CS452 Kernel 3

Christoph Ulshoefer (20751216) and Thomas Broadley (20522223)

February 05, 2018

1 Running the kernel

Execute the following commands in RedBoot to start the kernel:

```
load -b 0x00218000 -h 10.15.167.5 "ARM/csulshoe/k3.elf"
go
```

Our K3 ELF file is stored at `/u3/cs452/tftp/ARM/csulshoe/k3.elf` in the CS Student Computing Environment. Our K3 source code can be found at https://git.uwaterloo.ca/csulshoe/cs452/tree/k3-done.

2 Implementation

2.a Event handling system

In our kernel's event handling system, each task can wait on at most one event at a time, and each event type can be waited on by at most one task at a time. If a task calls `AwaitEvent` for an event on which another task is already waiting, the kernel returns an error.

This design is sufficient because we don't expect to use multiple notifiers for the same event type. It only makes sense to use multiple notifiers if a send/receive/reply loop is slower than the time between interrupts from one or more devices. If this ends up happening, we think that the proper solution would be to optimize our context switch, scheduling, and message-passing to speed up. If we end up using multiple notifiers for other reasons, we can implement event queues using the generic queue implementation that we currently use for ready and send queues.

2.b Clock server

The clock server stores records of waiting tasks in a min-heap. This implementation has $O(\log n)$ runtime for insertion and removal of the root node, and $O(1)$ for getting a reference to that node. We think that a heap is appropriate because every task that is added to the heap (because it called `Delay` or `DelayUntil`) will eventually be removed (because it has reached the end of its delay). It is also important to be able to quickly check if there is a task that should be woken, since the clock server will check if it should wake any delaying tasks much more frequently than it will respond to a call to `Delay` or `DelayUntil`. With a min-heap, this can be done by looking at the root node.

2.c Idle task

The idle task is implemented as a regular task that the first user task creates. It automatically configures itself by calling `Time()` to busy-loop as close to one second as possible, while trying to minimize the number of calls to `Time()`, to not artificially overestimate the time run in the kernel. Concretely, we always want to call `Time()` between one and five times per second, and increase/decrease the upper bound at which we are busy-looping by 5% depending on whether we called `Time()` too often/too little in the past second.

2.c.1 Idle task CPU usage

On each software or hardware interrupt, we record the number of clock ticks since the previous interrupt, which tells us how long the current task has run for. On leaving the kernel, we record how long it took to service the interrupt and schedule the next task.

We also add these runtime statistics to a buffer. After the idle task has been running for a second, we look up the time the task has been running, and return the fraction of the total runtime this task has been running.

3 Extra features

3.a CPU usage logging

We track the CPU usage of all tasks, not just the idle task. On kernel exit, we print a summary of how long tasks (and the kernel, listed under task ID zero) have run, both in milliseconds and as a percentage of total runtime.

3.b Syscalls

The `MyPriority` syscall returns the priority of the currently-running task. We've found this syscall useful for writing end-to-end tests whose results don't change, even if we change the priority of the test's first task.

The `Kill` syscall removes the given task from its ready queue, deregisters it from the event for which it is waiting (if any), and puts it in a zombie state. We use this syscall to kill the idle task and all notifiers before trying to exit the kernel.

3.c Logging

We implemented a logging framework that simply appends strings to a buffer and prints the buffer on kernel exit. We plan on using this framework in later assignments when printing debug information to the console will not work anymore due to cursor addressing.

