

TOOL – Lauterbach μTrace[®] for Cortex[®]-M with XMC4000 and XMC1000 family

XMC[™] microcontrollers
July 2016



Agenda

1 Lauterbach μ Trace®: Overview and features

2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

Agenda

1 Lauterbach μ Trace®: Overview and features

2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

Lauterbach debugger μ Trace[®] for Cortex[®]-M

› Overview

- μ Trace[®] is a new all-in-one debug and trace solution from Lauterbach to support Cortex[®]-M family
- μ Trace[®] supports Infineon XMC4000 and XMC1000 microcontroller families

› Debug features

- C/C++ debugging
- Simple and complex breakpoints
- Read and write memory during program runs
- Flash programming
- OS-aware debugging
- AMP multicore debugging of two or more Cortex[®]-M cores

› Trace features

- 4-bit ETMv3 in continuous mode
- ITM over TPIU and Serial Wire Output
- Multicore tracing
- Streaming trace information to the host computer for long-term tracing, streaming rate up to 100 Mbyte/s by using USB3 interface
- Analysis of task and function runtimes
- Code coverage analysis
- Trace evaluation even during recording
- Energy measurement using TRACE32 Analog Probe

Agenda

1 Lauterbach μ Trace®: Overview and features

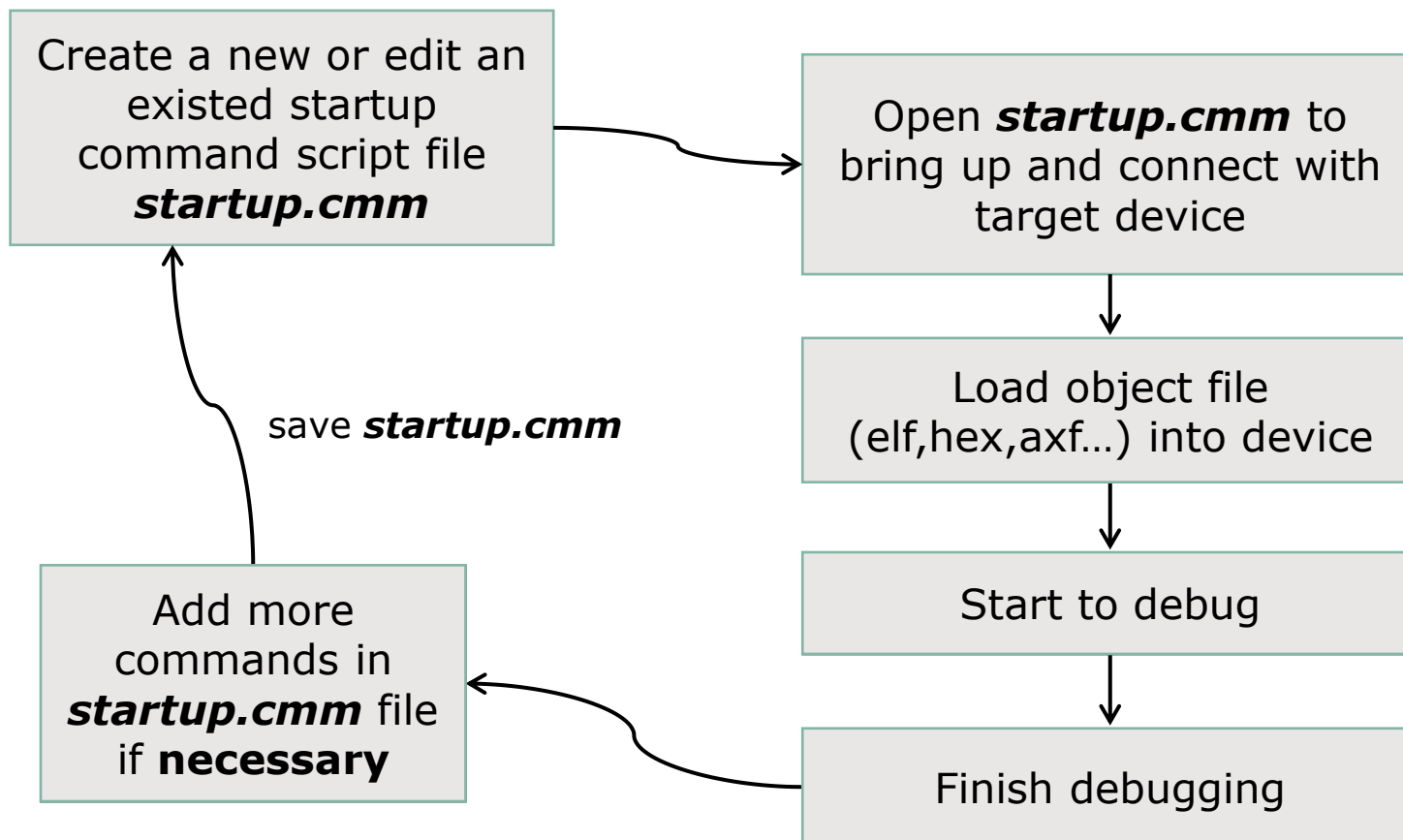
2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

Usage of Lauterbach debugger μ Trace[®]



Agenda

1 Lauterbach μ Trace®: Overview and features

2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

About this tutorial

> **Purposes**

- Quick start of Lauterbach μ Trace[®] with XMC4000 and XMC1000 family microcontrollers
- Quick start of creating command script batch file (.cmm) based on graphic configuration for XMC4000 and XMC1000
- Provides step-by-step how to use Lauterbach μ Trace[®] debugger with example

> **Prerequisites to follow the tutorial**

- Debugger hardware: μ Trace[®]
- Install the up-to-date μ Trace[®] software:
[Download Lauterbach \$\mu\$ Trace](#)



OS	Description	File	Date	Size
Windows	Update for Cortex-A Cortex-M Cortex-R	 _icdcortex_win.zip	2-DEC-2015	94.4 MB

