Rockchip SOC片内MCU开发指南

文件标识：RK-KF-YF-C06

发布版本：V1.0.0

日期：2024-05-27

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

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

本文提供了Rockchip SOC芯片片内开放MCU的开发指南。

产品版本

芯片名称	内核版本
不限	不限

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

版本号	作者	修改日期	修改说明
V1.0.0	陈谋春	2024-05-27	初始版本

目录

1. 概述

Rockchip的SOC芯片大部分都包含一些片内的MCU，这些MCU又分为两类：A) 特定功能专用的MCU（如PMU MCU和DDR MCU等）；B)公用开放的MCU，这类MCU会提供开发环境给客户做二次开发（如BUS MCU)。本文只作为后者的开发指南，每颗SOC的MCU分类以Datasheet和TRM文档为准。

2. 环境搭建

我们的服务器是通过SSH非对称加密认证的，以下的环境搭建都假设你已经正确配置了自己的SSH的密钥，如果新机器需要配置密钥，请自己参考SSH密钥的配置的相关文档。

2.1 repo工具

```
git clone ssh://10.10.10.29:29418/android/tools/repo -b stable
export PATH=/path/to/repo:$PATH
```

2.2 Toolchain

```
sudo apt-get install gcc-arm-embedded scons clang-format astyle libncurses5-dev
build-
essential python-configparser
wget https://developer.arm.com/-/media/Files/downloads/gnu/13.2.rel1/binrel/arm-
gnu-
toolchain-13.2.rel1-x86_64-arm-none-eabi.tar.xz
tar xvf arm-gnu-toolchain-13.2.rel1-x86_64-arm-none-eabi.tar.xz
# for RT-Thread
export RTT_EXEC_PATH=/path/to/toolchain/arm-gnu-toolchain-13.2.Rel1-x86_64-arm-
none-
eabi/bin
# for HAL
export PATH=/path/to/toolchain/arm-gnu-toolchain-13.2.Rel1-x86_64-arm-none-
eabi/bin:$PATH
```

2.3 HAL工程

用于编译Bare Metal的固件，如果跑纯RTOS可以跳过

```
mkdir hal
cd hal
repo init --repo-url ssh://10.10.10.29:29418/android/tools/repo -u
ssh://10.10.10.29:29418/rk/mcu/hal/manifest -b master
.repo/repo/repo sync
```

2.4 RT-Thread工程

用于编译RTOS固件，如果跑纯Bare Metal可以跳过

```
mkdir rt-thread
cd rt-thread
repo init --repo-url ssh://10.10.10.29:29418/android/tools/repo -u
ssh://10.10.10.29:29418/rtos/rt-thread/manifests -b master -m v4.1.x.xml
.repo/repo/repo sync
```

2.5 U-Boot工程

```
git clone ssh://10.10.10.29:29418/rk/rkbin
git clone ssh://10.10.10.29:29418/android/rk/u-boot
wget https://releases.linaro.org/components/toolchain/binaries/6.3-2017.05/arm-
linux-gnueabihf/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz
tar xvf gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf.tar.xz
wget https://releases.linaro.org/components/toolchain/binaries/6.3-
2017.05/aarch64-linux-gnu/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-
gnu.tar.xz
tar xvf gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz
// 32位：
mkdir -p prebuilts/gcc/linux-x86/arm
mv gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf prebuilts/gcc/linux-
x86/arm/gcc-linaro-6.3.1-2017.05-x86_64_arm-linux-gnueabihf
// 64位：
mkdir -p prebuilts/gcc/linux-x86/aarch64
mv gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu prebuilts/gcc/linux-
x86/aarch64/gcc-linaro-6.3.1-2017.05-x86_64_aarch64-linux-gnu/
```

3. MCU固件编译和打包

3.1 Bare Metal编译打包

下面以RK3576的BUS MCU为例，编译和打包命令如下：

```
cd hal/project/rk3576-mcu/GCC
make clean;make -j16
cd ..
./mkimage.sh
ls Image/amp.img -l
-rw-r--r-- 1 cmc cmc 73216  5月 27 16:15 Image/amp.img
```

