说明：本文档是介绍LPCXpresso 自带库文件的各个变量和函数所用，并不全面，只是针对需要用到的时钟，串口，GPIO功能，但是可以类推其他变量函数。

**一：寄存器访问（寄存器的地址是内存地址）**

对于**每个寄存器**，都有一**寄存器组**，这组寄存器，库文件**定义了一个结构体**，每个寄存器四个字节，然后把该组**寄存器的基地址转换为**这个**结构体的指针**。利用该宏定义的指针，就可以直接**用指针的方法**来设置**该寄存器组**的每个寄存器。

**二：寄存器的配置**

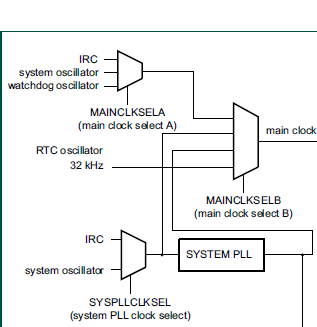
对于每个寄存器，都会有对应的宏定义去配置，只要把**对应的宏定义解释看清**楚，就知道配置的意义是什么。

**三：各种函数**

**1：获取时钟频率函数**

**Chip\_Clock\_GetMainClockRate()**

   该函数是用来获取主时钟源的频率的。 先介绍 LPC1549的时钟源如下图

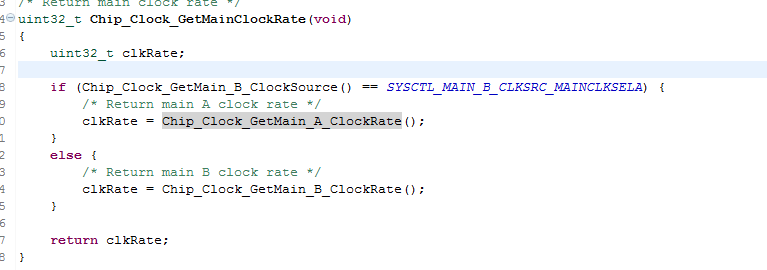


LPC1549 的主时钟源首先由 MAINCLKSELB 来选择来源，可以四种时钟源，而针对不同的时钟源，例如 main clock select A，从该时钟源，还要进一步去确定 MAINCLKSELA如何选择的时钟源，才能最终确定时钟源的频率，其他的二级选择都需这样确定，比如 system PLL clock select。

该函数确定主时钟频率的方法也是如此，先确定 MAINCLKSELB 寄存器的选择（寄存器不同值代表不同来源，这个有对应 enum 类型值确定）

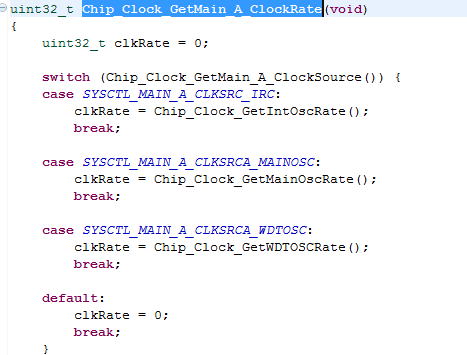
分别获取不同时钟源的频率

确定MAINCLKSELB选择的时钟源,进而作进一步判断



比如

不同类型时钟源，不同的数值



然后根据不同来源再去确定是否需要继续判断寄存器，需要则继续，不需要则直接返回对应的数值，比如 RTC 就是 32000。

**2：STATIC INLINE void Chip\_SWM\_MovablePortPinAssign(CHIP\_SWM\_PIN\_MOVABLE\_T movable, uint8\_t port, uint8\_t pin)**

该函数是用来**绑定可移动的数字功能**到特定的I/O，其中 movable 变量为要绑定的功能，是CHIP\_SWM\_PIN\_MOVABLE\_T 这种 enum 的一种枚举值， port 为要绑定的端口号，pin 为该端口的第几号引脚。

**3：UART相关函数**

Chip\_Clock\_SetUARTBaseClockRate(Chip\_Clock\_GetMainClockRate(), false);

Chip\_UART\_Init(DEBUG\_UART);

Chip\_UART\_ConfigData(DEBUG\_UART, UART\_CFG\_DATALEN\_8 | UART\_CFG\_PARITY\_NONE | UART\_CFG\_STOPLEN\_1);

Chip\_UART\_SetBaud(DEBUG\_UART, 115200);

Chip\_UART\_Enable(DEBUG\_UART);

Chip\_UART\_TXEnable(DEBUG\_UART);

