嵌入式C语言之-

函数指针和回调函数

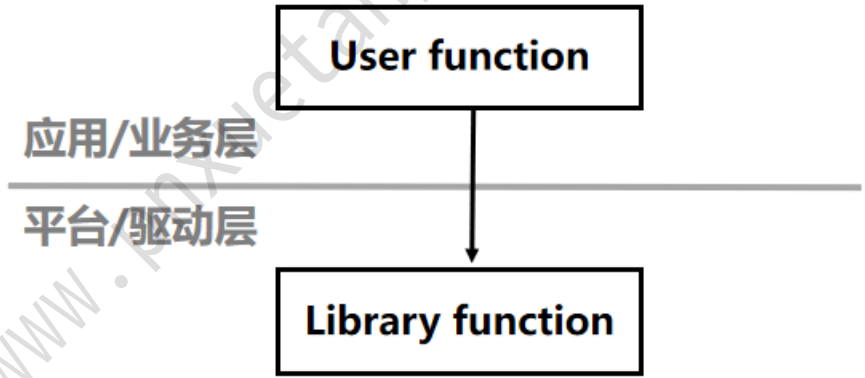
讲师：叶大鹏

助力你成为优秀的电子工程师！

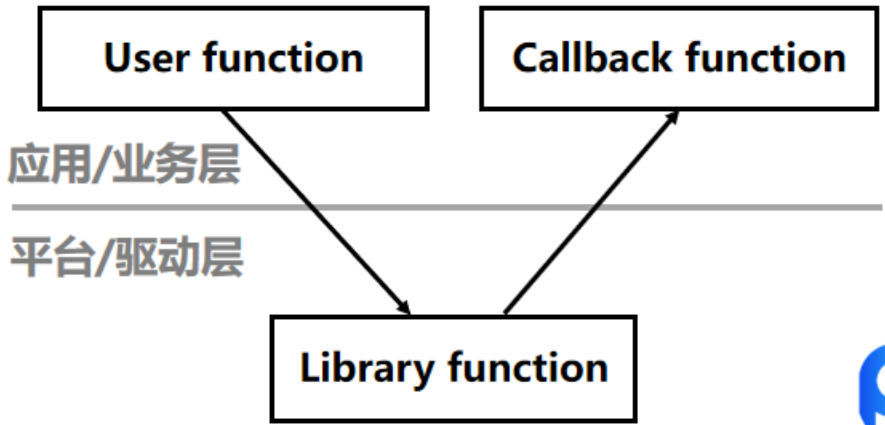
PN学堂

应用案例

- 我们编写的上层应用代码毫无疑问会调用下层的库代码，包括标准库、单片机或者GUI的库，像printf()、malloc()等等，通常调用流程是：



- 在某些场景，调用流程要反过来，比如使用下层库里的定时功能，每隔1S，即时通知上层的业务去刷新时钟的UI界面，如何通知？很多库代码是闭源的，我们不可能在库中添加业务代码函数，也就没办法直接调用，但是C语言提供了函数指针和回调函数机制通过实现下层通知上层的功能。

