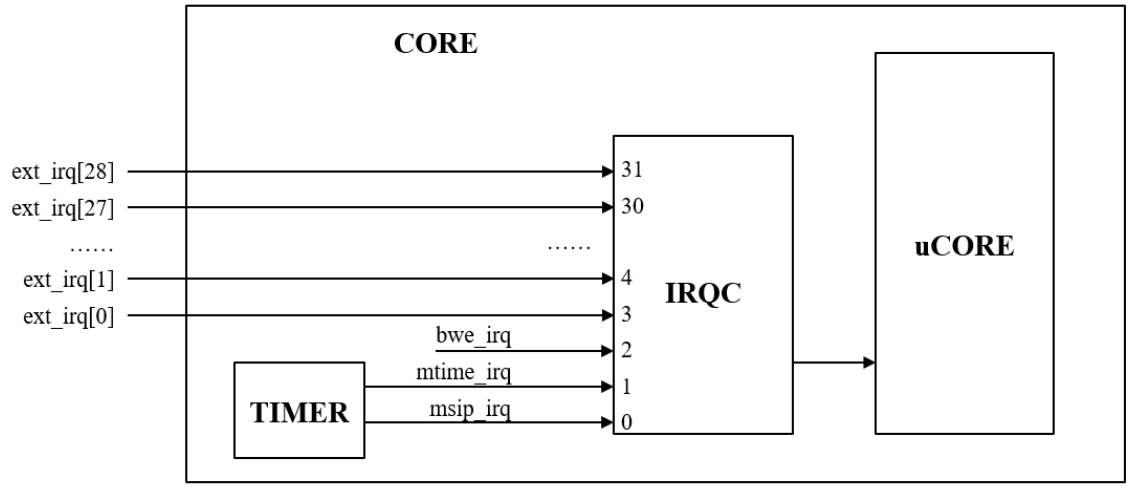

图 2-1 中断与内核关系结构图

2.6.3. 中断屏蔽

N100 内核的中断可以被屏蔽掉， CSR 寄存器 mstatus 的 MIE 域控制中断的全局使能。

对于不同的中断类型而言，IRQC 统一管理外部中断、软件中断以及计时器中断。IRQC 为每个中断源分配了各自的中断使能寄存器，用户可以通过配置 IRQC 寄存器来管理各个中断源的屏蔽，达到使能或屏蔽某一个对应的中断的效果。详情请参见《Nuclei_N100 系列指令架构手册》第 4.4 节了解其详情。

2.6.4. 中断级别、优先级与仲裁

当多个中断同时出现时，需要进行仲裁。对于 N100 系列内核处理器而言，IRQC 统一管理所有的中断。IRQC 的每个中断源有固定的的优先级（中断源的编号越大则优先级越高），当多个中断同时发生时，IRQC 会仲裁出优先级最高的中断，如图 2-2 中所示。详情请参见《Nuclei_N100 系列指令架构手册》第 4.5 节了解其详情。

