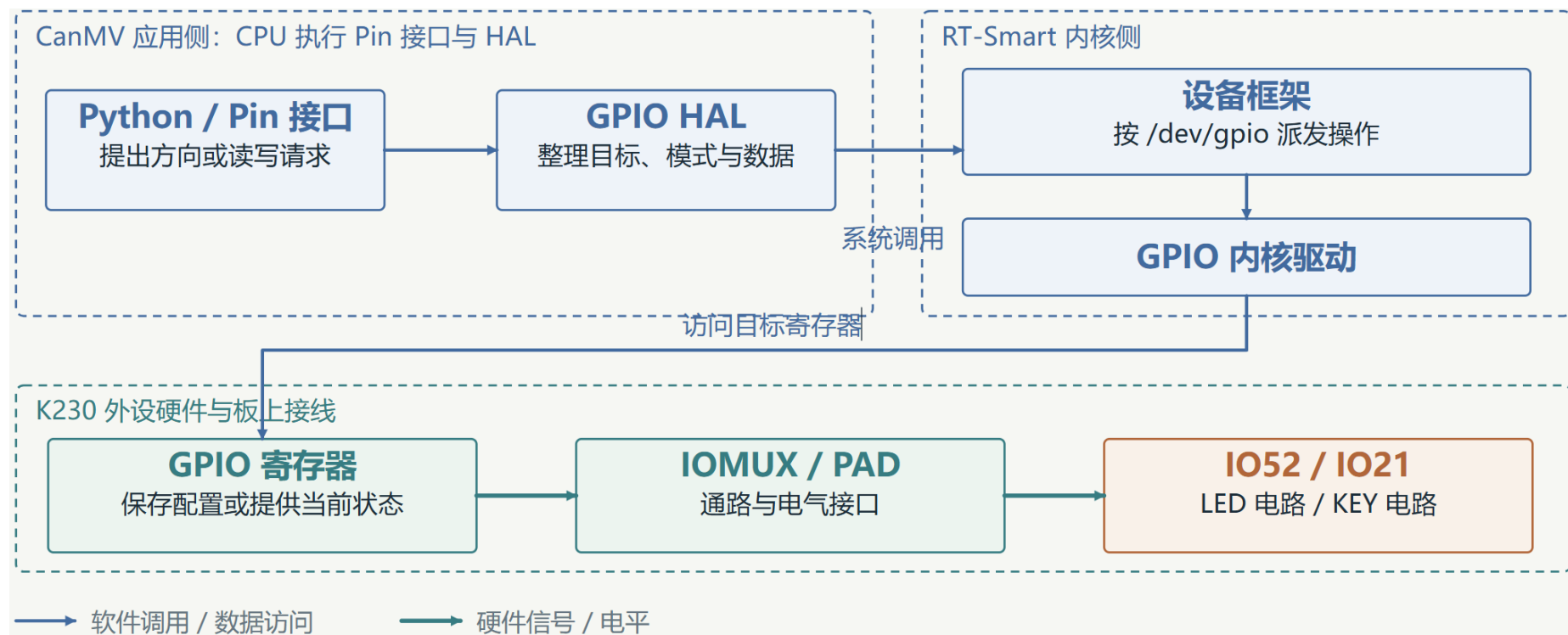


GPIO 原理与 LED、按键实验

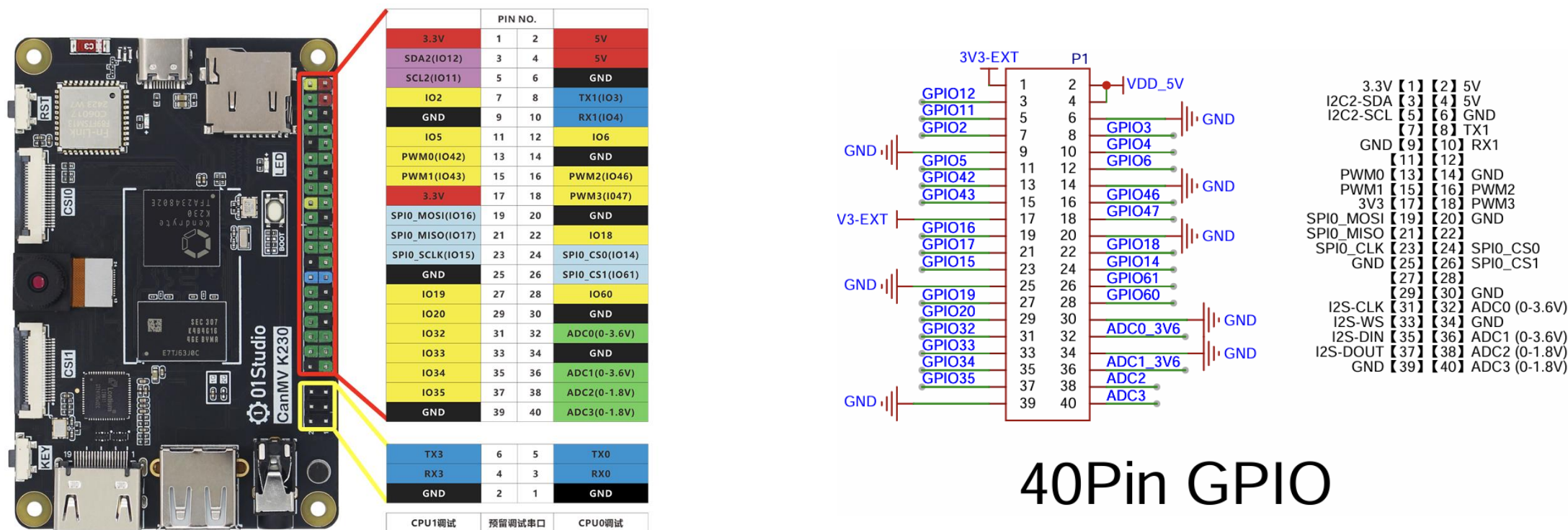
周亦帆

2026.10.13

- GPIO
