龙芯中科
LOONGSON TECHNOLOGY

2K0300先锋派Openharmony开发介绍

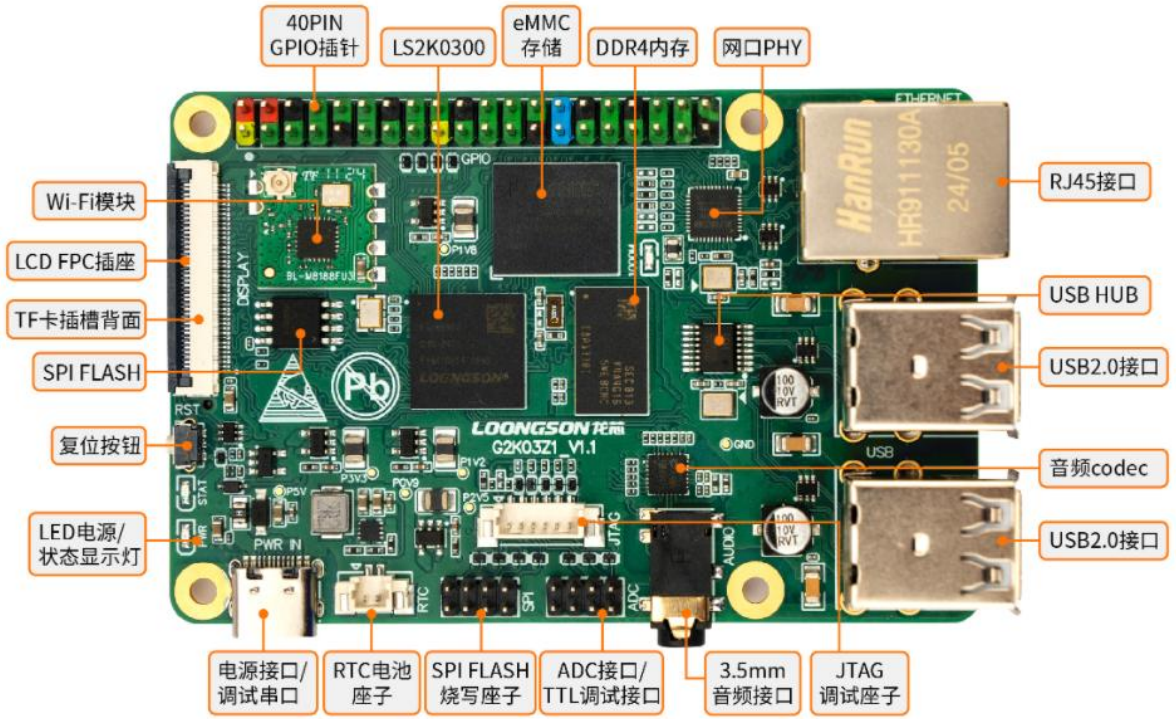
广东龙芯中科 毛小川

2024年11月14日

龙芯中科
LOONGSON TECHNOLOGY

目录

1.1 2K0300先锋派--板卡接口简介



2K0300先锋派

处理器	型号	Loongson 2K0300
	核数	1个LA264处理器核
	主频	主频1.0GHz
内存	类型	DDR4
	容量	512MB
存储	SPI Flash	1*SPI Flash, 容量2MB，存储启动固件uBoot
	eMMC	1*eMMC，容量8GB
接口	网络	1*千兆以太网RJ45接口
	LCD	1*RGB LCD，支持24位色输出，最大分辨率支持1080P
	Audio	1*3.5mm音频接口，支持录音播放
	USB	4*USB2.0 HOST
	WIFI	1*WIFI，USB接口
	ADC	1*12位ADC接口，引出4通道
	TF卡	1*TF卡接口
	GPIO	1*40 PIN GPIO（可复用5路UART/2路I2C/3路SPI/4路CAN/4路PWM）
	RTC	1*RTC接口
	WTD	1*WTD
	JTAG	1*JTAG调试接口
	调试串口	1*UART调试接口
	复位按钮	1*复位按钮
	电源输入	1*TYPE-C接口
	指示灯	1*电源指示灯 1*运行状态指示灯
系统软件	固件	uBoot2022.04
	内核	Linux5.10
	系统	Buildroot/Loongnix/SylixOS/OpenHarmony/openWRT
机构与环境	输入电源	TYPE-C 接口，DC 5V/2A输入
	工作温度	0~70℃
	相对湿度	95%，无凝结
	存储温度	-40~85℃
	典型功耗	1.5W
	板卡尺寸	85mm*56mm

1.2 龙芯2K0300--主要特征参数

龙芯中科 LOONGSON TECHNOLOGY

LS2K0300	规格及参数
处理器核	单核LA264，典型主频：1.0GHz 32KB 数据/指令Cache（**ECC校验**） 512KB 共享 Cache（**ECC校验**）
工艺	28nm国产工艺
内存	支持DDR4-**1600**速率，支持**ECC校验**
外设接口	(1).**高速通信接口**：USB2.0*2，GMAC*2； (2).**音视频**：LCD高清显示，I2S音频； (3).**其他接口**：**4*CAN-FD**，8通道**ADC**，2***QSPI**/SPI，2***eMMC**/SDIO，LIO，3*TIMER，4*PWM，4*I2C，10*UART，RTC，106*GPIO、JTAG等； (4).**安全模块**：硬件加解密等。
功耗	工作功耗：<1W，**无散热片应用**场景
封装	CSP286-12mm*12mm，0.65pitch，**合封内存**（下半年）
电源种类	1.0V(CORE)、1.2V(DDR4-IO)、1.8V(PLL)、3.3V(IO)
温度范围	0℃ - 70℃（商业级），-40℃ - 85℃（工业级）

