STM32 – Debugging

Petr Novák / novakpe@cvut.cz / 2022-04-22

Obsah

1	Úvod	. 1
2	Breakpoint	. 1
3	Zobrazení proměnných	. 1
4	OnLine zobrazení proměnných při běhu	. 4
5	Zobrazení FLASH a SRAM	. 6

1 Úvod

Zde jsou pospány některé možnosti, a techniky jak ladit program v procesoru STM32 za pomoci STM32CubeIDE.

2 Breakpoint

Breakpointy lze vkládat (skoro) na libovolné řádky kódu. Je vhodné je vkládat na výkonné řádky kódu a nikoli na příklad na definici nějaké proměnné (občas jsou s tímto problémy). Breakpoint je zobrazen vlevo na začátku programového řádku jako celkem malý puntík.



Vloží se dvojklikem na tuto modrou čáru. Opětovným dvojklikem se odstraní.

Poznámky:

 Před odstraněním programového řádku / kódu s breakpointem, doporučuji nejprve odstranit tento breakpoint a teprve potom programový kód. Občas se breakpoint po odstranění kódu někam ztratí / přesune a dělá to problémy. Často pomůže celý projekt zavřít a zase jej otevřít.

3 Zobrazení proměnných

Po zastavení programu (například dosažením breakpointu) si zle prohlížet definované proměnné. V naprosté většině případů stačí podržet ukazatel myši nad požadovanou proměnnou a zobrazí se dialog obsahující její hodnotu.

onnee eeu a	(lend read tx char handle)				
rtReadTX(Ex	pression	Туре			
	(x)= connected	volatile int			
art RX ha					**
if (conne					
rtReadRXC					>
					(
					> (
onnected					-
<			>		<
LED Off(^{Na}	ame : connected		^		
bleNotifi	Details:0				
	Default:0			•	
	Decimal:0				
	Hex:0x0				
k if the	Binary:0			,	
K II THE	Octal:0				<
ems 🖸 Execut			×		32 8
	1		2		

Aktuální hodnota proměnné je zobrazena v několika číselných soustavách. Takto lze však zobrazit obsah pouze jedné proměnné a to ještě těch základních číselných.

Pokud je potřeba zobrazit více proměnných současně, nebo těch co mají poněkud složitější obsah (pole, struktury, ...) tak je vhodný seznam proměnných / hodnot na pravé straně (zobrazen samozřejmě pouze při po-zastavení běhu dříve spuštěného programu).

workspace_1.2.0 - SampleApp/Ap	plication/User/app_bluenrg_ms.c - 51Ms2CubelDE			
<u>File Edit Source Refactor Nav</u>	igate Search <u>P</u> roject <u>R</u> un <u>W</u> indow <u>H</u> elp	/		
📑 🕶 🔛 🕼 🛍 🕹 🕶 🔍	🖡 🕪 🗉 🛢 🗱 3. 👁 요) 🕪 🗏 🛞 🕸 * 🔾 * 隆 🦧 * 🧶 🥙 🖉 * 🚺 * 🖥 * 🏷 수 * 수 * 🔂 🚺		Q	1 🖻 🔞 🏘 🛄
🎄 Deb 🔀 🛅 Proj 📟 🗖	🖻 main.c 👔 app_bluenrg 🖄 🖻 sample_servi 🖻 bluenrg_gat 🚯 startup_stm 🕼 hci_const.h 🤭 🖳	🗱 🗘 🗣 🕹 🖓 E 😒 🔳	🗛 M 🐰 R 🍕	(L 🐴 💙 🗆
🖻 🛬 i+ 🕴	266 }		6 🖂 🔶 🕽	🍇 📑 👌 🕴
v III SampleApp [STM32 Cortex-N	267	Expression	Type	Value
✓	268 if (BLE Role == (LIENT)	done		Error: Multiple error
V 🧬 Thread #1 [main] 1 [cor	269 1	dataValues		Error: Multiple error
user_notify() at sam	270 /* Start TX handle Characteristic dynamic discovery if not yet dyna	accelDataValues		Error: Multiple errol
hci_user_evt_proc()	270 / State TA handle characteristic dynamic discovery if hot yet the	dataOut		Error: Multiple error
main() at main c:10(271 IT (connected ad read read (x char handle){	data		Error: Multiple error
arm-none-eabi-gdb (8.3.1	<pre>2/2 startReadIXCharHandle();</pre>	moje		Error: Multiple error
ST-LINK (ST-LINK GDB ser	2/3 }	ntcuevice	In advertis	Error: Multiple error
	274 /* Start RX handle Characteristic dynamic discovery if not yet done	(the exit type	uint8 t	0.50
	275 else if (connected && !end_read_rx_char_handle){	00- bdaddr type	uint8 t	1 \001'
	276 startReadRXCharHandle();	> 🥟 bdaddr	tBDAddr	[6]
	277 }	(v)- data_length	uint8_t	17 '\021'
	278	Carlos Deci	uint8_t [17]	0x200002ea <hcifei< td=""></hcifei<>
	279 if (connected && end read tx char handle && end read rx char handl	🔍 🌵 Add new expression		
	280 1			
	281 BSP LED Off(LED2): //end of the connection and chars discovery nha	<		
	analleNotification().			
	203]			
	204 }			
	#286 /* (heck if the User Button has been nushed */	4		× *
	🖸 Console 🖾 🎦 Problems 💽 Executables 🙀 Debugger Console 🚺 Memory	📕 🗶 🔆 🛼 🚂 🐼 🚺	- C - C	J • 📑 • • •
	SampleApp [STM32 Cortex-M C/C++ Application]			
				^
	Download verified successfully			*
·				>
	Writable Smart Insert 278:1:816	9		

