Debug session 2023-10-06

```
$ git checkout 35814d94849b103038aef05fdb1ec18e93cee3df

$ make test/modem_demo-griffin-nuttx-dev

$ cd target/test/modem_demo-griffin-nuttx-dev/
$ arm-none-eabi-gdb modem_demo-griffin-nuttx-dev

(gdb) target extended-remote :3333
(gdb) load

(gdb) run

^C
Program received signal SIGINT, Interrupt.
0x08050122 in up_mdelay (milliseconds=250) at common/arm_mdelay.c:51
51            for (j = 0; j < CONFIG_BOARD_LOOPSPERMSEC; j++)
(gdb) bt
#0  0x08050122 in up_mdelay (milliseconds=250) at common/arm_mdelay.c:51
#1  0x08044ad4 in _assert (filename=0x8056ee4 "armv7-m/arm_hardfault.c", linenum=175,
msg=0x8056edc "panic") at misc/assert.c:530
#2  0x08025132 in __assert (filename=0x8056ee4 "armv7-m/arm_hardfault.c", linenum=175,
msg=0x8056edc "panic") at assert/lib_assert.c:36
#3  0x080140ca in arm_hardfault (irq=3, context=0x3001187c, arg=0x0) at
armv7-m/arm_hardfault.c:175
#4  0x080178be in irq_dispatch (irq=3, context=0x3001187c) at irq/irq_dispatch.c:171
#5  0x08013686 in arm_doirq (irq=3, regs=0x0) at armv7-m/arm_doirq.c:70
#6  0x0801093a in exception_common () at armv7-m/arm_exception.S:224

(gdb) monitor cortex_m vector_catch all

(gdb) run

Program received signal SIGTRAP, Trace/breakpoint trap.
exception_common () at armv7-m/arm_exception.S:144
144           mrs           r0, ipsr                            /* R0=exception number */
(gdb) bt
#0  exception_common () at armv7-m/arm_exception.S:144
#1  <signal handler called>
#2  0x00010000 in ?? ()
#3  0x08017d2a in sched_unlock () at sched/sched_unlock.c:231
#4  0x20000000 in ?? ()
```

Stack is already corrupt when the exception triggers, so I cannot get info. I need to see when the stack gets broken.

```
(gdb) info registers
[...]
sp             0x300118e8          0x300118e8
[...]

(gdb) set $i=0
(gdb) while ($i < 8)
 >p g_pidhash[$i]->stack_base_ptr
 >set $i=$i + 1
 >end
$1 = (void *) 0x24006008 <idle_stack+24>
$2 = (void *) 0x30000758
```

```
$3 = (void *) 0x30001318
$4 = (void *) 0x300121c8
$5 = (void *) 0x3000a998
$6 = (void *) 0x30005ca8
$7 = (void *) 0x3000e9b8
$8 = (void *) 0x300119c0
```

We are inside the stack `g_pidhash[7]` stack. Let's see what that task is:

```
(gdb) p *g_pidhash[7]
$9 = {flink = 0x0, blink = 0x3000a578, group = 0x3000a620, pid = 7, sched_priority = 100 'd',
init_priority = 100 'd', start = 0x8056241 <pthread_start>, entry = {pthread = 0x8030459
<_sld_thread>,
    main = 0x8030459 <_sld_thread>}, task_state = 6 '\006', flags = 4097, lockcount = 0,
errcode = 0, waitdog = {next = 0x30008fdc, arg = 805342976, func = 0x801cf99 <nxsig_timeout>,
lag = 1},
  adj_stack_size = 2024, stack_alloc_ptr = 0x300119a8, stack_base_ptr = 0x300119c0, waitobj =
0x0, sigprocmask = {_elem = {0, 0}}, sigwaitmask = {_elem = {0, 0}}, sigpendactionq = {head =
0x0, tail = 0x0},
  sigpostedq = {head = 0x0, tail = 0x0}, sigunbinfo = {si_signo = 255 '\377', si_code = 2
'\002', si_errno = 110 'n', si_value = {sival_int = 0, sival_ptr = 0x0}, si_user = 0x0}, mhead
= 0x0, xcp = {
    sigdeliver = 0x0, saved_regs = 0x0, regs = 0x30011e5c}, name = "main", '\000' <repeats 27
times>}
```

It is `_sld_thread`. Stack bottoms should never be traversed under normal conditions, so let's put a watchpoint there and see who corrupts this.

```
(gdb) watch *0x300119c0
(gdb) run

Hardware watchpoint 1: *0x300119c0

Old value = 0
New value = -559038737
0x0802e40a in arm_stack_color (stackbase=0x300119a8, nbytes=2048) at
common/arm_checkstack.c:179
179       while (nwords-- > 0)
(gdb) c
Continuing.
```

