

 Common Problems

 No STLink device connected

 Failed to start GDB server Error in initialising ST-Link device

 Missing ST-Link drivers

 Debugging Execution

 Enabling Debugging

 Adding Code

 Intro to Debug mode, resume, suspend, and watch variables

 Breakpoints

 The Debugger SWO

 The ST-Link Virtual COM Port

## **Common Problems**

#### No STLink device connected

- Make sure you're logged into the IDE (should say "Hello NAME" in the top bar instead of 'myST")
- An upgrade window may appear saying that the st link is out of date
  - Click 'open in upgrade mode', wait a few seconds
  - Click 'upgrade' button down the bottom and wait for it to finish once finished you can try and run your program again

#### Failed to start GDB server Error in initialising ST-Link device

- Make sure you're logged into the IDE (should say "Hello NAME" in the top bar instead of 'myST")
- An upgrade window may appear saying that the st link is out of date
  - Click 'open in upgrade mode', wait a few seconds
  - Click 'upgrade' button down the bottom and wait for it to finish once finished you can try and run your program again

#### **Missing ST-Link drivers**

- Go to this website <u>here</u>, or just download the zip file below and install the drivers
  - Click the .msi file if you use windows
  - Click the .pkg file if you use mac

en.st-link-server-v2-1-1.zip

 You may then get asked to upgrade your st link after you install the drivers follow the steps in the Failed to start GDB server Error in initialising ST-Link device

# **Debugging Execution**

In this tutorial we will focuses on software debugging, with breakpoints, watch lists, trace analysers, and so forth. STM32CubeIDE has a powerful integrated debugger.

This tutorial is based off of Hammond's found here.

#### **Enabling Debugging**

Note: You may need to enable debugging via your CubeMX config. I had it enabled already since it's included in the default configuration for my development board. If you need to enable it, do this in the CubeMX view:

₩ <u>hello.ioc</u> 🛛 🕻	main.c 🔝	startup_stm	32f303retx.s	
Pinout	& Configuration	on	Clock Configuratio	n
			Additional Software	✓ Pi
Q	~ 🔅		SYS Mode and Configuration	
Categories A-3	>Z		Mode	
System Core  DMA GPIO IWDG INVIC SYS SYS Tec WWDG	~	System System Timebas 2. Sec.	Gerial Wire em Wake-Up 1 em Wake-Up 2 se Source SysTick lect based on f ring	v

Save your config using *File>Save* and regenerate the code.

### Adding Code

Have a play by debugging the code. But first, add something to debug, since there's not much going on here. Add a few snippets of code to calculate some prime numbers.

Add a few lines of code to a few of those USER code blocks in main.c, as detailed here:

```
// . . .
/* USER CODE BEGIN Includes */
#include <stdbool.h>
/* USER CODE END Includes */
// . . .
/* USER CODE BEGIN PD */
```

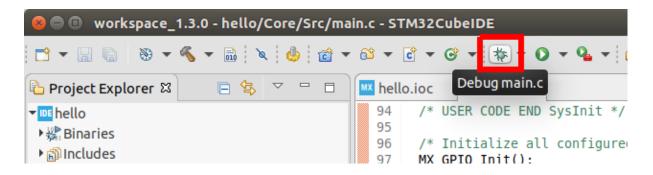
```
#define PRIMES_LEN 62
/* USER CODE END PD */
// . . .
/* USER CODE BEGIN PV */
uint16_t primes[PRIMES_LEN] = {0};
/* USER CODE END PV */
// . . .
/* USER CODE BEGIN 0 */
//check if a number is prime
bool is_prime(uint16_t v) {
  for(uint16_t i=2; i<(v/2 + 1); i++) {</pre>
    if(v % i == 0) return false;
  }
 return true;
}
/* USER CODE END 0 */
// . . . (inside int main())
  /* USER CODE BEGIN 2 */
  //calculate a list of primes:
  uint16_t prime_index = 0;
  for(uint16_t i = 2; i < 300; i++) {</pre>
    if(is_prime(i)) {
      primes[prime_index] = i;
      prime_index++;
    }
  }
  /* USER CODE END 2 */
```

Compile and execute this; there won't be any observable changes (except for maybe the slightest delay before the LED starts blinking the first time).

First, look at the end result, then observe the looping calculation using a breakpoint.

#### Intro to Debug mode, resume, suspend, and watch variables

To simply observe the end result, we launch the program using the Debug mode, with the **Debug** button.