4 Demo output

Here is the full output of a representative run of our K3 demo tasks:

```
Tid: 8 Delay interval: 10  1 out of 20 delays completed Delayed  10 ticks so far
Tid: 8 Delay interval: 10  2 out of 20 delays completed Delayed  20 ticks so far
Tid: 7 Delay interval: 23  1 out of  9 delays completed Delayed  23 ticks so far
86.4%
Tid: 8 Delay interval: 10  3 out of 20 delays completed Delayed  30 ticks so far
Tid: 6 Delay interval: 33  1 out of  6 delays completed Delayed  33 ticks so far
Tid: 8 Delay interval: 10  4 out of 20 delays completed Delayed  40 ticks so far
Tid: 7 Delay interval: 23  2 out of  9 delays completed Delayed  46 ticks so far
Tid: 8 Delay interval: 10  5 out of 20 delays completed Delayed  50 ticks so far
84.3%
Tid: 8 Delay interval: 10  6 out of 20 delays completed Delayed  60 ticks so far
Tid: 6 Delay interval: 33  2 out of  6 delays completed Delayed  66 ticks so far
Tid: 7 Delay interval: 23  3 out of  9 delays completed Delayed  69 ticks so far
Tid: 8 Delay interval: 10  7 out of 20 delays completed Delayed  70 ticks so far
Tid: 5 Delay interval: 71  1 out of  3 delays completed Delayed  71 ticks so far
Tid: 8 Delay interval: 10  8 out of 20 delays completed Delayed  80 ticks so far
80.8%
Tid: 8 Delay interval: 10  9 out of 20 delays completed Delayed  90 ticks so far
Tid: 7 Delay interval: 23  4 out of  9 delays completed Delayed  92 ticks so far
Tid: 6 Delay interval: 33  3 out of  6 delays completed Delayed  99 ticks so far
Tid: 8 Delay interval: 10 10 out of 20 delays completed Delayed 100 ticks so far
Tid: 8 Delay interval: 10 11 out of 20 delays completed Delayed 110 ticks so far
82.9%
Tid: 7 Delay interval: 23  5 out of  9 delays completed Delayed 115 ticks so far
Tid: 8 Delay interval: 10 12 out of 20 delays completed Delayed 120 ticks so far
Tid: 8 Delay interval: 10 13 out of 20 delays completed Delayed 130 ticks so far
Tid: 6 Delay interval: 33  4 out of  6 delays completed Delayed 132 ticks so far
Tid: 7 Delay interval: 23  6 out of  9 delays completed Delayed 138 ticks so far
Tid: 8 Delay interval: 10 14 out of 20 delays completed Delayed 140 ticks so far
80.7%
Tid: 5 Delay interval: 71  2 out of  3 delays completed Delayed 142 ticks so far
Tid: 8 Delay interval: 10 15 out of 20 delays completed Delayed 150 ticks so far
Tid: 8 Delay interval: 10 16 out of 20 delays completed Delayed 160 ticks so far
Tid: 7 Delay interval: 23  7 out of  9 delays completed Delayed 161 ticks so far
Tid: 6 Delay interval: 33  5 out of  6 delays completed Delayed 165 ticks so far
82.9%
Tid: 8 Delay interval: 10 17 out of 20 delays completed Delayed 170 ticks so far
Tid: 8 Delay interval: 10 18 out of 20 delays completed Delayed 180 ticks so far
Tid: 7 Delay interval: 23  8 out of  9 delays completed Delayed 184 ticks so far
Tid: 8 Delay interval: 10 19 out of 20 delays completed Delayed 190 ticks so far
85.2%
Tid: 6 Delay interval: 33  6 out of  6 delays completed Delayed 198 ticks so far
Tid: 8 Delay interval: 10 20 out of 20 delays completed Delayed 200 ticks so far
Tid: 7 Delay interval: 23  9 out of  9 delays completed Delayed 207 ticks so far
Tid: 5 Delay interval: 71  3 out of  3 delays completed Delayed 213 ticks so far
Ticks: 213
TOTAL: 2125
0:  80ms  (3.8%)
1:  1ms  (0.0%)
2:  0ms  (0.0%)
3:  1776ms  (83.6%)
4:  6ms  (0.3%)
5:  21ms  (1.0%)
```

```
6:  42ms  (2.0%)
7:  63ms  (2.9%)
8:  139ms  (6.5%)
9:  4ms  (0.2%)
```

The first user task creates four client tasks with increasing priority, *i.e.* task 8 has the highest priority of the clients and task 5 has the lowest. Therefore, task 8 makes the first request for parameters and receives the pair (20, 10).

Each client task prints a single line of output when it wakes from a call to `Delay`. These lines are sorted by the total number of ticks delayed so far ascending. Also, the idle task prints its CPU usage every quarter of a second.

When the kernel quits, the idle task prints a more detailed breakdown of the CPU usage of each task. Task 0 represents the kernel. We found that about 1,400 context switches occur during the K3 demo tasks, for an average of 57 milliseconds per context switch. Given that we built our K3 binary without `-O2` optimization, we feel good about this number.

The first user task is task 1. It has the highest priority out of all the K3 tasks, so it always gets to run if it is ready. It creates the nameserver first, which receives the task ID 2. Both of these tasks spend most of their time waiting on messages from other tasks, so they barely use the CPU at all.