接口丰富

灵活适配

性价比高

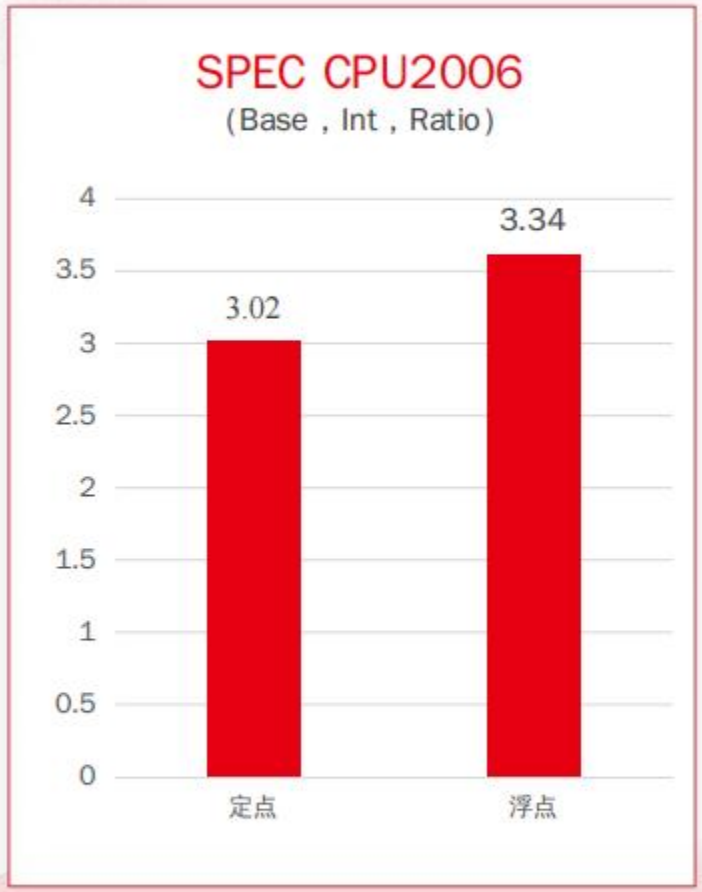
高能低耗

安全可靠

1.3 龙芯2K0300--处理器核

龙芯中科 LOONGSON TECHNOLOGY

LA264@1.0GHz



LoongArch

中国架构

开源开放

完全自主
技术先进
兼容生态

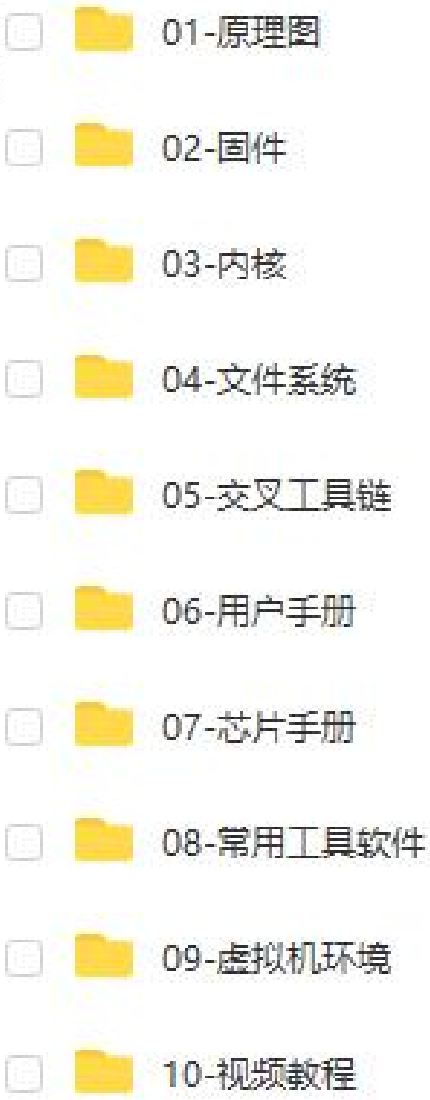
	Cortex-A53	LA264
频率	1GHz@TSMC28nm	1GHz@28nm
Dhrystone(-Ofast)	~3.1 DMIPS/MHz	3.4 DMIPS/MHz
Coremark(-O2)	~3.26 Coremark/MHz	3.73 Coremark/MHz

说明：1. Cortex-A53的Dhrystone和Coremark分值来自实测，编译器采用GCC8.3。
2. LA264的Dhrystone/Coremark分值来自2K0300实测，共享Cache容量为512KB。

LS2K0300中使用的处理器核为LA264，主频1.0GHz

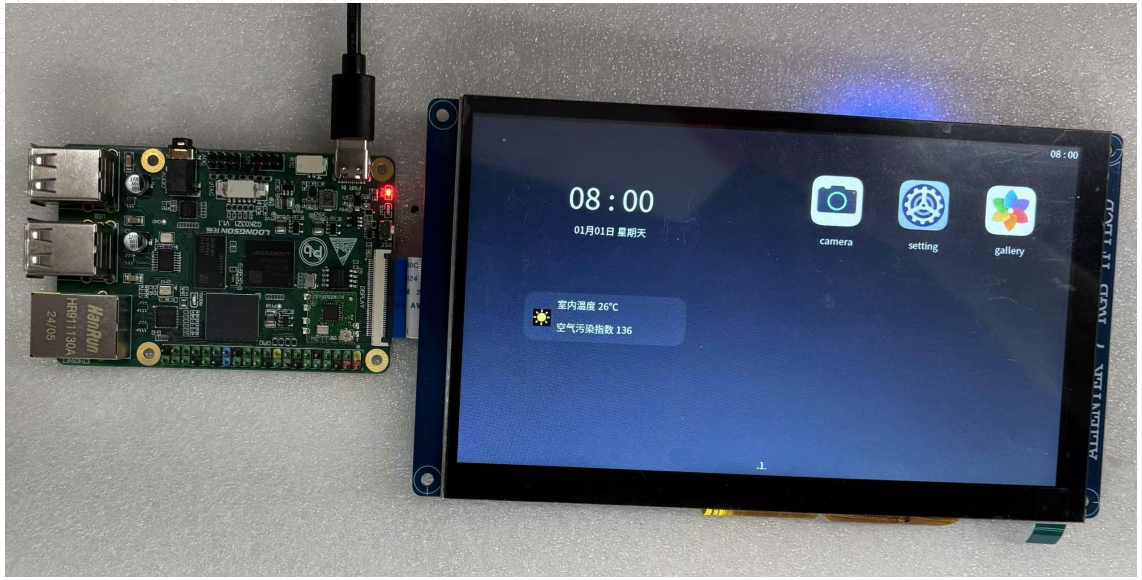
（数据来自第三方测试结果,本次以实测数据为准）

2.1 2K0300先锋派上手使用--BSP包介绍



1. 先锋派BSP（滚动更新）包下载链接：
https://pan.baidu.com/s/111-X7BOmLPOjimjzj4qHeg?pwd=1234
提取码：1234
2. 虚拟机和ubuntu的安装，ubuntu版本建议采用18.04

2.2 2K0300先锋派上手使用--开发板上手



- 2K0300先锋派采用的是uboot+linux+rootfs的模式运行。即uboot作为板卡的固件，使用linux内核，文件系统可以是OpenHarmony、busybox(由buildroot构建而来)、OpenWrt。
- 从上电后的启动顺序来说，先启动固件，固件引导内核启动，内核再引导文件系统启动。最终文件系统启动完毕后，出现终端或UI界面，用户便可以进行操作。
- 如何与板卡交互：
  - Shell终端是和文件系统交互的一个重要手段。2K0300先锋派有调试串口，调试串口从固件开始就生效，会把启动过程中的信息打印出来。等文件系统启动后，可以通用调试串口在shell终端输入命令和系统进行交互。
  - 除了调试串口，也可以使用ssh，或通过LCD屏进行调试，各有特点。比如ssh服务要文件系统启动完成且网络工作正常才可使用。各调试方式的使用方法详见用户手册第四章第一节。