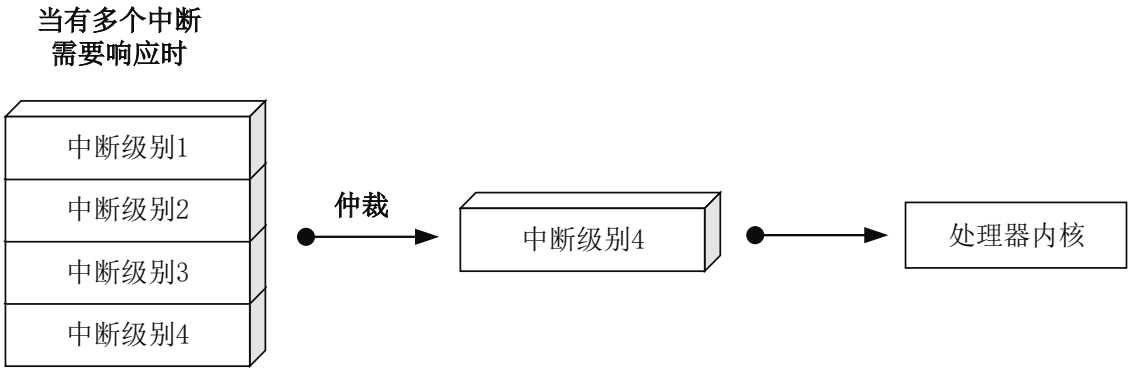

图 2-2 中断仲裁示意图

在下文讲解的 Demo_irqc 实例中，按键 1 外部中断和按键 2 外部中断的使用的中断源编号，分别为 3、4。当这 2 个中断并发时，经过仲裁后，内核将优先响应优先中断源编号为 4 的中断，即按键 2 外部中断。

2.6.5. 中断服务程序

当处理器内核响应并进入中断后，立即根据中断源编号跳转至该中断对应的中断服务程序开始

执行。譬如下文讲解的 Demo_irqc 实例中，函数 button_1_handler()是按键 1 外部中断的服务程序，函数 button_2_handler()是按键 2 外部中断的服务程序。

注意：Nuclei 的 N200 及以上型号（如 N200，N300，N600，N900 等）支持向量模式和非向量两种模式。

2.6.6. 中断嵌套

N100 系类内核主要面向超低功耗场景的超小面积的实现，为了简化设计，缩减面积，N100 系类内核不支持中断嵌套。

注意：Nuclei 的 N200 及以上型号（如 N200，N300，N600，N900 等）支持中断嵌套特性。

2.6.7. 中断咬尾

N100 系类内核主要面向超低功耗场景的超小面积的实现，为了简化设计，缩减面积，N100 系类内核不支持中断咬尾。

注意：Nuclei 的 N200 及以上型号（如 N200，N300，N600，N900 等）支持中断咬尾特性。

3. 背景知识——N100 系列内核配套 SoC

N100 系列内核配套的 MCU 级别 SoC，该 SoC 用于 N100 系列处理器内核的原型验证和用户评估。

3.1. N100 系列内核配套 SoC 框图

N100 系列内核配套 SoC 的框图如图 3-1 所示：

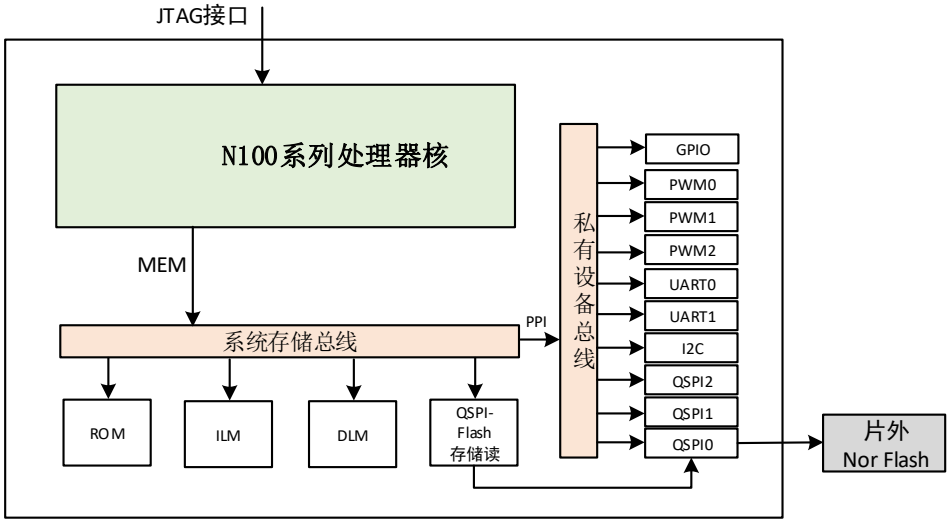

图 3-1 N100 系列内核配套 SoC 结构图

3.2. N100 系列内核配套 SoC 储存资源

如图 3-1 中所示，N100 系列内核配套 SoC 中的存储器资源分为片上存储资源和片外 Flash 存储资源。

有关 N100 系列内核配套 SoC 存储资源的详细介绍，请参见文档《Nuclei_N100 系列配套 SoC 介绍》。

3.3. N100 系列内核配套 SoC 外设介绍

有关 N100 系列内核配套 SoC 外设的详细介绍，请参见文档《Nuclei_N100 系列配套 SoC 介绍》。

4. 应用实例解析——Demo_irqc

4.1. Demo_irqc 简述

Demo_irqc 程序是一个完整的示例程序，相比 Dhrystone 和 CoreMark 这样纯粹的跑分程序，Demo_irqc 更加接近一个常见的嵌入式应用程序，它使用到了 SoC 系统中的外设，调用了中断服务程序等，其功能简述如下：