The first user task then creates the idle task (task 3), the clock server (task 4), and the clients (tasks 5 to 8). The client tasks take more CPU time than the other tasks because they use busy-wait I/O functions. Each line printed by a client contains exactly 80 characters, which gives a lower bound of 6.94 milliseconds for printing one line. This is consistent with our observation that it takes $139 \approx 6.94 \cdot 20$ milliseconds for task 8 to print 20 lines.

Finally, the clock notifier is created by the clock server. Since the first user task has the highest priority, it does not allow the clock server to create the clock notifier until after it has created the client tasks. Since the notifier spends most of its time either event-, receive-, or reply-blocked, it makes sense that this task also uses very little CPU time.

5 File and repository hashes

The commit before adding this report had SHA `61317387632a052b34b92c8d6e5478592918f9dd`.

5.a MD5 hashes

Here is a list of the MD5 hashes of the files included in our repository.

```
$ git ls-files | grep -v 'test/googletest' | xargs md5sum

3baa32ac2b0d0342c6975830905a3da0  .gitignore
4bbdec35a4b9dc9ca8615b1e321de502  .gitmodules
9abc0052245d3d46c0dd217499006d59  .travis.yml
cdbd7fbeb502f2adcba5aeb12b7e8e8c  Doxyfile
ddc373f47b3cde9ce458bdcdc02a3fad  Makefile
e4cd344dae45ffb7851fba8861c75f73  README.md
b5c56249bd2a019688db580020d8e0fd  include/myio.c
6fb5e49cc7fde40fc830b3a9def05e9a  include/myio.h
0ff6bee28590e1b949a208361291deab  include/mytimer.c
0776bbc8e09735c39ee6ad1a5465b08b  include/mytimer.h
a51b441bd12f84e40333d2dfde291780  include/timer_data.h
```

```
74c63a11fc85af6c146dbcb5fa19b14a  k2benchmark.txt
700f7d739ab608d3eff30ff0ede15cf9  kernel/asm/kernToUser.s
d1d5fe7833e97870de0e1539f8581727  kernel/asm/startup.s
4e2713f6d3e909275898f30830b6d9dd  kernel/asm/trap.s
770dc066a65a334181fb303745f14dad  kernel/include/labenv/timer.c
a696fe702d49142a7a1d9961e0817256  kernel/include/labenv/timer.h
e507887d767f88763b039b331c79279e  kernel/include/labenv/ts7200.h
ee90c5ee9fadbe5f7cf574db5241d256  kernel/include/versatilepb/timer.c
eb6c4ecdbda72702ca44cc8132804d9b  kernel/include/versatilepb/timer.h
683e991024991dd265d82f5ea4f2da08  kernel/include/versatilepb/versatilepb.h
0421e442c4abc4951d2224f4d88ad8eb  kernel/src/cp_vec.c
503fb56ff1929bcc8d7a6b3d19b2524e  kernel/src/cp_vec.h
327869ea50f6243d3a90357980a734e1  kernel/src/events.c
a5d97dbe6fb51ef1d1c36d6259e7caf2  kernel/src/events.h
ee3e3cf6322f5e8e22f92fe6b2bd1c39  kernel/src/handle_abort.c
d4b4b45dde29a262d359bf1ad0e49063  kernel/src/interrupt.c
d7053b414dcd12abb4948e53b8dfcdaa  kernel/src/interrupt.h
a0946900f956766d853fa86ea49f1628  kernel/src/kassert.c
8baadeeeeb32c820cce4b6420efcb92b  kernel/src/kassert.h
d97f9b367d708c6f2d395d840d5dc091  kernel/src/kusage_stats.c
c1ed6afa70cb921f87c0e90770edb87b  kernel/src/kusage_stats.h
f65445e06ed7f0bd89cc194e6f9f3182  kernel/src/multitasking/messaging.c
