Getting started with SEGGER Eval Software and IAR

Document AN00002-R1

Date: July 8, 2011

A product of SEGGER Microcontroller GmbH & Co. KG

www.segger.com

Disclaimer

Copyright notice

Trademarks

Contact address

SEGGER Microcontroller GmbH & Co. KG

In den Weiden 11
D-40721 Hilden

Germany

Tel.+49 2103-2878-0
Fax.+49 2103-2878-28
Email: support@segger.com
Internet: http://www.segger.com

Manual versions

This manual describes the latest software version. If any error occurs, please inform us and we will try to assist you as soon as possible.

For further information on topics or routines not yet specified, please contact us.

Revision	Date	By	Explanation
0	080709	OO	Initial version.
1	110708	OO	- Some minor changes - Added emUSB Host descriptions.

About this document

Assumptions

This document assumes that you already have a solid knowledge of the following:

- The software tools used for building your application (assembler, linker, C compiler)
- The C programming language
- The target processor
- DOS command line.

If you feel that your knowledge of C is not sufficient, we recommend The C Programming Language by Kernighan and Richie (ISBN 0-13-1103628), which describes the standard in C-programming and, in newer editions, also covers the ANSI C standard.

How to use this manual

This manual explains all the functions and macros that the product offers. It assumes you have a working knowledge of the C language. Knowledge of assembly programming is not required.

Typographic conventions for syntax

This manual uses the following typographic conventions:

Style	Used for
Body	Body text.
`Keyword`	Text that you enter at the command-prompt or that appears on the display (that is system functions, file- or pathnames).
`Parameter`	Parameters in API functions.
`Sample`	Sample code in program examples.
Reference	Reference to chapters, sections, tables and figures or other documents.
GUIElement	Buttons, dialog boxes, menu names, menu commands.
Emphasis	Very important sections

Table 1.1: Typographic conventions

j-link
USB
JTAG
SEGGER

Table of Contents

Chapter 1

Introduction

This chapter describes the purpose of this manual and of the entire software package, which consists of different component.

1.1 About this manual

The purpose of this document is to describe the SEGGER eval software components and samples delivered in this eval package.

1.2 What is the purpose of this eval package ?

This eval package has been designed to provide customers and potential customers (you) with a complete, easy to use software package for the specified target hardware and IAR’s Embedded Workbench for ARM (target compiler).

It allows you to easily check out the target hardware, the target compiler and our software components. This evaluation process typically does not take a lot of time since the software can be easily recompiled and downloaded to the target.

1.2.1 What are the components of the software package?

The SEGGER software components are provided in library form, the applications are provided in source code form. Most packages also come with a "`Prebuild`"-folder, containing prebuild executables, which simply need to be downloaded to the target.

1.2.2 Can I recompile the application supplied?

Yes. All you need is the target compiler/eval version of the target compiler. Refer to "`ReadMe.txt`" in the "`Start`"-folder for more information.

1.2.3 Can I write my own applications with this eval package?

Yes, you can write your own applications. However the purpose of this eval package is to test the soft- and hardware for fitnes. You may not use the result in a product.

1.3 Software components in the package

1.3.1 emFile

emFile is SEGGER's embedded file system that can be used on any media for which you can provide basic hardware access functions.
emFile is a high-performance software that has been optimized for speed, versatility and memory footprint.

emFile documentation can be found under "`Doc\UM02001_emFile.pdf`".

1.3.2 emWin

emWin is SEGGER's embedded Graphical User Interface (GUI) using a feature rich API and providing an efficient, processer- and LCD controller-independent GUI for any application that operates with a display.

emWin documentation can be found under "`Doc\UM03001_emWin5.pdf`".

1.3.3 embOS

embOS is SEGGER's embedded priority-controlled multitasking system. It is designed to be used as an embedded operating system for the development of real-time applications and has been optimized for minimum memory consumption in both RAM and ROM, as well as high speed and versatility.

embOS documentation can be found under "`Doc\UM01001_embOS_Generic.pdf`" and "`Doc\UM01002_embOS_ARM_IAR.pdf`".

1.3.4 embOS/IP

embOS/IP is SEGGER's embedded TCP/IP stack. It is a CPU independent, high-performance TCP/IP stack that has been optimized for speed, versatility and small footprint.

embOS/IP documentation can be found under "`Doc\UM07001_embOSIP.pdf`".

1.3.5 emUSB Device

emUSB Device is SEGGER's embedded USB Device stack. It is written in ANSI C and features bulk communication as well as device classes such as MSD, CDC or HID.

emUSB Device documentation can be found under "`Doc\UM09001_emUSB.pdf`".

1.3.6 emUSB Host

emUSB Host is SEGGER's embedded USB Host stack. It is written in ANSI C and features bulk communication as well as device classes such as MSD or HID.

emUSB Host documentation can be found under "`Doc\UM10001_emUSBH.pdf`".

Chapter 2

Licensing

This chapter gives information about the licensing terms under which the SEGGER Eval Software is published.

2.1 License terms

The SEGGER Eval Software package may only be used when fully agreeing to the terms mentioned in this chapter and agreeing to the license terms mentioned in "`License.txt`" .

If this description contradicts the terms of the "`License.txt`", the terms of the license file should superseed this description.

In case of doubt, please contact us: info@segger.com .

2.1.1 What you may do

You may use the software contained in this eval package to evaluate SEGGER software on the target hardware.
You may recompile and modify the sample programs provided as part of the package.

2.1.2 What you are not allowed to do

You are not allowed to use the software in this eval package for something other than evaluating the SEGGER software. You are not allowed to use this software package in a product.

Chapter 3

Software components

This chapter explains the folder structure used for the software components as well as the project structure and the limitations of the eval version.

3.1 Setup

Requirements

In order to recompile the projects you will need the IAR Embedded Workbench as indicated in the ReadMe.txt file.

In order to recompile the emWin simulation you will need a C-compiler. The shipping contains an ready to go project for Microsoft Visual C++ 6.0. Microsoft Visual C++ 6.0 or Microsoft Visual Studio .Net are required to use this project.

Installation

All eval packages are supplied as a zip-file. The latest version of the software can be archieved from the following location:
`http://www.segger.com/cms/evalboards.html`

Extract it to any folder of your choice, preserving the directory structure of the zip-file. Assuming that you are using the IAR Embedded Workbench project manager to develop your application, no further installation steps are required. You will find prepared sample start applications, which you should use and modify to write your application.

All eval packages include evaluation builds some or all of the following products:

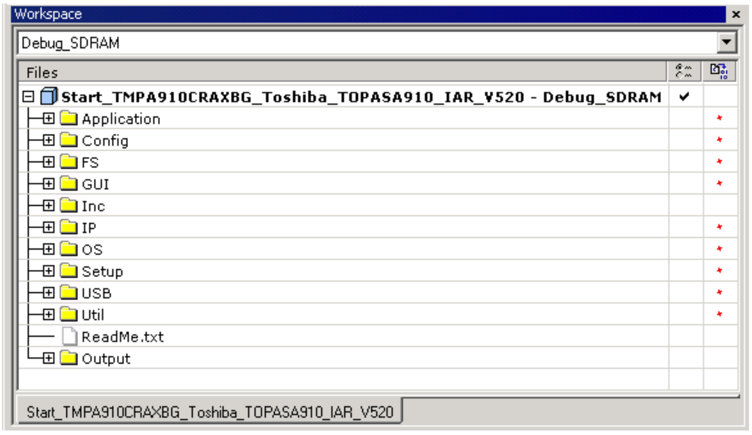
Component	Description
emFile	SEGGER's file system for embedded applications
embOS	SEGGER's priority-controlled real time operating system
emOS/IP	SEGGER's CPU independent TCP/IP stack
emUSB Device	SEGGER's high speed USB Device stack
emUSB Host	SEGGER’s high speed USB Host stack
emWin	SEGGER’s graphics software and GUI

Table 3.1: Eval package components

All products can be combined and purchased separately. For ordering information, please contact SEGGER, *www.segger.com*.

3.2 Project structure

All components of the eval package are provided as libraries and the required header files. The sample applications are provide in source form. The sample projects are organized in the following way:



Workspace

Debug_SDRAM

Files

```
Start_TMPA910CRAXBG_Toshiba_TOPASA910_IAR_V520 - Debug_SDRAM
├─ Application
├─ Config
├─ FS
├─ GUI
├─ Inc
├─ IP
├─ OS
├─ Setup
├─ USB
├─ Util
├── ReadMe.txt
└─ Output
```

Start_TMPA910CRAXBG_Toshiba_TOPASA910_IAR_V520

The root directory of the eval package includes three folders, "Doc", "Prebuild", "Start", License.txt and this documentation.

The "Prebuild"-folder may contain prebuild executables, which simply need to be downloaded to the target.

The "Doc"-folder contains the user guides for the included SEGGER Eval Software. The "Start"-folder contains all files necessary for the SEGGER Eval Software itself.

