Open static firmware analysis

... or what is a stack, why should I care and how to do it effectifly :)

Press Space for next page →

Slidev template neversink by Todd Gureckis

What are we talking about

1. How to tackle limited device memory on maturing firmware with increasing features
2. Motivation - Proposals where we can improve existing tools for application development

How to tackle limited device memory on maturing firmware with increasing features

Tools to help you start thinking

1. Understand RAM and ROM usage of your application (at one point)
    1. Determine dynamic stack usage - To paint a stack
    2. Looking into MAP files
    3. Looking into ELF files - static stack usages with some help of GCC
2. Understand RAM and ROM usage of your application (over time)
    1. Tracking Firmware Size Metrics

Understand RAM and ROM usage of your application (at one point)

Determine dynamic stack usage - To paint a stack

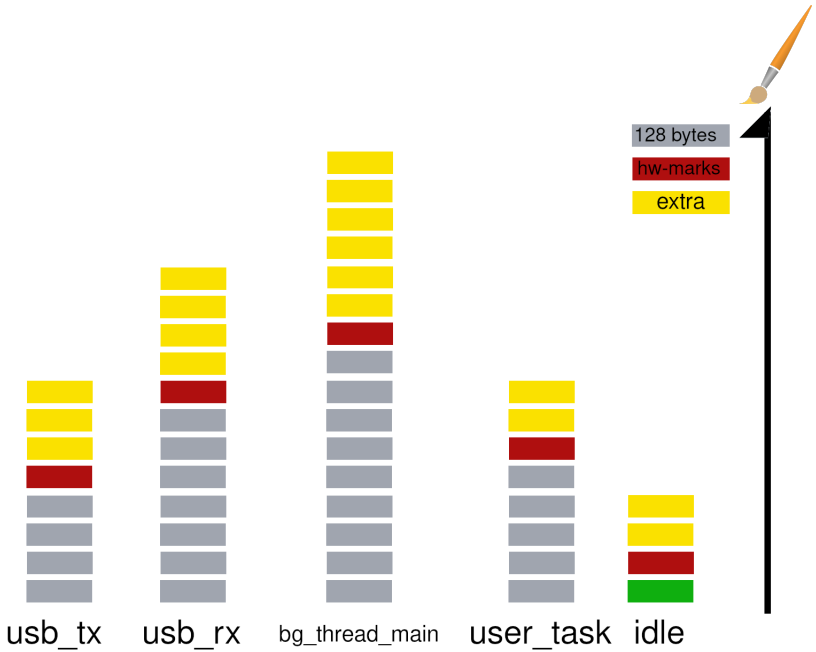
In zephyr part of the Thread analyzer or in FreeRTOS as Highwarter marks...

Stack on init

```
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
```

Stack after running

```
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
AAAAD8EF421234DD9
9000AADFEEFFF1234
33FFE096723945678
FFDA98BCD543FC998
799FFADBC45602CDA
```



...running and trigger events to enter suspected largest call chain.

1. interrupt.memfault.com/blog/measuring-stack-usage

In zephyr we can tweak our user or also the default and driver stack threads then...

Optimizing for Footprint

Stack Sizes

Stack sizes of various system threads are specified generously to allow for usage in different scenarios on as many supported platforms as possible. You should start the optimization process by reviewing all stack sizes and adjusting them for your application:

`CONFIG_ISR_STACK_SIZE`

Set to 2048 by default

`CONFIG_MAIN_STACK_SIZE`

Set to 1024 by default

`CONFIG_IDLE_STACK_SIZE`

Set to 320 by default

`CONFIG_SYSTEM_WORKQUEUE_STACK_SIZE`

Set to 1024 by default

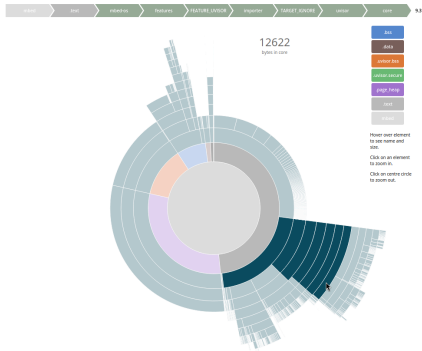
`CONFIG_PRIVILEGED_STACK_SIZE`

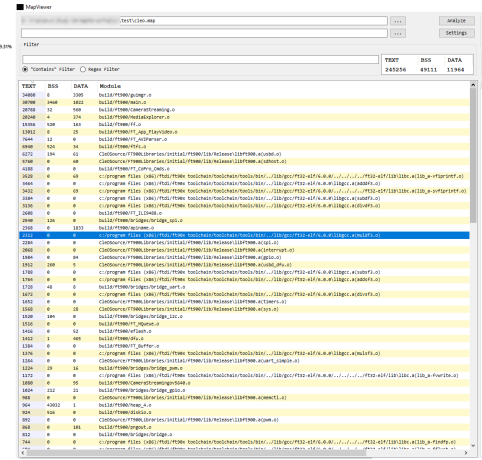
Set to 1024 by default, depends on userspace feature.