Firstly, STM32CubeIDE will change into its *debug* perspective. There may be a popup asking about this.

Notice that the dev kit in front of you has the communication LED blinking continuously, but the user LED isn't blinking yet. This is because the program is not actually running yet. To run the program, press the 'Resume' key to get started. The resume key, as well as the other debug control keys, will have appeared due to the *debug* perspective activating.

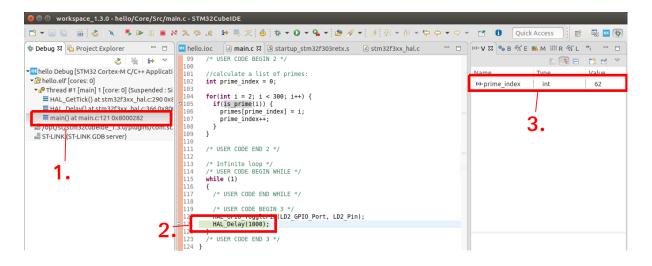


Once you notice the dev board's LED light is blinking, pause the execution by pressing the suspend key:

😕 🗖 🗊 workspace_1.3.0 - hello/Core/Src/mai	n.c - STM32CubeIDE
🔁 🗝 🔚 💼 🕹 🔍 🔍 🕒 🛄 🗖 🕺	1 3. 3 .R ið 🤜 😿 🎍 🕸 🔻 🔾
🎄 Debug 🕱 🔁 Project Explorer 🛛 Suspend	🔤 hello.ioc 🛛 🖻 main.c 🛱 🔝 startup_s
🕹 💥 it 🗸	103 104 for(int i = 2; i < 300; i++
✓ Image: weighted the second seco	pilos if(is_prime(i)) { primes[prime index] = i

Notice that the IDE immediately throws you somewhere in the C code for your project, almost certainly in something to do with the HAL\_Delay function. That's where the program was when suspend was hit.

Get back home by going to the left Debug panel and selecting main().



Note that the IDE highlights the function that is currently being executed (2) and presents a list of variables in the current scope (3).

Examine the contents of the global variable by double-clicking it.

