

**Technnolultion**

The image features the word "Technnolultion" in a bold, black, sans-serif font. The letters are three-dimensional, casting soft shadows on the surface below. The text is positioned diagonally, starting from the bottom left and moving towards the top right. The background is split into two main colors: a bright yellow on the left and a white on the right. A black, jagged, arrow-like shape points from the right side towards the word, partially overlapping the white background.

# Technolution

An Open Source Board for NuttX  
Dave Marples

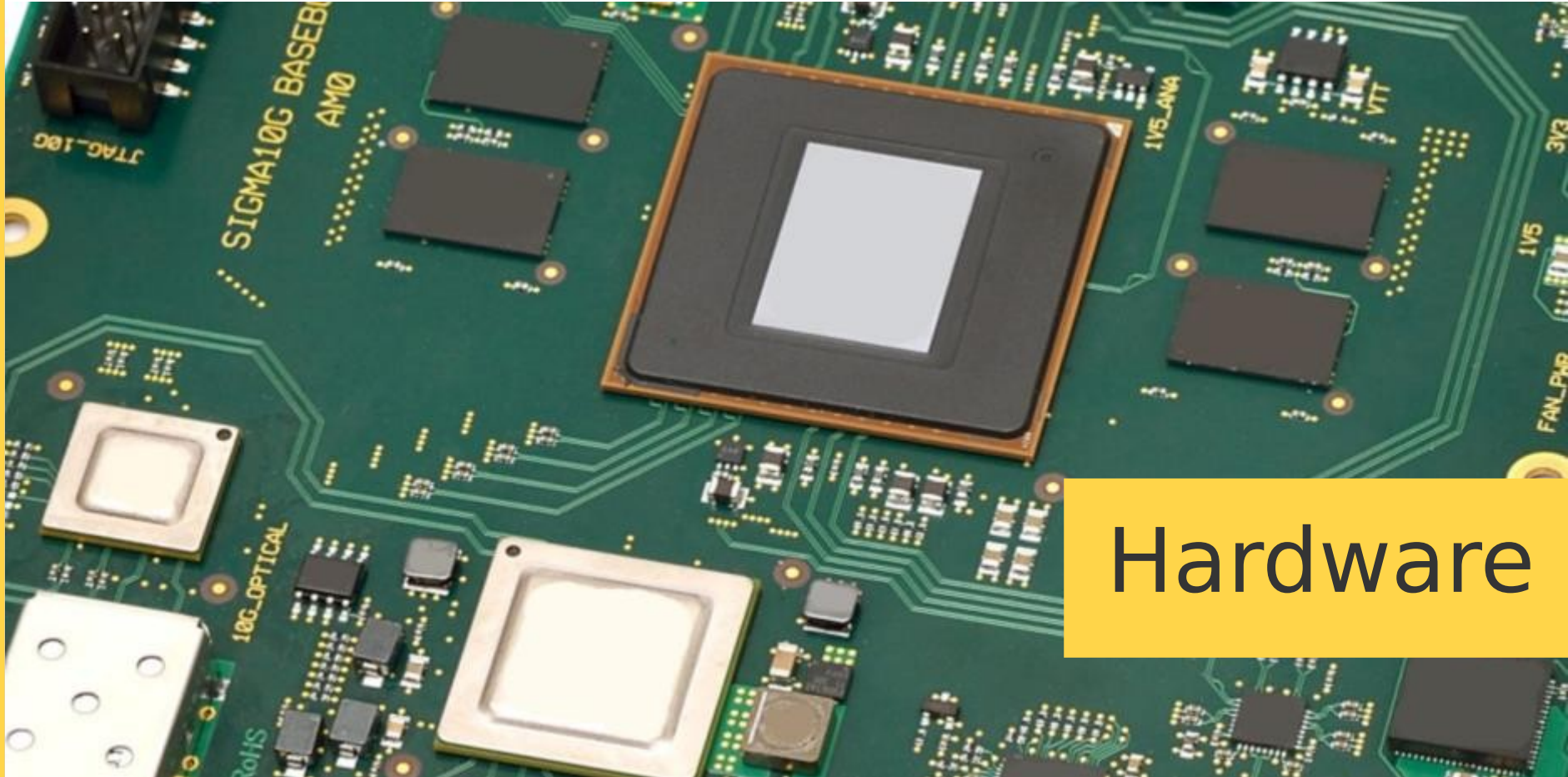
- Why we need a development setup
- Desirable requirements
- Actions to address those requirements
  - Hardware
  - Software
  - Debug and Systems Support
- Examples of the tools we can use
- Next steps

- Lead times on product development are becoming **too compressed** to allow development from scratch
- Hardware and software are **too complex** for a single product development to carry all of the risk
- We want to **retain** and **reuse** what we know from one development to the next
- Think of it like part-baked bread... all the ingredients are there and mixed, there's just the final cooking to be done.

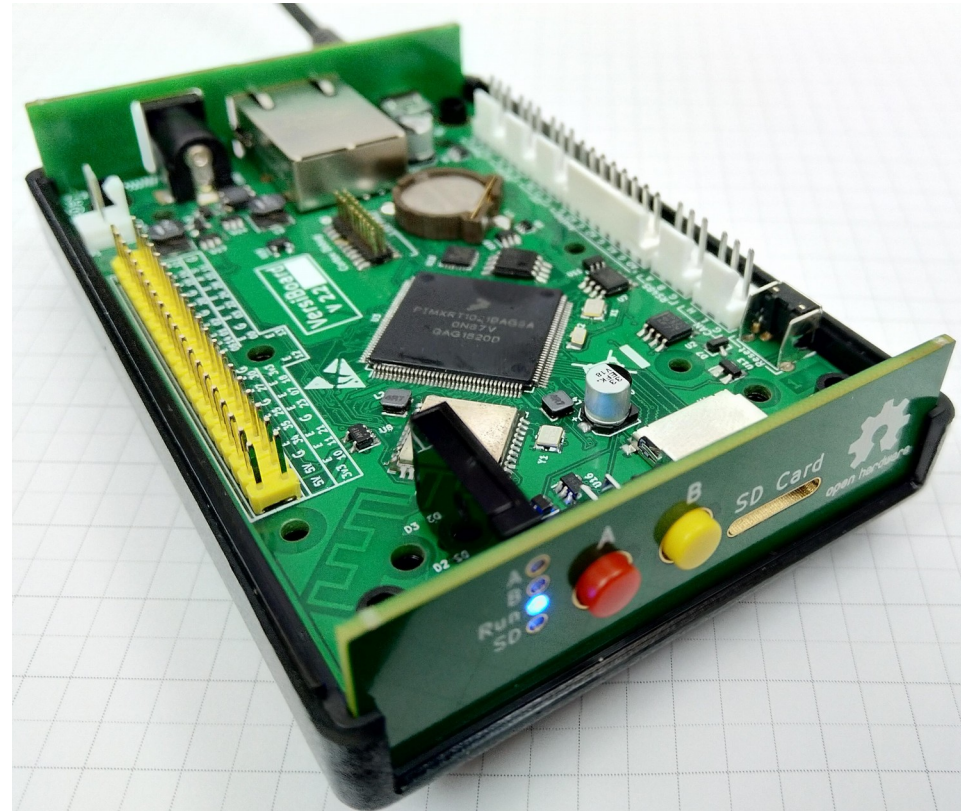
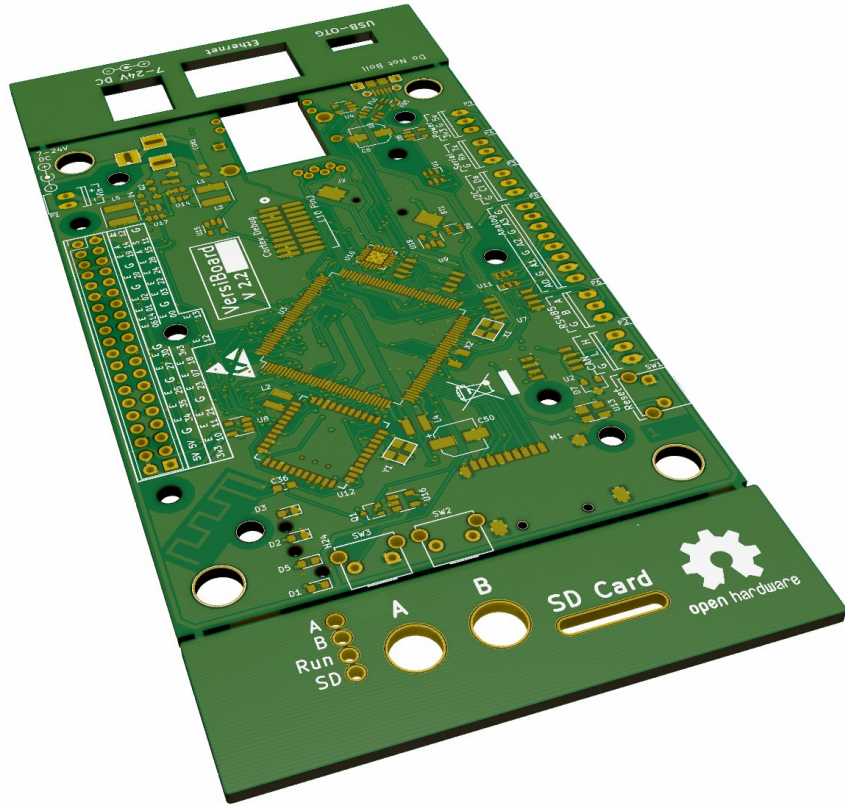


