HCC Network Driver for Atmel EMAC Devices User Guide

Version 1.00

For use with Network Driver for Atmel EMAC module versions 1.01 and above

Exported on 03/19/2019

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

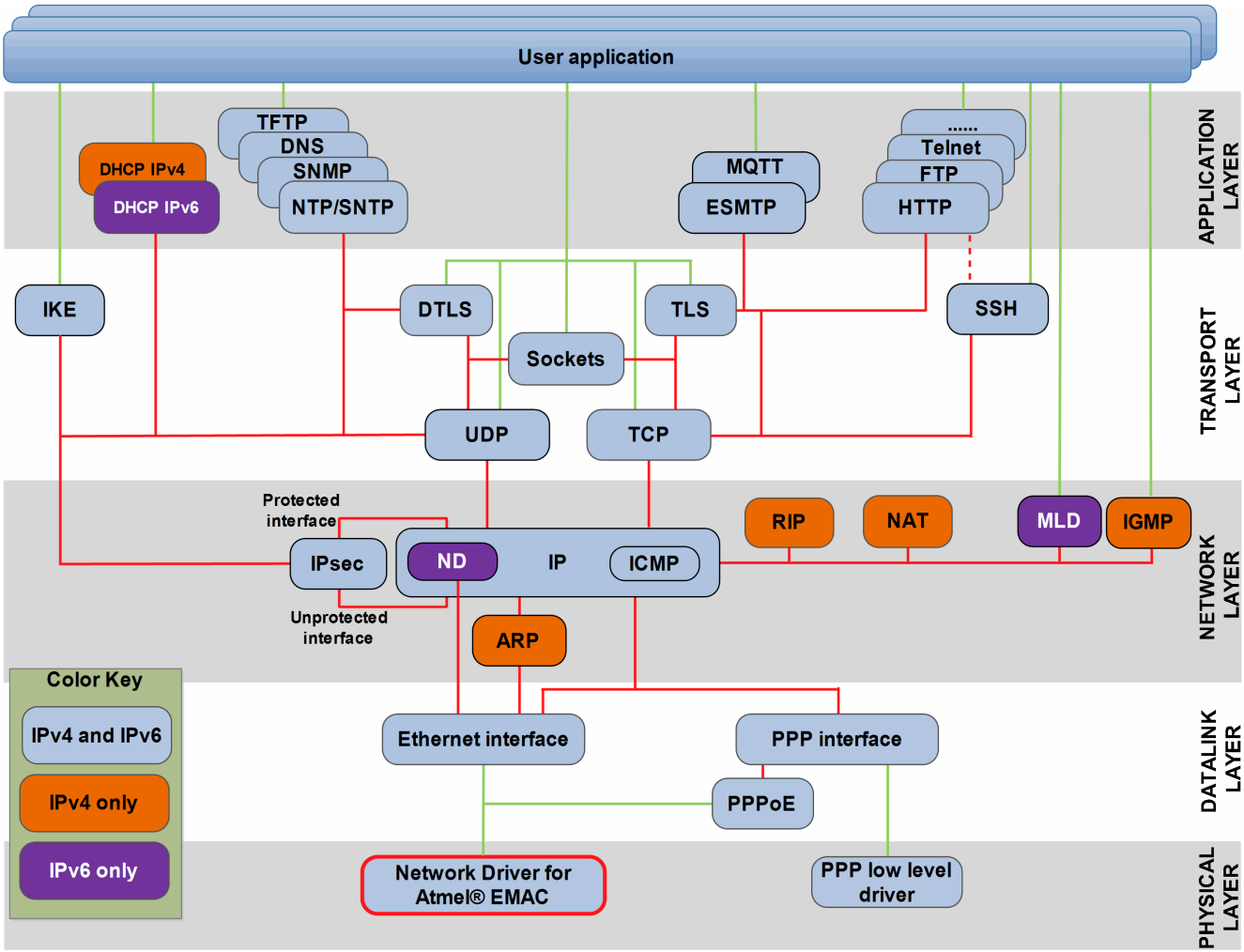
- Introduction – describes the main elements of the module. This section includes a diagram showing the position of the driver interface within HCC's TCP/IP stack.
- Feature Check – summarizes the main features of the module as bullet points.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

1.1 Introduction

This guide is for those who want to implement a network driver for the Ethernet Media Access Controller (EMAC) core in Atmel® SMART microcontroller families. These devices are produced by Microchip Technology Inc. The driver was tested with Microchip Technology's SAMA5D3 evaluation board.

