
Technolution

Technolution

/ the right development

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

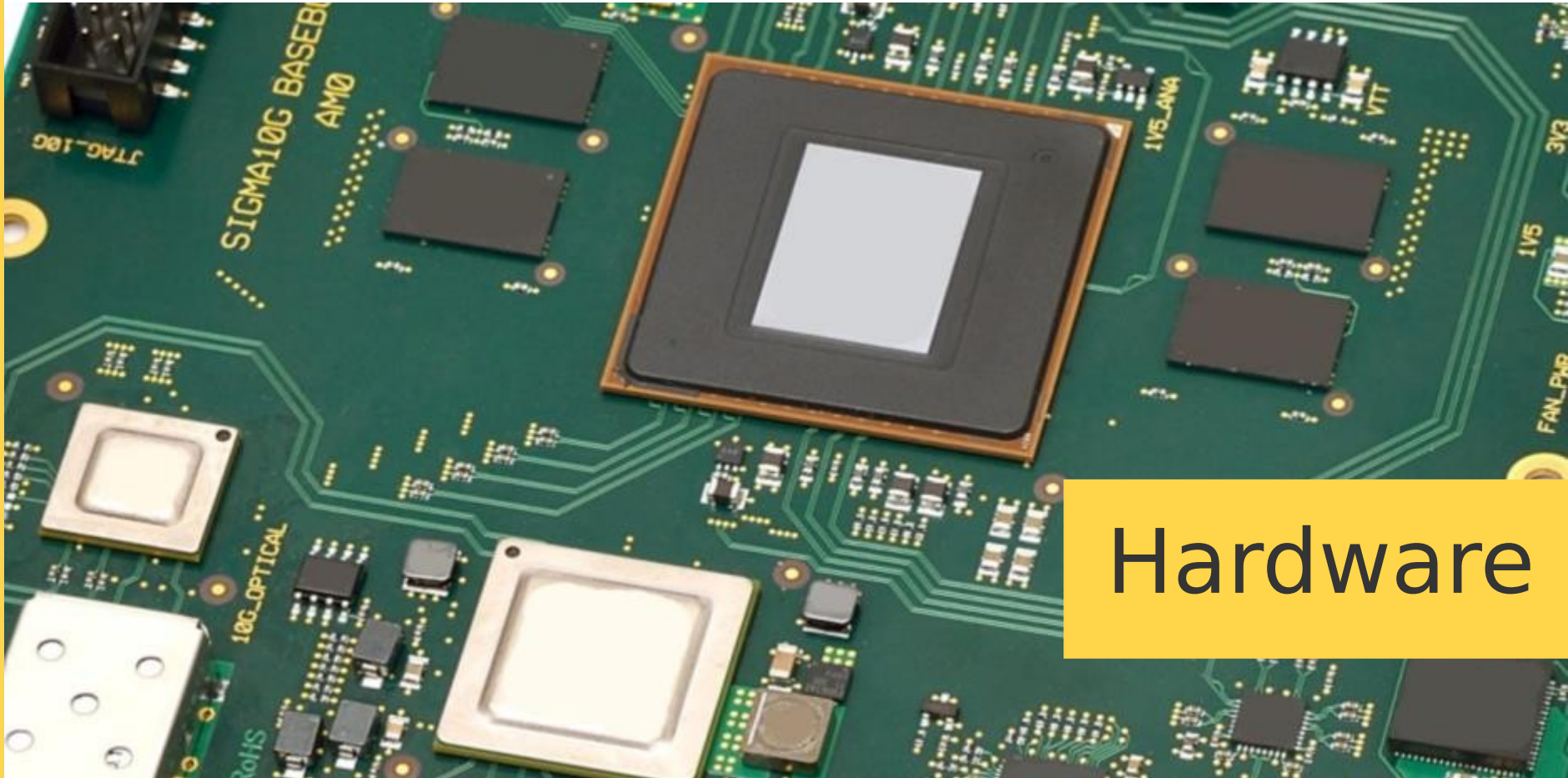
- Think of it like part-baked bread… all the ingredients are there and mixed, there's just the final cooking to be done.

