关于ATOMIC\_BLOCK：详见《avr-libc-user-manual.pdf》page303。

https://forum.arduino.cc/t/using-cli-and-sei-in-a-library/70588/17

https://www.arduino.cc/reference/en/language/variables/variable-scope-qualifiers/volatile/

https://www.mianshigee.com/note/detail/2039yph/

原子操作：在操作的时候不能被中断的操作，故原子操作时，中断被屏蔽。

为何要进行原子操作？看下面的代码：

#include <inttypes.h>

#include <avr/interrupt.h>

#include <avr/io.h>

volatile uint16\_t ctr;

ISR(TIMER1\_OVF\_vect)

{

ctr--;

}

...

int

main(void)

{

...

ctr = 0x200;

start\_timer();

while (ctr != 0)

// wait

;

...

}

上述代码，在某种情况下，会在ctr!=0时跳出while循环，但这是不期望的。例如ctr的值为0x100，由于ctr是16位的，而AVR单片机是8位的，编译器先判断ctr的低8位，再判断其高8位。当ctr为0x100时，其低8位为0，接着，编译器判断高8位时，此时，中断正好发生，则先执行中断，中断结束后，ctr的值为0x0ff，此时，ctr的高8位也为0，这样，在中断结束后返回再判断ctr的高8位时，编译器就认为ctr的高8位也为0了。此时，编译器会认为ctr的低8位与高8位均为0，从而跳出while循环。

为解决这一问题，可以添加atomic.h头文件进行原子操作：

#include <inttypes.h>

#include <avr/interrupt.h>

#include <avr/io.h>

#include <util/atomic.h>

volatile uint16\_t ctr;

ISR(TIMER1\_OVF\_vect)

{

ctr--;

}

...

int

main(void)

{

...

ctr = 0x200;

start\_timer();

sei();

uint16\_t ctr\_copy;

do

{

ATOMIC\_BLOCK(ATOMIC\_FORCEON)

{

ctr\_copy = ctr;

}

}

while (ctr\_copy != 0);

...

}

ATOMIC\_BLOCK()有两个参数，分别是ATOMIC\_FORCEON和ATOMIC\_RESTORESTATE，这两个参数的区别是，当ATOMIC\_BLOCK()之前有sei()，即使能全局中断指令，则用ATOMIC\_FORCEON。若ATOMIC\_BLOCK()之前没有sei()，或者不太确定，则应该用ATOMIC\_RESTORESTATE。总之，还是建议使用ATOMIC\_RESTORESTATE。