LAUTERBACH TRACE32 GDB FRONTEND DEBUGGER
FOR PORTENTA H7 OR NICLA VISION

This tutorial explains how to get a free, fully functional version of the Lauterbach TRACE32 GDB Front End debugger for ARM and debug your Portenta H7 or Nicla Vision application via GDB on a serial interface.

Arduino boards Portenta H7 and Nicla Vision are based on the same STM32H7 chip. In this tutorial, the term **board** means both Portenta H7 and Nicla Vision indifferently.

In this tutorial you will load on the **board** an application which includes the MRI (Monitor for Remote Inspection). This is a GDB compatible serial monitor which is included with the ThreadDebug sketch in the IDE Examples for this **board** (M7 core) and with all examples in the TRACE32 /demo directory.

This tutorial assumes that you have already installed the Arduino IDE 1.x or Arduino IDE 2.x and configured it to support your **board**. Please refer to Arduino tutorials before you proceed, eg:
https://docs.arduino.cc/tutorials/portenta-h7/setting-up-portenta

What You Will Learn

- How to get a free license key for TRACE32 GDB Front End debugger for your **board** - M7 core
- How to download and start the Lauterbach TRACE32 GDB Front End debugger
- How to flash and debug some ready-to-run demos

Required Hardware and Software

- Portenta H7 or Nicla Vision board
- USB C cable (either USB A to USB C or USB C to USB C)
- Arduino IDE 1.8.19+ or Arduino IDE 2.1.1+

TRACE32 GDB Front End debugger

Throughout this document the double-tilde (~~) is used as a place holder for the directory where you unzipped the TRACE32 software.

1) Download the TRACE32 debugger

The zip file with TRACE32 for Arduino boards is available here:

https://www.lauterbach.com/download_demo.html

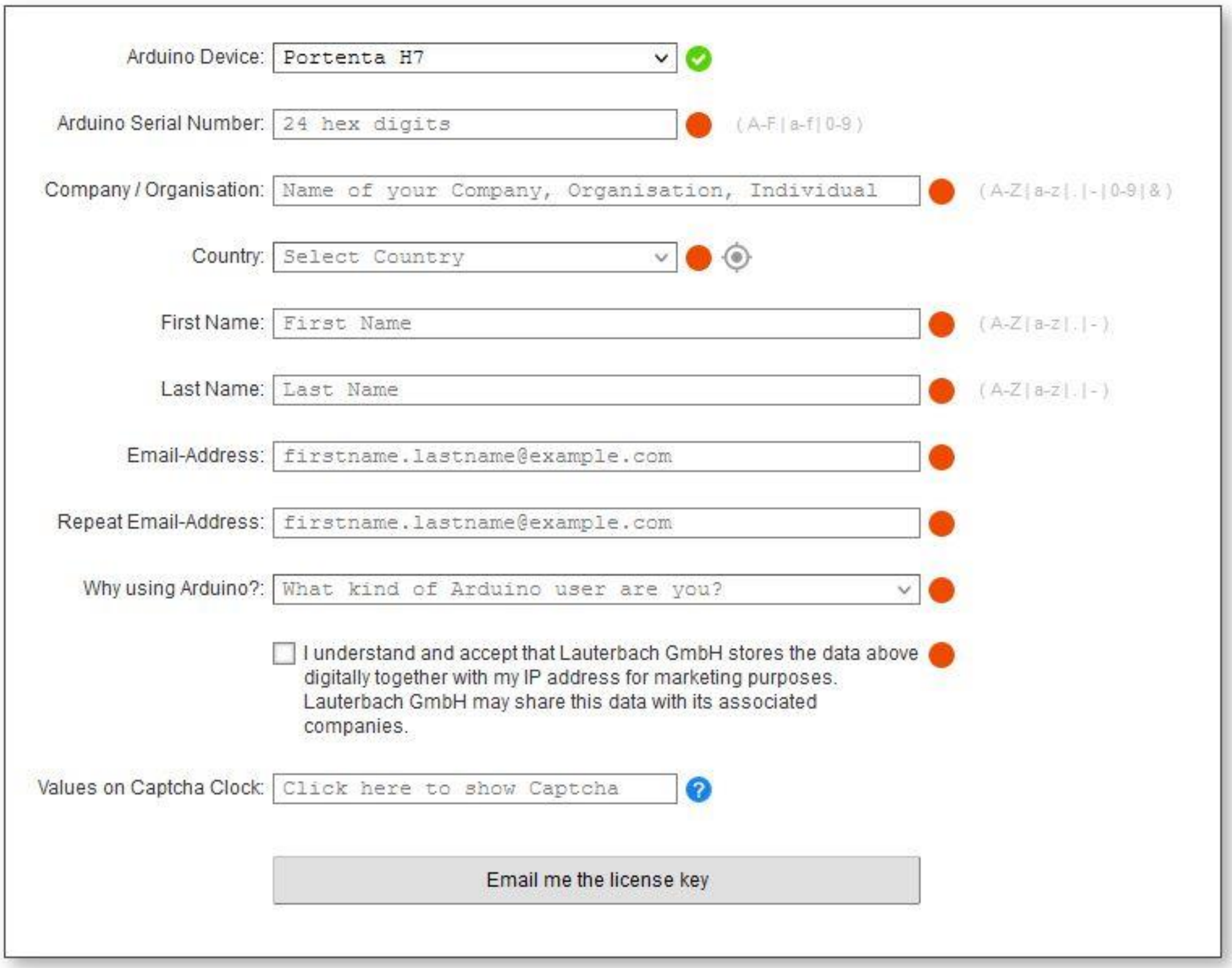
Select the file called "Debugger for GDB target (Arduino Pro)".