- **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

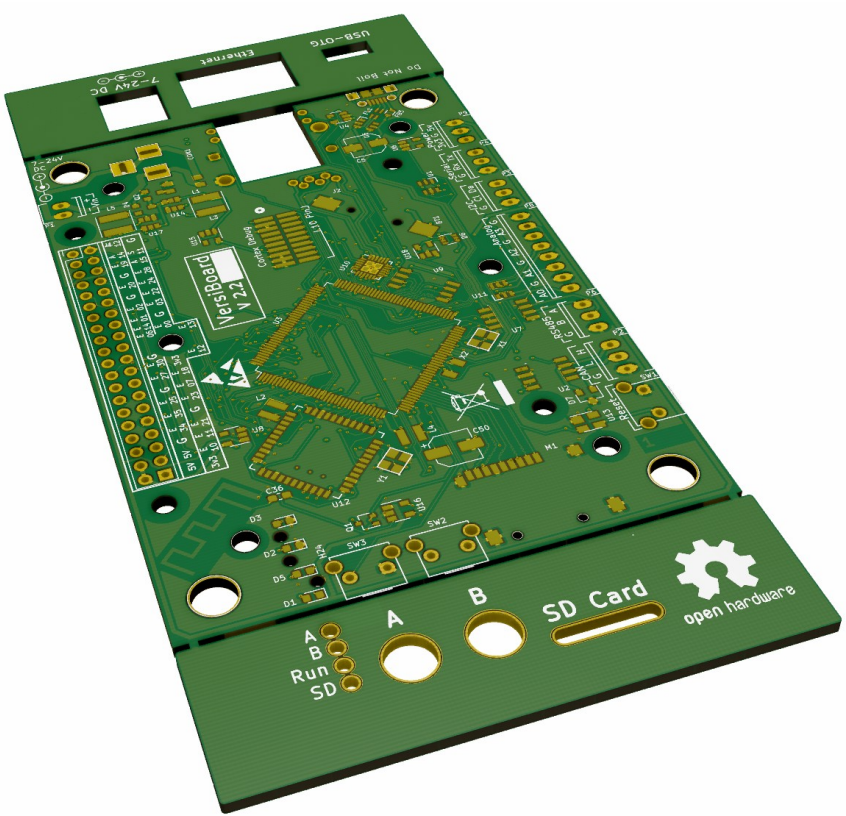
Technolution

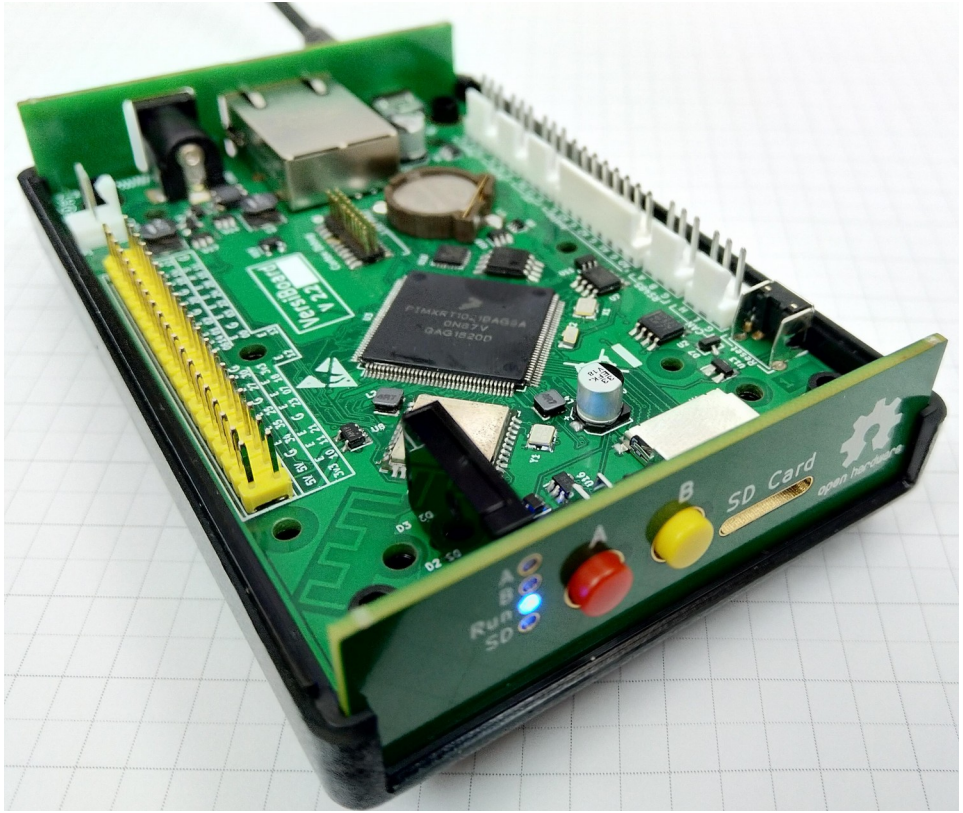


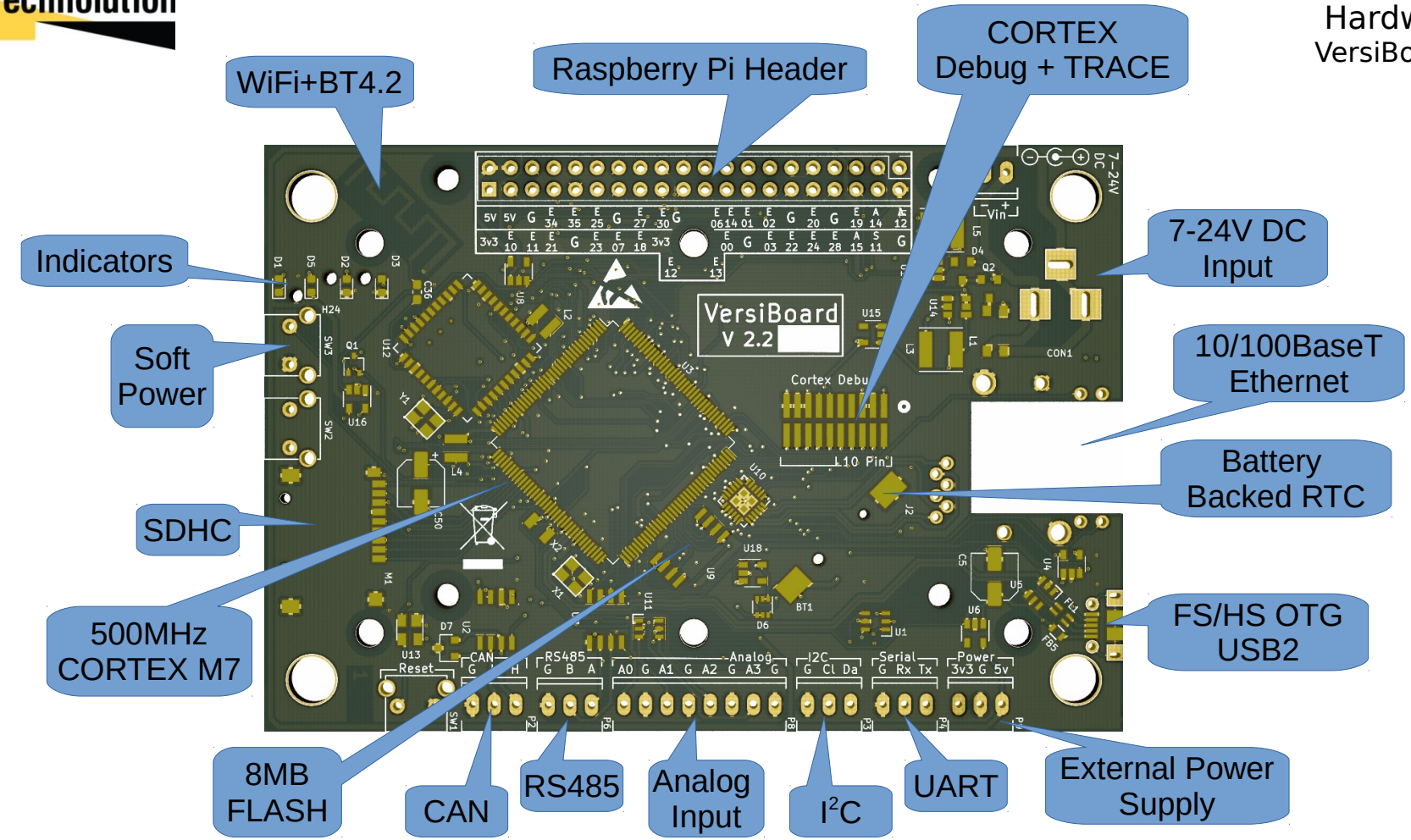
Hardware

Technolution

Hardware
VersiBoard2






WiFi+BT4.2
Raspberry Pi Header
CORTEX
Debug + TRACE
Indicators
7-24V DC
Input
Soft
Power
10/100BaseT
Ethernet
Battery
Backed RTC
SDHC
500MHz
CORTEX M7
FS/HS OTG
USB2
8MB
FLASH
CAN
RS485
Analog
Input
I2C
UART
External Power
Supply
VersiBoard
V 2.2
Cortex Debu
10 Pin
CAN
RS485
Analog
I2C
Serial
Power
Reset
7-24V
DC
Vin
CON1

Hardware
Why the imxrt?

