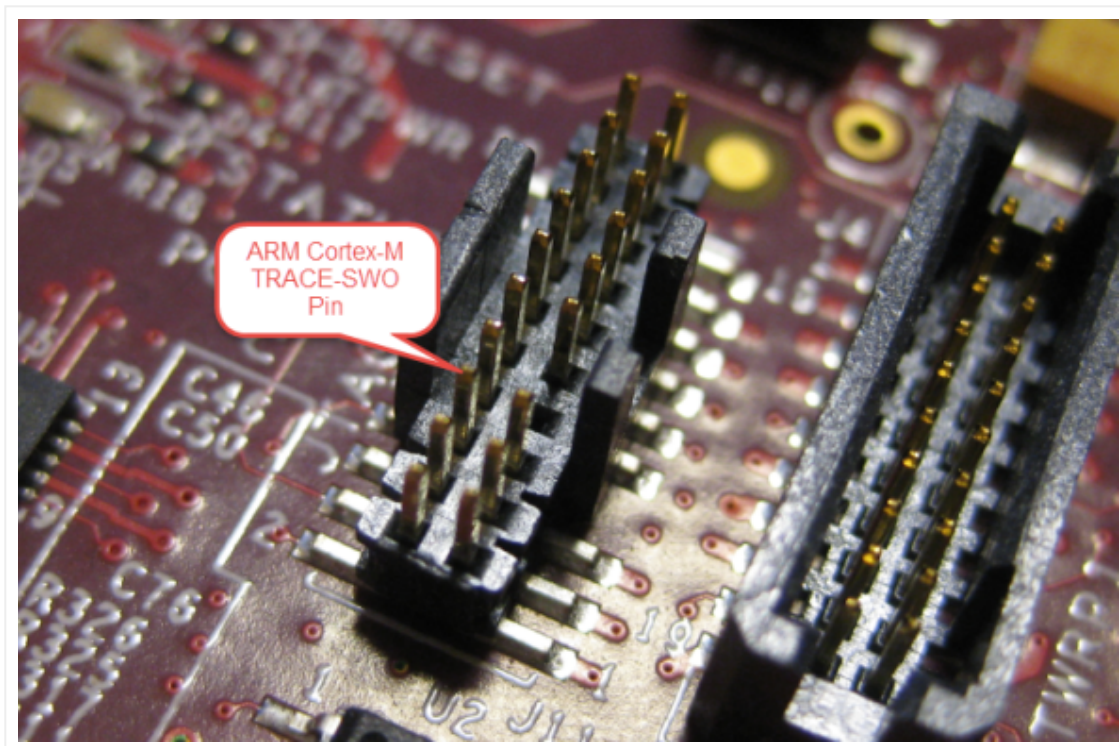


Tutorial: Using Single Wire Output SWO with ARM Cortex-M and Eclipse

Posted on [October 17, 2016](#) by [Erich Styger](#)

As a standard procedure, I add some console functionality to my embedded applications. That way I have a command line interface and can inspect and influence the target system. One interesting hardware feature of ARM Cortex-M is Single Wire Output (SWO): it allows to send out data (e.g. strings) over up to 32 different stimulus ports, over a single wire.



— swo-pin-on-arm-debug-header

Debug Trace Output? SWO!

As the standard text and command line interface to my target boards I'm using a normal UART/SCI. However, on many boards the UART's are used by the application.

There is [semihosting](#), but this is very slow, depends on the debugger and toolchain/library used plus is a waste of FLASH and RAM so I don't recommend using semihosting at all.

There is [USB CDC](#), but this requires a USB connector, a USB stack and a microcontroller capable of USB. Not applicable in all cases.

Stočte boty, narovnejte zá

Minimalistické boty s
maximálním komfortem.

[PRIVACY](#)[REPORT THIS AD](#)

There is [Segger RTT](#) which is small, fast and best of all does not need any special pins. But works only with Segger debugging probes.

ARM SWO

But there is yet another thing: [ARM SWO trace port](#) as defined by ARM for Cortex-M. Technically SWO is a single trace pin which is used to stream out data packets with a certain clock rate, derived from the CPU core clock. You can think of SWO as a kind of UART TX pin using a special format to send out data packets. Up to 32 packet types (or stimulus) can be used. What kind of data is sent is up to the application, and there is only very little CPU processing or code needed.

Common SWO usages are:

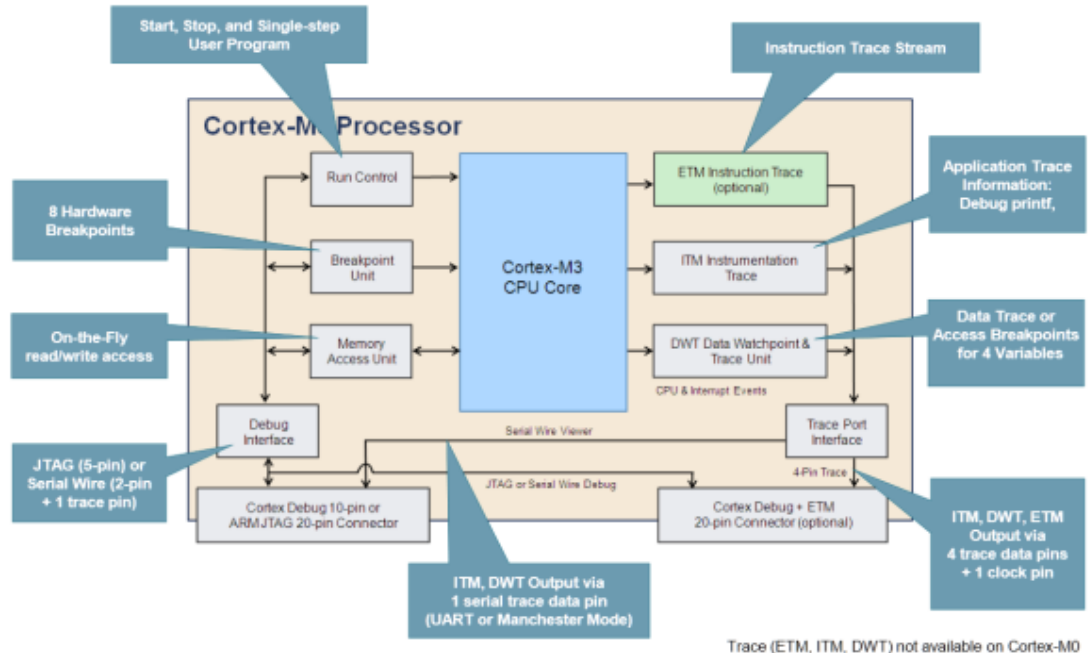
- Sending debug messages as strings
- Recording interrupt entry/exit
- Recording function entry/exit
- Periodic PC value sampling
- Event notification
- Variable or memory cell change over time

One of the most common usage is the first one: using SWO to print debug messages from the target in UART style. And this is what I'm going to show in this article. There would be another encoding (Manchester encoded) which is not covered here.

ARM CoreSight

SWO is part of the ARM CoreSight Debug block which usually is part of Cortex-M3, M4 and M7:

CoreSight Debug (Cortex-M)



- coresight-debug (Source: http://www.arm.com/files/pdf/AT_-_Advanced_Debug_of_Cortex-M_Systems.pdf)

Advertisements

Stočte boty, narovnejte zá

Minimalistické boty s
maximálním komfortem.

