emNet

CPU independent TCP/IP stack
for embedded applications

Migration Guide

Document: AN07001
Software Version: 3.42.0
Revision: 0
Date: November 23, 2021

A product of SEGGER Microcontroller GmbH

www.segger.com

Disclaimer

Specifications written in this document are believed to be accurate, but are not guaranteed to be entirely free of error. The information in this manual is subject to change for functional or performance improvements without notice. Please make sure your manual is the latest edition. While the information herein is assumed to be accurate, SEGGER Microcontroller GmbH (SEGGER) assumes no responsibility for any errors or omissions. SEGGER makes and you receive no warranties or conditions, express, implied, statutory or in any communication with you. SEGGER specifically disclaims any implied warranty of merchantability or fitness for a particular purpose.

Copyright notice

Trademarks

Names mentioned in this manual may be trademarks of their respective companies.

Brand and product names are trademarks or registered trademarks of their respective holders.

Contact address

SEGGER Microcontroller GmbH

Ecolab-Allee 5
D-40789 Monheim am Rhein

Germany

Tel.	+49 2173-99312-0
Fax.	+49 2173-99312-28
E-mail:	ticket_emnet@segger.com*
Internet:	www.segger.com

*By sending us an email your (personal) data will automatically be processed. For further information please refer to our privacy policy which is available at https://www.segger.com/legal/privacy-policy/.

Manual versions

This manual describes the current software version. If you find an error in the manual or a problem in the software, please inform us and we will try to assist you as soon as possible. Contact us for further information on topics or functions that are not yet documented.

Print date: November 23, 2021

Revision	Date	By	Description
14	211123	OO	Use of new version number scheme. Added migration information for V3.42.0: • *DTASK API renaming* on page 23 • *`BSP_UART_Init()` is no longer called with interface index by PPP code* on page 23
13	200402	OO	Product renamed from embOS/IP to emNet.
12	170622	OO	Added migration information for V3.20: • *SMTP client (add-on) multipart callback return value changed* on page 22
11	170323	OO	Added migration information for V3.16: • *FTP server (add-on) file names changed* on page 18 • *`IP_BSP_API` has been renamed to `BSP_IP_API`* on page 19 • *All drivers have been changed to `BSP_IP_API`* on page 20 • *New or changed generic drivers* on page 21
10	160912	OO	Added migration information for V3.10: • K60/K70 driver renamed to Kinetis.
9	160630	OO	Added migration information for V3.08: • `IP_BSP_API` changes.
8	160511	OO	Added migration information for V3.06 .
7	160419	OO	Minor corrections on text.
6	151124	OO	Added migration information for V3.02 . Added missing information regarding moved IP/OS layer for V3.00a .
5	151007	OO	Added migration information for V3.00a .
4	150813	OO	Added migration information for V3.00 .
3	141002	OO	Added migration information for V2.20b .
2	140430	OO	Added migration information for V2.20 .
1	130312	OO	Added migration information for V2.12 .
0	120511	OO	First version.

Table of contents

Chapter 1

Migrating from a previous version

This chapter provides a guideline on how to migrate to the latest emNet software version from a previously used older version. Although trying to provide a minimum of changes that need to be applied for new versions this can not be guaranteed in any case. This chapter shall provide step by step instructions about the necessary changes that need to be applied in case there was a change in emNet not compatible with a previous version.

1.1 Migrating from embOS/IP V1

One of the new features of embOS/IP V2 is that the user now can save ROM and RAM by not including modules that are not needed for the users functionality.

For this to achieve embOS/IP V2 now allows to add only those protocols that you plan to use in your project. For this the customer will have to apply minimal changes to their previously used configuration. The steps necessary are explained in more detail in the next sections.

1.1.1 Changes to the configuration file

Adding one or more protocols to the IP stack is fairly easy. At the moment the following protocols can be added:

- TCP
- UDP
- ICMP

The following table shows the API that is needed to be added to your config file to add protocols.

Function	Description
`IP_ICMP_Add()`	Adds ICMP to the stack.
`IP_TCP_Add()`	Adds TCP to the stack.
`IP_UDP_Add()`	Adds UDP to the stack.