2.3 2K0300先锋派上手使用--更新u-boot,kernerl,rootfs

龙芯中科 LOONGSON TECHNOLOGY

```
*** U-Boot Boot Menu ***

   [1] System boot select
   [2] Update kernel
   [3] Update rootfs
   [4] Update u-boot
   [5] Update ALL
   [6] System install or recover
   [7] Board product
   [8] Video resolution select
   [9] Video rotation select
   [a] U-Boot console

Hit any key to stop autoboot:  9
Press UP/DOWN to move, ENTER to select, ESC/CTRL+C to quit
```

```
*** U-Boot Boot Menu ***

   [1] BOE BP101WX1-206 1280x800 60Hz
   [2] ALIENTEK ATK-MD1010R 1280x800 60Hz
   [3] ALIENTEK ATK-MD0430R 800x480 60Hz
   [4] ALIENTEK ATK-MD0700R 1024x600 60Hz
   [5] use board default panel
   [6] Return


Press UP/DOWN to move, ENTER to select
```

1. 板卡启动之后串口就会有信息输出,如果在启动的时候，一直按住字母‘m’键，就会进入uboot的菜单，如下图所示，分别有对应的烧录选项。如果在启动的时候，一直按住‘c’，就会进入uboot的操作终端。

2. 更新支持U盘和网络更新两种方式，要更新的内核（文件名为：uImage）、文件系统（文件名为：rootfs.img）、固件（文件名为：u-boot-with-spl.bin 或 u-boot.bin）。

3. 通过U盘或TF卡更新，U盘或TF卡需格式化为FAT32，并在U盘目录下创建update文件夹，并将要更新的文件放到此目录。

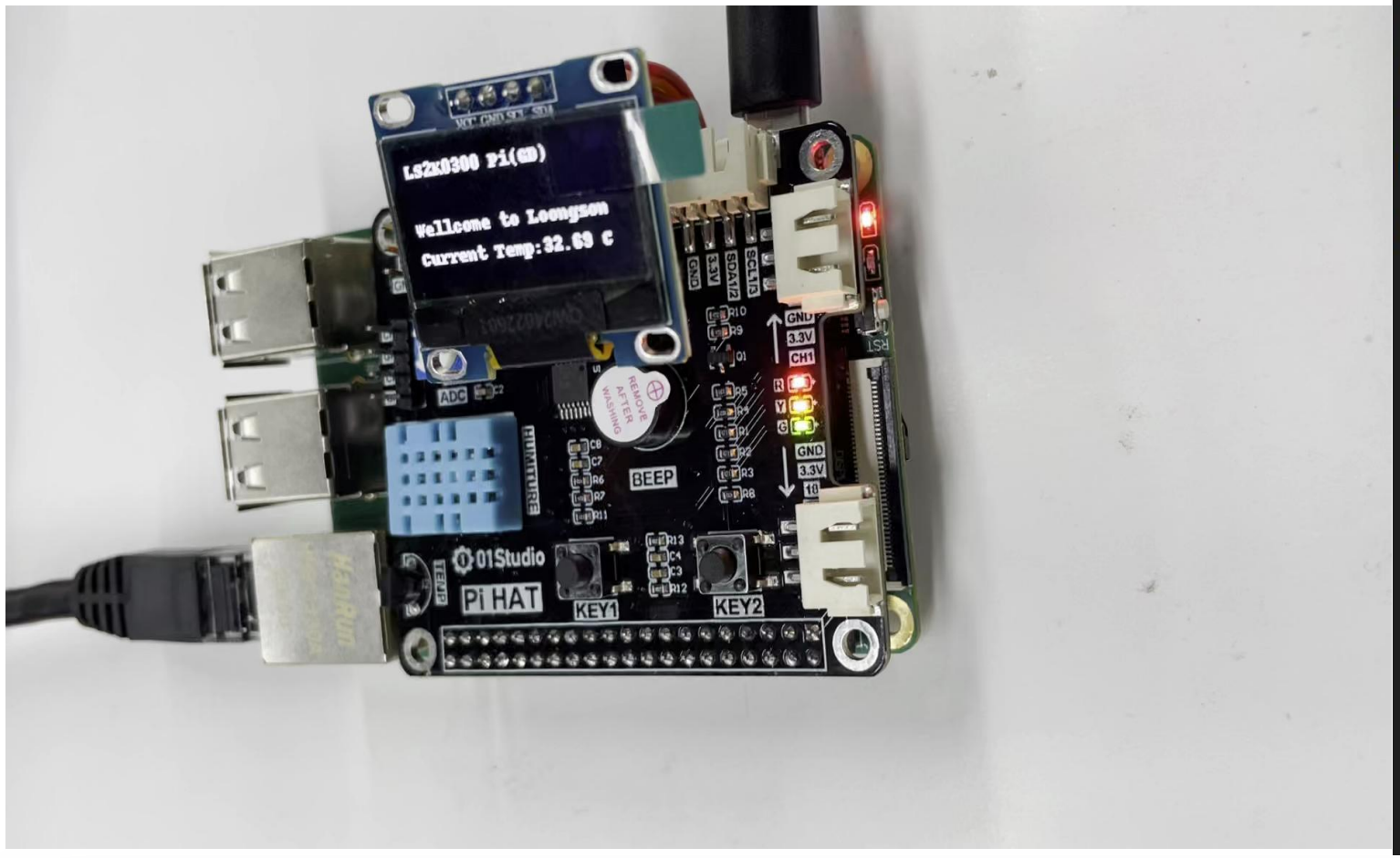
4. 通过网络更新，将要更新的文件放到tftp服务的根目录

5. 支持不同尺寸多种分辨率的LCD屏，屏幕分辨率支持通过U-Boot菜单进行切换

6. 详见用户手册第四章第三节

2.4 2K0300先锋派上手使用--python应用

已适配RPI.GPIO、luma.core、luma.oled、w1thermsensor等树莓派常用python包。




```python
import RPi.GPIO as GPIO
from w1thermsensor import W1ThermSensor

#导入luma相关库,oled lib
from luma.core.render import canvas
from luma.oled.device import ssd1306
from time import sleep

#初始化oled, I2C接口1,oled地址是0x3c
device = ssd1306(port=1, address=0x3C)

################DS18B20对象构建#################################
#构建方法2: 无需知道传感器地址，只有1个传感器连接时适用
DS18B20=W1ThermSensor()

value_1 = GPIO.HIGH
while True:
    temperature = DS18B20.get_temperature() #数据采集

    #oled显示温度信息,结果保留2位小数
    with canvas(device) as draw:
        draw.text((0, 0),  'LS2K0300 Pi(GD)', fill="white")
        draw.text((0, 30), 'Wellcome to Loongson', fill="white")
        draw.text((0, 45), 'Current Temp:%.2f'%temperature+' C', fill="white")

    GPIO.setup(88, GPIO.OUT)
    GPIO.setup(72, GPIO.OUT)
    GPIO.setup(73, GPIO.OUT)
    print("{0}".format(value_1))
    GPIO.output(72, value_1)
    GPIO.output(73, value_1)
    GPIO.output(88, value_1)

    if value_1 == GPIO.HIGH:
        value_1 = GPIO.LOW
    else:
        value_1 = GPIO.HIGH

    sleep(1) #采集间隔1秒
```