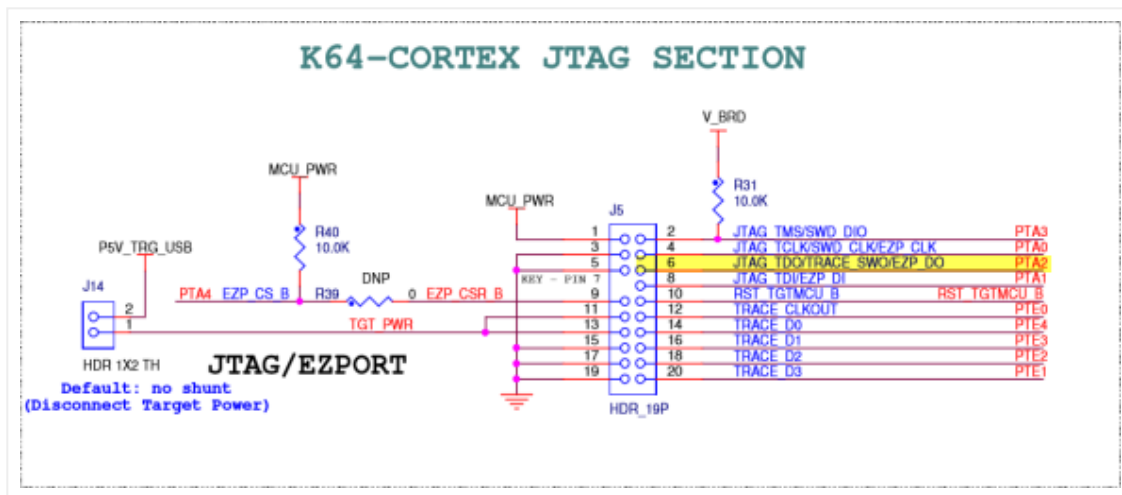
PRIVACY

REPORT THIS AD

As shown in that overview slide, over SWO (or SWV) ITM and DWT trace messages can be sent. For instruction trace up to 4 extra trace pins are required (see “[First Steps with Ozone and the Segger J-Link Trace Pro](#)” how to get instruction trace).

SWO Pin

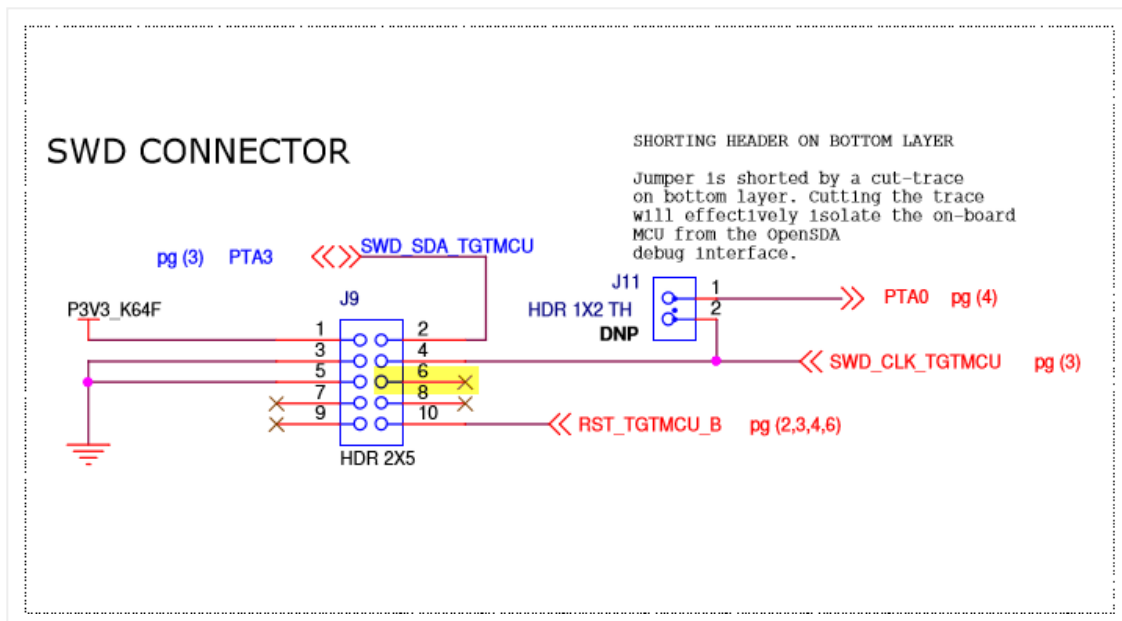
The precondition to use SWO is that the this pin is available on the debug header. This is the case for my TWR-K64F120M board:



— trace_swo_pin (Source: TWR-K64F120M Schematics)

As shown above, the SWO trace pin is shared with the JTAG TDO pin. So this means that SWO cannot be used with JTAG, it only can be used with SWD (see “[SWD Debugging with the FRDM-KL25Z Board](#)“).

Carefully check your board schematics if it supports SWO. For example on the [FRDM-K64F](#) (same device as on above [TWR-K64F120M](#)), the SWO pin is *not* routed to the debug header:



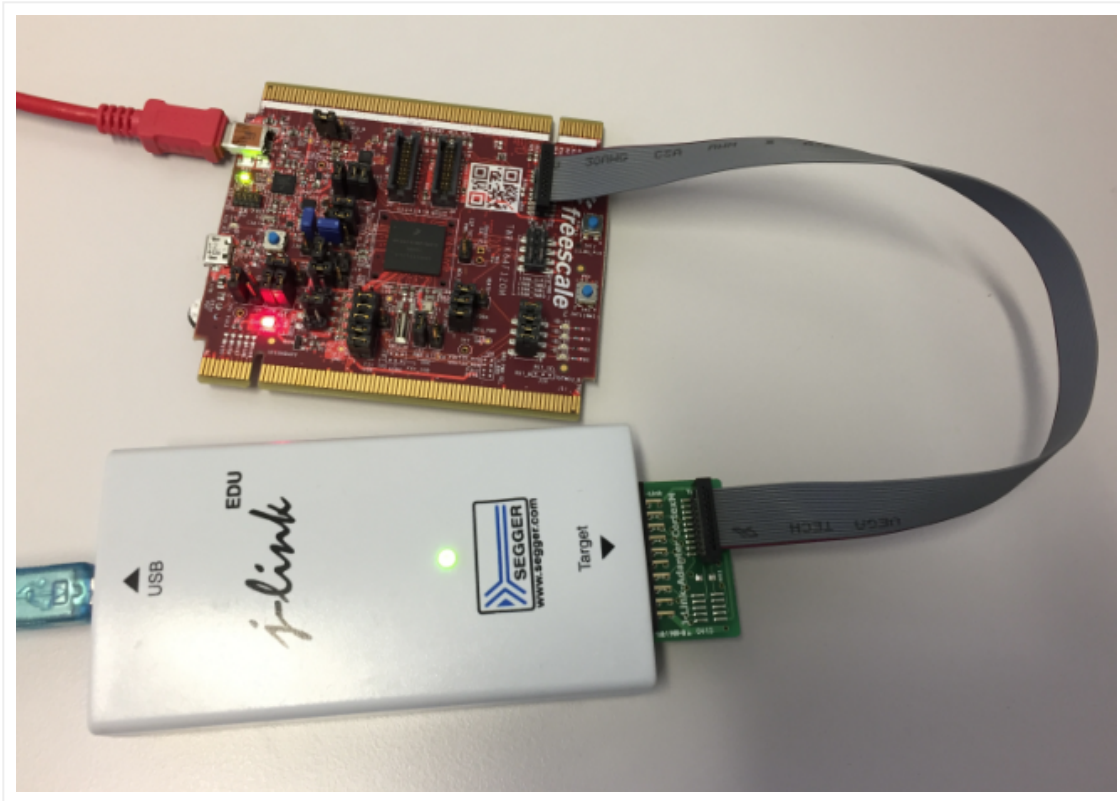
— no-swo-on-frdm-k64f

Debug Probe and SWO

In order to use SWO, I need a debug probe capable reading the SWO pin. For example the Freescale/NXP OpenSDA onboard debug interface hardware on the [Freedom](#) and [Tower](#) modules

does **not** support SWO (see “[Solving “The connected emulator does not support serial wire output \(SWO\)”](#)”).