You need to extract the zip file to a directory of your choice. On Windows systems, please avoid C:\T32, because this is the default installation directory for the full TRACE32 distribution.

2) Registration and license key

Without a valid license, the TRACE32 debugger only works for few minutes in demo mode. Lauterbach can generate a one-year free license based on the serial number of your **board**. In order to get a license, you need to register here:
www.lauterbach.com/4543

Enter the board serial number, your name and email and you will be sent your license key. Please copy & paste the license string in a text file named **license.t32** in the main directory of your TRACE32 installation.

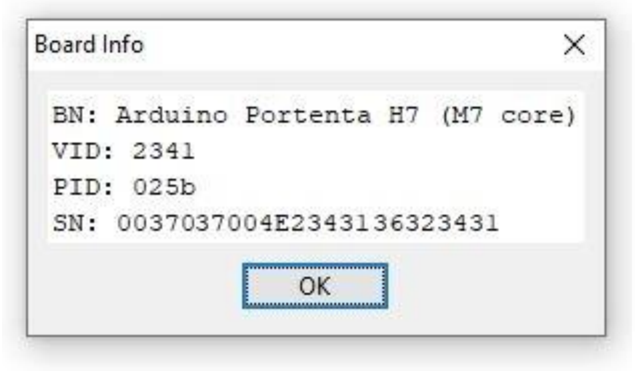


Arduino Device:	Portenta H7	✓
Arduino Serial Number:	24 hex digits	● (A-F | a-f | 0-9)
Company / Organisation:	Name of your Company, Organisation, Individual	● (A-Z | a-z | . | - | 0-9 | &)
Country:	Select Country	●
First Name:	First Name	● (A-Z | a-z | . | -)
Last Name:	Last Name	● (A-Z | a-z | . | -)
Email-Address:	firstname.lastname@example.com	●
Repeat Email-Address:	firstname.lastname@example.com	●
Why using Arduino?:	What kind of Arduino user are you?	●
	☐ I understand and accept that Lauterbach GmbH stores the data above digitally together with my IP address for marketing purposes. Lauterbach GmbH may share this data with its associated companies.	●
Values on Captcha Clock:	Click here to show Captcha	?

