

C S4680-101_EMBEDDED SYSTEMS (SPRING 2021)

[Dashboard](#) / [My courses](#) / [C S4680101-14385202110 \(SPRING 2021\)](#) / [Assignments](#)
/ [Grad Project \(ACX 5\): Complete Remaining ACX functions](#)

Grad Project (ACX 5): Complete Remaining ACX functions

Introduction

Sometimes it is useful for one thread to control another's state (whether or not it is READY to run), and sometimes it is useful for threads to put themselves to sleep until an event causes them to wake. The suspend/resume and disable/enable functions provide a way to do this.

IMPORTANT: As you write these functions, consider that since they do not perform any rescheduling (no call to `x_yield`) they may be called from ISRs. Since the status variables require read-modify-write (non-atomic) access, it is necessary to disable interrupts during the updates. If called from ISR code, we don't want to re-enable interrupts, but rather just restore the state of enable that was in place when the function was called. To do this, code of the following form should be used:

- `uint8_t temp = SREG; // save SREG --holds global interrupt enable bit`
- `cli(); // disable interrupts`
- do the atomic access
- `SREG = temp; // restore interrupt state`

These steps may be carried out by using the `ATOMIC_BLOCK` macro that is available in the `<util/atomic.h>` include file:

```
ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {  
    // do atomic access  
}
```

See [ATOMIC in AVR-LibC documentation](#).

Details

Complete the following ACX kernel functions:

```
x_suspend (uint8_t tid) - suspend the specified thread by setting its "suspend" status bit.  
x_resume  (uint8_t tid) - resume the specified thread by clearing its "suspend" status bit.  
x_disable (uint8_t tid) - disable the specified thread by setting its "disable" status bit.  
x_enable  (uint8_t tid) - enable the specified thread by clearing its "disable" status bit.  
unsigned long x_gtime() - return the value of the 4-byte tick counter. This would best be implemented as a macro (assuming the counter is global)  
uint8_t x_getID() - return the ID of the calling thread. This should be implemented as a macro (assuming x_thread_id is a global variable).
```

Testing

Test your functions by having one thread suspend/resume and disable/enable another. The target thread may be "blinking" at a fast rate and it can be turned off and on by calls to suspend/resume or disable/enable made by another thread.


Submission

Submit your ACX project in a file `acx.zip` at the prompt.

Submission status

**Submission
status**

Submitted for grading

Grading status	Not graded
Due date	Tuesday, April 20, 2021, 11:55 PM
Time remaining	Assignment was submitted 4 hours 40 mins early
Last modified	Tuesday, April 20, 2021, 7:14 PM
File submissions	<div><div> ACX05.zip</div><div>April 20 2021, 7:14 PM</div></div>

Submission
comments

▶ [Comments \(0\)](#)

Edit submission

Remove submission

You can still make changes to your submission.

◀ [Grad Project \(ACX 4\): x_delay - Delay a thread by specified number of "system ticks"](#)

Jump to...

[AVRDUDE Command for Atmel Studio setup](#) ▶



You are logged in as Alex Clarke (Log out)

AsULearn Sites

[AsULearn-2018-19](#)

[AsULearn-Projects](#)

[AsULearn-Global](#)

Get the mobile app