The external Segger J-Link however do support the SWO pin. Below I have a J-Link EDU connected to the debug and trace port of the TWR-K64F120M board:



— j-link-edu-connected-to-trace-port

Source Code to Send Debug Messages over SWO Trace Pin

In order to write debug message over SWO to the host, a small piece of code is needed.

💡 An example project with all the sources is available on GitHub:

https://github.com/ErichStyger/mcuoneclipse/tree/master/Examples/KDS/TWR-K64F120M/TWR-K64F120M_Demo/Sources

External tools (like the Segger RTT viewer) can set up the SWO in the hardware. My recommendation is to initialize it from the application. Because the SWO trace output clock is derived from the CPU clock, the Init function needs that clock plus which SWO port number to be initialized. Below is the code I'm using to initialize the SWO output to a default of 64k baud:

```
1  /*!  
2   * \brief Initialize the SWO trace port for debug message printing  
3   * \param portBits Port bit mask to be configured  
4   * \param cpuCoreFreqHz CPU core clock frequency in Hz  
5   */  
6  void SWO_Init(uint32_t portBits, uint32_t cpuCoreFreqHz) {
```

```

7   uint32_t SWOSpeed = 64000; /* default 64k baud rate */
8   uint32_t SWOPrescaler = (cpuCoreFreqHz / SWOSpeed) - 1; /* SWOSpeed in Hz,
9
10  CoreDebug->DEMCR = CoreDebug_DEMCR_TRCENA_Msk; /* enable trace in core deb
11  *((volatile unsigned *) (ITM_BASE + 0x400F0)) = 0x00000002; /* "Selected PI
12  *((volatile unsigned *) (ITM_BASE + 0x40010)) = SWOPrescaler; /* "Async Clo
13  *((volatile unsigned *) (ITM_BASE + 0x00FB0)) = 0xC5ACCE55; /* ITM Lock Acc
14  ITM->TCR = ITM_TCR_TraceBusID_Msk | ITM_TCR_SWOENA_Msk | ITM_TCR_SYNCENA_M
15  ITM->TPR = ITM_TPR_PRIVMASK_Msk; /* ITM Trace Privilege Register */
16  ITM->TER = portBits; /* ITM Trace Enable Register. Enabled tracing on stim
17  *((volatile unsigned *) (ITM_BASE + 0x01000)) = 0x400003FE; /* DWT_CTRL */
18  *((volatile unsigned *) (ITM_BASE + 0x40304)) = 0x00000100; /* Formatter an
19  }

```

```

1  #define CPU_CORE_FREQUENCY_HZ 120000000 /* CPU core frequency in Hz */
2
3  SWO_Init(0x1, CPU_CORE_FREQUENCY_HZ);

```

The printing is done in SWO_PrintChar():

```

1  /*!
2  * \brief Sends a character over the SWO channel
3  * \param c Character to be sent
4  * \param portNo SWO channel number, value in the range of 0 to 31
5  */
6  void SWO_PrintChar(char c, uint8_t portNo) {
7      volatile int timeout;
8
9      /* Check if Trace Control Register (ITM->TCR at 0xE0000E80) is set */
10     if ((ITM->TCR & ITM_TCR_ITMENA_Msk) == 0) { /* check Trace Control Register
11         return; /* not enabled? */
12     }
13     /* Check if the requested channel stimulus port (ITM->TER at 0xE0000E00) i
14     if ((ITM->TER & (1u << portNo)) == 0) { /* check Trace Enable Register if req
15         return; /* requested port not enabled? */
16     }
17     timeout = 5000; /* arbitrary timeout value */
18     while (ITM->PORT[0].u32 == 0) {
19         /* Wait until STIMx is ready, then send data */
20         timeout--;
21         if (timeout == 0) {
22             return; /* not able to send */
23         }
24     }
25     ITM->PORT[0].u16 = 0x08 | (c << 8);
26 }

```

The above code uses a very simple timeout mechanism: the important point is not to block if SWO is not enabled or if SWO port is not ready, otherwise the application will be blocked.

To make it easier to print a string, I'm using the following function:

```

1  /*!
2  * \brief Sends a string over SWO to the host
3  * \param s String to send
4  * \param portNumber Port number, 0-31, use 0 for normal debug strings

```



```

5 |  */
6 |  void SWO_PrintString(const char *s, uint8_t portNumber) {
7 |      while (*s!='\0') {
8 |          SWO_PrintChar(*s++, portNumber);
9 |      }
10 |  }

```

To send a 'hello' over SWO it gets as easy as this:

```

1 | SWO_PrintString("hello world with SWO\r\n", 0);

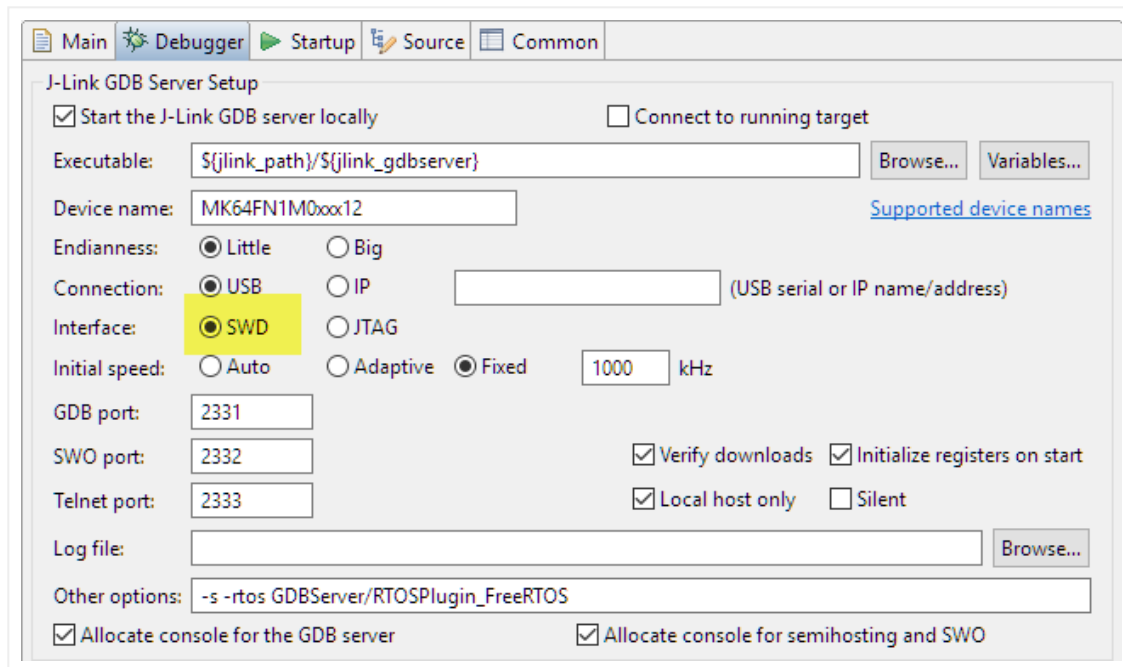
```

The first parameter is the string to send, the second is the SWO trace channel number.