The "Start"-folder may contain an additional directory (Windows), which is not included in the project tree of the IAR Embedded Workbench project. The Start folder may contain some or all of the directories described in the table below:

Directory	Description
Application	Includes the sample applications.
Config	Includes the configuration files of the included software packages.
FS	Includes the emFile header files and libraries.
GUI	Includes the emWin header files and libraries.
Inc	Includes utility header files which are not directly related to one of the included software packages.
IP	Includes the embOS/IP header files and libraries.
OS	Includes the embOS header files and libraries as well as `Main.c` and `OS_Error.c` .
Setup	Includes the Board Support Package (BSP). The BSP consists all files which are required to initialize the hardware and build a project.
USB	Includes the emUSB Device header files and libraries.
USBH	Includes the emUSB Host header files and libraries.
Util	Includes utility functions which are not directly related to one of the included software packages.
Windows	Includes the Microsoft Windows examples and the required drivers for the USB components BULK communication and Communication Device Class (CDC). Refer to the `User's and reference manual for emUSB` [UM09001]for more information about the installation of these drivers.

Table 3.2: Eval package structure

3.3 Eval limitations

The included eval versions of the different components of the eval package have the following limitations:

Component	Description
emFile	The eval version of the emFile libraries can only handle one open file at any given time.
embOS	The eval version of the embOS libraries runs without a time limit with a maximum of three tasks. If your application creates more than three tasks embOS stops after a defined time limit. For embOS versions before v3.82t the time limit is 12 minutes. Versions including v3.82t and later the time limit is 12 hours.
emOS/IP	The eval version of the embOS/IP libraries have a time limit of 12 hours on the connection.
emUSB Device	The eval version of the emUSB Device libraries have a time limit of 12 hours on the connection.
emUSB Host	The eval version of the emUSB Host libraries have a time limit of 12 hours on the connection.
emWin	The eval version of the emWin library shows an evaluation notification before the actual application starts.

Table 3.3: Limitations of eval package components

Your use of the eval package or of any part included in the project indicates your acknowledgment and agreement to the SEGGER eval software license . `License.txt` is located in the `root` directory of the eval package.

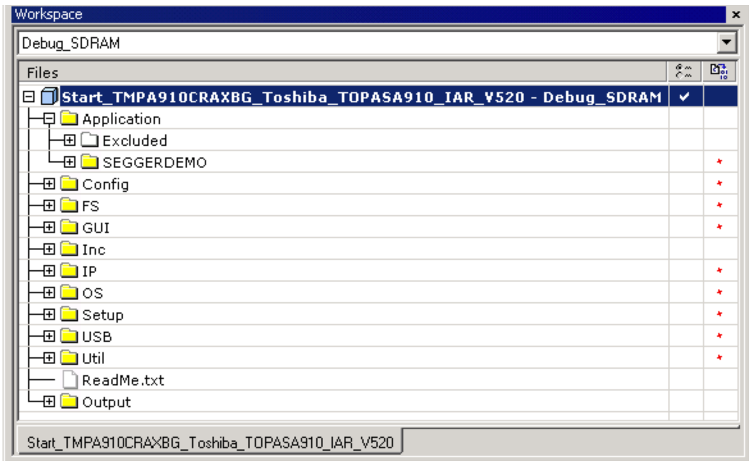
Chapter 4

Getting started

This chapter contains all required information to start working with the sample applications.

4.1 Running a sample application

The application folder includes the selected sample application which may consist of a file or a folder, normally `SEGGERDEMO` and a folder which is excluded from build. The `SEGGERDEMO` example is only available if `emWin` is supported for the hardware. If `emWin` is not available, the default sample may be either `OS_Start_LEDBlink.c` or `OS_Start_2Tasks.c` if no LEDs are supported. The **Excluded** folder contains all available sample applications for this hardware. All available samples contain a function `MainTask()`. `Maintask()` is called from program main() which is located in the **OS** folder in the `Main.c` file.

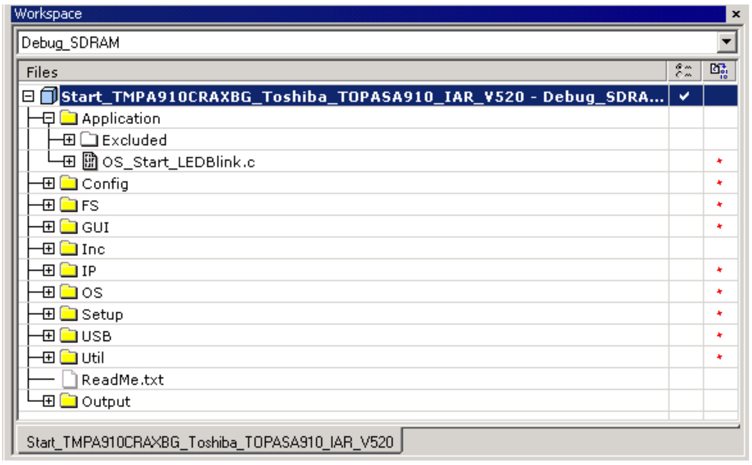

Workspace
Debug_SDRAM
Files
Start_TMPA910CRAXBG_Toshiba_TOPASA910_IAR_V520 - Debug_SDRAM ✓
Application
Excluded
SEGGERDEMO *
Config *
FS *
GUI *
Inc
IP *
OS *
Setup *
USB *
Util *
ReadMe.txt
Output
Start_TMPA910CRAXBG_Toshiba_TOPASA910_IAR_V520

`Main.c` is required by all supplied sample applications.

```
#include "RTOS.H"
#include "BSP.h"

void MainTask(void);

static OS_STACKPTR int Stack0[512]; /* Task stacks          */
static OS_TASK TCB0;                /* Task-control-blocks */

/**********************************************************************
*
*       main
*/
void main(void) {
  OS_IncDI();                      /* Initially disable interrupts  */
  OS_InitKern();                   /* Initialize OS                 */
  OS_InitHW();                     /* Initialize Hardware for OS    */
  BSP_Init();                      /* Initialize BSP module         */
  BSP_SetLED(0);
  OS_CREATETASK(&TCB0, "MainTask", MainTask, 100, Stack0);
  OS_Start();
}
```

`main()` creates a single task called `MainTask`, which is implemented in every sample application.

Open the **Application** folder and use "drag and drop" in the **Workspace** window of the IAR Embedded Workbench to change the sample application. Move the included sample application (e.g. `OS_Start_LEDBlink.c`) in the **Excluded** folder and move the favored sample application from the **Excluded** folder to the **Application** folder.

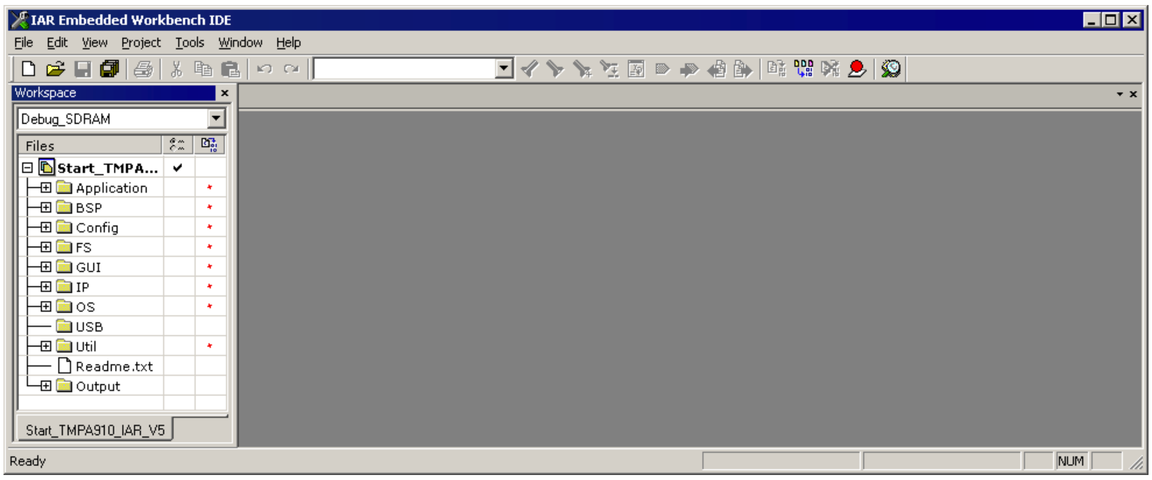


Some embOS/IP and emUSB sample applications are client/server applications and consist of an embedded side and a PC side. The PC applications are provided in source code and as precompiled executable (.exe). For compiling the example application you need a Microsoft compiler. The compiler is part of Microsoft Visual C++ 6.0 or Microsoft Visual Studio .Net. All Windows host sample applications are located in the Windows folder of your eval project.