 - GPIO基础概念
 - 硬件控制通路：GPIO controller、IOMUX
- 硬件抽象层HAL
- LED、key实验实例



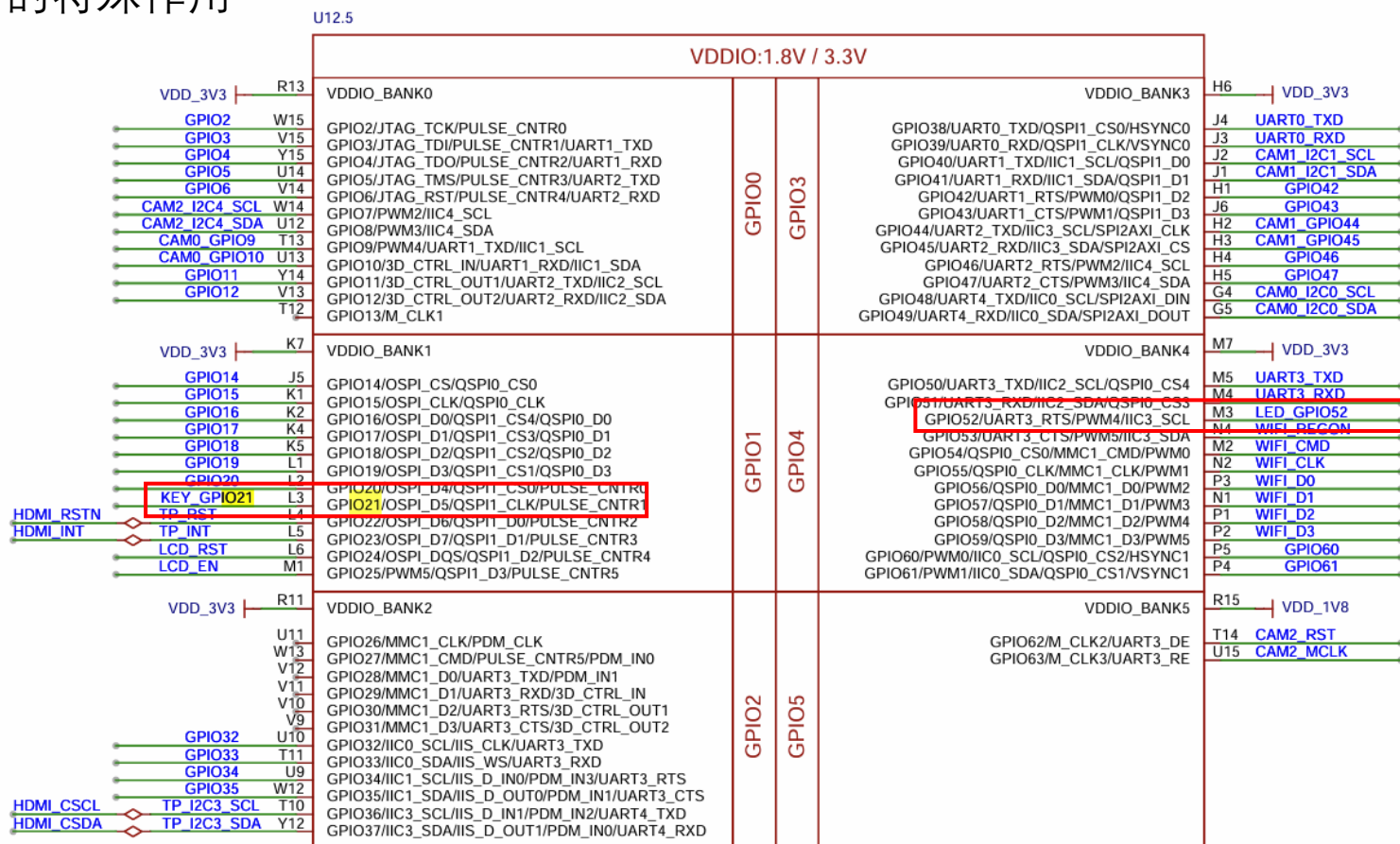
- 和IO的区别：IO是芯片的物理I/O接点，可以对应实际的PAD
- GPIO是指某个IO可以选择通用的输入、输出的功能，由专门的GPIO controller控制
- 输出：软件写入1/0，控制对应的引脚形成高、低电平
- 输入：外部电路改变引脚电压，识别成1/0，由软件读取





