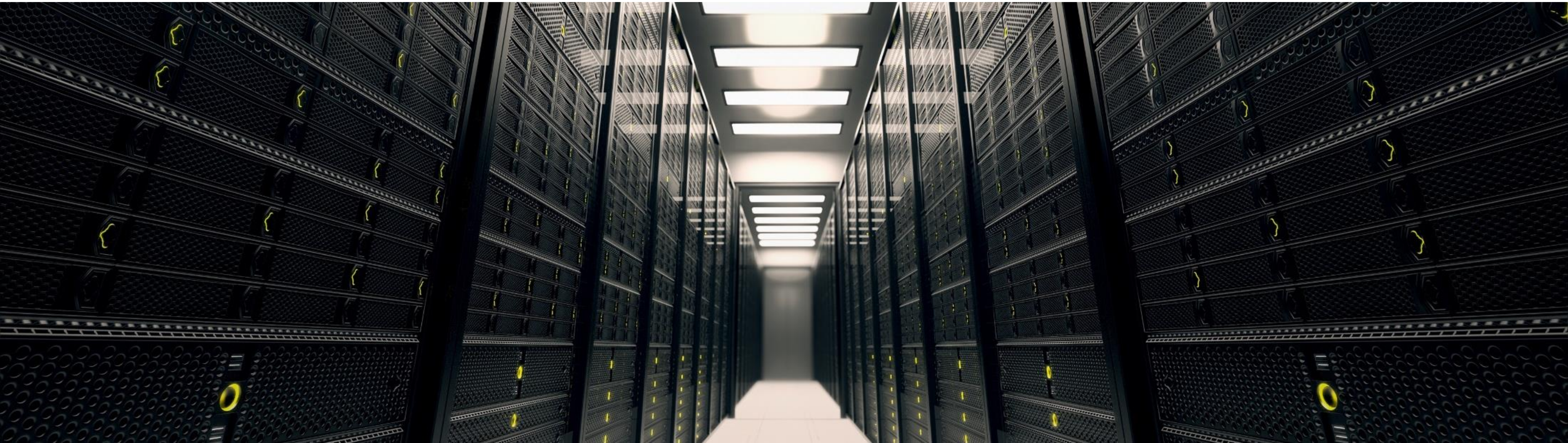
ARM mbed with us

Debugging and Tracing



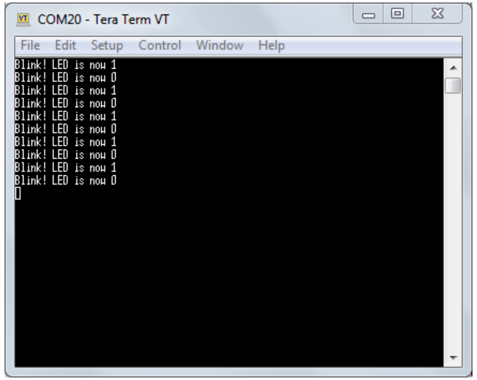
Summary

- We are going to explain how to debug – trace the Sensor Node project
- You will learn two way of debugging – tracing :
    1. The embedded mbed OS tracing
    2. A classic way with an IDE: uVision 5

- Prerequisites:
    - Driver en.stsw-link009.zip file
    - mbed CLI and uVision 5 installed

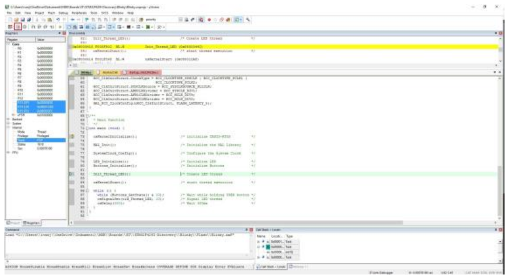
Debugging

- Embedded mbed OS tracing: debugging with printf calls
  - The simplest way to debug your code is to augment your code with log statements, which can be observed from your computer
  - It requires to add trace functions and enable defines to obtain more information about what is going on