Přidat další proměnnou do seznamu zobrazených lze klikem na zelené "plus", zobrazí se editační pole, do něhož lze název proměnné vepsat (ideálně zkopírovat ze zdrojového kódu / textu). V seznamu / tabulce jsou tři sloupce:

- Název Název proměnné zapsaný / zkopírovaný do tabulky.
- Typ Jakého typu je zobrazená proměnná.
- Hodnota Jakou má proměnná aktuálně hodnotu.

Pokud se jedná o proměnnou běžného typu (int, char, …), tak je zobrazení její hodnoty vcelku jednoduché a přehledné. Dokonce i u statického pole lze jeho prvky zobrazit jeho snadným rozbalením (značka pro rozbalení před názvem proměnné). Jsou však proměnné, jejichž zobrazení je potřeba vhodně upravit aby se v jejich datech člověk vyznal:

Odkaz do paměti na pole hodnot

Pokud je potřeba zobrazit podle hodnot, na který je pouze odkaz (například vstupující do funkce) zle jej rovněž zapsat do seznamu proměnných. Bohužel po rozbalení je zobrazena pouze první hodnota, protože zobrazovač neví jak je pole dlouhé.

	in manue	in abbinoning'num in samble servicer. IN innendigablarm in station sunder in incliconsen		(10) Y - O D - XE L - CO		98 C 1 D
8	375	{	^	X.	야 🖂 🕂	🗙 🧏 📑 📑 🕴
tex-N	3760	<pre>le_advertising_info *pr = (void *)(evt->data+1); /* evt->data[0]</pre>		Expression	Туре	Value
: 0]	377	number of reports (On BlueNRG-MS is always 1) */	-	CLIENT_BDADDR		enon Multiple erro
sam	378⊝	<pre>/* le_advertising_info parameters:</pre>		✓ ➡ myData	char*	0x200002ea hciR
oc()	379	<pre>pr->evt_type: event type (advertising packets types);</pre>		(x)= "myUata	char	2 1002
MS_P	380	<pre>pr->bdaddr_type: type of the peer address (PUBLIC_ADDR, BANDOM_ADD</pre>	R	and then express		
.c:100	381	pr->bdaddr: address of the peer device found during scanning;				
B sen	382	<pre>pr->length: length of advertising or scan response data;</pre>				
	383	<pre>pr->data_RSSI[]: length advertising or scan response data + RSSI.</pre>				
	384	RSSI is last octect (signed integer)				
	385	*/				
	386⊝	/* Add user code for decoding the le_advertising_info event data	b			
	387	on the specific pr->ext_type (ADV_IND, SCAN_RSP,)*/				
	388 🤇	<pre>char* myData = pr->data_RSSI;</pre>				
	389	<pre>break;)/*_EVT_LE_ADVERTISING_REPORT */</pre>				
	390	}		<		,

To lze však velmi snadno napravit. Stačí na název pole kliknout pravým tlačítkem a z lokálního menu vybrat položku "Display As Array …".



