RK3399 MCU 开发指南

文件标识：RK-KF-YF-124

发布版本：V1.3.0

日期：2021-01-14

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文档主要介绍Rockchip RK3399 MCU开发的基本方法。

产品版本

芯片名称	内核版本
RK3399	4.4

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2017-09-20	V1.0.0	王明成	初始版本
2017-12-27	V1.1.0	王明成	修订文档格式
2020-08-17	V1.2.0	王明成	修订文档格式，修正代码下载路径等
2021-01-14	V1.3.0	王明成	文档重命名，修正demo目录显示等

Rockchip MCU简介

ARM® Cortex®-M处理器系列具有灵活性、易用性、高性能、低功耗等特点。同时，Cortex-M处理器能够帮助开发者以更低的成本提供更多的功能，其在代码重用和提高开发效率方面有显著优势，所以在嵌入式设备领域的应用非常广泛。如下为Cortex-M0基本简介。

- Cortex-M0采用ARMv6-M结构，基于一个高集成度、低功耗的32位处理器内核；它采用冯·诺伊曼结构，基于16位的Thumb指令集，并包含Thumb-2技术。

基于以上ARM® Cortex®-M优点，目前，Rockchip SoC上集成的MCU说明如下：

- RK3399 集成2个Cortex-M0，其一PMU M0为ATF所用，其二Perilp M0开放给客户使用。

开发基础

运行前配置

本章节主要以RK3399 Perilp M0为例介绍Rockchip MCU开发的基础方法。

启动地址

以常见miniloader + ATF + u-boot的启动方式为例。

采用这种启动方式，通常将MCU代码编译生成的BIN和ATF的BIN一起打包为trust.img，因此，需要在u-boot添加如下打包配置。

```
tools/rk_tools/RKTRUST/RK3399TRUST.ini
...
[BL30_OPTION]
SEC=1
PATH=tools/rk_tools/bin/rk33/rk3399bl30_v1.00.bin
ADDR=0x00080000
...
```

其中，PATH为MCU BIN文件存放路径，ADDR为MCU在DDR中被加载的地址（MCU的0地址）。当然这个地址需要是在u-boot中reserve出来的一段安全地址。这个地址会传递给miniloader，其会负责将MCU的代码从ROM中加载到DDR的这个地址处。

当然，如果使用u-boot或其它loader作为一级boot loader，也可参考上面的方法对M0的固件进行打包和加载。

MCU BIN编译方法参阅3.2.2章节。

地址映射

Coretex-M0/Coretex-M3拥有固定的Memory Map，这样方便软件在不同系统之间的轻松移植，其地址空间被分为许多不同的段，可参阅Cortex-M0 Devices Generic User Guide
Chapter 2.2章节Memory model和RK3399 TRM Chapter 7.4.2章节。通常，我们只需要配置MCU的0x00000000-0x1FFFFFFF地址映射。

注意，RK3399 M0的地址映射需要通过SGRF配置，所以务必在能访问SGRF的模块进行配置（通常放在miniloader或ATF中进行），以2.1.1章节中的加载地址为例，具体内存映射配置方式如下：

- 启动地址配置

```
sgrf_perilp_m0_con7 = 0xf << (4 + 16) | (0x080000 >> 28) & 0x0f
sgrf_perilp_m0_con15 = 0xffff << 16 | (0x80000 >> 12) & 0xffff
```

- 外设地址配置

Rockchip MCU已默认配置了外设的映射地址(0x40000000-0x5FFFFFFF)，即：

ADDR_MCU = ADDR_CA72 – 0xB8000000

这里的外设就是RK3399 TRM Chapter 2 System Overview中所列出的外设。

时钟配置

Rockchip MCU 时钟源可选择CPLL或GPLL，可参考RK3399 TRM Chapter 3 CRU章节。在其章节中指出clk配置寄存器为CRU_CLKSEL_CON24(0x0160)，其中：

bit[15] : 时钟源选择，1’b0: CPLL；1’b1: GPLL
bit[12:8] : 分频设置，用于配置MCU的运行频率。

- u-boot 参考代码

```
arch/arm/cpu/armv8/rk33xx/clock-rk3399.c
#ifdef CONFIG_PERILP_MCU
    /* peril m0 clk = 300MHz, select gpll as the source clock */
    clk_parent_hz = RKCLK_GPLL_FREQ_HZ;
    clk_child_hz = 300000000; /* HZ */
    div = rkclk_calc_clkdiv(clk_parent_hz, clk_child_hz, 1);

    div = div ? (div - 1) : 0;
    cru_writel((1 << 31) | (0x1F << 24) | (1 << 15) | (div << 8),
    CRU_CLKSELS_CON(24));
#endif
```