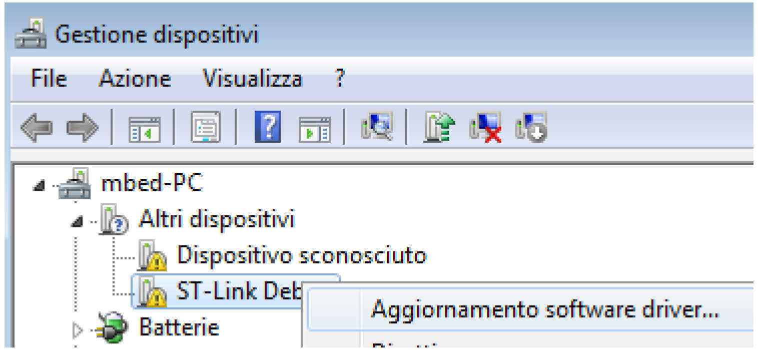
```
Blink! LED is now 1
Blink! LED is now 0
Blink! LED is now 1
Blink! LED is now 0
Blink! LED is now 1
Blink! LED is now 0
Blink! LED is now 1
Blink! LED is now 0
Blink! LED is now 1
Blink! LED is now 0
```

- Debugging from an IDE: uVision 5, standard ICE tool
  - you can do things such as set breakpoints, set watchpoints, view registers, view disassembly, browse memory and examine the callstack.
  - Keil uVision natively supports debugging mbed OS applications. mbed also supports debugging using any IDE that supports GDB.



Setup

- Installing drivers:
  - Connect the Sensor Node to the PC
  - Install manually the drivers provided from the STSW-LINK009 zip file
    - Driver for ST-Link v2 (ST-Link Debug)
    - Driver for VCom Port (unknown device)



mbed-trace 1/6

- mbed-trace is a library for tracing via serial line:
    - It's a light, simple and general tracing solution for mbed devices
    - The memory space required by the library is allocated at the initialization only once (see mbed_trace_init function)
    - The trace function uses stdout as the default output target : it goes directly to serial port when initialized
    - A trace method call produces a single line containing <level>, <group> and <message>

      where <level>, <group> and <message> are module or common module defines
    - The solution is not Interrupt safe and it is not Thread safe by default (see mbed_trace_mutex_wait_function_set/mbed_trace_mutex_release_function_set functions)
    - **Tracing may affect the timing response of the system**

mbed-trace 2/6

- Format of the mbed-trace messages:

```
[DBG ][abc ]: This is a debug message from module abc<cr><lf>
[INFO][br  ]: Hi there.<cr><lf>
[WARN][br  ]: Oh no, br warning occurs!<cr><lf>
[ERR ][abc ]: Something goes wrong in module abc<cr><lf>
```

- The <level> "DBG", "INFO", "WARN", "ERR", specify the level of information that gets included in debug log. It is the type and amount of information that is logged for different events.
- tr<level> macros: tr_debug, tr_info, tr_warning, tr_error

- In every source module .c/.cpp where trace is needed a TRACE_GROUP must be defined. It is a 1-4 characters long char-array.

- The messages are assembled with sprintf implementation (see mbed_trace_print_function_set function).

mbed-trace 3/6

- mbed_app.json is the main file in the mbed project where you can edit your project in order to compile it in different ways. Example:
  - Adding macros you can insert defines in the compilation
  - You can enable tracing debug