To add one or more of these protocols to the IP stack follow these steps:

1. Open your configuration file which is typically called `IP_Config_*.c` in a text editor or the editor of your IDE.
2. Add one or more protocols to the routine `IP_X_Config()`

```
IP_TCP_Add();
IP_UDP_Add();
IP_ICMP_Add();
```


1.2 Migrating to embOS/IP V2.12

The changes in this version are kept to a minimum and may not apply to you depending on what MCU you are using. The following changes are necessary to know when:

- You are using an assembler routine for checksum calculation.
- You are using VFile hooks as they are used with the UPnP add-on.

1.2.1 Changes regarding checksum calculation

In previous releases the checksum calculation as it was used for e.g. TCP did not follow recommendations in RFC 1624. RFC 1624 describes a case where a network participant may send a TCP checksum of 0xFFFF due to an older RFC and how to handle it as 0xFFFF is not treated a valid value for a TCP checksum.

In order to support RFC 1624 the assembler routines for checksum calculation now get a third parameter Sum which is the start checksum.

The following changes need to be applied to your IP_Conf.h for each compiler. For demonstrational purposes the ARM branch is shown:

Old

```
U32 ARM_IP_cksum(void *ptr, unsigned NumHWords);
#define IP_CKSUM(p, NumItems)  ARM_IP_cksum((p), (NumItems))
```

New

```
U32 ARM_IP_cksum(void *ptr, unsigned NumHWords, U32 Sum);
#define IP_CKSUM(p, NumItems, Sum)  ARM_IP_cksum((p), (NumItems), (Sum))
```

1.2.2 Changes regarding VFile hooks

VFile hooks have been introduced to support an easy way of serving a small dynamically built config file such as an XML file for UPnP that does not need further parsing. As UPnP does not seem to be able to handle chunked encoding the XML file needs to be served in RAW encoding for UPnP. In previous releases all VFile hooks were serving RAW encoded content. However this is not always desired as with larger files being served by VFile hooks it might be desired to have chunked encoding.

For this to achieve VFile hooks will now as with regular web site requests by a browser try to serve chunked encoding when possible. To support UPnP which needs to use RAW encoding the VFile hook function has been changed to get another parameter to force RAW encoding.

The following changes need to be applied where VFile hooks have been used:

Using a VFile hook for a web page

```
IP_WEBS_AddVFileHook( &_UPnP_VFileHook,
                      &_UPnP_VFileAPI,
                      0);
```

Using a VFile hook for UPnP XML file

```
IP_WEBS_AddVFileHook( &_UPnP_VFileHook,
                      &_UPnP_VFileAPI,
                      HTTP_ENCODING_RAW);
```


1.3 Migrating to embOS/IP V2.20

The changes in this version are kept to a minimum and may not apply to you depending on what configuration you are using. The following changes are necessary to know when:

- You are using the PPP add-on.
- You are using the FTP Server add-on.

1.3.1 Changes regarding PPP and BSP_UART.c module

To allow the `BSP_UART.c` module to be used with several other middleware the API of the `BSP_UART_*` routines has been extended. Basically the old routines can be easily extended by wrapping the old functions with the new API.

A sample of the new API is shipped in the BSP folder that comes with the PPP / PPPoE add-on shipment.

1.3.2 Changes regarding the file system application layer

To allow moving files and folders when accessing the FTP server add-on the function `IP_FS_API` structure has been extended by two function pointers.

Function pointer	Description
`pfIsFolder`	Checks if the given path is a folder.
`pfMove`	Moves a file or folder.

The new functionality is optional but not implementing these functionality means that the FTP server will not be able to successfully check if the given path is really a folder (which might be necessary for batch operations) or moving files and folders.

1.4 Migrating to embOS/IP V2.20b

The following changes need to be addressed when migrating from a previous version.

1.4.1 New files added in V2.20b

The following files have been added to allow better handling of CPUs with cache.

- `IP_CACHE.c`

1.4.2 Files changed in V2.20b

The following files that reside outside of the IP folder have been changed and need to be updated as well.

- SEGGER.h

1.4.3 Other changes in V2.20b

If multicast support is enabled with `IP_SUPPORT_MULTICAST = 1` (default) the file `IP_IGMP.c` now needs to be included into the project as well.

1.5 Migrating to embOS/IP V3.00

The following changes need to be addressed when migrating from a previous version.

1.5.1 New files added in V3.00

The following files have been added:

- `IP_ETH.c`

1.5.2 Files changed in V3.00

The following files that reside outside of the IP folder have been changed and need to be updated as well.

