

NDIS 미니포트 드라이버

Intel 82571 Case Study

○ 작성

- 2009년 11월 10일
- 양봉열 (xeraph@nchovy.com)

○ 목표

- NDIS 미니포트 드라이버의 구조 및 인텔 82571 동작 방식 이해

○ 라이선스

- 크리에이티브 커먼즈 저작자표시-비영리-변경금지 2.0 대한민국
<http://creativecommons.org/licenses/by-nc-nd/2.0/kr/>

○ 정의

- Network Driver Interface Specification
- 마이크로소프트와 3Com 합작으로 개발된 NIC용 API

○ 버전의 변천사

- NDIS 5.0: Windows 2000, 9x
- NDIS 5.1: Windows XP, Server 2003, CE
- NDIS 5.2: Windows Server 2003 SP2
- NDIS 6.0: Windows Vista
- NDIS 6.1: Windows Vista SP1, Server 2008
- NDIS 6.2: Windows 7, Server 2008 R2

○ 프로토콜 드라이버

- TCP/IP, IPX/SPX 등 전송 프로토콜 스택을 구현하는 네트워크 드라이버
- 상위 TDI 계층과 연결되어 있음
- 하드웨어 독립적
- WinPcap

○ 인터미디엇 드라이버

- 프로토콜 드라이버와 미니포트 드라이버 사이에 위치하는 드라이버
- 패킷을 임의로 가공하거나 버릴 수 있음
- 주로 패킷 필터링이나 로드밸런싱 등의 구현에 이용됨

○ 미니포트 드라이버

- NIC 하드웨어를 직접 제어하는 드라이버
- 데이터링크(MAC) 및 물리(PHY) 계층 전담

○ DriverEntry

- 모든 윈도우 드라이버의 진입점
- [NdisMInitializeWrapper](#): 초기화 과정에서 사용할 NDIS 핸들 획득
- [NdisMRegisterMiniport](#): NDIS 미니포트 드라이버 등록

NDIS_MINIPORT_CHARACTERISTICS 멤버	콜백 시그니처	기능
InitializeHandler	MiniportInitialize	NIC 활성화 시 호출됨
HaltHandler	MiniportHalt	NIC 비활성화 시 호출됨
QueryInformationHandler	MiniportQueryInformation	NIC 정보 조회 시 호출됨
SetInformationHandler	MiniportSetInformation	NIC 설정 시 호출됨
CheckForHangHandler	MiniportCheckForHang	NIC 정상 동작 여부를 주기적으로 호출하여 확인 (기본값 4초)
ResetHandler	MiniportReset	NIC 리셋 수행, Hang 발생 시 호출됨
ISRHandler	MiniportIsr	인터럽트 서비스 루틴
HandleInterruptHandler	MiniportHandleInterrupt	ISR에서 예약한 DPC 작업 수행

31		16 15		0		
Device ID		Vendor ID				00h
Status		Command				04h
Class Code			Revision ID			08h
BIST	Header Type	Lat. Timer	Cache Line S.			0Ch
Base Address Registers						10h 14h 18h 1Ch 20h 24h
Cardbus CIS Pointer						28h
Subsystem ID			Subsystem Vendor ID			2Ch
Expansion ROM Base Address						30h
Reserved				Cap. Pointer		34h
Reserved						38h
Max Lat.	Min Gnt.	Interrupt Pin	Interrupt Line			3Ch

○ 드라이버 초기화 (1)

MiniportInitialize

방향	타입	변수명	설명
출력	PNDIS_STATUS	OpenErrorStatus	장치 초기화 성공 여부
출력	PUINT	SelectedMediumIndex	선택한 매체 번호 (보통 802.3)
입력	PNDIS_MEDIUM	MediumArray	NDIS에서 전달하는 매체 목록
입력	UINT	MediumArraySize	매체 목록 길이
입력	NDIS_HANDLE	MiniportAdapterHandle	어댑터 핸들
입력	NDIS_HANDLE	WrapperConfigurationContext	초기화 과정에서만 사용되는 핸들

- Non-Paged 풀에서 메모리를 할당 받아 디바이스 컨텍스트 생성 (NDIS_HANDLE 저장)
- **NdisMSetAttributesEx** 함수를 이용하여 설정
 - 콜백 시 다시 넘겨줄 디바이스 컨텍스트 설정 (콜백으로 넘어옴)
 - NDIS_ATTRIBUTE_BUS_MASTER: DMA 전송에 관련된 버스 마스터 설정
 - NDIS_ATTRIBUTE_DESERIALIZE: 자체적으로 동기화 처리