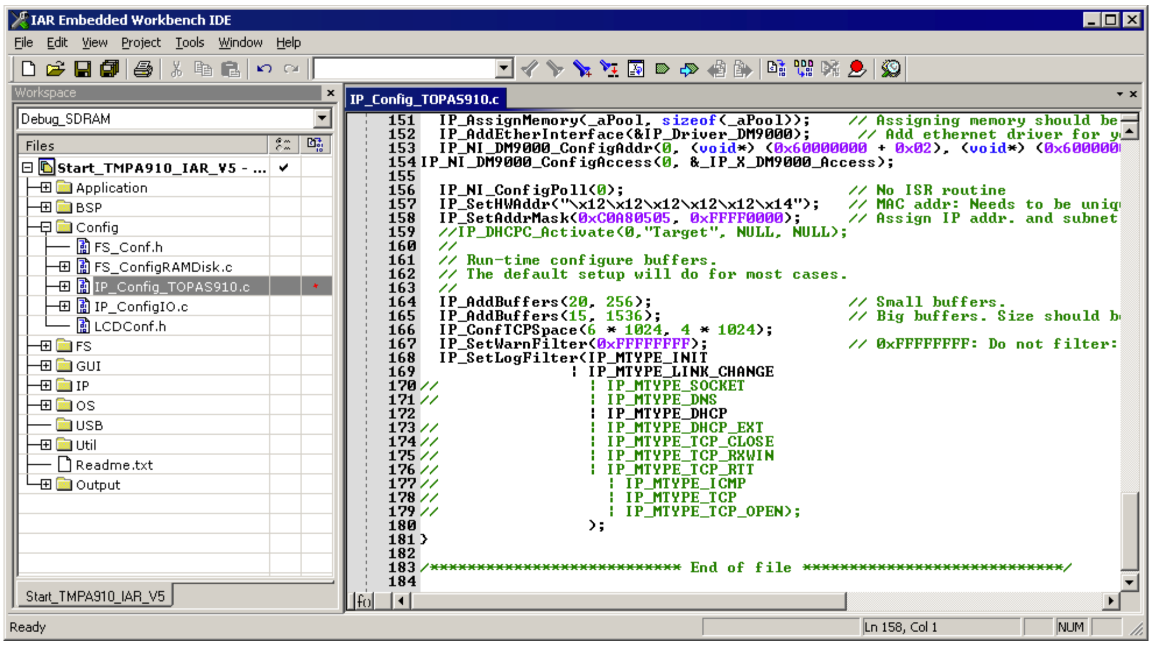
4.2 Evaluating the SeggerDemo

The following steps describe what to do step by step to fully evaluate the Segger-Demo software on a Toshiba TOPASA910 eval board. The steps described below may differ from hardware to hardware but should be similar for other hardware in general.

1. Get a fresh copy of the project and files.
2. Open the project file "`Start_<BSPNAME>_IAR_V5xx.eww`" with the IAR Embedded Workbench. Your screen should look similar to the screenshot below.



3. If you are not using a DHCP server in your network you will need to enter an IP address and subnetmask into the file "`Start\Config\IP_Config_TOPAS910.c`" manually. The screenshot below shows an example for the IP address `192.168.5.5` with subnetmask `255.255.0.0` . *



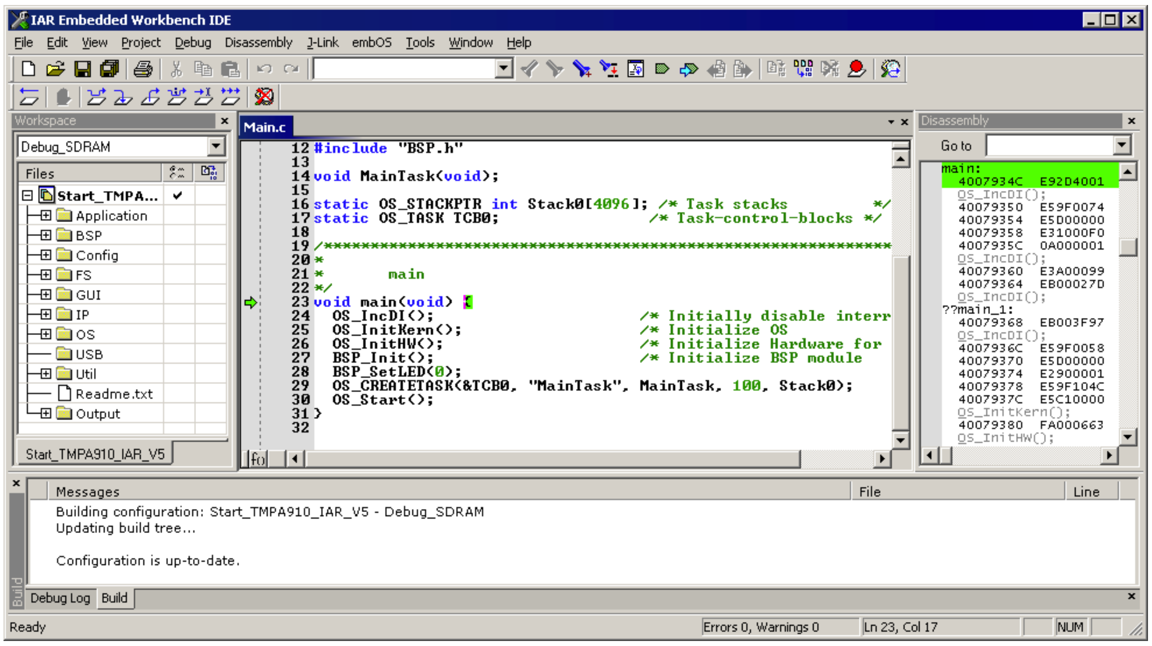
```
  IP_AssignMemory(_aPool, sizeof(_aPool));     // Assigning memory should be
  IP_AddEtherInterface(&IP_Driver_DM9000);      // Add ethernet driver for y
  IP_NI_DM9000_ConfigAddr(0, (void*) (0x60000000 + 0x02), (void*) (0x600000
IP_NI_DM9000_ConfigAccess(0, &_IP_X_DM9000_Access);

  IP_NI_ConfigPoll(0);                         // No ISR routine
  IP_SetHWAddr("\x12\x12\x12\x12\x12\x14");    // MAC addr: Needs to be uniq
  IP_SetAddrMask(0xC0A80505, 0xFFFF0000);      // Assign IP addr. and subnet
  //IP_DHCPC_Activate(0,"Target", NULL, NULL);
  //
  // Run-time configure buffers.
  // The default setup will do for most cases.
  //
  IP_AddBuffers(20, 256);                      // Small buffers.
  IP_AddBuffers(15, 1536);                     // Big buffers. Size should b
  IP_ConfTCPSpace(6 * 1024, 4 * 1024);
  IP_SetWarnFilter(0xFFFFFFFF);                // 0xFFFFFFFF: Do not filter:
  IP_SetLogFilter(IP_MTYPE_INIT
                 | IP_MTYPE_LINK_CHANGE
//                 | IP_MTYPE_SOCKET
//                 | IP_MTYPE_DNS
                   | IP_MTYPE_DHCP
//                 | IP_MTYPE_DHCP_EXT
//                 | IP_MTYPE_TCP_CLOSE
//                 | IP_MTYPE_TCP_RXWIN
//                 | IP_MTYPE_TCP_RTT
//                   | IP_MTYPE_ICMP
//                   | IP_MTYPE_TCP
//                   | IP_MTYPE_TCP_OPEN);
                 );
}

/*************************** End of file ****************************/
```

4. Press [F7] to compile the SeggerDemo sample application. The build log should report:

   Total number of errors: 0
   Total number of warnings: 0

5. Make sure that the RS232 cable is connected to your eval board and your PC. **
6. Make sure that the ethernet cable is connected to a hub/switch and your PC is connected to the same hub/switch. *
7. Make sure your eval board is powered.

8. Press `Shift`+[D] to download the application and start a debug session. After download your screen should be similar to the screenshot below, halting at `main()`.



```
12 #include "BSP.h"
13
14 void MainTask(void);
15
16 static OS_STACKPTR int Stack0[4096]; /* Task stacks          */
17 static OS_TASK TCB0;                 /* Task-control-blocks */
18
19 /*******************************************************************
20 *
21 *       main
22 */
23 void main(void) {
24   OS_IncDI();                        /* Initially disable interr
25   OS_InitKern();                     /* Initialize OS
26   OS_InitHW();                       /* Initialize Hardware for
27   BSP_Init();                        /* Initialize BSP module
28   BSP_SetLED(0);
29   OS_CREATETASK(&TCB0, "MainTask", MainTask, 100, Stack0);
30   OS_Start();
31 }
32
```

