Kernel Integration Flow Report

김동민 교육생

Table of Contents

1. 전체 구조도 및 관련 서브시스템

2. 송신
   2-1) 송신 흐름
   2-2) stmmac_xmit
   2-3) 송신 완료 처리

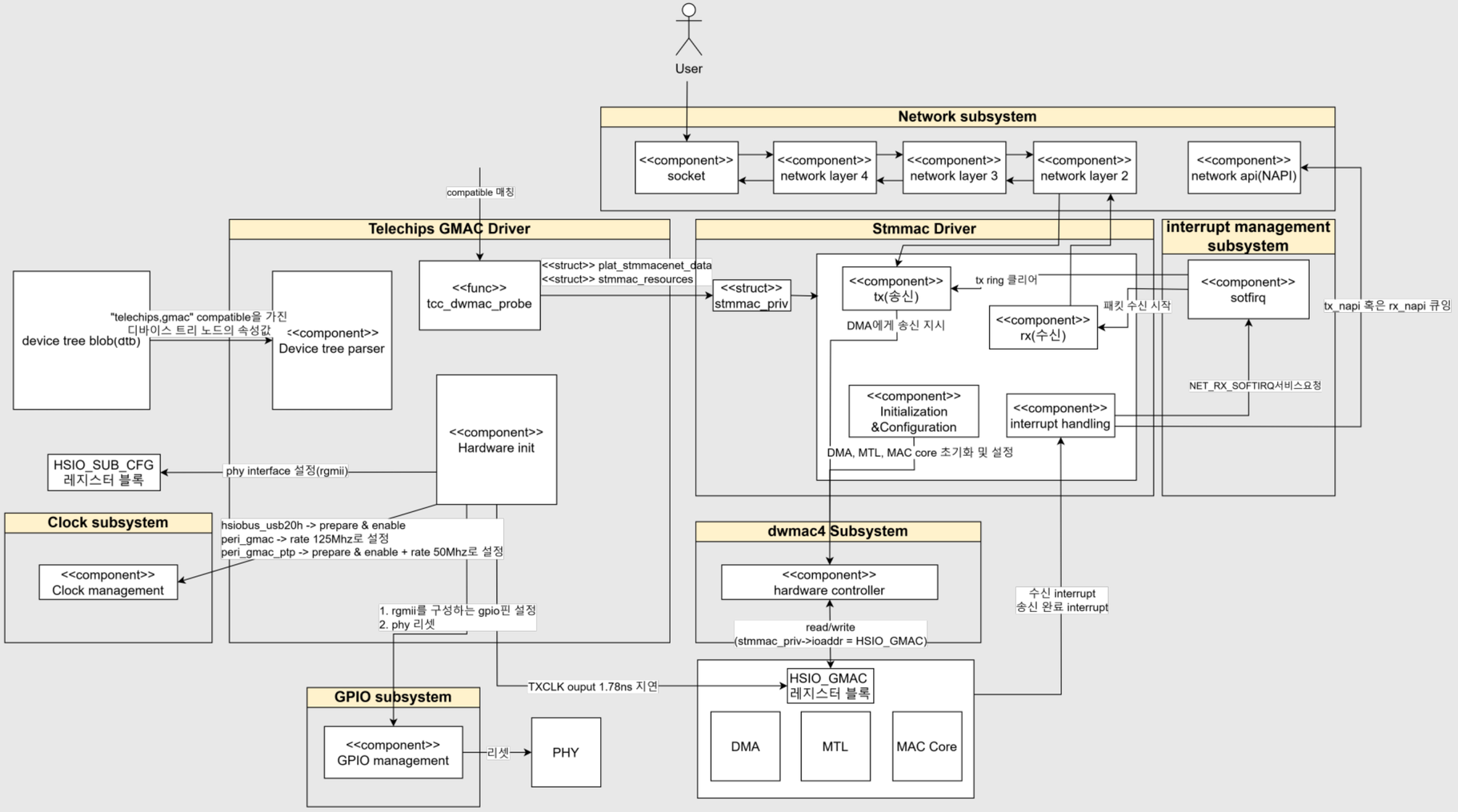
3. 수신
   3-1) 수신 흐름
   3-2) stmmac_rx

4. 송수신에서 DMA 작동 방식

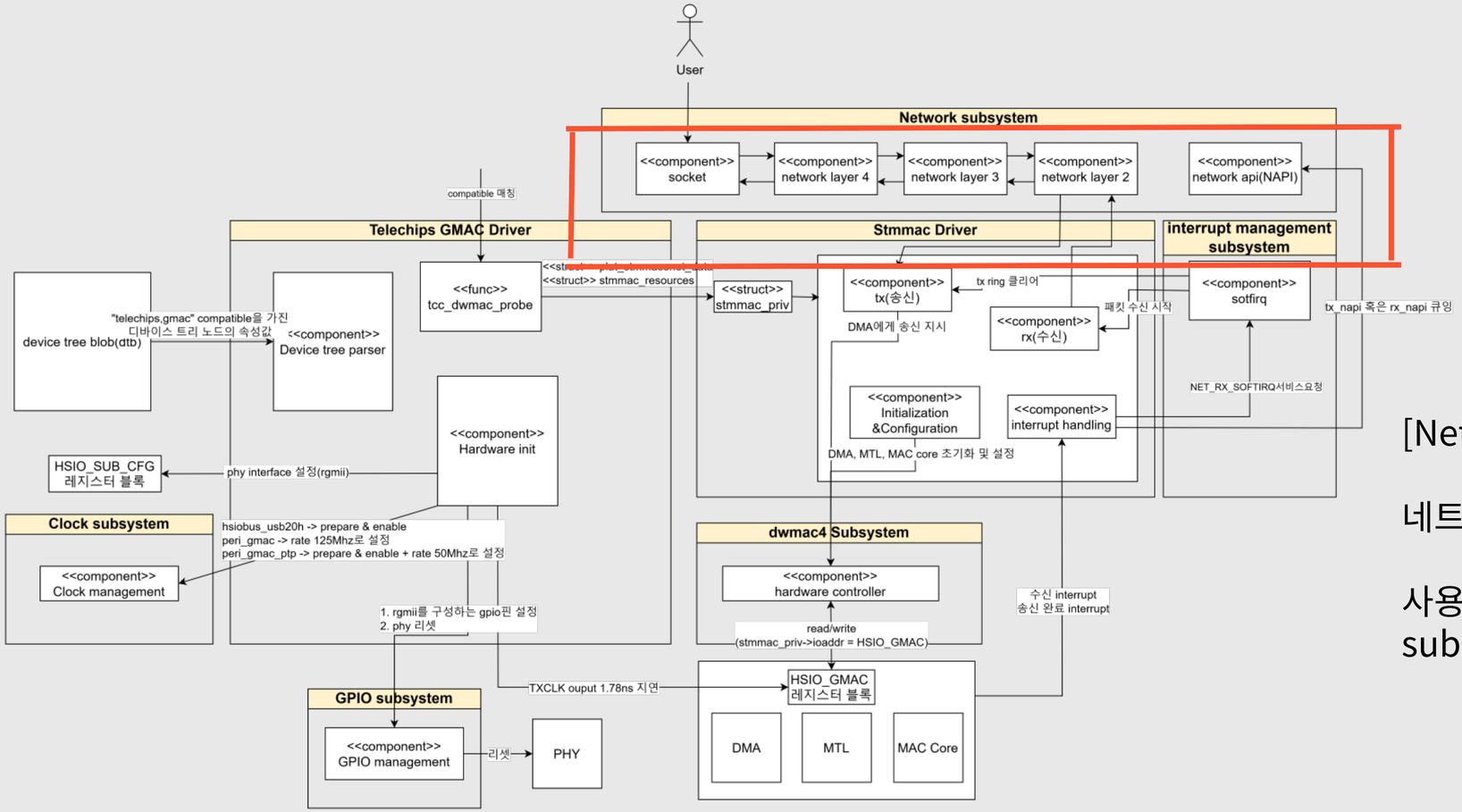
전체 구조도 및 연관 서브시스템

01

전체 구조도 및 연관 서브시스템



전체 구조도 및 연관 서브시스템

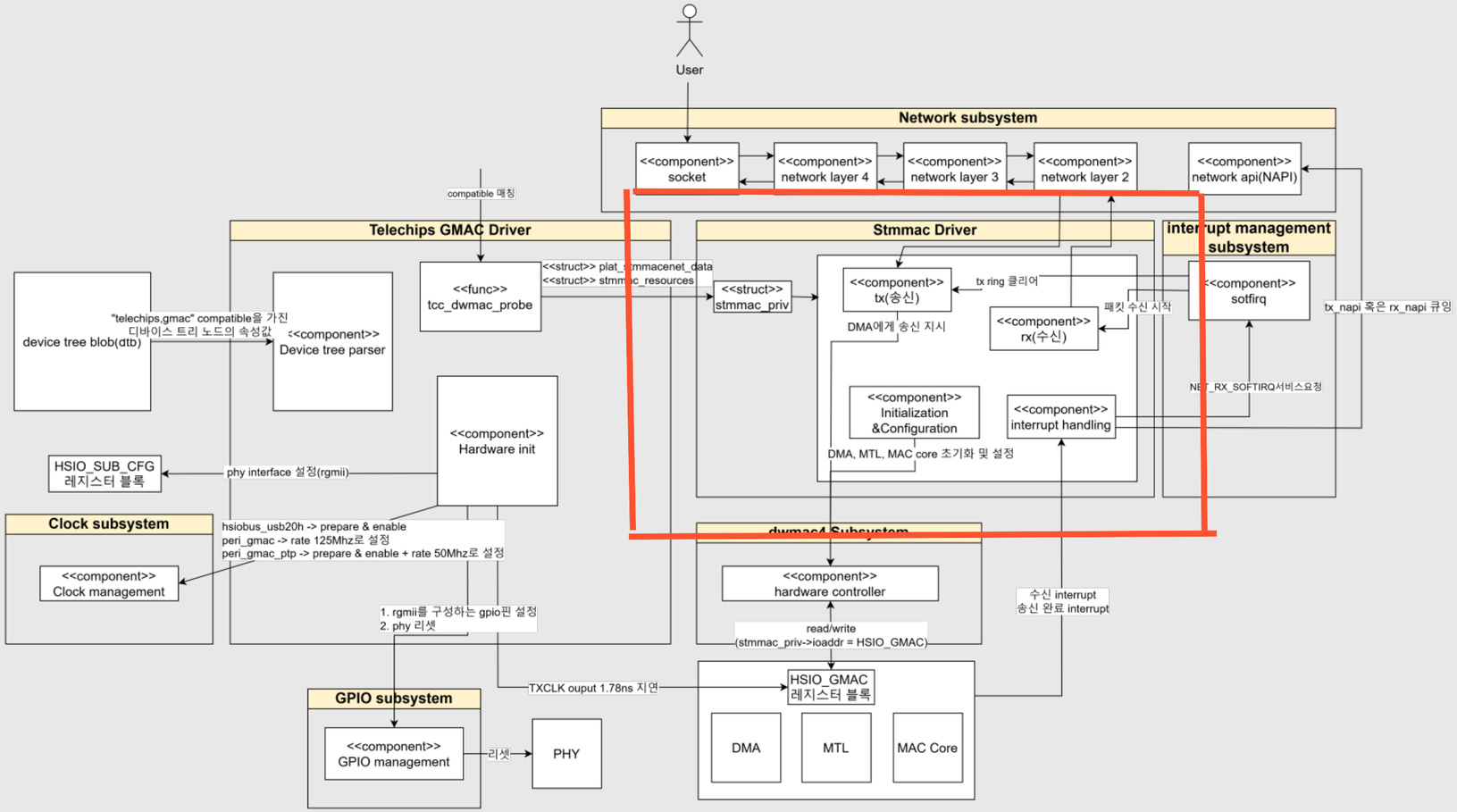


[Network subsystem]

네트워크 로직을 구현한 계층.