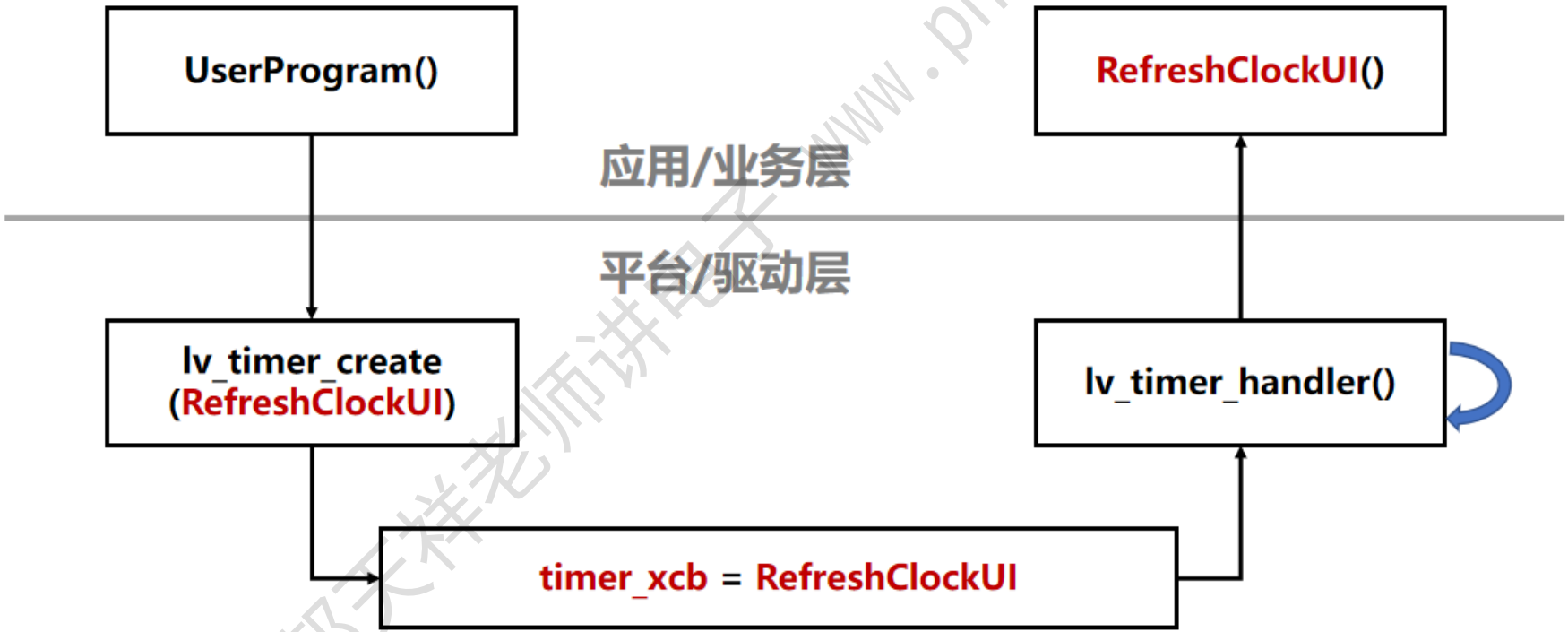


应用案例

```
void UserProgram(void)
{
    lv_timer_t * timer = lv_timer_create(RefreshClockUI, 1000, NULL);
}
```

```
void RefreshClockUI(lv_timer_t *timer)
{
    time_t rawtime;
    struct tm *info;
    time(&rawtime);
    info = localtime(&rawtime);
    lv_label_set_text_fmt(label, "%02d:%02d:%02d",
        info->tm_hour, info->tm_min, info->tm_sec);
}
```

```
void UserProgram(void)
{
    lv_timer_t * timer = lv_timer_create(RefreshClockUI, 1000, NULL);
}
```



回调函数和函数指针

```
void UserProgram(void)
{
    lv_timer_t * timer = lv_timer_create(RefreshClockUI, 1000, NULL);
}
```

- void RefreshClockUI(lv_timer_t *timer)，称为回调函数，回调函数本身也是普通函数，只是因为调用关系比较特别，它的代码位于上层业务层，却是由下层库代码去调用，所以叫做回调函数；

```
lv_timer_t * lv_timer_create(lv_timer_cb_t timer_xcb, uint32_t period, void * user_data)
```

- lv_timer_cb_t timer_xcb，timer_xcb称为函数指针，它用来保存回调函数的地址，严谨一些，应该称为函数指针类型的变量/函数指针变量；

函数指针变量

- 格式为：

  函数返回值类型 (* 函数指针变量名) (函数参数列表);

- int32_t (*pSum)(int32_t a, int32_t b);

