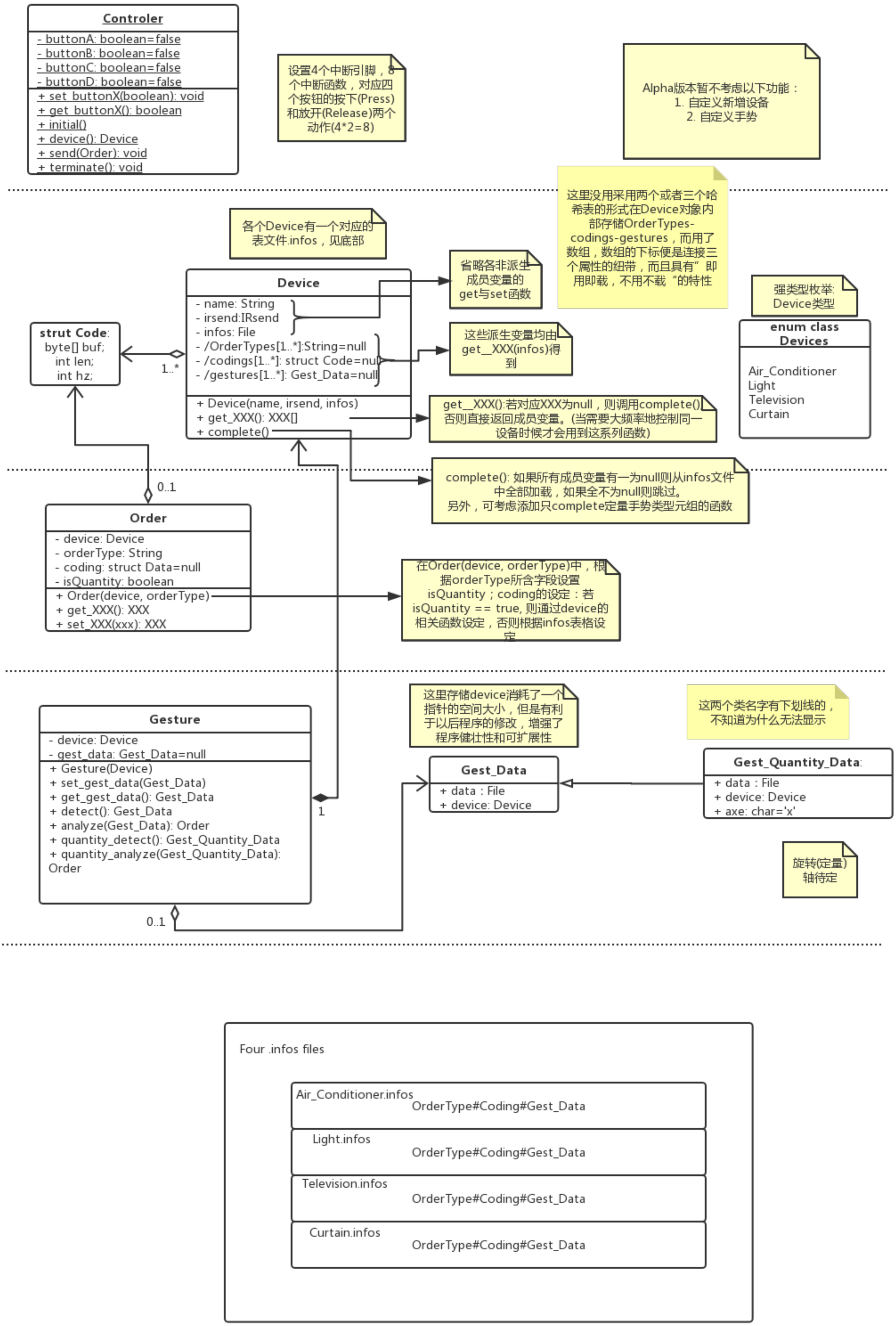
Controler
- buttonA: boolean=false
- buttonB: boolean=false
- buttonC: boolean=false
- buttonD: boolean=false
+ set_buttonX(boolean): void
+ get_buttonX(): boolean
+ initial()
+ device(): Device
+ send(Order): void
+ terminate(): void
设置4个中断引脚，8个中断函数，对应四个按钮的按下(Press)和放开(Release)两个动作(4*2=8)
Alpha版本暂不考虑以下功能：
1. 自定义新增设备
2. 自定义手势
这里没用采用两个或者三个哈希表的形式在Device对象内部存储OrderTypes-codings-gestures，而用了数组，数组的下标便是连接三个属性的纽带，而且具有"即用即载，不用不载"的特性
各个Device有一个对应的表文件.infos，见底部
Device
- name: String
- irsend:IRsend
- infos: File
- /OrderTypes[1..*]:String=null
- /codings[1..*]: struct Code=null
- /gestures[1..*]: Gest_Data=null
+ Device(name, irsend, infos)
+ get_XXX(): XXX[]
+ complete()
省略各非派生成员变量的get与set函数
这些派生变量均由get__XXX(infos)得到
get__XXX():若对应XXX为null，则调用complete()否则直接返回成员变量。(当需要大频率地控制同一设备时候才会用到这系列函数)
强类型枚举: Device类型
enum class Devices
Air_Conditioner
Light
Television
Curtain
strut Code:
byte[] buf;
int len;
int hz;
1..*
0..1
complete(): 如果所有成员变量有一为null则从infos文件中全部加载，如果全不为null则跳过。
另外，可考虑添加只complete定量手势类型元组的函数
Order
- device: Device
- orderType: String
- coding: struct Data=null
- isQuantity: boolean
+ Order(device, orderType)
+ get_XXX(): XXX
+ set_XXX(xxx): XXX
在Order(device, orderType)中，根据orderType所含字段设置isQuantity；coding的设定：若isQuantity == true, 则通过device的相关函数设定，否则根据infos表格设定
这里存储device消耗了一个指针的空间大小，但是有利于以后程序的修改，增强了程序健壮性和可扩展性
这两个类名字有下划线的，不知道为什么无法显示
Gesture
- device: Device
- gest_data: Gest_Data=null
+ Gesture(Device)
+ set_gest_data(Gest_Data)
+ get_gest_data(): Gest_Data
+ detect(): Gest_Data
+ analyze(Gest_Data): Order
+ quantity_detect(): Gest_Quantity_Data
+ quantity_analyze(Gest_Quantity_Data): Order
1
0..1
Gest_Data
+ data : File
+ device: Device
Gest_Quantity_Data:
+ data : File
+ device: Device
+ axe: char='x'
旋转(定量)轴待定
Four .infos files
Air_Conditioner.infos
OrderType#Coding#Gest_Data
Light.infos
OrderType#Coding#Gest_Data
Television.infos
OrderType#Coding#Gest_Data
Curtain.infos
OrderType#Coding#Gest_Data