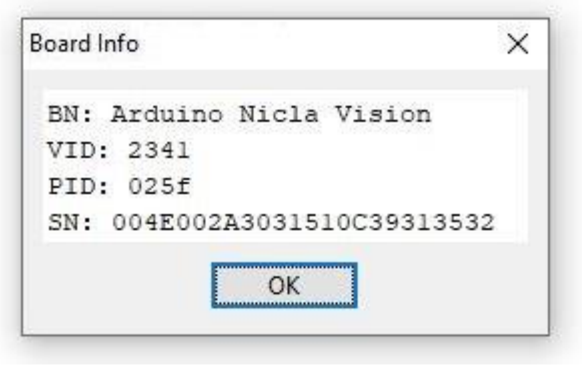
Request a Debug License for Arduino Pro

There are 2 alternative ways to detect the board serial number:

- *use the Arduino IDE.* Select the right COM port, then the menu item Tools->Get Board Info. This should show a 96 bit serial number.



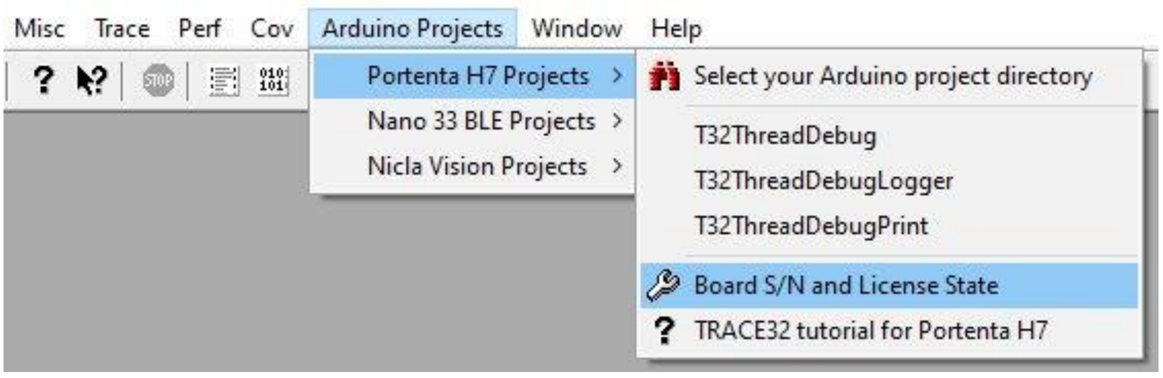
```
Board Info
BN: Arduino Portenta H7 (M7 core)
VID: 2341
PID: 025b
SN: 0037037004E2343136323431
```

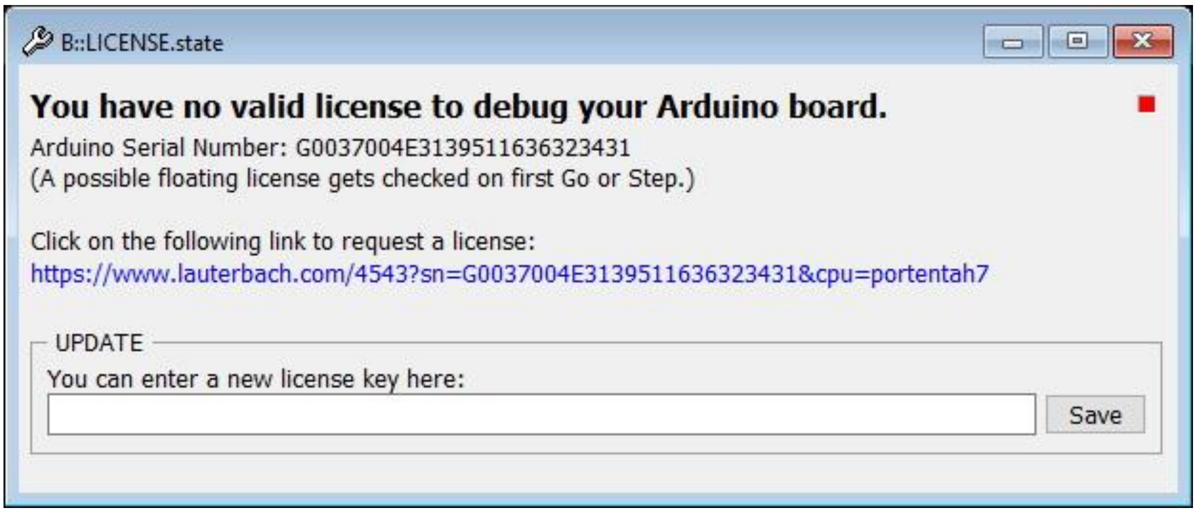


```
Board Info
BN: Arduino Nicla Vision
VID: 2341
PID: 025f
SN: 004E002A3031510C39313532
```

INFORMATION: if you see a 64-bit serial number, then you need to update your Arduino IDE and/or the Arduino Core software in your board. Details of how to do this can be found on the Arduino website.