- Install DAVE[™] version 4.x.x: www.infineon.com/DAVE
- XMC4000 application kit or XMC1000 boot kit:



Agenda

1 Lauterbach μ Trace®: Overview and features

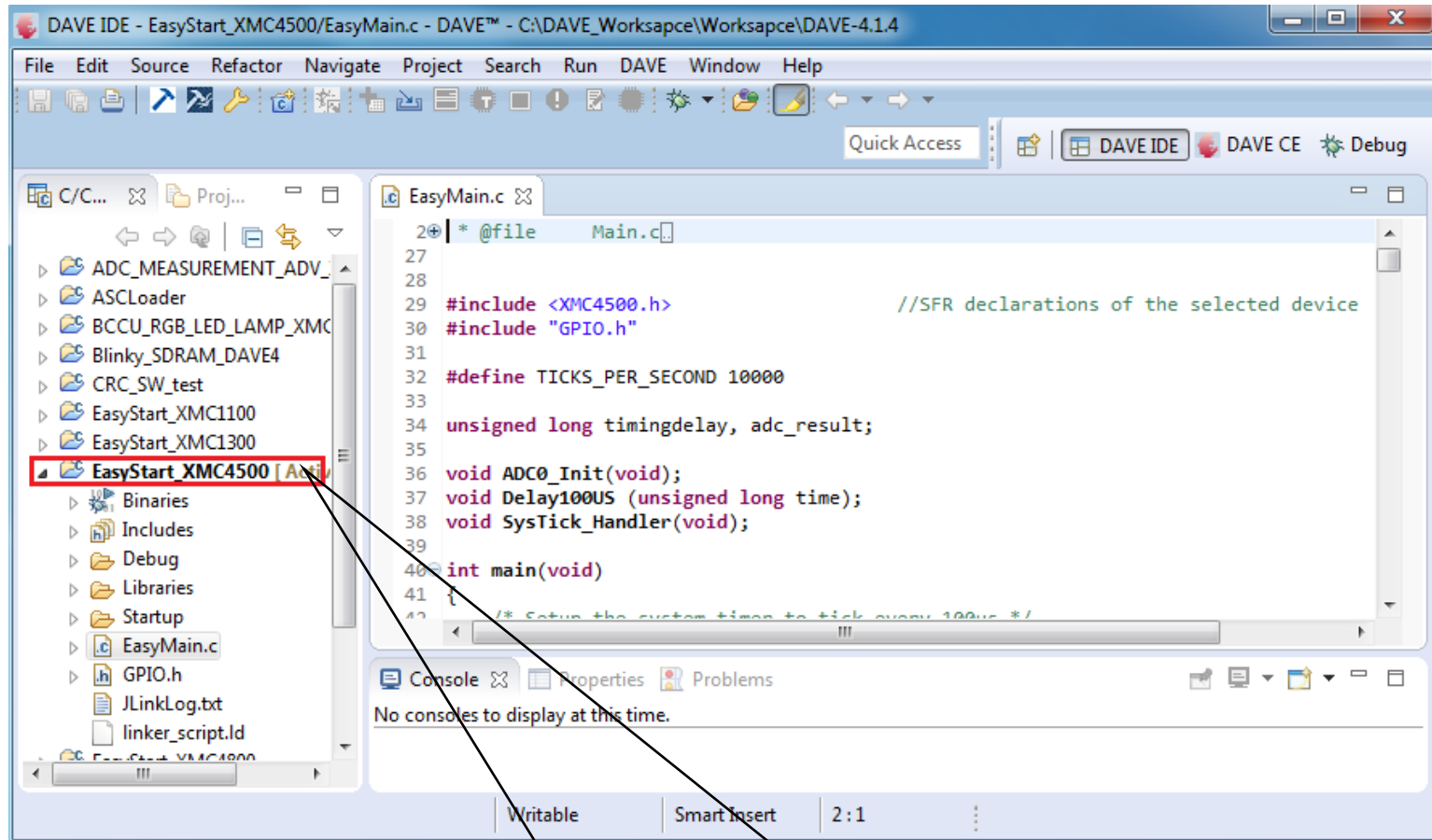
2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

Step 1: Create a XMC4000 example project in DAVE™



DAVE™ example project: **EasyStart_XMC4500**

Step 2: Create a new startup.cmm file

Step 2.1: Use a text editor, e.g. Notepad++, to create a text file with following commands and save the file into DAVE™ project folder as "startup.cmm"

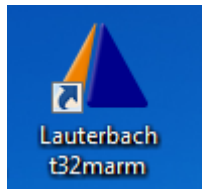
```
DO ~/demo/arm/flash/xmc4500 CPU=XMC4500-X1024 PREPAREONLY //flash
declaration

DIALOG.YESNO "Program flash memory?" //open ELF object file
LOCAL &progload
ENTRY &progload
IF &progload
(
    FLASH.ReProgram ALL //enable flash programming
    Data.LOAD.auto *
    FLASH.ReProgram OFF
)
SYStem.Up
List //open the source window
Go main //run program to main function
```

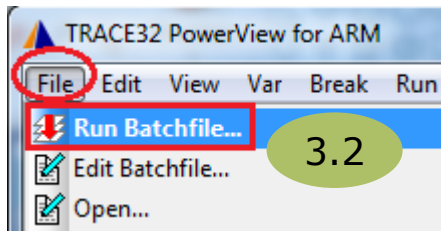
Note:

1. CPU: device name used, can be changed based on application
2. xmc4500.cmm: script for flash declaration of XMC4000 family in μ Trace® install folder