Product	CPU	DSP	Package	Memory	2D acceleration	LCD	CSI	USB with PHY	Ethernet	CAN	Quad ENC/ Quad Timer/ FlexPWM
i.MX RT1064 ›	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
i.MX RT1060 ›	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
i.MX RT106A ›	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
i.MX RT1050 ›	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
i.MX RT1020 ›	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
i.MX RT1015 ›	Cortex-M7 @500 MHz	-	100 LQFP	128 kB SRAM 16 kB I-cache 16 kB D-cache	-	-	-	OTG, HS/FS x1	-	-	1/1/1
i.MX RT1010 ›	Cortex-M7 @500 MHz	-	80 LQFP	128 kB SRAM 16 kB I-cache 8 kB D-cache	-	-	-	OTG, HS/FS x1	-	-	0/0/1

Technolution

Software

Tooling...

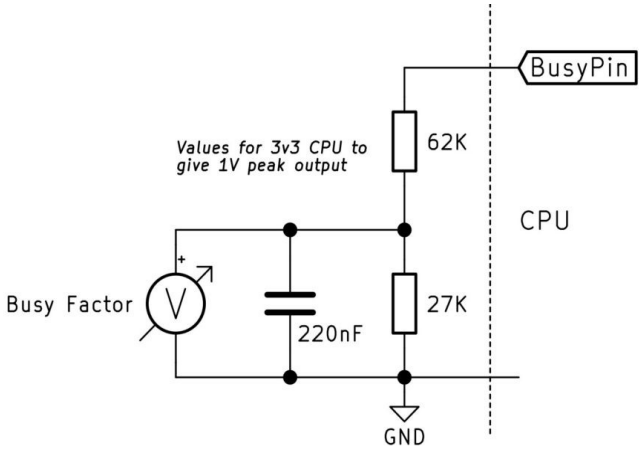
- Starting point; GCC, GDB

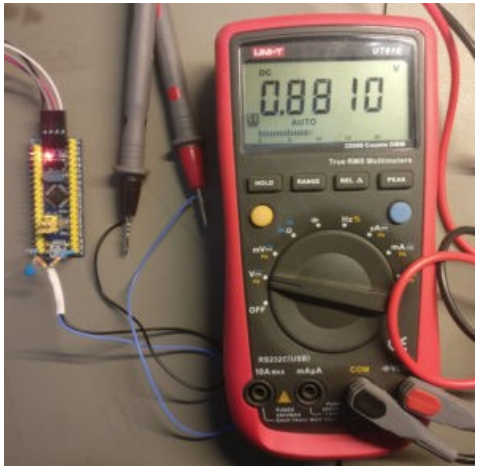
...but what about gprof, gcov, top etc? Its not enough to be able to write and deploy good code, we need to be able to verify and validate its correct operation too.

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

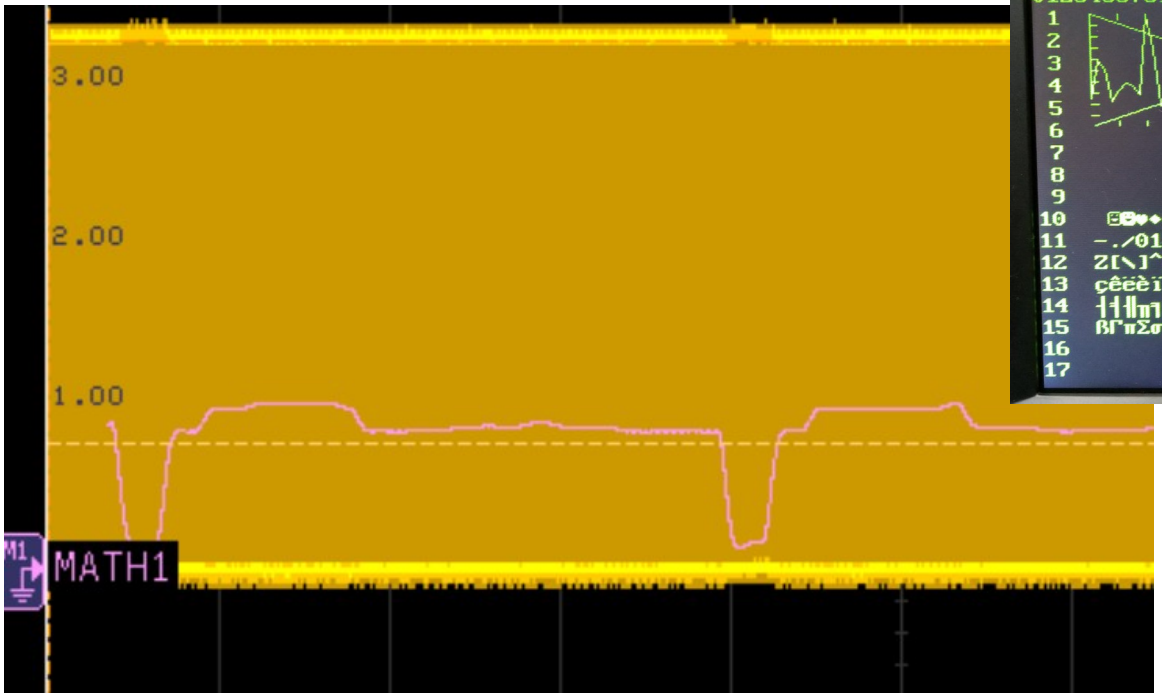
Time honored method (i.e. from my day) of figuring out whats going on

```
#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...



https://www.youtube.com/watch?v=5UFpp3ao460

- 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;
}
```

Semihosting...