3.2 RT-Thread编译打包

```
cd rt-thread/bsp/rockchip/rk3576-mcu
scons -j16
./mkimage.sh
ls Image/amp.img -l
-rw-r--r-- 1 rk rk 108032 May 30 11:55 Image/amp.img
```

4. U-Boot编译

```
cd u-boot
./make.sh rk3576-amp
ls -l uboot.img
-rw-r--r-- 1 cmc cmc 4194304  5月 27 16:13 uboot.img
```

5. 固件烧写

为了方便开发，目前服务器上已经放了一份编译好的固件:`smb://10.10.10.164/Linux_Repository/RK3576/1_Software/Firmware/bus_mcu`

5.1 添加amp分区

```
FIRMWARE_VER: 1.0
MACHINE_MODEL: RK3576
MACHINE_ID: 007
MANUFACTURER: RK3576
MAGIC: 0x5041524B
ATAG: 0x00200800
MACHINE: 0xffffffff
CHECK_MASK: 0x80
PWR_HLD: 0,0,A,0,1
TYPE: GPT
CMDLINE:
mtdparts=rk29xxnand:0x00000800@0x00002000(security),0x00002000@0x00004000(uboot),
0x00010000@0x00006000(boot),0x00200000@0x00016000(rootfs),0x00001000@0x0x00216000
(amp),-@0x00217000(userdata:grow)
uuid:rootfs=614e0000-0000-4b53-8000-1d28000054a9
```

5.2 烧写

```
# 更新分区表
~/opt/upgrade_tool_v2.30_for_linux/upgrade_tool di -p ./parameter.txt
# 烧录uboot
~/opt/upgrade_tool_v2.30_for_linux/upgrade_tool di -uboot ./uboot.img
# 烧录amp
~/opt/upgrade_tool_v2.30_for_linux/upgrade_tool di -amp ./amp.img
```

5.3 启动log

```
# uart0，ap端串口，uboot启动mcu的log
## Loading loadables from FIT Image at fbdaff80 ...
   Trying 'mcu' loadables subimage
     Description:  bus_mcu
     Type:         Standalone Program
     Compression:  uncompressed
     Data Start:   0xfbdb0d80
     Data Size:    68432 Bytes = 66.8 KiB
     Architecture: ARM
     Load Address: 0x48200000
     Entry Point:  unavailable
     Hash algo:    sha256
     Hash value:
8431f79deb46340f47c5025ecb3bc90c2faffe07f874aae9c41829facdfb2890
   Verifying Hash Integrity ... sha256+ OK
   Loading loadables from 0xfbdb0d80 to 0x48200000
Handle standalone: 'bus_mcu' at 0x48200000 ...INFO:    busmcu_config 33
func_id:0x1, config:0xffff4820
INFO:    busmcu_config 37 func_id:0x1, config:0xffff4820
OK
# uart5，mcu端，mcu进bare metal的log
Hello RK3576 mcu
# uart5，mcu端，mcu进rtthread的log
\ | /
- RT -     Thread Operating System
```

```
 / | \     4.1.1 build May 30 2024 11:55:49
 2006 - 2022 Copyright by RT-Thread team
main entry
         msh >
```

6. JTAG调试

目前大部分芯片的AP和MCU都是共用一个JTAG，通过GRF寄存器来切换，以RK3576为例，默认JTAG是连到AP的，需要写这两个寄存器才能切到MCU:

```
# 可以在arch/arm/mach-rockchip/rk3576/rk3576.c的fit_standalone_release函数加上如下代码去切
writel(0x003f0010, SYS_GRF_BASE + SYS_GRF_SOC_CON7);
writel(0xff009900, TOP_IOC_BASE + GPIO2A_IOMUX_SEL_L);
# 也可以在AP端控制台通过io命令去写这两个寄存器
```

这些寄存器配置，请以具体芯片的TRM文档介绍为准。