Image: Section of the sectin of the section of the section of the section of the	😣 🖱 🗉 workspace_1.3.0 - hello/Core/Src/ma	in.c - STM32Cube	IDE					
Image: Strate Context M C/C++ Application         Image: Strate Context M C/C++ Applicati	🖆 🕶 📓 🕼 🕹 🔍 🖡 🕨 🗉 🖷 🌢	* 3. 15. je i⇒	≅ ≈ 🕹 🚸 + • • • •	2 A + 3 4 + 6 + 4	⇔ - ⇔ -	🖬 🚺 🛛 Quick	Access	🔁 🔤 🔯
Thelio Debug [STM32 Cortex M (C/+ Applicat * Thelio Leff (core: 0) (Suppended: St # HAL_CetTick() at stm327/sx, balc:280 006 # HAL_CetTick() at stm327/sx, balc:280 007 # USER CODE EBOIN PM */ # / USER CODE E	🎋 Debug 🛿 🏠 Project Explorer 🛛 🗖 🗖	hello.ioc	े main.c 🛱 🗟 startup_stm32f303	Bretx.s 🗟 stm32f3xx_hal.c		(x)= V 🖾 💁 B 🕸 E	🛋 M 🕮 R 🕸 L	"1 🗆 🗆
* UseR code Exp PD */ ** Detailed bebug [STM32 cortex-M c/C++ Application 374 / USER code Exp PD */ ** Detailed in [Core: 0] (Superedd: Si ** AL_Cleay() as than C:210 Accode EXD: PD */ ** Detailed in the state of the state	🕹 💥 🍺 🔻		CODE BEGIN PTD */				約 📑 🖻	📑 🖻 🗵
<pre>HAL_CetTrk() at stm32f3xx_balc.250 otc HAL_Dolay() at stm32f3xx_balc.256 0xe0 Emmain() At thm32f3xx_balc.256 0xe0 # // USER CODE EBD PD */ # // USER CODE EBD PD */ # // USER CODE EBD PM */ # USER CODE EBD PM */ # USER CODE EBCIN PW */ #</pre>	→ → → hello.elf [cores: 0]	32 /* USER ( 33						
<pre>3 / Opt/st/stm32cubeide_1.3.0/plugins/com.st 3 / STELINK (STELINK GDB server) 4 / / USER CODE EBGIN PM */ 4 / / Private BGE ROP M */ 4 / / Private variables 4 / / Private variables 5 / / / / / Private variable</pre>	■ HAL_GetTick() at stm32f3xx_hal.c:290 0x8 ■ HAL_Delay() at stm32f3xx_hal.c:366 0x800	35 /* USER ( 36 <b>#define F</b> 37 /* USER (	CODE BEGIN PD */ PRIMES_LEN 62		-			
43       /* Private variables         44       /* Private variables         45       UMAT HandleTypeDef huart2;         46       /* USER (Conset Series Private	/opt/st/stm32cubeide_1.3.0/plugins/com.st.	39⊖/* Privat 40 /* USER ( 41	CODE BEGIN PM */					
47       uintlôt t orines petimes LEN] = {0};         48       / USER C Expression Type Value         50       /* Privat ©primes[0] uintlôt 2       0x20000028 <primes>         51       void Syst       primes[0] uintlôt 2       2         52       static vo       primes[2] uintlôt 7       3         54       /* USER C ≥ primes[2] uintlôt 7       5         55       &gt; orimes[3] uintlôt 7       7         56       /* USER C Hame : primes       0         57       Detall: (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 56% /* Privat Default: 0x2000028 cyrimes&gt;         59       /* USER C Decimal: 336870852       primes&gt;         61       //check i Binary: 1000000000000000000000000000000000000</primes>		43 44 /* Privat 45 UART_Hand	te variables lleTypeDef huart2;					
49     Expression     Type     Value       50     /* Primes     unit16_t[62]     0x2000028 sprimes>       51     void Syst     * primes[0]     unit16_t     2       52     static void     * primes[2]     unit16_t     3       53     /* USER C     ** primes[3]     unit16_t     7       56     /* USER C     Name: primes     primes[3]     unit16_t       57     Default:s(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 58@ /* Privat     Default:s02000028 sprimes>       59     /* USER C     Default:s02000028 sprimes>     Default:s02000028 sprimes>       61     //check 1     Binary:1000000000000000000000000000000000000		47 uint16_t	<pre>primes[PRIMES_LEN] = {0};</pre>					
51       void Syst       >> primes[0]       uint16_t       2         53       static vo       >> primes[1]       uint16_t       3         54       /* USER C       >> primes[2]       uint16_t       5         55       /* USER C       >> primes[3]       uint16_t       7         56       /* USER C       >> primes[3]       uint16_t       7         57       Default:2(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, Default:0820800208       cprimes>         59       /* USER C       Decimal:338870952       Hex:0x2000028       primes>         60       J//check I       Binary:180090900000000000000000000000000000000		49	Expression	Туре	Value			
52       static vo       ™ primes[0]       Unit(0_L       2         53       static vo       ™ primes[1]       Unit(1_L       3         54       /* USER C       № primes[3]       Unit(1_L       5         55       W primes[3]       Unit(1_L       5         56       /* USER C       Name : primes[3]       Unit(1_L       7         57       Details:(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 59%       9 /* Primes       9 /* USER C         59       /* USER C       Decimal:15326909028 <primes>       6       1 //check 1       Binary:108008090800800000000000000000000000000</primes>		50 /* Privat				0028 <primes></primes>		
53       /* USER (       ≥ primes[2]       uint10_t       5         55       /* USER (Name : primes)       uint10 t       7         56       /* USER (Name : primes)       petails:(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 589/* Privat       Details:(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 599/* Privat         57       Details:(2, 10000008000000000000000000000000000000		52 static vo						
55       >> primes[3]       uint10 t       7         56       /* USER C Name : primes       Detail:s(2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, Default:s020000028 <primes>         57       /* USER C Decimal:S08070952       +         60       //check is 308070952         61       //check is 308070952         62       bool is p         63       -         63       -         64       -         65       -         66       -         67       -         68       -         69       -         60       -         61       -         62       -         63       -         64       -         65       -         61       -         62       -         63       -         64       -         65       -         66       -         7       -         67       -         68       -         69       -         60       -         61       -         &lt;</primes>								
57       Details:{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 580 % Files Decimal:536870952         59       /* USER C       Decimal:536870952         61       //check 1       Binary:10000000800000000000000000000000000000		55	🔊 primes[3]		7			
580 /* Privat       Default:0x2000028 <primes>         590 /* USER C       Decumal:536870952         60       Hex:0x2000028         61 //check 1       Binary:1000000000000000000000000000000000000</primes>				12 17 10 22 20 21 27	41 42 47	52 50 61 67	71	
60       Hex:0x20000238         61 //check I       Binary:1000000000000000000000000000000000000					41, 43, 47	, 53, 59, 61, 67,	/1,	
61       //check i       Binary:1000000000000000000000000000000000000								
Image: Structure       Image: Structure         Image: Structure				000000101000				
© Console X Console X Problems    Executables    Debugger Console    Memory								
hello Debug [STM32 Cortex-M C/C++ Application] ST-LINK (ST-LINK GDB server) Download verified successfully			-	burger Console 👖 Memory	<b>•</b> ×			<b>⊳</b>
Download verified successfully						· · · · · · · · · · · · · · · · · · ·		
			ý <u>-</u>					
Writable         Smart Insert         47:16[6]		Download verif:	ied successfully					
Writable Smart Insert 47:16[6]								
	·	IL		Writable Smart Insert	47:16[6]			