- 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
_
```

Enter the SWO...

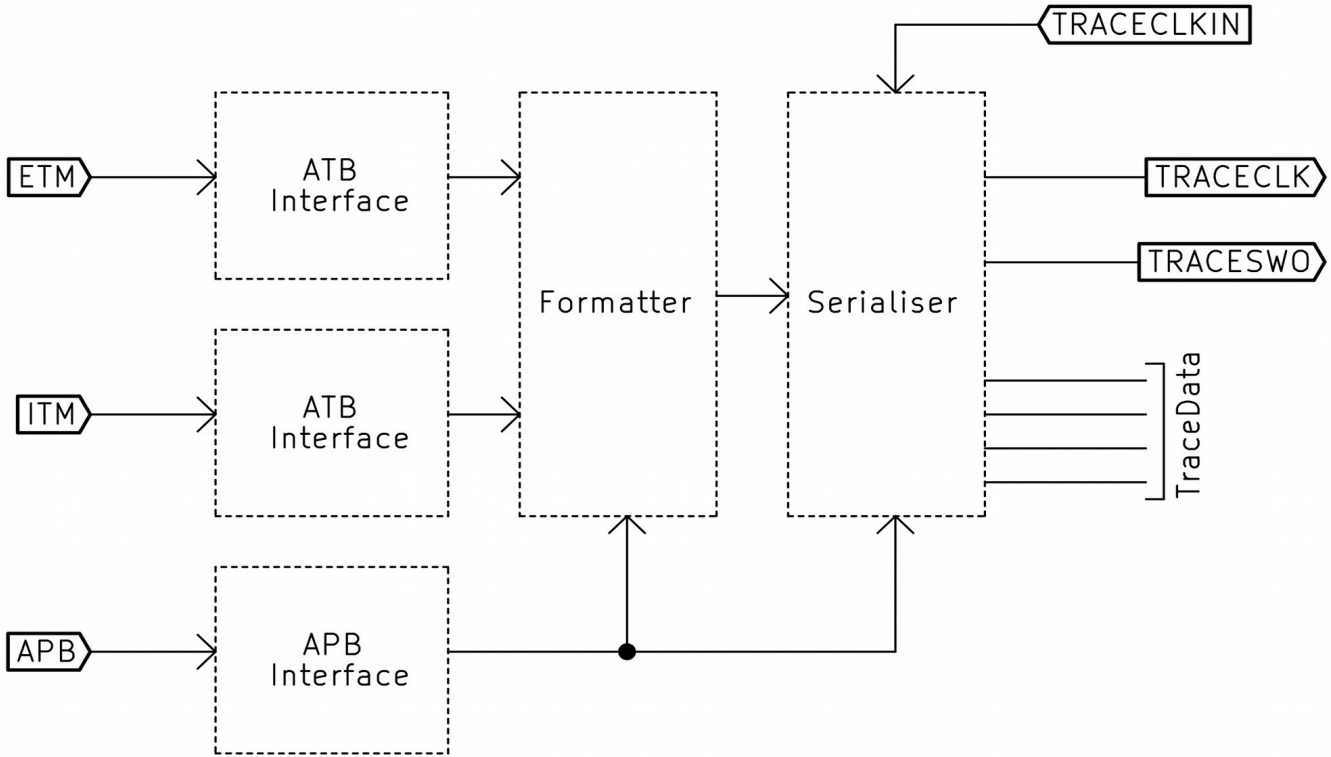
- Comes for free with SWD with most probes – not available via JTAG (re-uses TDO pin)

- Provides lots of functionality;
  - 32 Distinct debug channels
  - Hardware tracing and sampling
  - Debug Watch and Traps
  - Time stamps

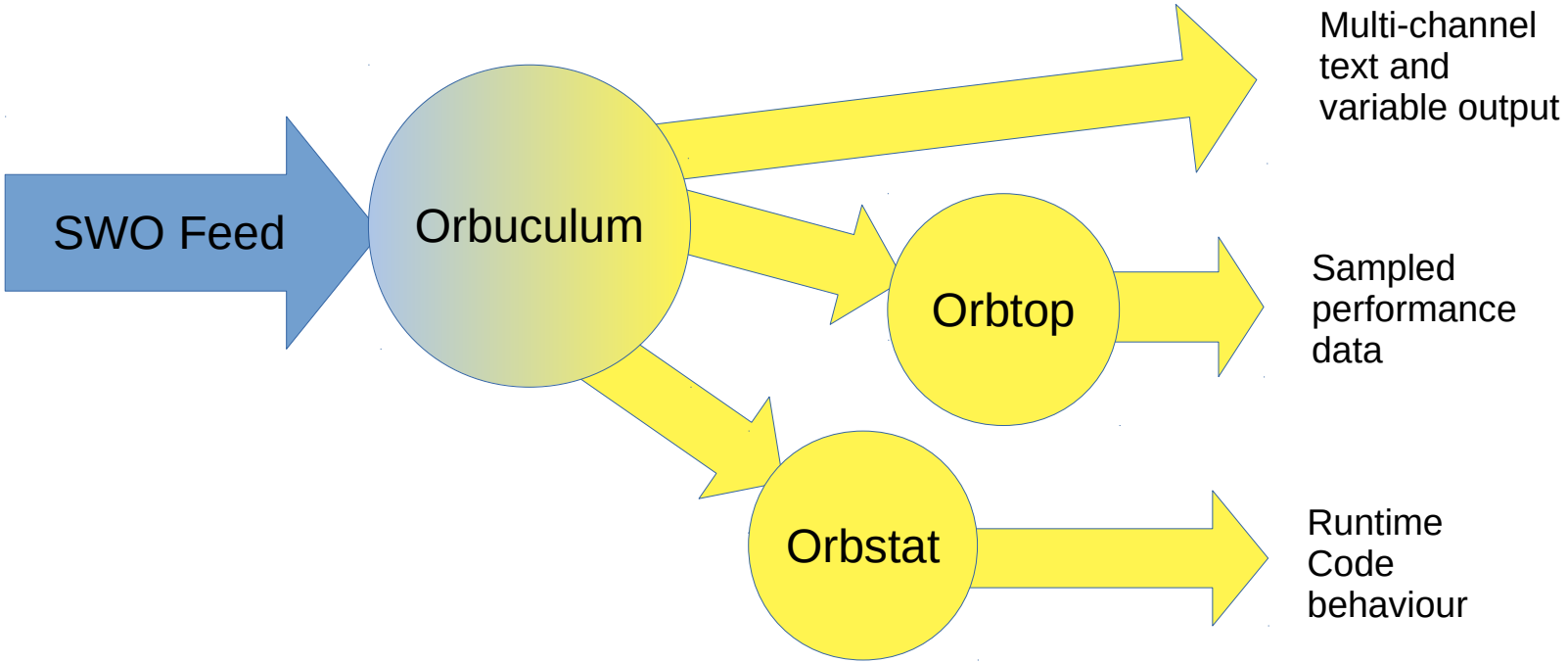
- Approx 111K bytes/sec
- Packetised format, needs to be depacketised to be useful (in the general case).

Technolution

Enter the SWO...

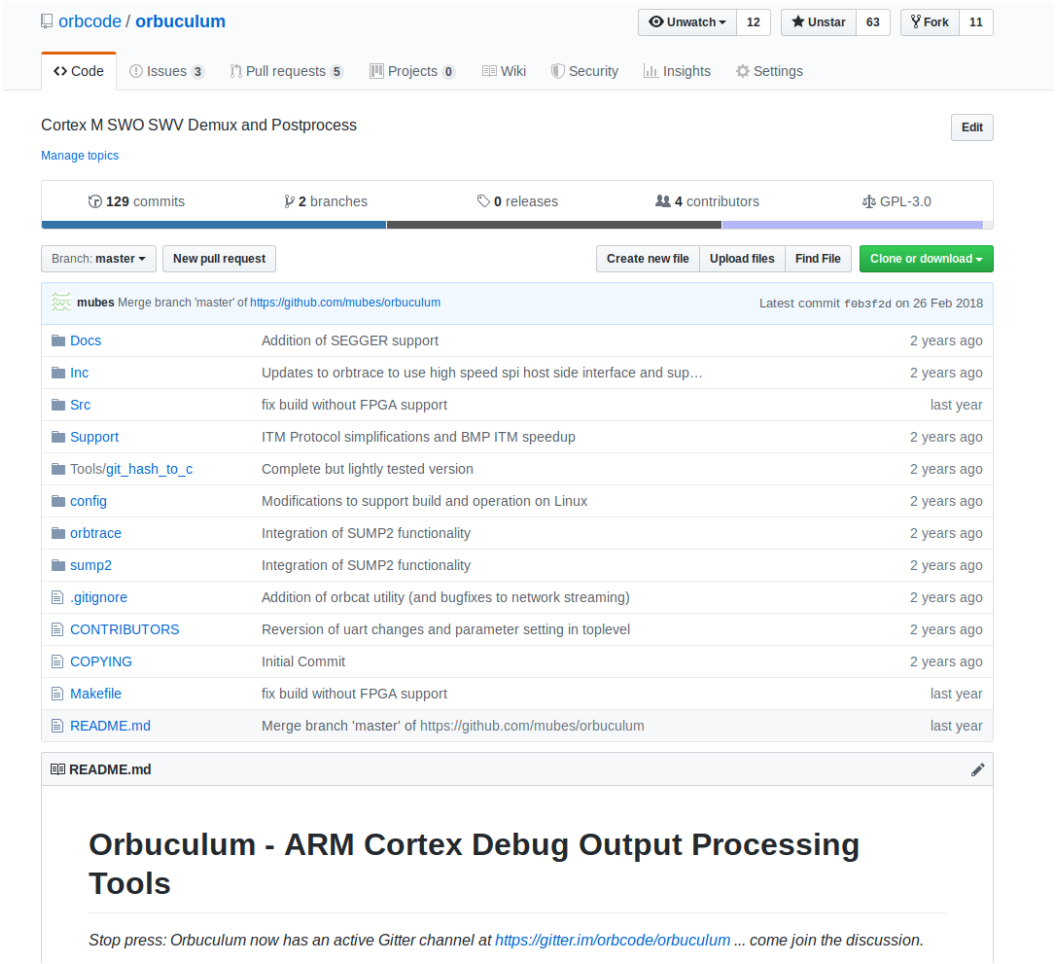


Enter the SWO



Technolution

https://github.com/orbcode/orbuculum



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 f6b3f2d on 26 Feb 2018

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

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.

Technolution

SWO: Debug Channels