- **Hardware:** Find wide families of devices that have good abilities and common characteristics. Build prototypes, encourage open use and generally ‘learn how to bake’ using them.
- **Software:** Find wide families of software that have flexible and well documented structures. Build prototypes, encourage open use, and more baking.
- **Tooling:** Build, or buy, best of breed open tooling that minimises the effort needed to design, build and qualify any one specific product





Hardware









Product	CPU	DSP	Package	Memory	2D acceleration	LCD	CSI	USB with PHY	Ethernet	CAN	Quad ENC/ Quad Timer/ FlexPWM
<b>i.MX RT1064 &gt;</b>	Cortex-M7 @600 MHz	-	196 BGA	4MB Flash 1MB SRAM 32 kB I-cache 32 KC D-cache	PxP	8/16/24-bit Parallel	8/10/16-bit Parallel	OTG, HS/FS x2	2x 10/100	2x FlexCAN, 1x CANFD	4/4/4
<b>i.MX RT1060 &gt;</b>	Cortex-M7 @600 MHz	-	196 BGA	1MB SRAM 32 kB I-cache 32 kB D-cache	PxP	8/16/24-bit Parallel	8/10/16-bit Parallel	OTG, HS/FS x2	2x 10/100	2x FlexCAN, 1x CANFD	4/4/4
<b>i.MX RT106A &gt;</b>	Cortex-M7 @600 MHz	AVS Software	196 BGA	1MB SRAM 32 kB I-cache 32 kB D-cache	PxP	8/16/24-bit Parallel	8/10/16-bit Parallel	OTG, HS/FS x2	2x 10/100	2x FlexCAN, 1x CANFD	4/4/4
<b>i.MX RT1050 &gt;</b>	Cortex-M7 @600 MHz	-	196 BGA	512 kB SRAM 32 kB I-cache 32 kB D-cache	PxP	8/16/24-bit Parallel	8/10/16-bit Parallel	OTG, HS/FS x2	1x 10/100	2x FlexCAN	4/4/4
<b>i.MX RT1020 &gt;</b>	Cortex-M7 @500 MHz	-	100 LQFP 144 LQFP	256 kB SRAM 16 kB I-cache 16 kB D-cache	-	-	-	OTG, HS/FS x1	1x 10/100	2x FlexCAN	2/2/2
<b>i.MX RT1015 &gt;</b>	Cortex-M7 @500 MHz	-	100 LQFP	128 kB SRAM 16 kB I-cache 16 kB D-cache	-	-	-	OTG, HS/FS x1	-	-	1/1/1
<b>i.MX RT1010 &gt;</b>	Cortex-M7 @500 MHz	-	80 LQFP	128 kB SRAM 16 kB I-cache 8 kB D-cache	-	-	-	OTG, HS/FS x1	-	-	0/0/1



Software





Tooling...



- Starting point; GCC, GDB

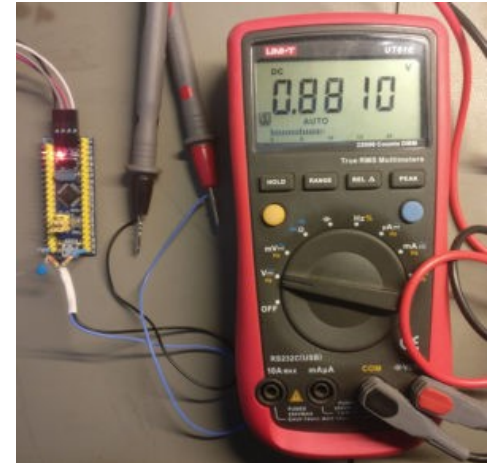
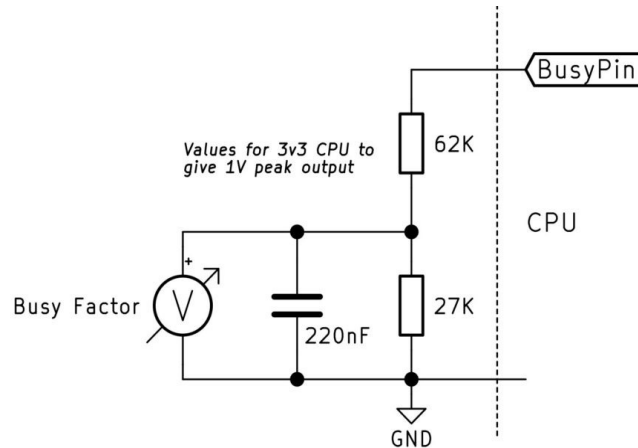
...but what about gprof, gcov, top etc? Its not enough to be able to write and deploy good code, we need to be able to verify and validate its correct operation too.