- *use TRACE32 debugger.* See below how to start it and how to load the T32ThreadDebug demo program. Then click the menu item "Board S/N and License State". Your 96-bit board serial number will be printed in the AREA window and a dialog LICENSE.state will be opened.
-





B::LICENSE.state

You have no valid license to debug your Arduino board.

Arduino Serial Number: G0037004E3139511636323431
(A possible floating license gets checked on first Go or Step.)

Click on the following link to request a license:
https://www.lauterbach.com/4543?sn=G0037004E3139511636323431&cpu=portentah7

UPDATE
You can enter a new license key here:

You can copy & paste to the Lauterbach registration page the serial number printed in the AREA window, or you can click on the link suggested in LICENSE.state.

Note: newer TRACE32 software version should automatically detect and show the board serial number, at first attach to the target.

3) Start the TRACE32 debugger

In order to use the debugger, you need to start the right executable for your host operating system. These are found in the appropriate sub-directory for your operating system under ~~/bin:

~~/bin/windows for 32bit Windows hosts
~~/bin/windows64 for 64-bit Windows hosts
~~/bin/pc_linux64 for 64-bit Linux hosts.

We suggest making a desktop link to the right executable:
- for Windows, this is t32marm.exe
- for Linux, this is t32marm

On Windows systems, the TRACE32 start-up script will automatically search for the right COM port attached to the **board**.

INFORMATION: On Linux systems, you will need to edit the **system-settings.cmm** file to manually add the serial port to connect to the **board**. This is a text file and can be opened with your favorite text editor. Edit the line that defines &GDBPORT to refer to the serial port, for example:

```
&GDBPORT="/dev/ttyACM0"
```

This must be done before you start the TRACE32 software.

Now you can start the TRACE32 debugger.