사용자가 네트워크 송수신을 요청하면 Network subsystem을 거쳐 Stmmac Driver에서 송수신을 수행

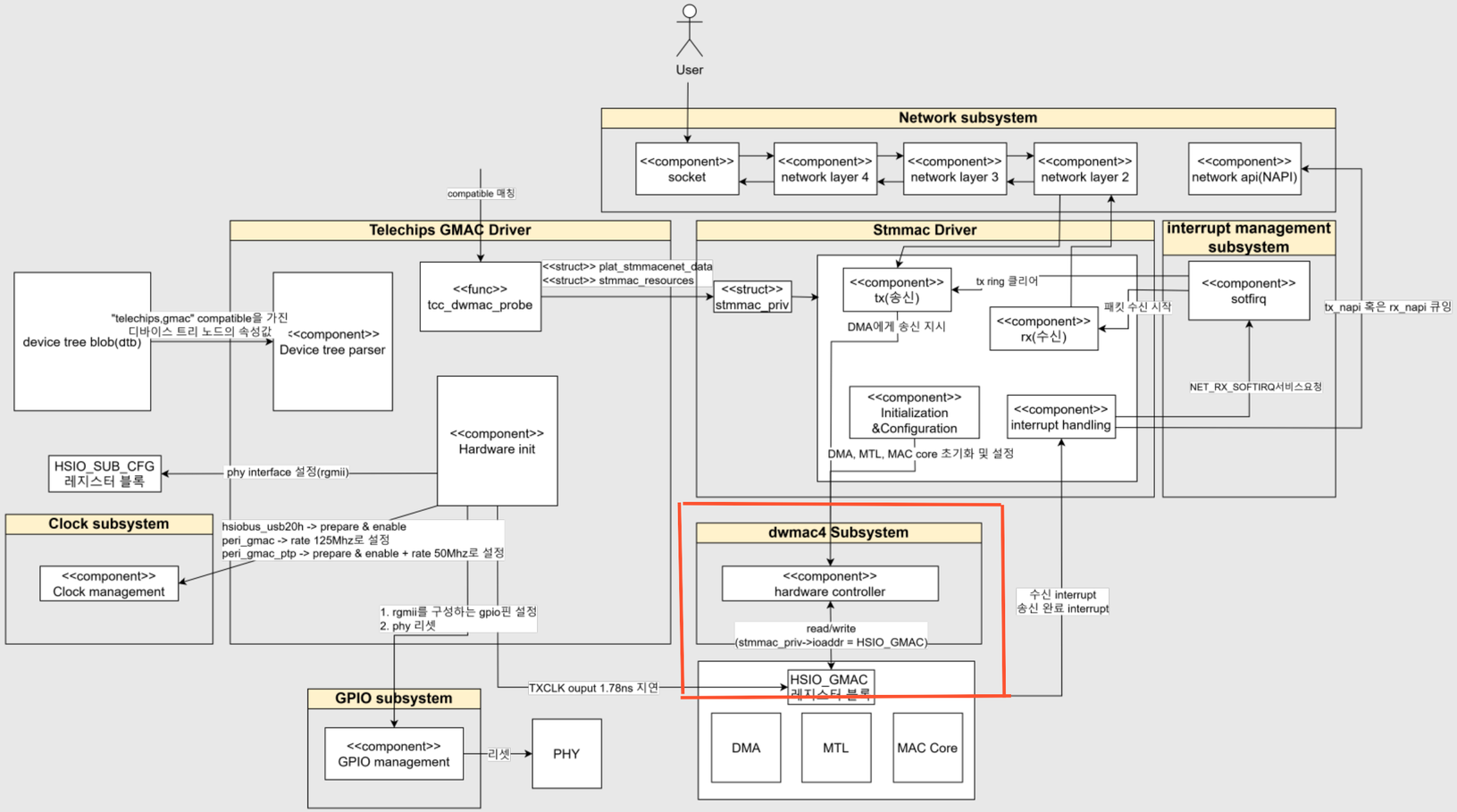
전체 구조도 및 연관 서브시스템



[Stmmac Driver]

네트워크 장치 초기화, 송신, 수신 등
실제 이더넷 기능은 여기서 구현된다.

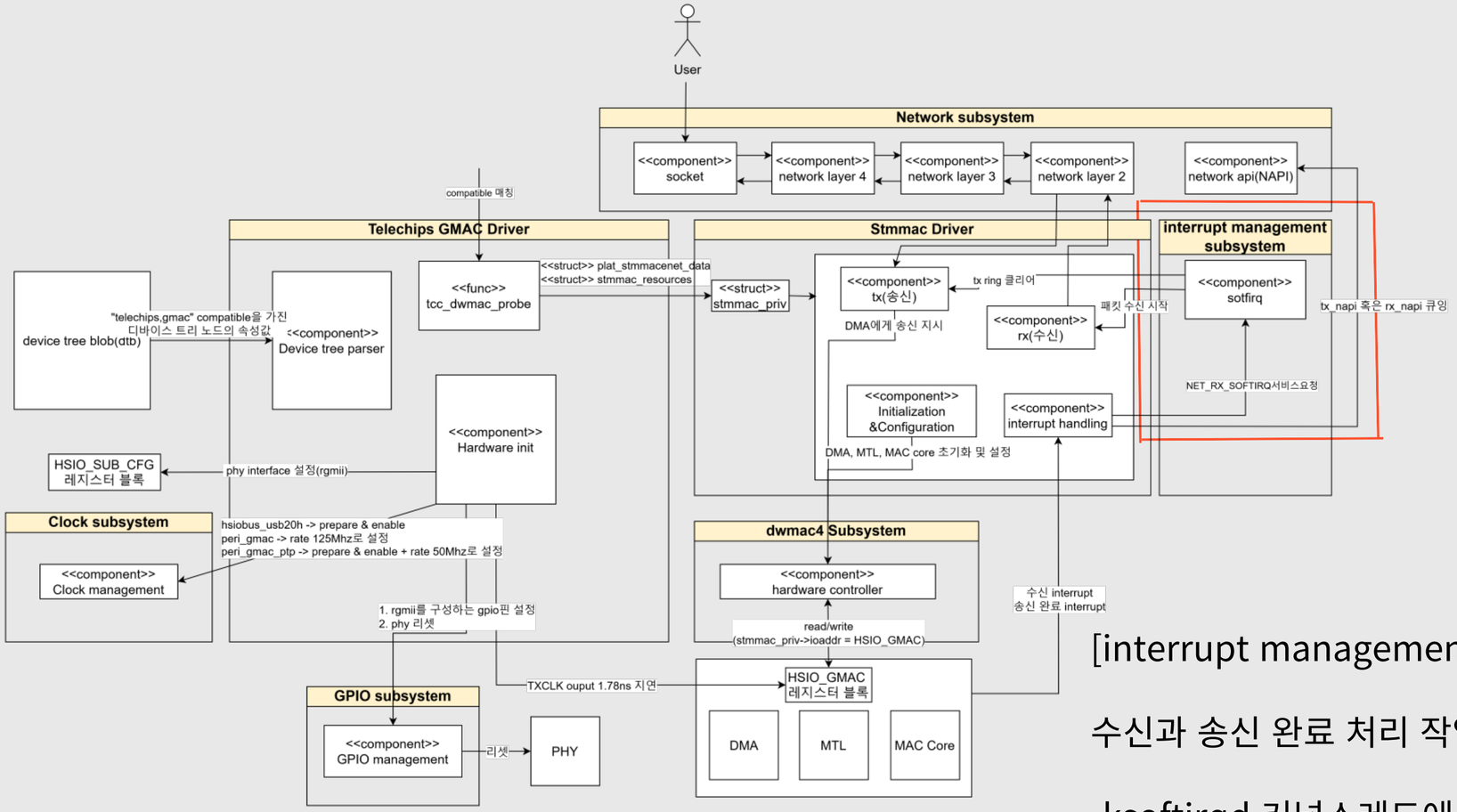
전체 구조도 및 연관 서브시스템



[dwmac4 subsystem]

GMAC 디바이스의 HSIO_GMAC레지스터에 직접 read/write하는 역할

전체 구조도 및 연관 서브시스템



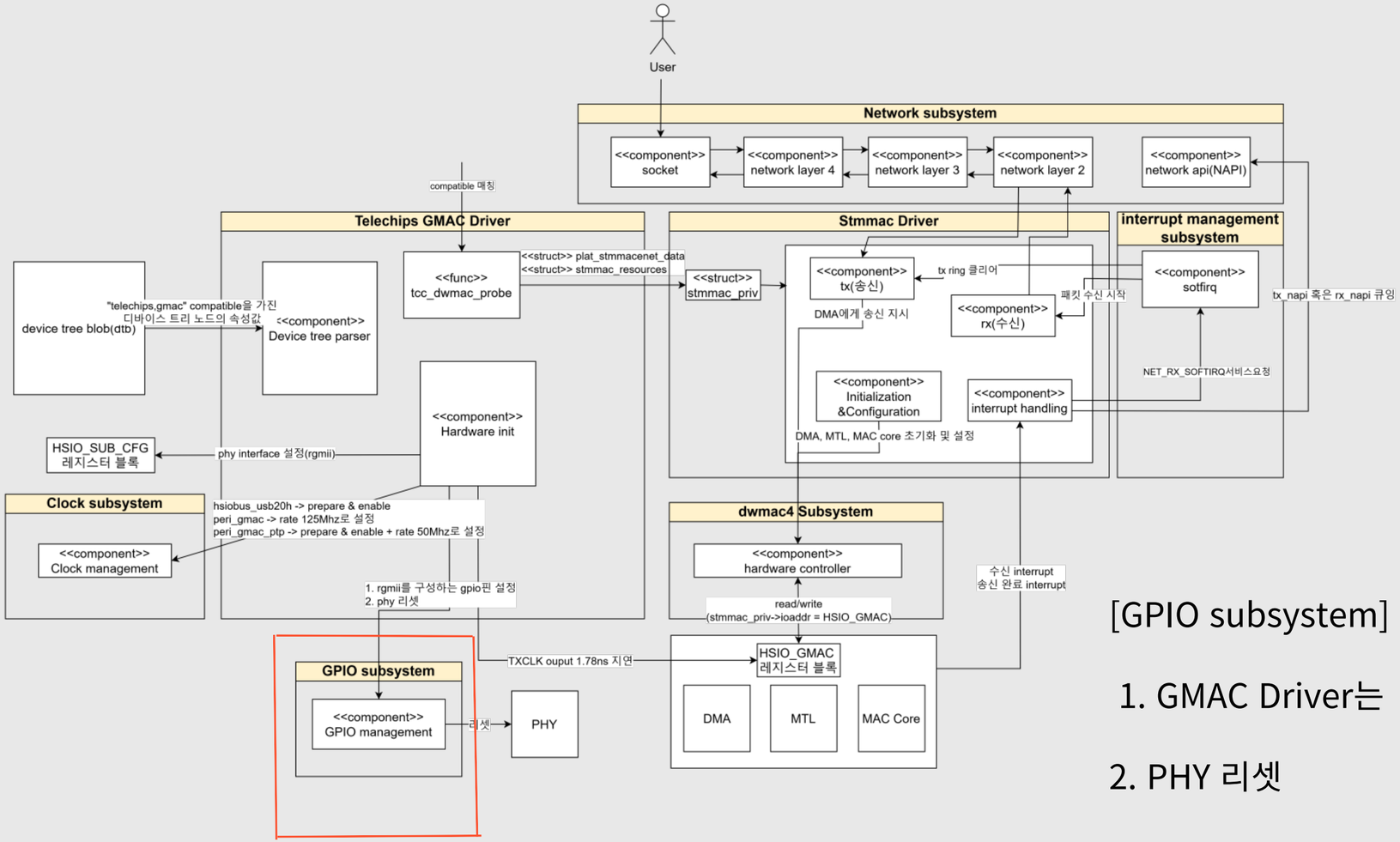
[interrupt management subsystem]