Zobrazí se (malý) dialog pro zadání délky pole. Po zadání čísla a potvrzení dialogu lze již pole rozbalit a zobrazit všechny jeho hodnoty.

Zobrazení struktury pomocí odkazu

workspace_1.2.0 - SampleApp/#	Application/User/sample_service.c - STM32CubeIDE		– 🗆 ×
Eile Edit Source Refactor Na	wigate Search Project Bun Window Help		
📑 🕶 🔛 🔞 🛍 🕹 🕶 🔌	5		Q 🤫 🖻 🐺 🔤
🎄 Deb 🛛 陷 Proj 📟 🗖	👔 main.c 👔 app_bluenrg_m 👔 sample_service.c 💠 👔 bluenrg_gap_ac 🛐 startup_stm321 🖻 hci_const.h 👘 🗖	1 🗱 V 🎭 🛛 🎯 E 🖾 🔳	M ### R 6% L 🦄 🖳 🗖
🖻 🐜 it 🔒	375 {	ka 🚽	😑 🔶 × 🖗 📩 🖻 🗄
✓ 🔤 SampleApp [STM32 Cortex-№	376° le advertising info *pr = (void *)(evt->data+1); /* evt->data[0]	Expression	Type Value ^
SampleApp.elf [cores: 0]	377 number of reports (On BlueNRG-MS is always 1) */	CLIENT_BDADDR	Error: Multiple o
✓ P Thread #1 [main] 1 [co	3780 /* le advertising info parameters:	🚽 🥭 myData	char [17] 0x200002ea <hc< td=""></hc<>
user_notify() at sam	379 nr->evt type: event type (advertising nackets types).	(v)+ myData[0]	char 2 '\002'
MY RiveNRG MS R	and provided type: the part address (PIRIT ADDR PANDOM ADDR	(x)= myData[1]	char 1 '\001'
main() at main.c:10	200 provided days of the peer duries (Politic Abbr, Malour, Abbr	(i) myData[2]	char 26 \032'
arm-none-eabi-gdb (8.3.1	provide the part of advertising on some advertising;	(x): myData[3]	char 2 \002
📕 ST-LINK (ST-LINK GDB ser	pr-stength: tength of advertising or scan response data;	ph: myData[5]	char 4 3004
	pr->data_KSSI[]: length advertising or scan response data + KSSI.	00+ myData[6]	char 10 \n
	384 RSSI is last octect (signed integer).	(x): myData[7]	char 255 'ÿ'
	385 */	(v): myData[8]	char 76 'L'
	386° /* Add user code for decoding the le_advertising_info event data b	(x): myData[9]	char 0 '\0'
	387 on the specific pr->evt_type (ADV_IND, SCAN_RSP,)*/	(v): myData[10]	char 16 '\020'
	388 char* myData = pr->data RSSI;	(x): myData[11]	char 5 \005'
	9389 break: /* EVT LE ADVERTISING REPORT */	myData[12]	char 3 (003
	390	< King and a second sec	24 (030 >
	391 1		^
	asyst evt_blue_ac1 *blue_evt = (Void*)event_pckt->data;		
	switch(blue_evt->ecode)}	4	×
	📮 Console 💠 Problems 🕡 Executables 🙀 Debugger Console 🚺 Memory	📕 🗶 🔆 🖌 😹 🖉 💆	
	SampleApp [STM32 Cortex-M C/C++ Application]		
	Download verified successfully		^
			~
< >	<		>

...

4 OnLine zobrazení proměnných při běhu

Pokud je potřeba se podívat na hodnoty nějakých proměnných tak v předešlé kapitole bylo nutno program zastavit (například pomocí breakpointu) a teprve poté bylo možno prohlížet hodnoty jednotlivých proměnných. Nová verze ST-Link umožňuje v omezené míře zobrazovat průběžně hodnoty některých proměnných v prostředí STM32CubeIDE i bez zastavení programu, tedy i přímo při jeho běhu. Postup je následující:

Zkontrolovat, zda je povoleno / zatrženo Enable Live Expressions. Uvedeno v: Run (horní menu) – Debug Configurations (položka v rozbalovacím menu) – Debugger (záložka na nově otevřeném dialogu)