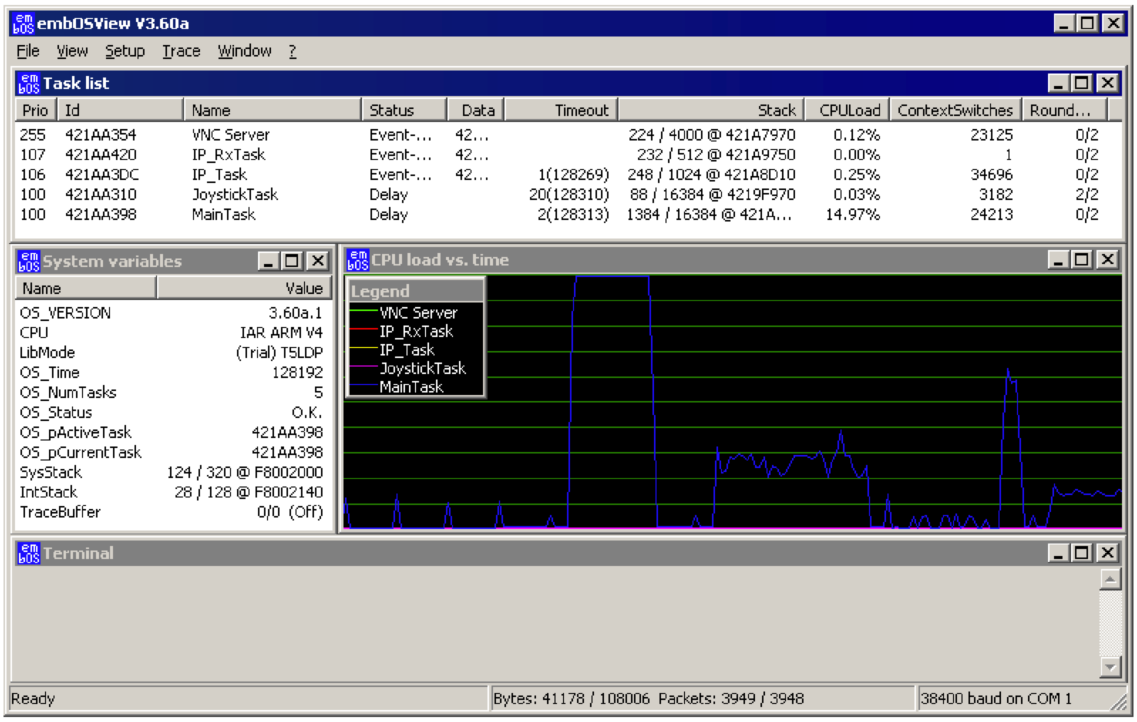
```
main:
  4007934C  E92D4001
  OS_IncDI();
  40079350  E59F0074
  40079354  E5D00000
  40079358  E31000F0
  4007935C  0A000001
  OS_IncDI();
  40079360  E3A00099
  40079364  EB00027D
  OS_IncDI();
??main_1:
  40079368  EB003F97
  OS_IncDI();
  4007936C  E59F0058
  40079370  E5D00000
  40079374  E2900001
  40079378  E59F104C
  4007937C  E5C10000
  OS_InitKern();
  40079380  FA000663
  OS_InitHW();
```

```
Messages                                                     File        Line
Building configuration: Start_TMPA910_IAR_V5 - Debug_SDRAM
Updating build tree...

Configuration is up-to-date.
```

```
Ready                     Errors 0, Warnings 0    Ln 23, Col 17    NUM
```

9. Press [F5] to run the application. You should see the SeggerDemo on the LCD display.

10. Start "`Windows\OS\embOSView.exe`" to verify that the UART is sending OS specific data. Your screen should look similar to the screenshot below. **



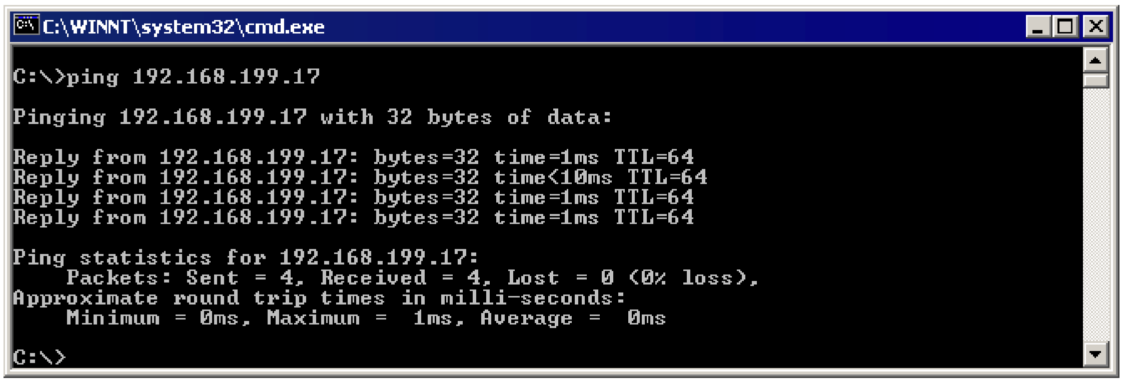
Prio	Id	Name	Status	Data	Timeout	Stack	CPULoad	ContextSwitches	Round...
255	421AA354	VNC Server	Event-...	42...		224 / 4000 @ 421A7970	0.12%	23125	0/2
107	421AA420	IP_RxTask	Event-...	42...		232 / 512 @ 421A9750	0.00%	1	0/2
106	421AA3DC	IP_Task	Event-...	42...	1(128269)	248 / 1024 @ 421A8D10	0.25%	34696	0/2
100	421AA310	JoystickTask	Delay		20(128310)	88 / 16384 @ 4219F970	0.03%	3182	2/2
100	421AA398	MainTask	Delay		2(128313)	1384 / 16384 @ 421A...	14.97%	24213	0/2

Name	Value
OS_VERSION	3.60a.1
CPU	IAR ARM V4
LibMode	(Trial) T5LDP
OS_Time	128192
OS_NumTasks	5
OS_Status	O.K.
OS_pActiveTask	421AA398
OS_pCurrentTask	421AA398
SysStack	124 / 320 @ F8002000
IntStack	28 / 128 @ F8002140
TraceBuffer	0/0 (Off)

```
Ready          Bytes: 41178 / 108006  Packets: 3949 / 3948     38400 baud on COM 1
```

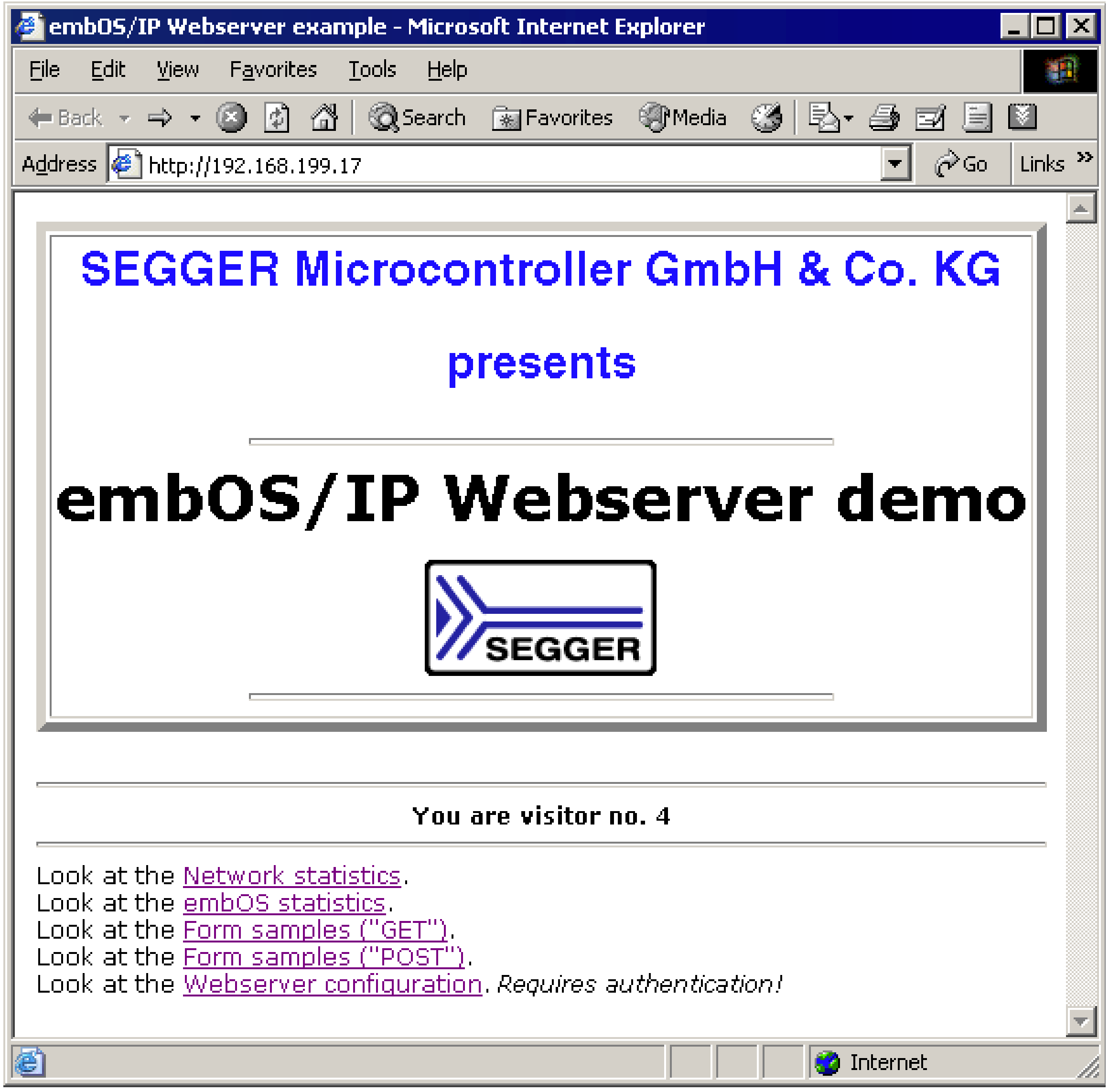
If not please refer to the embOS Generic Documentation (`Doc\embOS_Generic.pdf`) for troubleshooting.