  函数指针变量pSum，就像int32_t *ptr里的ptr一样；

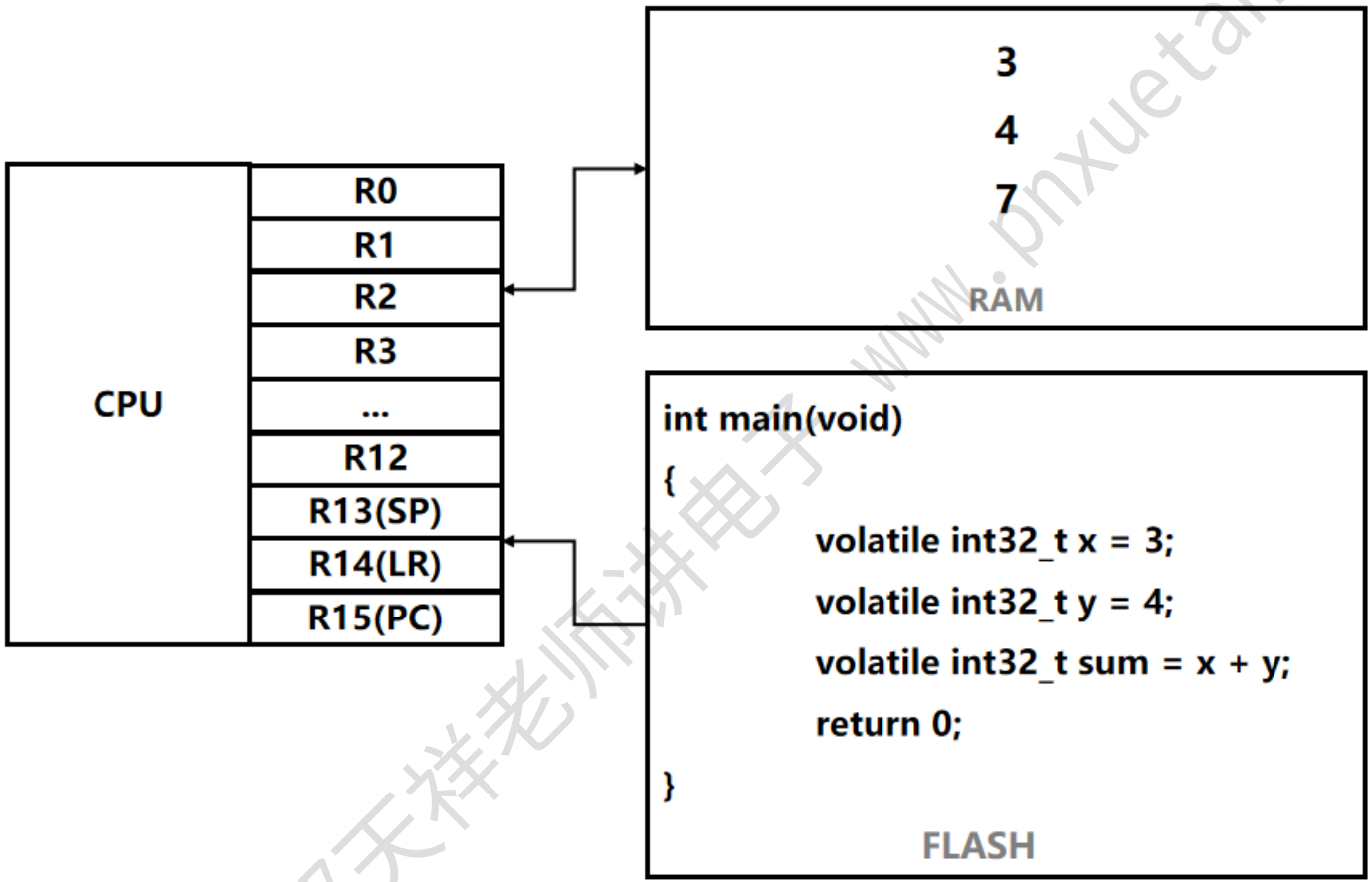
- 函数名称就像数组名称一样保存了函数地址：

```
pSum = 0x0000070f, Sum = 0x0000070f
```