3.1 2K0300先锋派OpenHarmony介绍

龙芯中科 LOONGSON TECHNOLOGY

表1 基础类型系统简介

类型	处理器	最小内存	能力
轻量系统（mini system）	MCU类处理器（例如Arm Cortex-M、RISC-V 32位的设备）	128KiB	提供多种轻量级网络协议，轻量级的图形框架，以及丰富的IOT总线读写部件等。可支撑的产品如智能家居领域的连接类模组、传感器设备、穿戴类设备等。
小型系统（small system）	应用处理器（例如Arm Cortex-A的设备）	1MiB	提供更高的安全能力、标准的图形框架、视频编解码的多媒体能力。可支撑的产品如智能家居领域的IP Camera、电子猫眼、路由器以及智慧出行域的行车记录仪等。
标准系统（standard system）	应用处理器（例如Arm Cortex-A的设备）	128MiB	提供增强的交互能力、3D GPU以及硬件合成能力、更多控件以及动效更丰富的图形能力、完整的应用框架。可支撑的产品如高端的冰箱显示屏。

表1 系统关系对应表

系统级别	轻量系统	小型系统	标准系统
LiteOS-M	√	×	×
LiteOS-A	×	√	√
Linux	×	√	√

2K0300先锋派目前适配的是小型系统，采用Linux内核，采用LiteOS-M内核的轻量系统在适配完善中。

3.2 2K0300先锋派OpenHarmony 源码编译

```
[OHOS INFO] ----------------------------------------------
[OHOS INFO] ccache summary:
[OHOS INFO] ccache version: 3.4.1
[OHOS INFO] cache hit (direct): 0
[OHOS INFO] cache hit (preprocessed): 0
[OHOS INFO] cache miss: 0
[OHOS INFO] hit rate: 0.00%
[OHOS INFO] miss rate: 0.00%
[OHOS INFO] Cache size (GB):
[OHOS INFO] ----------------------------------------------
[OHOS INFO] c targets overlap rate statistics
[OHOS INFO] subsystem       files NO. percentage builds NO. percentage overlap rate
[OHOS INFO] ability            80 1.8%       80 1.8% 1.00
[OHOS INFO] applications       30 0.7%       30 0.7% 1.00
[OHOS INFO] arkui            1020 23.4%     1020 23.4% 1.00
[OHOS INFO] bundlemanager      38 0.9%       38 0.9% 1.00
[OHOS INFO] commonlibrary       9 0.2%        9 0.2% 1.00
[OHOS INFO] communication     331 7.6%      331 7.6% 1.00
[OHOS INFO] developtools        9 0.2%        9 0.2% 1.00
[OHOS INFO] distributeddatamgr  2 0.0%        2 0.0% 1.00
[OHOS INFO] distributedhardware 54 1.2%       54 1.2% 1.00
[OHOS INFO] global             17 0.4%       17 0.4% 1.00
[OHOS INFO] graphic            42 1.0%       42 1.0% 1.00
[OHOS INFO] hdf                88 2.0%       88 2.0% 1.00
[OHOS INFO] hiviewdfx          12 0.3%       12 0.3% 1.00
[OHOS INFO] multimedia         21 0.5%       21 0.5% 1.00
[OHOS INFO] powermgr           25 0.6%       25 0.6% 1.00
[OHOS INFO] security          334 7.7%      334 7.7% 1.00
[OHOS INFO] sensors             5 0.1%        5 0.1% 1.00
[OHOS INFO] startup           131 3.0%      131 3.0% 1.00
[OHOS INFO] systemabilitymgr   25 0.6%       25 0.6% 1.00
[OHOS INFO] test               92 2.1%       92 2.1% 1.00
[OHOS INFO] thirdparty       1947 44.7%    1947 44.7% 1.00
[OHOS INFO] updater             4 0.1%        4 0.1% 1.00
[OHOS INFO] window             21 0.5%       21 0.5% 1.00
[OHOS INFO] xts                92 2.1%       92 2.1% 1.00
[OHOS INFO]
[OHOS INFO] c overall build overlap rate: 1.00
[OHOS INFO]
[OHOS INFO]
[OHOS INFO] ls2k300_vanguard_pi_linux build success
[OHOS INFO] Cost time:  0:15:30
=====build  successful=====
```