8da39e0fbf99d840a126cad76ea4ed57  kernel/src/multitasking/messaging.h
db3c340adf6f2f90834328e4a7414daf  kernel/src/multitasking/ready_queue.c
acdaabac90092feb2d815d93daa6c160  kernel/src/multitasking/ready_queue.h
1c6f60c335ef23ed725c4cacb490162d  kernel/src/multitasking/schedule.c
a11a9befd765f6c6c048f6b84860d0f9  kernel/src/multitasking/schedule.h
79d188029fc40d37d495f9ba4df52032  kernel/src/multitasking/scheduler.c
7eec14840f74239d954e1ee02527de33  kernel/src/multitasking/scheduler.h
ce79a33fb4e09a3b869b2d2f187df6a7  kernel/src/multitasking/send_queue.c
d2fd55f45ab57e8e5c75a5a072bb74ec  kernel/src/multitasking/send_queue.h
5fccc6982f47aed5fadc361876da6763  kernel/src/multitasking/task.c
c3919a92905c226ce2c44a835a900ac4  kernel/src/multitasking/task.h
4ad860fd0d43e6a9cf29eacf89e2b06d  kernel/src/queue.h
267a3ee133d90349d2042dbb42d373ef  kernel/src/syscall/syscall.c
0d26653864f2251c8a286335fc5d4c2a  kernel/src/syscall/syscall.h
bec5edf5b40af4209c289f8c83f248ef  lib/attributes.h
59506a2fc8c1e78f1287e2d9eb454b34  lib/benchmark.c
3b7d15e91a197c4a1a4cdfbe674f88b1  lib/benchmark.h
6da0cae2c90858162e55b14403cf1f37  lib/buffer.h
7949f4bf8c52321c0bcf49e3b67817ff  lib/buffertypes/char_buffer.c
8f75f97be05b4e5fa3ceb739f3a6d61b  lib/buffertypes/char_buffer.h
0dd2910bf5314d67758f17c17ef855ee  lib/buffertypes/int32_t_buffer.c
81ddf219ea1acff14e8237a88742c258  lib/buffertypes/int32_t_buffer.h
0df89969fdfa653c60f63d192121e423  lib/codes.c
f0550b4bdf17614a8a122576f05219ac  lib/codes.h
d41d8cd98f00b204e9800998ecf8427e  lib/constants.c
ad6bdbc46213d391b4a4a8ae3a9a0248  lib/constants.h
38424f2856725ba4ce4919eefe5e3142  lib/crash.h
608c3732de6719e206bf1c76772bd549  lib/crash.s
03a15f526c691f42cd9376310d423b30  lib/event_data.h
981090a062243d693482675cac4d54d1  lib/messages.h
ca86a48b51092733767b9cd6e4e341b4  lib/priority_queue.h
047756622572e4bdc3b73e2b8316d53e  lib/tstdlib.c
ac73fc06fd2e590943eea16a464e8a7d  lib/tstdlib.h
06e369b76c2d125b61ac6a95d7d31309  lib/usage_stats.c
e98dffa6b46a95515e24bc35cb508992  lib/usage_stats.h
1c6db5cd1d7c798edb7ecbc1b733215b  main.c
```

```
04a99fa4e57230b2b2b7a525a1105097  main.ld
d9cd792c1413ac79bf45f799ae903f42  reports/a0-csulshoe.pdf
2d81bda8e6453ea87e964edefb8d37c5  reports/k1.pdf
3c476edf2886bdb295d9f5e37a748d9c  reports/k2.pdf
1c7a429e1ced54dbb23098d3078a30e8  stats.md
da4d2d5a9d3d7971c83928f051ef6b58  test-resources/assert.c
0675503235fb2225159e6de0586c0673  test-resources/assert.h
af3d9d8c9db5792eb3f32c966185c60a  test/Makefile
d41d8cd98f00b204e9800998ecf8427e  test/e2e/__init__.py
0086aa5e32e8bd5f061180ff7d08e7f3  test/e2e/qemu_tcp_wrapper.py
f1e50c9027d900c43570595e642fd518  test/e2e/snapshots/k2.txt
57e0cf79450fa3e8dc465928d0e82668  test/e2e/snapshots/k3.txt
db3db999e689845a6c62dd81e0ae1689  test/e2e/test_clock.py
bcf5c432bb44137fa41cae0396202d30  test/e2e/test_interrupts.py
ba7c5ac14b877764610a9656dd4663a5  test/e2e/test_kernel_demo.py
c80009eeb7b6b7f1b858d71f53cc5133  test/e2e/test_messaging.py