Visualization and instrumentation is as important for writing good code as the compiler and the OS on which it runs

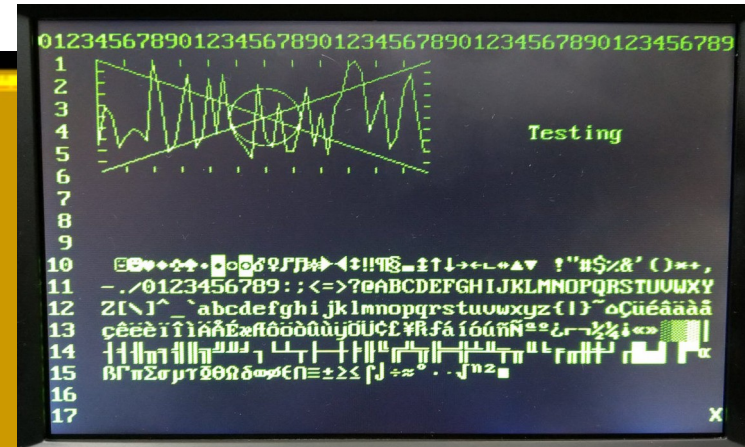
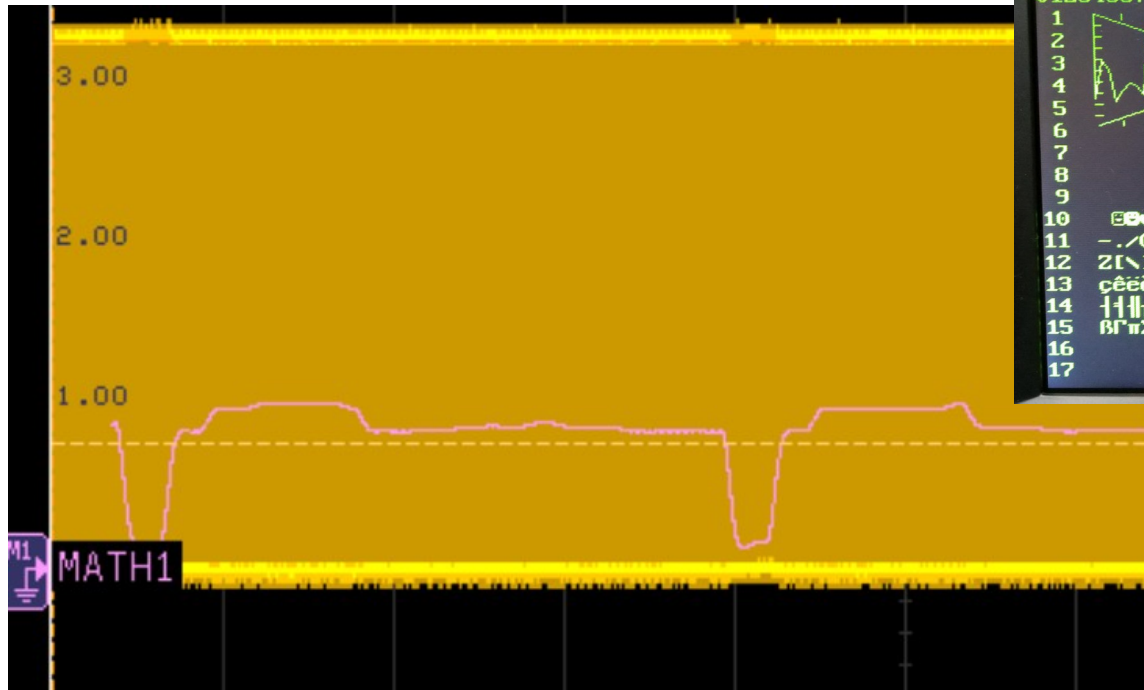
# Time honored method (i.e. from my day) of figuring out whats going on

```
#define SETUP_BUSY GPIOB->CRH=((GPIOB->CRH)&0xFFF0FFFF)|0x30000
#define AM_IDLE    GPIOB->BRR=(1<<12)
#define AM_BUSY    GPIOB->BSRR=(1<<12)
```

```
SETUP_BUSY;
while (1)
{
    flagSet = flag_get();
    if (!flagSet)
    {
        AM_IDLE;
        __WFI();
        AM_BUSY;
    }
    else
    {
        <Do things>
    }
}
```



Use the low pass filter on your scope to watch the behavior of the system in realtime...



- Another Traditional method but it needs program support and dedicated hardware
- Output is easy, input demands polling or interrupts
- Approx. 11500 bytes/sec.

```
void txDbgString(char *s)
{
    while (*s) {
        while (!USART1->SR&USART_SR_TXE);
        USART1->DR=*s++;
    }
}
```

```
int rxDbgChr(char *r)
{
    if (!USART1->SR&USART_SR_RXNE)
        return 0;
    *r=USART1->DR;
    return 1;
}
```



- Comes for free with JTAG or SWD with most probes
- Provides 'standard' file semantics, including stdio - great for test cases!
- Relatively slow, around 1200 bytes/sec

**Big Warning!!**

**Wrecks Realtime Semantics!**

```
IN Makefile;
```

```
#ifdef DEBUG
```

```
LINKOPTIONS+=--specs=rdimon.specs
```

```
else
```

```
LINKOPTIONS+=-lnosys
```

```
#endif
```

```
extern void initialise_monitor_handles(void); /* prototype */
```

```
int main( void )
```