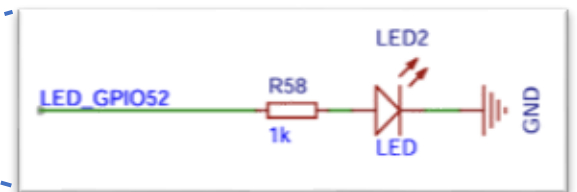
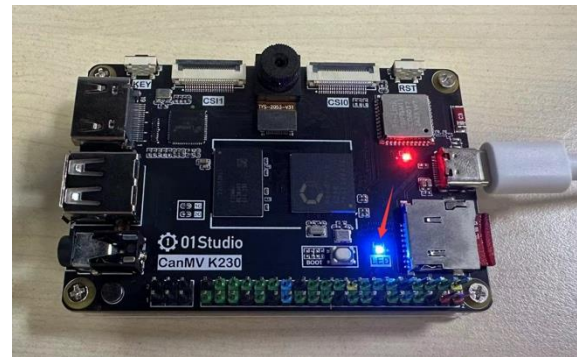
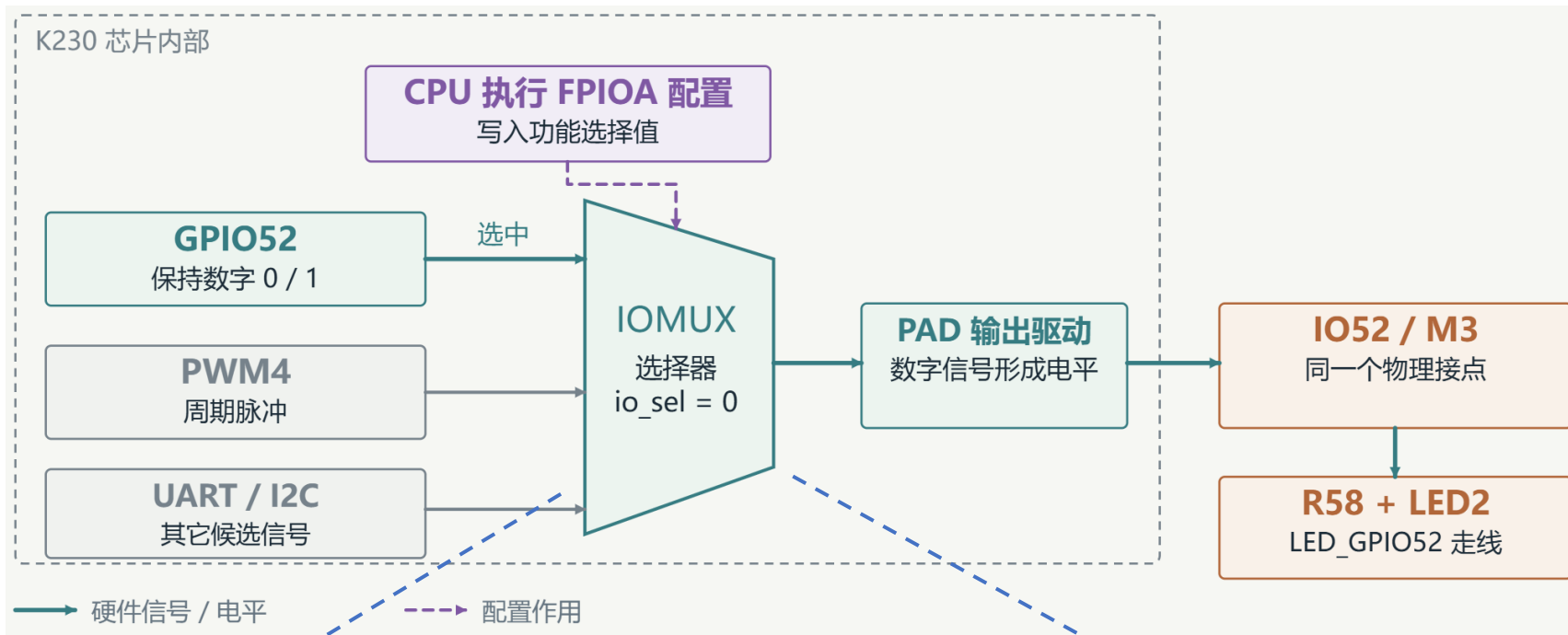
GPIO

- GPIO: General Purpose Input / Output
 - K230常规GPIO有64个，其中24个在40pin当中，
 - 还有40个分布于电路中，有各自的特殊作用
 - LED_GPIO52
 - KEY_GPIO21



□ IOMUX

- 通过软件配置，在若干个候选信号中选中GPIO52信号
- 选中的信号会传输到K230 chip外面影响蓝色的LED2



引脚	alt0	alt1	alt2	alt3	alt4
io52	GPIO52	UART3_RTS	PWM4	I2C3_SCL	TEST_PIN26

□ IOMUX register

- 每一个物理IO都会配备对应的32 bit寄存器
- 物理地址: $0x91105000 + \text{IO-index} * 4$ ($0x91105000 + 52 * 4 = 0x911050D0$)
- $\text{Io_sel}=0x000$: 选中GPIO52