GNU Arm Eclipse Viewer

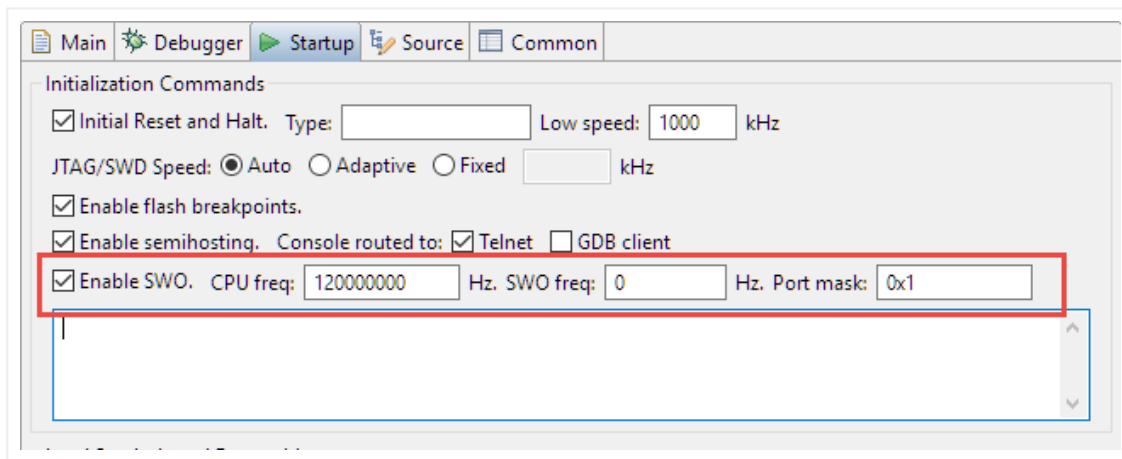
To receive the SWO trace output on the host, the [GNU ARM Eclipse plugins](#) have built-in SWO support for the Segger J-Link probes.

SWO only is supported in SWD (Single Wire Debug) mode, and not in JTAG mode. So make sure that SWD is selected as debugging protocol:



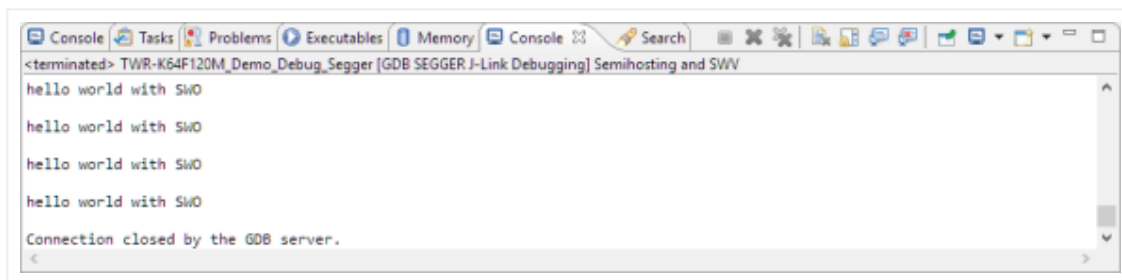
— swd-debug

In the GNU ARM Eclipse debug configuration, enable SWO and specify the CPU frequency and the SWO frequency (see the documentation about the frequencies on <http://gnuarmclipse.github.io/debug/jlink/>). I have to provide the CPU frequency (120 MHz in my case), and can leave the SWO frequency at 0 so the J-Link will automatically determine the speed). Specify in the port mask the ports (as bitmask) used, so 0x1 is for using port 0:



— swo-settings

With this, running the application on the target it shall show the output in the Eclipse Console View:

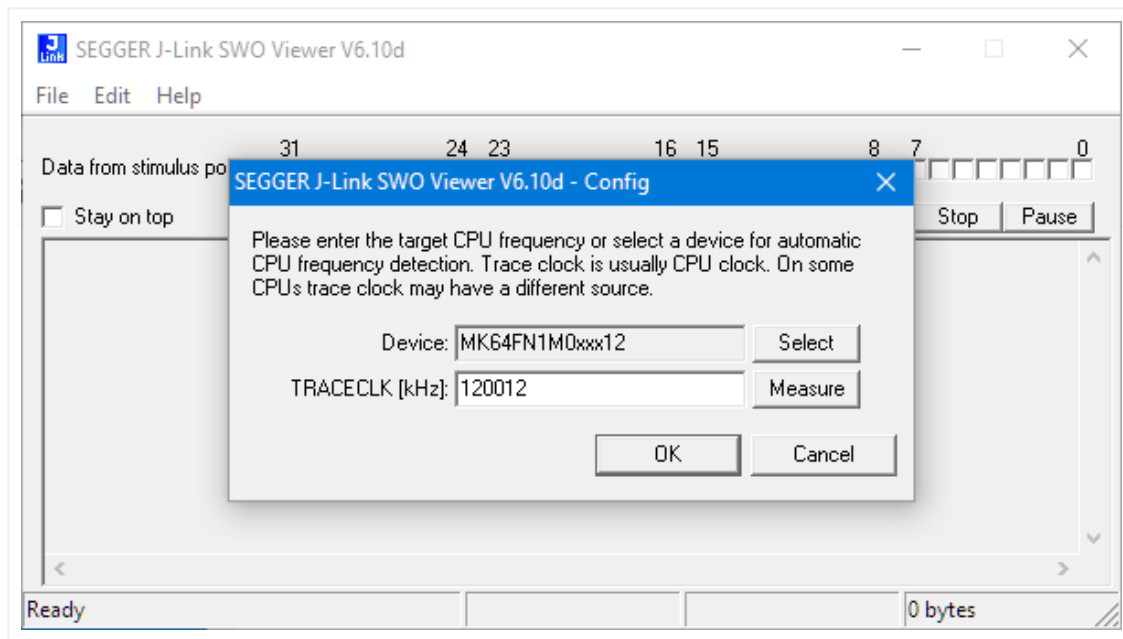


— eclipse-console-view

Segger SWO Viewer

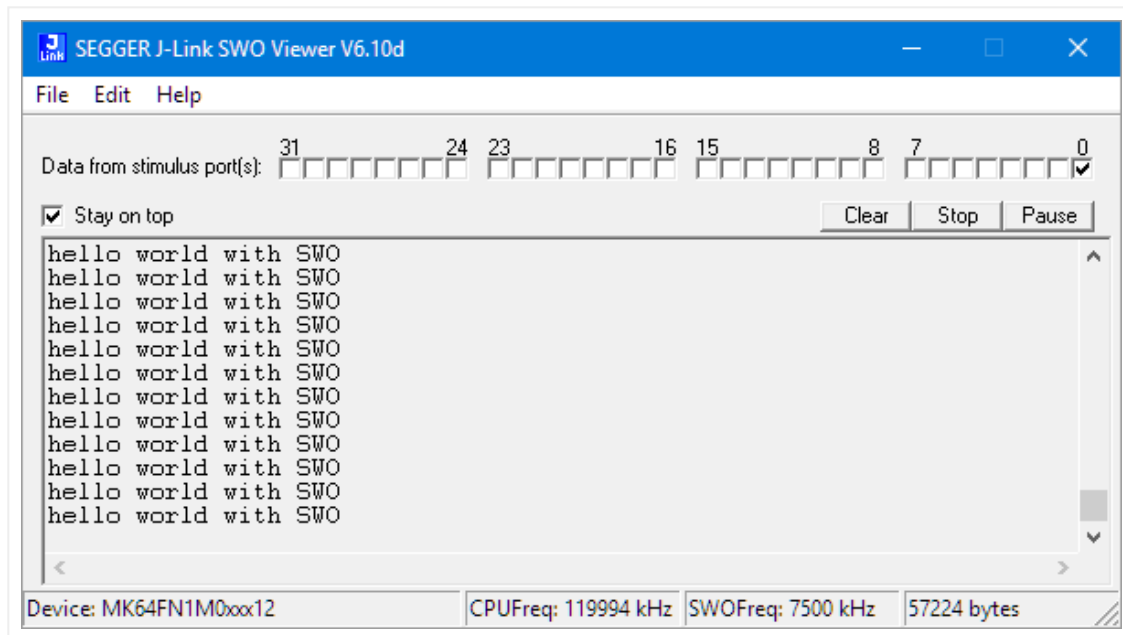
Segger has a special [SWO Viewer](#) (both command line and GUI version).

In the GUI version I specify the device used, and it can sense the trace clock:



— segger-gui-swo-viewer

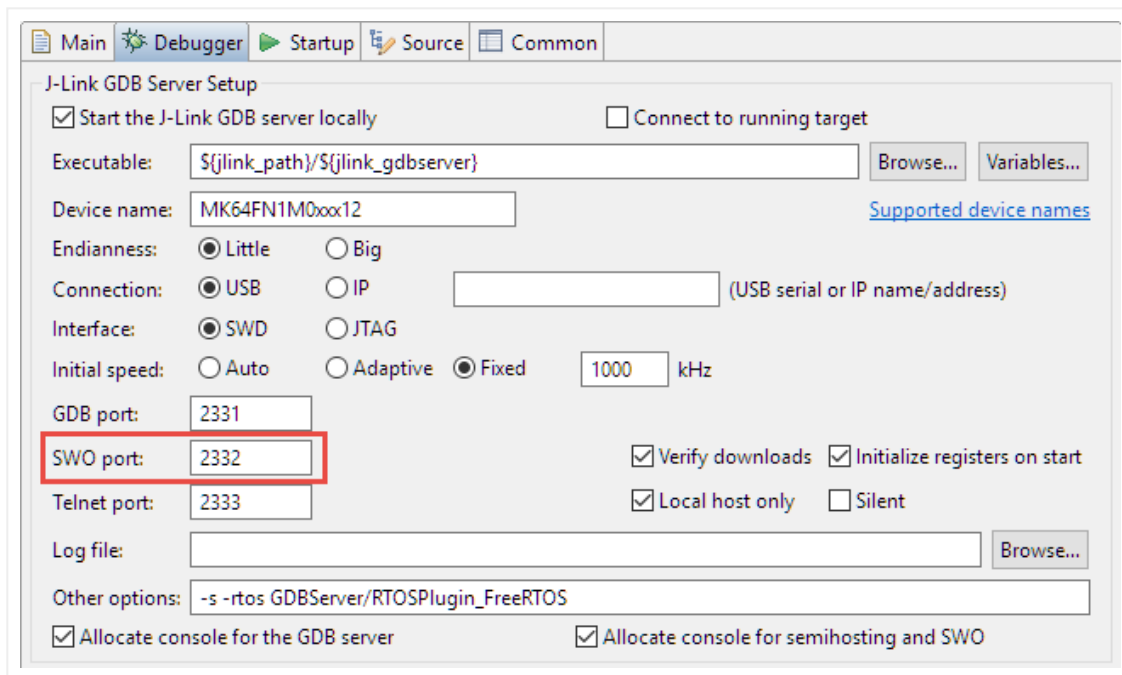
In the viewer I can turn on/off ports and see the data received:



— segger-j-link-swo-viewer

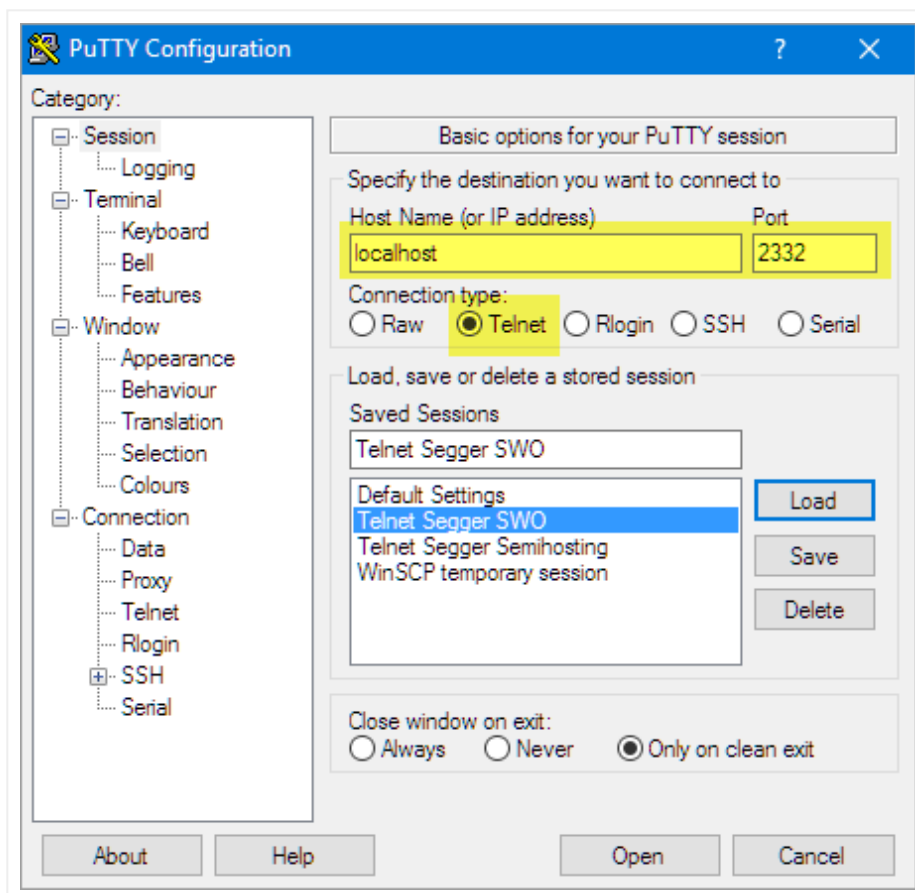
Telnet: Putty

But there is no fancy viewer or Eclipse needed to see the SWO data. Segger is using by default the port 2332:



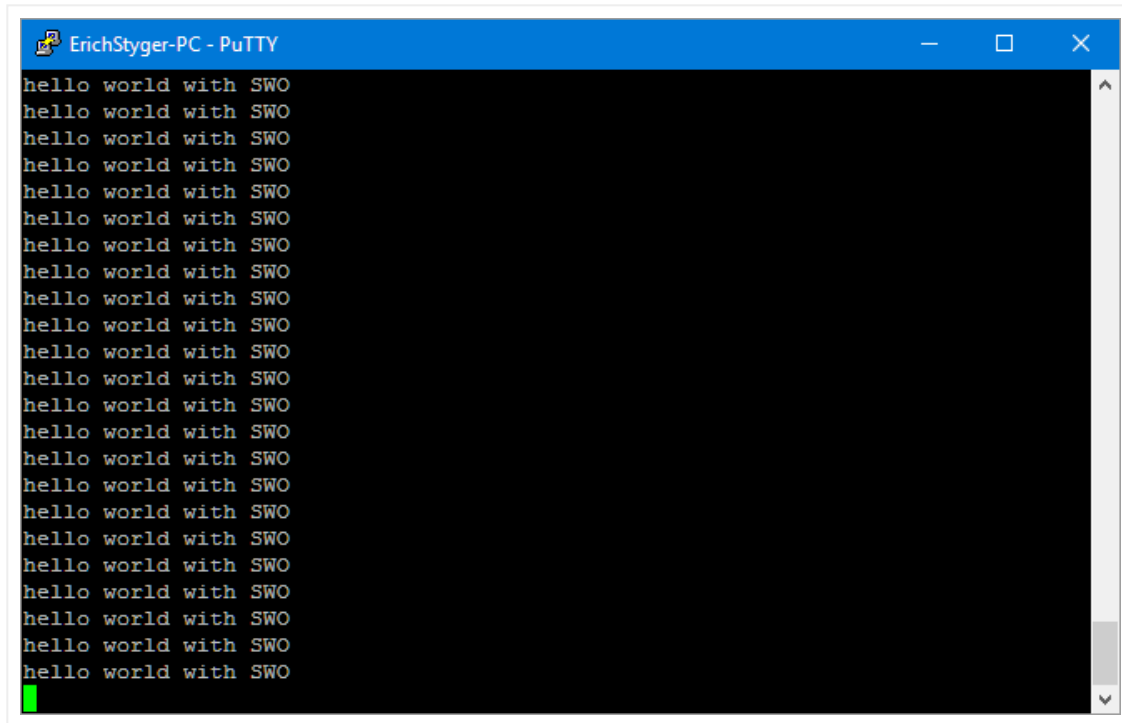
— segger-swo-port

I can configure any telnet client (e.g. [PuTTY](#)) to open a session on port 2332:



— putty-telnet-session-settings

And then I get the output in PuTTY:



```
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
hello world with SWO
```

— swo-output-in-putty

Summary

The ARM SWO trace pin allows to send trace messages to the host. One common usage is to send debug or other messages to the host. SWO only needs a single pin, works only with SWD (not JTAG) and requires little code and resources on the target. Unfortunately many boards do not have the SWO trace pin routed to the debug header, so if you are making your own design, routing SWO to the debug header should be at least considered.

While SWO trace output is great, it is limited to the higher end Cortex-M, I did not find it in the Cortex-M0(+) available, and it is output only, and requires a debug probe/interface supporting it. At least with Eclipse and GNU ARM Eclipse plugins in combination with Segger J-Link probes SWO output has worked great for me.

On the other side, the Segger RTT is much more versatile and very fast too. It works on all ARM Cortex, and best of all, it does not need an extra pin :-). However, it requires little more overhead and RAM resources on the target system. Plus it allows both to send and receive data. So for the serial debug message printing, the Segger RTT sounds a better solution to me.

Happy SWO'ing 😊

LINKS

- SWO on Kinetis: <https://community.nxp.com/thread/318058>
- SWO with GNU ARM Eclipse plugins: <http://gnuarmecclipse.github.io/debug/jlink/>

- Segger SWO Viewer: <https://www.segger.com/j-link-swo-viewer.html>
- GNU ARM Eclipse plugins: <http://gnuarmclipse.github.io/debug/>
- Eclipse RxTx Plugin for parsing SWO data: <http://forum.segger.com/index.php?page=Thread&threadID=1010>
- ARM CoreSight Overview: http://www.arm.com/files/pdf/AT_-_Advanced_Debug_of_Cortex-M_Systems.pdf
- Example code on GitHub: https://github.com/ErichStyger/mcuoneclipse/tree/master/Examples/KDS/TWR-K64F120M/TWR-K64F120M_Demo/Sources

This entry was posted in [ARM](#), [Boards](#), [CPU's](#), [Debugging](#), [Eclipse](#), [Embedded](#), [Freescale](#), [gcc](#), [KDS](#), [Kinetis](#), [LPC](#), [LPC](#), [NXP](#), [Tips & Tricks](#), [Tutorial](#) and tagged [ARM](#), [Cortex](#), [Cortex-M](#), [Debugging](#), [Eclipse](#), [FRDM-K64F](#), [freedom board](#), [Freescale](#), [JTAG](#), [Kinetis](#), [NXP](#), [software](#), [SWO](#), [technology](#), [Thoughts](#), [Tips&Tricks](#), [Trace](#) by [Erich Styger](#). Bookmark the [permalink \[https://mcuoneclipse.com/2016/10/17/tutorial-using-single-wire-output-swo-with-arm-cortex-m-and-eclipse/\]](https://mcuoneclipse.com/2016/10/17/tutorial-using-single-wire-output-swo-with-arm-cortex-m-and-eclipse/) .



About Erich Styger

Embedded is my passion....

[View all posts by Erich Styger](#) →

42 THOUGHTS ON "TUTORIAL: USING SINGLE WIRE OUTPUT SWO WITH ARM CORTEX-M AND ECLIPSE"



Liviu Ionescu (ilg)

on [October 17, 2016 at 09:08](#) said:

The projects generated by the GNU ARM Eclipse templates, and subsequently the projects using the μ OS++/CMSIS++ APIs, all benefit from a trace channel, which can be routed either to semihosting, SWO or Segger RTT. The high level API is simple and very convenient: `trace_putchar()`, `trace_puts()` and `trace_printf()` (and their C++ versions, see http://micro-os-plus.github.io/reference/cmsis-plus/group__cmsis-plus-diag.html).

Highly recommended!

★ Like



[Chad Williams \(@AptronAustralia\)](#)

on [October 19, 2016 at 07:32](#) said:

So many choices for debugging. I attended the LPCxpresso training at NXPFTF which used the LPC43xx boards I will need to check whether the trace used SWO Periodic PC value sampling or accessed the ETB – Embedded trace buffers. Whats the lowest cost method for instruction tracing ? Is it possible to obtain SWO sampling or ETB trace information for Kinetis devices using Ozone?

★ Like



Erich Styger

on **October 20, 2016 at 21:22** said:

Hi Chad,

choices are good, right? Yes, LPCXpresso provides PC sampling over SWO. I have not found out how to explicitly see this in Ozone, but it seems to me that the profiling they provide is based on exactly that. I have a K64F setup for the trace already, but I don't see the profiling information yet.

I have seen that PC sampling in the Segger Embedded Studio, see

<https://blog.segger.com/profilingsoftwareinembeddedsystems/>

I plan to work on the SWO/ETB tracing with Ozone for Kinetis over the week-end if time permits.

★ Liked by [1 person](#)



brice

on **December 6, 2016 at 16:34** said:

I'm extremely interested if you succeed to get trace on swo/etb with ozone.

by the way i was sure that cortex M7 provides ETB buffer but i can't find any information

★ Like



Erich Styger

on **December 6, 2016 at 16:44** said:

Hi Brice,

I'm getting ETM trace on the trace pins (did not use SWO for this) with Kinetis K64F (M4F), see

<https://mcuoneclipse.com/2016/11/05/tutorial-getting-etm-instruction-trace-with-nxp-kinetis-arm-cortex-m4f/>.