11. At the end of the SeggerDemo slideshow you will see the targets IP address. Open a command prompt and try to ping your target. Your screen should look similar to the screenshot below. *



```
C:\>ping 192.168.199.17

Pinging 192.168.199.17 with 32 bytes of data:

Reply from 192.168.199.17: bytes=32 time=1ms TTL=64
Reply from 192.168.199.17: bytes=32 time<10ms TTL=64
Reply from 192.168.199.17: bytes=32 time=1ms TTL=64
Reply from 192.168.199.17: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.199.17:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  1ms, Average =  0ms

C:\>
```

12. You can now try to connect to the webserver by entering "`http://<target_ip>`" into your webbrowser. your screen should look similar to the screenshot below. *


embOS/IP Webserver example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://192.168.199.17
Go
Links
SEGGER
Internet

SEGGER Microcontroller GmbH & Co. KG

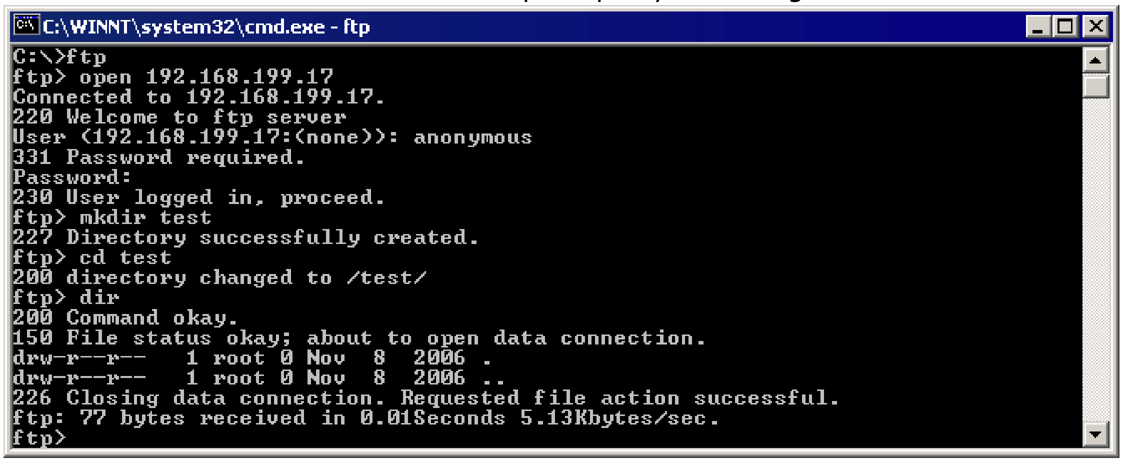
presents

embOS/IP Webserver demo

You are visitor no. 4

Look at the Network statistics.
Look at the embOS statistics.
Look at the Form samples ("GET").
Look at the Form samples ("POST").
Look at the Webserver configuration. *Requires authentication!*

13. You can test the FTP server by connecting to the target on port 21 with a FTP client. You can do this via a command prompt by following the screenshot below. *



```
C:\>ftp
ftp> open 192.168.199.17
Connected to 192.168.199.17.
220 Welcome to ftp server
User (192.168.199.17:(none)): anonymous
331 Password required.
Password:
230 User logged in, proceed.
ftp> mkdir test
227 Directory successfully created.
ftp> cd test
200 directory changed to /test/
ftp> dir
200 Command okay.
150 File status okay; about to open data connection.
drw-r--r--    1 root 0 Nov  8  2006 .
drw-r--r--    1 root 0 Nov  8  2006 ..
226 Closing data connection. Requested file action successful.
ftp: 77 bytes received in 0.01Seconds 5.13Kbytes/sec.
ftp>
```

14. After evaluating all parts of the SeggerDemo you may evaluate the other samples located in the "`Application`"-folder.

*   Only applies if the BSP contains embOS/IP
** Only applies if embOSView is supported by the BSP

Chapter 5

Sample applications

This chapter describes the usage of the most important sample applications.

5.1 List of sample applications

The list below contains a short description of the samples available. All samples are well documented and should have a sample description in the header. In addition the most important sample applications are described in detail in the following sections. All sample applications available for the specific hardware can be found in the "`Start\Application`"-folder.

File(s)	Description
SeggerDemo*	SEGGER product demo showing usage of emWin for showing a product features and usage of emWin features, embOS/IP for running a webserver and FTP server, emFile for providing a filesystem for the FTP server, emUSB for providing MSD access to the filesystem and embOS as RTOS.
emFile samples	
FS_APIValidation.c	Simple generic emFile API test.
FS_CheckDisk.c	Sample program demonstrating FS_CheckDisk functionality.
FS_DirOperations.c	Sample program creating directories and files.
FS_NORSectorInfo.c	Sample program showing sector information of NOR flash devices.
FS_Performance.c	Sample program for performance measuring. Outputs detailed results.
FS_PerformanceSimple.c	Sample program for performance measuring. Outputs reduced result details.
FS_Start.c	Simple application starting and testing the filesystem.
FS_Start_ReadWriteHook.c	Simple application for outputting information on read/write access.
FS_WritePerformanceTest.c	Sample program for performance measuring.
emWin samples	
GUI_ALPHA_TransparentDialogDemo.c	Sample demonstrating alpha blending.
GUI_HelloWorld.c	Simple sample showing Hello World on the LCD.
GUI_HouseControlDemo.c	Sample showing a house control unit.
GUI_IP_ALPHA_TransparentDialogDemo.c	Sample demonstrating alpha blending. A VNC server is running in the background.
GUI_IP_emExplorer.c	Demonstration of using emWin as file explorer to show the content of a SD-card. A VNC server is running in the background.
GUI_IP_ReversiDemo.c	Sample showing a Reversi game. A VNC server is running in the background.
GUI_IP_WIDGET_GraphXYDemo.c	Sample showing usage of the Graph widget. A VNC server is running in the background.
GUI_MEMDEV_ImageFlow.c	Sample showing image flows by using memory devices.
GUI_MEMDEV_ZoomAndRotate.c	Sample zooming and rotating pictures by using memory devices.
GUI_IP_WIDGET_GraphYtDemo.c	Sample showing usage of the Graph widget. A VNC server is running in the background.
GUI_OS_Status.c	Sample showing the actual OS status and task status on the display.

Table 5.1: Sample applications

File(s)	Description
GUI_ReversiDemo	Sample showing a Reversi game.
GUI_USB_MSD_FS_Start.c	MSD sample using the GUI component to display status.
GUI_USB_MSD_FS_StartNew.c	MSD sample using the GUI component to display status using some images.
embOS/IP samples	
OS_IP_DNSClient.c	Sample showing domain name resolved to ip address.
OS_IP_FTPServer.c	FTP server example using the filesystem as storage space.
OS_IP_NonBlockingConnect.c	Sample showing how to connect to a server using non-blocking sockets.
OS_IP_Ping.c	Sample printing status of incoming and outgoing ICMP communication.
OS_IP_SendMail.c	Sample showing how to send an email from an embedded system.
OS_IP_Shell.c	Shell server example listening on port 23.
OS_IP_SimpleServer.c	Server example listening on port 23. Returns the actual OS time on keypress.
OS_IP_SpeedClient.c	Sample used for TCP/IP speed measurement.
OS_IP_Start.c	Sample using the TCP/IP stack.
OS_IP_Webserver.c	Webserver example using a readonly filesystem.
OS_IP_UDPDiscover.c	Sample showing how to discover a target without knowing its IP address by UDP broadcast.
OS_IP_UDPDiscoverZeroCopy.c	Sample showing how to discover a target without knowing its IP address by UDP broadcast using zero copy API.
embOS samples	
OS_Main_EVENT.c	Sample program for embOS using EVENT object.
OS_Main_OS_Q.c	Sample program for embOS using queues.
OS_Main_TaskEx.c	Sample program for embOS using extended task contexts.
OS_MeasureCST_HRTimer_embOSView.c	Performance test program for embOS using embOS-View for outputs.
OS_MeasureCST_HRTimer_Printf.c	Performance test program for embOS using the IAR EW console for outputs.
OS_MeasureCST_Scope.c	Performance test program for embOS designed for oscilloscope measurement.
OS_PerformanceTest.c	embOS sample used for measuring the RTOS.
OS_Start_2Tasks.c	Sample program for embOS using 2 tasks.
OS_Start_LEDBlink.c	Sample program for embOS using 2 tasks to toggle LEDs.
emUSB samples	
USB_BULK_Echo1.c	USB BULK sample showing a simply 1-byte echo server.
USB_BULK_EchoFast.c	USB BULK sample fast echo server to test the stack.
USB_BULK_ShowDeviceState.c	USB BULK sample showing the status of the current USB state.
USB_BULK_Test.c	USB BULK sample to test the stack.
USB_CDC_Echo.c	emUSB sample showing usage of virtual COM port.
USB_CDC_Start.c	emUSB sample showing usage of virtual COM port.

Table 5.1: Sample applications

File(s)	Description
USB_CompositeDevice_CDC_MSD.c	emUSB sample showing a composite device of CDC and MSD.
USB_HID_Echo1.c	emUSB sample showing how to utilize HID for data transfer without the need for drivers.
USB_HID_Mouse.c	emUSB sample showing a USB mouse moving.
USB_MSD_CDROM_Start.c	emUSB sample showing an emulated USB CDROM.
USB_MSD_FS_DisconnectOnWrite.c	emUSB sample showing `DisconnectOnWrite` hook.
USB_MSD_FS_Start.c	MSD sample using the filesystem driver as MSC storage driver.
USB_MSD_FS_WriteOnDisconnect.c	emUSB sample showing `WriteOnDisconnect` hook.
USB_MSD_Start_StorageByName.c	emUSB sample showing device access by name.
USB_MSD_StartStorageRAM.c	emUSB sample showing MSD using a RAMdisk.