- (*pSum)(1, 2)，表示间接访问并调用Sum函数。

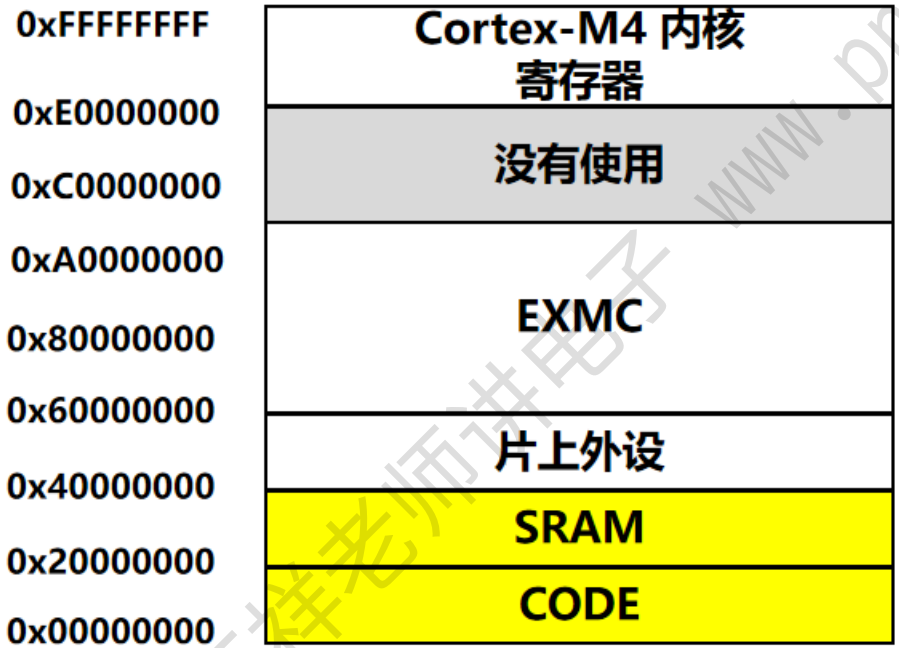
```
int32_t Sum(int32_t x, int32_t y)
{
    return x + y;
}
int main(void)
{
    int32_t (*pSum)(int32_t a, int32_t b);
    pSum = Sum;
    printf("pSum = 0x%p, Sum = 0x%p\n",
            pSum, Sum);
    int32_t sum = (*pSum)(1, 2);
    printf("%d\n", sum);
    return 0;
}
```

单片机程序运行时，代码指令还是保存在FLASH中



单片机寻址范围

- 单片机通过地址来访问FLASH、内存和寄存器，ARM寻址范围4GB，分为多个块，FLASH对应地址范围是0x00000000-0x20000000。



注：基于GD32F303单片机

函数指针和指针函数

- int32_t (*pSum)(int32_t a, int32_t b)；为什么(*pSum)要使用()?

- 如果不使用()，变成了int32_t *pSum(int32_t a, int32_t b)；基于运算符优先级，pSum先结合()再结合*，这种格式被称为指针函数，表示返回值为指针类型的函数，比如常见的：

  void *malloc(size_t size)

  char *strcpy(char *dest, const char *src)

- 使用()，基于运算符优先级，pSum先结合*再结合后面的()，这种格式用来定义函数指针变量，变量是pSum。

函数指针

```
int32_t Sum(int32_t x, int32_t y)
{
    return x + y;
}
int main(void)
{
    int32_t (*pSum)(int32_t a, int32_t b);
    pSum = Sum;
    int32_t sum = (*pSum)(1, 2);
    printf("%d\n", sum);
    return 0;
}
```

```
int32_t Sum(int32_t x, int32_t y)
{
    return x + y;
}
void Handle(int32_t (*pSum)(int32_t a, int32_t b))
{
    int32_t sum = (*pSum)(1, 2);
    printf("%d\n", sum);
}
int main(void)
{
    Handle(Sum);
    return 0;
}
```

函数指针类型和函数指针变量

```
void Handle(int32_t (*pSum)(int32_t a, int32_t b))
```

- 如果程序中很多地方都需要定义这种函数指针类型的变量，书写起来太繁琐，可以使用typedef重定义：

```
typedef int32_t (*PFUNC)(int32_t a, int32_t b);

int32_t Sum(int32_t x, int32_t y)
{
    return x + y;
}
void Handle(PFUNC pSum)
{
    int32_t sum = (*pSum)(1, 2);
    printf("%d\n", sum);
}
int main(void)
{
    Handle(Sum);
    return 0;
}
```

函数指针类型和函数指针变量

```
typedef int32_t (*PFUNC)(int32_t a, int32_t b);
```

- 为什么函数指针类型的变量PFUNC还可以作为数据类型?

1. 这里typedef，和常规用法不太一样：

   typedef signed char int8_t;

2. typedef int32_t (*PFUNC)(int32_t a, int32_t b);这里并不是将int32_t重定义为(*PFUNC)(int32_t a, int32_t b)，

函数指针类型和函数指针变量

```
typedef int32_t (*PFUNC)(int32_t a, int32_t b);
```

3. 可以这样理解：原有数据类型是int32_t (*)(int32_t a, int32_t b)，即参数为int32_t a, int32_t b，返回值为int32_t的函数指针类型，重定义的数据类型是PFUNC，就像：

```
typedef    int8_t *    PINT8;
PINT8  ptr;
```

