FAQ0138

Frequently Asked Questions

AT32F407/437 Ethernet data packet lost

Questions:

If data packets are lost when AT32F407/437 is using EMAC port to transceive Ethernet data, how to fix this problem?

Answer:

1. **Check the clock source of Ethernet PHY**

   Never use PLL-divided 25MHz/50MHz to clock Ethernet PHY, as PLL-based clock may not be able to meet Ethernet PHY's clock requirements and cause possible loss of data packets.

   Here we will show how to provide a clock source to Ethernet PHY.

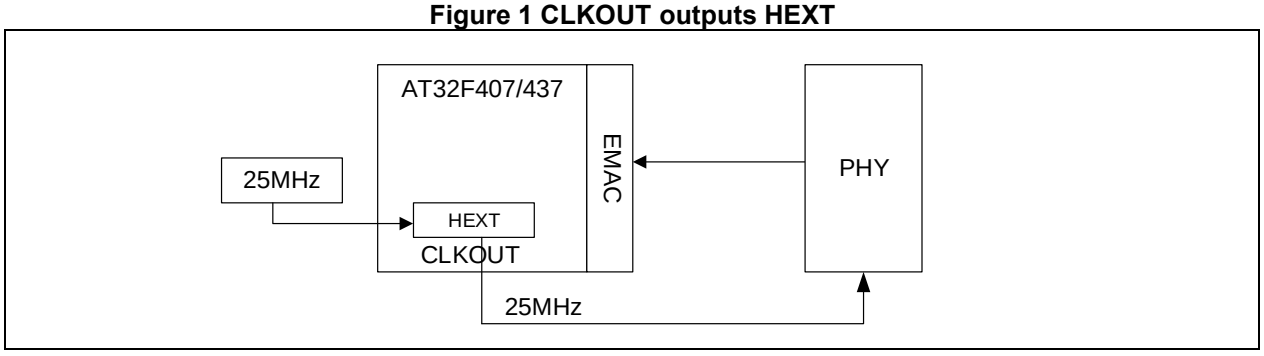
   In order to use EMAC on AT32F407/437, it is necessary to provide 25MHz/50MHz clock to Ethernet PHY.

   - To provide 25MHz clock to Ethernet PHY, follow Figure 1 and Figure 2 below for more information.
   - To provide 50MHz clock to Ethernet PHY, follow Figure 3 below for details. It is recommended to directly connect a 50 crystal oscillator to Ethernet PHY

   Note: Never use PLL-divided 25MHzz/50MHz as a clock source of Ethernet PHY.

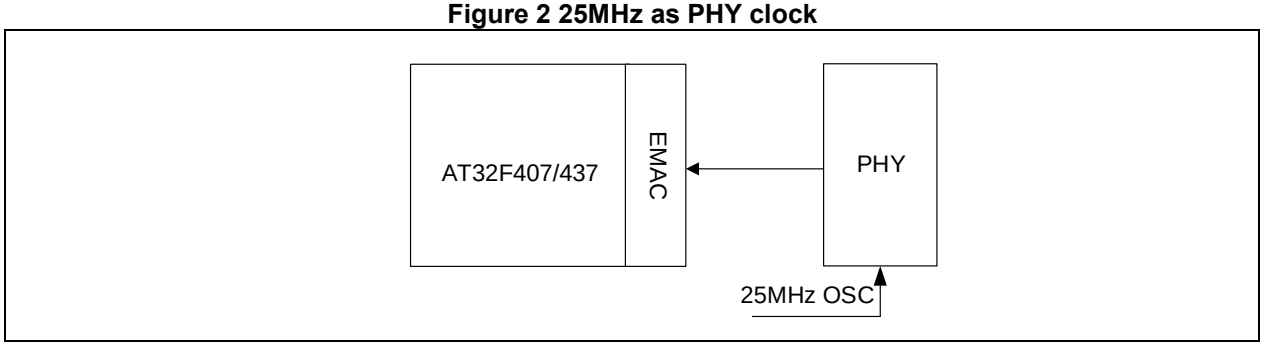
   Use a 25MHz crystal oscillator as HEXT of AT32F407, select HEXT as CLKOUT(PA8) clock source, and CLKOUT outputs a 25MHz to Ethernet PHY, as shown in Figure 1.

Figure 1 CLKOUT outputs HEXT



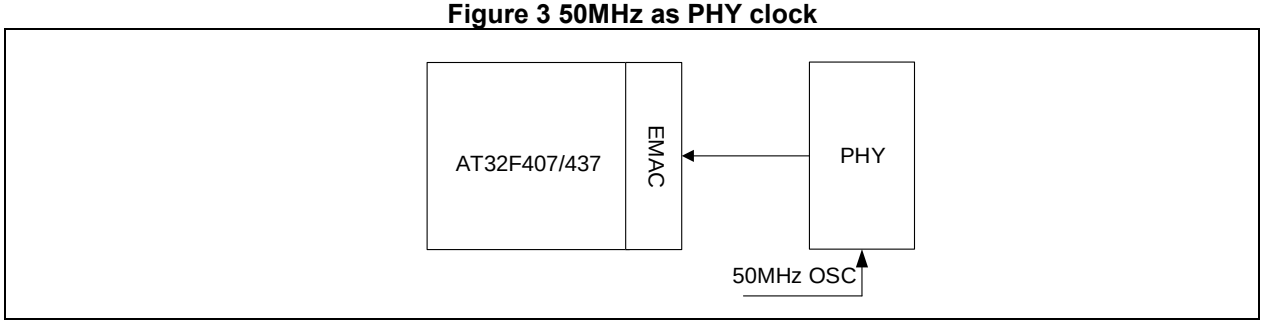
In Figure 2, directly select a 25MHz crystal oscillator as a clock source of Ethernet PHY.