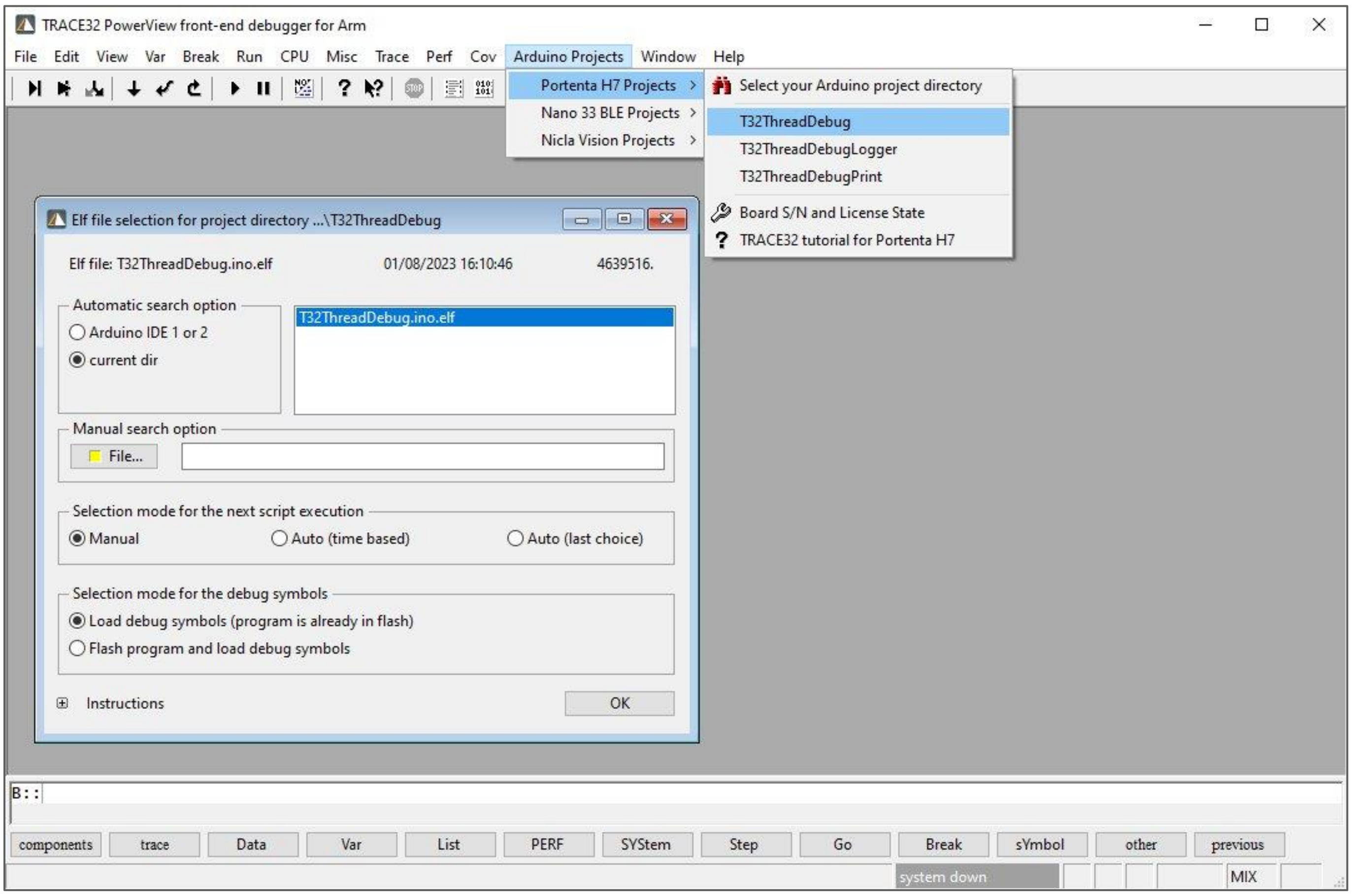
INFORMATION: The manual port setting is also useful for Windows systems where you have multiple **board**s connected, and you want to select a specific board to be used by TRACE32 for debugging. The automatic port selection is disabled when a &GDBPORT definition is found in **system-settings.cmm**.

4) Run your first demo

A number of pre-built demo programs are available and these can be accessed from the “Portenta H7 Projects” or “Nicla Vision Projects” menu. These instructions are for the first one (T32ThreadDebug), although they all follow a similar pattern.

The demo directory already includes the symbolic file for debugging (.elf) and the binary file for flash programming (.bin).

Select “T32ThreadDebug” from the projects menu and you will be presented with a dialog called “Elf File Selection”. This is where the TRACE32 initial environment is set-up.



TRACE32 PowerView front-end debugger for Arm
File Edit View Var Break Run CPU Misc Trace Perf Cov Arduino Projects Window Help
Portenta H7 Projects
Nano 33 BLE Projects
Nicla Vision Projects
Select your Arduino project directory
T32ThreadDebug
T32ThreadDebugLogger
T32ThreadDebugPrint
Board S/N and License State
TRACE32 tutorial for Portenta H7
B::
components trace Data Var List PERF SYStem Step Go Break sYmbol other previous
system down
MIX

Elf file selection for project directory ...\T32ThreadDebug

Elf file: T32ThreadDebug.ino.elf 01/08/2023 16:10:46 4639516.

Automatic search option: Arduino IDE 1 or 2 [✓ current dir]

T32ThreadDebug.ino.elf

Manual search option: File...

Selection mode for the next script execution: [✓ Manual] Auto (time based) Auto (last choice)

Selection mode for the debug symbols: [✓ Load debug symbols (program is already in flash)] Flash program and load debug symbols

Instructions OK

Elf File Selection

First, select either which version of the IDE you have or if you want to use the current directory. The list to the right of that should then become populated with a number of available ELF files for downloading and debugging. Select the one you want with a double-click, this will also show file info such as date, time and size. If you can’t see the one you want there, click the “File” button underneath the “User’s choice” field and browse for the required ELF file.

Next, you can opt to change the behavior of this script the next time it is executed.