Table 5.1: Sample applications

5.2 SeggerDemo

This sample demonstrates how easy all SEGGER software products can be combined in one application. This sample features the following components:

- embOS as RTOS coordinating all components
- embOS/IP providing TCP/IP support
- emFile providing a filesystem for server applications
- emUSB providing MSD access to the server filesystem
- emWin showing pictures and emWin widget samples.

This sample features the webserver example, ftp server example, USB MSD example running in the background while showing GUI samples on the LCD.

For further information please refer to *Evaluating the SeggerDemo* on page 20.

The libraries located in the folder "Start\Application\SEGGERDEMO" are needed by the SeggerDemo. They contain the SeggerDemo which is delivered as source in the folder "Start\Application\SEGGERDEMO\Src" and is provided as library to be compileable with the 32k limitation of the IAR EW ARM V5 KS version.

5.3 emFile samples

These samples use SEGGER emFile to provide a filesystem on the target.

5.3.1 FS_APIValidation.c

Sample program showing a simple generic API test.

Console output

```
High level formatting volume...Ok

Simple file test
Creating file...Ok
Write some data...Ok
Check file pos.......Ok
Read written data back...Ok
Write Burst test...Ok
Read Burst test...Ok
Creating a lot of file in root directory.......................................Ok
Test FS_FOpen modes
Mode 'w'...OK
Mode 'a'...OK
Mode 'r+'...OK
Mode 'w+'...OK
Mode 'a+'...OK

Directory API test
Checking FS_MkDir()...Ok
Checking FS_CreateDir()...Ok
Checking FS_FindFirstFile()/FS_FindNextFile...Ok
Checking FS_RmDir()....Ok

Extended API test
Preparing test...Ok
Checking FS_CopyFile()...Ok
Checking FS_Move()...Ok
Checking FS_Remove()...Ok
Checking FS_Rename()...Ok
Finished
```

5.3.2 FS_CheckDisk.c

This sample shows a simple checkdisk program.

This sample has no function when using a RAMDisk as filesystem.

5.3.3 FS_DirOperations.c

This sample creates 3 directories. In each directory 32 files are created. After creating the directories and files, the content of each directory is shown.

Console output

```
High level formatting:
...............................Ok
...............................Ok
...............................Ok
Contents of
 DIR00 (Dir) Attributes: ---- Size: 0
 Contents of \DIR00
  . (Dir) Attributes: ---- Size: 0
  .. (Dir) Attributes: ---- Size: 0
  FILE0000.TXT       Attributes: A--- Size: 19
  FILE0001.TXT       Attributes: A--- Size: 19
  FILE0002.TXT       Attributes: A--- Size: 19
  FILE0003.TXT       Attributes: A--- Size: 19
  FILE0004.TXT       Attributes: A--- Size: 19
  FILE0005.TXT       Attributes: A--- Size: 19
  FILE0006.TXT       Attributes: A--- Size: 19
  FILE0007.TXT       Attributes: A--- Size: 19
  FILE0008.TXT       Attributes: A--- Size: 19
  FILE0009.TXT       Attributes: A--- Size: 19
  FILE0010.TXT       Attributes: A--- Size: 19
  FILE0011.TXT       Attributes: A--- Size: 19
  FILE0012.TXT       Attributes: A--- Size: 19
  FILE0013.TXT       Attributes: A--- Size: 19
  FILE0014.TXT       Attributes: A--- Size: 19
  FILE0015.TXT       Attributes: A--- Size: 19
  FILE0016.TXT       Attributes: A--- Size: 19
  FILE0017.TXT       Attributes: A--- Size: 19
  FILE0018.TXT       Attributes: A--- Size: 19
  FILE0019.TXT       Attributes: A--- Size: 19
  FILE0020.TXT       Attributes: A--- Size: 19
  FILE0021.TXT       Attributes: A--- Size: 19
  FILE0022.TXT       Attributes: A--- Size: 19
  FILE0023.TXT       Attributes: A--- Size: 19
  FILE0024.TXT       Attributes: A--- Size: 19
  FILE0025.TXT       Attributes: A--- Size: 19
  FILE0026.TXT       Attributes: A--- Size: 19
  FILE0027.TXT       Attributes: A--- Size: 19
  FILE0028.TXT       Attributes: A--- Size: 19
  FILE0029.TXT       Attributes: A--- Size: 19
  FILE0030.TXT       Attributes: A--- Size: 19
  FILE0031.TXT       Attributes: A--- Size: 19

 DIR01 (Dir) Attributes: ---- Size: 0
 Contents of \DIR01
  . (Dir) Attributes: ---- Size: 0
  .. (Dir) Attributes: ---- Size: 0
  FILE0000.TXT       Attributes: A--- Size: 19
  FILE0001.TXT       Attributes: A--- Size: 19
  FILE0002.TXT       Attributes: A--- Size: 19
  FILE0003.TXT       Attributes: A--- Size: 19
  FILE0004.TXT       Attributes: A--- Size: 19
  FILE0005.TXT       Attributes: A--- Size: 19
  FILE0006.TXT       Attributes: A--- Size: 19
  FILE0007.TXT       Attributes: A--- Size: 19
  FILE0008.TXT       Attributes: A--- Size: 19
  FILE0009.TXT       Attributes: A--- Size: 19
  FILE0010.TXT       Attributes: A--- Size: 19
  FILE0011.TXT       Attributes: A--- Size: 19
  FILE0012.TXT       Attributes: A--- Size: 19
  FILE0013.TXT       Attributes: A--- Size: 19
 FILE0014.TXT       Attributes: A--- Size: 19
 FILE0015.TXT       Attributes: A--- Size: 19
 FILE0016.TXT       Attributes: A--- Size: 19
 FILE0017.TXT       Attributes: A--- Size: 19
 FILE0018.TXT       Attributes: A--- Size: 19
 FILE0019.TXT       Attributes: A--- Size: 19
 FILE0020.TXT       Attributes: A--- Size: 19
 FILE0021.TXT       Attributes: A--- Size: 19
 FILE0022.TXT       Attributes: A--- Size: 19
 FILE0023.TXT       Attributes: A--- Size: 19
 FILE0024.TXT       Attributes: A--- Size: 19
 FILE0025.TXT       Attributes: A--- Size: 19
 FILE0026.TXT       Attributes: A--- Size: 19
 FILE0027.TXT       Attributes: A--- Size: 19
 FILE0028.TXT       Attributes: A--- Size: 19
 FILE0029.TXT       Attributes: A--- Size: 19
 FILE0030.TXT       Attributes: A--- Size: 19
 FILE0031.TXT       Attributes: A--- Size: 19

DIR02 (Dir) Attributes: ---- Size: 0
Contents of \DIR02
 . (Dir) Attributes: ---- Size: 0
 .. (Dir) Attributes: ---- Size: 0
 FILE0000.TXT       Attributes: A--- Size: 19
 FILE0001.TXT       Attributes: A--- Size: 19
 FILE0002.TXT       Attributes: A--- Size: 19
 FILE0003.TXT       Attributes: A--- Size: 19
 FILE0004.TXT       Attributes: A--- Size: 19
 FILE0005.TXT       Attributes: A--- Size: 19
 FILE0006.TXT       Attributes: A--- Size: 19
 FILE0007.TXT       Attributes: A--- Size: 19
 FILE0008.TXT       Attributes: A--- Size: 19
 FILE0009.TXT       Attributes: A--- Size: 19
 FILE0010.TXT       Attributes: A--- Size: 19
 FILE0011.TXT       Attributes: A--- Size: 19
 FILE0012.TXT       Attributes: A--- Size: 19
 FILE0013.TXT       Attributes: A--- Size: 19
 FILE0014.TXT       Attributes: A--- Size: 19
 FILE0015.TXT       Attributes: A--- Size: 19
 FILE0016.TXT       Attributes: A--- Size: 19
 FILE0017.TXT       Attributes: A--- Size: 19
 FILE0018.TXT       Attributes: A--- Size: 19
 FILE0019.TXT       Attributes: A--- Size: 19
 FILE0020.TXT       Attributes: A--- Size: 19
 FILE0021.TXT       Attributes: A--- Size: 19
 FILE0022.TXT       Attributes: A--- Size: 19
 FILE0023.TXT       Attributes: A--- Size: 19
 FILE0024.TXT       Attributes: A--- Size: 19
 FILE0025.TXT       Attributes: A--- Size: 19
 FILE0026.TXT       Attributes: A--- Size: 19
 FILE0027.TXT       Attributes: A--- Size: 19
 FILE0028.TXT       Attributes: A--- Size: 19
 FILE0029.TXT       Attributes: A--- Size: 19
 FILE0030.TXT       Attributes: A--- Size: 19
 FILE0031.TXT       Attributes: A--- Size: 19
```

5.3.4 FS_Performance.c

Sample program for measure the performance.

Console output

```
High level formatting
W0 Writing chunks of 524288 Bytes (Clusters/file size preallocated):
.......OK
Second time writing chunks of 524288 Bytes (Dynamic allocation of clusters):
.......OK
R0 Reading chunks of 524288 Bytes (80% fill)
.......OK
Test:  0  (Min/Max/Av):  4/5/4;  (First  write  (Clusters/file  size  preallocated))
Speed:128000.00 kByte/s
Test:  1  (Min/Max/Av):  6/6/6;  (W1  Second  write  (Dynamic  allocation  of  clusters))
Speed: 85333.34 kByte/s
Test: 2 (Min/Max/Av): 8/8/8; (Read) Speed: 64000.00 kByte/s
Test 0 Speed: 128000.00 kByte/s
Test 1 Speed: 85333.34 kByte/s
Test 2 Speed: 64000.00 kByte/s
Finished...
```

5.3.5 FS_Start.c

Start application for file system.