수신과 송신 완료 처리 작업을 수행하기 위해 NET_RX_SOFTIRQ 서비스를 요청

-ksoftirqd 커널스레드에서 아래의 작업을 수행

- 수신
- 송신 완료 처리

전체 구조도 및 연관 서브시스템

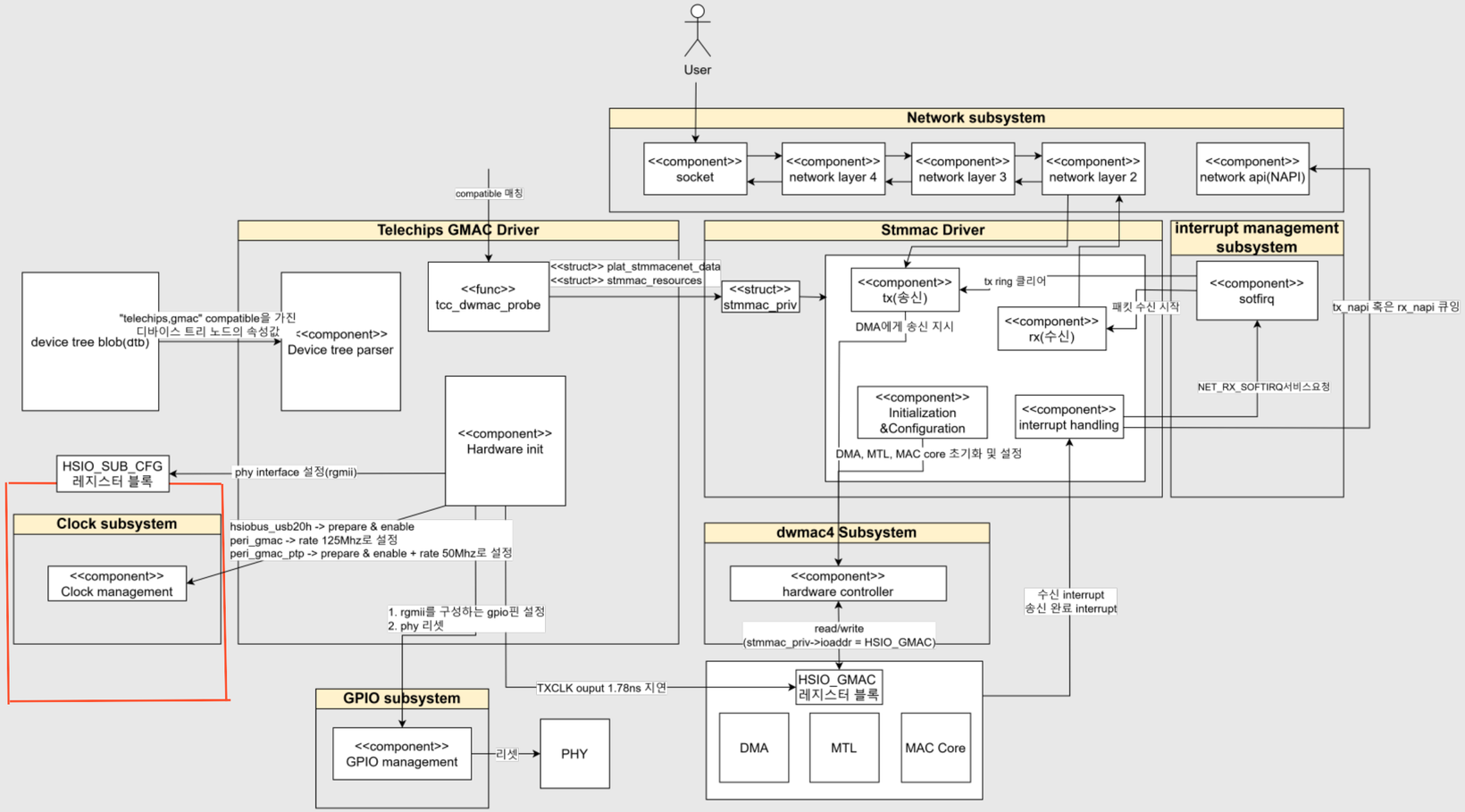


[GPIO subsystem]

1. GMAC Driver는 rgmii인터페이스에 필요한 GPIO 핀을 설정

2. PHY 리셋

전체 구조도 및 연관 서브시스템



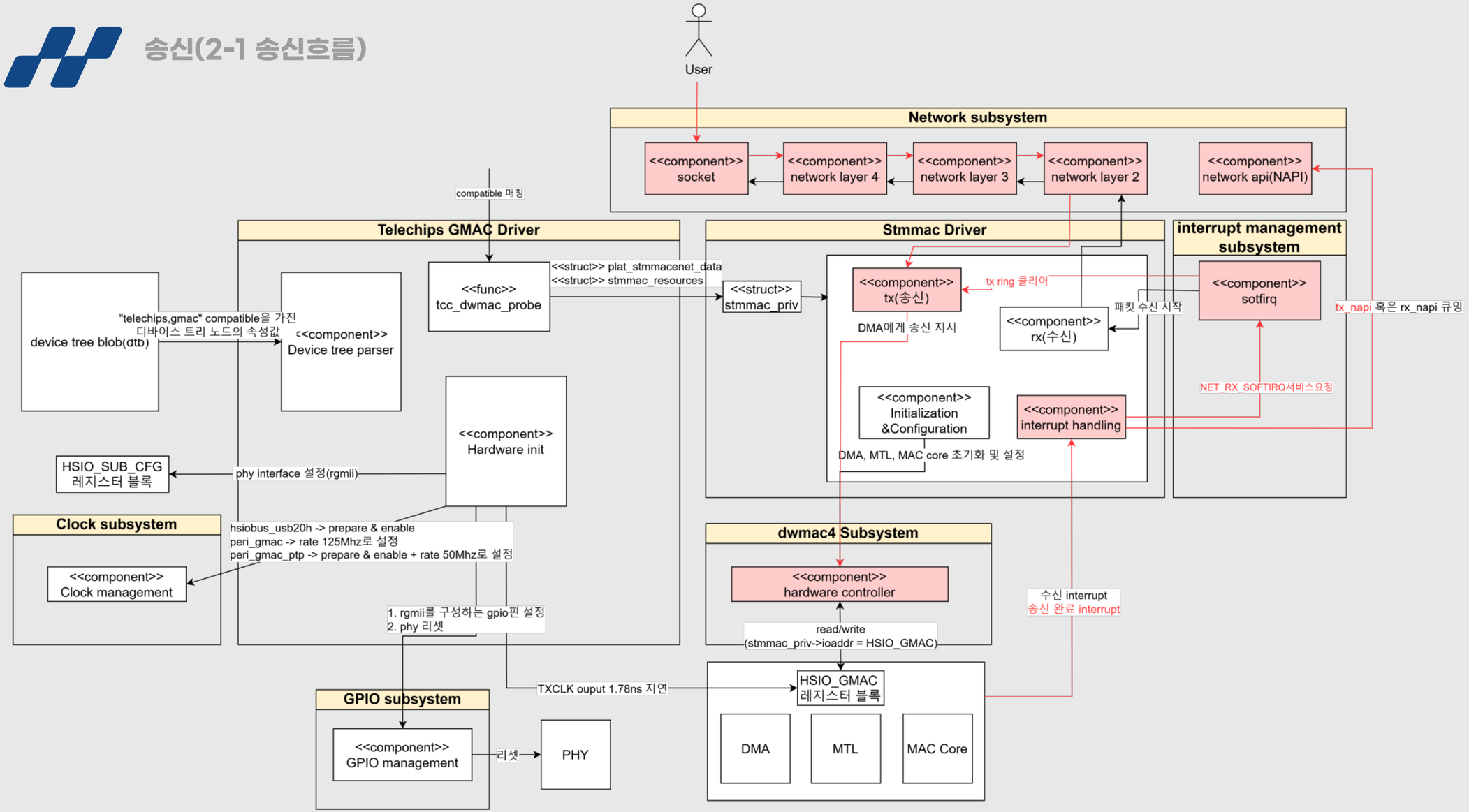
[Clock subsystem]
GMAC 디바이스에 필요한 클럭소스를 설정

송신

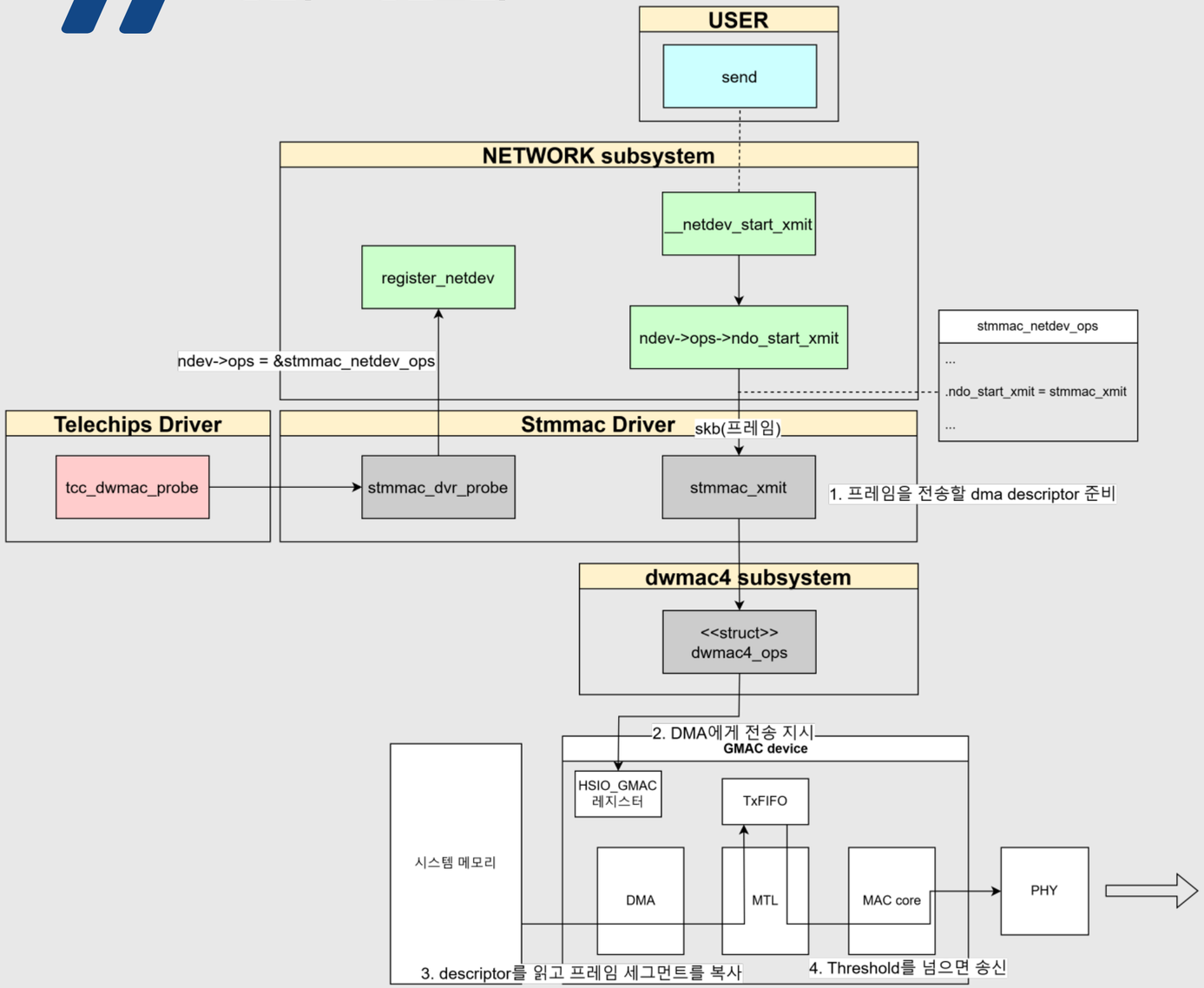
2-1) 송신 흐름

02

송신(2-1 송신흐름)