```
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);
}
```

```
>orbuculum -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
>_
```

Technolution

SWO: Debug Channels

```
sendString(DEBUG_CHANNEL, “Debug Event Happened”);
```

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

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

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

```
sendString(CLIENTEVENTS_CHANNEL, “Port Opened”);
```

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

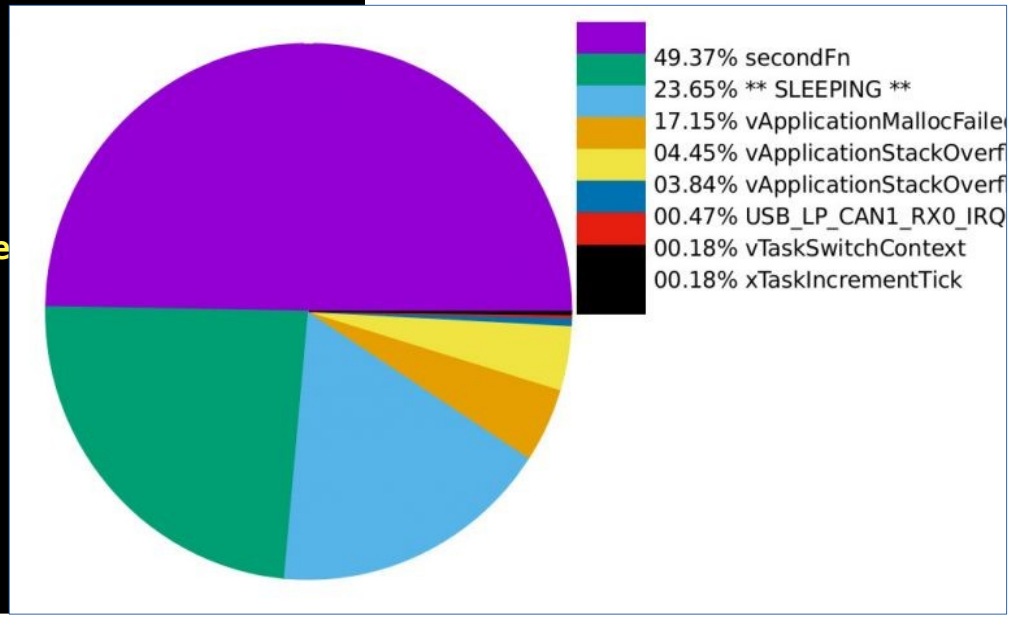
```
gdb>dwtTraceException 1
_
```

```
>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
_
```

Technolution

SWO: Processing Debug Data