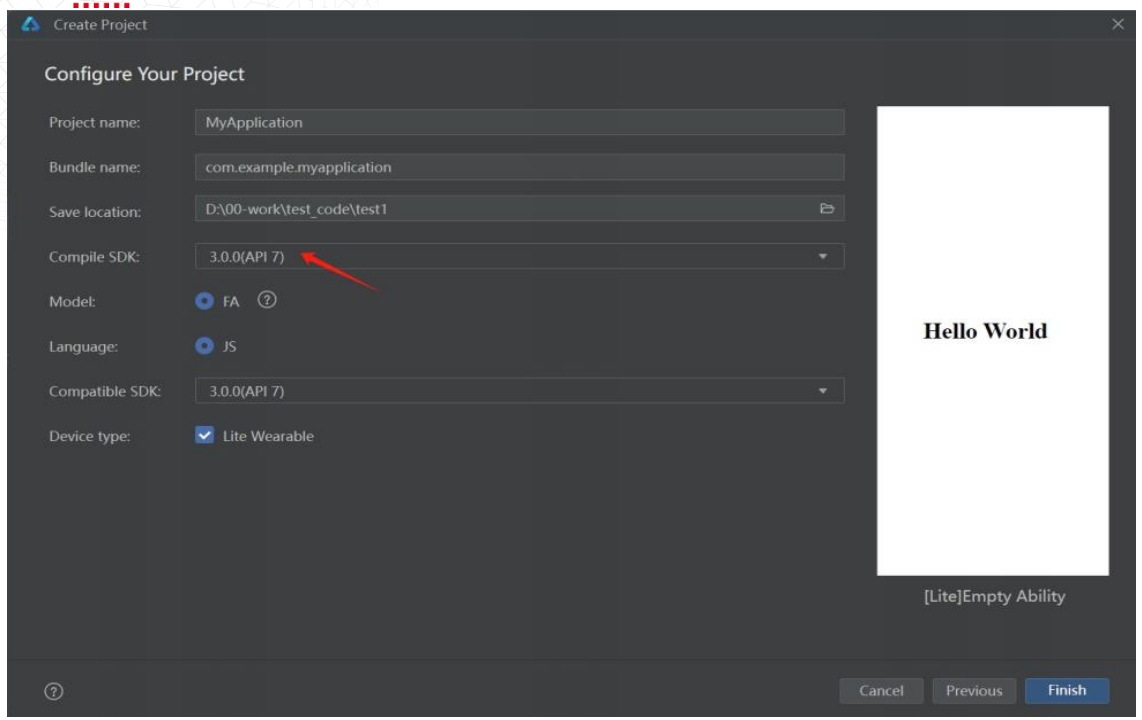
编译步骤：

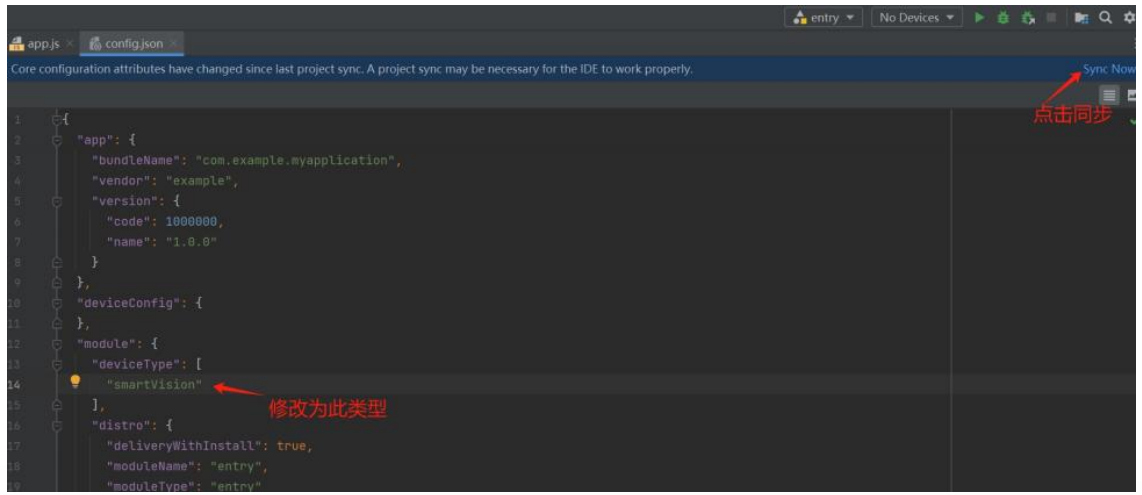
1. 拉取源码 （gitee.com:open-loongarch/manifest.git 分支为ohos-v4.1）
2. 打补丁 （配置龙芯工具链，龙芯平台的代码修改）
3. 选定产品编制 （ls2k300_vanguard_pi）
4. 编译 （./build.sh --product-name ls2k300_vanguard_pi_linux --ccache --no-prebuilt-sdk）
5. 烧录验证

参考链接：

https://mp.weixin.qq.com/s/8XMh599aEzogNN8Ax715Cg

3.3 2K0300先锋派OpenHarmony--应用开发(JS)





1. 下载 DevEco Studio 并安装
3.1.1(https://developer.huawei.com/consumer/cn/deveco-studio/archive/)

2. 小型系统不支持ArkTS 开发，但所有系统都支持JS。目前只有 API8 及以下才支持JS， API7 及以下支持 smartVision 这种设备类型。

3. JS 开发选择 lite Empty Ability 模板 和 API 7

4. 创建完工程之后，先修改 deviceType 为 smartVision ，再同步"Sync Now"工程。

5. 代码编写完之后编译生成hap 包。

6. 将hap 包放到板卡上之后，先修改hap包的权限（chmod 777 xxx.hap），然后执行bm 进行安装。

3.4 2K0300先锋派OpenHarmony--应用开发(C/C++)

```
vm@vm:~/oh/openharmony_l1/applications/sample/loongson/hello$ tree
.
├── BUILD.gn
└── hello_world.c
```

```
vm@vm:~/oh/openharmony_l1/vendor/loongson/ls2k300_vanguard_pi_linux$ git diff
diff --git a/ls2k300_vanguard_pi_linux/config.json b/ls2k300_vanguard_pi_linux/config.json
index a63565c..4cbc590 100644
--- a/ls2k300_vanguard_pi_linux/config.json
+++ b/ls2k300_vanguard_pi_linux/config.json
@@ -134,7 +134,8 @@
       "subsystem": "applications",
       "components": [
         { "component": "camera_sample_app", "features":[] },
-        { "component": "camera_screensaver_app", "features":[] }
+        { "component": "camera_screensaver_app", "features":[] },
+        { "component": "my_hello_app", "features":[] }
       ]
     },
```

```
vm@vm:~/oh/openharmony_l1/out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux$ find . -name hello_world
./rootfs/bin/hello_world
./unstripped/bin/hello_world
./bin/hello_world
vm@vm:~/oh/openharmony_l1/out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux$
```

```
# hello_world
***********************************************
        Hello LoongArch!
***********************************************
```

```
vm@vm:~/oh/openharmony_l1/build/lite/components$ git diff communication.json
diff --git a/lite/components/communication.json b/lite/components/communication.json
index efbbed6f..58d9e0e9 100644
--- a/lite/components/communication.json
+++ b/lite/components/communication.json
@@ -1,5 +1,26 @@
 {
   "components": [
+    {
+      "component": "my_hello_app",
+      "description": "hello world",
+      "optional": "true",
+      "dirs": [
+        "applications/sample/loongson/hello"
+      ],
+      "targets": [
+        "//applications/sample/loongson/hello:hello"
+      ],
+      "rom": "",
+      "ram": "",
+      "output": [ ],
+      "adapted_kernel": [ "linux" ],
+      "features": [],
+      "deps": {
+        "components": [
+        ],
+        "third_party": []
+      }
+    },
```

1. 在applications/sample下创建应用目录，该目录下存放c文件和编译文件BUILD.gn
2. 编辑build/lite/components/communication.json 添加支持
3. 编辑vendor/loongson/ls2k300_vanguard_pi_linux/config.json，将其编译进系统
4. 编译成功后在out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux下(./bin 和 ./rootfs/bin/)能找到编译好的二进制文件。
5. 运行烧录系统或将编译后的二进制文件放到板卡里面，直接运行即可。

3.4 2K0300先锋派OpenHarmony--应用开发(C/C++)

龙芯中科 LOONGSON TECHNOLOGY

BUILD.gn | hello_world.c | communication.json M | config.json

applications > sample > loongson > hello > BUILD.gn

```
import("//build/lite/config/component/lite_component.gni")

executable("hello_exe") {
    output_name = "hello_world"
    sources = [ "hello_world.c" ]
    include_dirs = []
    defines = []
    cflags_c = []
    ldflags = []
}

lite_component("hello") {
    features = [
        ":hello_exe",
    ]
}
```

BUILD.gn | communication.json M | config.json M | hello_world.c

vendor > loongson > ls2k300_vanguard_pi_linux > config.json > [ ] subsystems

```
"subsystems": [
  {
  },
  {
    "subsystem": "applications",
    "components": [
      { "component": "camera_sample_app", "features":[] },
      { "component": "camera_screensaver_app", "features":[] },
      { "component": "loongson_helloworld_app", "features":[] },
      { "component": "my_hello_app", "features":[] }
    ]
  },
  {
```

build > lite > components > communication.json > [ ] components > {} 0

```
{
  "components": [
    {
      "component": "my_hello_app",
      "description": "hello world",
      "optional": "true",
      "dirs": [
        "applications/sample/loongson/hello"
      ],
      "targets": [
        "//applications/sample/loongson/hello:hello"
      ],
      "rom": "",
      "ram": "",
      "output": [ ],
      "adapted_kernel": [ "linux" ],
      "features": [],
      "deps": {
        "components": [
        ],
        "third_party": []
      }
    },
```

```
vm@vm:~/oh/openharmony_l1/out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux$ find . -name hello_world
./rootfs/bin/hello_world
./unstripped/bin/hello_world
./bin/hello_world
vm@vm:~/oh/openharmony_l1/out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux$
```