User
Network subsystem
<<component>> socket
<<component>> network layer 4
<<component>> network layer 3
<<component>> network layer 2
<<component>> network api(NAPI)
compatible 매칭
Telechips GMAC Driver
Stmmac Driver
interrupt management subsystem
<<func>> tcc_dwmac_probe
<<struct>> plat_stmmacenet_data
<<struct>> stmmac_resources
<<struct>> stmmac_priv
<<component>> tx(송신)
tx ring 클리어
<<component>> rx(수신)
패킷 수신 시작
<<component>> sotfirq
tx_napi 혹은 rx_napi 큐잉
"telechips,gmac" compatible을 가진 디바이스 트리 노드의 속성값
device tree blob(dtb)
<<component>> Device tree parser
DMA에게 송신 지시
NET_RX_SOFTIRQ서비스요청
<<component>> Initialization &Configuration
<<component>> interrupt handling
DMA, MTL, MAC core 초기화 및 설정
<<component>> Hardware init
HSIO_SUB_CFG 레지스터 블록
phy interface 설정(rgmii)
Clock subsystem
hsiobus_usb20h -> prepare & enable
peri_gmac -> rate 125Mhz로 설정
peri_gmac_ptp -> prepare & enable + rate 50Mhz로 설정
<<component>> Clock management
dwmac4 Subsystem
<<component>> hardware controller
수신 interrupt
송신 완료 interrupt
1. rgmii를 구성하는 gpio핀 설정
2. phy 리셋
read/write
(stmmac_priv->ioaddr = HSIO_GMAC)
HSIO_GMAC 레지스터 블록
TXCLK ouput 1.78ns 지연
GPIO subsystem
DMA
MTL
MAC Core
<<component>> GPIO management
리셋
PHY

송신(2-1 송신흐름)



- 송신함수 등록
  - probe과정에서 stmmac_netdev_ops를 등록
    해당 구조체에 송신함수(stmmac_xmit)포함

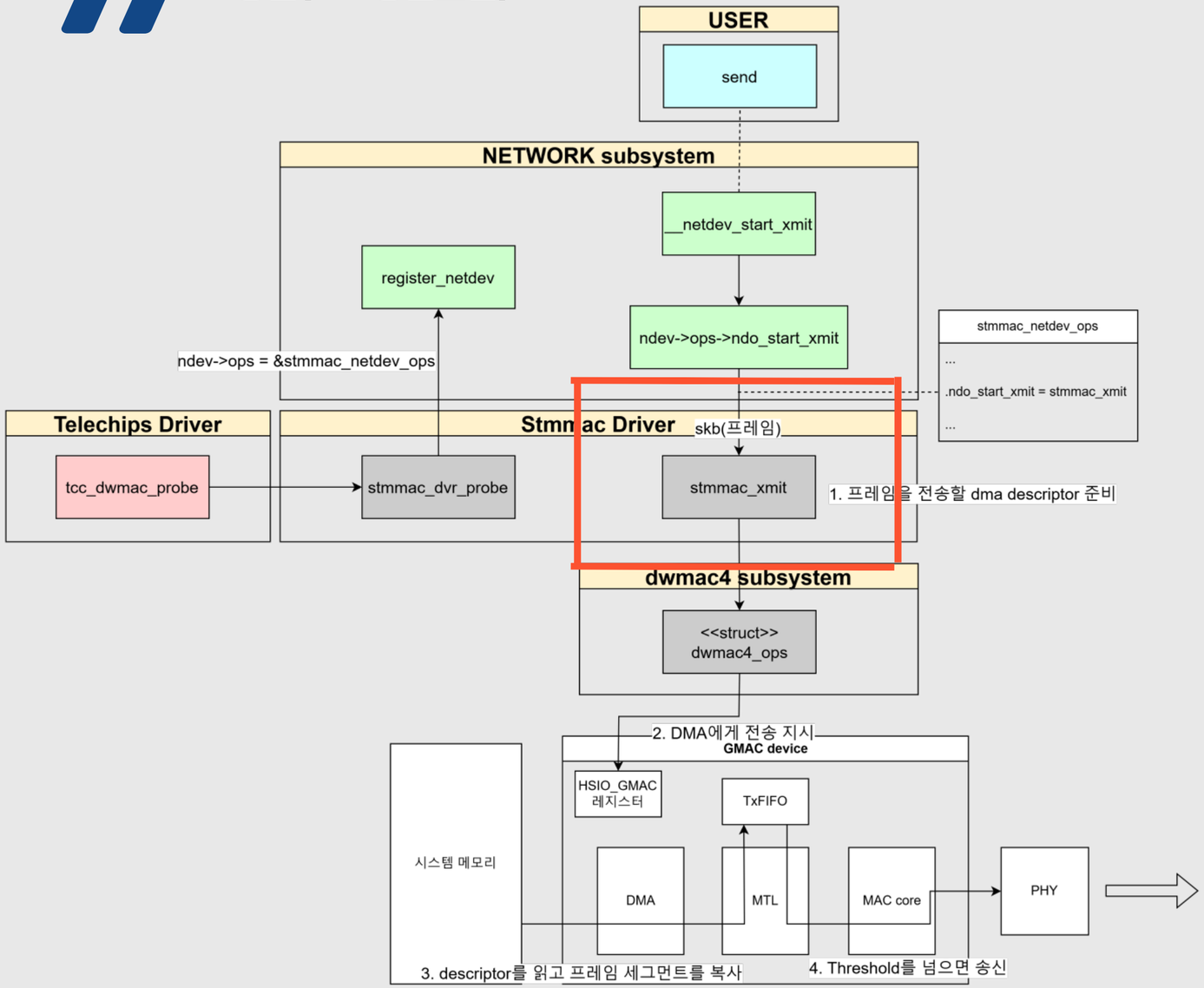
- 송신 수행과정

  1. application에서 send()

  2. network subsystem

  2. stmmac_xmit()
     - ■송신 프레임에 대해서 descriptor 준비
     - ■DMA에게 전송 지시
     - ■DMA가 TxFIFO로 데이터 복사
     - ■TxFIFO가 threshold를 초과하면 전송시작

송신(2-1 송신흐름)



- 송신함수 등록
  - probe과정에서 stmmac_netdev_ops를 등록
    해당 구조체에 송신함수(stmmac_xmit)포함

- 송신 수행과정

  1. application에서 send()

  2. network subsystem

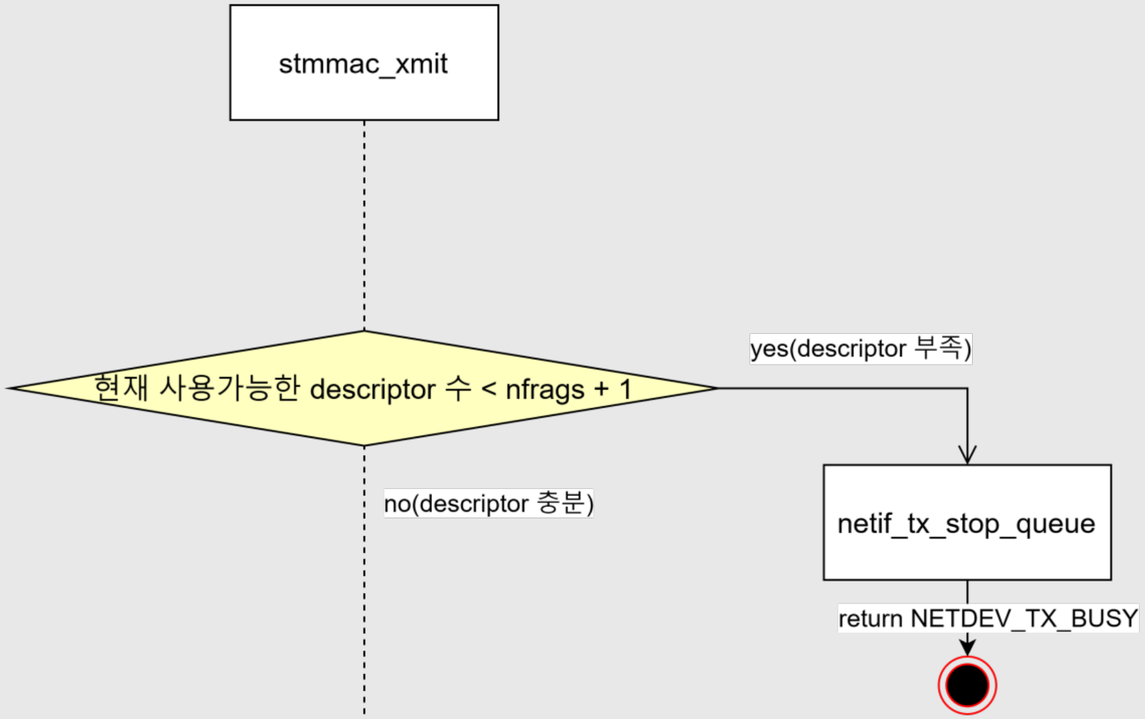
  2. stmmac_xmit()
     - ■송신 프레임에 대해서 descriptor 준비
     - ■DMA에게 전송 지시
     - ■DMA가 TxFIFO로 데이터 복사
     - ■TxFIFO가 threshold를 초과하면 전송시작

송신

2-2) stmmac_xmit

02

송신(2-2 stmmac_xmit)