- 通过 printf 函数输出一串“RISC-V”logo 的字符，printf 输出将会通过 UART 串口重定向至主机 PC 的屏幕上。
- 等待通过 getc 函数输入一个字符，然后将得到的字符通过 printf 输出到主机 PC 的屏幕上。
- 进入死循环不断地对 SoC 的 GPIO 13 的输出管脚进行翻转，如果使用示波器观测此 GPIO 输出管脚，可以看到其产生规律的输出方波。
- 评估板上的两个用户按键，通过杜邦线连接到 SoC 的 GPIO 管脚，这两个 GPIO 管脚各自作为一个 IRQC 的外部中断，在其中断服务程序中会将对 GPIO 的输出管脚（对应三色灯的蓝灯和绿灯）进行设置，从而造成评估板上三色灯的颜色发生变化。

4.2. Demo_irqc 程序代码结构

用户可以通过 GitHub 链接：https://github.com/riscv-mcu/n100-sdk，获取完整程序代码。

在 n100-sdk 环境中，Demo_irqc 示例程序的相关代码结构如下所示。

```
n100-sdk                              // 存放 n100-sdk 的目录
         |----software                  // 存放示例程序的源代码
              |----Demo_irqc            // Demo_irqc 示例程序目录
                   |----demo_irqc.c     //Demo_irqc 的示例 Demo 程序代码
                   |----Makefile        //Makefile 脚本
```

Makefile 为主控制脚本，其代码片段如下：

```
    //指明生成的 elf 文件名
TARGET = Demo_irqc

    //指明 Demo_irqc 程序所需要的特别的 GCC 编译选项
CFLAGS += -O2

BSP_BASE = ../../bsp
```

```
    //指明 Demo_irqc 程序所需要的 C 源文件
C_SRCS += Demo_irqc.c

    //调用板级支持包（bsp）目录下的 common.mk
include $(BSP_BASE)/core/env/common.mk
```