○ 드라이버 초기화 (2)

- NIC와 관련된 레지스트리 설정 값 로딩 (WrapperConfigurationContext 핸들 이용)

함수	설명
NdisOpenConfiguration	NIC 드라이버의 레지스트리 설정에 대한 핸들 확보
NdisReadConfiguration	레지스트리 설정 조회
NdisCloseConfiguration	핸들 해제

- **NdisMInitializeScatterGatherDma**를 이용하여 DMA 초기화
 - 64비트 주소 지원 여부, 단일 DMA 전송 최대 크기 (패킷 최대 크기로 설정)
- PCI 설정 영역에서 하드웨어 정보 수집
 - **NdisReadPciSlotInformation**: PCI 설정 정보 조회
 - **NdisMQueryAdapterResources**: 포트, 메모리, 인터럽트 정보 수집
- NIC의 레지스터와 포트를 메모리에 매핑
 - **NdisMMapIoSpace**: 메모리 영역 매핑
 - **NdisMRegisterIoPortRange**: 포트 영역 매핑
- DMA용 공유 메모리 할당 (호스트 시스템과 NIC 사이의 공유)
 - **NdisMAllocateSharedMemory**: 할당 후 가상 주소와 DMA 물리 주소 쌍을 반환
- 마지막으로 인터럽트 등록 및 디바이스 처리 시작
 - **NdisMRegisterInterrupt**

○ 어댑터 정보 조회 처리

MiniportQueryInformation

방향	타입	변수명	설명
입력	NDIS_HANDLE	MiniportAdapterContext	어댑터 핸들
입력	NDIS_OID	Oid	조회할 정보 유형
입력	PVOID	InformationBuffer	버퍼
입력	ULONG	InformationBufferLength	버퍼 길이
출력	PULONG	BytesWritten	출력 길이
출력	PULONG	BytesNeeded	필요한 바이트 수

- [OID_GEN_SUPPORTED_LIST](#): 지원하는 OID 목록을 배열로 반환해야 함
- [OID_GEN_MEDIA_IN_USE](#): 사용 중인 물리 매체의 종류. 이더넷의 경우 NdisMedium802_3
- [OID_GEN_MAC_OPTIONS](#): 하드웨어 지원 기능 조회 (루프백 등)
- [OID_802_3_CURRENT_ADDRESS](#): 현재 MAC 주소 값 조회
- [OID_802_3_MAXIMUM_LIST_SIZE](#): 지원하는 최대 멀티캐스트 주소 수
- [OID_GEN_MEDIA_CONNECT_STATUS](#): 연결 상태 조회
- [OID_GEN_LINK_SPEED](#): 링크 연결 속도 조회 (100bps 단위)

○ 어댑터 비활성화

– MiniportHalt

- MiniportInitialize 콜백의 초기화와 반대 순서대로 자원 해제
- 인터럽트 비활성화
 - 레지스터를 제어하여 인터럽트 비활성화 처리
 - [NdisMDeregisterInterrupt](#): 인터럽트 연결 제거
- DMA 링 버퍼에 사용된 공유 메모리 해제
 - [NdisMFreeSharedMemory](#)
- 하드웨어 제어에 사용한 레지스터 및 포트 매핑 해제
 - [NdisMUnmapIoSpace](#)
 - [NdisMDeregisterIoPortRange](#)

○ 인터럽트 서비스

MiniportIsr

방향	타입	변수명	설명
출력	PBOOLEAN	InterruptRecognized	인터럽트 처리 여부
출력	PBOOLEAN	QueueMiniportHandleInterrupt	DPC 예약 여부
입력	NDIS_HANDLE	MiniportAdapterContext	어댑터 핸들

- DPC (Deferred Procedure Call)
 - 하드웨어 인터럽트는 커널을 선점하므로 최대한 ISR을 빨리 완료해야 함
 - 일반적으로 ISR에서는 신호 인식만 하고 나머지 작업을 DPC로 처리하도록 예약함
 - 리눅스의 경우 Bottom Half가 동일한 의미를 가지고 있음
- Spurious Interrupt
 - IRQ를 공유하게 되거나 다른 문제로 NIC와 관계없는 인터럽트가 발생하는 경우가 있음
 - 실제 NIC에서 발생시킨 인터럽트인지 제대로 확인하지 못하면 오동작하거나 성능 문제 발생
 - 따라서 반드시 자신의 인터럽트인지 확인 후 InterruptRecognized 설정 필요