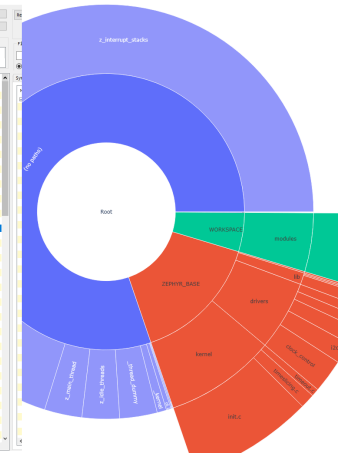
...think of networking, USB, BLE, ...

...but how far can we go and are we still in save margins as development continues?

Looking into MAP files

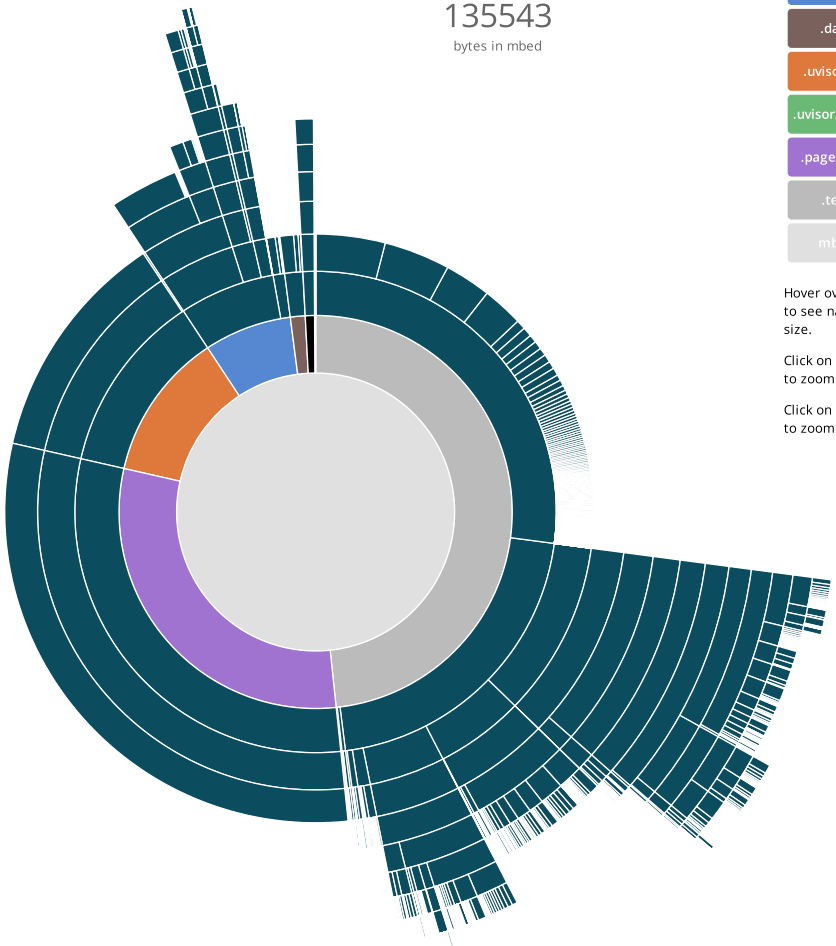






1. github.com/govind-mukundan/MapViewer
2. interrupt.memfault.com/blog/get-the-most-out-of-the-linker-map-file
3. docs.zephyrproject.org/latest/develop/optimizations/tools.html#build-targets-ram-plot-rom-plot
4. github.com/ARMmbed/mbed-os-linker-report
5. github.com/eleciawhite/MapFiles/blob/main/BuriedTreasureMapFiles_slides_ewhite.pdf
6. github.com/bmwcarit/Emma/tree/master

mbed 100%



.bss
.data
.uvisor.bss
.uvisor.secure
.page_heap
.text
mbed

Hover over element to see name and size.