4.3. Demo_irqc 程序解析

4.3.1. 主程序流程图

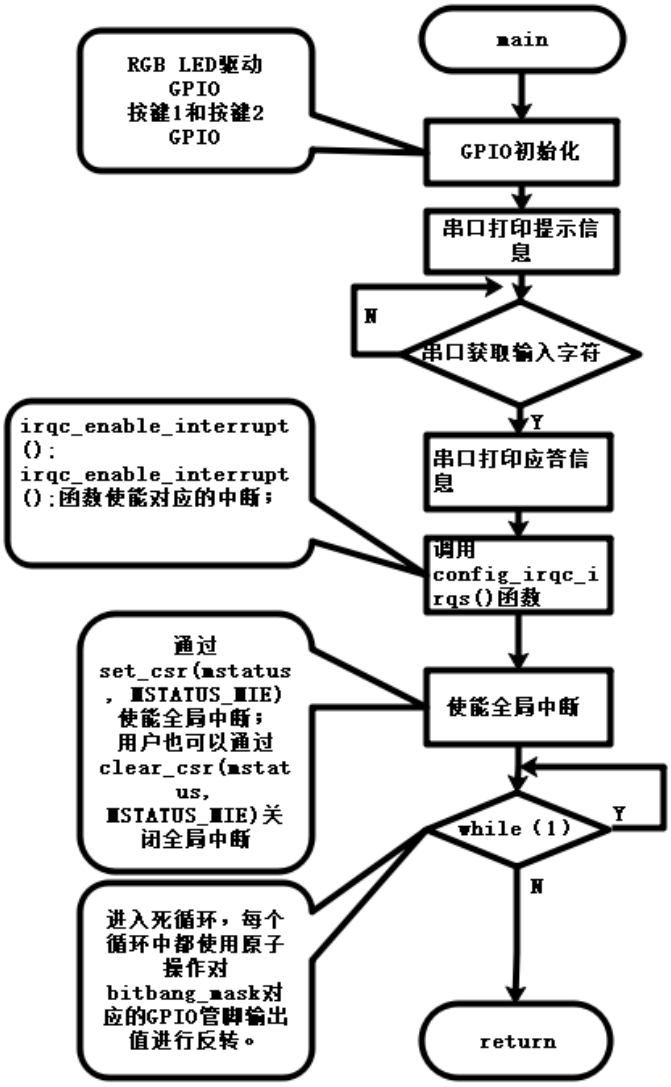

图 4-1 主程序流程图

4.3.2. 用户按键中断处理函数流程图

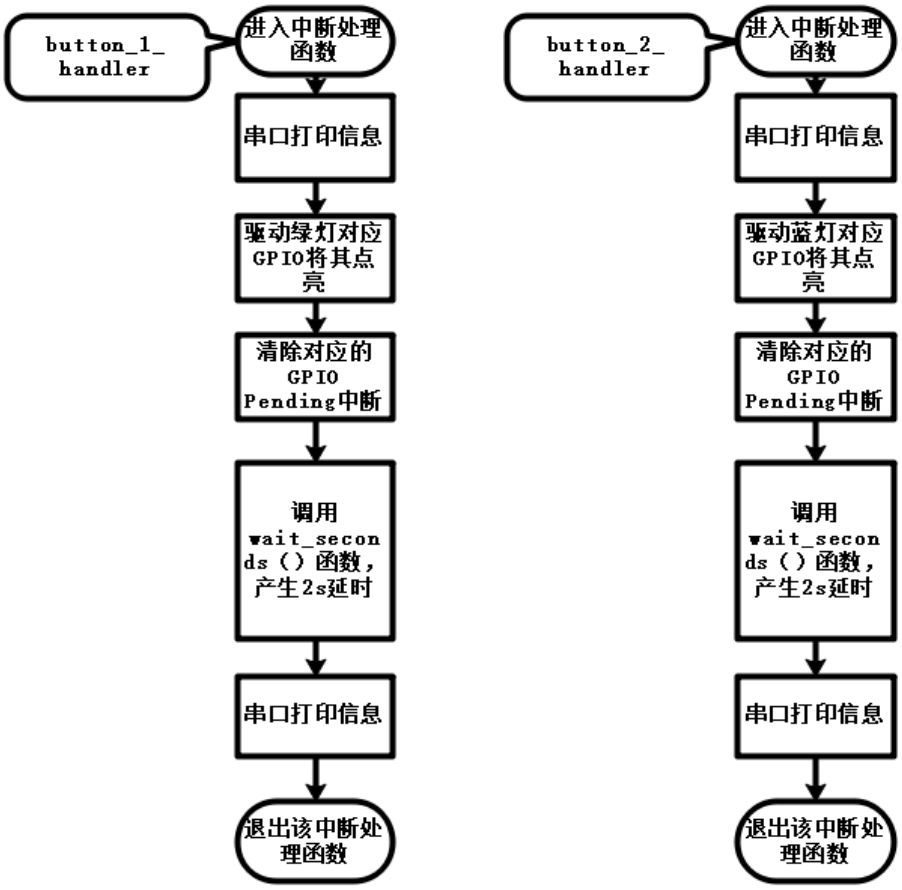

图 4-2 按键中断处理函数流程图

4.3.3. 快速移植中断应用

本节将结合 Demo_irqc 源码来详细介绍如何快速使用 N100 系列处理器内核中断。

- 步骤一：
  - 外部中断 GPIO 初始化

```
// n100-sdk/software/Demo_irqc/Demo_irqc.c 代码片段

#define BUTTON_1_GPIO_OFFSET 0
#define BUTTON_2_GPIO_OFFSET 1
//按键 1 对应管脚为 MCU_GPIO 0
//按键 2 对应管脚为 MCU_GPIO 1

#define IRQC_INT_DEVICE_BUTTON_1 (SOC_IRQC_INT_GPIO_BASE + BUTTON_1_GPIO_OFFSET)
#define IRQC_INT_DEVICE_BUTTON_2 (SOC_IRQC_INT_GPIO_BASE + BUTTON_2_GPIO_OFFSET)
```

```
//配置 IRQC 中断源映射

//设置评估板上按键相关的 GPIO 寄存器

//通过“与”操作将 GPIO_OUTPUT_EN 寄存器某些位清 0，即将评估板按键对应的 GPIO 输出使能关闭
  GPIO_REG(GPIO_OUTPUT_EN)  &=
          ~(
              (0x1 << BUTTON_1_GPIO_OFFSET) |
              (0x1 << BUTTON_2_GPIO_OFFSET)
          );

//通过“与”操作将 GPIO_PULLUP_EN 寄存器某些位清 0，即将评估板按键对应的 GPIO 输入上拉关闭
  GPIO_REG(GPIO_PULLUP_EN)  &=
          ~(
               (0x1 << BUTTON_1_GPIO_OFFSET) |
              (0x1 << BUTTON_2_GPIO_OFFSET)