Figure 2 25MHz as PHY clock



In Figure 3, use a 50MHz crystal oscillator as Ethernet PHY clock.

Figure 3 50MHz as PHY clock



2. Check if there is overflow and error occurring during EMA receive and transmit

A. For EMA receiver, confirm if there is receive overflow by checking the OVF bit of the EMAC_DMASTS register

Bit 4	OVF	0x0	rw1c	Receive Overflow This bit indicates that the receive buffer has an overflow during a frame reception. If the partial frame has been transferred to the application, the overflow status is set in the RDES0[11].

If the OVF bit is set, it indicates a lower software processing speed so that it cannot read all data from buffer area in time and thus cause overflow.

In this case, proceed as follows:

- Add the number of receive buffers (this will add the use of internal memory)

```
#define EMAC_RXBUFNB        10   //the number of receive buffer
#define EMAC_TXBUFNB        10   //the number of transmit buffer
```

- Increase software processing speed

  This can be achieved by using a maximum frequency (must be within spec)

B. For EMAC transmitter, check if there is a transmit error. For example, check if there is an ERROR reported as shown below.

```
error_status emac_txpkt_chainmode(u16 FrameLength)
{
  /* Check if the descriptor is owned by the ETHERNET DMA (when set) or CPU (when reset) */
  if((dma_tx_desc_to_set->status & EMAC_DMATXDESC_OWN) != (u32)RESET)
  {
    /* Return ERROR: OWN bit set */
    return ERROR;
  }
```

If there is an ERROR reported, it means that the software writes the transmit buffer so fast that EMAC is not able to send all data in the transmit buffer in time

In this case, proceed as follows:

- Add the number of transmit buffers (this will increase the use of internal memory)

```
#define EMAC_RXBUFNB        10   //the number of receive buffer
#define EMAC_TXBUFNB        10   //the number of transmit buffer
```

- Reduce the speed of software writing the transmit buffer area

  If EMAC speed is 100Mbps, the theoretical bus speed can get to 12.5MB/s. However it is not recommend to exceed such speed when writing the transmit buffer. The reason is that there may be other protocol overheads running on the bus, which means that the actual bus speed will be lower than 12.5MB/s.

- Add a “transfer complete” judgement to guarantee enough buffer for data storage

In the following example, it will check if the next buffer area has been fully transmitted after each transmit complete, in which, “timeout” value can be modified according to actual data size.

```
error_status emac_txpkt_chainmode(u16 FrameLength)
{
  uint32_t timeout = 0xFFFF;
  /* Check if the descriptor is owned by the ETHERNET DMA (when set) or CPU (when reset) */
  if((dma_tx_desc_to_set->status & EMAC_DMATXDESC_OWN) != (u32)RESET)
  {
    /* Return ERROR: OWN bit set */
    return ERROR;
  }

  /* Setting the Frame Length: bits[12:0] */
  dma_tx_desc_to_set->controlsize = (FrameLength & EMAC_DMATXDESC_TBS1);

  /* Setting the last segment and first segment bits (in this case a frame is transmitted in one descriptor) */
  dma_tx_desc_to_set->status |= EMAC_DMATXDESC_LS | EMAC_DMATXDESC_FS;

  /* Set Own bit of the Tx descriptor Status: gives the buffer back to ETHERNET DMA */
  dma_tx_desc_to_set->status |= EMAC_DMATXDESC_OWN;
  /* When Tx Buffer unavailable flag is set: clear it and resume transmission */
  if(emac_dma_flag_get(EMAC_DMA_TBU_FLAG))
  {
    /* Clear TBUS ETHERNET DMA flag */
    emac_dma_flag_clear(EMAC_DMA_TBU_FLAG);
    /* Resume DMA transmission*/
    EMAC_DMA->tpd_bit.tpd = 0;
  }

  /* Update the ETHERNET DMA global Tx descriptor with next Tx decriptor */
  /* Chained Mode */
  /* Selects the next DMA Tx descriptor list for next buffer to send */
  dma_tx_desc_to_set=(emac_dma_desc_type*) (dma_tx_desc_to_set->buf2nextdescaddr);

  while((dma_tx_desc_to_set->status & EMAC_DMATXDESC_OWN) != (u32)RESET)
  {
    if((timeout --) == 0)
    {
      break;
    }
  }
  /* Return SUCCESS */
  return SUCCESS;
}
```

Type: MCU application
Applicable products: AT32F407/437
Main function: EMAC
Other function: None

Document revision history

Date	Revision	Changes
2022.11.16	2.0.0	Initial release

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license granted by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner of such third party products or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement on any patent, copyright or other intellectual property right.

Purchasers hereby agree that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any aircraft application; (C) any aerospace application or environment; (D) any weapon application, and/or (E) or other uses where the failure of the device or product could result in personal injury, death, property damage. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and Purchasers are solely responsible for meeting all legal and regulatory requirements in such use.

Resale of ARTERY products with provisions different from the statements and/or technical characteristics stated in this document shall immediately void any warranty grant by ARTERY for ARTERY's products or services described herein and shall not create or expand any liability of ARTERY in any manner whatsoever.