The purpose of this driver is to provide an Ethernet physical port interface at the device end of a USB connection, so that the host system sees that remote physical port as a local Ethernet port.

The driver's location within HCC's TCP/IP stack is shown below. (In this diagram green lines show interfaces available to users of the TCP/IP stack, red lines show internal TCP/IP interfaces.)



Note: Although every attempt has been made to simplify the system's use, you need a good understanding of the requirements of the systems you are designing in order to obtain the maximum practical benefits. HCC Embedded offers hardware and firmware development consultancy to help you implement your system.

1.2 Feature Check

The main features of the network driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Fully compatible with the HCC Network Driver Interface specification.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to the HCC Coding Standard.
- Supports Atmel® SMART devices that have an EMAC core.
- HCC provides fully tested reference drivers for this module.

1.3 Packages and Documents

Packages

The table below lists the packages that you need in order to use this module.

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
nw_drv_eth_emac	The Network Driver for Atmel® EMAC package.
nw_drv_base	The network driver base package. This is the base system on which the Atmel® EMAC driver is built.
psp_template_base	The base Platform Support Package (PSP).
oal_base	The OS Abstraction Layer (OAL) package.
mutil_timer	The MISRA-compliant timer utility.

Documents

For an overview of HCC's TCP/IP stack software, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC Network Driver User Guide

This document describes the network driver base system.

HCC Network Driver for Atmel EMAC User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To download this manual as a PDF, see Network Driver PDFs.
- For the history of changes made to the package code itself, see History: nw_drv_eth_emac.

The current version of this manual is 1.00.

Manual version	Date	Software version	Reason for change
1.00	2019-03-19	1.01	First online version.

2 Source File List

This section lists and describes all the source code files included in the system. These files follow HCC Embedded's standard source tree system, described in the HCC Source Tree Guide. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

Note: Do not modify any files except the configuration file and PSP files.

2.1 API Header File

The file **src/api/api_ethdriver_emac.h** is the only file that should be included by any application using this module. It defines the emac_eth_drv_init() function.

2.2 Configuration File

The file **config_ethdriver_emac.h** contains all the configurable parameters of the system. Configure these as required. For details of these options, see Configuration Options.

2.3 System Files

The following files are in the directory **src/driver/network/ethernet/emac**. **These files should only be modified by HCC**.

File	Description
eth_emac.c	Ethernet driver source code.
eth_phy_ksz8031.c	KSZ8031 PHY source code.
eth_phy_ksz8031_reg.c	KSZ8031 PHY registers header file.

2.4 Version Files

These files in the directory **src/version** contain the version numbers of components of this module. The version number is checked by all modules that use a component to ensure system consistency over upgrades.

File	Description
ver_ethdriver_emac.h	Module version.
ver_eth_phy_ksz8031.h	Version of KSZ8031 PHY.

2.5 Platform Support Package (PSP) Files

These files provide functions the core code needs to call, depending on the hardware. The following generalized PSP implementation files are in the directory **psp_atsama5d3_xplained** (for the SAMA5D3-EDS evaluation board) in **src/psp**. These provide templates for you to produce your own PSP.

Note:

- You may need to modify these PSP implementations for your specific microcontroller and development board; see PSP Porting for details.
- In the package these files are offset to avoid overwriting an existing implementation. Copy them to the root **hcc** directory for use.