          );

//通过“或”操作将 GPIO_INPUT_EN 寄存器某些位设置为 1，即将评估板按键对应的 GPIO 输入使能关闭
  GPIO_REG(GPIO_INPUT_EN)   |=
          (
              (0x1 << BUTTON_1_GPIO_OFFSET) |
              (0x1 << BUTTON_2_GPIO_OFFSET)
          );

//通过“或”操作将 GPIO_RISE_IE 寄存器某些位设置为 1，即将评估板按键对应的 GPIO 管脚设置为上升沿触发的中断来源
  GPIO_REG(GPIO_RISE_IE) |= (1 << BUTTON_1_GPIO_OFFSET);
  GPIO_REG(GPIO_RISE_IE) |= (1 << BUTTON_2_GPIO_OFFSET);
```

- 步骤二：IRQC 初始化

Demo_irqc 例程中，使用函数 config_irqc_irqs ()完成对计时器中断和两个按键中断的初始化。其代码片段如下：

```
// n100-sdk/software/Demo_irqc/Demo_irqc.c 代码片段

//设置 irqc_irq3 的中断处理函数为 BUTTON_1_HANDLER
#define BUTTON_1_HANDLER irqc_irq3_handler
//设置 irqc_irq4 的中断处理函数为 BUTTON_2_HANDLER
#define BUTTON_2_HANDLER irqc_irq4_handler

void config_irqc_irqs(){

  //通过配置 IRQC 的寄存器使能“”计时器中断“和”“评估板两个按键的 GPIO 中断”。注意：//IRQC_enable_interrupt 的函数原型定义在 bsp/core/drivers/func.c 中

  irqc_enable_interrupt (IRQC_INT_DEVICE_BUTTON_1);
  irqc_enable_interrupt (IRQC_INT_DEVICE_BUTTON_2);


//评估板两个按键 BT1 和 BT2 的 GPIO 中断分别使用优先级为 3 和 4 的中断
```

```
}
```