That is normal, when starting the thread, stack coloring initializes the stack so it writes on stack bottom. We should never reach that point while the thread is running.

```
Hardware watchpoint 1: *0x300119c0

Old value = -559038737
New value = 0
0x08017890 in irq_dispatch (irq=68, context=0x30011ab4) at irq/irq_dispatch.c:135
135       if ((unsigned)irq < NR_IRQS)
(gdb) bt
#0  0x08017890 in irq_dispatch (irq=68, context=0x30011ab4) at irq/irq_dispatch.c:135
#1  0x08013686 in arm_doirq (irq=68, regs=0x0) at armv7-m/arm_doirq.c:70
#2  0x0801093a in exception_common () at armv7-m/arm_exception.S:224
Backtrace stopped: previous frame identical to this frame (corrupt stack?)
```

The problem here is `exception_common` modifies the stack to allow for some extra space for signal handling, and gdb is unable to traverse it. To be able to know where we come from, we need to go back to `exception_common`.

```
(gdb) finish
Run till exit from #0  0x08017890 in irq_dispatch (irq=68, context=0x30011ab4) at
irq/irq_dispatch.c:135

Hardware watchpoint 1: *0x300119c0

Old value = 0
New value = 536871312
0x080178ba in irq_dispatch (irq=68, context=0x30011ab4) at irq/irq_dispatch.c:171
171      CALL_VECTOR(ndx, vector, irq, context, arg);
(gdb) finish
Run till exit from #0  0x080178ba in irq_dispatch (irq=68, context=0x30011ab4) at
irq/irq_dispatch.c:171
arm_doirq (irq=68, regs=0x0) at armv7-m/arm_doirq.c:78
78       if (regs == NULL)
(gdb) finish
Run till exit from #0  arm_doirq (irq=68, regs=0x0) at armv7-m/arm_doirq.c:78
exception_common () at armv7-m/arm_exception.S:230
230           ldmia         r0!, {r2-r11,r14}     /* Recover R4-R11, r14 + 2 temp values */
Value returned is $26 = (uint32_t *) 0x30011ab4
(gdb) bt
```

Now we do `next` until we are about to return from `exception_common`, and we can examine the backtrace.

```
(gdb) next
280           bx            r14                                   /* And return */
(gdb) bt
#0  exception_common () at armv7-m/arm_exception.S:280
#1  <signal handler called>
#2  0x081fcc16 in __CRYPTO_START ()
#3  0x0803d652 in file_write (filep=0x23829314, buf=0xff3a2040, nbytes=1510187929) at
vfs/fs_write.c:89
```

But gdb is missing the information for the crypto module so it can not traverse those stacks. Add it.

```
(gdb) add-symbol-file ../../baremetal/griffin-crypto/griffin-crypto
add symbol table from file "../../baremetal/griffin-crypto/griffin-crypto"
(y or n) y
Reading symbols from ../../baremetal/griffin-crypto/griffin-crypto...
(gdb) bt
#0  exception_common () at armv7-m/arm_exception.S:280
#1  <signal handler called>
#2  0x081fcc16 in __CRYPTO_START ()
#3  0x081fcf4e in __CRYPTO_START ()
#4  0x081fc41a in __CRYPTO_START ()
#5  0x081fc2c6 in __CRYPTO_START ()
#6  0x081fb810 in __CRYPTO_START ()
#7  0x08043220 in flash_eMMC_EncryptAndWritePage (pPageData=pPageData@entry=0x300091f8 "",
uiPage=uiPage@entry=65535, bFlags=bFlags@entry=2 '\002') at
../../../../../driver/flash/griffin/../flash_crypt.c:486
#8  0x08043516 in flash_API_Encrypt (write_buff=write_buff@entry=0x300091f8 "",
page=page@entry=13542) at ../../../../../driver/flash/griffin/flash_crypt_griffin.c:142
```

```
#9  0x08042c9c in golegacyfs_write (filep=0x3000a928, buffer=0x30012048 '\377' <repeats 21
times>, "\207\004\b\200", buflen=21) at
../../../../../driver/golegacyfs/nuttx/golegacyfs.c:617
#10 0x0803d652 in file_write (filep=0x3000a928, buf=0x30012048, nbytes=21) at
vfs/fs_write.c:89
#11 0x0803d694 in nx_write (fd=4, buf=0x30012048, nbytes=21) at vfs/fs_write.c:138
#12 0x0803d6b6 in write (fd=4, buf=0x30012048, nbytes=21) at vfs/fs_write.c:202
#13 0x080359a2 in dataLog_write (self=self@entry=0x30008dc0, log=log@entry=0x300120d8) at
../../../service/genericQueue/dataLog.c:216
#14 0x08030988 in dlQ_Send (self=0x30008da0, data=0x300120d4 "", len=<optimized out>) at
../../../service/genericQueue/dataLogQueue.c:195
#15 0x08030640 in genQ_Send (self=<optimized out>, data=data@entry=0x300120d4 "",
len=len@entry=28) at ../../../service/genericQueue/genericQueue.c:148
#16 0x08030528 in _sld_processor (ctx=0x30008d8c) at
../../../service/syslogDebug/sld_processor.c:63
#17 _sld_thread (arg=0x30008d70) at ../../../service/syslogDebug/sld_processor.c:118
#18 0x08048630 in pthread_startup (entry=0x8030459 <_sld_thread>, arg=0x30008d70) at
pthread/pthread_create.c:59
#19 0x0805627a in pthread_start () at pthread/pthread_create.c:140
#20 0x00000000 in ?? ()
```