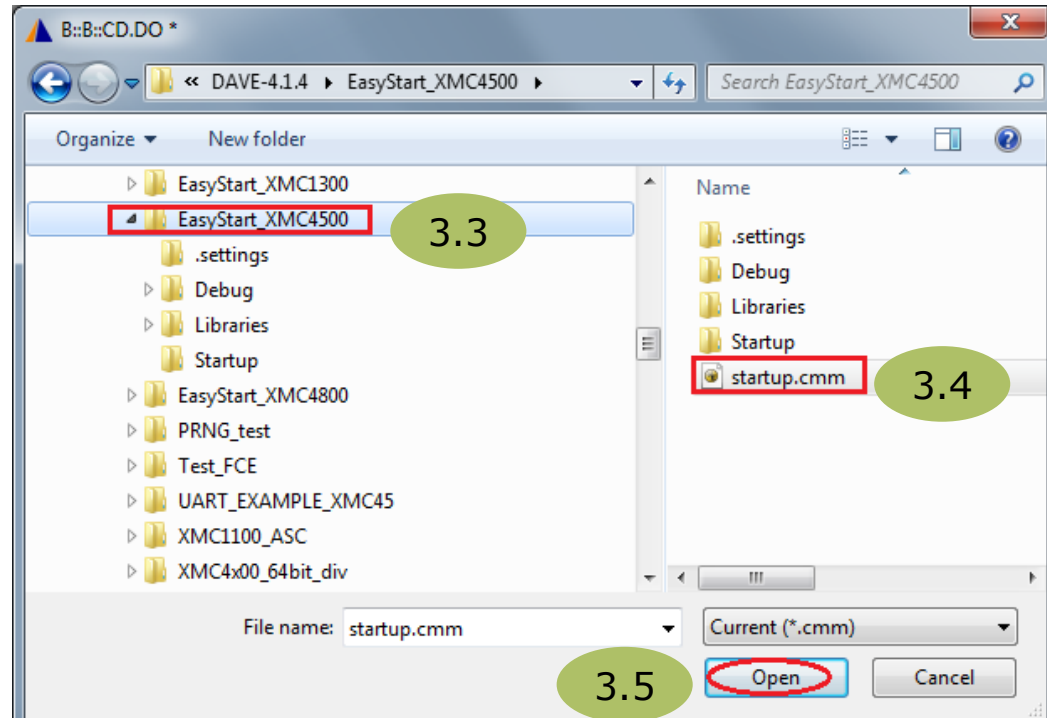
Step 3: Open startup.cmm file in μ Trace[®]



3.1



3.2



3.3

3.4

3.5

Step 3.1: open Lauterbach μ Trace[®] from desktop

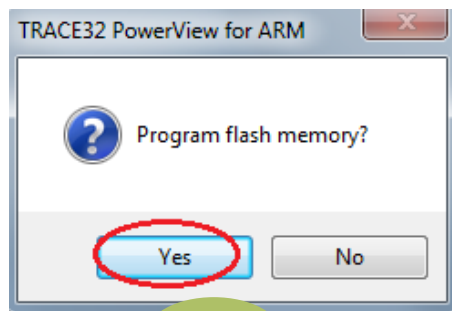
Step 3.2: open "File->Run Batchfile..."

Step 3.3: go to DAVE[™] project folder

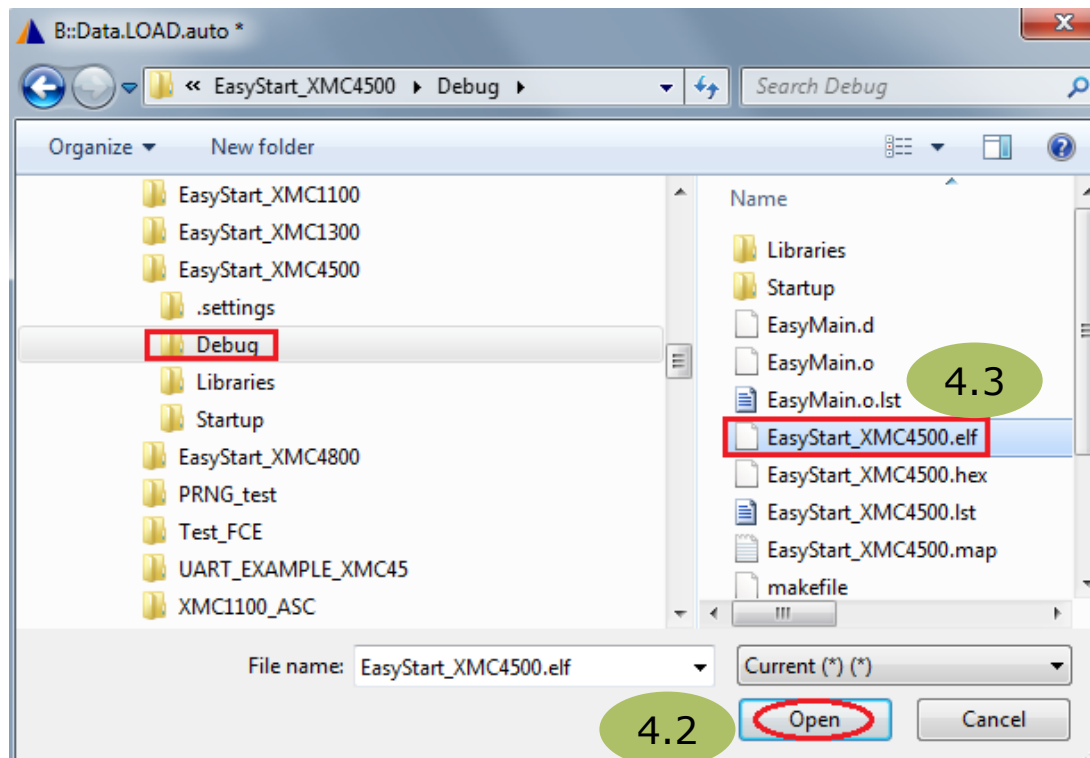
Step 3.4: select startup.cmm file

Step 3.5: click "Open"

Step 4: Load the program into flash (1)