○ DPC 수행

– MiniportHandleInterrupt

- 실질적인 인터럽트 처리를 수행

- 수신 디스크립터 링을 점검하여 완료 여부 확인

- DMA 전송이 완료된 패킷은 상위 스택으로 올려보내고 빈 버퍼로 교체

○ DeviceIoControl

– CTL_CODE 정의에 따라 메모리 버퍼 접근 방식이 다름

• Buffered I/O

- 유저모드 버퍼를 넘길 때마다 별도의 커널 버퍼에 복사함
- Irp->AssociatedIrp.SystemBuffer로 데이터 읽기/쓰기 가능
- Irp->IoStatus.Information에 읽기/쓰기 바이트 수 기록

• Direct I/O

- 응용 프로그램의 가상주소 버퍼에 대응하는 물리 메모리 페이지를 MDL로 제공
- MmGetSystemAddressForMdlSafe로 커널 가상주소를 얻을 수 있음
- 추가적인 메모리 복사 없고 직접적인 접근이 가능함

• Neither I/O

- I/O 관리자의 도움을 받지 않음
- 유저모드 버퍼를 유효성 검사 후 사용
- 페이지 아웃 되었을 수 있으므로 예외 처리 필요

○ Zero Copy

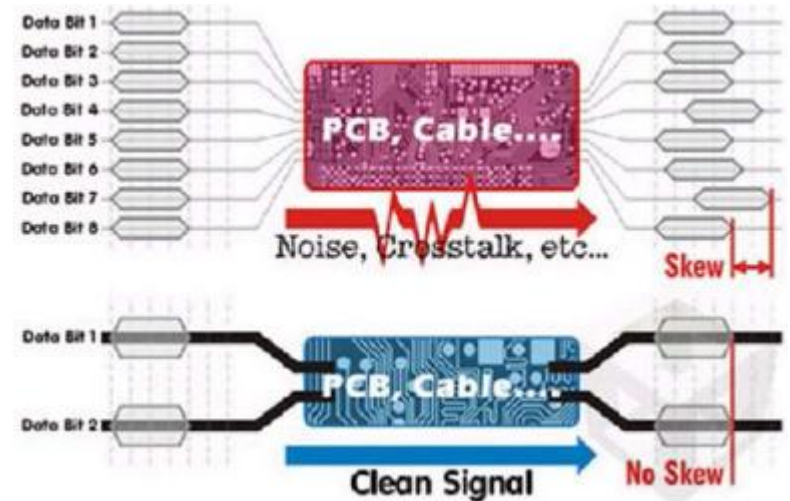
- 커널모드와 유저모드의 메모리 공유 및 DMA를 이용한 메모리 복사 제거
 - DMA를 위한 공유 메모리 버퍼를 먼저 생성
 - 특정 프로세스 컨텍스트에서 `MmMapLockedPagesSpecifyCache`를 호출하면 현재 프로세스에 매핑된 가상 주소를 얻을 수 있음
 - 패킷 포인터만 넘겨서 유저모드 응용프로그램으로 트래픽 펌핑 가능
- 호출 과정
 1. `NdisMAllocateSharedMemory`로 DMA 주소와 커널 가상 주소를 얻음
 2. `IoAllocateMdl`로 MDL 생성
 3. `MmBuildMdlForNonPagedPool` 호출하여 물리 주소 획득
 4. `MmMapLockedPages`를 이용하여 유저모드 매핑

○ PCI의 단점

- 병렬 버스 방식으로 인해 신호왜곡에 취약
- 다수의 기기를 동기화하려면 클럭과 전압에서 자유도가 떨어짐

○ PCI Express

- 직렬 전송 방식으로 라우팅 룰이 단순해지고 회로 단순화
- Transaction, Data Link, Physical 계층만 변경되므로 소프트웨어는 변경 불필요
- 양방향 최소 200MB/s에서 최대 6.4GB/s 대역폭 구현
 - x1: 100MB/s
 - x2: 200MB/s
 - x4: 400MB/s
 - x8: 800MB/s
 - x12: 1.2GB/s
 - x16: 1.6GB/s
 - x32: 3.2GB/s