And now we can see the stack usage.

```
(gdb) frame apply all info frame
#0  exception_common () at armv7-m/arm_exception.S:280
Stack level 0, frame at 0x30011b20:
[...]
#1  <signal handler called>
Stack level 1, frame at 0x30011b88:
[...]
#2  0x081fcc16 in __CRYPTO_START ()
Stack level 2, frame at 0x30011cc8:
[...]
#3  0x081fcf4e in __CRYPTO_START ()
Stack level 3, frame at 0x30011ce8:
[...]
#4  0x081fc41a in __CRYPTO_START ()
Stack level 4, frame at 0x30011d00:
[...]
#5  0x081fc2c6 in __CRYPTO_START ()
Stack level 5, frame at 0x30011d18:
[...]
#6  0x081fb810 in __CRYPTO_START ()
Stack level 6, frame at 0x30011d30:
[...]
#7  0x08043220 in flash_eMMC_EncryptAndWritePage (pPageData=pPageData@entry=0x300091f8 "",
uiPage=uiPage@entry=65535, bFlags=bFlags@entry=2 '\002') at
../../../../../driver/flash/griffin/../flash_crypt.c:486
Stack level 7, frame at 0x30011f88:
[...]
#8  0x08043516 in flash_API_Encrypt (write_buff=write_buff@entry=0x300091f8 "",
page=page@entry=13542) at ../../../../../driver/flash/griffin/flash_crypt_griffin.c:142
Stack level 8, frame at 0x30011f90:
[...]
#9  0x08042c9c in golegacyfs_write (filep=0x3000a928, buffer=0x30012048 '\377' <repeats 21
times>, "\207\004\b\200", buflen=21) at
../../../../../driver/golegacyfs/nuttx/golegacyfs.c:617
Stack level 9, frame at 0x30011fe8:
[...]
#10 0x0803d652 in file_write (filep=0x3000a928, buf=0x30012048, nbytes=21) at
vfs/fs_write.c:89
```

```
Stack level 10, frame at 0x30012008:
[...]
#11 0x0803d694 in nx_write (fd=4, buf=0x30012048, nbytes=21) at vfs/fs_write.c:138
Stack level 11, frame at 0x30012028:
[...]
#12 0x0803d6b6 in write (fd=4, buf=0x30012048, nbytes=21) at vfs/fs_write.c:202
Stack level 12, frame at 0x30012048:
[...]
#13 0x080359a2 in dataLog_write (self=self@entry=0x30008dc0, log=log@entry=0x300120d8) at
../../../service/genericQueue/dataLog.c:216
Stack level 13, frame at 0x30012078:
[...]
#14 0x08030988 in dlQ_Send (self=0x30008da0, data=0x300120d4 "", len=<optimized out>) at
../../../service/genericQueue/dataLogQueue.c:195
Stack level 14, frame at 0x300120c0:
[...]
#15 0x08030640 in genQ_Send (self=<optimized out>, data=data@entry=0x300120d4 "",
len=len@entry=28) at ../../../service/genericQueue/genericQueue.c:148
Stack level 15, frame at 0x300120c8:
[...]
#16 0x08030528 in _sld_processor (ctx=0x30008d8c) at
../../../service/syslogDebug/sld_processor.c:63
Stack level 16, frame at 0x30012188:
[...]
#17 _sld_thread (arg=0x30008d70) at ../../../service/syslogDebug/sld_processor.c:118
Stack level 17, frame at 0x30012188:
[...]
#18 0x08048630 in pthread_startup (entry=0x8030459 <_sld_thread>, arg=0x30008d70) at
pthread/pthread_create.c:59
Stack level 18, frame at 0x30012198:
[...]
#19 0x0805627a in pthread_start () at pthread/pthread_create.c:140
Stack level 19, frame at 0x300121a8:
[...]
#20 0x00000000 in ?? ()
Stack level 20, frame at 0x300121a8:
```

Stack is not overflowed here, but when the interrupt triggers, the extra code of the ISR is run on the interrupted thread stack, causing an overflow.

Adding code to set `_sld_thread` stack to 16384 prevents the hardfault.