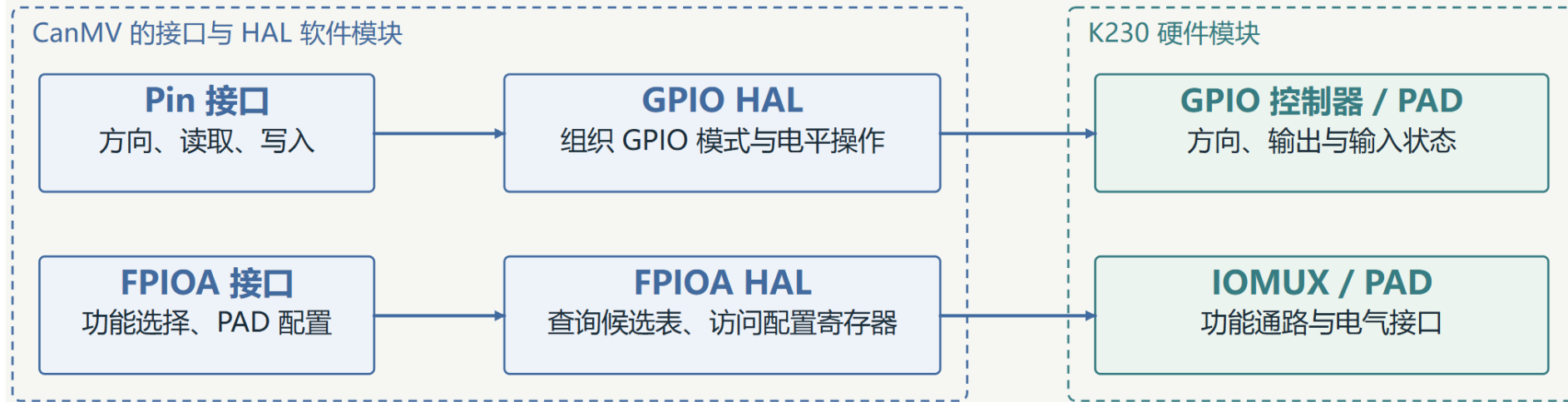
位	字段	类型	描述
31	DI	RO	输入数据
30:14	-	RO	保留
13:11	IO_SEL	RW	功能选择 (000=func1, 依此类推)
10	SL	RW	摆率使能
9	MSC	RW	电压控制 (双电压焊盘)
8	IE	RW	输入使能 (1=启用)
7	OE	RW	输出使能 (1=启用)
6	PU	RW	上拉 (1=启用)
5	PD	RW	下拉 (1=启用)
4:1	DS	RW	驱动强度选择 (0-15)
0	ST	RW	施密特触发器 (1=启用)

- GPIO controller
 - GPIO0 controller: 0 – 31
 - GPIO1 controller: 32 – 63
- 每组GPIO controller主要管理3个32 bit寄存器
 - DR输出数据register: LED实验中, 软件驱动修改GPIO1 bit20=1, 来点亮LED
 - DDR (Data Direction Register): 输入/输出
 - 输入状态register: key被按下时, 硬件会使得GPIO0 bit21=0, 软件通过驱动读取

寄存器	作用	GPIO0~31 的物理地址	GPIO32~63 的物理地址
DR: 输出数据	保存希望输出的 0/1	0x9140B000	0x9140C000
DDR: 方向	每位: 0 输入, 1 输出	0x9140B004	0x9140C004
输入状态, 驱动中为 INPUT[0]	读取引脚电平状态	0x9140B050	0x9140C050



- HAL: hardware abstract layer
 - GPIO controller -> GPIO HAL, 负责配置GPIO模式和读写
 - IOMUX -> FPIOA HAL, 负责IO的功能选择