○ 용어 정의

– 디스크립터

- 패킷 버퍼에 대한 부가 정보를 담은 메타데이터
- 버퍼가 위치한 물리 주소, 길이, 송신/수신 완료 여부, 체크섬 등 포함

– 버퍼

- 실제 패킷 데이터 기록

– MDI/MDIX: MediumDependent Interface (+ Crossover)

- MDI 1/2번 핀 (TX) – MDIX 1/2번 핀 연결 (RX)
- MDI 3/6번 핀 (RX) – MDIX 3/6번 핀 연결 (TX)



– SerDes

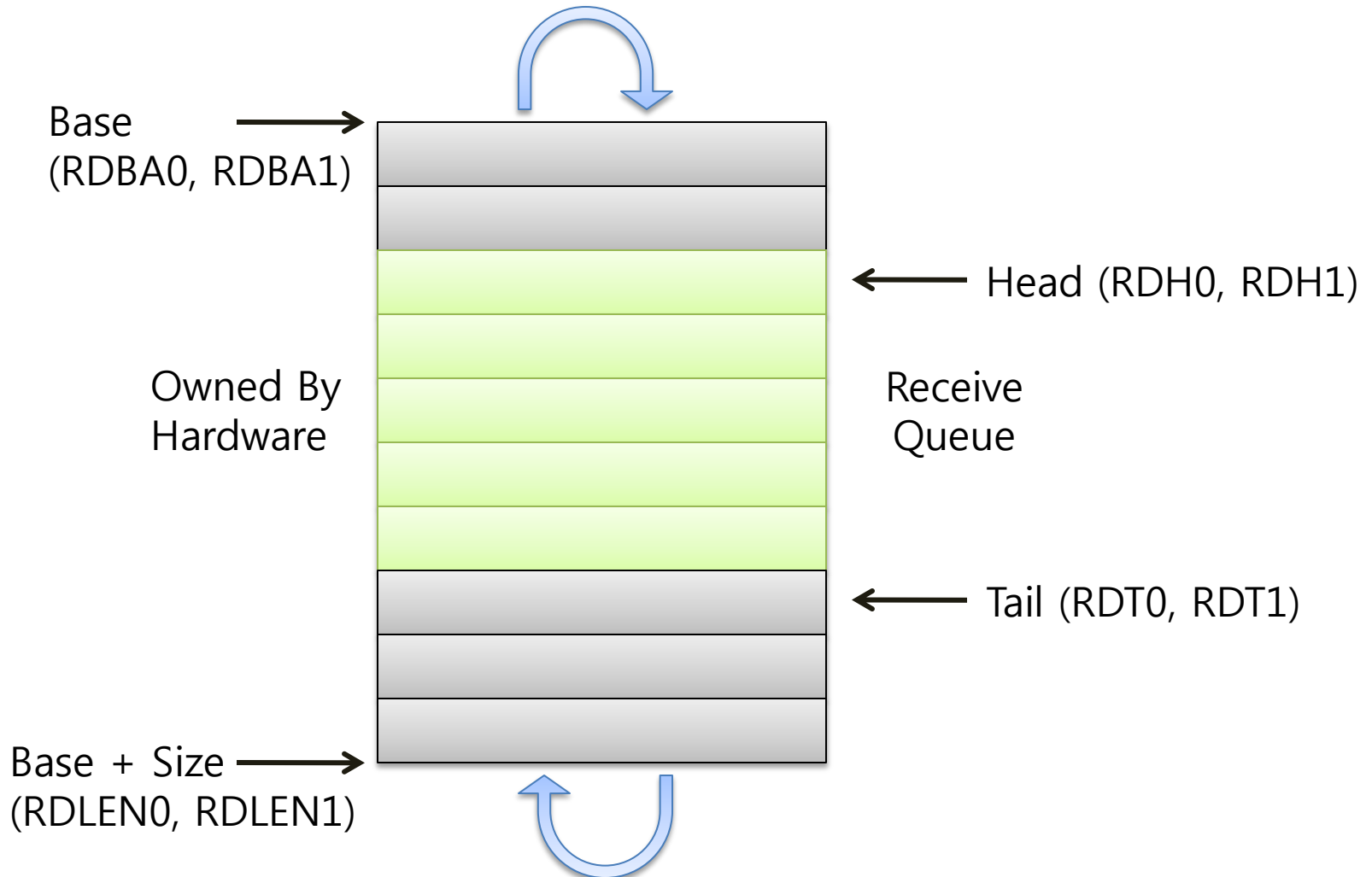
- 최대 클럭 속도보다 더 높은 시리얼 비트스트림 데이터 처리 모듈

– GBIC/SPF: 광 커넥터 모듈

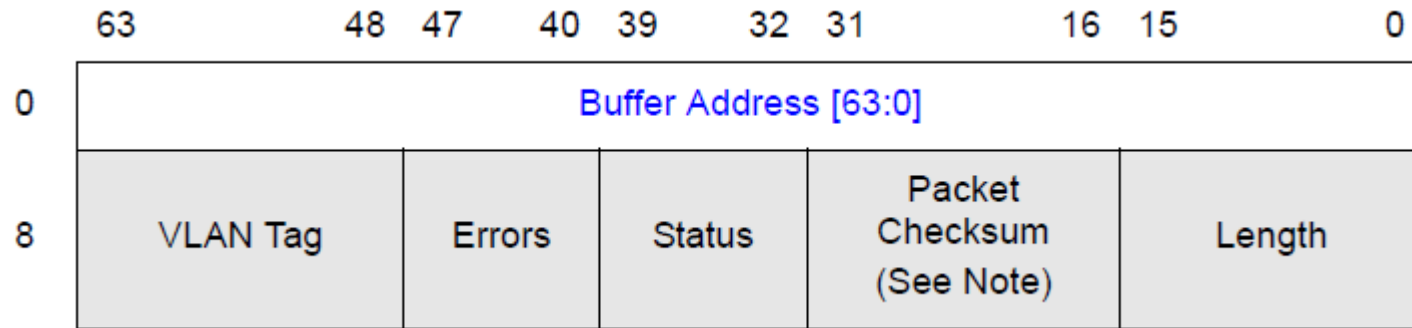
- GigaBit Interface Converter
- Small Form-factor Pluggable Transceiver