Or in a more convenient manner (since this is a big array) by right clicking on the variable primes and selecting Add Watch Expression (and then pressing OK)

😣 😑 💿 workspace_1.3.0 - hello/Core/Src/ma		≋ 🎂 🌣 ▾ 🔿 ▾ 💁 ▾ 🥭 🖋	• 🕖 🖗 • 🖗 • 🏷 🗢	• 🗘 •	📑 🚺 Qui	k Access	🕸 🔤 🙀
🛠 Debug 🛱 🔓 Project Explorer 🛛 🗖 🗖	🔤 hello.ioc 🛛 🗟 mai	n.c 🛱 🔝 startup_stm32f303retx.s	stm32f3xx_hal.c		(x)= V 🖾 💁 B 🖧 E	➡ M 배 R 🕸 L	»1 🗆 🗆
🕹 💥 it 🗸	46 /* USER CODE I	BEGIN PV */				ها 🛋 🖾	1 1
■ hello Debug [STM32 Cortex-M C/C++ Applicati	48 /* USER CODE 49 50 /* Private f	<ul> <li>✓ <u>Undo Typing</u></li> <li>Revert File</li> <li>■ Save</li> </ul>	Ctrl+Z Ctrl+S		Name ⇔•prime_index	Type int	Value 62
= HAL_UetTick() at stim3213XX_hatC290 0XE = HAL_Delay() at stm3213XX_hatC300 0XE0 = main() at main.c:121 0X8000282	53 static void 53 static void 54 /* USER CODE 55 56 /* USER CODE 57 58@/* Private u 59 /* USER CODE	Open Declaration Open Type Hierarchy Open Call Hierarchy Quick Outline Quick Type Hierarchy Explore Macro Expansion	F3 F4 Ctrl+Alt+H Ctrl+O Ctrl+T Ctrl+T				
	60 61 //check if a 62⊕bool is_prim 63 for(uint16 64 if(v % i 65 } 66 return tru	Toggle Source/Header Open With Sho <u>w</u> In ☆ Cu <u>t</u>	> Shift+Alt+W > Ctrl+X				
	67 } 68 /* USER CODE 69 70⊕/** 71 * @brief 72 * @retval 73 * @retval 73 ⊕ int main(voi	<u>Copy</u> <u>Paste</u> Quick Fix Source Surround <u>With</u> Refactor	Ctrl+C Ctrl+V Ctrl+1 Shift+Alt+S > Shift+Alt+Z >				
	75 { 76 /* USER C0 77 78 /* USER C0 79	Declarations References Search Text Resource Configurations	>				
	E Console 🛱 🔝 Pro	Step Into Selection	Ctrl+F5	• ×	💥 🖹 🖬 🕑 🕻		📑 👻 🗖
	Tello Debug (31H32 C	→] Run to <u>L</u> ine . <b></b> <u>M</u> ove to Line <b>.</b> Resume at Li <u>n</u> e	Ctrl+R				
	Download verified	<ul> <li>Add Watch Expression</li> <li>Run As</li> <li>Debug As</li> <li>Profile As</li> </ul>	> > > t Insert	47 : 16 [6]			