File	Description
include/psp_eth_mii.h	Media Independent Interface (MII) header file.
include/psp_eth_phy.h	PHY header file.
target/eth/psp_eth_emac.c and **.h**	EMAC functions source code and header file.
target/eth/psp_isr.c and **.h**	ISR functions source code and header file.
config/config_psp_eth_emac.h	PSP configuration file.

The PSP also has the following version files:

File	Description
ver_eth_phy_ksz8031.h	Version of KSZ8031 PHY.
ver_psp.h	Version of PSP.
ver_psp_eth_emac.h	Version of EMAC PSP.
ver_psp_emac_mii.h	Version of EMAC MII.
ver_psp_emac_phy.h	Version of EMAC PHY.
ver_psp_isr.h	Version of ISR .

3 Configuration Options

Set the system configuration options in the file **src/config/config_ethdriver_emac.h**, as described below. This section lists the available configuration options and their default values.

EMAC_USE_RMII

Keep the default of 1 if Reduced Media-Independent Interface (RMII) mode is used, otherwise set it to 0. The RMII standard is designed to reduce the number of signals required to connect a PHY to a MAC.

EMAC_ETHERNET_BUF_SIZE

The Ethernet buffer size. This must be larger than MAX_RX_DESC * 1536 + MAX_TX_DESC * 1536. The default is 16384.

ETH_ISR_ID

The ISR ID. The default is ID_EMAC0.

ETH_ISR_PRIO

The ISR priority. The default is 0. On Atmel® ATSAM9A5D33 five priority bits are implemented so this is the lowest priority.

ETH_PHY_DEV_ADDR

The address of the external PHY. The default is 0x01.

ETHDRV_LINK_STA_POLL_INTERVAL

The link status poll interval in milliseconds. The default is 100.

MAX_RX_DESC

The maximum number of RX descriptors. The hardware does not use these descriptors for actual reception; data is copied into buffers that these descriptors point to. The default value is 4.

MAX_TX_DESC

The maximum number of TX descriptors. The default value is 4.

MAC_ADDRESS

The MAC address of the driver.

The default value is `{ 0x00u, 0xA0u, 0x91u, 0xFBu, 0x93u, 0xCDu }`

4 Application Programming Interface

This section describes the single Application Programming Interface (API) function and the error codes it may return.

4.1 emac_eth_drv_init

Use this function to initialize the network driver.

Format

```
t_nwdriver_ret emac_eth_drv_init (
    uint32_t         param,
    t_nwdriver * *   const p_ethdriver )
```

Arguments

Parameter	Description	Type
param	The driver parameter.	uint32_t
p_ethdriver	Where to write the pointer to the driver.	t_nwdriver * *

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

4.2 Error Codes

This table lists all the error codes that may be generated by the API calls:

Error code	Value	Meaning
NWDRIVER_SUCCESS	0	Execution successful.
NWDRIVER_ERROR	1	Operation failed.

5 Integration

This section describes all aspects of the network driver that require integration with your target project. This includes porting and configuration of external resources.

5.1 OS Abstraction Layer

The network driver uses the OS Abstraction Layer (OAL) that allows it to run seamlessly with a wide variety of RTOSes, or without an RTOS.

The network driver uses the following OAL components:

OAL Resource	Number Required
Tasks	0
Mutexes	1
Events	0
ISRs	1

5.2 Utilities

The code creates and uses a single timer in the **hcc_timer** module.

The **hcc_timer** module is included in your system when you install the base network driver module.

5.3 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer. For full details of its functions and macros, see the *HCC Base Platform Support Package User Guide*.

The driver makes use of the following standard PSP functions:

Function	Package	Element	Description
psp_membar()	psp_base	psp_membar	Executes a memory barrier.
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.

The driver makes use of the following PSP functions which must be provided by the Platform Support Package. These are designed for you to port them easily to work with your hardware solution. The package includes samples in the file **include/eth_emac.c**.