3.5 2K0300先锋派OpenHarmony--应用开发(C/C++)（HAP）

```
vm@vm:~/oh/openharmony_l1/applications/sample/loongson/helloworld$ tree
.
├── BUILD.gn
├── bundle.json
├── cert
│   └── com.sample.helloworld.invalid_signature.p7b
├── README.md
└── src
    ├── config.json
    ├── main
    │   ├── event_listener.h
    │   ├── main_ability.cpp
    │   ├── main_ability.h
    │   ├── main_ability_slice.cpp
    │   └── main_ability_slice.h
    └── resources
        └── Icon_HelloWorld.png
```

```
vm@vm:~/oh/openharmony_l1/vendor/loongson/ls2k300_vanguard_pi_linux$ git diff config.json
diff --git a/ls2k300_vanguard_pi_linux/config.json b/ls2k300_vanguard_pi_linux/config.json
index a63565c..ebf1b63 100644
--- a/ls2k300_vanguard_pi_linux/config.json
+++ b/ls2k300_vanguard_pi_linux/config.json
@@ -134,7 +134,9 @@
       "subsystem": "applications",
       "components": [
         { "component": "camera_sample_app", "features":[] },
-        { "component": "camera_screensaver_app", "features":[] }
+        { "component": "camera_screensaver_app", "features":[] },
+        { "component": "loongson_helloworld_app", "features":[] },
+        { "component": "my_hello_app", "features":[] }
       ]
```

```
vm@vm:~/oh/openharmony_l1/out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux$ find . -name helloworld.hap
./rootfs/system/internal/helloworld.hap
./system/internal/helloworld.hap
```

1. 在applications/sample下创建应用目录（可以基于cameraApp为样本）
2. 编辑 vendor/loongson/ls2k300_vanguard_pi_linux/config.json，将其编译进系统
3. 编译成功后在 out/ls2k300_vanguard_pi/ls2k300_vanguard_pi_linux下(./rootfs/system/internal)能找到编译好的hap包
4. 安装

```
 # 关闭系统的签名验证功能，否则无法安装未签名的hap
bm set -s disable
# 安装未签名的hap
bm install -p helloworld.hap
# 如果要卸载hap，执行如下命令后，重启开发板即可
bm uninstall -n com.sample.helloworld
```

3.5 2K0300先锋派OpenHarmony--应用开发(C/C++)(HAP)

龙芯中科 LOONGSON TECHNOLOGY

BUILD.gn | config.json M | communication.json M | hello_world.c | app_module.h

applications > sample > loongson > helloworld > BUILD.gn

```
import("//build/lite/config/hap_pack.gni")

shared_library("helloworld") {
  sources = [
    "src/main/main_ability.cpp",
    "src/main/main_ability_slice.cpp"
  ]

  deps = [
    "${aafwk_lite_path}/frameworks/ability_lite:aafwk_abilitykit_lite",
    "${appexecfwk_lite_path}/frameworks/bundle_lite:bundle",
    "//foundation/arkui/ui_lite:ui_lite",
    "//foundation/graphic/graphic_utils_lite:utils_lite",
    "//foundation/graphic/surface_lite",
    "//foundation/systemabilitymgr/samgr_lite/samgr:samgr",
  ]

  include_dirs = [
    "${aafwk_lite_path}/interfaces/kits/ability_lite",
    "${aafwk_lite_path}/interfaces/kits/want_lite",
    "${appexecfwk_lite_path}/interfaces/kits/bundle_lite",
  ]

  ldflags = [ "-shared" ]
  ldflags += [ "-lstdc++" ]
  ldflags += [ "-L$ohos_root_path/sysroot/usr/lib" ]
  ldflags += [ "-Wl,-rpath-link=$ohos_root_path/sysroot/usr/lib" ]
  ldflags += [
    "-lui",
    "-lability",
  ]

  defines = [
    "ENABLE_WINDOW=1",
    "ABILITY_WINDOW_SUPPORT",
    "OHOS_APPEXECFWK_BMS_BUNDLEMANAGER",
  ]

}

hap_pack("helloworld_hap") {
  deps = [ ":helloworld" ]
  mode = "hap"
  json_path = "src/config.json"
  ability_so_path = "$root_out_dir/libhelloworld.so"
  force = "true"
  # NOTE: invalid_signature, so auto-pack hw.hap will fail, we need manually pack hw.hap
  # when this cert_profile is valid, hw.hap will auto-packed and installed
  cert_profile = "cert/com.sample.helloworld.invalid_signature.p7b"
  resources_path = "src/resources"
  hap_name = "helloworld"
}
```

bundle.json | config.json M | communication.json M | hello_world.c

applications > sample > loongson > helloworld > bundle.json > component > adapted_system_type

```
{
    "name": "@ohos/loongson_helloworld_app",
    "description": "helloworld related samples for small system.",
    "version": "3.1",
    "license": "Apache License 2.0",
    "publishAs": "code-segment",
    "segment": {
        "destPath": "applications/sample/loongson/helloworld"
    },
    "dirs": {},
    "scripts": {},
    "component": {
        "name": "loongson_helloworld_app",
        "subsystem": "applications",
        "syscap": [],
        "features": [],
        "adapted_system_type": [
            "mini",
            "small"
        ],
        "rom": "",
        "ram": "",
        "deps": {
            "components": [
                "ability_lite",
                "bundle_framework_lite",
                "surface",
                "ui",
                "graphic_utils",
                "kv_store",
                "syspara_lite",
```

bundle.json | config.json M | communication.json M | hello_world.c

vendor > loongson > ls2k300_vanguard_pi_linux > config.json > subsystems > 15 > components

```
    "subsystems": [
      {
      },
      {
        "subsystem": "applications",
        "components": [
          { "component": "camera_sample_app", "features":[] },
          { "component": "camera_screensaver_app", "features":[] },
          { "component": "loongson_helloworld_app", "features":[] },
          { "component": "my_hello_app", "features":[] }
        ]
      },
```

3.6 2K0300先锋派OpenHarmony--应用开发(JS/C++ 混合)