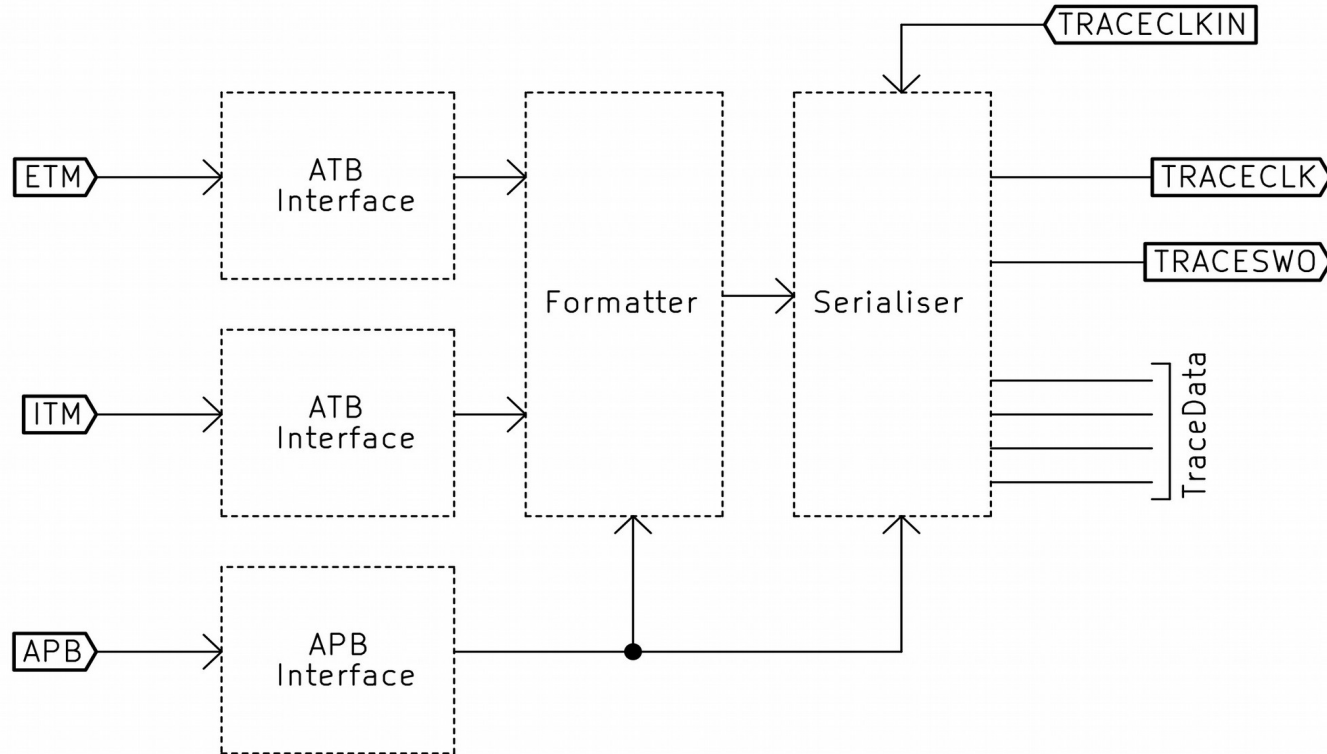
```
initialise_monitor_handles();
```

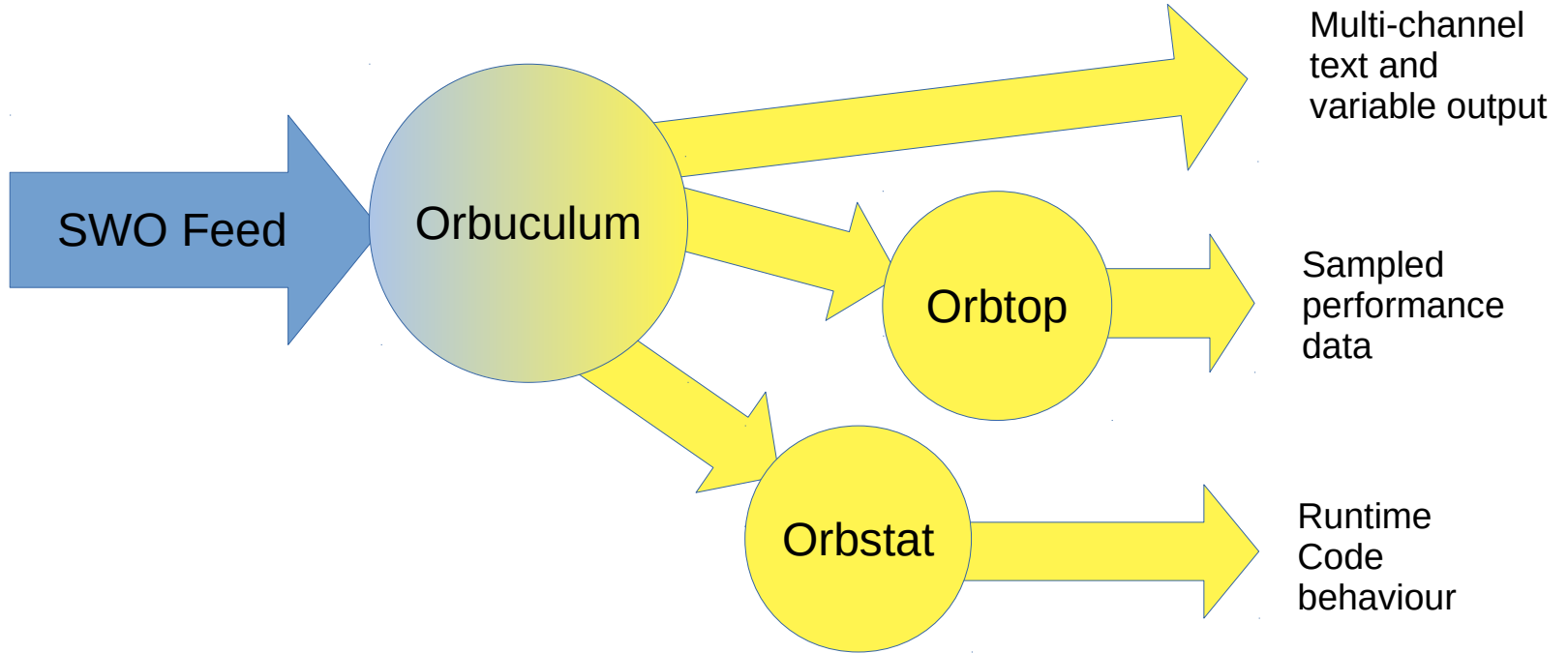
```
fprintf(stdout, "Hello, good evening and welcome\n");
```

```
...  
}
```

```
>telnet localhost 2333  
Hello, good evening and welcome  
_
```

- Comes for free with SWD with most probes – not available via JTAG (re-uses TDO pin)
- Provides lots of functionality;
  - 32 Distinct debug channels
  - Hardware tracing and sampling
  - Debug Watch and Traps
  - Time stamps
- Approx 111K bytes/sec
- Packetised format, needs to be depacketised to be useful (in the general case).







orbcode / orbuculum

Unwatch 12
★ Unstar 63
🍴 Fork 11

<> Code
🔔 Issues 3
🔀 Pull requests 5
📁 Projects 0
📖 Wiki
🛡 Security
📊 Insights
⚙ Settings

Cortex M SWO SWV Demux and Postprocess Edit

Manage topics

📄 129 commits
🌿 2 branches
📦 0 releases
👤 4 contributors
📄 GPL-3.0

Branch: master ▾
New pull request
Create new file
Upload files
Find File
Clone or download ▾

mubes Merge branch 'master' of https://github.com/mubes/orbuculum	Latest commit feb3f2d on 26 Feb 2018
📁 Docs	Addition of SEGGER support <span style="float: right;">2 years ago</span>
📁 Inc	Updates to ortrace to use high speed spi host side interface and sup... <span style="float: right;">2 years ago</span>
📁 Src	fix build without FPGA support <span style="float: right;">last year</span>
📁 Support	ITM Protocol simplifications and BMP ITM speedup <span style="float: right;">2 years ago</span>
📁 Tools/git_hash_to_c	Complete but lightly tested version <span style="float: right;">2 years ago</span>
📁 config	Modifications to support build and operation on Linux <span style="float: right;">2 years ago</span>
📁 ortrace	Integration of SUMP2 functionality <span style="float: right;">2 years ago</span>
📁 sump2	Integration of SUMP2 functionality <span style="float: right;">2 years ago</span>
📄 .gitignore	Addition of orbcat utility (and bugfixes to network streaming) <span style="float: right;">2 years ago</span>
📄 CONTRIBUTORS	Reversion of uart changes and parameter setting in toplevel <span style="float: right;">2 years ago</span>
📄 COPYING	Initial Commit <span style="float: right;">2 years ago</span>
📄 Makefile	fix build without FPGA support <span style="float: right;">last year</span>
📄 README.md	Merge branch 'master' of https://github.com/mubes/orbuculum <span style="float: right;">last year</span>