- 步骤三：编写中断处理函数

在蜂鸟 FPGA 评估板上，例程中使用到的两个用户按键，需要使用杜邦线将 GPIO_BTN1 和 GPIO_BTN2 连接到 GPIO0 和 GPIO1 上，这两个 GPIO 管脚各自作为一个 IRQC 的外部中断。Demo_irqc 使用这两个外部中断，所以定义了 button_1_handler 和 button_2_handler 分别作为它们的中断处理函数，其代码如下：

```
  //向量处理模式时，由于在跳入中断服务程序之前，处理器并没有进行上下文的保存，因此，理论上中断服务程序函数本身不能够进行子函数的调用（即必须是 Leaf Function）。
  //此处 BUTTON_1_HANDLE 明显会调用其他的子函数，如果不加处理则会造成功能的错误。为了规避这种情形，当中断被设置为“向量模式”时，用户必须使用特殊的 __attribute__ (( interrupt)) 来修饰该中断服务程序函数，那么编译器会自动的进行判断，当编译器发现该函数调用了其他子函数时，便会自动的插入一段代码进行上下文的保存。

void __attribute__ ((interrupt)) BUTTON_1_HANDLER(void) {
  printf ("%s","----Begin button1 handler\n");

    // 点亮绿灯
  GPIO_REG(GPIO_OUTPUT_VAL) |= (1 << GREEN_LED_GPIO_OFFSET);

  GPIO_REG(GPIO_RISE_IP) = (0x1 << BUTTON_1_GPIO_OFFSET);

   // 等待 2 秒钟
  wait_seconds(2);

  printf ("%s","----End button1 handler\n");


};


void __attribute__ ((interrupt)) BUTTON_2_HANDLER(void) {


  printf ("%s","--------Begin button2 handler\n");

    // 点亮蓝灯
  GPIO_REG(GPIO_OUTPUT_VAL) |= (1 << BLUE_LED_GPIO_OFFSET);

  GPIO_REG(GPIO_RISE_IP) = (0x1 << BUTTON_2_GPIO_OFFSET);

  // 等待 2 秒钟
  wait_seconds(2);

  printf ("%s","--------End button2 handler\n");
};
```

有关 N100 系列内核中断机制以及中断控制器的详细介绍，请参见文档《Nuclei_N100 系列指令架构手册》第 4 章和第 5 章内容。

5. 快速上手——基于 n100-sdk 的 Demo_irqc 软硬件快速上手

用户购买到的蜂鸟 FPGA 评估板，在出厂时如果已经预烧写了 N100 内核及其配套 SoC，用户可以直接把评估板，“当作”一颗集成了 N100 系列内核的 MCU 来进行开发和调试。如果用户需要自己烧写 N100 内核及其配套 SoC，关于 FPGA 评估板程序的烧写方法，请用户参见文档《Nuclei_N100 系列配套 FPGA 实现》。

下文将介绍在 Linux 环境下，使用 n100-sdk 来进行软硬件快速启动。SDK 使用 Linux Makefile 进行项目和文件的组织，适用于开发水平较高的用户。

注意：关于 Linux 环境下 SDK 的详细介绍和更多的软件示例，请用户参见《Nuclei_N100 系列 SDK 使用说明》。

5.1. 第一步：硬件线缆连接

- 连接蜂鸟 FPGA 评估板 5V 电源线；
- 将蜂鸟下载器通过排线连接至蜂鸟 FPGA 评估板；
- 将蜂鸟下载器连接至调试主机 PC 的 USB 接口。
- 将评估板电源开关拨至“ON”档，接通电源。