龙芯中科 LOONGSON TECHNOLOGY

```
vm@vm:~/oh/openharmony_l1/foundation/arkui/ace_engine_lite/frameworks/src/core/modules$ git diff app
_module.h
diff --git a/frameworks/src/core/modules/app_module.h b/frameworks/src/core/modules/app_module.h
index 8eef43e..012f50f 100644
--- a/frameworks/src/core/modules/app_module.h
+++ b/frameworks/src/core/modules/app_module.h
@@ -30,6 +30,7 @@ public:
     ~AppModule() = default;
     static JSIValue GetInfo(const JSIValue thisVal, const JSIValue *args, uint8_t argsNum);
     static JSIValue Terminate(const JSIValue thisVal, const JSIValue *args, uint8_t argsNum);
+    static JSIValue ToggleLed(const JSIValue thisVal, const JSIValue* args, uint8_t argsNum);
 #if (FEATURE_SCREEN_ON_VISIBLE == 1)
     static JSIValue ScreenOnVisible(const JSIValue thisVal, const JSIValue *args, uint8_t argsNum);
 #endif
@@ -61,6 +62,7 @@ void InitAppModule(JSIValue exports)
 {
     JSI::SetModuleAPI(exports, "getInfo", AppModule::GetInfo);
     JSI::SetModuleAPI(exports, "terminate", AppModule::Terminate);
+    JSI::SetModuleAPI(exports, "ledcontrol", AppModule::ToggleLed);
 #if (FEATURE_SCREEN_ON_VISIBLE == 1)
     JSI::SetModuleAPI(exports, "screenOnVisible", AppModule::ScreenOnVisible);
 #endif
```

```
diff --git a/frameworks/src/core/modules/app_module.cpp b/frameworks/src/core/modules/app_module.cpp
index 410632e..e9bbe0a 100644
--- a/frameworks/src/core/modules/app_module.cpp
+++ b/frameworks/src/core/modules/app_module.cpp
@@ -45,6 +45,60 @@ struct AsyncParams : public MemoryHeap {
 };
 #endif

+static int OnDevEventReceived(void *priv, uint32_t id, struct HdfSBuf *data)
+{
+    uint32_t value;
+    HdfSbufReadUint32(data, &value);
+    HILOG_ERROR(HILOG_MODULE_ACE,"%s: dev event received: %u %u\n", (char *)priv, id, value);
+
+    return HDF_SUCCESS;
+}
+
+JSIValue AppModule::ToggleLed(const JSIValue thisVal, const JSIValue *args, uint8_t argsNum)
+{
+    HILOG_ERROR(HILOG_MODULE_ACE, "led button pressed.");
+
+    struct HdfIoService *serv = HdfIoServiceBind(LED_SERVICE);
+    if (serv == NULL)
+    {
+        HILOG_ERROR(HILOG_MODULE_ACE,"fail to get service2 %s\n", LED_SERVICE);
+        return JSI::CreateUndefined();
+    }
+
+    if ((args == nullptr) || (argsNum == 0) || (JSI::ValueIsUndefined(args[0]))) {
+        return JSI::CreateUndefined();
+    }
+
+    JSIValue success = JSI::GetNamedProperty(args[0], CB_SUCCESS);
+    JSIValue fail = JSI::GetNamedProperty(args[0], CB_FAIL);
+    JSIValue complete = JSI::GetNamedProperty(args[0], CB_COMPLETE);
+
+    int32_t num = (int32_t)JSI::GetNumberProperty(args[0], "code");
+
```

```
export default class App {
  /**
   * Obtains the declared information in the config.json file of an application.
   */
  static getInfo(): AppResponse;

  /**
   * Destroys the current ability.
   */
  static terminate(): void;
  //自定义接口
  static ledcontrol(options: {
      code: number;
      success?: (res: string) => void;
      fail?: (res: string, code: number) => void;
      complete?: () => void;
  }): void;
}
```

南向

1. foundation/arkui/ace_engine_lite/frameworks/src/core/modules/app_module.h 中添加接口 foundation/arkui/ace_engine_lite/frameworks/src/core/modules/app_module.c 中实现接口

北向

1. 添加接口定义
（调用系统未提供的接口时需要现在系统接口文件中定义一个JS接口），接口的参数定义和南向实现的一致。接口名要和南向导出的接口名一致。
2. 编写页面代码（index.html, index.css）
3. 编写JS代码导入接口，编写业务逻辑

4.1 软件移植--应用软件移植

开源软件源码编译移植（C/C++代码）

1、获取源码：通过github或第三方开源社区获取源码；
2、准备编译环境：安装编译器gcc等；
3、使用开源软件源码中的CMake或AutoConfig脚本生成 Makefile；
由于旧版本的config.guess 不支持loongarch，在执行./configure时会出如下的错误
configure: error: cannot guess build type; you must specify one
解决方法：
方法一. 找到 config.guess 文件编缉它，在ia64:Linux:*：*） 下面添加Loongarch 架构的支持。

```
loongarch*:Linux:*:*)
    echo ${UNAME_MACHINE}-unknown-linux-gnu
    exit ;;
```

方法二. 从网上获取最新的config.guess 和 config.sub 文件然后进行替换

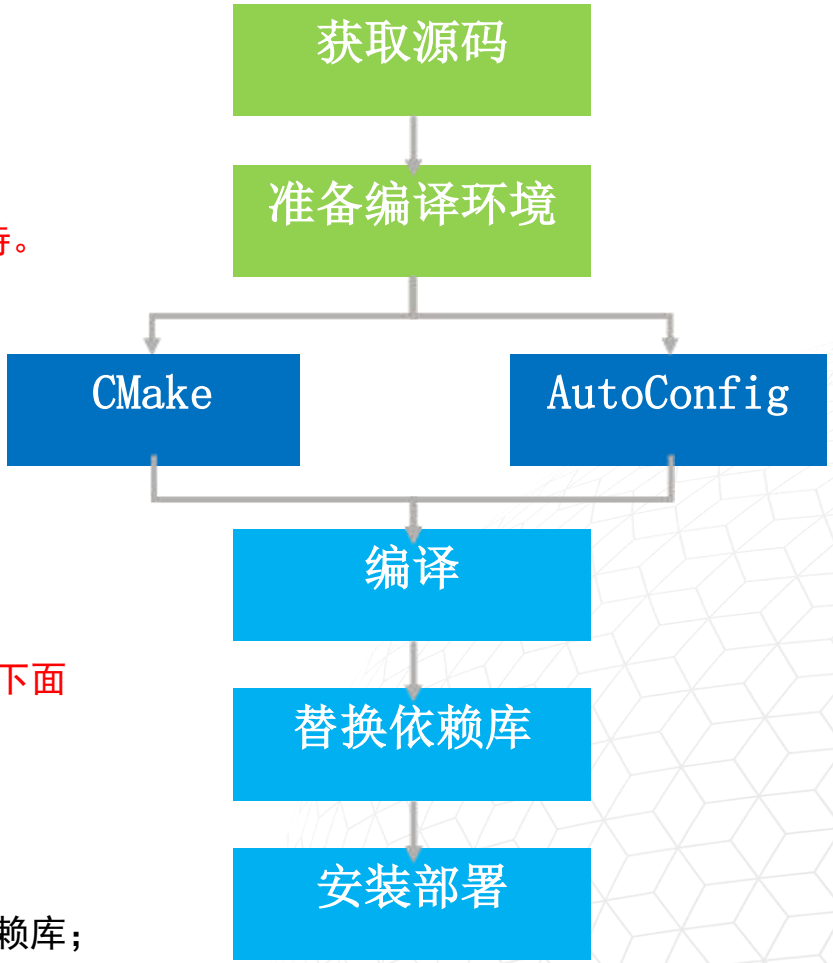
```
$ sudo wget -O /usr/share/misc/config.sub
"git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.sub;hb=HEAD"
$ sudo wget -O /usr/share/misc/config.guess
"git.savannah.gnu.org/gitweb/?p=config.git;a=blob_plain;f=config.guess;hb=HEAD"
```

在源码目录执行下面的命令会将/usr/share/misc/config.sub 和 config.guess 替换掉工程目录下面的config.sub 和 config.guess 文件

```
$ libtoolize -f -i -c
```