- SEGGER.h
- Global.h

1.6 Migrating to embOS/IP V3.00a

The following changes need to be addressed when migrating from a previous version.

1.6.1 Util folder moved

The files from the `Util` folder have been moved into the `SEGGER` folder to match their prefix.

1.6.2 Assembler files moved

Assembler files have been moved up one folder into `ASM` folders to make it clear that these are special optimized versions of the default C-routines. Overwriting the default C-routines is typically done by a macro that can be found in the file `\Config\IP_Conf.h`. This applies to the following folders:

- IP\ASM
- SEGGER\ASM

1.6.3 OS abstraction layer moved

The OS abstraction layer and espacially the file `IP_OS_embOS.c` can now be found in the folder `\Sample\IP\OS`. Exactly one IP/OS layer has to be used at the same time.

1.7 Migrating to embOS/IP V3.02

The following changes need to be addressed when migrating from a previous version.

1.7.1 Changes regarding PHY drivers

With embOS/IP V3.02 PHY driver support has been added. The previously fixed generic PHY driver is now an exchangeable PHY driver that has been moved into the new file `IP_PHY_GENERIC.c` and has to be added to your project.

1.7.2 RX BSP_ETH_* changes

The Renesas RX Ethernet driver now supports RX64M and RX71M in general as well as dual MAC support. For this change the call to `BSP_ETH_InstallISR()` is exchanged with `BSP_ETH_InstallISR_Ex()`.

The conversion is shown in the following example:

Before

```
/*********************************************************************
*
*       BSP_ETH_InstallISR()
*/
void BSP_ETH_InstallISR(void (*pfISR)(void)){
  _pfEthISRHandler = pfISR;
}
```

After

```
/*********************************************************************
*
*       BSP_ETH_InstallISR_Ex()
*/
void BSP_ETH_InstallISR_Ex(int ISRIndex, void (*pfISR)(void), int Prio){
  (void)ISRIndex;
  (void)Prio;
  _pfEthISRHandler = pfISR;
}
```


1.8 Migrating to embOS/IP V3.06

The following changes need to be addressed when migrating from a previous version.

1.8.1 Changes regarding file/folder structure

Some files that provide functionality not directly related to the IP stack itself or not necessary for running the stack have been moved into a "Shared" folder. Examples for these files are connection layers between IP and FS as well as sample code that is shared between more than one sample like the "`Webserver_DynContent.c`" that contains most of the sample code used by various web server samples that basically only differ in the setup phase.

1.9 Migrating to embOS/IP V3.08

The following changes need to be addressed when migrating from a previous version.

1.9.1 IP_BSP_API has been added

In previous versions hardware specific setup that the driver can not handle in a generic way was located in BSP_ETH_* functions in the file BSP.c that was shipped for different eval boards in the "\BSP" folder of the embOS/IP shipment.

As this file might be shipped in different versions with different middleware components the BSP.c that contained routines for several middlewares has been split up into middleware specific BSP_<MW>.c files like BSP_IP.c .

Drivers will be switched from the old BSP.c implementation to the new IP_BSP_API structure over time. Such a change will be announced in the release notes and will not be mentioned in the migration guide again.

Drivers that have been changed to the IP_BSP_API will be shipped with matching sample BSP.c and BSP_IP.c in the "\BSP" folder for your reference and are easy to adopt to the new API.

The main difference is that for each driver added the following line now needs to be added to the IP_X_Config():

```
IP_BSP_SetAPI(IFaceId, &BSP_IP_Api);  // Set BSP callbacks for hardware access.
```


1.10 Migrating to embOS/IP V3.10

The following changes need to be addressed when migrating from a previous version.

1.10.1 Freescale Kinetis adopted to IP_BSP_API

The Freescale/NXP K60/K70 driver has been renamed to Freescale/NXP Kinetis. Along with this change the BSP abstraction layer of the driver has been changed to the `IP_BSP_API`. This change includes moving the port pin setup previously executed within the driver into the `BSP_IP.c`.

The BSP samples shipped with the driver already contain the port pin setup that was previously part of the driver itself. Please make sure that this is included in your `BSP_IP.c` as well or change your project to use the shipped file at all (recommended).