4.1



4.3

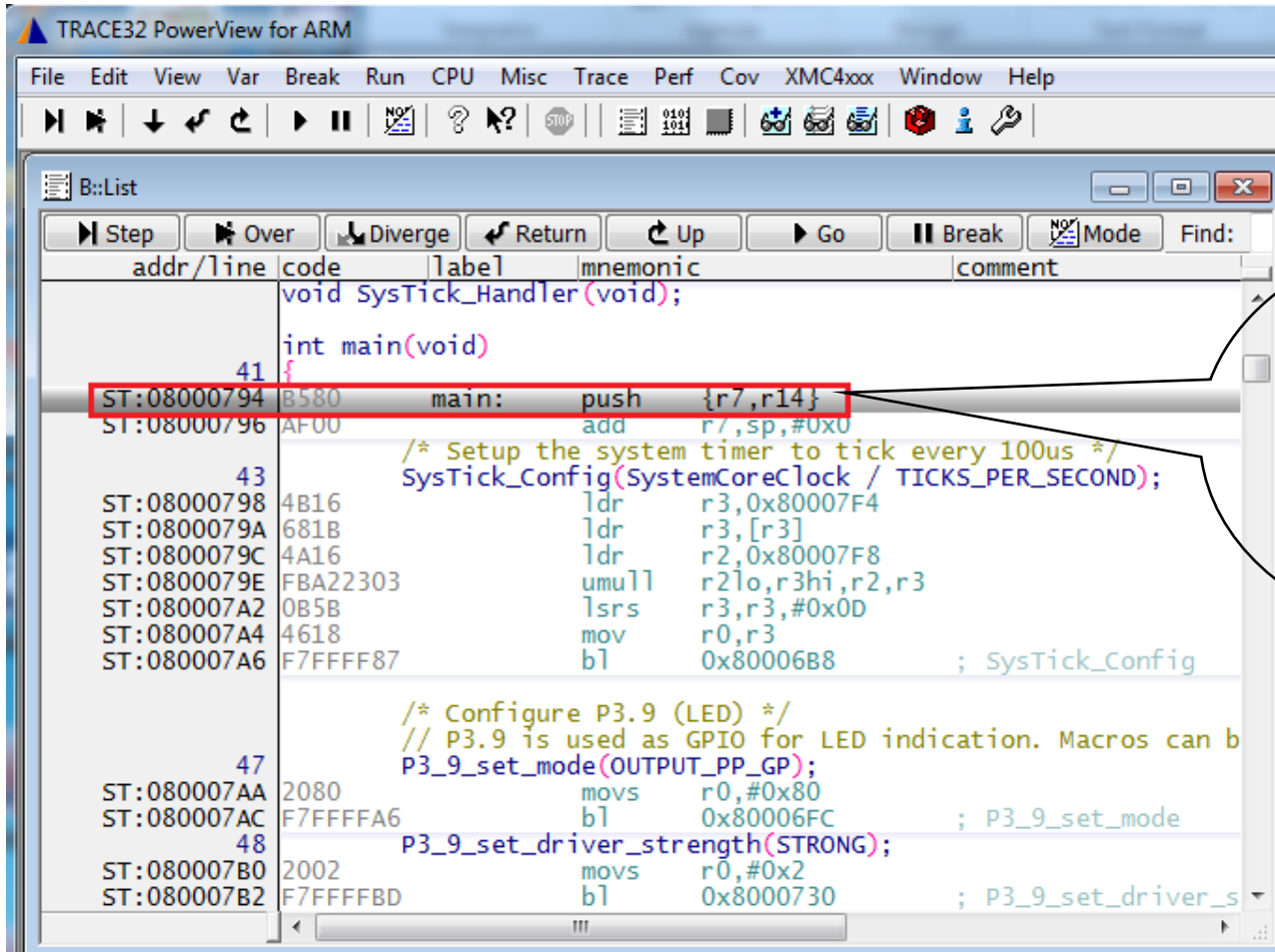
4.2

Step 4.1: click "Yes"

Step 4.2: go to DAVE™ project folder, and select elf object file

Step 4.3: click "Open"

Step 4: Load the program into flash (2)



The screenshot shows the TRACE32 PowerView for ARM interface. The 'B::List' window is open, displaying the source code of the program. The 'main' function is highlighted in red, and the 'push {r7,r14}' instruction is also highlighted. A callout bubble points to this instruction with the text: 'After loading the program the source window is displayed and the code is pointed to main function'.

addr/line	code	label	mnemonic	comment
			void SysTick_Handler(void);	
			int main(void)	
41			{	
ST:08000794	B580	main:	push {r7,r14}	
ST:08000796	AF00		add r7,sp,#0x0	
			/* Setup the system timer to tick every 100us */	
43			SysTick_Config(SystemCoreClock / TICKS_PER_SECOND);	
ST:08000798	4B16		ldr r3,0x80007F4	
ST:0800079A	681B		ldr r3,[r3]	
ST:0800079C	4A16		ldr r2,0x80007F8	
ST:0800079E	FBA22303		umull r2lo,r3hi,r2,r3	
ST:080007A2	0B5B		lsls r3,r3,#0x0D	
ST:080007A4	4618		mov r0,r3	
ST:080007A6	F7FFFF87		bl 0x80006B8	; SysTick_Config
			/* Configure P3.9 (LED) */	
			// P3.9 is used as GPIO for LED indication. Macros can b	
47			P3_9_set_mode(OUTPUT_PP_GP);	
ST:080007AA	2080		movs r0,#0x80	
ST:080007AC	F7FFFFA6		bl 0x80006FC	; P3_9_set_mode
48			P3_9_set_driver_strength(STRONG);	
ST:080007B0	2002		movs r0,#0x2	
ST:080007B2	F7FFFFBD		bl 0x8000730	; P3_9_set_driver_s

After loading the program the source window is displayed and the code is pointed to main function

Step 5: Start debugging

The screenshot shows the TRACE32 PowerView for ARM interface. The main window displays the assembly code for the `main` function. The `CPU` menu is highlighted, and the `XMC4xxx` menu is also highlighted. The `Target reset` option is selected in the `CPU` menu. The `XMC4000 peripheral` is listed in the `XMC4xxx` menu.