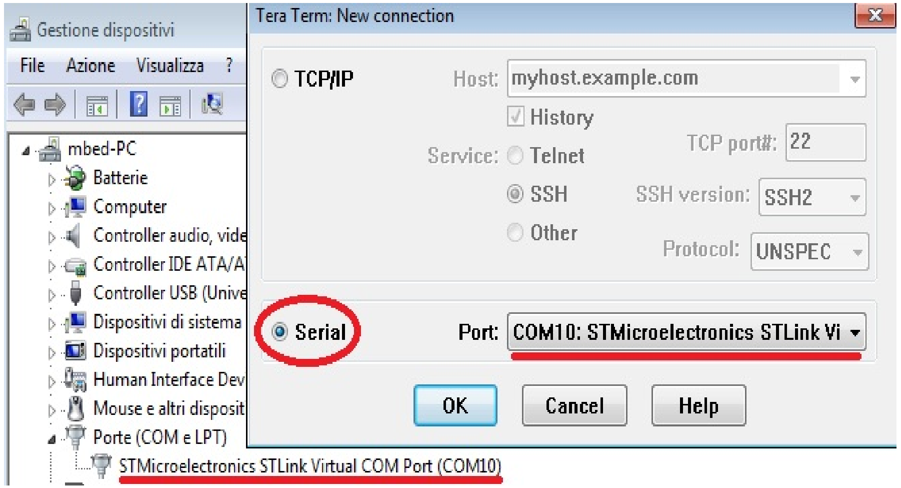
- To enable trace for tracing:
  - Add the feature COMMON_PAL into the build via mbed_app.json in the section target_overrides:

```
"target_overrides": {
"*": {
        "target.features_add": ["NANOSTACK", "LOWPAN_ROUTER", "COMMON_PAL"],
```

mbed-trace 4/6

– Set MBED-TRACE-ENABLE to 1 or true:

```
{
        "target_overrides": {
                "*": {
                        "target.features_add": ["COMMON_PAL"],
                        "mbed-trace.enable": 1
                }
        }
}
```



mbed-PC
- Batterie
- Computer
- Controller audio, vide
- Controller IDE ATA/A
- Controller USB (Unive
- Dispositivi di sistema
- Dispositivi portatili
- Human Interface Dev
- Mouse e altri disposit
- Porte (COM e LPT)
  - STMicroelectronics STLink Virtual COM Port (COM10)

TCP/IP Host: myhost.example.com
History [✓]
Service: Telnet [✓ SSH] Other
TCP port#: 22
SSH version: SSH2
Protocol: UNSPEC
[✓ Serial] Port: COM10: STMicroelectronics STLink Vi

– Hands on:
  – Change the mbed_app.json file and compile the project in order to have the trace enabled
  – Upload the firmware and use Tera-term to see the tracing debug (baud: 115200, 8N1)

mbed-trace 5/6

- The trace library is initialized in the *main* function
- You can set the verbosity of the debug by the function:

```
mbed_trace_config_set(TRACE_MODE_COLOR | TRACE_ACTIVE_LEVEL_INFO | TRACE_CARRIAGE_RETURN);
```

- Where:
  - TRACE_ACTIVE_LEVEL_ALL or TRACE_ACTIVE_LEVEL_DEBUG: to activate all trace levels
  - TRACE_ACTIVE_LEVEL_INFO, TRACE_ACTIVE_LEVEL_WARN, TRACE_ACTIVE_LEVEL_ERROR, TRACE_ACTIVE_LEVEL_CMD: different levels of trace priorities
  - TRACE_LEVEL_NONE: no trace at all

- Hands on:
  - change the level of the trace and test it on the board
  - Save the tracing on file using Putty

mbed-trace 6/6

- Printf function is time consuming. In some modules the time can be critical and the debug enabled may alter the right functionality of the demo

- Sometime you don’t want some “group” trace messages printed. Using the define MBED_TRACE_MAX_LEVEL you can choose the verbosity in a specific module. To silence the trace in some specific module create the define:

```
#define MBED_TRACE_MAX_LEVEL 0
```

- This define must be created before the line #include “mbed_trace.h”

- *Hands on:*
  - *disable the SPIRIT group trace messages adding the definition in the module: easy-connect/stm-spirit1-rf-driver/source/NanostackRfPhySpirit1.cpp*
  - *Check the SPIRIT group messages are not printed*

Debug by macros

- Some debug can be enabled via macros written in the mbed_app.json file
- For example you can enable the debug of the TLS (Transport Layer Security) protocol:
    - TLS protocol is composed by initial frames exchanged from the node and the device connector in order to start a secure connection
- From the mbed_app.json add these macros:

```
"macros": [ ...
        “MBEDTLS_DEBUG_C=1",
          "ENABLE_MBED_CLIENT_MBED_TLS_DEBUGS=1"],
```

- From the file "mbedtls_mbed_client_config.h" comment the line:

```
#undef MBEDTLS_DEBUG_C
```

- Note: mbedtls_mbed_client_config.h file could interfere with the macros defined by mbed_app.json. If you define macros check if this file disable your macro.