Console output

```
High level formatting
Running sample on
  Free space: 4172800 bytes
  Write test data to file \File.txt
  Free space: 4171776 bytes
  Finished
```

5.3.6 FS_Start_ReadWriteHook.c

Console output

```
High level formatting
Running sample on
Open/create file
  Read  : StartSector: 0x00000000, NumSectors: 0x00000001, SectorType: DATA
  Read  : StartSector: 0x00000000, NumSectors: 0x00000001, SectorType: DATA
  Read  : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
  Write : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR  1st Write
(4 bytes)to file
  Read  : StartSector: 0x00000001, NumSectors: 0x00000001, SectorType: MAN
  Write : StartSector: 0x00000029, NumSectors: 0x00000001, SectorType: DATA
  Read  : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
  Write : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
  Write : StartSector: 0x00000001, NumSectors: 0x00000001, SectorType: MAN  2nd write
(511 bytes) to file.
  Read  : StartSector: 0x00000029, NumSectors: 0x00000001, SectorType: DATA
  Write : StartSector: 0x00000029, NumSectors: 0x00000001, SectorType: DATA
  Write : StartSector: 0x0000002a, NumSectors: 0x00000001, SectorType: DATA
  Read  : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
  Write : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR  Close
file
  Read  : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
  Write : StartSector: 0x00000019, NumSectors: 0x00000001, SectorType: DIR
Finished
```

5.3.7 FS_WritePerformanceTest.c

Sample program for measure the performance.

Console output

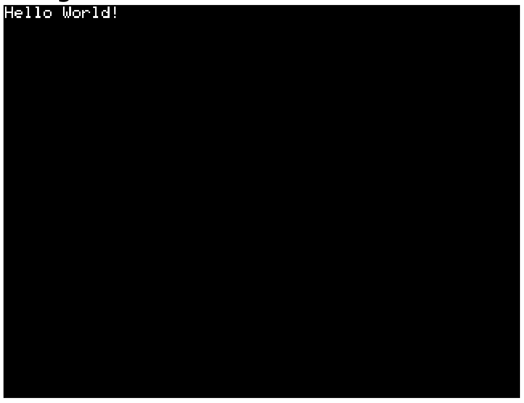
```
High level formatting
Writing chunks of 2048 x 1024 Bytes
(Min/Avg/Max/Total): 0, 0, 1, 55
Writing chunks of 2048 x 1024 Bytes
(Min/Avg/Max/Total): 0, 0, 1, 27
Finished...
```

5.4 emWin samples

These samples use SEGGER emWin to show graphical features on the LCD display.

5.4.1 GUI_HelloWorld.c

This sample shows a simple text output on the LCD. Your display should show something similar to the screenshot below.

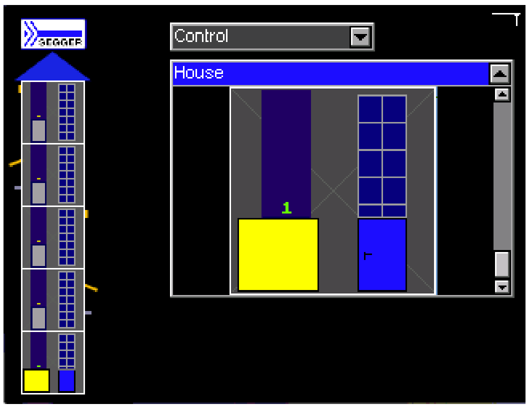


```
Hello World!
```

5.4.2 GUI_HouseControlDemo.c

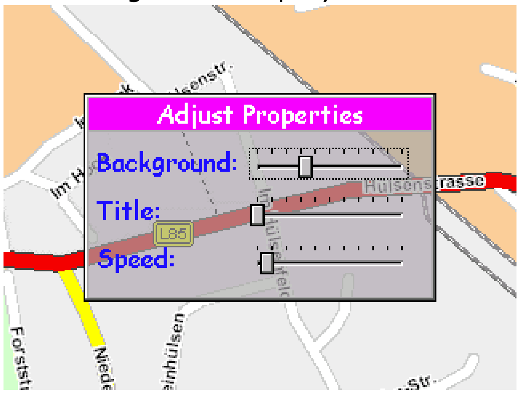
This sample shows how a possible house control unit may look like.

You can control several stations of the house. The actual states of the stations can be monitored in real time. Your display should show something similar to the screenshot below.



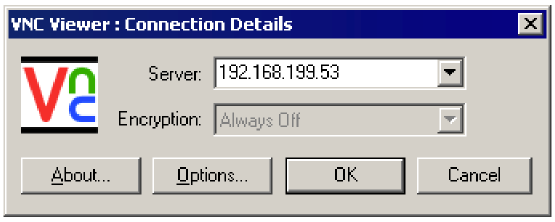
5.4.3 GUI_IP_ALPHA_TransparentDialogDemo.c

This sample shows a naviagtion like sample using alpha blending to show a transparent dialog. Your display should show something similar to the screenshot below.



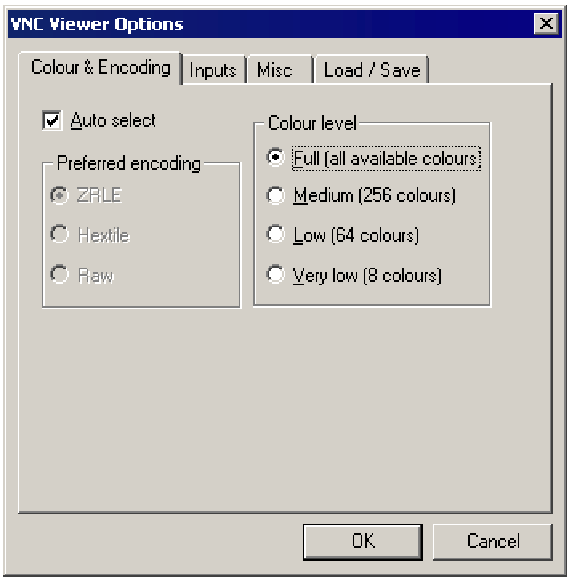
A VNC server is running in the background.

You can connect to the VNC server with the VNC client located under `"Start\Windows\GUI\vncviewer.exe"` . Start the `vncviewer.exe` and use settings similar to the screenshot below.


VNC Viewer : Connection Details

Server: 192.168.199.53

Encryption: Always Off


VNC Viewer Options

[✓] Auto select

Preferred encoding: [● ZRLE] Hextile Raw

Colour level: [● Full (all available colours)] Medium (256 colours) Low (64 colours) Very low (8 colours)

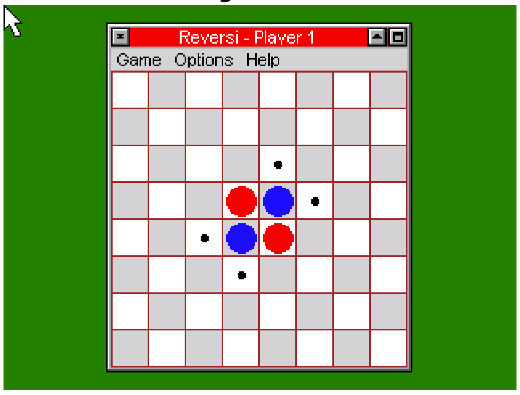
5.4.4 GUI_IP_emExplorer.c

This sample uses emWin widgets to show the content of a connected SD-card in an explorer like window.

A VNC server is running in the background. For further information please refer to *GUI_IP_ALPHA_TransparentDialogDemo.c* on page 36.

5.4.5 GUI_IP_ReversiDemo.c

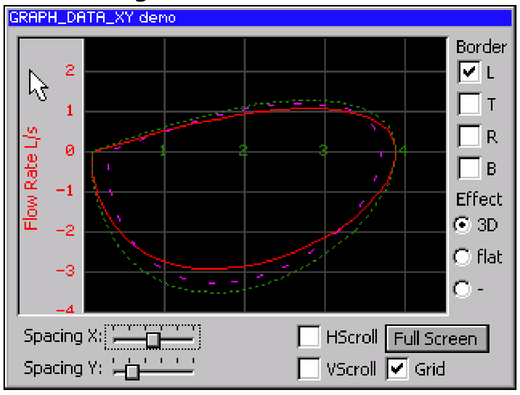
This sample shows a reversi game created with emWin widgets. Your display should show something similar to the screenshot below.



A VNC server is running in the background. For further information please refer to *GUI_IP_ALPHA_TransparentDialogDemo.c* on page 36.

5.4.6 GUI_IP_WIDGET_GraphXYDemo.c

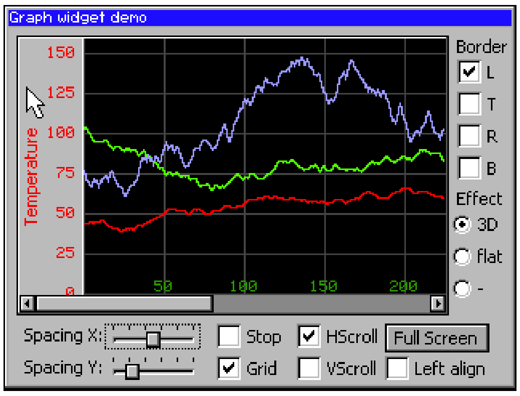
This sample shows a graph created with emWin widgets. Your display should show something similar to the screenshot below.