复位撤销

MCU运行起来的最后一步就是进行复位撤销。其寄存器信息阅RK3399 TRM Chapter 3 CRU章节。
RK3399 Perilp M0的复位撤销寄存器为：

PMUCRU_SOFTRST_CON0(0x0110)
需要设置PMUCRU_SOFTRST_CON0[5:0] = 4b’0000.

- u-boot 参考代码

```
arch/arm/cpu/armv8/rk33xx/clock-rk3399.c
#ifdef CONFIG_PERILP_MCU
    /* perilp m0 dereset */
    cru_writel(0x00160000, CRU_SOFTRSTS_CON(11));
#endif
```

提示：时钟和复位的配置可放在最后期望MCU跑起来的地方，目前Rockchip SDK是放在u-boot当中。

其它配置

JTAG使能配置

MCU开发过程中，常常需要借助于JTAG来跟踪、调试和解决问题。Rockchip MCU JTAG接口采用SWD（2线）模式，需要配置JTAG iomux后才能连接上。

RK3399 Perilp M0的iomux配置信息详见RK3399 TRM Chapter 7.3章节，包括如下两个寄存器。

```
GRF_GPIO4B_IOMUX[9:8] = 2’b10
GRF_GPIO4B_IOMUX[9:8] = 2’b10
```

MCU与主控通信

Mailbox

Rockchip SoC上集成的mailbox拥有4个通道，通过中断触发，数据通过共享内存传递。Rockchip MCU可通过mailbox外设与主控通信。RK3399 Mailbox编程可参阅RK3399 TRM Chapter 21 Mailbox章节；RK3368 Mailbox参考RK3368 TRM Chapter 11 Mailbox章节。

目前Linux 4.4 Kernel中，Mailbox Driver框架上层使用ARM SCPI协议，因此需要在Kernel开启CONFIG_RK3368_MBOX和CONFIG_RK3368_SCPI_PROTOCOL两个配置。同时，MCU这边代码也需要编写mailbox驱动和scpi协议支持。

Kernel DTS可参考下面代码进行配置。

```
mailbox: mailbox@ff6b0000 {
    compatible = "rockchip,rk3368-mbox-legacy";
    reg = <0x0 0xff6b0000 0x0 0x1000>,
          <0x0 0xff8cf000 0x0 0x1000>; /* the end 4k of sram */
    interrupts = <GIC_SPI 146 IRQ_TYPE_LEVEL_HIGH>,
                 <GIC_SPI 147 IRQ_TYPE_LEVEL_HIGH>,
                 <GIC_SPI 148 IRQ_TYPE_LEVEL_HIGH>,
                 <GIC_SPI 149 IRQ_TYPE_LEVEL_HIGH>;
    clocks = <&cru PCLK_MAILBOX>;
    clock-names = "pclk_mailbox";
    #mbox-cells = <1>;
    status = "disabled";
};

mailbox_scpi: mailbox-scpi {
    compatible = "rockchip,rk3368-scpi-legacy";
    mboxes = <&mailbox 0>, <&mailbox 1>, <&mailbox 2>;
    chan-nums = <3>;
    status = "disabled";
};
```

共享内存

Rockchip MCU亦可通过共享内存方式与主控通信，比如在INTMEM (SRAM)中划分一块空间，将其配置成对主控和MCU均可访问，即可实现共享内存方式通信。