Annotations and steps shown:

- Step into
- Step over
- Leave loop
- Go to last instruction
- Return the caller function
- Start execution
- Stop execution
- Target reset
- XMC4000 peripheral

The assembly code shown in the main window is:

```

void SysTick_Handler(void);

int main(void)
{
    ST:08000794 B580 main: push {r7,r14}
    ST:08000796 AF00 r7,sp,#0x0

```

Agenda

1 Lauterbach μ Trace®: Overview and features

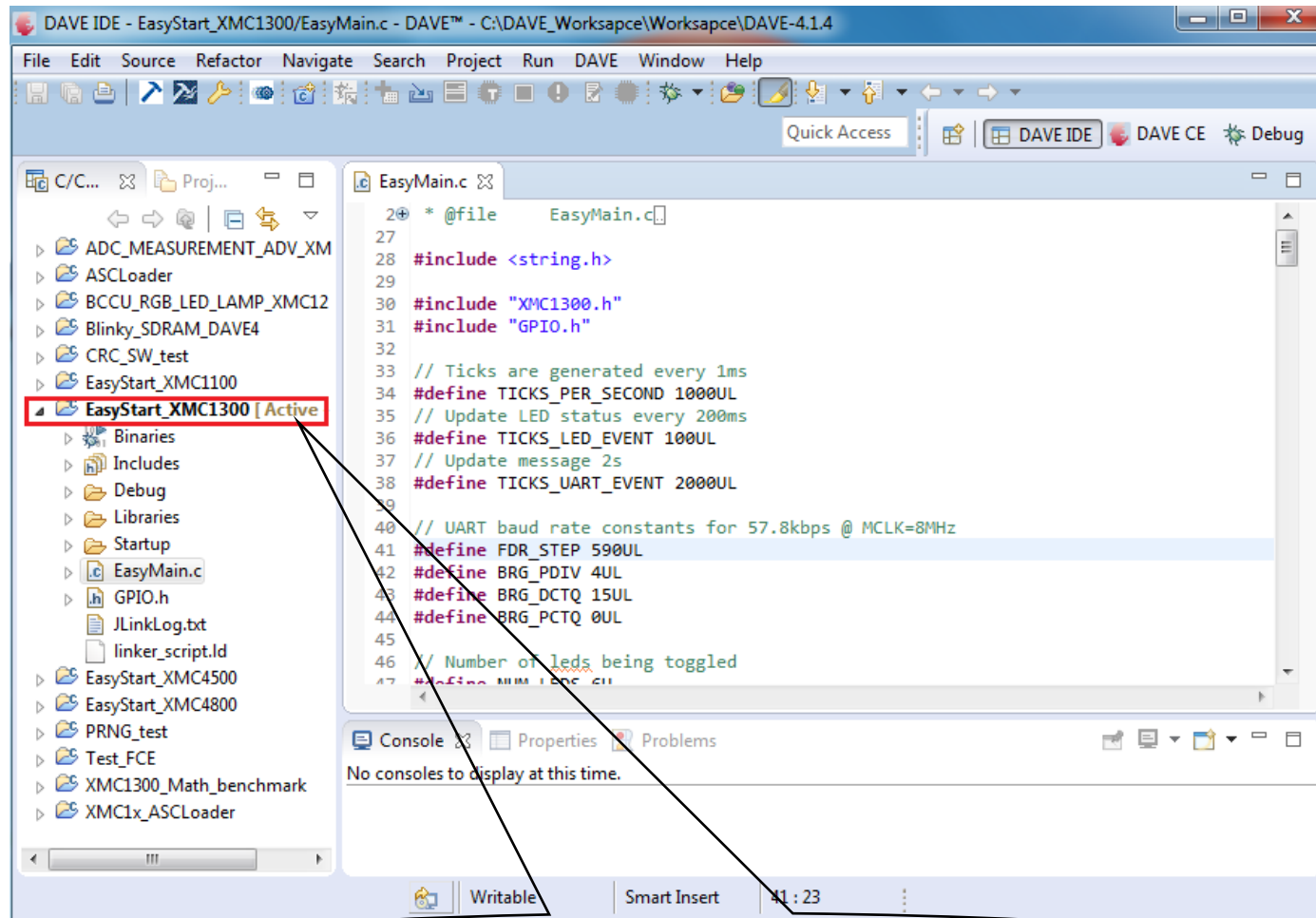
2 Usage of Lauterbach μ Trace®

3 About this tutorial

4 Quick start of Lauterbach μ Trace® with XMC4000 application kit

5 Quick start of Lauterbach μ Trace® with XMC1000 application kit

Step 1: Create a XMC1000 example project in DAVE™



DAVE™ example project: **EasyStart_XMC1300**

Step 2: Create a new startup.cmm file

Step 2.1: Use a text editor, e.g. Notepad++, to create a text file with following commands and save the file into DAVE™ project folder as "startup.cmm"

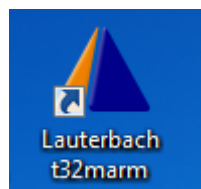
```
DO ~/demo/arm/flash/xmc1300 CPU=XMC1302-X0064 PREPAREONLY
//flash declaration

DIALOG.YESNO "Program flash memory?" //open ELF object file
LOCAL &progload
ENTRY &progload
IF &progload
(
  FLASH.ReProgram ALL //enable flash programming
  Data.LOAD.auto *
  FLASH.ReProgram OFF
)
SYStem.Up
List //open the source window
Go main //run program to main function
```