03ebf1bbaa3244f86aebe1be020ae601  test/e2e/test_mypriority.py
50f189623bb32bb07f840e9e948ecdea  test/e2e/test_nameserver.py
29b33f3609f3f5fc0e1fd93b8aaf0917  test/e2e/test_segfaults.py
30e544606327baabd6afe99c229faf9f  test/e2e/test_test.py
3834f5e022525034f78b19b6d5b764fa  test/unit/test_all.cc
fef09e15aad9df144989ba98bdd2abe1  test/unit/test_all.h
a04866b8db525d7dd623fe60b23f7b39  test/unit/test_buffer.cc
f5785dfb91f2c6c1050d6925a08f20c6  test/unit/test_buffer.h
584d33a90964dc237012ba401c475965  test/unit/test_clock_wait_queue.cc
825a054548d34057930fc5cbea517933  test/unit/test_clock_wait_queue.h
ecd6b66fdbf4fb3d583d97d3a69be115  test/unit/test_messaging.cc
46ec33cd074fbbbca13543e5787d9b9a  test/unit/test_messaging.h
1909548396588628d189701a805e7bb4  test/unit/test_ready_queue.cc
917d4146f8727af1b9986cd4b94565a3  test/unit/test_ready_queue.h
63007125279373ae1c9a4a5e718cac2e  test/unit/test_scheduler.cc
c31f7630705c36d377356eace6e86c9b  test/unit/test_scheduler.h
904f08fbbd2fe7c3db9220d000db886a  test/unit/test_stdlib.cc
4455b0fc9b8e5d75e65bf7f23f643ccb  test/unit/test_stdlib.h
7dd46791a1a085876e62a9966c46337e  test/unit/test_task.cc
36889e34775aefba76462f7ede00349b  test/unit/test_task.h
0cd18ba380707ddfbeb0c114c941aa1f  usr/a0.c
cb3a97bfb7a6d80cb876c69e67930bf6  usr/a0.h
093c6599d47d6bab2e7f59400abd9a64  usr/a0/a0terminal.c
519b95f1f36db8f926141438fab3a5be  usr/a0/a0terminal.h
c05bd5a3556cd85fb531d19cc9ffeb76  usr/a0/a0trackstate.c
1e3b90bcbf8fdfe87d4abe58f517483b  usr/a0/a0trackstate.h
cf3657f54cdbd22adbfb073d934a95e4  usr/clock.c
961f260e35478729e081309cef7e2d4d  usr/clock.h
82516f97779f2d82d852781d82749eed  usr/idle.c
bd5a02bc871ff2c4670c1cff796149e9  usr/idle.h
686bfad6b5f7a70f10865e0bc29a5249  usr/k1.c
758b67d7c795e165ba195c0d39d6bbe0  usr/k1.h
b8923150be8054c9bfa75bd511f9bc21  usr/k2.c
1396d3d3228e3fcd0d5adc0029d0dc3e  usr/k2.h
cf33b213852de5ad4347eee9c00359ae  usr/k3.c
ee39a5ea231bfbb216625c8b94fcf636  usr/k3.h
3f9b92e877e51c8a6556121920a974e6  usr/lib/clock_wait_queue.c
b9b4baf1f42585917df639bc883bdaf4  usr/lib/clock_wait_queue.h
e5ba67b5acdaf763dbb2bcad25611f02  usr/nameserver.c
0d3c6357c34ee60e0ace26f2fd267206  usr/nameserver.h
7c3d3b1b7553e85b5ac5c952b19eb0f2  usr/spam.c
30cf6c845d6a656b57eef852f64fec2f  usr/spam.h
```

```
324862e41991fa243099d43763f7d490  usr/test/awaitevent/test_timer_interrupt.c
779b3496de81921bf5f7656fba3f2698  usr/test/awaitevent/test_timer_interrupt.h
31041ff5a0154e679718472f2cd150eb  usr/test/clock/clock_util.c
34e4bb681e9713bab63d290506402a87  usr/test/clock/test_clock.h
ccb09175ef894c389e73c61fe702295d  usr/test/clock/test_clock_accuracy.c
7b58840919f048032eab36dcaec632bf  usr/test/clock/test_clock_errors.c
7e5373809fd8b91e2c3aa4dee4872665  usr/test/clock/test_clock_syscall_accuracy.c
bed867b2ad5bb4aae38a223eb2a750ba  usr/test/clock/test_clock_syscall_errors.c
fe79579c6904565f6bb5c7e9ab764a09  usr/test/messaging/test_messaging.h