Rockchip MCU还可以通过UART或其它方式与主控通信。

Demo程序

代码获取

Git仓库路径：

- https://github.com/rockchip-linux/mcu
- 参考rk3399-box-m0分支。

代码简介

目录结构

```
rk-mcu>ls -R
.:
build  include  Makefile  src

./build:
arm-gcc-link.ld  RK3399M0

./build/RK3399M0:
bin  obj
```

```
./build/RK3399M0/bin:
RK3399M0.bin  RK3399M0.dump  RK3399M0.elf  RK3399M0.map

./build/RK3399M0/obj:
main.o  startup.o

./include:
mcu.h rk3399.h

./src:
main.c startup.c
```

- build：用于存放编译生成的obj文件和bin文件。
- include：代码头文件。
- src：代码C文件。

  startup.c文件为M0入口程序，主要包括M0中断向量表和中断执行函数。
  main.c文件为M0程序的main函数。

编译方法

交叉编译工具链使用gcc-arm-none-eabi- v4.8版本或以上。
编译方法如下：

```
rk-mcu>make help
usage: make PLAT=<RK3399M0> <all|clean|distclean>

PLAT is used to specify which platform you wish to build.
If no platform is specified in first time, PLAT defaults to:

Supported Targets:
  all   Build all the project
  clean Clean the current platform project
  distclean Clean the current project and delete .config

example: build the targets for the RK3399M0 project:
  make PLAT=RK3399M0
```

编译后会生成build/RK3399M0/bin/RK3399M0.bin文件，将RK3399M0.bin拷贝到u-boot目录中tools/rk_tools/bin/rk33/目录，重命名为rk3399bl30_v1.00.bin；然后按照2.1.1章节配置，重新编译u-boot，即可将M0的bin打包到trust.img。

中断编程

M0中断向量表可参阅Cortex-M0 Devices Generic User Guide
Chapter 2.3 Exception model章节。对应到Demo程序，即src/startup.c中有如下参考代码：

```
/**
  * The minimal vector table for a Cortex M3.  Note that the proper constructs
  * must be placed on this to ensure that it ends up at physical address
  * 0x00000000.
  */
__attribute__ ((used,section(".isr_vector")))
void (* const g_pfnVectors[])(void) =
{
```

```
    /* core Exceptions */
    (void *)&pstack[STACK_SIZE], /* the initial stack pointer */
    reset_handler,
    nmi_handler,
    hardware_fault_handler,
    0,0,0,0,0,0,0,
    svc_handler,
    0,0,
    pend_sv_handler,
    systick_handler,

    /* external exceptions */
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
};
```

Coretex-M0内部有16个异常，从g_pfnVectors[0]到g_pfnVectors[15]，可根据实际应用需要注册并实现对应的异常处理函数。

Cortex-M0可处理32个外部中断，对应g_pfnVectors[16]到g_pfnVectors[47]。

RK3399 Perilp M0引入了中断仲裁器，将32个外部中断扩展到256个，可参阅RK3399 TRM Chapter 7.4.6 Interrupt Source Arbiter for PERILPM0章节。当然，使用仲裁器，需要配置对应的mask bit，而M0收到中断后，需要根据对应的mask bit来判断具体的中断源。

Arbiter的接口可参考include/rk3399.h中

M0_INT_ARB_SET_MASK() // 设置中断mask
M0_INT_ARB_GET_FLAG() // 获取中断bit

RK3399 Perilp M0支持的外部中断请参阅RK3399 TRM 2.4 System Interrupt Connection for Cortex-M0 章节。

MCU调试

JTAG调试

- GRF中设置JTAG相关iomux、tck、tms，请阅2.2.1章节。
- 开发板JTAG拨码开关或tck/tms开关拨至MCU处；
- DS-5或ICE连接MCU进行调试。

串口打印

- M0可直接访问UART寄存器进行打印调试。
- 如果MCU使用跟主控相同的UART，建议正常运行时关闭M0打印，防止UART访问异常导致系统异常。

读写寄存器

- 可将系统停留到u-boot或Kernel命令行，通过io读取MCU状态寄存器查看MCU状态。
- 也可将MCU关键点的运行状态写入空闲GRF寄存器，然后在u-boot或kernel命令行读取其值判断MCU的当前运行状态。

参考文档

Cortex-M0 Devices Generic User Guide
Cortex-M0 Technical Reference Manual
Rockchip RK3399 TRM V1.4