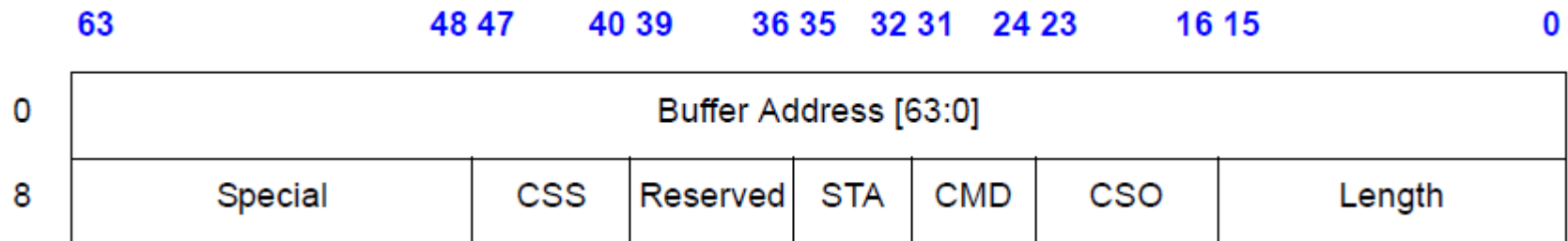
○ 디스크립터 큐 구조



○ 수신 디스크립터 레이아웃



○ 송신 디스크립터 레이아웃



○ 수신 처리

- 드라이버에서 디스크립터에 버퍼 물리 주소와 길이 입력
- 하드웨어에서 디스크립터를 보고 지정된 버퍼에 DMA로 전송
- DMA 전송이 완료되면 디스크립터에 완료 비트와 수신 패킷 길이 설정
 - 체크섬 오프로딩 계산 결과 기록, VLAN 태그, 각종 오류 여부 기록
- 하드웨어가 수신 인터럽트를 발생시키고 드라이버가 버퍼를 가져감

○ 송신 처리

- 상위 프로토콜 스택에서 전송할 패킷을 내려보냄
- 드라이버에서 프레임 확보 및 하드웨어 레지스터 제어
- 하드웨어에서 호스트 메모리의 패킷 데이터를 DMA로 복사
- 하드웨어가 인터럽트로 DMA 완료를 알리면 드라이버가 버퍼를 가져감