第一个函数用来设置串口的时钟频率。这组函数都是用来设置一个串口的相关属性，DEBUG\_UART 其实是 UART0 ，后面的各个设置配置都是用来设置串口的属性的。

**4：GPIO相关函数（在 gpio\_15xx.h中声明的最基本的几个静态函数）**

所有的这些静态函数分为两组：一组为设置**单个的 io 口**（单个pin）属性和状态的函数。

另外一组是设置**一组io端口（port）**的属性和状态函数。

**A:设置/查询 IO 属性相关函数**

1）**单个 pin 设置io为输出的相函数**

STATIC INLINE **void** **Chip\_GPIO\_SetPinDIROutput**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint8\_t pin)

{

pGPIO->DIR[port] |= 1UL << pin;

}

该函数用来设置**IO口单个引脚**的属性，为**输出属性**。所有与单个引脚的操作都是直接调用该函数**port是要设置的端口**，**pin 为该引脚**。

2）**单个 pin 设置io为输入的相函数**

STATIC INLINE **void** **Chip\_GPIO\_SetPinDIRInput**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint8\_t pin)

{

pGPIO->DIR[port] &= ~(1UL << pin);

}

**3）设置整个io端口某些引脚为输出的函数**

STATIC INLINE **void** **Chip\_GPIO\_SetPortDIROutput**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint32\_t pinMask)

{

pGPIO->DIR[port] |= pinMask;

}

通过**该函数可以**设置一**整个端口**的**所有引脚**的属性，通过 pinMask 参数，如果该参数是 1 ，则**相应的pin** 就被**设置成为输出**。为 0 的话维持原状。

4）**设置整个io端口某些引脚为输入的函数**

STATIC INLINE **void** **Chip\_GPIO\_SetPortDIRInput**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint32\_t pinMask)

{

pGPIO->DIR[port] &= ~pinMask;

}

作用类似上面的 Output 函数。

5）查询IO 口属性的函数就是把上述系列函数的 Set 修改成 Get 。

**B:设置/读取IO 口的数值**

1. **设置单个 pin 的引脚输出为高电平**

STATIC INLINE **void** **Chip\_GPIO\_SetPinOutHigh**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint8\_t pin){

pGPIO->SET[port] = (1 << pin);

}

**设置端口 port 的 pin 脚为高电平。**

1. 设置**单个 pin 输出为低电平**

STATIC INLINE **void** **Chip\_GPIO\_SetPinOutLow**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint8\_t pin)

{

pGPIO->CLR[port] = (1 << pin);

}

1. **一次性**设置**整个端口的某些引脚**的**电平为高**

STATIC INLINE **void** **Chip\_GPIO\_SetPortOutHigh**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint32\_t pins)

{

pGPIO->SET[port] = pins;

}

对**端口 port** 的所有引脚**一次性利用 pin** 来设置其为高电平。pin **值为 1** 的**那些位**都会被**设置成高电平**。

1. **设置整个端口某些输出引脚为低电平，一次性设置**

STATIC INLINE **void** **Chip\_GPIO\_SetPortOutLow**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint32\_t pins)

{

pGPIO->CLR[port] = pins;

}

1. **设置单个输出 pin 反转其状态。由高到底，由低到高的函数**

STATIC INLINE **void** **Chip\_GPIO\_SetPinToggle**(LPC\_GPIO\_T \*pGPIO, uint8\_t port, uint8\_t pin)

{

pGPIO->NOT[port] = (1 << pin);

}

设置 port 口的 pin 脚 反转其状态。

1. **设置整个 port 的某些pin 一起反转**

STATIC INLINE uint32\_t **Chip\_GPIO\_SetPortToggle**(LPC\_GPIO\_T \*pGPIO, uint8\_t portNum)

{

**return** pGPIO->PIN[portNum];

}

设置端口的。

1. **读取的**函数其实就只**需要把**上面的函数的set 修改成 read 或者 get。即可，可以读取单个 pin 的也可以读取整个端口的。但是读取时需要注意有个 mask 寄存的影响。

**Chip\_GPIO\_GETPinState(LPC\_GPIO\_T \*pGPIO, uint8\_t port, unint8\_t pin)**

**Chip\_GPIO\_ReadPort**

**Chip\_GPIO\_GETPortState(LPC\_GPIO\_T \*pGPIO, uint8\_t port)**

**Chip\_GPIO\_ReadPortBit(LPC\_GPIO\_T \*pGPIO,uint8\_t port)**