(x)= Variables 💁 Brea	kpoints 🞯 Expressions 🛱	🛋 Modules 🕮 Registers 🕸 Live
	A Expressions	
		1. Sta
Expression	Туре	Value
<b>▼</b> /= primes	uint16_t [62]	0x20000028 <primes></primes>
⇔=primes[0]	uint16_t	2
Ø=primes[1]	uint16_t	3
(X)=primes[2]	uint16_t	5
(X)= primes[3]	uint16_t	7
(×)=primes[4]	uint16_t	11
Ø=primes[5]	uint16_t	13
Ø=primes[6]	uint16_t	17
(X)= primes[7]	uint16_t	19
Ø=primes[8]	uint16_t	23
(×)=primes[9]	uint16_t	29
(X)=primes[10]	uint16_t	31
Ø=primes[11]	uint16_t	37
(X)=primes[12]	uint16_t	41
(X)=primes[13]	uint16_t	43
(×)=primes[14]	uint16_t	47
65tfam1		FD

That then adds it to the Expressions menu on the right:

Press the Red Stop button now in the top menu, and watch the execution of the prime calculation loop.

### **Breakpoints**

In addition to running and suspending execution, we can also ask the program to suspend at a point of our choosing. This is known as creating a breakpoint.

In STM32CubeIDE, you do this by double-clicking on the red bar next to the line numbers, which will cause a small blue breakpoint indicator dot to appear.

```
99 /* USER CODE BEGIN 2 */
 100
 101 //calculate a list of primes:
102
      int prime_index = 0;
103
104 for(int i = 2; i < 300; i++) {</pre>
       if(is_prime(i)) {
105
106
          primes[prime index] = i;
          prime index++;
107
      }
108
109
      }
110
 111
      /* USER CODE END 2 */
```

### Double click on the red bar next to the line numbers.

### A blue breakpoint dot will appear.

Now, without changing anything else, launch the debug mode again.

This time, when you hit resume, you'll notice that the program executes and then automatically halts when it reaches your breakpoint.

🕸 Debug 🛿 🍋 Project Explorer 🛛 🗖 🗌	🏧 hello.ioc 🛛 🖻 main.c 🛱 🕒 startup_stm32f303 🔂 stm32f3xx_hal.c 👘 🗖	🗱 🕬 🕼 🕼 🖾 🔊 🗰 🕅 🗰 R 🖓 Li 🎟 S 🦳 E
★ + ~ Thello Debug [STM32 Cortex-M C/C++ Applicati The at 1 [main] 1 [core: 0] (Suspended : Br main() at main.c:105 0x8000240 Jopt/st/stm32cubeide_1.3.0/plugins/com.st. ST-LINK (ST-LINK GDB server)	as       HAL_Init();         bs       /* USER CODE END Init */         bs       ystemClock_Config();         as       HAL_Init();         ystemClock_Config();       ystemClock_Config();         as       HAL_Init();         ystemClock_Configured peripherals */         ystemClock_Configur	Image: Second

Now if press suspend again, it will loop and stop at this same function.

🌾 Debug 🛱 🏠 Project Explorer 🛛 🗖 🗖	hello.ioc 🕼 main.c 🏼 🕼 startup_stm32f303 🕼 stm32f3xx_hal	l.c 🗆 🗆	(x)= V ⁰o Br 😤 E 🖾	🛋 M 👭 R 🛠 Li 🎟	■ S 👘 🗖
	<pre>neu.ooc [2] Main.c X [2] Startup_stm327303 [2] stm3273XX_main 33 HAL_Init(); 44 55 /* USER CODE BEGIN Init */ 66 77 /* USER CODE END Init */ 67 74 USER CODE END System clock */ 69 67 /* Initialize all configured peripherals */ 75 75 /* Initialize all configured peripherals */ 76 /* Initialize all configured peripherals */ 77 MX_GPIO_Init(); 78 MX_USART2_UART_Init(); 79 /* USER CODE END SysInit */ 70 // Calculate a list of primes: 71 initialize all configured peripherals */ 70 // Calculate a list of primes: 71 initialize all configured peripherals */ 76 /* Initialize all configured peripherals */ 77 MX_GPIO_Init(); 78 MX_USART2_UART_Init(); 79 /* USER CODE END 2 */ 70 70 // Calculate a list of primes: 70 // Calculate a list of primes: 70 // Calculate a list of primes: 71 // Calculate a list of primes: 72 /* Initialize index = 0; 73 /* USER CODE END 2 */ 74 /* USER CODE END 2 */ 74 /* USER CODE END 2 */ 74 /* USER CODE END 2 */ 75 /* Thinite loop */ 76 /* Console S2 [*] Problems</pre>	emory	Expression	Me and R vie L      Vie L	Value 0x200 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The first index of primes has changed, and it's highlighted the changed variable.