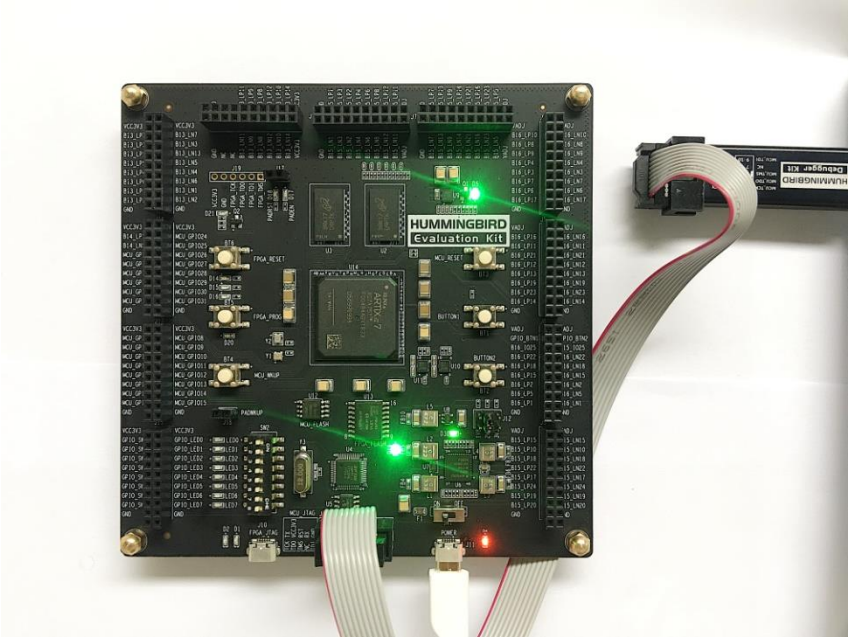

图 5-1 评估板及下载器线缆连接图

5.2. 第二步：设置下载器在 Linux 系统中的 USB 权限

如果使用 Linux 操作系统，需要按照如下步骤保证正确的设置蜂鸟下载器的 USB 权限。详细内容请参见文档《Nuclei_N100 系列 SDK 使用说明》第 2 章。

5.3. 第三步：连接外部中断引脚至拨码开关引脚

Demo_irqc 示例可以支持 2 个外部中断：按键 1 中断和按键 2 中断。在基于蜂鸟 FPGA 评估板运行该例程时，需要将 2 个外部中断引脚使用杜邦线，连接至按键引脚。

Demo_irqc 示例程序中，使用如下代码，将按键 1 和按键 2 外部中断，分别映射到 MCU_GPIO0 和 MCU_GPIO1。

```
#define BUTTON_1_GPIO_OFFSET 0

#define BUTTON_2_GPIO_OFFSET 1
```

所以如图 5-2 所示，用户需要使用两根杜邦线，将 MCU_GPIO0 连接至 GPIO_BTN1，MCU_GPIO1 连接至 GPIO_BTN2。

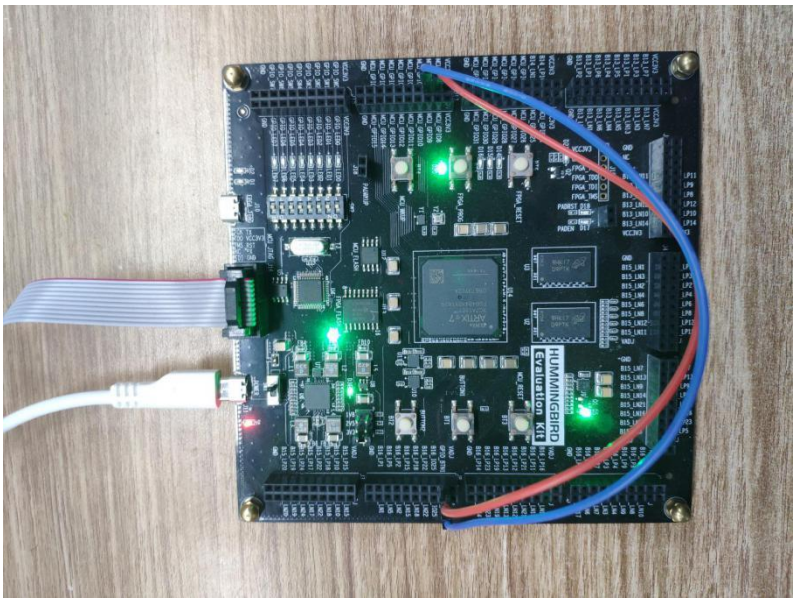

图 5-2 连接按键至 MCU_GPIO 图

5.4. 第四步：将 Demo_irqc 程序下载至评估板并运行

Demo_irqc 示例可运行于 N100 内核及配套 SoC 评估板平台中，使用基于 Linux 的 SDK 来进行演示。关于 Linux 环境下 SDK 的详细介绍，请用户参见《Nuclei_N100 系列 SDK 使用说明》。

具体操作方法按照如下步骤进行：

// 注意：确保在 n100-sdk 中正确的安装了 RISC-V GCC 工具链，请参见《Nuclei_N100 系列 SDK 使用说明》了解其详情。

// 步骤一：参照《Nuclei_N100 系列 SDK 使用说明》中描述的方法，编译 Demo_irqc 示例程序，使用如下命令：

```
make dasm PROGRAM=demo_irqc  CORE=n101    DOWNLOAD=flash
```