I have not tried this with M7 yet.

★ Like

Pingback: [Getting printf Output from Target to Debugger | SEGGER Blog](#)

Pingback: [Tutorial: Getting ETM Instruction Trace with NXP Kinetis ARM Cortex-M4F | MCU on Eclipse](#)



Gustavo Cardoso

on **November 11, 2016 at 12:16** said:

Hi Erich

Nice article!

I just got it to work with my stm32 application, but some adjustments were needed:

Changed ITM->PORT[0].u16 = 0x08 | (c<PORT[portNo].u8 = (uint8_t)c;
(because I was getting some garbage along with the messages)

And I also changed 2 references of PORT[0] to PORT[portNo] in the SWO_PrintChar function, to correct the port selection

Now it works like a charm 😊

I am getting the print thru the SWO viewer in the ST-LINK utility, I can even connect the STLINK in the application and start getting the messages “on the fly”, without resetting it! pretty handy

★ Like



Gustavo Cardoso

on **November 11, 2016 at 12:18** said:

For some reason the text was wrong:

Changed :

ITM->PORT[0].u16 = 0x08 | (c<PORT[portNo].u8 = (uint8_t)c;



★ Like



Erich Styger

on **November 11, 2016 at 20:58** said:

The problem is that wordpress thinks it is HTML code (unless you use the 'code' formatter in the comment?)

★ Like



Erich Styger

on **November 11, 2016 at 20:52** said:

Hi Gustavo,

thanks for the hint about the the port number: indeed, I have to fix this on my side too.

Not sure why you need to change the write to the ITM port: I saw something like this, but on my side just writing the character worked fine?

I saw as well <http://forum.segger.com/index.php?page=Thread&threadID=1010> which used my approach.

★ Like



Gustavo Cardoso

on **November 12, 2016 at 02:16** said:

Maybe is something related only to the STLINK SWO utility, and works just fine in GDB

I changed it based on ITM_SendChar function, from core_cm3.h

★ Like



Erich Styger

on **November 12, 2016 at 07:13** said:

Hi Gustavo,

it seems to me that this is somehow a difference between M3 and M4 (should not be?), because in the core_cm4.h it is using


```
ITM->PORT[0U].u8 = (uint8_t)ch;
```

Anyway, thanks for the hint, good to know if I want to run this on a Cortex-M3.

★ Like



George Socker

on **June 1, 2017 at 14:17** said:

Freescall/NXP seems to have routed SWO to the header on the FRDM-k64F at some point. The current schematic shows it routed and I've been able to get SWO working with my J-Link.

Bottom of the board says "SCH-28163 REV E3".

★ Like



Erich Styger

on **June 1, 2017 at 14:21** said:

Really interesting, thanks for sharing!

★ Like



Jeff

on **December 6, 2017 at 16:12** said:

Can you elaborate why the 3rd bit is set and why the data byte is shifted 8 bits?

```
ITM->PORT[0].u16 = 0x08 | (c<<8);
```

★ Like



Erich Styger

on **December 12, 2017 at 20:42** said:

Hi Jeff,

the swap is about little/big endian. I have to check about the 0x08 thing (I copied that from an example), but I think it is for marking a 16bit ITM transfer. I have to check that example. I see that other code simply writes to .u8, so this should be enough too.

★ Like



Joel Perez

on **January 25, 2018 at 23:07** said:

Hi Erich.

This is really a nice article, thanks a lot.

I was wondering, do you know about any SWO alternative for Cortex R4 processors?

★ Like



Erich Styger

on **January 26, 2018 at 05:52** said:

Hi Joel,

I cannot comment much about the R4, as I'm currently not using any R4 devices. Are you asking about SWO viewers or tools to get SWO data or in general about how to get data off the device? If you don't know the Segger RTT, maybe this is what you are looking for? See <https://mcuoneclipse.com/2015/07/07/using-segger-real-time-terminal-rtt-with-eclipse/> and <https://www.segger.com/products/debug-probes/j-link/technology/real-time-transfer/about-real-time-transfer/>. With this you only need a small piece of software on the device and you can stream in/out data through the debug interface.

I hope this helps,

Erich

★ Like



Joel Perez

on **January 26, 2018 at 17:37** said:

Hi again!

Yes I was talking about a solution to get data off the device.

Actually yesterday I take a look at RTT, it compiled successfully and I am able to debug it on the device but I do not see any output using a telnet client (putty). If I use SystemView provided by SEGGER I get the error "cannot find RTT control block", I followed the recommendations in the FAQ's section in <https://www.segger.com/products/debug-probes/j-link/technology/real-time-transfer/about-real-time-transfer/> but still no output. After some searching I've read both opinions that this "is" and "is not" possible on the R4 (TMS570 specifically). Have you heard something about it? Thanks a lot for your help!

★ Like



Erich Styger

on **January 27, 2018 at 13:35** said:

Hi Joel,

"cannot find RTT control block" means it does not find that special memory block

(https://www.segger.com/fileadmin/images/products/Feature_Explains/Real_Time_Transfer/RTT_Schematics_Simple_tn.png) in the RAM of your device. That block is a variable with name `_SEGGER_RTT`. Check your linker map file if that variable is present in your binary. Maybe you did not call the RTT Initialization and the linker has removed it?

If that `_SEGGER_RTT` is present, the Segger RTT client might look at the wrong place. The client uses your device name and knowledge where is the RAM of your device and scans that RAM. I have seen cases where that information was wrong, and the Segger RTT client did not search the correct areas. In the GUI client you can specify in the connection dialog where to look for that `_SEGGER_RTT`: specify the address and it hopefully will find it.

I hope this helps,

Erich

★ Like



ben (@ru4mj13)

on March 26, 2018 at 03:14 said:

Any thoughts what could be the issue using SWO and an Atmel AT-ICE?

This debugger isn't listed in Keil, however, it still works to download and step thru code, if I select "CMSIS-DAP".

The default Trace settings even show SWO and ITM Enabled for all 32 bits of the ports, however, it's all greyed out, so not sure if it's legit.

Share

Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

CMSIS-DAP - JTAG/SW Adapter

Any

Serial No: J41800079539

Firmware Version: 01.27.0082

☒ SWJ Port: SW

Max Clock: 10MHz

SW Device

IDCODE	Device Name	Move
0x0BD11477	ARM CoreSight SW-DP	Up Down

☒ Automatic Detection ID CODE: Device Name: AP: 0x00

☐ Manual Configuration

Add Delete Update

Debug

Connect & Reset Options

Connect: Normal Reset: SYSRESETREQ (I)

☒ Reset after Connect ☐ Stop after Reset

Cache Options

☒ Cache Code ☒ Cache Memory

Download Options

☐ Verify Code Download ☐ Download to Flash

Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

Core Clock: 10.000000 MHz ☐ Trace Enable

Trace Port

Serial Wire Output - UART/NRZ

SWO Clock Prescaler: 8 ☒ Autodetect

SWO Clock: 1.250000 MHz

Error: <SWO Port not supported>

Timestamps

☒ Enable Prescaler: 1

PC Sampling

☐ Periodic Period: <Disabled> ☐ on Data R/W Sample

Trace Events

☐ CPI: Cycles per Instruction ☐ EXC: Exception overhead ☐ SLEEP: Sleep Cycles ☐ LSU: Load Store Unit Cycles ☐ FOLD: Folded Instructions ☒ EXCTRC: Exception Tracing

ITM Stimulus Ports

Enable	Privilege	31	Port	24	23	Port	16	15	Port	8	7	Port	0
0xFFFFFFFF	0x00000008	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
		Port 31..24	<input checked="" type="checkbox"/>	Port 23..16	<input type="checkbox"/>	Port 15..8	<input type="checkbox"/>	Port 7..0	<input type="checkbox"/>				



However, if I manually enable in code, and send a character:

```
ITM->TCR |= ITM_TCR_ITMENA_Msk;  
ITM->TER |= 1; /* ITM Port #0 enabled */  
ITM->PORT[0U].u8 = (uint8_t)'A';
```

I don't ever see anything in Keil's Debug (printf) Viewer. I have a feeling the debugger hardware supports it, and is an Arm configuration issue, or Atmel's WinUSB.sys driver.