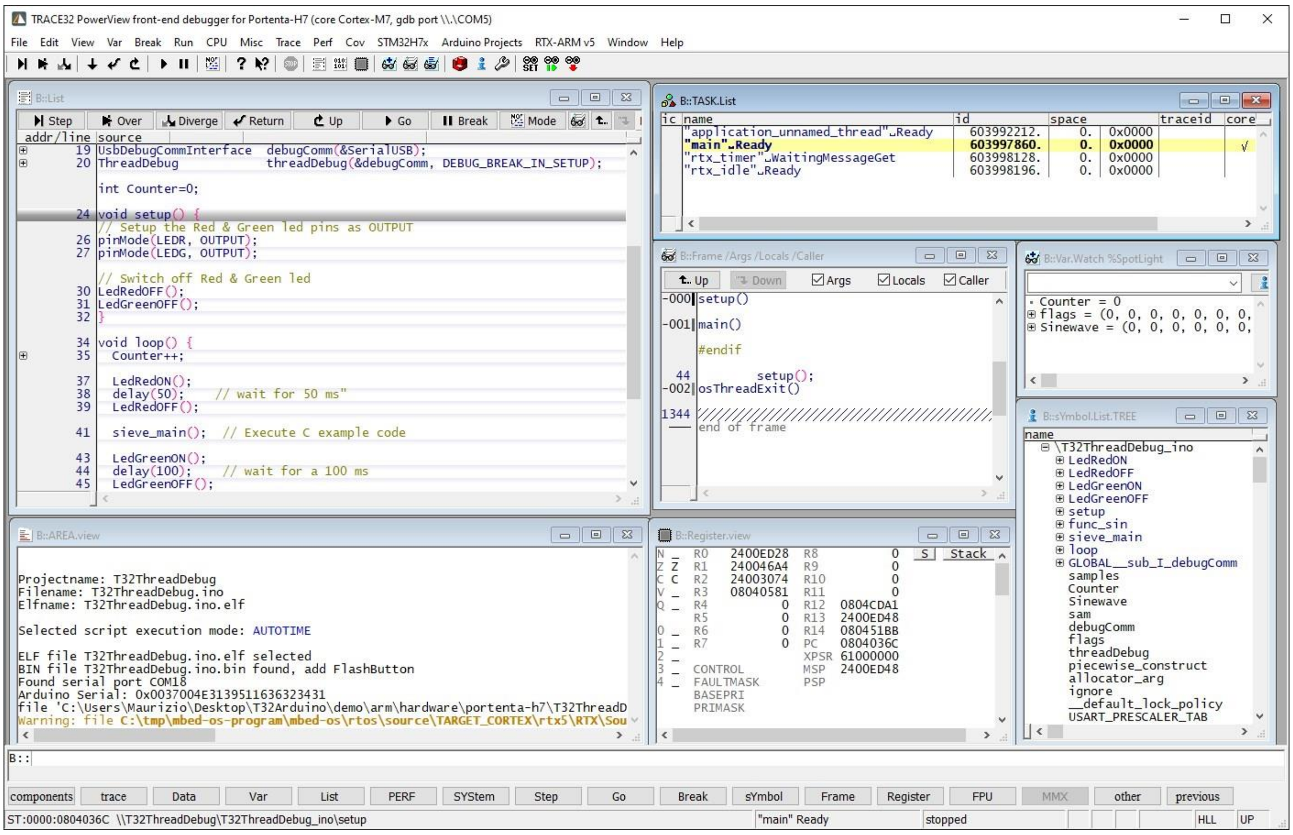
If the application has already been programmed to FLASH, maybe via the Arduino IDE or a previous TRACE32 session, select “Load debug symbols (program is already in flash)” to prevent an un-necessary erase and write of the on-chip FLASH.

If the application has changed or does not match the contents of the FLASH, then select “Flash program and load debug symbols”.

Finally, click the “OK” button to execute all of your choices.

INFORMATION: If the FLASH is being programmed, TRACE32 will prompt you to double-click the reset button on the board to enable the FLASH loader mode and then click the “OK” button when this has been done. Diagnostic messages will be displayed in the TRACE32 AREA window whilst the erase and programming take place.

The script will then attach to the **board** and cause TRACE32 to open some debug windows. When all is ready, you should see the Program Counter parked at the entry to the setup() function.