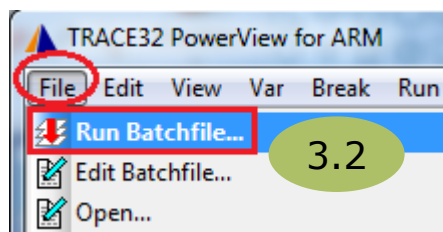
Note:

1. CPU: device name used, can be changed based on application
2. xmc1300.cmm: script for flash declaration of XMC1000 family in μ Trace® install folder

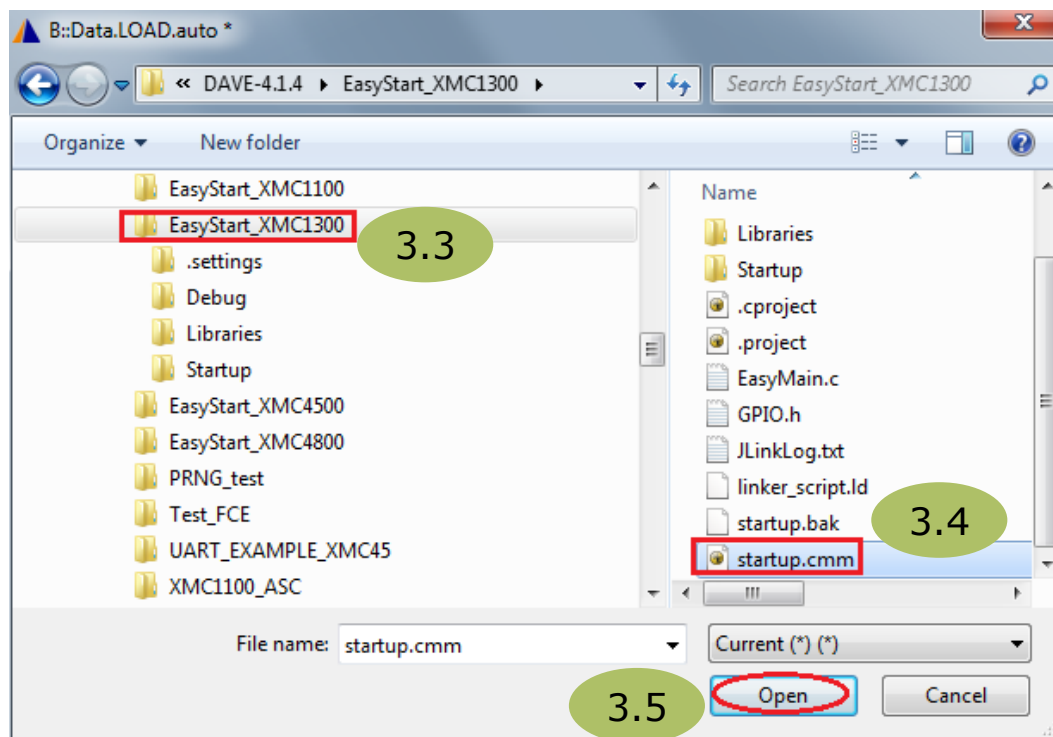
Step 3: Open startup.cmm file in μ Trace[®]



3.1



3.2



Step 3.1: open Lauterbach μ Trace[®] from desktop

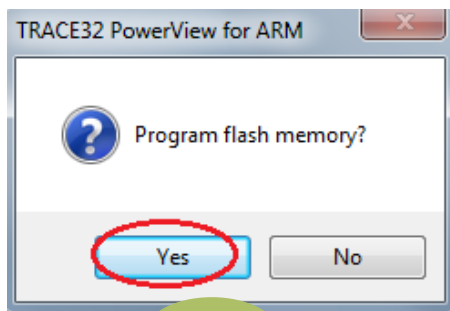
Step 3.2: open "File->Run Batchfile..."

Step 3.3: go to DAVE[™] project folder

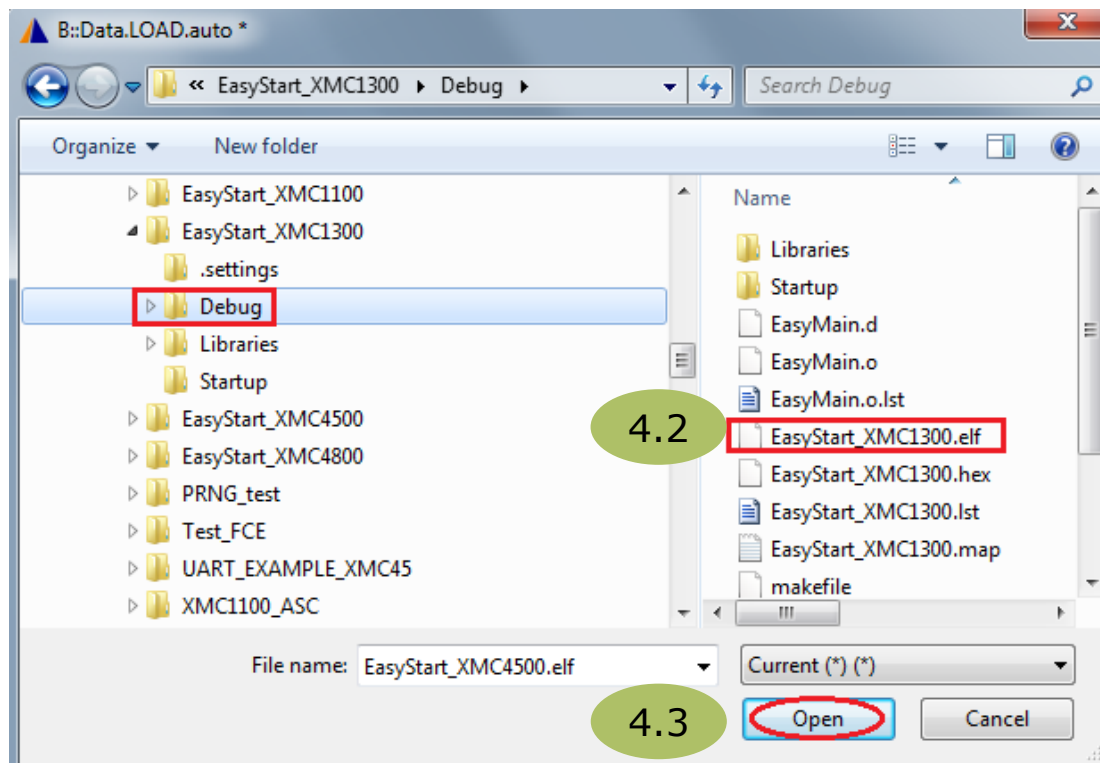
Step 3.4: select startup.cmm file

Step 3.5: click "Open"