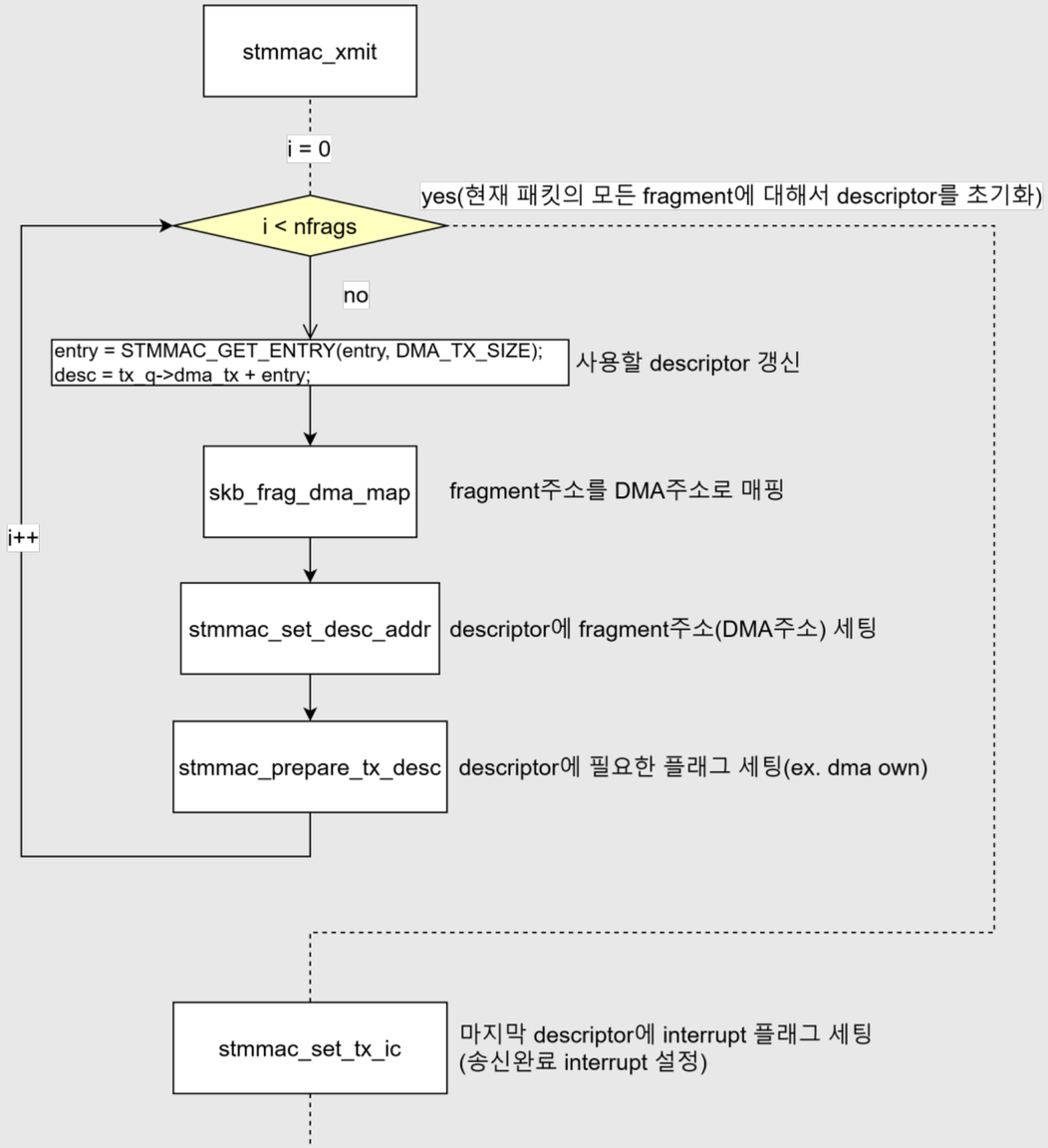


1. 송신에 descriptor 수가 충분한지 확인한다.
   (부족하면 송신 중지)

왜 nfrags + 1? :

- skb->data = 1
- nfrags = 프레임의 fragment개수(>=0)

송신(2-2 stmmac_xmit)

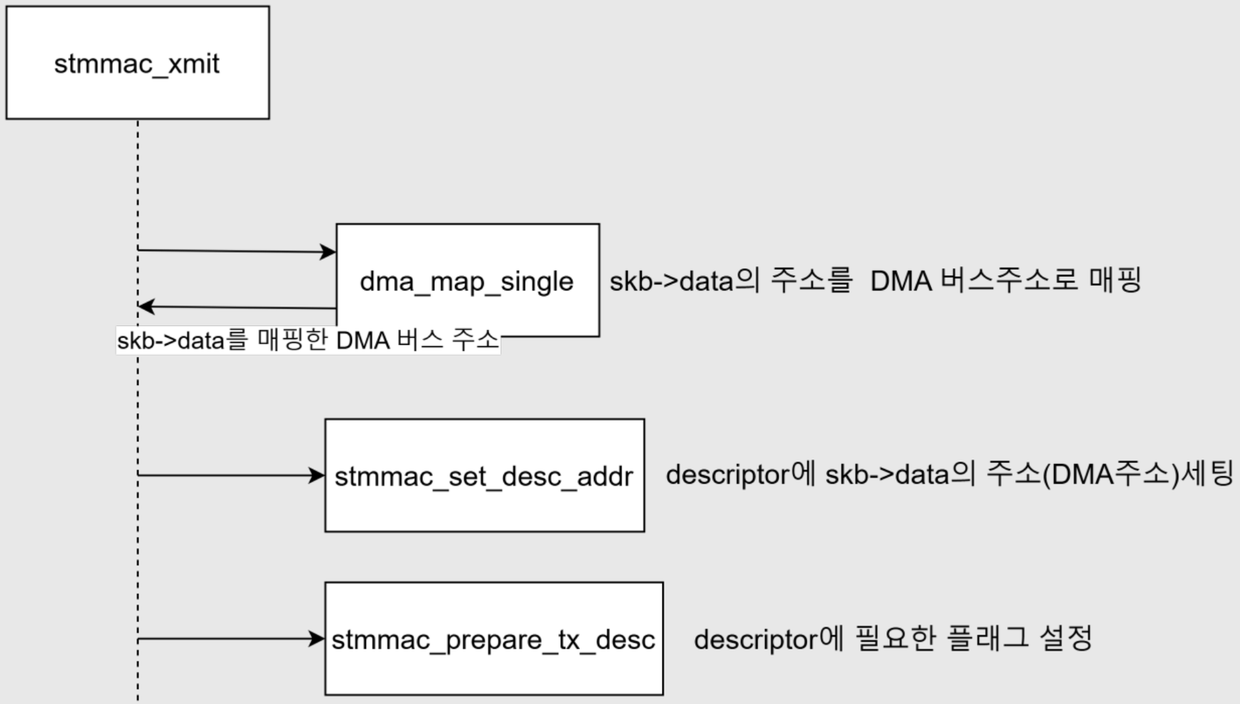


- fragment에 대해서 descriptor 준비

이후 DMA가 시스템메모리에 있는 fragment에 접근할 수 있도록 descriptor에 fragment의 주소를 저장

*fragment = 프레임의 payload부분을 저장

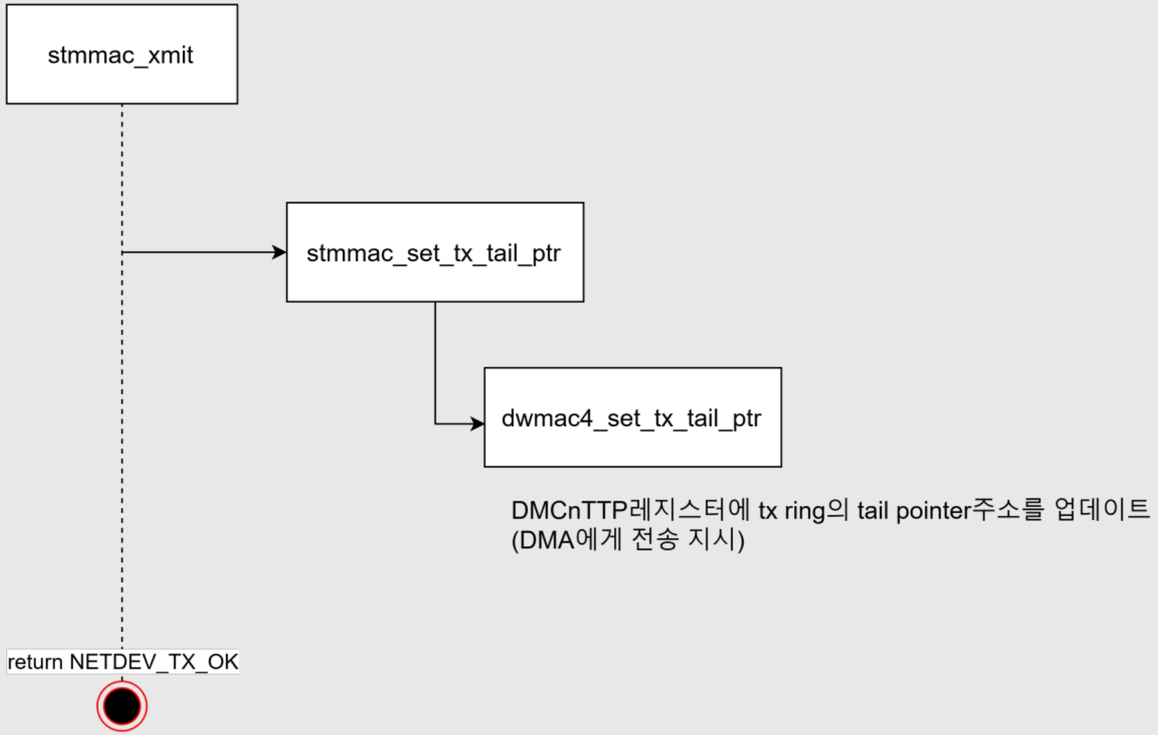
송신(2-2 stmmac_xmit)



- 마찬가지로 skb->data에 대해서도 descriptor를 준비

*skb->data = 프레임의 헤더 + payload 일부 저장

송신(2-2 stmmac_xmit)



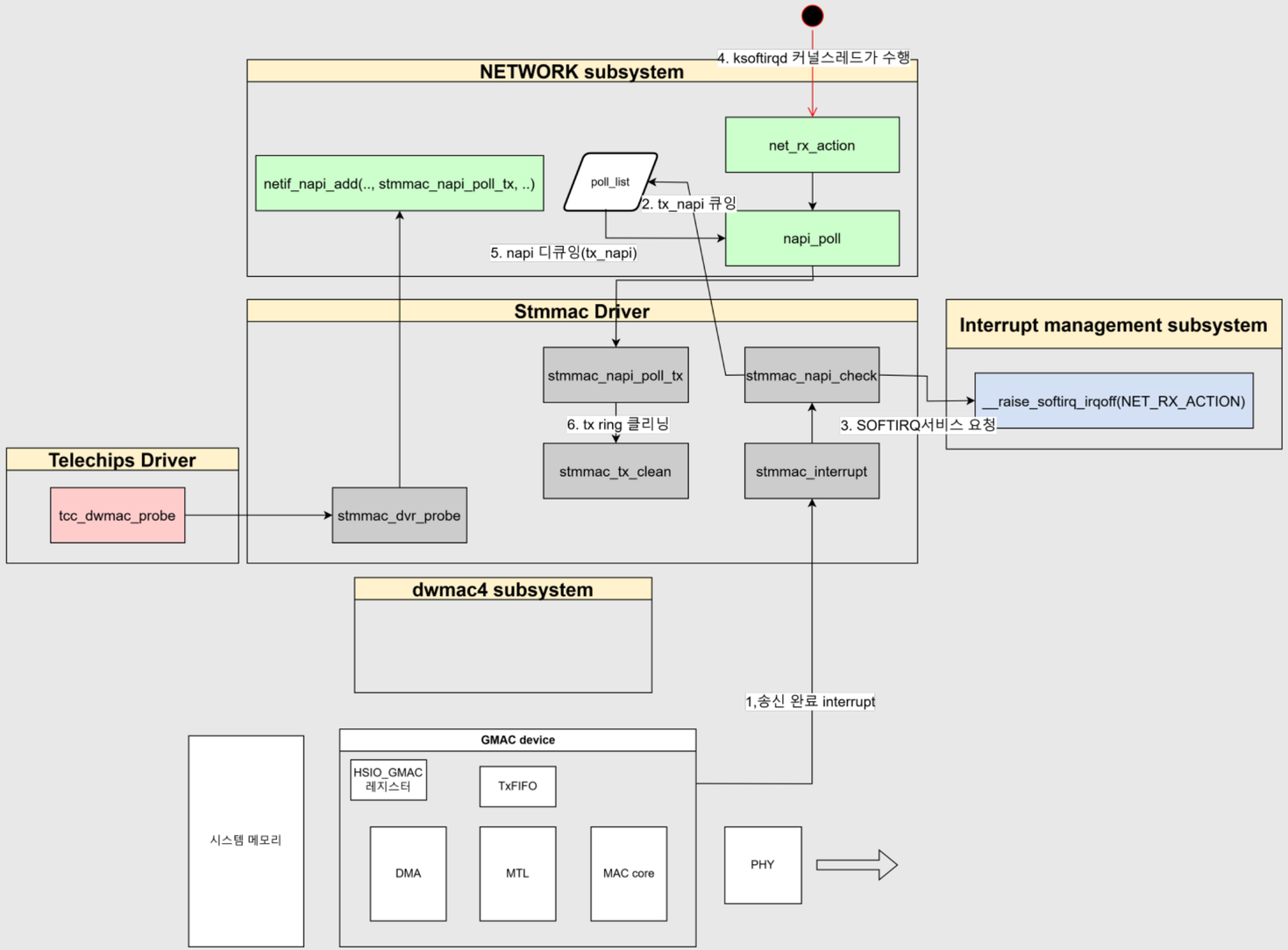
- DMA에게 tx ring의 tail pointer를 업데이트 -> 전송 지시
  - DMA는 descriptor를 읽고 프레임 조각을 TxFIFO로 복사
  - TxFIFO가 threshold를 넘으면 PHY로 데이터를 보내어 송신