4. 当有typedef时，PFUNC表示函数指针类型，PFUNC pSum；当没有typedef时，int32_t (*pSum)(int32_t a, int32_t b)，pSum表示变量，pSum = Sum。

结构体指针变量访问成员

```
TempHumiSensor tempHumiData;
TempHumiSensor  *tempHumiPtr;
tempHumiPtr = &tempHumiData;
```

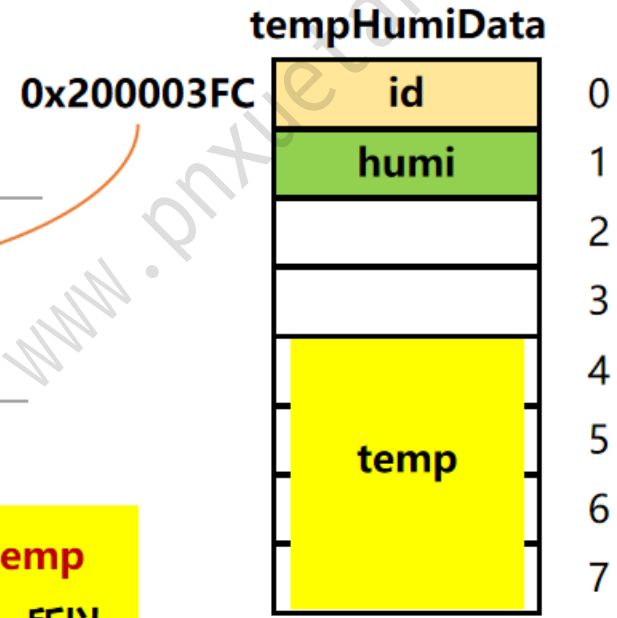
tempHumiPtr | 0x200003FC

0x200003F8~3FB

```
tempHumiPtr->temp = 20.5f;
```

tempHumiPtr->temp 等价于 (*tempHumiPtr).temp

表示间接访问tempHumiData成员temp的地址空间，所以&tempHumiPtr->temp 和 &tempHumiData.temp都可以获取temp的内存地址



PN学堂

位运算优先级

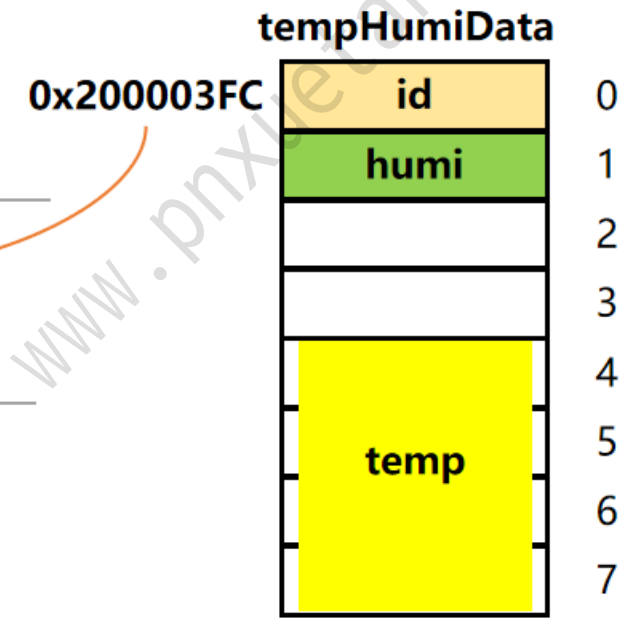
运算符（优先级从上往下）	运算符说明及应用场景	结合性
() [] -> .	括号（函数等），数组，结构体指针变量的成员访问，普通结构体变量的成员访问	由左向右
! ~ ++ -- + -	逻辑非，按位取反，自增1，自减1，正号，负号	由右向左
* & (类型) sizeof	指针，取地址，强制类型转换，求占用空间大小	
* / %	乘，除，取模	由左向右
+ -	加，减	由左向右
<< >>	左移，右移	由左向右

结构体指针变量访问成员，获取成员地址

```
TempHumiSensor tempHumiData;
TempHumiSensor  *tempHumiPtr;
tempHumiPtr = &tempHumiData;
```

tempHumiPtr | 0x200003FC

0x200003F8~3FB



tempHumiPtr-> 等价于 (*tempHumiPtr)．而*tempHumiPtr表示间接访问tempHumiData，所以tempHumiPtr->temp也就等价于tempHumiData.temp

指针类参数实质上就是要传递一个地址

THANK YOU!