📄 README.md ✎

## Orbuculum - ARM Cortex Debug Output Processing Tools

---

*Stop press: Orbuculum now has an active Gitter channel at <https://gitter.im/orbcode/orbuculum> ... come join the discussion.*

```
static __INLINE uint32_t ITM_SendChar (uint32_t c, uint32_t ch)
{
    if ((CoreDebug->DEMCR & CoreDebug_DEMCR_TRCENA_Msk) && /* Trace enabled */
        (ITM->TCR & ITM_TCR_ITMENA_Msk) && /* ITM enabled */
        (ITM->TER & (1ul << c) )) /* ITM Port c enabled */
    {
        While (0 == ITM->PORT[c].u32); /* Port available? */
        ITM->PORT[c].u8 = (uint8_t) ch; /* Write data */
    }
    return (ch);
}

void sendString(uint32_t ch, char *s)
{
    while (*s) ITM_SendChar(*s++,ch);
}
```

```
>orbuclum -b swo/ -c 0,debug,"%c" -c
1,clientEvents,"%c" -c 2,Actions,"%c" -c 3,Z,"Z=%d\n" -
c 4,Temperature,"Temp=%d\n"
>ls
swo/
  debug
  clientEvents
  Actions
  Z
  Temperature
  hwevent
>_
```

```
sendString(DEBUG_CHANNEL, "Debug Event Happened");
```

```
>cat debug  
Debug Event Happened  
_
```

```
sendUint32(Z_CHANNEL, 12345);
```

```
>cat Z  
Z=12345  
_
```

```
sendString(CLIENTEVENTS_CHANNEL, "Port Opened");
```

```
>orbcats -c 0,"%c" -c 1,"%c" -c  
3,"Z=%d\n"  
Debug Event Happened  
Z=12345  
Port Opened  
_
```

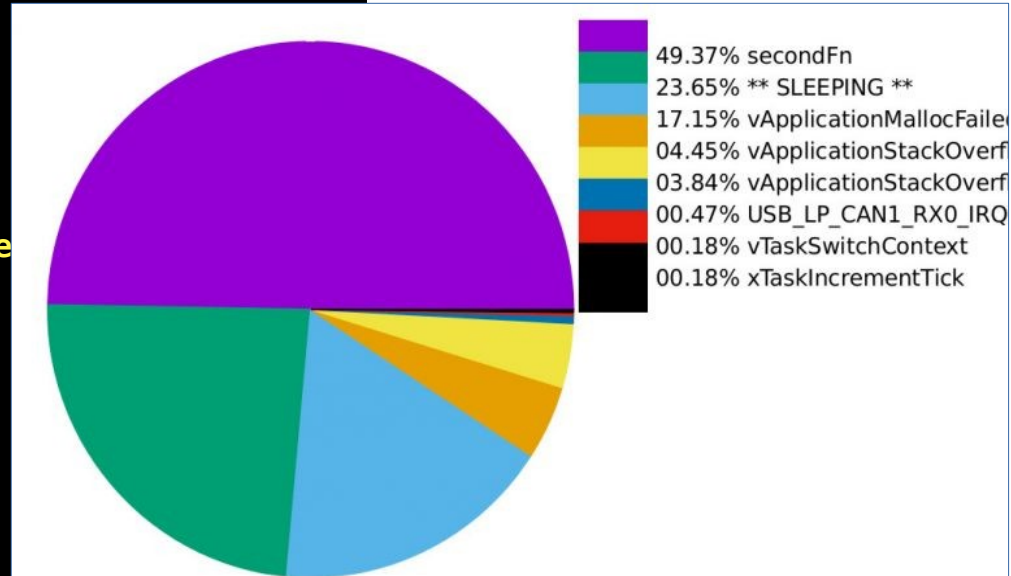
```
gdb>dwtTraceException 1  
_
```

```
>cat hwevent  
1,2,Resume,Thread  
1,989,Enter,SysTick  
1,6,Exit,SysTick  
1,1,Resume,Thread  
1,989,Enter,SysTick  
1,4,Exit,SysTick  
_
```

```
>Support/orbplot_top &
>orbttop -e ../STM32F103-skel/ofiles/firmware.elf -
dumpfile.out
```

```
97.90%      4308 ** Sleeping **
 1.25%       55 USB_LP_CAN1_RX0_IRQHandler
 0.20%        9 xTaskIncrementTick
 0.13%        6 Suspend
 0.09%        4 SysTick_Handler
 0.06%        3 Resume
 0.06%        3 __WFI
 0.04%        2 vTaskSwitchContext
 0.04%        2 TIM_Cmd
 0.02%        1 prvAddCurrentTaskToDelaye
 0.02%        1 xTaskResumeAll
 0.02%        1 vTaskDelay
 0.02%        1 PendSV_Handler
 0.02%        1 __ISB
 0.02%        1 taskIn
 0.02%        1 statsGetRTVal
 0.02%        1 taskOut
```

-----  
4400 Samples



```

__attribute__((no_instrument_function)) void __cyg_profile_func_enter (void *this_fn, void *call_site)
{
    if (!(ITM->TER&(1<<TRACE_CHANNEL)))
        return;
    uint32_t oldIntStat=__get_PRIMASK();
    __disable_irq();

    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);
    ITM->PORT[TRACE_CHANNEL].u32 = ((*((uint32_t *)0xE0001004))&0x03FFFFFF)|0x40000000;

    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);
    ITM->PORT[TRACE_CHANNEL].u32 = (uint32_t)(call_site)&0xFFFFFFFF;
    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);
    ITM->PORT[TRACE_CHANNEL].u32 = (uint32_t)this_fn&0xFFFFFFFF;
    __set_PRIMASK(oldIntStat);
}

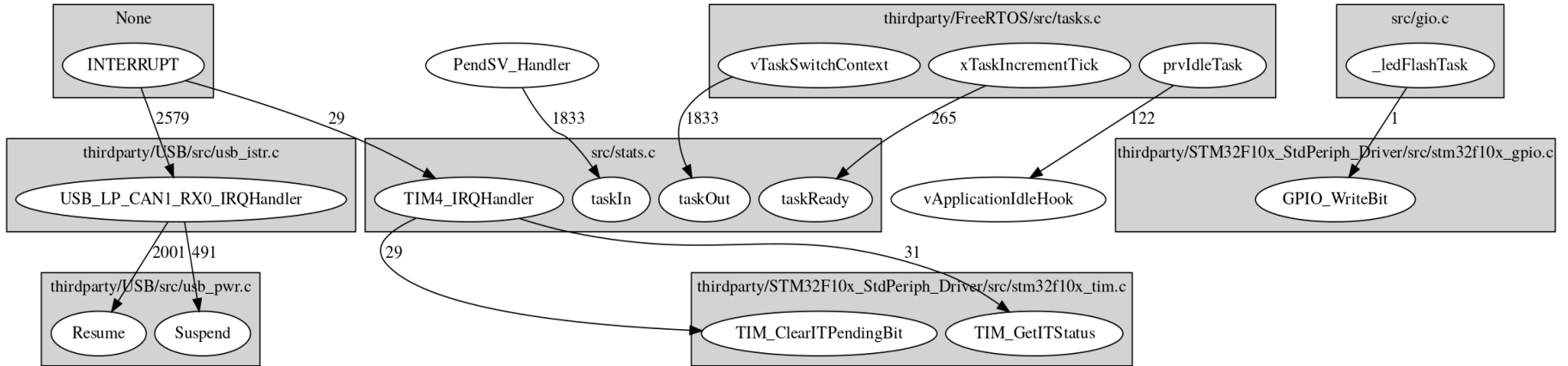
__attribute__((no_instrument_function)) void __cyg_profile_func_exit (void *this_fn, void *call_site)
{
    if (!(ITM->TER&(1<<TRACE_CHANNEL)))
        return;
    uint32_t oldIntStat=__get_PRIMASK();
    __disable_irq();
    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);

    ITM->PORT[TRACE_CHANNEL].u32 = ((*((uint32_t *)0xE0001004))&0x03FFFFFF)|0x50000000;
    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);
    ITM->PORT[TRACE_CHANNEL].u32 = (uint32_t)(call_site)&0xFFFFFFFF;
    while (ITM->PORT[TRACE_CHANNEL].u32 == 0);
    ITM->PORT[TRACE_CHANNEL].u32 = (uint32_t)this_fn&0xFFFFFFFF;
    __set_PRIMASK(oldIntStat);
}