TRACE32 PowerView front-end debugger for Portenta-H7 (core Cortex-M7, gdb port \\.\COM5)
File Edit View Var Break Run CPU Misc Trace Perf Cov STM32H7x Arduino Projects RTX-ARM v5 Window Help
B::List
Step Over Diverge Return Up Go Break Mode
B::TASK.List
B::Frame /Args /Locals /Caller
Up Down Args Locals Caller
B::Var.Watch %SpotLight
B::sYmbol.List.TREE
B::AREA.view
B::Register.view
components trace Data Var List PERF SYStem Step Go Break sYmbol Frame Register FPU MMX other previous

```
UsbDebugCommInterface  debugComm(&SerialUSB);
ThreadDebug            threadDebug(&debugComm, DEBUG_BREAK_IN_SETUP);

int Counter=0;

void setup() {
// Setup the Red & Green led pins as OUTPUT
pinMode(LEDR, OUTPUT);
pinMode(LEDG, OUTPUT);

// Switch off Red & Green led
LedRedOFF();
LedGreenOFF();
}

void loop() {
  Counter++;

  LedRedON();
  delay(50);     // wait for 50 ms"
  LedRedOFF();

  sieve_main();  // Execute C example code

  LedGreenON();
  delay(100);    // wait for a 100 ms
  LedGreenOFF();
```

```
ic name                                   id           space        traceid  core
   "application_unnamed_thread"␣Ready     603992212.   0.  0x0000
   "main"␣Ready                           603997860.   0.  0x0000             √
   "rtx_timer"␣WaitingMessageGet          603998128.   0.  0x0000
   "rtx_idle"␣Ready                       603998196.   0.  0x0000
```

```
-000 setup()

-001 main()

     #endif

  44         setup();
-002 osThreadExit()

1344 ////////////////////////////////
 ——  end of frame
```

```
· Counter = 0
⊞ flags = (0, 0, 0, 0, 0, 0, 0,
⊞ Sinewave = (0, 0, 0, 0, 0, 0,
```

```
name
  ⊟ \T32ThreadDebug_ino
    ⊞ LedRedON
    ⊞ LedRedOFF
    ⊞ LedGreenON
    ⊞ LedGreenOFF
    ⊞ setup
    ⊞ func_sin
    ⊞ sieve_main
    ⊞ loop
    ⊞ GLOBAL__sub_I_debugComm
      samples
      Counter
      Sinewave
      sam
      debugComm
      flags
      threadDebug
      piecewise_construct
      allocator_arg
      ignore
      __default_lock_policy
      USART_PRESCALER_TAB
```

```
Projectname: T32ThreadDebug
Filename: T32ThreadDebug.ino
Elfname: T32ThreadDebug.ino.elf

Selected script execution mode: AUTOTIME

ELF file T32ThreadDebug.ino.elf selected
BIN file T32ThreadDebug.ino.bin found, add FlashButton
Found serial port COM18
Arduino Serial: 0x0037004E3139511636323431
file 'C:\Users\Maurizio\Desktop\T32Arduino\demo\arm\hardware\portenta-h7\T32ThreadD
Warning: file C:\tmp\mbed-os-program\mbed-os\rtos\source\TARGET_CORTEX\rtx5\RTX\Sou
```

```
N _  R0   2400ED28  R8          0   S  Stack
Z Z  R1   240046A4  R9          0
C C  R2   24003074  R10         0
V _  R3   08040581  R11         0
Q _  R4          0  R12  0804CDA1
     R5          0  R13  2400ED48
0 _  R6          0  R14  080451BB
1 _  R7          0  PC   0804036C
2 _                 XPSR 61000000
3 _  CONTROL        MSP  2400ED48
4 _  FAULTMASK      PSP
     BASEPRI
     PRIMASK
```

```
B::
ST:0000:0804036C  \\T32ThreadDebug\T32ThreadDebug_ino\setup     "main" Ready     stopped     HLL  UP
```

TRACE32 debug session on Portenta H7 – T32ThreadDebug demo application

In case of errors, please check the physical connection, check if your host PC has detected the board serial port and if this is the port configured in TRACE32, reset the board once and re-try.