Step 4: Load the program into flash (1)



4.1



4.2

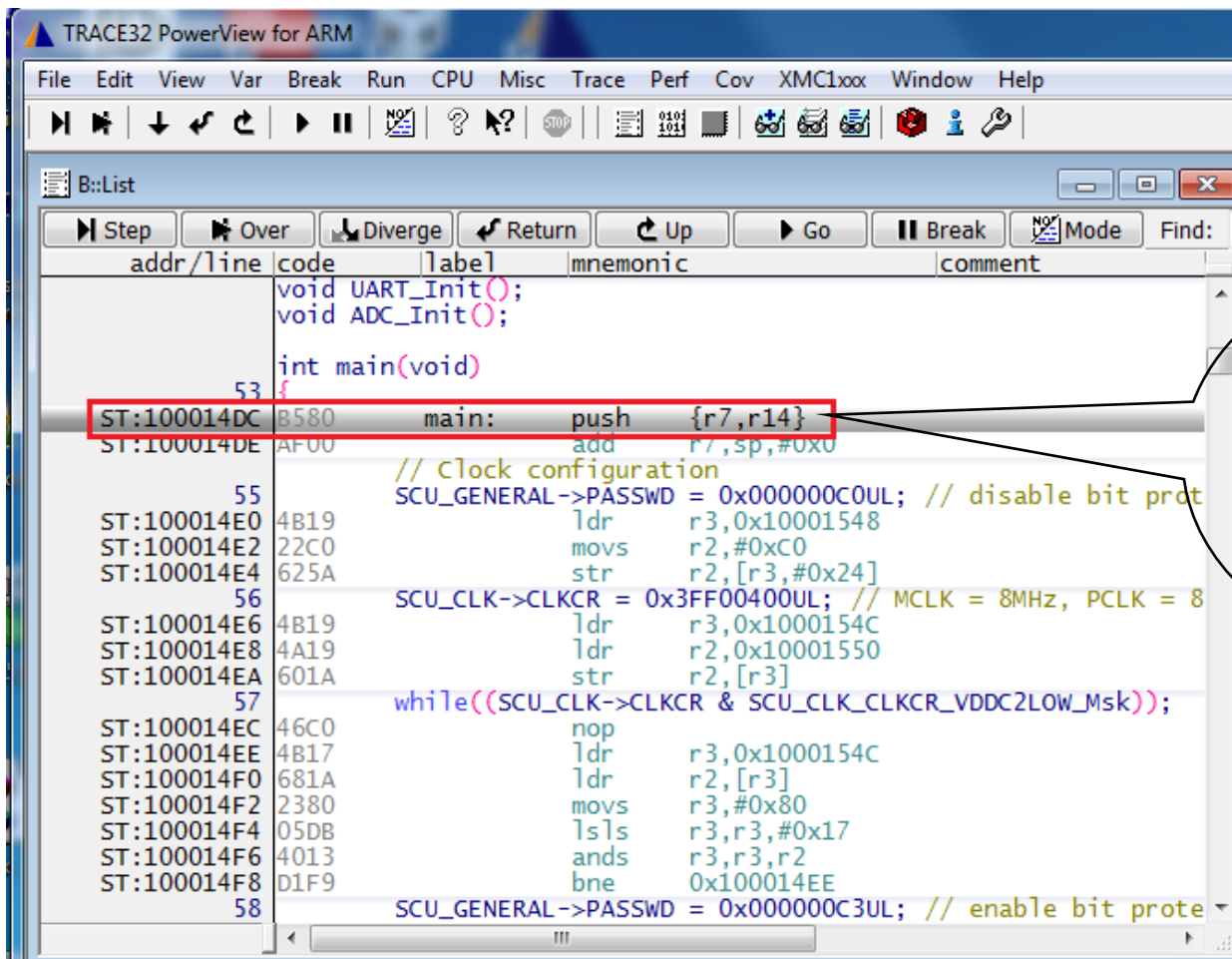
4.3

Step 4.1: click "Yes"

Step 4.2: go to DAVE™ project folder and select elf object file

Step 4.3: click "Open"

Step 4: Load the program into flash (2)