Hitting resume again:

۶ Debug 🛱 🎦 Project Explorer 🛛 🗖 🗌	🖷 hello.ioc 🛛 🗟 main.c 🛱 🔕 startup_stm32f303 🔂 stm32f3xx_hal.c 👘 🗖 📖	⊨V 🍫 Br 🖗 E 🕱 🛋 M ﷺ R ଐ Li 📟 S 🦳 E
	<pre>83 HAL_Init(); 84 85 /* USER CODE BEGIN Init */ 86 7 /* USER CODE END Init */ 87 88 99 /* Configure the system clock */ 90 SystemClock_Config(); 91 /* USER CODE BEGIN SysInit */ 93 /* USER CODE BEGIN SysInit */ 94 /* USER CODE END SysInit */ 95 /* Initialize all configured peripherals */ 97 MX_GFIO_Init(); 98 /* USER CODE END SysInit */ 99 /* USER CODE BEGIN 2 */ 100 //(calculate a list of primes: 101 int prime_index = 0; 103 for(int i = 2; i &lt; 300; i++) { 104 for(int i = 2; i &lt; 300; i++) { 105 prime_index++; 106 } 106 /* USER CODE END 2 */ 107 prime_index++; 108 } 109 } 109 } 109 /* USER CODE END 2 */ 102 /* Infinite loop */ 103 /* USER CODE BEGIN WHILE */ 104 for(int i = 1) /* USER CODE BEGIN WHILE */ 105 while (1)</pre>	V       • • • • • • • • • • • • • • • • • • •

Now the second index of primes has changed, and the changed variable is highlighted once more!

Keep pressing the resume button, and you'll see it slowly calculate the array.

Now, let's quit the debugger and move on. You can delete the breakpoint again by double-clicking the blue dot. Note that if you right-click on the red column, you can also toggle and create breakpoints this way. This also brings up advanced breakpoint options, including breakpoint conditions and breakpoint types.

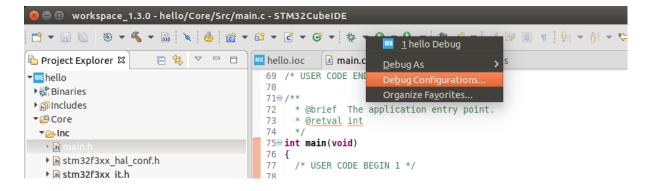
## The Debugger SWO

If properly configured, you can output arbitrary strings directly to the debugger via the programmer, rather than sending them via any other peripherals. It's a bit like a virtual UART that you can send data to.

This is a little involved to set up, and it can be worth simply using a UART if you must send out strings of characters to help your debugging, but I'll step through the basics here using STM32CubeIDE.

First, we must configure the reception clock rate. This is done via the debug configuration menu.

Press this:



Go to the Debugger tab, Enable SWV (Serial Wire Viewer), then change your clock rate to the FCLK from earlier (remember when we chose the clock rate for all of our components?)

You may leave the SWO Clock dropdown set to its maximum.

8 🗉	Debug	Configu	rations
-----	-------	---------	---------

Create, manage, and run configurations

🗋 🖻 🗫 🗎 🗶 🗐 🗞 🕇	Name: hello Debug						
	Aai S Debugger Startup Source Common						
C/C++ Application	GDB Connection Settings						
C/C++ Attach to Application	O Autostart local GDB server Host name or IP address localhost						
<ul> <li>C/C++ Postmortem Debugger</li> <li>C/C++ Remote Application</li> </ul>	O Connect to remote GDB server Port number 61234						
GDB Hardware Debugging							
Launch Group	Debug probe ST-LINK (ST-LINK GDB server) 🔻						
Launch Group (Deprecated) STM32 Cortex-M C/C++ Applicat	GDB Server Command Line Options						
hello Debug	Interface						
	SWD JTAG						
	ST-LINK S/N						
	Frequency (kHz): Auto						
	Access port: 0 - Cortex-M4						
	Reset behaviour						
	Type: Connect under reset  Halt all cores						
	Ceria Wire Viewer (SWV)						
	Clock Settings						
	swo clock: 2000 wHz Match your FCLK						
	swo clock. 2000 + kHz frage fragme operation						
	Port number: 61235 freq from earlier						
	☑ Wait for sync packet						
	Device settings						
	Debug in low power modes:						
	Suspend watchdog counters while halted: No configuration						
	Misc						
	🗹 Verify flash download						
	S Enable live expressions						
	Log to file:     //home/hammond/STM32CubeIDE/workspace_1.3.0/hello/Debug/s     Browse						
	External Loader:     Scan     Initialize						
	□ Shared ST-LINK						
Filter matched 9 of 9 items	Revert Apply						
?	Close Debug						

Now press Apply/Close.