//注意：此处指定 DOWNLOAD=flash 选项，则采用“将程序从 Flash 上载至 ILM 进行执行的方式”进行编译，请参见《Nuclei_N100 系列 SDK 使用说明》了解更多详情。

// 步骤二：参照《Nuclei_N100 系列 SDK 使用说明》中描述的方法，将编译好的 Demo_irqc 程序下载至 FPGA 评估板中，使用如下命令：

```
make upload  PROGRAM=demo_irqc  CORE=n101   DOWNLOAD=flash
```

// 步骤三：参照《Nuclei_N100 系列 SDK 使用说明》中描述的方法，在 FPGA 评估板上运行 Demo_irqc 程序：

// 由于示例程序将需要通过 UART 打印结果到主机 PC 的显示屏上。参考《Nuclei_N100 系列 SDK 使用说明》
// 所述方法将串口显示电脑屏幕设置好，使得程序的打印信息能够显示在电脑屏幕上。
//
// 由于步骤二已经将程序烧写进 FPGA 评估板的 Flash 之中，因此每次按评估板的
// MCU_RESET 按键，则处理器复位开始执行 Demo_irqc 程序，并将 RISC-V 字符串打印至主
// 机 PC 的串口显示终端上，如图 5-3 所示，然后用户可以输入任意字符（譬如字母 y），

//程序继续运行，评估板上将会以固定频率进行闪灯。

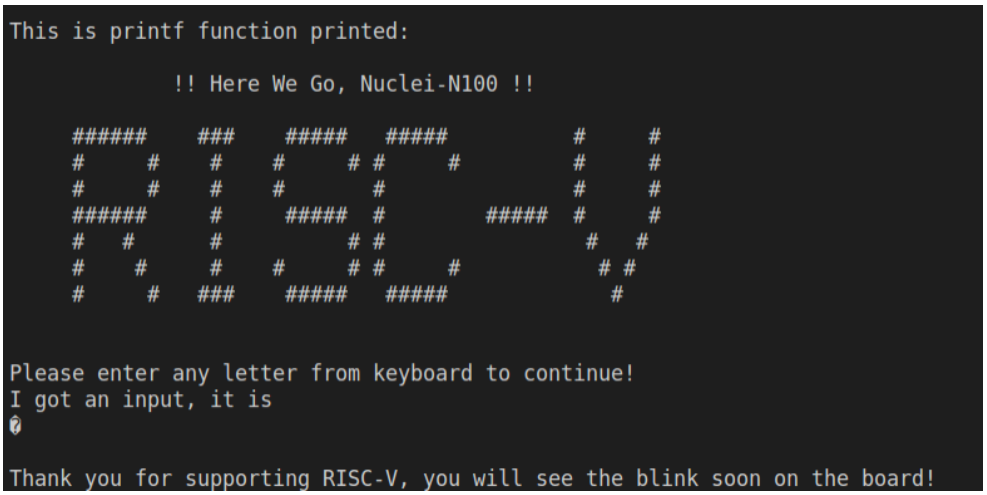

图 5-3 运行 Demo_irqc 示例后于主机串口终端上显示信息

6. 快速上手——基于 Embedded Studio IDE 的 Demo_irqc 软硬件快速上手

用户购买到的蜂鸟 FPGA 评估板，在出厂时如果已经预烧写了 N100 内核及其配套 SoC，用户可以直接把评估板，“当作”一颗集成了 N100 系列内核的 MCU 来进行开发和调试。如果用户需要自己烧写 N100 内核及其配套 SoC，关于 FPGA 评估板程序的烧写方法，请用户参见文档《Nuclei_N100 系列配套 FPGA 实现》。

N100 系列提供图形化集成开发环境（IDE，Integrated Development Environment）——Segger Embedded Studio。用户可以基于上述的 FPGA 评估板，结合 IDE，进行完整的嵌入式软件开发和调试，详细介绍请参见参见单独文档《Nuclei_N100 系列 IDE 使用说明》。

注意：IDE 使用可视化集成开发环境进行项目和文件的组织，适用于习惯图形化开发环境的用户。