Please take a look at the file **readme.txt** inside the demo directory for a demo description.

Compile and debug other projects

The demos or another project can be edited, compiled and flashed with the Arduino IDE. Flashing is also possible with TRACE32. You can open for example the T32ThreadDebug.ino file with Arduino IDE, build and flash it.

IMPORTANT: If you chose to program the FLASH from within the Arduino IDE Classic, do not close the IDE after programming; leave it open. This is very important because if you close the IDE, it cleans up the temporary build directory which includes the ELF file.

The startup script already present in the demo directory will copy the Elf file and the binary file in the current working directory. Later you can safely close the Arduino IDE Classic if you don’t need it any more.

Switch back to TRACE32, select Arduino IDE 1 or 2, if an ELF file is found double-click to select it, then select "Load debug symbols (program is already in flash)", and finally, click "OK".

INFORMATION: The minimal startup script is reported below. Please copy it into a text file and save it with a file extension ".cmm". Then call the menu "File-->Run Script..." from the TRACE32 GUI.

```
SYStem.Down
SYStem.CPU PORTENTAH7-CM7  ; or SYStem.CPU NICLAVISION-CM7
SYStem.PORT <serial_port>   ; e.g. COM8 (Windows) or /dev/ttyUSB0 (Linux)
SYStem.Option MMUSPACES ON
Break.CONFIG.METHOD.Program Onchip

SYStem.Mode Attach

Data.LOAD.Elf * /NoCODE

TASK.CONFIG ~~/demo/arm/kernel/rtxarm/v5/rtx.t32
MENU.ReProgram ~~/demo/arm/kernel/rtxarm/v5/rtx.men

List.auto
ENDDO
```

You can also copy the script **start.cmm** from the T32ThreadDebug demo directory to your working directory.

INFORMATION: The script **start.cmm** has a predefined window layout for each demo. For your own layout, manually open and arrange the windows as you prefer, then save this window layout using the "Store Windows..." command in the Window menu. Save the file as **win.cmm**. It will be automatically found and used at the next startup.

Conclusion

In this tutorial you learned how to acquire a free version of TRACE32 GDB Front End debugger fully licensed for Portenta H7 or Nicla Vision boards, how to start the debugger and debug some ready-to-run demos. You also learned how to debug an application compiled with Arduino IDE.

Next Steps

Lauterbach also provides hardware-based debug & trace tools. For more information please refer to:

https://www.lauterbach.com
https://www.lauterbach.com/microtrace.html

Troubleshooting

Debug connection issues

- Reset the **board** before running the startup script.
- In case of further errors while attaching to the **board**, please check the physical connection, check if your host PC has detected the board serial port and if this is the port configured in TRACE32, reset the board once and re-try.

Flashing issues

- Before flashing the **board** from the Arduino IDE, please disconnect the TRACE32 debugger typing the command: “SYStem.Down” on the command line interface or opening the menu: “CPU-> System Settings…” and pressing the radiobutton “Down” in the “Mode” section.

Debugger hang issues

- The TRACE32 GDB Front End debugger is a run-mode debugger: at a breakpoint only the user threads are stopped, while the kernel and all other system threads continue to run. It may happen that the debugger hangs if a breakpoint is set on critical system areas. In this case reset the board, remove any breakpoint and attach again to the target (SYStem.Mode Attach).

Issues while starting the TRACE32 executable for host Linux

- The TRACE32 executable for host Linux requires Qt libraries. Please check that Qt is installed:
  - Qt5 >= 5.9 (Linux 64 bit)

Issues with the GDB serial port on host Linux

- The user running the TRACE32 executable for host Linux must have the permissions to access serial devices. For example in Ubuntu a temporary permission can be set in this way:

  sudo chown :*username* /dev/*devicename*

  You can also set a permanent permission adding the user to the dialout group. For example in Ubuntu:

  sudo adduser *username* dialout

  or you can run the TRACE32 executable with root permissions.

Authors:	Marco Ferrario, Lauterbach Italy
Reviewed by:	Maurizio Menegotto, Lauterbach Italy, Richard Copeman, Lauterbach UK
First revision:	28 October 2020
Last revision:	02 August 2023