```
>Support/orbplot_top &
>orbtop -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
```



Technolution

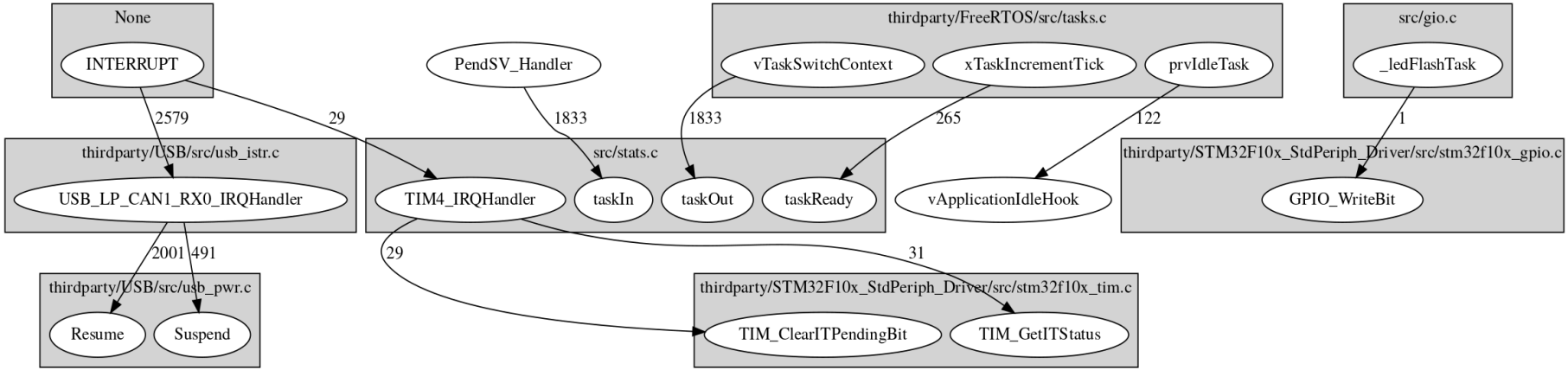
Using SWO For Instrumentation

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

Technolution

Usng SWO For Instrumentation

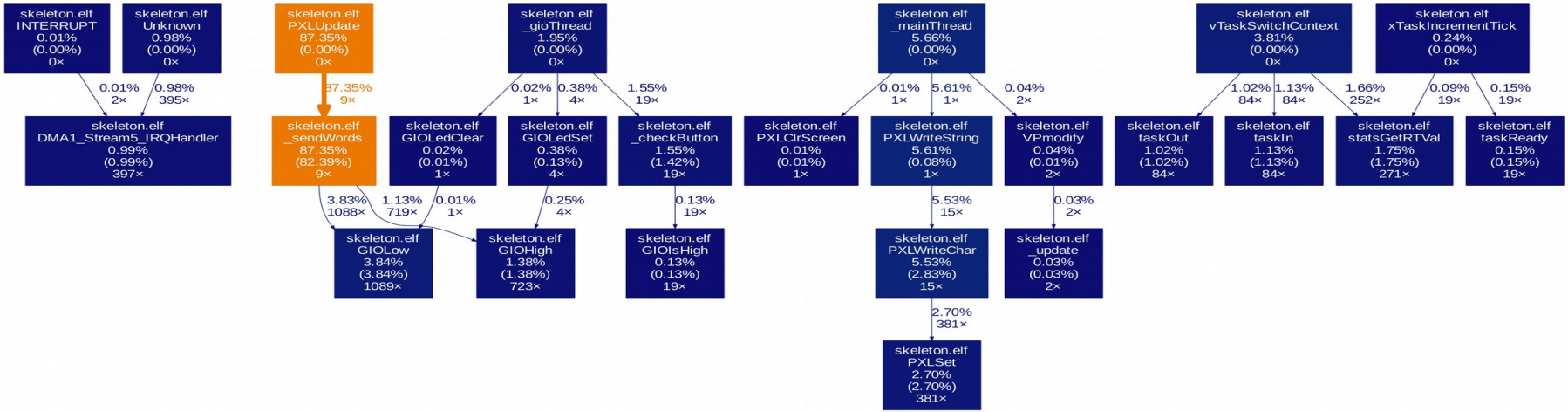
```
>orbstat -e firmware.elf -y
out.dot
>dot out.dot -o -Tpng > out.png
>evince out.png
_
```



Usng SWO For Instrumentation

Alternative Formats

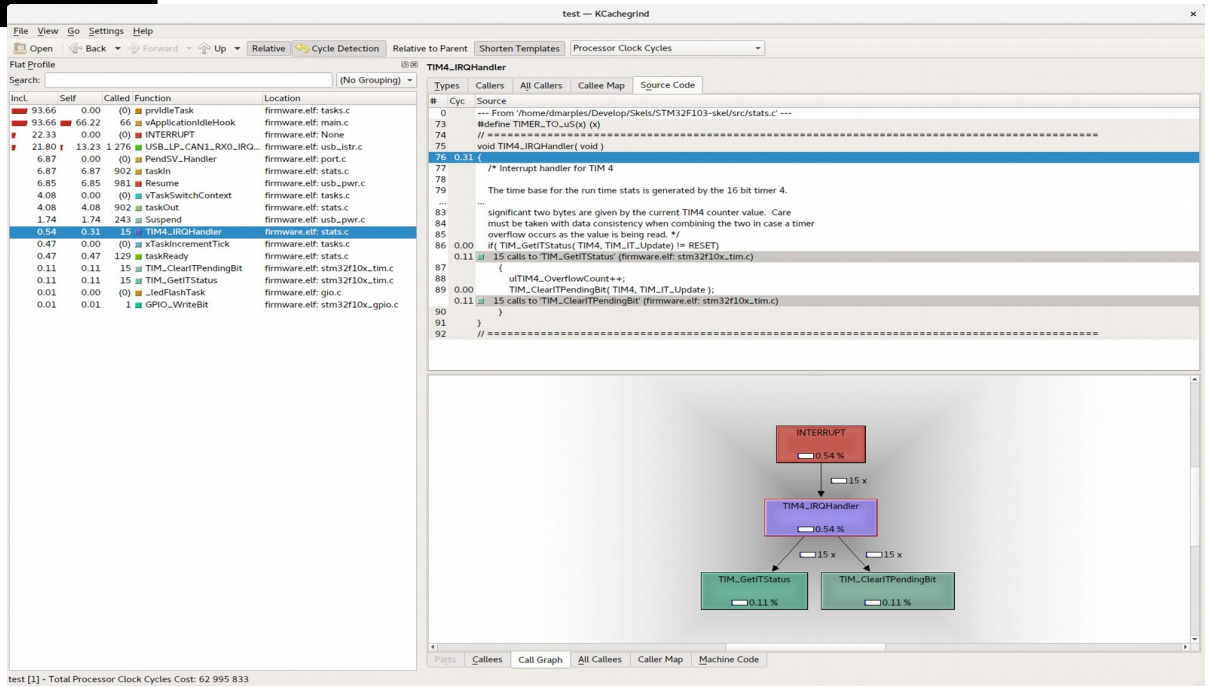
```
>orbstat -e firmware.elf -z demo.out
>gprof2dot -f callgrind demo.out >
demo.dot
>dot demo.dot -o -Tpng > out.png
>evince out.png
_
```



Technolution

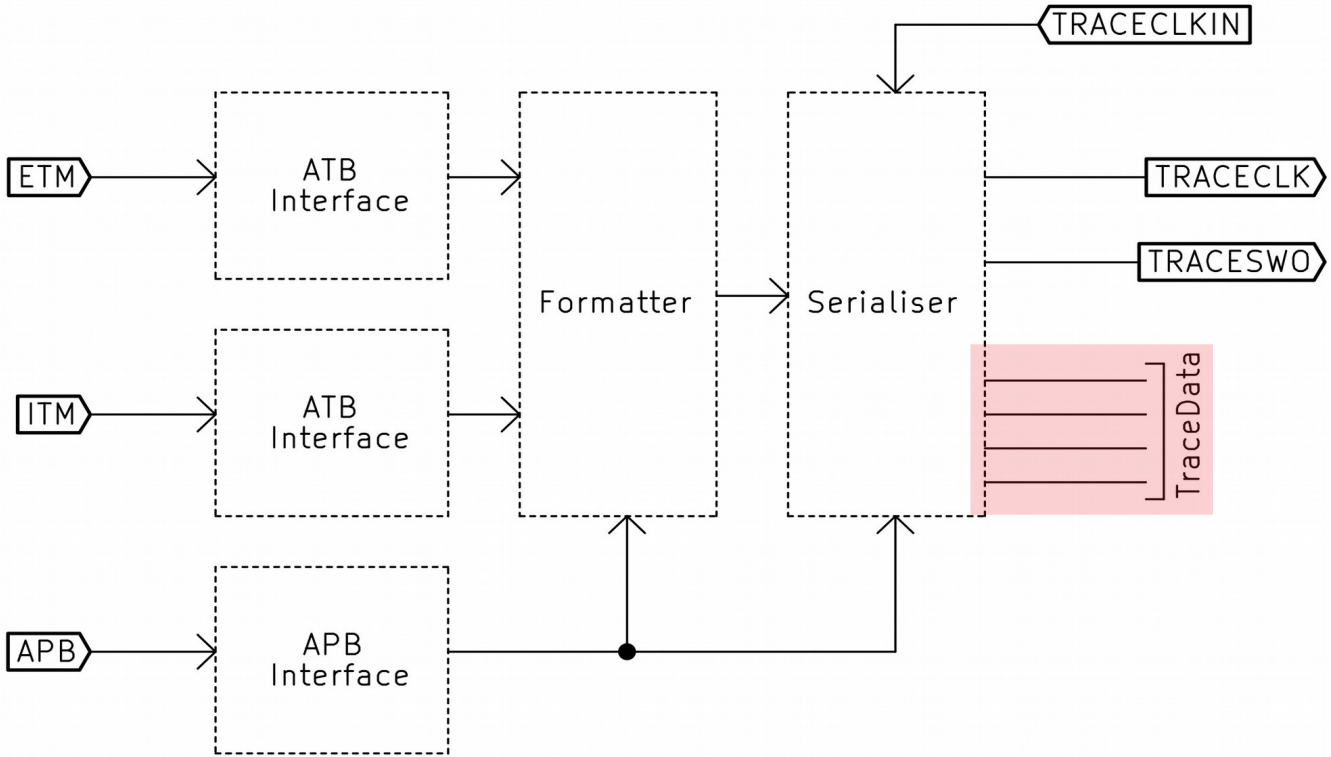
Usng SWO For Instrumentation
Interactive Output