Click on an element to zoom in.

Click on centre circle to zoom out.

Looking into ELF files - static stack usages with some help of GCC

ELF files are a one stop shop to get

- symbol names, addresses and sizes for variables and functions
- static calls between functions

... although parsing can be slow, gcc offers some info directly like (1)

- `-fstack-usage` → the size of a function on the stack
- `-fcallgraph-info` → all static calls of a function

...and some information we can not consider without extra info like

- dynamic calls
- which functions in an RTOS are thread entrypoints on their own stack

---

1. https://www.adacore.com/uploads/technical-papers/Stack_Analysis.pdf
2. gcc -fstack-usage → generates .su files - in Kconfig enabled by CONFIG_STACK_USAGE=y
3. west build -t puncover

Puncover

Folders

Name	Remarks	Code	Static
<unknown>		1,594	444
home		53,753	49,702
∑ **over all (2 folders, 0 files)**		**55,347**	**50,146**

Show all symbols | Analyze text snippet

Understand RAM and ROM usage of your application (over time)

Tracking Firmware Size Metrics

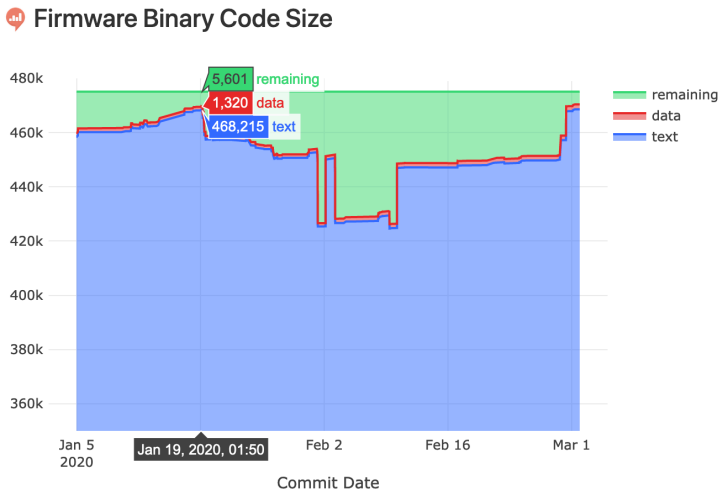
One nice example to set up firmware size tracking in CI pipeline (1).

Includes most high-level info, but we...

- need to setup a database
- miss more precice blames
- do not have data of thread stack data
- ...

→ So there is room to improve :)

1. https://interrupt.memfault.com/blog/code-size-deltas



	.text	.data	.bss	.text Delta	.data Delta	.bss Delta	Message
5ec6568c8327f7bb275d0b998196	146,272	4,124	41,807	-140	-96	0	net: purge NET_STACK ar APIs
7af2b822aebcf7fc7016df88348	146,412	4,220	41,807	66	0	72	kernel: add k_thread_join
ɔcc63d3bb06476ad63f99453d069	146,346	4,220	41,735	8	0	0	net: shell: Handle ENETU IPv4 ping
ɔ49f9dbe27b380a0fcfbc3d59db	146,338	4,220	41,735	-8	0	0	console: uart_console: Us static functions
l435aa5e86315aae25931e490a3d	146,346	4,220	41,735	8	0	0	kernel: fatal: check for es inspecting nested IRQ
f2213d8f2287d2891ebbe1fd2c6	146,338	4,220	41,735	4	0	0	arch: arm: aarch32: fix z_ implementation
7faf95e476c0fd002ec3c9ed04c	146,334	4,220	41,735	4	0	0	scripts: net: Enhance erro
l68f89609036522bb45992fa2124	146,330	4,220	41,735	16	0	0	shell: use correct data ty variables
1fd44f8eb0f38c2774a97bf97d9	146,314	4,220	41,735	-64	0	0	k fo sum

Motivation - Proposals where we can improve existing tools for application development

1. Extensions to puncover
    1. Define thread entries for reports and warnings+errors in CI
    2. Define dynamic calls manually
    3. Symbol and analysis exports to file
2. Independent tool to visualize and compare puncover exports - pexplorer
    1. Compare two builds - code reviews, learning and beyond
    2. More graphics, more intuitivity

Motivation - Ideas where we can improve for RTOS development

daddad

daddad

Motivation - Ideas where we can improve for RTOS development

Motivation - Ideas where we can improve for RTOS development

Compare two builds

daddad

More graphics, more intuitivity

daddad