□ FPIOA HAL

□ 1. 在RT-Smart系统中建立寄存器映射

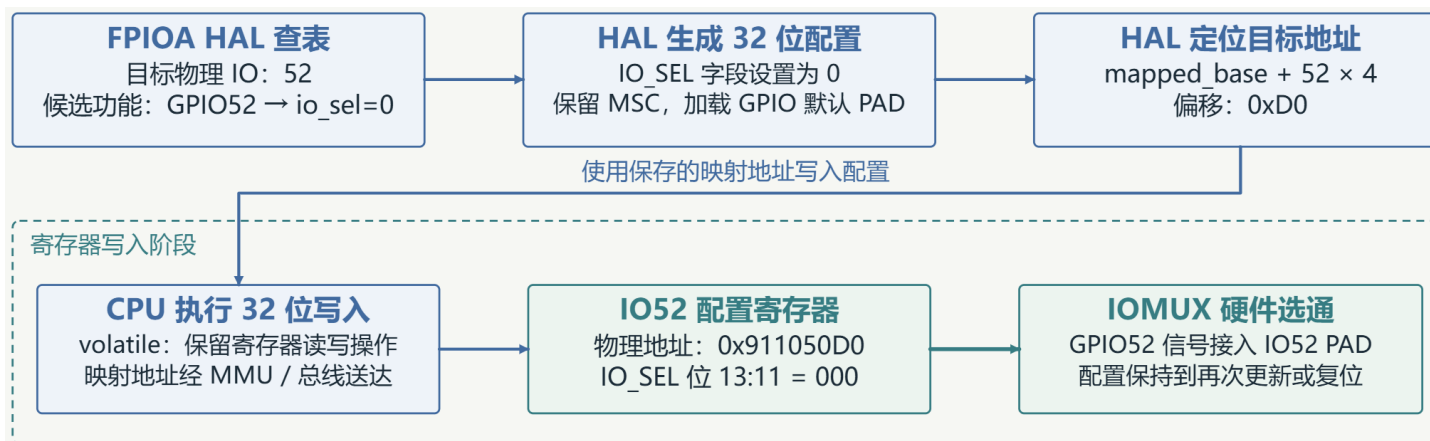
□ `open("/dev/mem")`请求访问物理地址区域

□ RT-Smart 内核返回FPIOA访问句柄fd

□ 获取寄存器base地址0x91105000（对应的虚拟地址），建立映射后FPIOA后续可以定位

□ 2. 查表GPIO52 -> io_sel=0，定位IO52的寄存器地址（偏置0xD0）

□ 3. CPU 执行 32 位写入



```
from machine import Pin #导入Pin模块
from machine import FPIOA
import time
-----

#将GPIO52配置为普通GPIO
fpioa = FPIOA()
fpioa.set_function(52,FPIOA.GPIO52)

LED=Pin(52,Pin.OUT) #构建led对象, GPIO52,输出
LED.value(1) #点亮LED, 也可以使用led.on()

while True:
    pass
```

□ GPIO HAL

- 功能：保存目标与当前配置、组织设备操作参数、取得读回值或操作结果
- Pin模块通过GPIO HAL软件库，调取RT-Smart系统的GPIO 驱动服务，修改GPIO寄存器