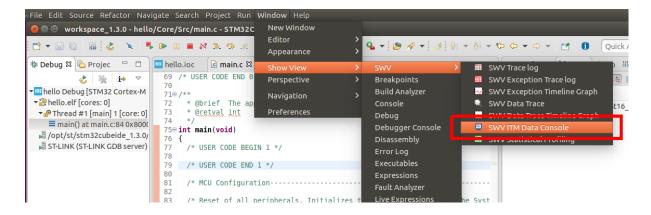
Add a test to send some characters. In the main loop, add the following:

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */
HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
HAL_Delay(1000);
ITM_SendChar('!');
}
/* USER CODE END 3 */
```

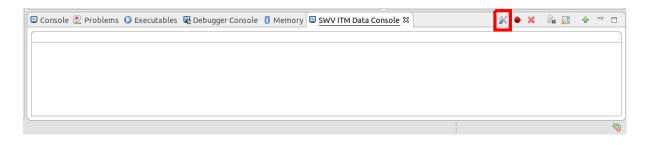
**ITM\_SendChar** is a special function that sends a character to the debugger's serial viewer. You shouldn't need to **#include** anything new or special to use this.

You can now launch the debug session as before. But, before you press **Resume** to get it started, we need to enable a few more debugging options.



First, open the ITM data console through the Window menu.

This will bring up the ITM data console window. Now, enter the configuration menu:



Enable ITM stimulus port 0:

😣 🗊 Serial Wire Viewer setti	ngs for hello Debug		
Clock Settings Core Clock: 72 MHz Clock Prescaler: 36 SWO Clock: 2000.0 kHz	SLEEP: Sleep cycles	Exception overhead	Sampling Enable Resolution: 16384 T Cycles/sample mestamps Enable Prescaler: 1
Data Trace			
Comparator 0 Enable	Comparator 1 — Enable	Comparator 2 Enable	Comparator 3
Var/Addr: 0x0	Var/Addr: 0x0	Var/Addr: 0x0	Var/Addr: 0x0
Access: Read/Write	Access: Read/Write 💌	Access: Read/Write	▼ Access: Read/Write ▼
Size: Word	Size: Word 👻	Size: Word	▼ Size: Word ▼
Generate: Data Value 🔻	Generate: Data Value 🔻	Generate: Data Value	▼ Generate: Data Value ▼
ITM Stimulus Ports Enable port: 31 🗌 🗌 💭 💭 Privileged only ports: 🗌 Port :	3124 Port 2316 Port 158	○ ○ 16 15 ○ ○ ○	Cancel OK

#### Press OK.

Port 0 will appear in the console view. Now press **Start** - this contacts your microcontroller and adjusts some registers internally to enable this mechanism:

🛢 Console 🖹 Problems 🔉 Executables 📓 Debugger Console 🔋 Memory 🗐 <u>SWV ITM Data Console</u> 🛿	* • ×	÷	
Port 0 🕱			
			<b>10</b>

Now, and only now, can you press Resume. You'll notice your Port 0 terminal slowly start filling with exclamation marks (since that's the character we're sending).

🖻 Console 🖹 Problems 📀 Executables 📓 Debugger Console 🔋 Memory 🗐 SWV ITM Data Console 🕱	Ж	۲	×	R	٠	
Port 0 X						
						5

As you can imagine, this is pretty handy when your design might not have a free UART for debugging.

Without too much difficulty, we can also spin up a custom **printf** function for debugging. There's a few options for this, but the preferred approach is to actually create a **debug\_printf** function, like so:

```
Adding more to main.c, as detailed:
```

```
// . . .
/* USER CODE BEGIN Includes */
#include <stdbool.h>
#include <stdio.h>
#include <stdarg.h>
/* USER CODE END Includes */
// . . .
/* USER CODE BEGIN PD */
#define PRIMES LEN 62
/* USER CODE END PD */
// . . .
/* USER CODE BEGIN PV */
uint16_t primes[PRIMES_LEN] = {0};
/* USER CODE END PV */
// . . .
/* USER CODE BEGIN 0 */
//check if a number is prime
bool is_prime(uint16_t v) {
 // . . .
}
// debug_printf sends a max of 256 characters to the ITM SWO
// It uses a variable length argument , same as normal print
// Call this function as if it was printf
void debug_printf(const char *fmt, ...) {
```