```

```

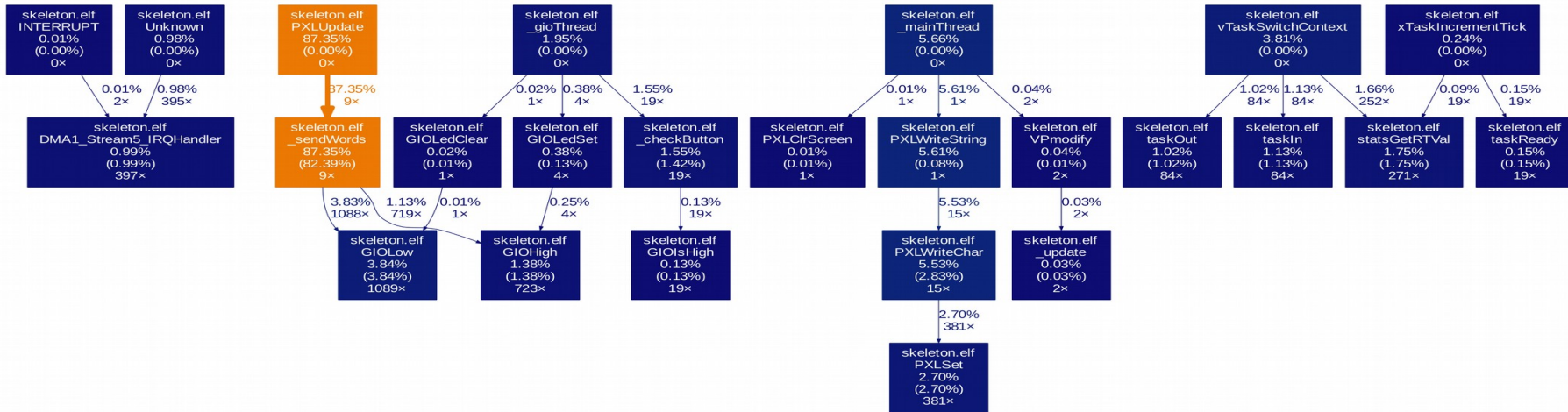
>orbstat -e firmware.elf -y
out.dot
>dot out.dot -o -Tpng > out.png
>evince out.png
_
    
```





```

>orbstat -e firmware.elf -z demo.out
>gprof2dot -f callgrind demo.out >
demo.dot
>dot demo.dot -o -Tpng > out.png
>evince out.png
    
```



```
>orbstat -e firmware.elf -z
demo.out
>kcachegrind demo.out
```

test — KCachegrind

File View Go Settings Help

Open Back Forward Up Relative Cycle Detection Relative to Parent Shorten Templates Processor Clock Cycles

Flat Profile

Search: (No Grouping)

Incl.	Self	Called	Function	Location
93.66	0.00	(0)	privileTask	firmware.elf: tasks.c
93.66	66.22	66	ApplicationIdleHook	firmware.elf: main.c
22.33	0.00	(0)	INTERRUPT	firmware.elf: None
21.80	13.23	1 276	USB_LP_CAN1_RX0_IRQ...	firmware.elf: usb_lstr.c
6.87	0.00	(0)	PendSV_Handler	firmware.elf: port.c
6.87	6.87	902	taskIn	firmware.elf: stats.c
6.85	6.85	981	Resume	firmware.elf: usb_pwr.c
4.08	0.00	(0)	vTaskSwitchContext	firmware.elf: tasks.c
4.08	4.08	902	taskOut	firmware.elf: stats.c
1.74	1.74	243	Suspend	firmware.elf: usb_pwr.c
0.54	0.11	15	TIM4_IRQHandler	firmware.elf: gpio.c
0.47	0.00	(0)	xTaskIncrementTick	firmware.elf: tasks.c
0.47	0.47	129	taskReady	firmware.elf: stats.c
0.11	0.11	15	TIM_ClearITPendingBit	firmware.elf: stm32f10x_tim.c
0.11	0.11	15	TIM_GetITStatus	firmware.elf: stm32f10x_tim.c
0.01	0.00	(0)	LedFlashTask	firmware.elf: gpio.c
0.01	0.01	1	GPIO_WriteBit	firmware.elf: stm32f10x_gpio.c

Types Callers All Callers Callee Map Source Code

```

76 0.31 (
77     /* Interrupt handler for TIM 4
78     ...
79     The time base for the run time stats is generated by the 16 bit timer 4.
80     ...
81     significant two bytes are given by the current TIM4 counter value. Care
82     must be taken with data consistency when combining the two in case a timer
83     overflow occurs as the value is being read. */
84     if (TIM_GetITStatus( TIM4, TIM_IT_Update ) != RESET)
85     {
86         /* 15 calls to TIM_GetITStatus( firmware.elf: stm32f10x_tim.c)
87         {
88             uTIM4_OverflowCount++;
89             TIM_ClearITPendingBit( TIM4, TIM_IT_Update );
90         }
91     }
92     // =====

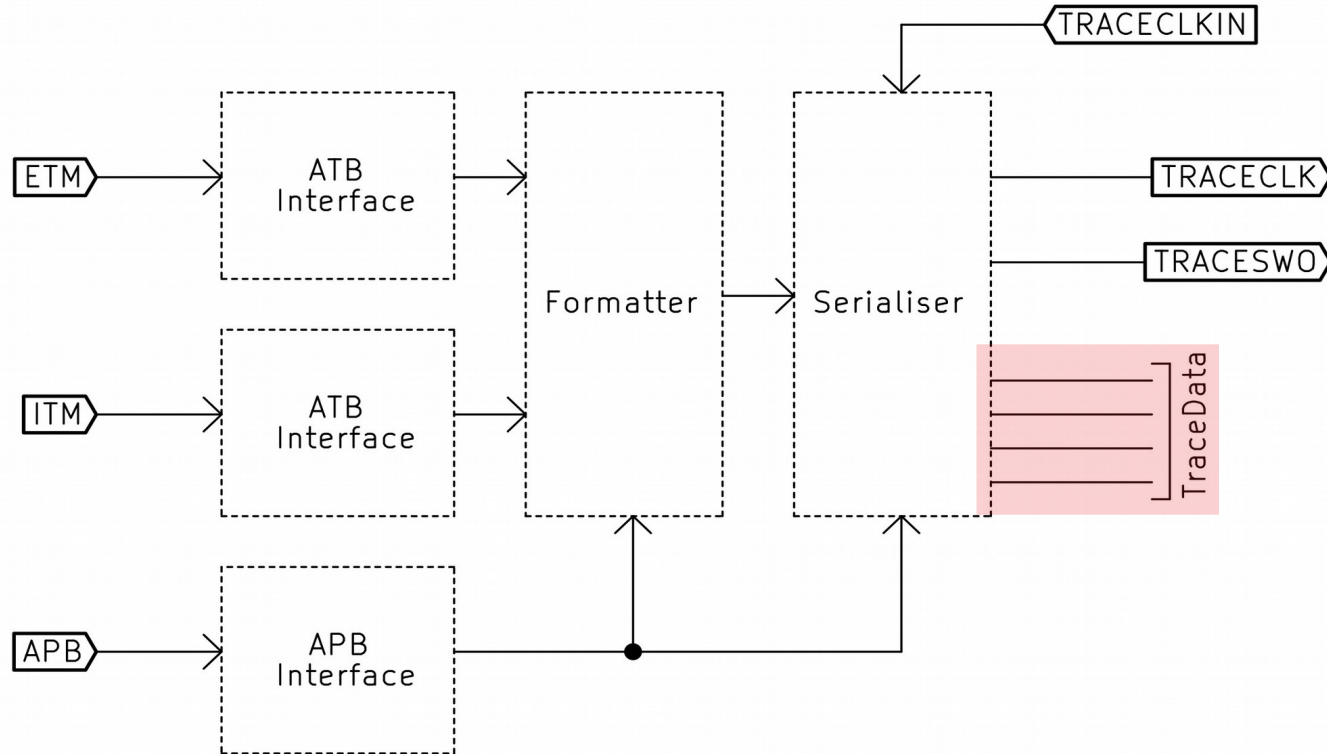
```