송신

2-3) 송신 완료 처리

02

송신(2-3 송신 완료 처리)



stmmac_tx_clean에서 송신완료 처리작업 수행

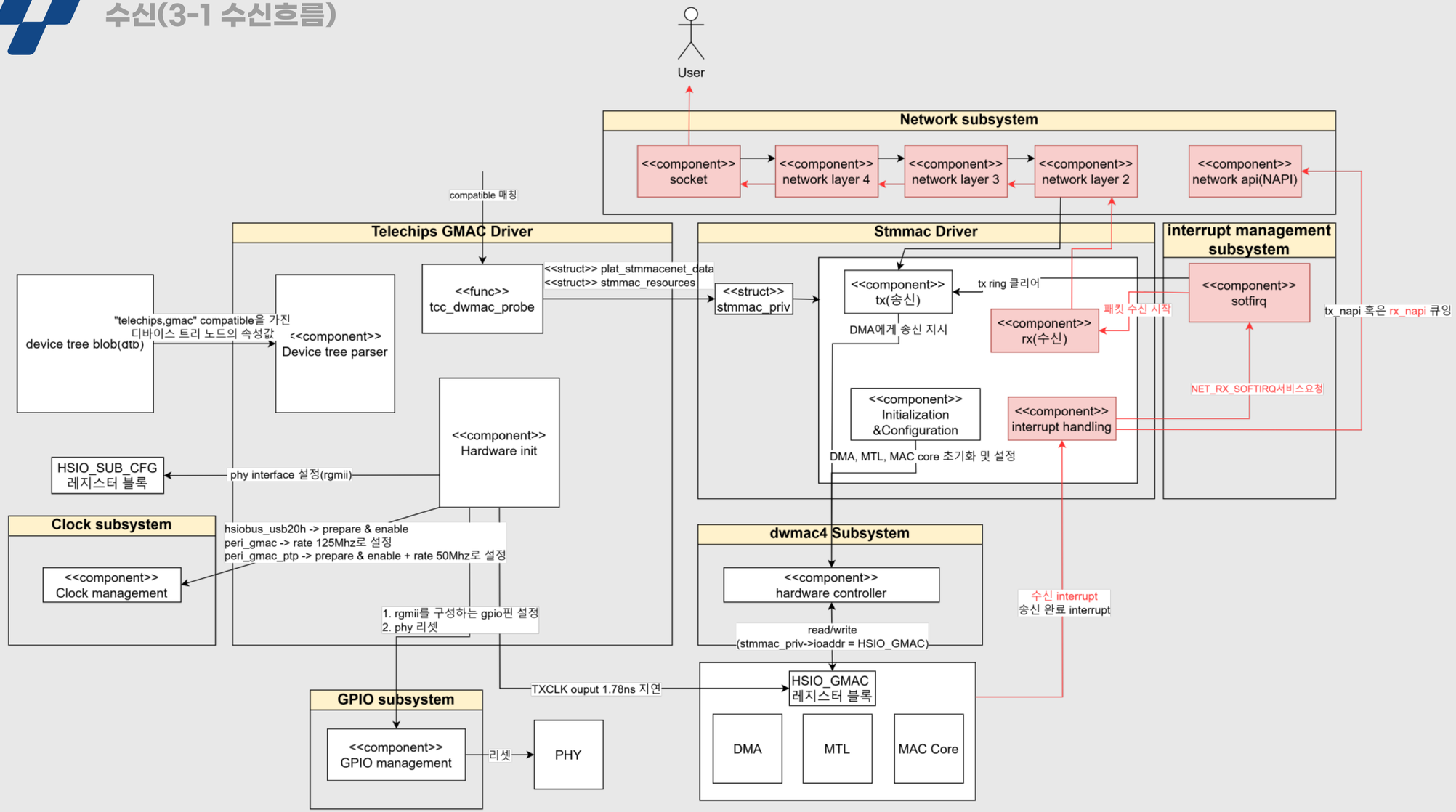
- dma 매핑 해제
- 송신에 사용했던 descriptor 정리
(재사용 가능하도록)

수신

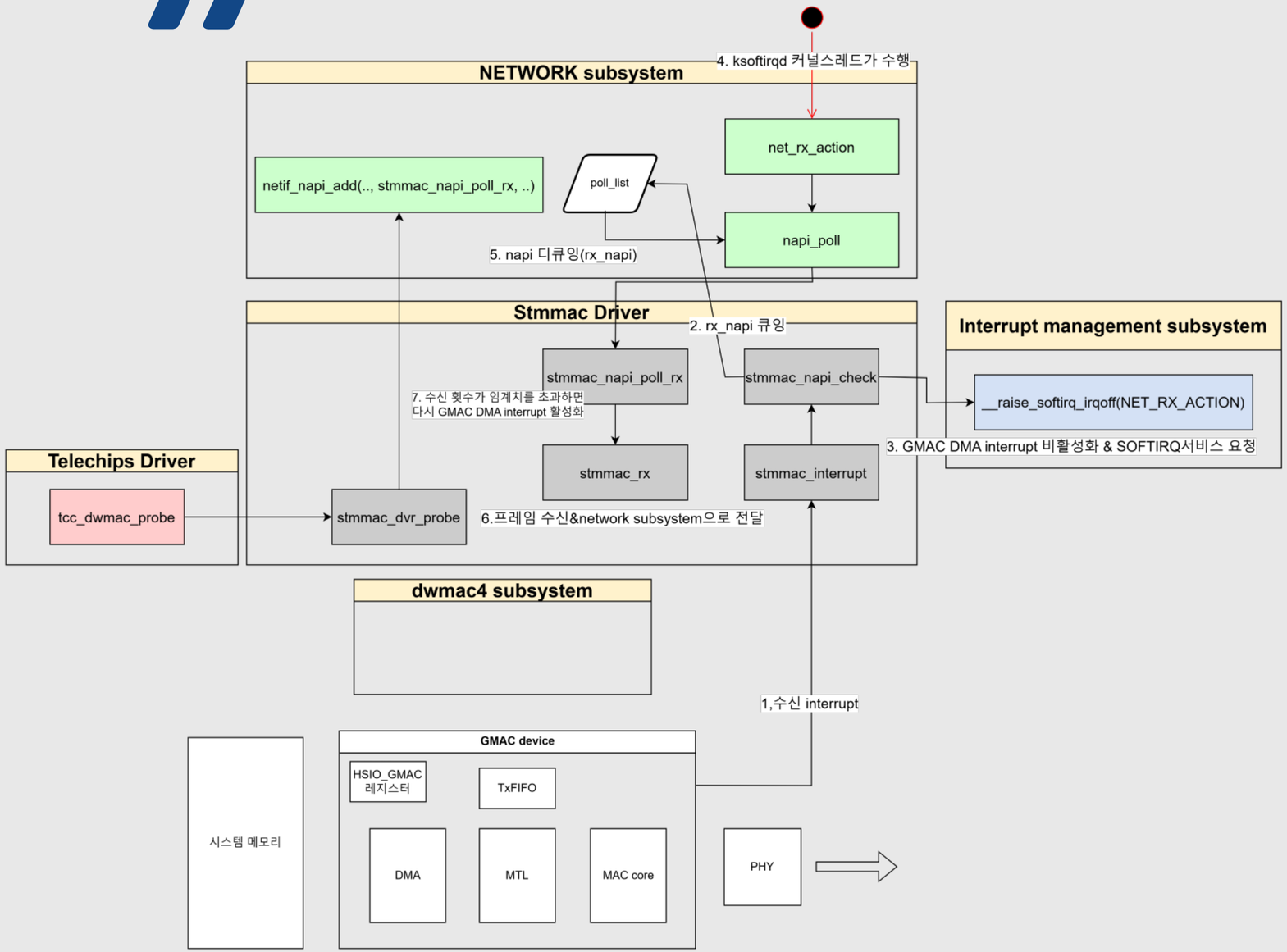
3-1) 수신 흐름

03

수신(3-1 수신흐름)