	Name: STM32L152_Nucelo_Base	
	🖺 Main 🕸 Debugger 🕨 Startup 🦆 Sour	rce 🔲 <u>C</u> ommon
+ Application + Attach to Application	Device settings	
+ Postmortem Debugger	Debug in low power modes:	Enable ~
+ Remote Application Hardware Debugging	Suspend watchdog counters while halted:	No configuration \checkmark
h Group	Serial Wire Viewer (SWV)	RTOS Kernel Awareness
2 Cortex-M C/C++ Applicatic	Enable	Enable RTOS Proxy
JCLEO_L476_TestA	Core Clock (MHz): 16.0	Driver settings
VI32F030C8T_BootLoader	Limit SWO clock	Driver: ThreadX 🗸
M32F030C8T_Valve Debug	Maximum SWO clock (kHz): auto detect	Port: cortex m0 ~
M32L152_Nucelo_Base	Port number: 61235	
M32L432KL_Base		Port number: 60000
M32L4/6RG_NUCLEO		
	Verify flack download	
	Enable live expressions	
	Log to file: C:\Projects-ARI	M\STM32L1xx\STM32L152_Nuce Browse
	External Loader	Scan Initialize
	Max halt timeout(s): 2	

- Sledovanou proměnnou vytvořit jako globální, tedy mimo funkci (v podstatě existující po celou dobu běhu programu).
- Spustit program pomocí ladění a zastavit jej na jeho začátku.
- Na pravé straně (nahoře / dole podle umístění) najít záložku "Live Expressions".

				~
				>
	🎄 Debug 🛷 Search 🍃 Type Hielarchy 🏘 Live Expressions 🗙	× %	8	- 0
^	Expression Type Value			

- Pokud tato záložka není (nelze ji najít) tak: kliknout na hledací lupu v horním pravém rohu začít psát Live Expression v zobrazeném seznamu najít tuto možnost a vybrat (například kliknout). Zobrazí se požadovaná záložka / dialog.
- Do připraveného řádku vepsat název (dříve vtvořenou globální) proměnnou.
- Spustit běh programu.
- Zhruba jednou za vteřinu (mnohdy i častěji) je průběžně čtena hodnota uvedené proměnné a zobrazována v této záložce.



Poznámky:

 Pozor na skutečnost, že Live Expression je vhodné zejména pro pomaleji se měnící proměnné, například pro globální stav programu.

5 Zobrazení FLASH a SRAM

Někdy je potřeba zobrazit přímo adresy a jejich hodnoty ve FLASH nebo SRAM. Nejprve je potřeba vyvolat okno pro zobrazení obsahu paměti pomocí:

- Window (položka v horní menu)
- Show View (položka v rozbalovacím menu)
- Memory Browser (položka v rozbalovacím pod-menu)

V dolní části STM32CubeIDE se zobrazí okno (se záložkou) zobrazující obsah paměti. Do horního řádku v tomto okně lze zapsat adresu a stisknout vlevo od něho tlačítko **GO**. Bude zobrazen obsah části požadované paměti. Pamětí lze listovat nahoru / dolu zcela bez omezení. Pozor však na následující skutečnosti:

- Tímto se zobrazuje část paměti z celkové paměti, tedy z celého adresního prostoru 4GB. Pozor tedy na tyto skutečnosti:
 - Programová FLASH paměť (nejčastěji) začíná od adrese 0x08000000. Pokud je tedy požadavek na zobrazení například adresy 0x1000 ve FLASH paměti je potřeba zadat skutečnou / celkovou adresu:

0x08000000 + 0x1000 = 0x08001000 (na toto pozor)

 Datová SRAM paměť (nejčastěji) začíná od adrese 0x02000000. Pokud je tedy požadavek na zobrazení například adresy 0x1000 ve SRAM paměti je potřeba zadat skutečnou / celkovou adresu:

0x02000000 + 0x1000 = 0x02001000 (na toto pozor)

 Pomocí tlačítka New Tab lze zobrazit / přidat další záložku s částí zobrazené paměti. Takto lze vytvořit několik záložek, v každé zobrazit požadovanou část paměti a ty pouze přepínat (místo rolování obsahu nahoru / dolu).

Poznámky:

 Toto zobrazení je (velmi) vhodné například při kontrola ukládání hodnot do FLASH využívané jako emulace paměti typu EEPROM. Data jsou ukládána do stránky FLASH paměti od známé adresy a lze je tedy takto snadno kontrolovat.