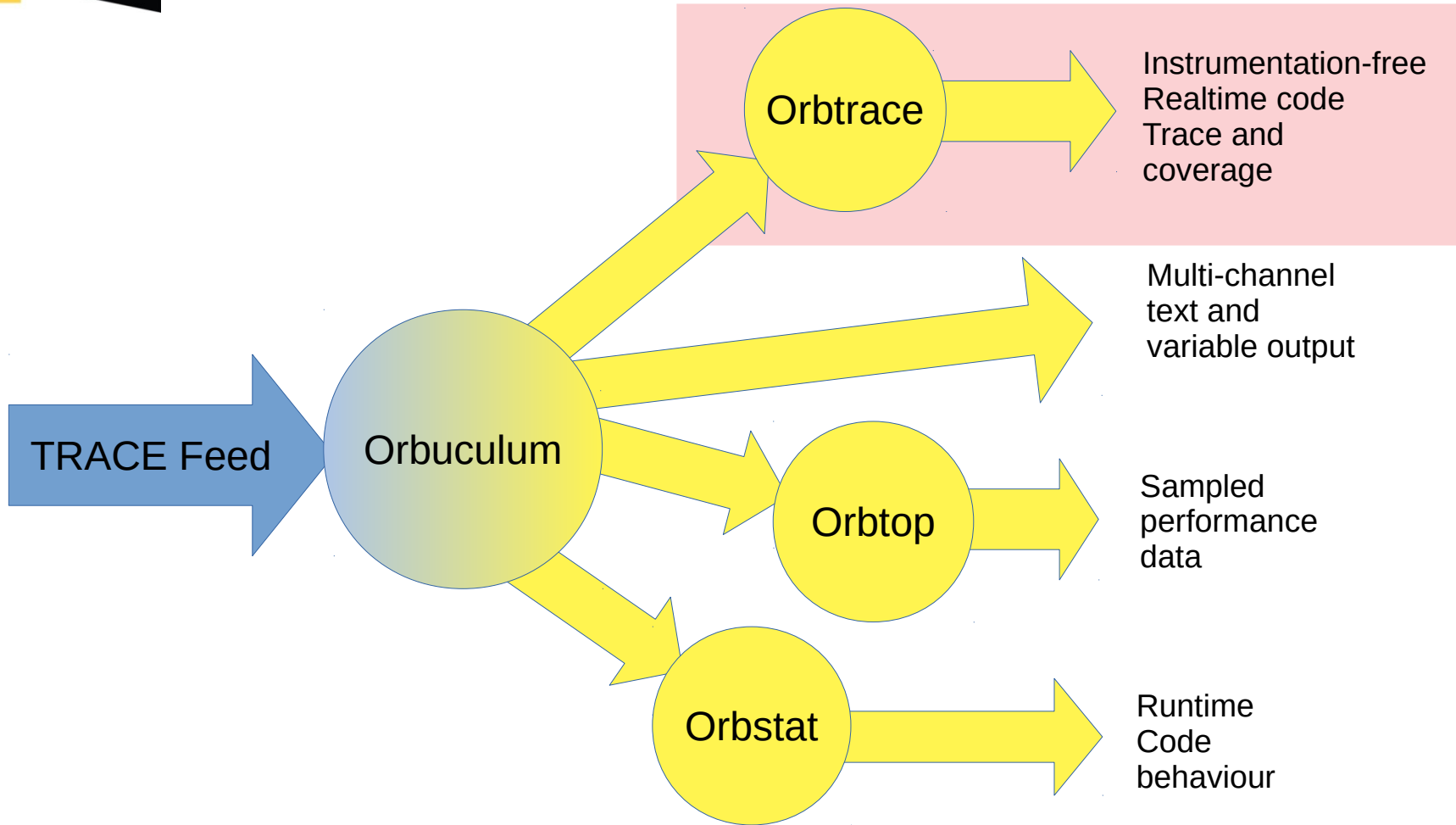
```

graph TD
    INTERRUPT[INTERRUPT  
0.54%] -- 15 x --> TIM4_IRQHandler[TIM4_IRQHandler  
0.54%]
    TIM4_IRQHandler -- 15 x --> TIM_GetITStatus[TIM_GetITStatus  
0.11%]
    TIM4_IRQHandler -- 15 x --> TIM_ClearITPendingBit[TIM_ClearITPendingBit  
0.11%]

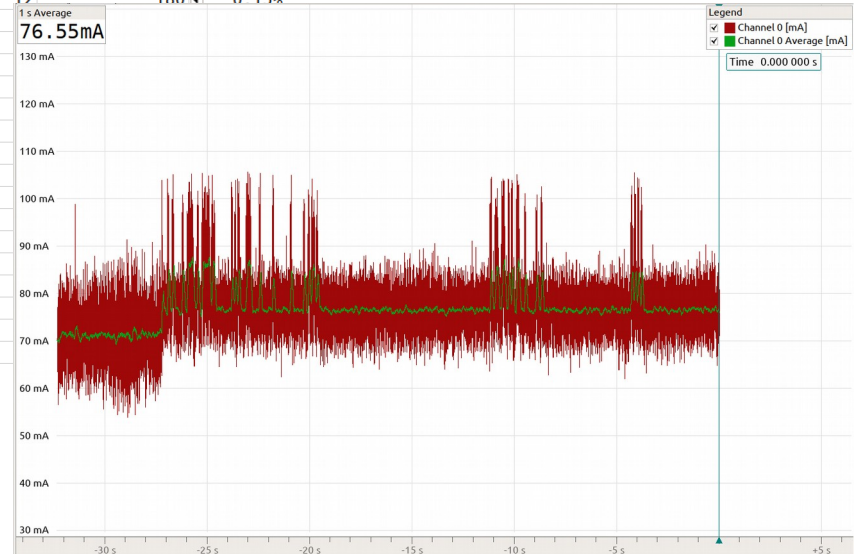
```