Function	Description
psp_emac_eth_init()	Initializes the hardware for ETH usage.
psp_emac_eth_start()	Starts the Ethernet driver.
psp_emac_eth_stop()	Stops the Ethernet driver.
psp_emac_eth_delete()	Deletes the Ethernet driver, releasing the associated resources.
psp_emac_get_bufs()	Gets the addresses of the Ethernet buffers.
psp_emac_set_speed()	Sets the speed to 10 Mbps or 100 Mbps.

These functions are described below in the sections after the PSP configuration files.

PSP Configuration Files

The PSP has two configuration files, described below.

config_ethdriver_emac.h

This PSP file has the same configuration options as the main configuration file but with appropriate values for the device, as described below. The only defaults that differ from the main file are those for MAX_RX_DESC and MAX_TX_DESC.

EMAC_USE_RMII

Keep the default of 1 if Reduced Media-Independent Interface (RMII) mode is used, otherwise set it to 0. The RMII standard is designed to reduce the number of signals required to connect a PHY to a MAC.

EMAC_ETHERNET_BUF_SIZE

The Ethernet buffer size. The default is 16384.

ETH_ISR_ID

The ISR ID. The default is ID_EMAC0.

ETH_ISR_PRIO

The ISR priority. The default is 0. On Atmel® ATSAM9A5D33 five priority bits are implemented so this is the lowest priority.

ETH_PHY_DEV_ADDR

The address of the external PHY. The default is 0x01.

ETHDRV_LINK_STA_POLL_INTERVAL

The link status poll interval in milliseconds. The default is 100.

MAX_RX_DESC

The maximum number of RX descriptors. The hardware does not use these descriptors for actual reception; data is copied into buffers that these descriptors point to. The default value is 40.

MAX_TX_DESC

The maximum number of TX descriptors. The default value is 40.

MAC_ADDRESS

The MAC address of the driver.

The default value is `{ 0x00u, 0xA0u, 0x91u, 0xFBu, 0x93u, 0xCDu }`

config_eth_phy_ksz9xxx.h

Set the PSP configuration options for the KSZ9xxx PHY in the file **src/config/config_eth_phy_ksz9xxx.h**. This section lists the available configuration options and their default values.

ETH_DUPLEX

The default is ETH_PHY_AUTONEG. This uses auto-negotiation to select full or half duplex.

ETH_SPEED

The default is ETH_PHY_AUTONEG. This uses auto-negotiation to select the speed.

psp_emac_eth_init

This function is provided by the PSP to initialize the Ethernet driver.

Format

```
t_nwdriver_ret psp_emac_eth_init ( void )
```

Arguments

None.

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

psp_emac_eth_start

This function is provided by the PSP to start the Ethernet driver.

Format

```
t_nwdriver_ret psp_emac_eth_start ( void )
```

Arguments

None.

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

psp_emac_eth_stop

This function is provided by the PSP to stop the Ethernet driver.

Format

```
t_nwdriver_ret psp_emac_eth_stop ( void )
```

Arguments

None.

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

psp_emac_eth_delete

This function is provided by the PSP to delete the Ethernet driver, releasing the associated resources.

Format

```
t_nwdriver_ret psp_emac_eth_delete ( void )
```

Arguments

None.

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

psp_emac_get_bufs

This function is provided by the PSP to get the addresses of the Ethernet buffers.

Format

```
t_nwdriver_ret psp_emac_get_bufs (
    uint8_t * * const   pp_buf,
    uint8_t * * const   pp_temp_buf )
```

Arguments

Parameter	Description	Type
pp_buf	Where to store the buffer address.	uint8_t * *
pp_temp_buf	Where to store the address of the temporary RX buffers.	uint8_t * *

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.

psp_emac_set_speed

This function is provided by the PSP to set the speed to 10 Mbps or 100 Mbps.

Format

```
t_nwdriver_ret psp_emac_set_speed ( uint32_t speed )
```

Arguments

Parameter	Description	Type
speed	The speed to use.	uint32_t

Return Values

Return value	Description
NWDRIVER_SUCCESS	Successful execution.
NWDRIVER_ERROR	Operation failed.