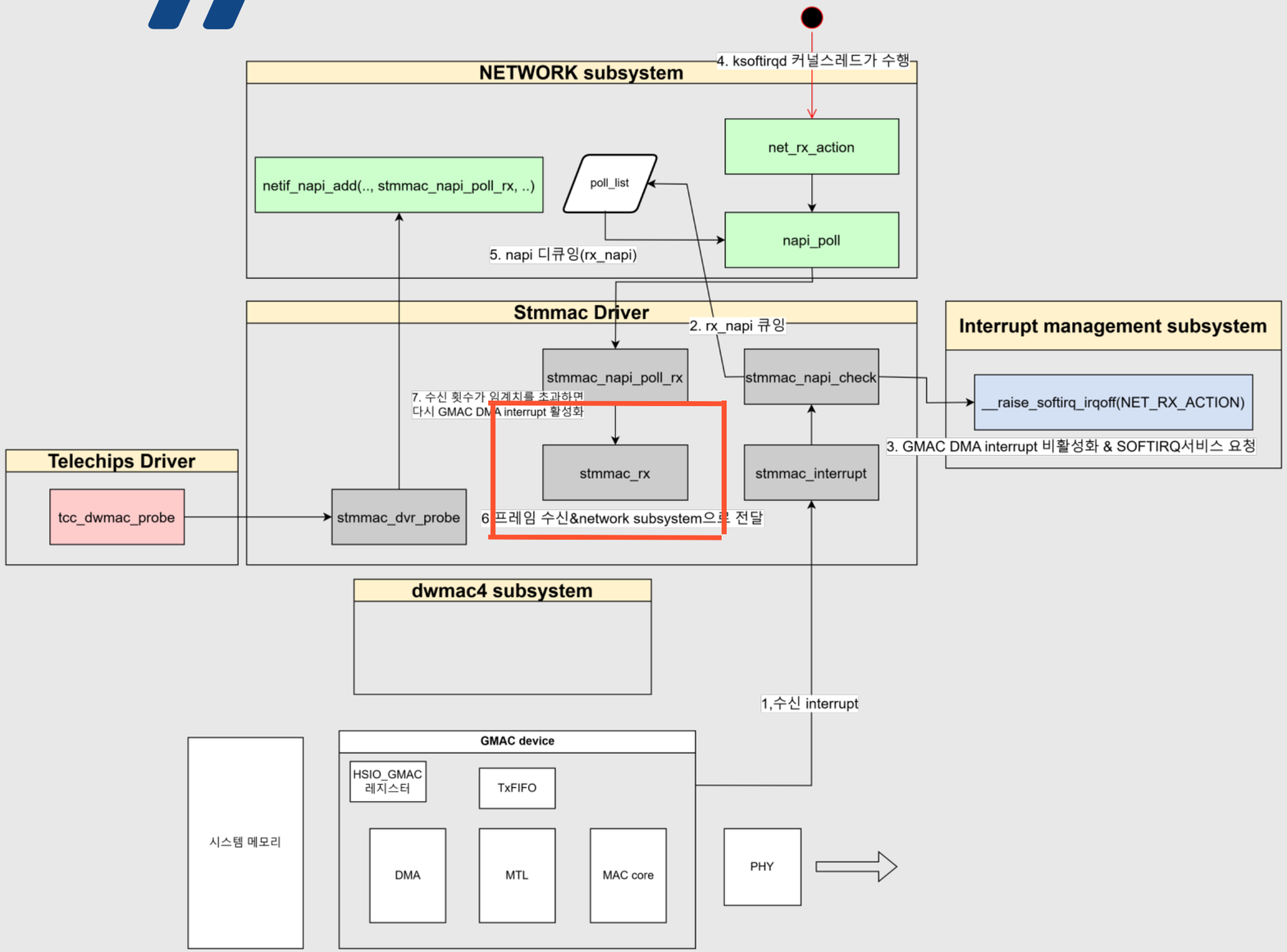

User
Network subsystem
<<component>> socket
<<component>> network layer 4
<<component>> network layer 3
<<component>> network layer 2
<<component>> network api(NAPI)
compatible 매칭
Telechips GMAC Driver
Stmmac Driver
interrupt management subsystem
<<func>> tcc_dwmac_probe
<<struct>> plat_stmmacenet_data
<<struct>> stmmac_resources
<<struct>> stmmac_priv
<<component>> tx(송신)
tx ring 클리어
<<component>> sotfirq
tx_napi 혹은 rx_napi 큐잉
패킷 수신 시작
<<component>> rx(수신)
DMA에게 송신 지시
"telechips,gmac" compatible을 가진 디바이스 트리 노드의 속성값
device tree blob(dtb)
<<component>> Device tree parser
NET_RX_SOFTIRQ서비스요청
<<component>> Initialization &Configuration
<<component>> interrupt handling
DMA, MTL, MAC core 초기화 및 설정
<<component>> Hardware init
HSIO_SUB_CFG 레지스터 블록
phy interface 설정(rgmii)
Clock subsystem
hsiobus_usb20h -> prepare & enable
peri_gmac -> rate 125Mhz로 설정
peri_gmac_ptp -> prepare & enable + rate 50Mhz로 설정
<<component>> Clock management
dwmac4 Subsystem
<<component>> hardware controller
수신 interrupt
송신 완료 interrupt
1. rgmii를 구성하는 gpio핀 설정
2. phy 리셋
read/write
(stmmac_priv->ioaddr = HSIO_GMAC)
TXCLK ouput 1.78ns 지연
HSIO_GMAC 레지스터 블록
GPIO subsystem
DMA
MTL
MAC Core
<<component>> GPIO management
리셋
PHY

수신(3-1 수신흐름)



- 수신 핸들러 등록
  - probe과정에서 수신 napi poll 핸들러 등록 (stmmac_napi_poll_rx)
- stmmac_napi_poll_rx -> stmmac_rx
  1. DMA가 복사해놓은 데이터를 모아 프레임을 완성해나감.
  2. 프레임을 완성하면 상위 네트워크 계층에 전송

수신(3-1 수신흐름)



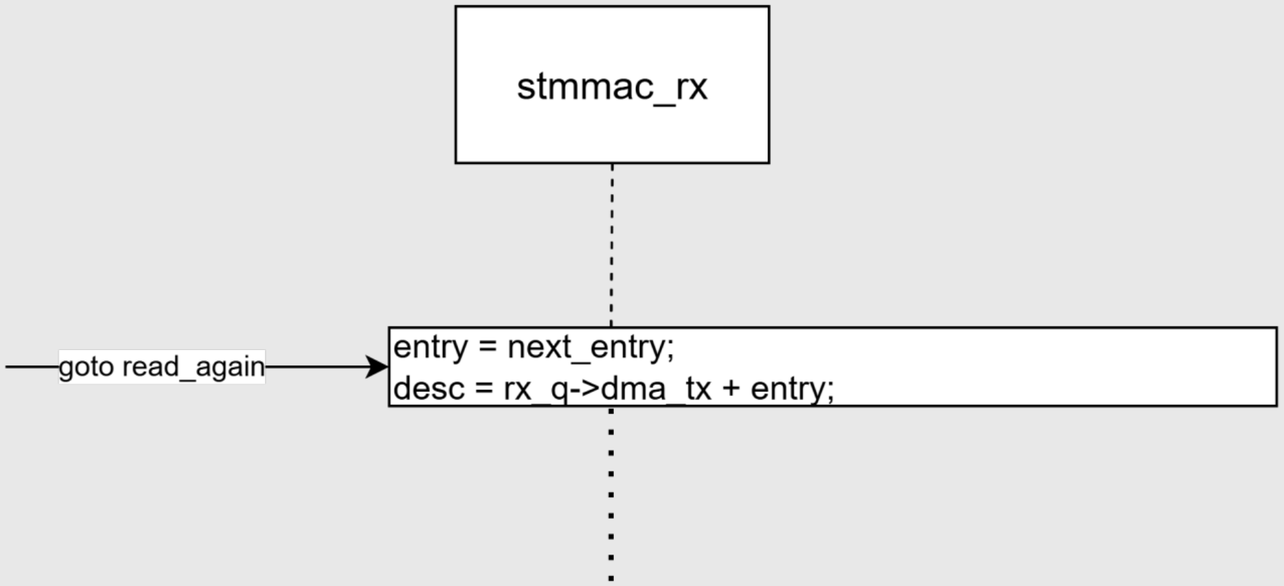
- 수신 핸들러 등록
  - probe과정에서 수신 napi poll 핸들러 등록 (stmmac_napi_poll_rx)
- stmmac_napi_poll_rx -> stmmac_rx
  1. DMA가 복사해놓은 데이터를 모아 프레임을 완성해나감.
  2. 프레임을 완성하면 상위 네트워크 계층에 전송

수신

3-2) stmmac_rx

03

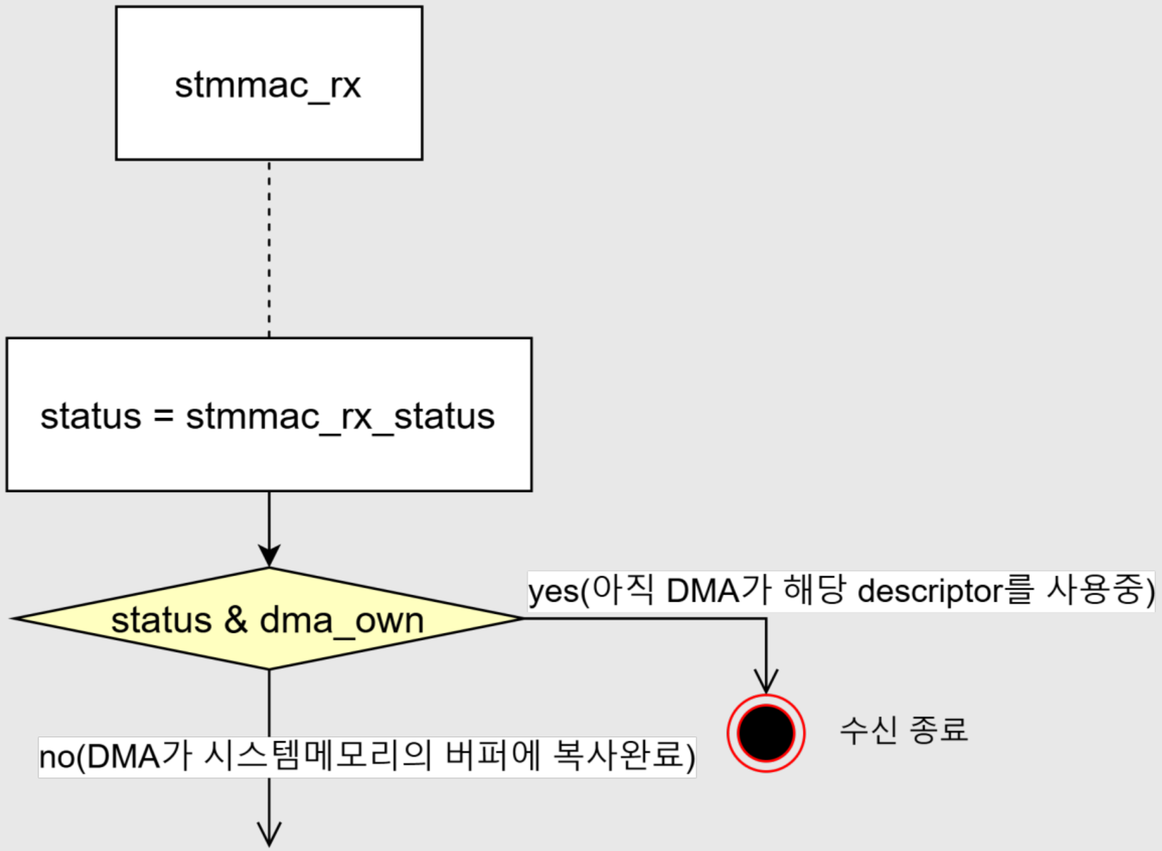
수신(3-2 stmmac_rx)



1. rx descriptor 갱신

*goto read_again? : RxFIFO로부터 데이터를 가져오는데, 아직 전체프레임을 완성하지 못했다?
-> 다시 이 지점으로 와서 다음 데이터를 수신

수신(3-2 stmmac_rx)

