simpleMSP Library

面向MSP430G2553的C语言库

编写这个库的目的是简化MSP430G2553开发过程，将繁琐的位操作简化为更直观的函数控制。

该库有针脚方向控制、针脚功能控制、针脚上拉电阻控制、数字输入、数字输出、模拟输入、中断控制、计时器控制等函数，可以直观地编写程序。

例如，将P1.6输出设置为高电平，

使用位操作：P1OUT |= BIT6 ;

使用simpleMSP库：digitalOutput(P16, HIGH) ;

例如，开启P1.3的输入中断，检测下降沿，

使用位操作：P1IE |= BIT3 ; P1IES |= BIT3 ;

使用simpleMSP库：interruptPin(P13, ENABLE, FALLING\_EDGE) ;

针脚名称定义：

|  |  |
| --- | --- |
| 板子上的名称 | 定义名称 |
| P1.0 | P10 |
| P1.1 | P11 |
| P1.2 | P12 |
| … | … |
| P2.5 | P25 |
| P2.6 | P26 |
| P2.7 | P27 |

计时器名称定义：

TimerA0：TIMER0

TimerA1：TIMER1

更多定义请参考simpleMSP.h

函数说明；

/\*\*

\* 设置一个针脚的输入输出方向

\*

\* \变量 pin ：针脚名称

\*

\* \变量 dir ：INPUT 或 OUTPUT

\*/

**void** **directionPin**(**const** uint8\_t pin, **const** uint8\_t dir);

/\*\*

\* 给一个针脚输出数字信号，高电平或低电平

\*

\* \变量 pin 针脚名称

\*

\* \value HIGH 或 LOW

\*/

**void** **digitalWrite**(**const** uint8\_t pin, **const** uint8\_t value);

/\*\*

\* 读取一个针脚的数字信号

\*

\* \变量 pin 针脚名称

\*

\* \返回 HIGH 或 LOW

\*/

**int** **digitalRead**(**const** uint8\_t pin);

/\*\*

\* 设置一个针脚的工作方式（设置PxSEL和PxSEL2）

\*

\* \变量 pin 针脚名称

\*

\* \变量 mode 从0到3，0为普通输入输出模式

\*/

**void** **selectPin**(**const** uint8\_t pin, **const** uint8\_t mode);

/\*\*

\* 为一个pin设置上拉电阻（启用或禁用）

\*

\* \变量 pin 针脚名称

\*

\* \变量 enable ENABLE 或 DISABLE

\*/

**void** **pullupPin**(**const** uint8\_t pin, **const** uint8\_t enable);

/\*\*

\* 为一个针脚设置中断（启用或禁用，上升沿检测或下降沿检测）

\*

\* \变量 pin 针脚名称

\*

\* \变量 enable ENABLE 或 DISABLE

\*

\* \变量 edge RISING\_EDGE 或 FALLING\_EDGE, 不输入该变量则默认为下降沿

\*/

**void** **interruptPin**(**const** uint8\_t pin, **const** uint8\_t enable, ...);

/\*\*

\* 启用按钮S2。（设置P1SEL为0, P1SEL2为0, P1DIR为0, P1REN为1, P1OUT为1）

\*/

**void** **enableS2**();

/\*\*

\* 初始化所有针脚为输出模式并且输出为0

\*/

**void** **initPorts**();

/\*\*

\* 读取模拟信号并转换成数字信号

\*

\* ADC10详细配置参数请看simpleMSP.h和simpleMSP.c

\*

\* \变量 input\_channel 模拟信号的来源，可以为以下值

\* A0, A1, A2, A3, A4, A5, A6, A7,

\* VeREFp, VREFnByVeREFn, Temperature\_sensor, HalfOfVCCMinusVSS

\*/

uint16\_t **analogRead**(**const** inch input\_channel);

/\*\*

\* 初始化timer，使用**timer\_create**函数前需要先使用该函数

\*/

**void** **timer\_init**();

/\*\*

\* 创建一个使用SMCLK的timer并开始计时，自动设置MC。

\*

\* \变量 timer TIMER0 或 TIMER1

\*

\* \变量 microsecond 计时器的触发时间（CCR），单位为微秒，从0到524280。

\*

\* \变量 mode\_control STOP\_MODE, UP\_MODE, CONTINUOUS\_MODE, UP\_DOWN\_MODE

\*/

**void** **timer\_create**(timer timer, uint32\_t microsecond,

timer\_mode\_control mode\_control);

/\*\*

\* 停止并删除一个设置好的timer

\*

\* \变量 timer TIMER0 或 TIMER1

\*/

**void** **timer\_delete**(timer timer);

/\*\*

\* 反转一个针脚的输出电平。高变低或低变高

\*

\* \变量 pin 针脚名称

\*/

**void** **reverseOutput**(**const** uint8\_t pin);

/\*\*

\* 清除一个针脚或timer的中断触发旗帜

\*

\* \变量 pin 针脚名称, TIMER0 或 TIMER1

\*/

**void** **clearFlag**(**const** uint8\_t pin);

blink例子：

// 闪烁红led

**#include** <msp430.h>

**#include** "simpleMSP.h" // 引入库

**void** **main**(**void**) {

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

directionPin(P10, OUTPUT); // 将P1.0设置为输出方向（P1.0连接着红led）

**while** (1) {

**\_\_delay\_cycles**(1000000); // 延迟一秒

reverseOutput(P10); // 将P1.0的输出电平反向（高变低，低变高）

}

}