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

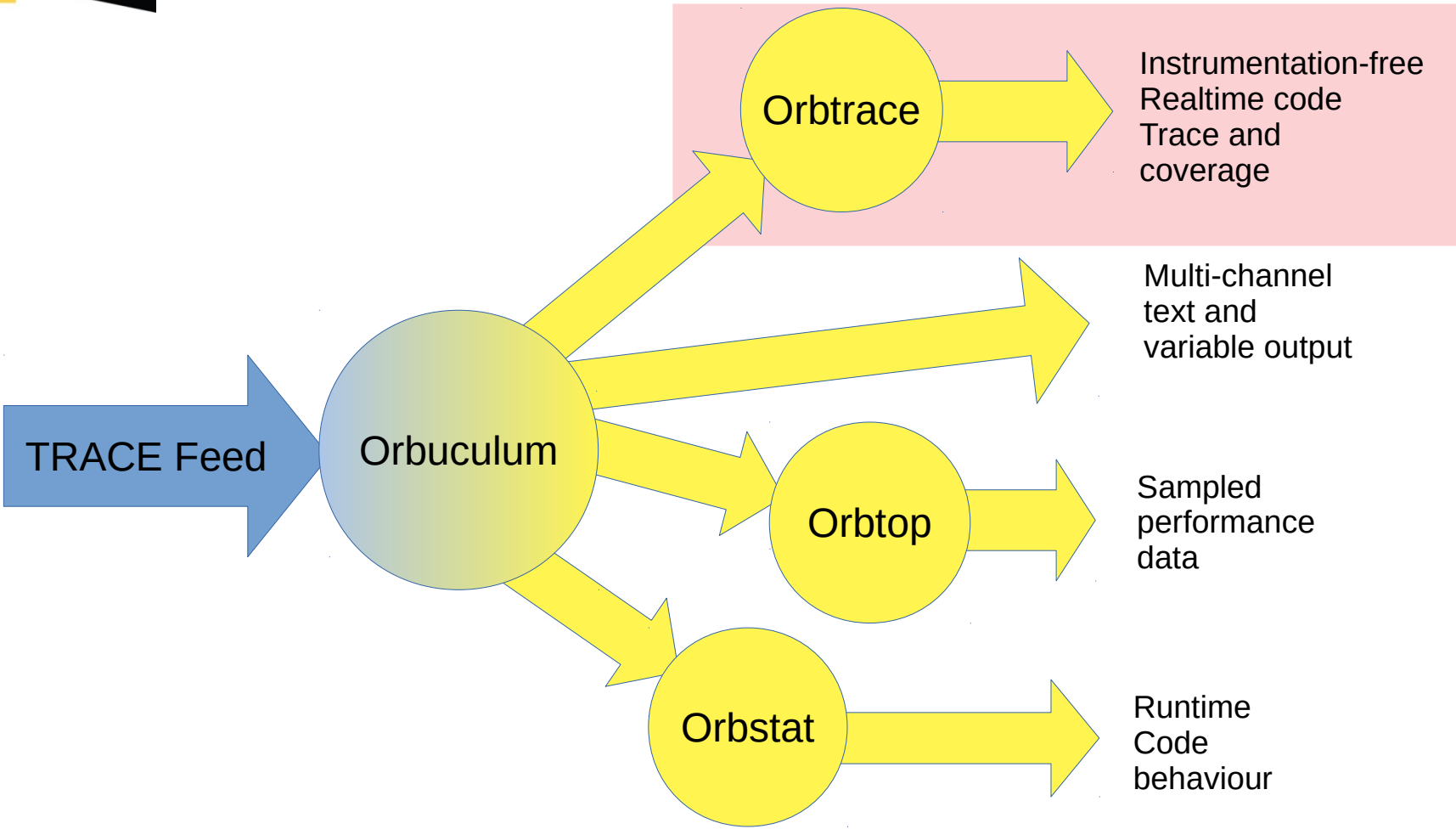


Technolution

Enter the SWO...



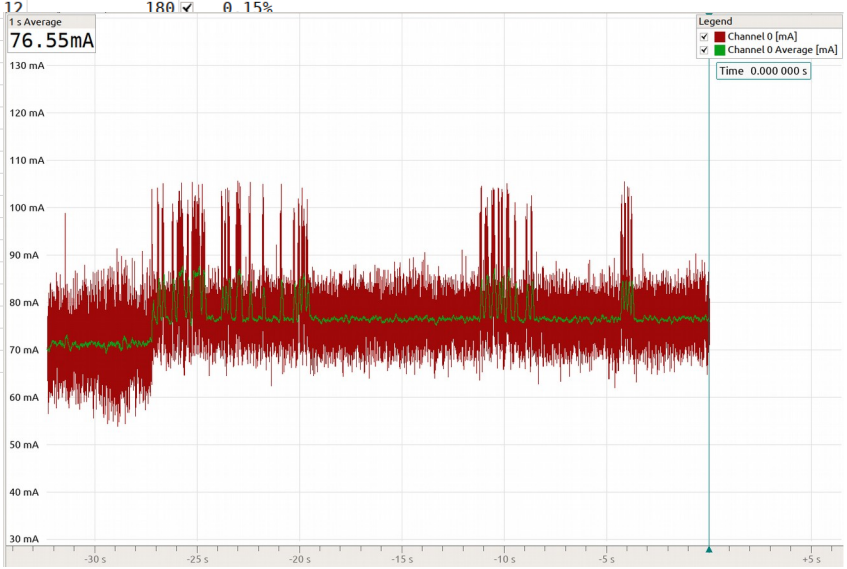
SWO to TRACE...


Orbtrace
Instrumentation-free
Realtime code
Trace and
coverage
Multi-channel
text and
variable output
TRACE Feed
Orbuculum
Orbtop
Sampled
performance
data
Orbstat
Runtime
Code
behaviour

Technolution

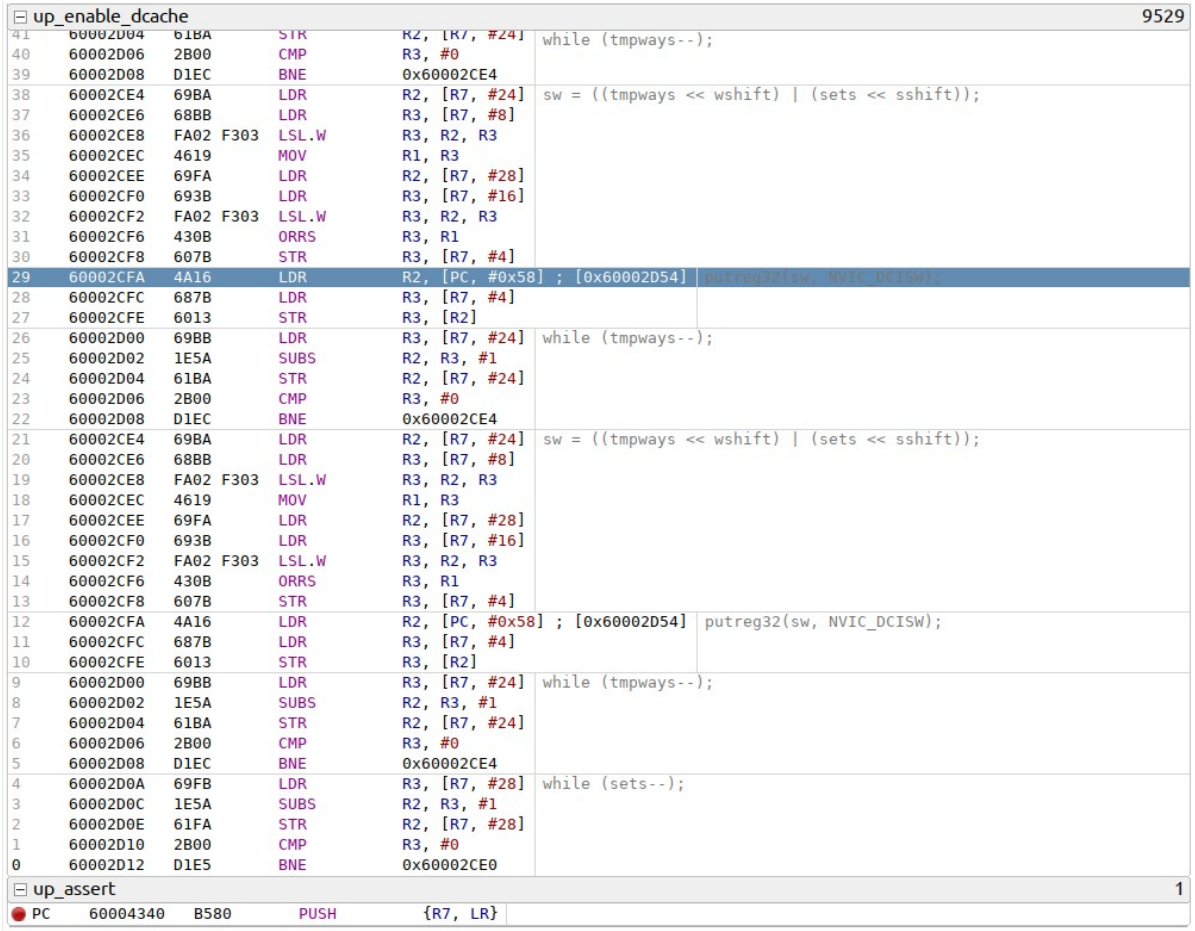
What TRACE can do
(Ozone/J-TRACE examples)

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_confi	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_configinpu	80.6% (58/72)			
90.9% (10/11)	f imxrt_fpuconfig	97.4% (37/38)			
77.8% (7/9)	f imxrt_gpio_configoutp	84.6% (33/39)			
60.0% (6/10)	f mpu_control	83.8% (31/37)			
62.5% (5/8)	f imxrt_gpio_setoutput	76.2% (32/42)			
80.0% (4/5)	f imxrt_gpio_dirin	96.9% (31/32)			
91.7% (11/12)	f imxrt_tcmenable	96.8% (30/31)			
100.0% (5/5)	f imxrt_gpio_dirout	100.0% (30/30)			
92.3% (12/13)	f imxrt_lowsetup	96.3% (26/27)			
90.0% (9/10)	f up_enable_icache	96.0% (24/25)			
100.0% (4/4)	f arm_clz	100.0% (13/13)			
100.0% (3/3)	f imxrt_gpio_select	100.0% (11/11)			
100.0% (5/5)	f imxrt_mpu_initialize	100.0% (10/10)			
100.0% (2/2)	f mpu_showtype	100.0% (6/6)			
66.7% (2/3)	f imxrt_autoled_initializ	85.7% (6/7)			
100.0% (3/3)	f imxrt_boardinitialize	100.0% (5/5)			



Technolution

What TRACE can do
(Ozone/J-TRACE examples)



The future...

- 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

www.github.com/mubes/orbuculum

Thanks for listening


Technolution