TRACE32 PowerView for ARM

File Edit View Var Break Run CPU Misc Trace Perf Cov XMC1xxx Window Help

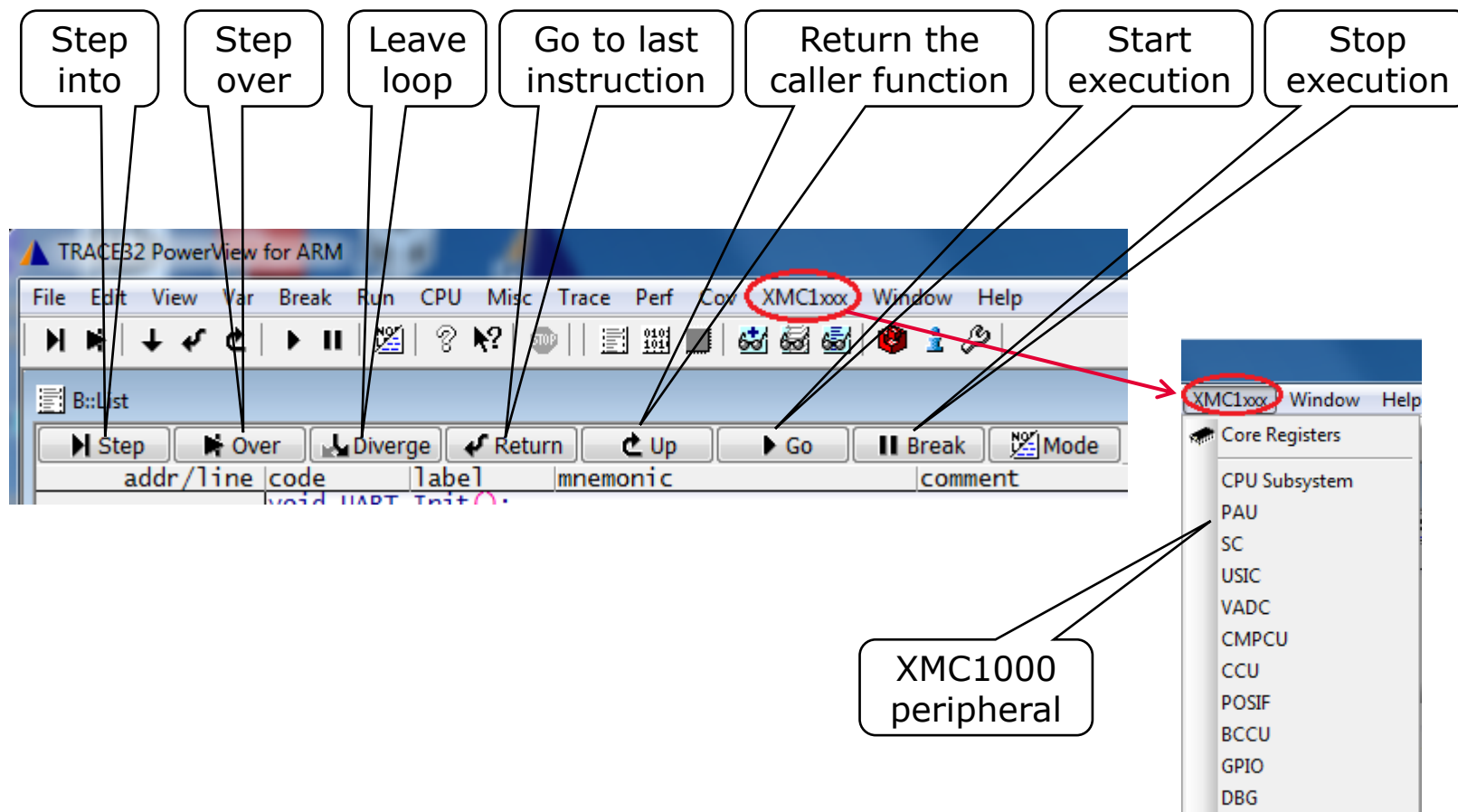
B::List

Step Over Diverge Return Up Go Break Mode Find:

addr/line	code	label	mnemonic	comment
	void UART_Init();			
	void ADC_Init();			
	int main(void)			
53	{			
ST:100014DC	B580	main:	push {r7,r14}	
ST:100014DE	AF00		add r7,sp,#0x0	
			// Clock configuration	
55			SCU_GENERAL->PASSWD = 0x000000C0UL; // disable bit prote	
ST:100014E0	4B19		ldr r3,0x10001548	
ST:100014E2	22C0		movs r2,#0xC0	
ST:100014E4	625A		str r2,[r3,#0x24]	
56			SCU_CLK->CLKCR = 0x3FF00400UL; // MCLK = 8MHz, PCLK = 8	
ST:100014E6	4B19		ldr r3,0x1000154C	
ST:100014E8	4A19		ldr r2,0x10001550	
ST:100014EA	601A		str r2,[r3]	
57			while((SCU_CLK->CLKCR & SCU_CLK_CLKCR_VDDC2LOW_Msk));	
ST:100014EC	46C0		nop	
ST:100014EE	4B17		ldr r3,0x1000154C	
ST:100014F0	681A		ldr r2,[r3]	
ST:100014F2	2380		movs r3,#0x80	
ST:100014F4	05DB		lsls r3,r3,#0x17	
ST:100014F6	4013		ands r3,r3,r2	
ST:100014F8	D1F9		bne 0x100014EE	
58			SCU_GENERAL->PASSWD = 0x000000C3UL; // enable bit prote	

After loading the program the source window is displayed and the code is pointed to main function

Step 5: Start debugging



Glossary abbreviations

- › DAVE™ Free development IDE for XMC™
- › JTAG Joint Test Action Group
- › SWD Serial Wire Debug

Support material

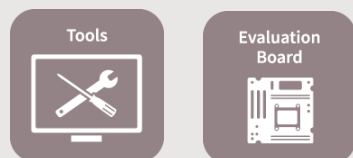
Collaterals and Brochures



- › Product Briefs
- › Selection Guides
- › Application Brochures
- › Presentations
- › Press Releases, Ads

› www.infineon.com/XMC

Technical Material



- › Application Notes
- › Technical Articles
- › Simulation Models
- › Datasheets, MCDS Files
- › PCB Design Data

› www.infineon.com/XMC

› [Kits and Boards](#)

› [DAVE™](#)

› [Software and Tool Ecosystem](#)

Videos



- › Technical Videos
- › Product Information Videos

› [Infineon Media Center](#)

› [XMC™ Mediathek](#)

Contact



- › Forums
- › Product Support

› [Infineon Forums](#)

› [Technical Assistance Center \(TAC\)](#)

Disclaimer

The information given in this training materials is given as a hint for the implementation of the Infineon Technologies component only and shall not be regarded as any description or warranty of a certain functionality, condition or quality of the Infineon Technologies component.

Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this training material.



Part of your life. Part of tomorrow.