□ 1. 在RT-Smart系统中建立设备、寄存器映射

- 定位到GPIO1寄存器位置

□ 2. 修改DDR寄存器和DR寄存器

- 上述过程在mutex互斥锁保护中，是不同GPIO的请求不干扰



```
from machine import Pin #导入Pin模块
from machine import FPIOA
import time
.....

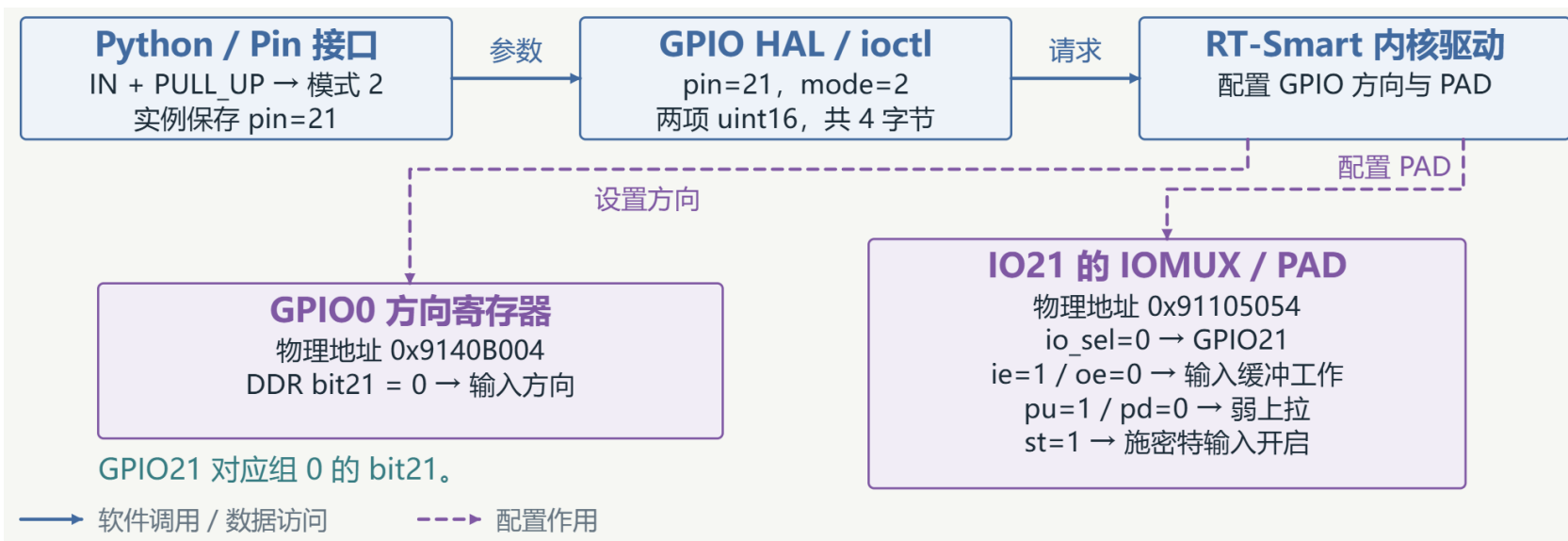
#将GPIO52配置为普通GPIO
fpioa = FPIOA()
fpioa.set_function(52,FPIOA.GPIO52)

LED=Pin(52,Pin.OUT) #构建led对象, GPIO52,输出
LED.value(1) #点亮LED, 也可以使用led.on()

while True:
    pass
```

GPIO HAL

输入过程：IO21电压经过输入缓冲器得到片内数字状态



```
from machine import Pin
from machine import FPIOA
import time

#将GPIO52、GPIO21配置为普通GPIO模式
fpioa = FPIOA()
fpioa.set_function(52,FPIOA.GPIO52)
fpioa.set_function(21,FPIOA.GPIO21)

LED=Pin(52,Pin.OUT) #构建LED对象,开始熄灭
KEY=Pin(21,Pin.IN,Pin.PULL_UP) #构建KEY对象

state=0 #LED引脚状态

while True:

    if KEY.value()==0: #按键被按下
        time.sleep_ms(10) #消除抖动
        if KEY.value()==0: #确认按键被按下

            state=not state #使用not语句而非~语句
            LED.value(state) #LED状态翻转
            print('KEY')

    while not KEY.value(): #检测按键是否松开
        pass
```

输入电压范围	保证的数字结果
$\leq 0.35 \times VDDIO$	逻辑低：0
$\geq 0.65 \times VDDIO$	逻辑高：1

□ K210 与 K230 的 GPIO API

项目	K230 • 本课开发板	K210 • 01Studio 教程板
Python 接口	machine.Pin / FPIOA	maix.GPIO / fpioa_manager.fm
物理 IO 与 GPIO	IO52 配置为 GPIO52	IO35 映射到内部 GPIO0
板载蓝灯有效电平	写 1 点亮	写 0 点亮
K230 点亮蓝灯	<pre>1 from machine import Pin, FPIOA 2 fpioa = FPIOA() 3 fpioa.set_function(52, FPIOA.GPIO52) 4 led = Pin(52, Pin.OUT) 5 led.value(1)</pre>	
K210 点亮蓝灯	<pre>1 from maix import GPIO 2 from fpioa_manager import fm 3 fm.register(35, fm.fpioa.GPIO0) 4 led = GPIO(GPIO.GPIO0, GPIO.OUT) 5 led.value(0)</pre>	

两者均用 value() 读取、value(0/1) 写入。编号、映射与有效电平按芯片 API 和开发板接线选择。