通过上面的方法修改之后再运行./configure 就可以了

4、执行Makefile编译可执行程序；
5、替换依赖库：通过开源软件readme文件、编译时的动态库缺失或链接错误，重新编译或替换依赖库；
6、将可执行程序安装部署到生产或测试系统。
7、详见用户手册第5.3.1



龙芯中科 LOONGSON TECHNOLOGY

C/C++代码编译选项、编译宏移植

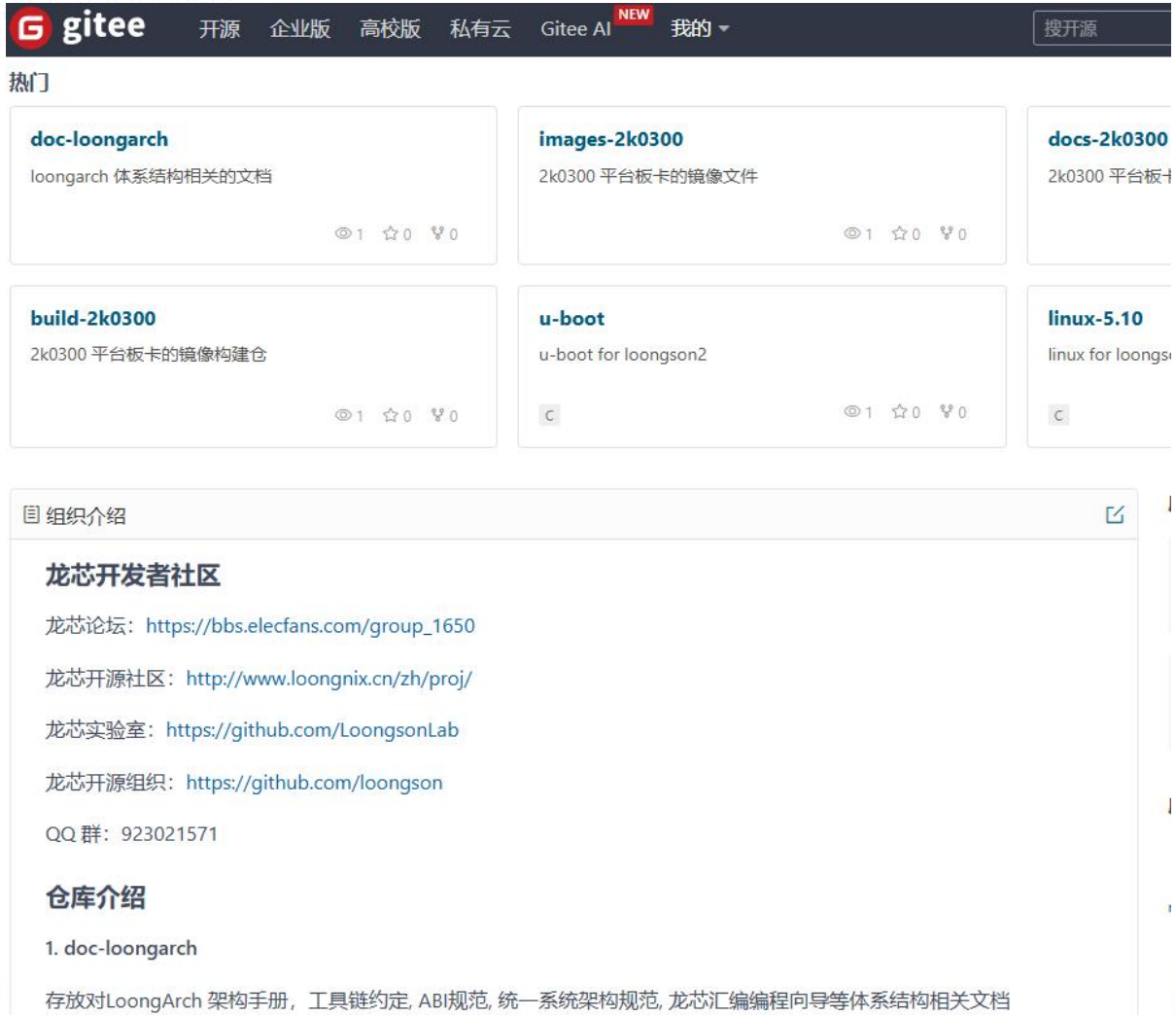
	功能	X86编译选项	LoongArch64编译选项
64位编译	定义编译生成的应用程序为64 位，编译选项不同	-m64	-mabi=lp64d
指令集	Makefile中定义指令集类型,由X86修改为LoongArch	-march=skylake	-march=loongarch64
编译宏	原有x86版编译宏替换为LoongArch宏	__X86_64__	__loongarch__

查看编译器自定义译宏： echo | gcc -E -dM -

4.3 软件移植--相关资源

龙芯中科 LOONGSON TECHNOLOGY

广东龙芯开源代码仓库(https://gitee.com/open-loongarch)



龙芯论坛(https://bbs.elecfans.com/group_1650)

4.4 软件移植--相关资源

龙芯中科 LOONGSON TECHNOLOGY

龙芯开源社区(http://www.loongnix.cn),龙芯基础软件最新成果的发布地。

操作系统

Loongnix操作系统

API与基础软件

龙芯浏览器 | Java | .NET | 音视频技术 | Electron | Node.js | CEF | NW.js |龙芯Python仓库 | LACF应用二进制兼容框架 | Ruby | Miniforge | 龙架构Maven仓库

图形图像

龙芯VBIOS生成工具 | 龙芯LoongGPU

工具链

GNU 编译工具链 | 龙芯LLVM编译器 | 龙芯Golang编译器 | 龙芯Rust编译器

调测工具

ebpf | ftrace | gdb | kdump | kprobe | strace

云计算

KVM | 容器镜像仓库 | Kubernetes | Docker | Alpine

应用工具

vscodium/code-oss | 打印驱动引擎 | 网卡驱动 | GitHub actions/runner

LoongArch软件开发--开发文档

1.计算机体系结构基础(LoongArch)-3rd
https://foxsen.github.io/archbase/bookdown.pdf

2.龙芯架构参考手册 - 卷一：基础架构
https://github.com/loongson/LoongArch-Documentation/releases/download/2023.04.20/LoongArch-Vol1-v1.10-CN.pdf

3.龙芯架构 ELF psABI
https://github.com/loongson/la-abi-specs/releases/download/v2.30/la-abi-v2.30.pdf

4.龙芯架构工具链约定
https://github.com/loongson/la-toolchain-conventions/releases/download/releases%2Fv1.1/la-tc-v1.1.pdf

5.LoongArch汇编编程手册V1.1
https://github.com/loongson/la-asm-manual/releases/download/release-1.1/la-asm-manual-v1.1.pdf

6.LoongArch软件开发约定
https://github.com/loongson/la-softdev-convention/releases/download/v0.1/la-softdev-convention.pdf

7.龙芯架构文档
https://loongson.github.io/LoongArch-Documentation/README-CN.html

自 主 决 定 命 运

创 新 成 就 未 来

龙芯中科

LOONGSON TECHNOLOGY