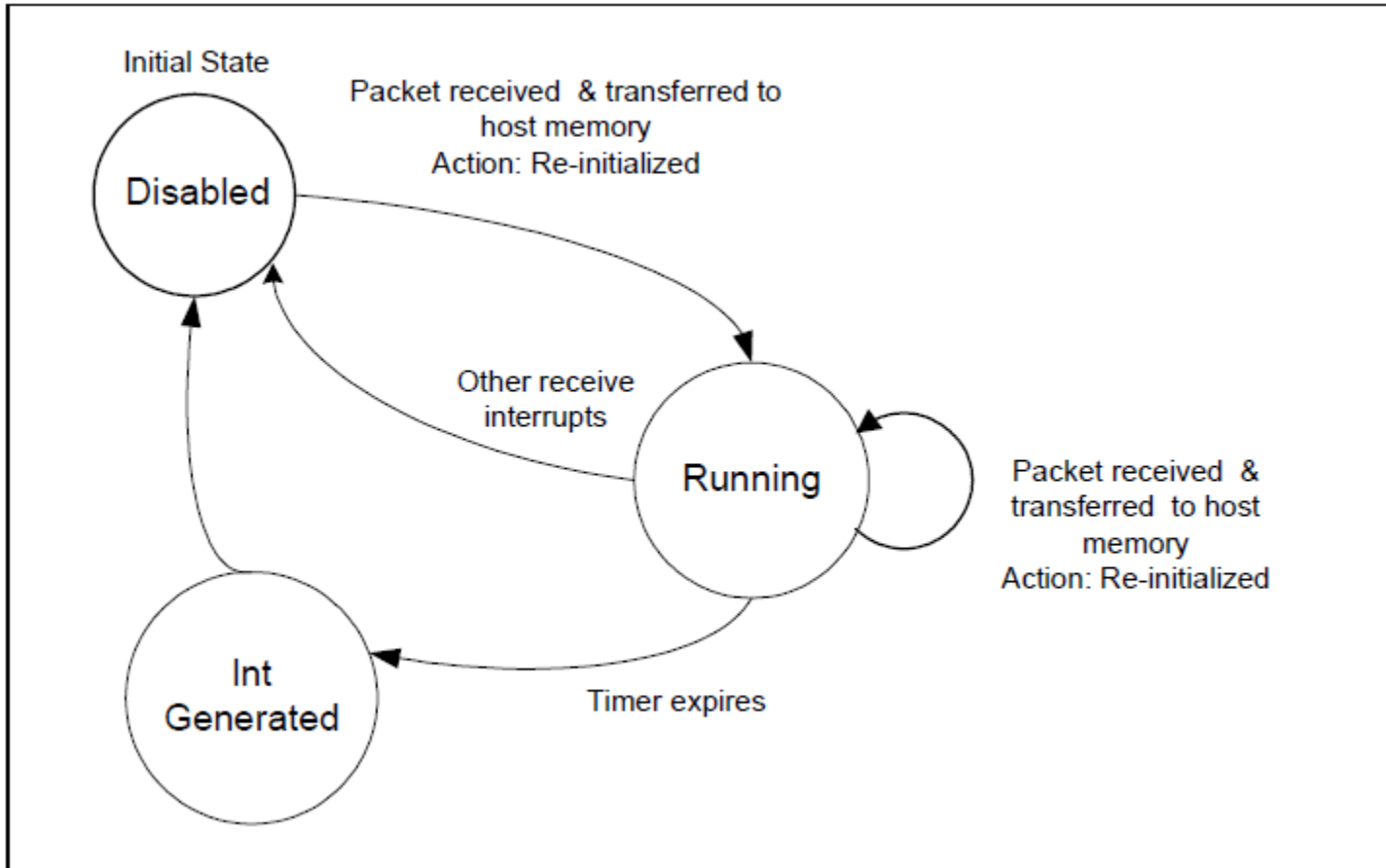
○ 수신 인터럽트 제어

- Receiver Timer Interrupt
- Small Receive Packet Detect
- Receive ACK Frame Detect
- Receive Descriptor Minimum Threshold
- Receiver FIFO Overrun

○ 송신 인터럽트 제어

- Transmit Queue Empty (TXQE)
- Descriptor done
- Transmit Delayed Interrupt
- Transmit Descriptor Ring Low Threshold Hit

○ 인터럽트 지연 처리 (상태 전이)



○ 인터럽트 지연 처리 (타이머)