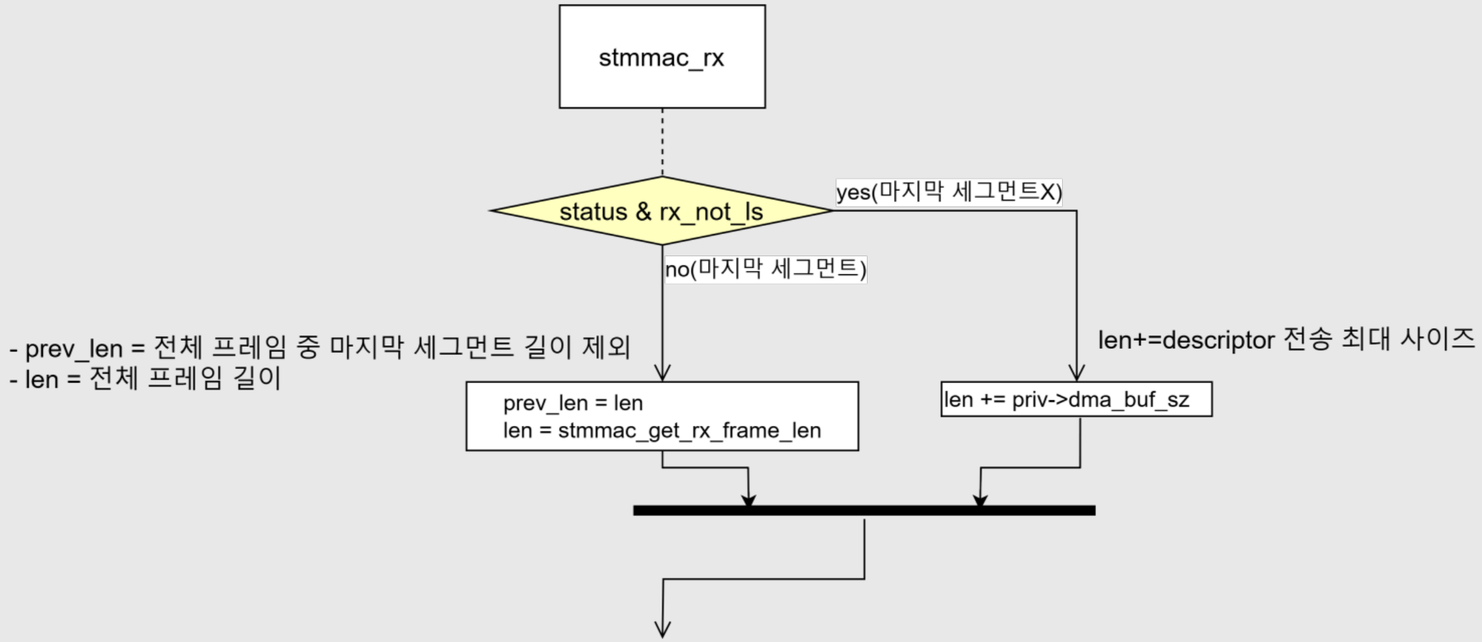


2. 현재 rx descriptor의 dma_own 플래그 check

set -> 더이상 수신을 진행할 수 없으므로 수신 종료 (break)

clear -> DMA가 버퍼로 데이터 복사를 완료. 이어서 진행

수신(3-2 stmmac_rx)



3. 수신한 프레임 길이를 갱신

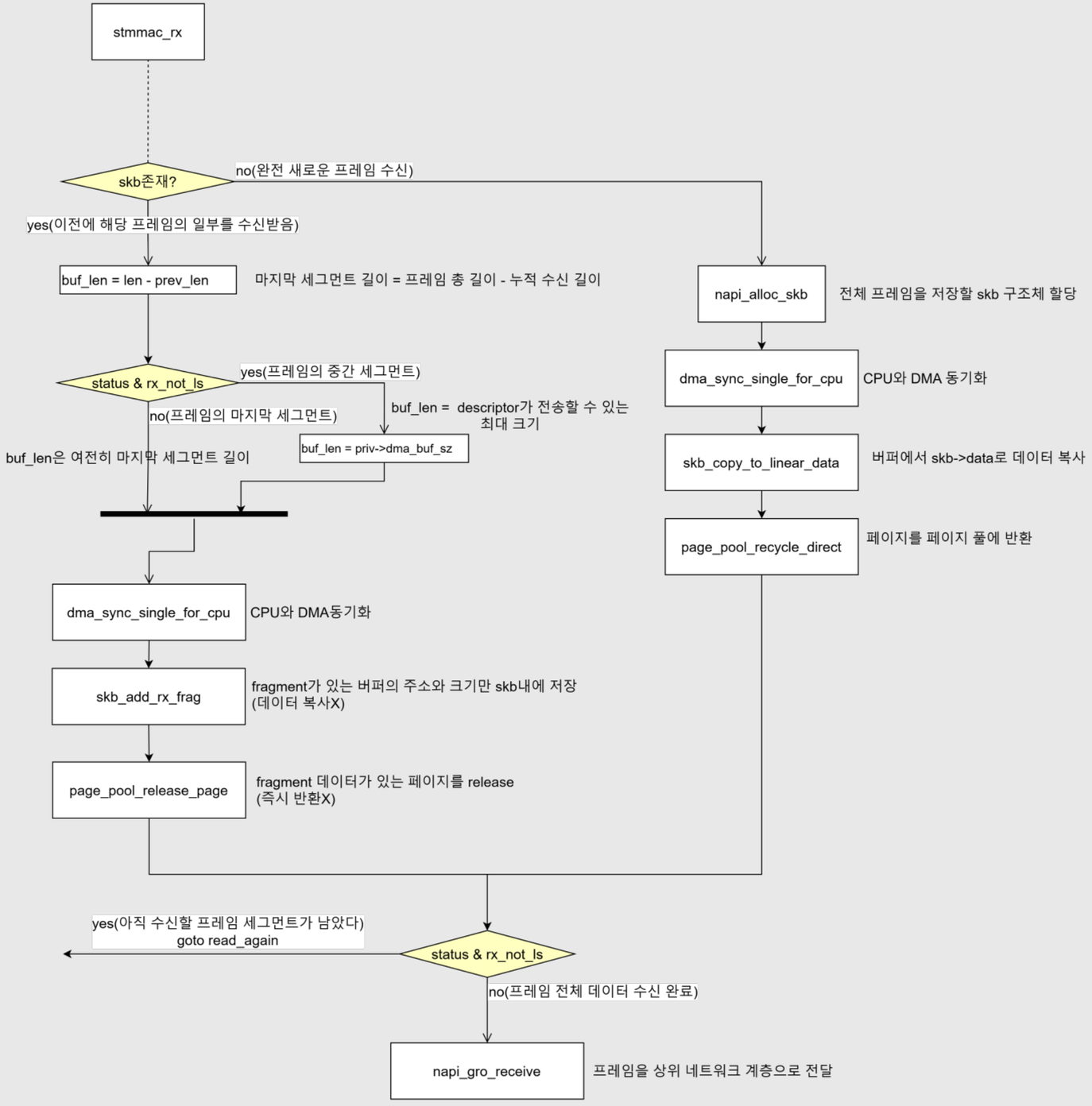
- 마지막 세그먼트 X :
  len += descriptor로 전송할 수 있는 최대 사이즈

- 마지막 세그먼트 O :
  len = 전체 프레임길이(프레임 완성)

수신(3-2 stmmac_rx)

4. DMA가 버퍼에 복사해둔 프레임 세그먼트를 가져와 프레임을 완성해간다.

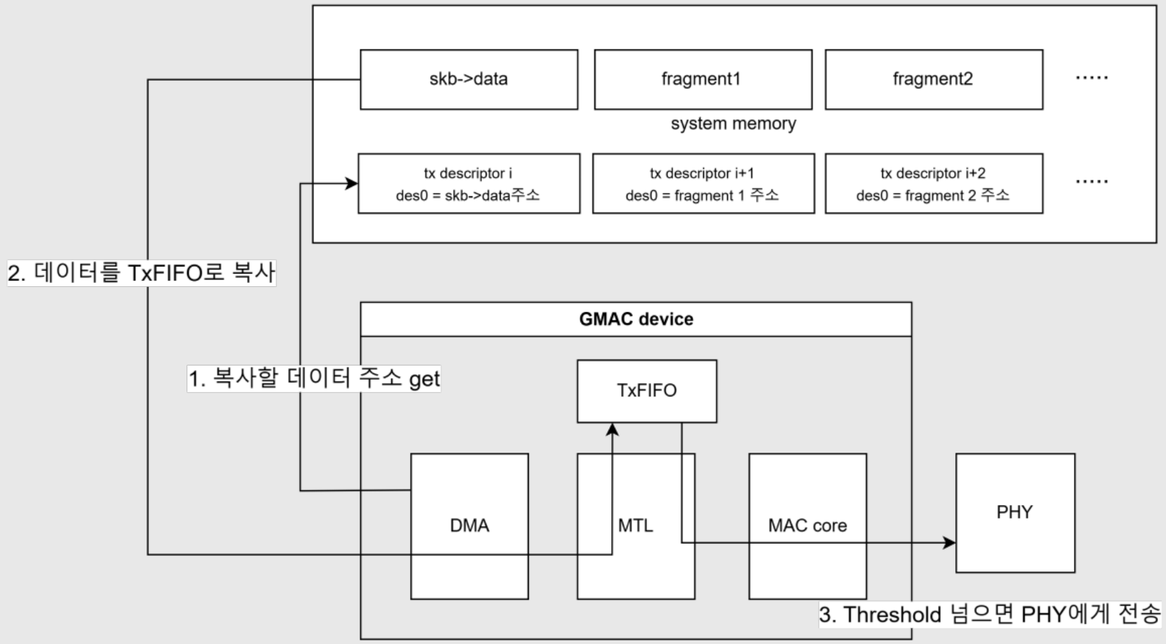
- 만약 전체 프레임을 완성했다면 상위네트워크계층으로 전달
- 아직 전체 프레임을 수신하지 못했다면 이어서 수신 진행



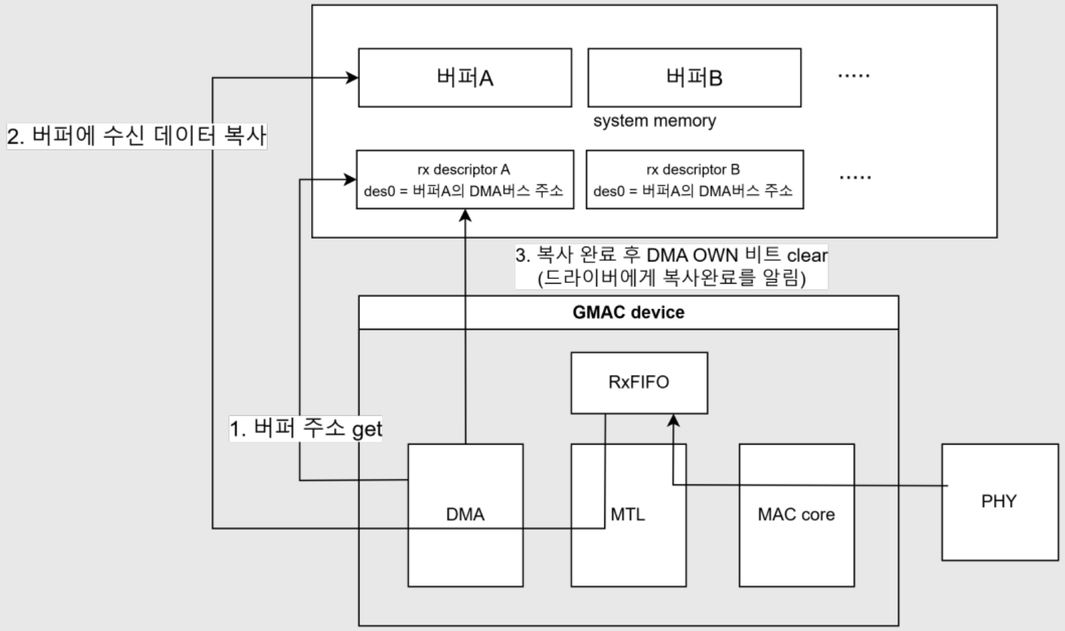
송수신에서 DMA 작동 방식

04

송수신에서 DMA 작동 방식



송신 시 DMA가 데이터를 GMAC으로 보내는 방법



수신 시 DMA가 데이터를 system memory로 넘기는 방법

End