Any suggestions? I haven't tried VisualGDB or Atmel Studio or Eclipse, since Keil 64K eval version is so convenient!

★ Like



Erich Styger

on **March 26, 2018 at 08:15** said:

I understand that Keil is convenient for you, but have you considered to use unlimited and 100% free tools (see <https://mcuoneclipse.com/2013/07/20/dyi-free-toolchain-for-kinetis-part-1-gnu-arm-build-tools/>) based on Eclipse and GDB? If you use Keil, this is all based on proprietary stuff and you would need to ask their support. I have used Keil for a while too (not really any more).

★ Like

Pingback: [Overview of MCUXpresso IDE v10.2.0 | MCU on Eclipse](#)



Vinod Pv

on **November 27, 2018 at 05:31** said:

Thanks Erich for this very useful tutorial!

I have one question. Could you clarify?

Without connecting the J-Link tool, you can see any output on SWO line while probing it?

★ Like



Erich Styger



on **November 27, 2018 at 07:30** said:

Yes. But there are two things you have to do:

- initialize the SWO to spit out the SWO data
- having something on the host to get that data.

★ Like

Pingback: [New NXP MCUXpresso IDE V10.3.0 Release | MCU on Eclipse](#)

Pingback: [SWO with NXP i.MX RT1064-EVK Board | MCU on Eclipse](#)



Peter Silie

on **June 6, 2019 at 20:05** said:

Hello all,

is there a possibility to have a PC sampling based profiling in eclipse?

I used this feature in ke*I M*K and liked it very much.

Thanks a lot in advance!

★ Like



Erich Styger

on **June 6, 2019 at 21:03** said:

Yes, there is gprof (see <https://mcuoneclipse.com/2015/08/23/tutorial-using-gnu-profiling-gprof-with-arm-cortex-m/>) and you can use SWO profiling (see for example <https://mcuoneclipse.com/2019/06/03/swo-with-nxp-i-mx-rt1064-evk-board/>).

I hope this helps,

Erich

★ Like



Peter Silie

on **June 7, 2019 at 00:28** said:

Thanks for your quick reply.

How can this be done manufacturer independent?

I'd rather not use gprof to not change the memory and runtime behavior of the system.

A hacky and labour intense way could be to use the swo data from the segger swo terminal and to a address → function lookup using some tool I don't remember (help?)

How the raw data could look like is shown at the bottom of this thread:

https://www.silabs.com/community/mcu/32-bit/forum.topic.html/configuring_itm_for-7ek0

★ Like



Erich Styger

on **June 7, 2019 at 06:50** said:

gprof would be vendor independent for sure, and I use it with stock eclipse too. You are correct that gprof affects runtime behaviour because of the sampling.

I'm using Segger SystemView

<https://mcuoneclipse.com/2015/11/16/segger-systemview-realtime-analysis-and-visualization-for-freertos/> to measure execution time too, but only for specific functions/places as I have to set marker in the code. Usuable for some hot spot optimizations.

Or you can dump all the SWO data as you have found in that article: the challenge might be to feed this data in a form to match the functions. Probably a script could do this based on the symbols of the ELF file.

★ Like



Ken Lin

on **July 11, 2019 at 01:59** said:

When i'm using Segger SWO viewer, i need to input the TRACECLK frequency which supposedly is the same as my CPU frequency. If my HSE (external oscillator) is 8Mhz and my PLL is x9, my SYSCLK is 72Mhz – Is this the “CPU frequency”?

Or maybe there's a more reliable to manually verify TRACECLK frequency in the code so I know what to put down in Segger SWO viewer? I am using STM32F302R8.

★ Like



Erich Styger

on **July 11, 2019 at 15:02** said:

The MXUXpresso IDE has a 'SWO detect clock frequency detection' built in. Yes, in most cases the SWO clock is the CPU clock, and this would be the TRACECLK. The simplest way might be just to hook up an oscilloscope to the SWO pin to verify the actual clock speed.

★ Like



Gary

on **February 12, 2021 at 10:57** said:

To get ITM data out of Cortex-M processor, is single SWO pin enough? Or two pins, SWDCLK + SWO, are needed to make it work?

★ Like



Erich Styger

on **February 12, 2021 at 12:07** said:

The SWO pin is enough.

★ Like

Pingback: [SWO with ARM Cortex-M33 | MCU on Eclipse](#)

Pingback: [Standalone SWO | MCU on Eclipse](#)



John

on **August 13, 2021 at 10:20** said:

Erich,

Thanks for this example, and many of your others they have helped me get many projects off the ground with Eclipse and ARM.

I have an interesting issue when trying to setup SWO using Eclipse, and was hoping to get your thoughts on it.

I have SWO working, and I am getting the printouts in Eclipse, but this only works after I open the J-Link SWO viewer. So if I program and run my project I do not see any printouts at all. If I open the SWO viewer while I have Eclipse connected via the J-Link I get the print outs in Eclipse (and not SWO Viewer) as expected. I have tried running the J-Link SWO viewer with both -swoattach 0, and -swoattach1 thinking that maybe when I was running it there was some configuration going on that Eclipse was not doing properly.

This is all with a Microchip AATSAME70N21 if that makes any difference (so Cortex-M7).

Not a huge deal, mostly just an annoyance, but any light you can shine on this would be greatly appreciated.

Thanks,

John

★ Liked by [1 person](#)



Erich Styger

on **August 15, 2021 at 12:16** said:

Hi John,

It is probably one of the 'tool quirks' you are running into. I faced the same until I initialized SWO in the application and not from any viewer. You probably have seen already my article about it on <https://mcuoneclipse.com/2021/07/12/standalone-swo/>. But the command line would be something like JLinkSWOViewerCL -swoattach on -swofreq 64000 -device LPC55S16 -itimport 0x1

★ Like



pat

on **June 7, 2022 at 10:04** said:

Hi, thanks for the guide and code, really useful and helpful!

One note though, to get it working I had to change the check line 14 on one of your code snippets as below. It worked for me after doing this change. I believe this is correct because we are checking if the port is enabled, i.e. true?

Hope this helps someone 😊

ORIGINAL

```
if ((ITM->TER & (1ul<TER & (1ul<<portNo))==1)
```

★ Liked by [1 person](#)



Erich Styger

on **June 7, 2022 at 10:36** said:

Thanks for your note. I'm not sure about that code piece you have posted, and most likely WordPress did not like the code characters (yes, it is painful)).

But I think it is about line 25 in

https://github.com/ErichStyger/mcuoneclipse/blob/master/Examples/KDS/TWR-K64F120M/TWR-K64F120M_Demo/Sources/swo.c, correct?

That line checks if the corresponding bit in the TER (Trace Enable Register) is cleared, and if so, it returns. Sounds ok to me, unless I'm overlooking something?

★ Like