1.11 Migrating to embOS/IP V3.16

The following changes need to be addressed when migrating from a previous version:

1) *FTP server (add-on) file names changed* on page 18
2) *IP_BSP_API has been renamed to BSP_IP_API* on page 19
3) *All drivers have been changed to BSP_IP_API* on page 20
4) *New or changed generic drivers* on page 21

1.11.1 FTP server (add-on) file names changed

FTP server files have been renamed from `IP_FTPS*` or `IP_FTPServer*` to `IP_FTP_SERVER` for a cleaner naming scheme.

> **Note**
>
> Please make sure that you exchange/remove the old files.

1.11.2 IP_BSP_API has been renamed to BSP_IP_API

To allow an even better abstraction and to eliminate more dependencies, elements of the IP_BSP_API (*Migrating to embOS/IP V3.08* on page 16) have been changed to BSP_IP_API.

With all elements of the BSP abstraction layer now being independent from embOS/IP (no need to include IP.h), the BSP files are typically able to compile with or without the embOS/IP sources. This makes it even easier to include/exclude the IP stack for more flexibility in projects.

The concept remains the same but now IP.h includes BSP_IP.h instead the other way around.

1.11.3 All drivers have been changed to BSP_IP_API

All embOS/IP drivers beginning with V3.16 are using the `BSP_IP_API` for low level abstraction of BSP specific operations.

1.11.4 New or changed generic drivers

The following drivers have been changed.

1.11.4.1 Renesas EtherC driver

- Replaces Renesas RX driver.
- Support for Renesas Synergy S5 & S7 added.

1.11.4.2 NXP LPC driver

- Replaces NXP LPC17xx/40xx driver.
- Replaces NXP LPC23xx/24xx driver.
- Replaces NXP LPC32xx driver.

> **Note**
>
> Newer NXP LPC devices like LPC18xx/43xx are using a Synopsis Ethernet IP. These devices are supported by the generic Synopsys driver.

1.11.4.3 Synopsis driver

- Support for NXP LPC18xx/43xx added.

1.12 Migrating to embOS/IP V3.20

The following changes need to be addressed when migrating from a previous version:

1) *SMTP client (add-on) multipart callback return value changed* on page 22

1.12.1 SMTP client (add-on) multipart callback return value changed

The callback (`_cbSendAttachment()` in our samples) for sending the multipart content was previously a void function, not using a return value. This has been changed to the return type int to allow an abort to happen from within the callback. The reason for an abort might be a problem when reading a file from a file system or similar.

Your callback should be updated according to the new return type and values as shown by the header below:

```
/*********************************************************************
*
*       _cbSendAttachment()
*
*  Function description
*    Sends the content of an attachment.
*
*  Parameters
*    pContext: SMTPc context.
*    pItem   : Item to send.
*    pAPI    : API used to send the content.
*
*  Return value
*    == 0: O.K., attachment sent.
*    <  0: Error.
*/
```


1.13 Migrating to emNet V3.42.0

The following changes need to be addressed when migrating from a previous version:

1) *DTASK API renaming* on page 23
2) *BSP_UART_Init() is no longer called with interface index by PPP code* on page 23

1.13.1 DTASK API renaming

In previous versions WiFi module support required an additional task that has been named "WiFi ISR Task". This task is now used for other drivers as well (typically external MAC/PHY controllers for example using an SPI interface). Therefore the name was not suitable anymore and the task including its API have been renamed. The new name is "Driver Task" with the abbreviation "DTASK" .

The specific API has been renamed with macros with the old names in place for backwards compatibility. Please also update your IP.h in case you use the task size defines for the WiFi ISR task.

1.13.2 BSP_UART_Init() is no longer called with interface index by PPP code

For historical reasons the PPP code calls `BSP_UART_Init()` with a zero baudrate and other parameters as zero as well. Older implementations of the `BSP_UART_*` API fall back to using defaults when being called with the parameters at zero. Newer implementaiton of the `BSP_UART_*` API does not always load these defaults, so it might be necessary to call them with real values for unit, baudrate and other parameters. For this the new API `IP_MODEM_SetUartConfig()` can be used to set these parameters before `BSP_UART_Init()` gets called.

In addition to this, previously the unit parameter was called with the interface index of the PPP interface. The unit parameter now defaults to the parameter given using `IP_MODEM_SetUartConfig()` whiczh by default is zero.