A VNC server is running in the background. For further information please refer to *GUI_IP_ALPHA_TransparentDialogDemo.c* on page 36.

5.4.7 GUI_IP_WIDGET_GraphYtDemo.c

This sample shows an oscilloscope like graph sample calculated from random values displayed on a timeline. Your display should show something similar to the screenshot below.



A VNC server is running in the background. For further information please refer to *GUI_IP_ALPHA_TransparentDialogDemo.c* on page 36.

5.4.8 GUI_OS_Status.c

This sample shows current embOS values.

The following details are displayed:

- Number of tasks
- OS time
- System stack
- Interrupt stack

Furthermore the following details are displayed for each task:

- Task ID
- Task priority
- Task name
- Number of activations (context switches)
- Used task stack
- Size of task stack

5.4.9 GUI_USB_MSD_FS_Start.c

MSD sample using the filesystem driver as MSC storage driver. The GUI part is showing the enumeration status on the display.

5.5 embOS/IP samples

These samples use SEGGER embOS/IP to demonstrate TCP/IP server applications.

5.5.1 OS_IP_DNSClient.c

This sample demonstrates use of the integrated DNS client.

5.5.2 OS_IP_FTPServer.c

This sample application runs an FTP server. The FTP server listens on port 21 for an incoming connection.

You can easily login to the FTP by using the "`anonymous`" account.

The storage space for the FTP server is provided by the emFile filesystem.

5.5.3 OS_IP_GUI_Webserver.c

This sample application runs a webserver. The webserver listens on port 80 for incoming connections.

The IP of the target is displayed on the LCD display.

For further information about the webserver example please refer to *OS_IP_Webserver.c* on page 40.

5.5.4 OS_IP_Shell.c

This sample demonstrates using the IP-shell to diagnose the IP stack. It opens TCP-port 23 (telnet) and waits for a connection.

The actual Shell server is part of the stack, which keep the application program nice and small.

To connect to the target, use the command line:

> telnet <target-ip>

Where <target-ip> represents the IP address of the target.

5.5.5 OS_IP_SimpleServer.c

This sample demonstrates setup of a simple server which simply sends back the target system tick for every character received. It opens TCP-port 23 (telnet) and waits for a connection.

To connect to the target, use the command line:

> telnet <target-ip>

Where <target-ip> represents the IP address of the target.

5.5.6 OS_IP_SpeedClient.c

This sample application is a client sample for measuring the network speed. For this sample you have to start the SpeedTestServer located under `Start\Windows\TCPIP\SpeedTestServer`.

You have to modify the IP address which can be found in the file `Start\Config\IP_Config_TOPAS910.c`.

Example

Change the lines

```
//IP_SetAddrMask(0xC0A80505, 0xFFFF0000);     // Assign IP addr. and subnet mask
  IP_DHCPC_Activate(0,"Target", NULL, NULL);
```

to

```
IP_SetAddrMask(0xC0A80505, 0xFFFF0000);     // Assign IP addr. and subnet mask
//IP_DHCPC_Activate(0,"Target", NULL, NULL);
```

for IP address 192.168.5.5 / 16.

5.5.7 OS_IP_Start.c

This sample demonstrates use of the IP stack without any server or client program.

To ping the target, use the command line:

> ping <target-ip>

Where <target-ip> represents the IP address of the target.

5.5.8 OS_IP_UDPDiscover.c

This sample demonstrates use of the IP stack to discover a device without knowledge of the IP address by using UDP broadcast.

The windows client needed to evaluate this sample is located under "`Start\Windows\TCPIP\UDPDiscover\UDPDiscover.exe`".

5.5.9 OS_IP_UDPDiscoverZeroCopy.c

This sample demonstrates use of the IP stack to discover a device without knowledge of the IP address by using UDP broadcast. This sample uses zero copy API to directly process the packet instead of retrieving it through the IP stack. This increases speed on time critical transfers.

The windows client needed to evaluate this sample is located under "`Start\Windows\TCPIP\UDPDiscover\UDPDiscover.exe`".

5.5.10 OS_IP_Webserver.c

This sample application runs a webserver. The webserver listens on port 80 for incoming connections.

The storage space for the webserver is provided by the emFile filesystem. The webserver sample uses a readonly filesystem using generated websites in C-code.

For further information about the webserver example please refer to *OS_IP_Webserver.c* on page 40.

5.6 embOS samples

These samples use SEGGER embOS to demonstrate RTOS features.

5.6.1 OS_Main_EVENT.c

This sample program for embOS demonstrates usage of EVENT objects.

5.6.2 OS_Main_OS_Q.c

This sample program for embOS demonstrates usage of QUEUE objects.

5.6.3 OS_Main_TaskEx.c

This sample program for embOS demonstrates usage extended task contexts.

5.6.4 OS_MeasureCST_HRTimer_embOSView.c

This benchmark measures the OS context switch time and displays the result in the terminal window of embOSView.

It is completly generic and runs on every target that is configured for embOSView.

5.6.5 OS_MeasureCST_HRTimer_Printf.c

This benchmark measures the context switch time and displays the result on CSpy's terminal I/O window. It runs with every IAR workbench that supports terminal I/O with printf.

5.6.6 OS_MeasureCST_Scope.c

This benchmark uses the LED.c module to set and clear a port pin. This allows measuring the context switch time with an oscilloscope.

The context switch time is

Time = (d - c) - (b - a)

```
-----   --                ---------------
     | |  |              |
      -   -----------------
     ^ ^  ^                ^
     a b  c               d
```

The time between c and d ist the context switch time, but note that the real context switch time is shorter, because the signal also contains the overhead of switching the LED on and off. The time of this overhead is also displayed on the oscilloscope as a small peak between a and b.

5.6.7 OS_PerformanceTest.c

This sample tests how many primes can be calculated in one second to determine the speed of the embOS RTOS.

5.6.8 OS_Start_2Tasks.c

These samples show some basic functionality of embOS by using two tasks.

5.6.9 OS_Start_LEDBlink.c

These samples show some basic functionality of embOS by toggling LEDs in two tasks.

5.7 emUSB samples

These samples use SEGGER emUSB to demonstrate USB communication.

5.7.1 USB_BULK_Echo1.c

This sample program for emUSB demonstrates a simple 1-byte USB BULK echo server.

The windows client needed to evaluate this sample is located under "`Start\Windows\USB\Bulk\SampleApp\Exe\Echo1.exe`".

5.7.2 USB_BULK_EchoFast.c

This sample program for emUSB demonstrates a fast echo server to test the stack.

The windows client needed to evaluate this sample is located under "`Start\Windows\USB\Bulk\SampleApp\Exe\EchoFast.exe`".

5.7.3 USB_BULK_ShowDeviceState.c

This sample program for emUSB sample shows the status of the current USB state in the debugger terminal I/O.

5.7.4 USB_BULK_Test.c

This sample program for emUSB demonstrates a modified echo server to test the stack.

The windows client needed to evaluate this sample is located under "`Start\Windows\USB\Bulk\SampleApp\Exe\Test.exe`".

5.7.5 USB_CDC_Start.c

This sample program for emUSB demonstrates a simple USB2COM echo server.

You can connect to the echo server by connecting the target with your PC via USB and using a simple terminal program like Hyper Terminal to connect to the target via the virtual COM port.

5.7.6 USB_HID_Echo1.c

This sample program for emUSB demonstrates a simple 1-byte USB BULK echo server using HID drivers to eliminate the need for extra drivers.

The windows client needed to evaluate this sample is located under "`Start\Windows\USB\HID\SampleApp\Exe\`HIDEcho1.exe".

5.7.7 USB_HID_Mouse.c

This sample program for emUSB demonstrates usage of the HID component of the USB stack as mouse. Makes the mouse jump left & right.

5.7.8 USB_MSD_FS_Start.c

This sample program for emUSB demonstrates a sample startup for MSD, using the file system driver as MSC storage driver.

Chapter 6

Literature and references

This chapter lists documents, which we think may be useful to gain deeper understanding of technical details.

Reference	Title	Comments
[UM08001]	SEGGER J-Link / J-Trace User's Guide.	This document gives information about using the SEGGER J-Link / J-Trace ARM. It is publicly available from SEGGER (*www.segger.com*).
[UM01002]	embOS for ARM and IAR Embedded Workbench	This document gives information about using embOS for IAR EWARM. It is publicly available from SEGGER (*www.segger.com*).
[UM01001]	User's & reference manual for embOS	This document gives information about using the generic parts of embOS. It is publicly available from SEGGER (*www.segger.com*).
[UM07001]	embOS/IP User Guide	This document gives information about using the SEGGER IP stack. It is publicly available from SEGGER (*www.segger.com*).
[UM09001]	User's and reference manual for emUSB	This document gives information about using the SEGGER USB stack. It is publicly available from SEGGER (*www.segger.com*).
[UM02001]	emFile User's Guide	This document gives information about using the SEGGER embedded filesystem. It is publicly available from SEGGER (*www.segger.com*).
[UM03001]	User's and reference manual for emWin	This document gives information about using the SEGGER GUI software. It is publicly available from SEGGER (*www.segger.com*).

Table 6.1: Literature and References