- *hands on:*
    - *Compile the project enabling the debug of the TLS communication and save it in a log*
    - *Advanced: find where the macros affect the code*

Debug by macros - details

– The macros defined previously affects the module "mbed-client/mbed-client-mbed-tls/source/m2mconnectionsecuritypimpl.cpp enabling the define:

```
//Comment out following define to enable tracing from mbedtls
//#define ENABLE_MBED_CLIENT_MBED_TLS_DEBUGS
#ifdef ENABLE_MBED_CLIENT_MBED_TLS_DEBUGS
```

– The other macro affects the module mbed-os/features/mbedtls/inc/mbedtls/debug.h

```
#if defined(MBEDTLS_DEBUG_C)

#define MBEDTLS_DEBUG_STRIP_PARENS( ... )   __VA_ARGS__

#define MBEDTLS_SSL_DEBUG_MSG( level, args )                    \
   mbed_tls_debug_print_msg( ssl, level, __FILE__, __LINE__,    \
 ...
```

– Where MBEDTLS_SSL_DEBUG_MSG is used in the module mbed-os/features/mbedtls/src/ssl_tls.c responsable for the TLS communication

uVision 5 (1/2)

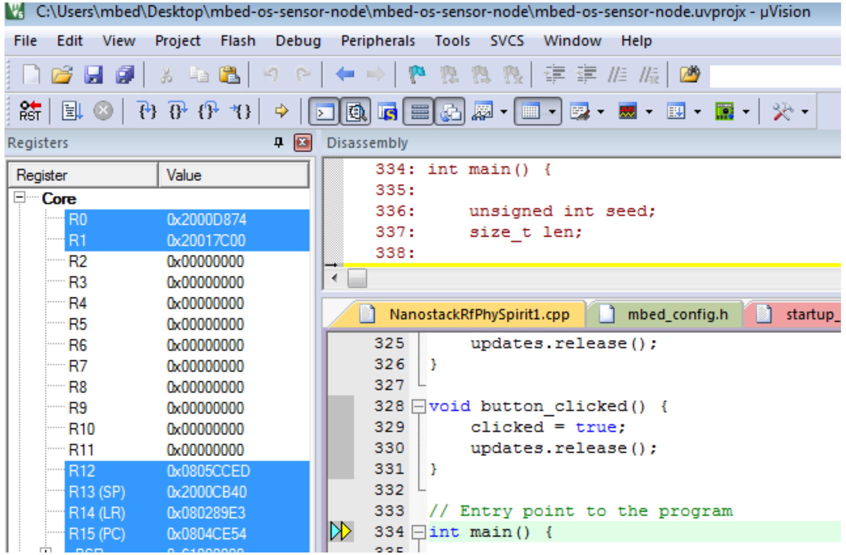
– mbed CLI permits to export the project to use on different IDE using the command:

```
mbed export -i uvision5
```

– The IDE is the easiest way to debug the code:
    1. Launch uVision 5
    2. Open the project clicking on “*Project → Open Project...*” and select the *mbed-os-sensor-node.uvprojx* file project
    3. Build the project clicking on ”*Project → Build target*”
    4. Now you can debug clicking on *“Debug → Start/Stop Debug Session“*

uVision 5 (2/2)

– Now you can debug the board with the standard debugging interface



C:\Users\mbed\Desktop\mbed-os-sensor-node\mbed-os-sensor-node\mbed-os-sensor-node.uvprojx - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
NanostackRfPhySpirit1.cpp
mbed_config.h
startup_

Register	Value
Core	
R0	0x2000D874
R1	0x20017C00
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x0805CCED
R13 (SP)	0x2000CB40
R14 (LR)	0x080289E3
R15 (PC)	0x0804CE54

```
334: int main() {
335:
336:     unsigned int seed;
337:     size_t len;
338:
```

```
    updates.release();
}

void button_clicked() {
    clicked = true;
    updates.release();
}

// Entry point to the program
int main() {
```

Thank you!