d262f35cce581eda07aff2484fd2d8b5  usr/test/messaging/test_messaging_basic.c
a958c4a1ff47cd3910266ac17c4ef746  usr/test/messaging/test_messaging_exit_with_blocked.c
497af6748eae1fdaf6a437d80dcce337  usr/test/messaging/test_messaging_fifo_send.c
f0abb91510a1c1433da5977ba2a1881d  usr/test/messaging/test_messaging_invalid_tid.c
6ac01327ddbffa06649f0c20ed04e8ec  usr/test/messaging/test_messaging_receive_before_send.c
76583e962409b9a90c62df1e97b74598  usr/test/messaging/test_messaging_reply_target_zombie.c
47c902d889407fee469b052bec91585c  usr/test/messaging/test_messaging_same_priority.c
a17bf4e6f1b05c814f24c35fe3cd5677  usr/test/messaging/test_messaging_send_recipient_zombie.c
c85eef4e381e81e765ec21f4154f21d3  usr/test/messaging/test_messaging_sequence.c
4cc729e5efa8e04a090173ad1600ecd5  usr/test/messaging/test_messaging_tree.c
8bd72ff578d51d007fe156e698e1b48c  usr/test/messaging/test_messaging_truncation.c
448f29b991c4a33c947a7fa3e1d1a139  usr/test/nameserver/fake_nameserver_functions.c
0625e69203f3e3a475a8df0d12405293  usr/test/nameserver/fake_nameserver_functions.h
e1a6690a20d3eb08bf2a1f50d10bc79c  usr/test/nameserver/test_nameserver_errors.c
a725bbc505cd3e0d275d2a04c9446c5c  usr/test/nameserver/test_nameserver_errors.h
0551649209d1fede587ee459ad60eb5a  usr/test/nameserver/test_nameserver_happypath.c
7eb3bf98e971086e9f89dceff0d125b2  usr/test/nameserver/test_nameserver_happypath.h
40fe9275ab98f76e62fdad348d3653b6  usr/test/nameserver/test_nameserver_too_many.c
e78a3bd4f80fc4ad7fedb0d1e33c300b  usr/test/nameserver/test_nameserver_too_many.h
49d53acc6624017ae17101673c15e7cf  usr/test/nameserver/test_nameserver_wrapper_errors.c
e1e47ea96833387e5df5b3d537cb77fb  usr/test/nameserver/test_nameserver_wrapper_errors.h
cf6d1110fb2c09939294b49e9b9946cd  usr/test/test_message_benchmark.c
1b1cb09716499801f120bd6427dca84a  usr/test/test_message_benchmark.h
83869951c54ac75ae2d68223a8cde7fd  usr/test/test_mypriority.c
8407db4bb1932ad2f21625c54f66eb1c  usr/test/test_mypriority.h
0a912fa2ec8da33c3b3488fc0ffaff1a  usr/test/test_runner.c
e872e1ee3e8178691accc8a5d7867964  usr/test/test_runner.h
e22f4b33d4b2703e4fb73230ed9d8195  usr/test/test_undefined_handler.c
8bbd9e31e2e699efd055a1c4cbf6baa5  usr/test/test_undefined_handler.h
af9cdf9014bec08c6df74cead3ca105d  usr/track/README.txt
7707b57cb3a6c3703c0e3da3acbef763  usr/track/parse_track.py
1e06c8505279ed9f4d10136465c81618  usr/track/parts_tracka
a408ef1736b356024bf0dc2bc05d98b0  usr/track/parts_trackb
b0c6cde547fce865e4b9184ff6481c0c  usr/track/track_data.c
8039d21694d3741e459dca0fb9b7eaa0  usr/track/track_data.h
0559a3921e16ebdb2c41f53be73ccc75  usr/track/track_node.h
50e0b1150b39a2425cb2180c33e0e57f  usr/track/tracka
dab2764f1d7f07aa454bad2ec01158b8  usr/track/trackb
6cae62c97dc3e1b1de204974a295e55a  usr/user_data_abort.h
a7668b81a997cb9d602415320f5d33f3  versatilepb.ld
3f052cb7f6b25c5e80dca05f284701d3  versatilepb_e2e.ld
03a9a6a402360d35fe9d1aedfd8e98e7  versatilepb_e2e_tmr.ld
```