```
char buffer[256];
  va_list args;
  va_start(args, fmt);
  vsnprintf(buffer, sizeof(buffer), fmt, args);
  va_end(args);
  uint16_t i = 0;
  while(buffer[i] != '\0') {
    ITM_SendChar(buffer[i]);
   i++;
  }
}
/* USER CODE END 0 */
// . . . (inside int main())
 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  uint32_t count = 0;
  while (1)
  {
   /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
   HAL_GPI0_TogglePin(LD2_GPI0_Port, LD2_Pin);
   HAL_Delay(1000);
    debug_printf("Hello debugger, this is iteration %d\r\n",
   count++;
  }
  /* USER CODE END 3 */
```

Now we save, build, and debug that. Remember to start the ITM trace before pressing Resume !

📮 Console 🖹 Problems 🜔 Executa	oles 🖳
Port 0 🛱	
Hello debugger, this is iteration 0	
Hello debugger, this is iteration 1	
Hello debugger, this is iteration 2	
Hello debugger, this is iteration 3	
Hello debugger, this is iteration 4	
Hello debugger, this is iteration 5	
Hello debugger, this is iteration 6	

### **The ST-Link Virtual COM Port**

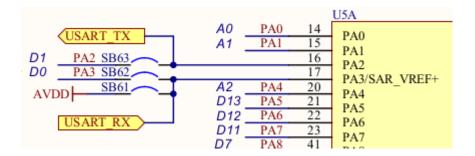
The inbuilt ST-Link v2.1 interface that we've been using for programming and debugging *also* includes a *virtual COM port*. The COM port uses drivers which are included natively in most operating systems (including Windows and Ubuntu Linux).

Running dmesg | grep tty in the terminal, we can see it has been made available as /dev/ttyACM0 thus:

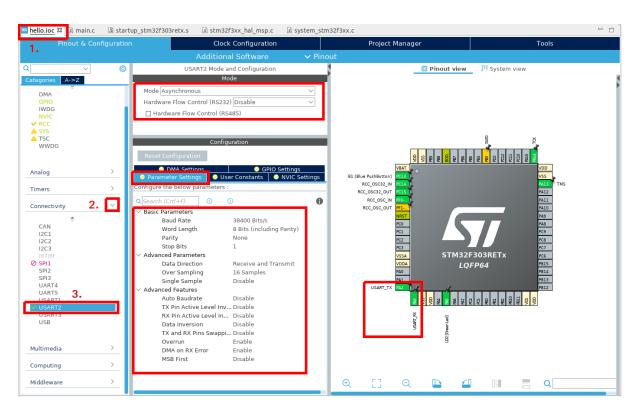
```
$ dmesg | grep tty
[30318.354183] cdc_acm 3-10.4:1.2: ttyACM0: USB ACM device
```

This is really handy for your user applications, as this virtual COM port is wired directly onto one of the USART peripherals on the nucleo board! Recall from the CubeMX view that pins *PA2* and *PA3* were automatically configured as a USART for us.

A quick sanity check to make sure this makes sense by looking on the schematic:



Sure looks like they're connected to a USART (tracing it through the rest of the schematic shows them connected to the ST-Link V2 programmer). So let's



quickly jump back into the CubeMX view and see how the port was set up:

It's configured as asynchronous, at 38400 baud, 8 data bits, no parity, 1 stop bit.

We could change these settings now if we wanted to. The virtual COM port works the same as any other USB to serial adapter, and so any baud rate and config can work with it.

Let's send some characters to it.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
uint32_t count = 0;
while (1)
{
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
    HAL_GPIO_TogglePin(LD2_GPI0_Port, LD2_Pin);
    HAL_Delay(1000);
    debug_printf("Hello debugger, this is iteration %d\r\
    HAL_UART_Transmit(&huart2, (uint8_t*)"Hello world, th
```

count++;

```
}
/* USER CODE END 3 */
```

We now build and download that, before running Minicom:

\$ minicom -b 38400 -D /dev/ttyACM0

```
Melcome to minicom 2.7
Welcome to minicom 2.7
OPTIONS: I18n
Compiled on Nov 15 2018, 20:18:47.
Port /dev/ttyACM0, 09:26:51
Press CTRL-A Z for help on special keys
Hello world, this is a UART
```

If we wanted, we could now make a <code>usart\_printf</code> like our <code>debug\_printf</code> from earlier.