test [1] - Total Processor Clock Cycles Cost: 62 995 833





Source Coverage	Function	Inst. Coverage	Run Count	Fetch Count	Load
87.5% (14/16)	f __start	89.4% (42/47)	1	104393	87.26%
54.5% (12/22)	f up_enable_dcache	65.8% (48/73)	1	9549	7.98%
100.0% (11/11)	f modifyreg32	100.0% (48/48)	24	1152	0.96%
76.9% (20/26)	f imxrt_iomux_configur	88.9% (80/90)	12	954	0.80%
86.7% (13/15)	f imxrt_gpio_configperi	94.1% (48/51)	11	539	0.45%
100.0% (3/3)	f imxrt_periphclk_conf	100.0% (22/22)	24	528	0.44%
56.5% (13/23)	f imxrt_config_gpio	65.8% (52/79)	12	520	0.43%
30.0% (6/20)	f imxrt_pllsetup	41.2% (28/68)	1	418	0.35%
87.5% (7/8)	f imxrt_daisy_select	97.5% (39/40)	11	359	0.30%
95.3% (82/86)	f imxrt_clockconfig	98.5% (266/270)	1	270	0.23%
40.0% (2/5)	f imxrt_padmux_addres	65.2% (15/23)	12	192	0.16%
40.0% (2/5)	f imxrt_padctl_address	65.2% (15/23)	12	192	0.16%
75.0% (3/4)	f imxrt_padmux_map	77.8% (14/18)	12	180	0.15%
61.9% (13/21)	f imxrt_gpio_configinp	80.6% (58/72)	12	156	0.13%
90.9% (10/11)	f imxrt_fpuconfig	97.4% (37/38)	12	108	0.09%
77.8% (7/9)	f imxrt_gpio_configout	84.6% (33/39)	12	108	0.09%
60.0% (6/10)	f mpu_control	83.8% (31/37)	12	108	0.09%
62.5% (5/8)	f imxrt_gpio_setoutput	76.2% (32/42)	12	108	0.09%
80.0% (4/5)	f imxrt_gpio_dirin	96.9% (31/32)	12	108	0.09%
91.7% (11/12)	f imxrt_tcmenable	96.8% (30/31)	12	108	0.09%
100.0% (5/5)	f imxrt_gpio_dirout	100.0% (30/30)	12	108	0.09%
92.3% (12/13)	f imxrt_lowsetup	96.3% (26/27)	12	108	0.09%
90.0% (9/10)	f up_enable_icache	96.0% (24/25)	12	108	0.09%
100.0% (4/4)	f arm_clz	100.0% (13/13)	12	108	0.09%
100.0% (3/3)	f imxrt_gpio_select	100.0% (11/11)	12	108	0.09%
100.0% (5/5)	f imxrt_mpu_initialize	100.0% (10/10)	12	108	0.09%
100.0% (2/2)	f mpu_showtype	100.0% (6/6)	12	108	0.09%
66.7% (2/3)	f imxrt_utooled_initializ	85.7% (6/7)	12	108	0.09%
100.0% (3/3)	f imxrt_boardinitialize	100.0% (5/5)	12	108	0.09%



up_enable_dcache				9529	
41	60002D04	61BA	STK	R2, [R7, #24]	while (tmpways--);
40	60002D06	2B00	CMP	R3, #0	
39	60002D08	D1EC	BNE	0x60002CE4	
38	60002CE4	69BA	LDR	R2, [R7, #24]	sw = ((tmpways << wshift)   (sets << sshift));
37	60002CE6	68BB	LDR	R3, [R7, #8]	
36	60002CE8	FA02 F303	LSL.W	R3, R2, R3	
35	60002CEC	4619	MOV	R1, R3	
34	60002CEE	69FA	LDR	R2, [R7, #28]	
33	60002CF0	693B	LDR	R3, [R7, #16]	
32	60002CF2	FA02 F303	LSL.W	R3, R2, R3	
31	60002CF6	430B	ORRS	R3, R1	
30	60002CF8	607B	STR	R3, [R7, #4]	
29	60002CFA	4A16	LDR	R2, [PC, #0x58] ; [0x60002D54]	putreg32(sw, NVIC_DCISW);
28	60002CFC	687B	LDR	R3, [R7, #4]	
27	60002CFE	6013	STR	R3, [R2]	
26	60002D00	69BB	LDR	R3, [R7, #24]	while (tmpways--);
25	60002D02	1E5A	SUBS	R2, R3, #1	
24	60002D04	61BA	STR	R2, [R7, #24]	
23	60002D06	2B00	CMP	R3, #0	
22	60002D08	D1EC	BNE	0x60002CE4	
21	60002CE4	69BA	LDR	R2, [R7, #24]	sw = ((tmpways << wshift)   (sets << sshift));
20	60002CE6	68BB	LDR	R3, [R7, #8]	
19	60002CE8	FA02 F303	LSL.W	R3, R2, R3	
18	60002CEC	4619	MOV	R1, R3	
17	60002CEE	69FA	LDR	R2, [R7, #28]	
16	60002CF0	693B	LDR	R3, [R7, #16]	
15	60002CF2	FA02 F303	LSL.W	R3, R2, R3	
14	60002CF6	430B	ORRS	R3, R1	
13	60002CF8	607B	STR	R3, [R7, #4]	
12	60002CFA	4A16	LDR	R2, [PC, #0x58] ; [0x60002D54]	putreg32(sw, NVIC_DCISW);
11	60002CFC	687B	LDR	R3, [R7, #4]	
10	60002CFE	6013	STR	R3, [R2]	
9	60002D00	69BB	LDR	R3, [R7, #24]	while (tmpways--);
8	60002D02	1E5A	SUBS	R2, R3, #1	
7	60002D04	61BA	STR	R2, [R7, #24]	
6	60002D06	2B00	CMP	R3, #0	
5	60002D08	D1EC	BNE	0x60002CE4	
4	60002D0A	69FB	LDR	R3, [R7, #28]	while (sets--);
3	60002D0C	1E5A	SUBS	R2, R3, #1	
2	60002D0E	61FA	STR	R2, [R7, #28]	
1	60002D10	2B00	CMP	R3, #0	
0	60002D12	D1E5	BNE	0x60002CE0	
up_assert				1	
● PC	60004340	B580	PUSH	{R7, LR}	



- Full Profiling of code run
- Pre-trigger recording (I've crashed, how did that happen?)
- Non-intrusive, full speed, code path observation with no requirement for instrumentation
- Full speed software channel output
- Triggers

..more at

[www.shadetail.com](http://www.shadetail.com)

[www.github.com/mubes/orbuculum](http://www.github.com/mubes/orbuculum)

Technolution

Thanks for listening



**Technnolultion**

The image features the word "Technnolultion" in a bold, black, sans-serif font. The letters are three-dimensional, casting soft shadows on the surface below. The text is positioned diagonally, starting from the bottom left and moving towards the top right. The background is split into two main colors: a bright yellow on the left and a white on the right. A black, jagged, torn-paper-like shape separates the yellow and white areas, extending from the bottom right towards the center of the text.