3장 비동기 직렬통신과 RS-232

IBM-PC에서 직렬 인터페이스에 대한 바이오스(BIOS) 차원의 지원은 아주 보잘 것 없다. 그렇기 때문에 MS-DOS에서 통신 프로그래밍을 작성할 때도 사실 바이오스의 직렬 인터페이스 기능은 거의 사용하지 않았다. 바이오스를 거치지 않고 직렬 인터페이스를 하려면 8250UART와 8259인터럽트 컨트롤러(최근에는 16650 UART가 사용되지만, 기본적인 구조는 8250UART에서 크게 달라지지 않았다)를 직접 제어해야 하는데, 윈도 프로그래밍에서는 통신 API들이 이런 저수준의 제어들은 모두 해 주기 때문에 프로그래머가 직접 이를 제어할 필요는 없지만, 실력있는 통신 프로그래머가 되려면 이 정도 기본 사항은 알아두는 것이 좋다.

직렬 통신 (Serial Communication)

자료를 전송할 때, 하나의 선을 통해 비트(bit)들을 하나씩 보내는 방법을 일반적으로 직렬통신이라고 한다. 사실 프로그래머의 입장에서 생각해 보면 PC에서의 통신이라는 것은 의외로 단순하다. 흔히들 통신하면 전화선으로 연결된 상대방 다른 컴퓨터와의 어떤 상호작용을 생각하기 쉬운데, 사실은 프로그래머가 신경써야 할 대부분은 자기가 사용하고 있는 PC내부에서 일어나고 있는 일들이다. 왜냐하면 실제로 통신이라고 하는 것은 직렬포트와 그 포트에 연결되어 있는 모뎀장치와의 통신이기 때문이다. 통신포트를 통해 내리는 모든 명령은 사실은 모뎀장치가 해석하고 그 응답을 보내오는 것이다. 모뎀장치는 말그대로 모듈레이션(modulation)-디모듈레이션(demoulation)/변조-복조 장치이기 때문에 PC의 디지틀 신호를 전화선을 통해 전달될 수 있는 아날로그 신호로 바꾸어 주거나 반대로 전화선을 통해 전달된 아날로그 신호를 디지틀 신호로 바꾸어 주는 역할을 할 뿐인 것이다. 따라서 직렬통신 프로그래밍이란 통신포트와 모뎀사이의 명령/응답 프로그래밍이라고 해도 지나친 말은 아닐 것이다.

병렬 전송(Parallel Transfer)과 직렬 전송(Serial Transfer)

병렬 전송은 8비트 한 바이트를 한꺼번에 보내는 것으로 PC에서는 프린터로 데이터를 보내기 위해 이 병렬 전송을 사용한다. 한비트씩 보내는 것이 아니라 한바이트씩 보내는 것이므로 직렬 전송에 비해서는 속도가 빠르지만, 이렇게 병렬전송을 하려면 8가닥 이상의 전송선이 필요하게 되므로, 경제적인 이유로 해서 직렬통신이 더 널리 쓰인다.

직렬 전송은 한비트 씩 보내는 것으로, 송신하는 쪽에서는 바이트를 연속적인 비트열로 변환하여 통신 회선을 통해 한비트씩 송신하며, 수신하는 쪽에서는 연속적인 비트열을 받아 다시 바이트로 변환한다. 그런데 이러한 비트열-바이트, 바이트-비트열 변환작업은 CPU를 많이 잡아먹는 일이기 때문에 PC에서는 별도의 하드웨어를 통해 이러한 작업을 수행하도록 했는데, 8250 UART가 바로 그것이다.

전이중(Full Duplex)과 반이중(Half Duplex)

데이터를 전송할 때, 한쪽은 일방적으로 받기만 할 수 있고, 다른 한쪽은 일방적으로 보내기만 할 수 있는 경우가 있을 수 있다. 이런 것을 단방향(Simplex) 통신이라 한다. 또한 양쪽이 다 보내기 받기를 할 수 있는 경우도 있을 것이다. 이런 것은 이중(Duplex)통신이라고 한다. 이 경우는 다시, 한쪽이 보내고 있을 때는 다른 한쪽에서는 받기만 할 수 있는 경우와 한쪽이 보내고 있을 때도 다른 한쪽에서 받기는 물론 동시에 보내기를 할 수 있는 경우로 나눌 수 있다. 양쪽에서 동시에 보내고 받기를 할 수 있는 것을 전이중(Full Duplex)통신이라 하고, 한쪽에서 보낼 때는 다른 한쪽에서는 받기밖에 할 수 없는 형태를 반이중(Half Duplex)통신이라고 한다. 전이중 통신의 좋은 예는 일반 전화 통화이다. 양쪽에서 동시에 말해도 모두 전달되기 때문이다. 반이중 통신의 가장 좋은 예는 무전기이다. 무전기는 한사람이 데이터를 보내고 있을 때는 다른 사람이 받기밖에 할 수 없기 때문에 한쪽 사람이 말을 마칠 때는 ".... 이상"이라는 말을 붙여줌으로써 말이 끝났음을 알리고, 말을 마친 사람은 수신 상태로 돌아가고, 다른 한쪽 사람은 수신 상태에서 송신상태로 전환한 다음, 자신의 말을 해야 한다.

요즘 인터넷 폰이 값싼 해외통화 수단으로 인기를 얻고 있는데, 인터넷 폰의 선전문

구에서도 전이중(Full Duplex)이란 말을 찾아 볼 수 있을 것이다. 인터넷 폰 사용할 때 자신의 소리(sound)카드가 전이중 통신을 지원하는지 확인해 보아야 한다.

동기와 비동기

비트열을 직렬 전송방식으로 전달할 때, 받는 쪽에서는 어떤 식으로 다시 바이트로 조합해 낼 것이다. 이때 수신하는 쪽에서 조합한 하나의 바이트가 생길 때 마다 송신한 쪽과 어떤 정보를 주고 받음으로써 송수신이 제대로 이루어졌음을 아는 방식이 바로 동기 방식이다. 송신하는 쪽에서 바이트의 시작과 끝을 지시해 주는 비트를 추가해서 보냄으로써 수신하는 쪽에서는 송신하는 쪽과는 수신데이타만을 갖고 바이트로 복원하는 방식이 바로 비동기 방식이다.

비동기 직렬 통신

송신할 때 시작과 끝을 알리는 비트를 추가해서 데이터를 보냄으로써 송수신 장치 사이의 긴밀한 통신이 필요하지 않기 때문에 송수신 장치를 싼 가격으로 만들 수 있어서 많이 사용하는 방식이다.

시작 비트

통신회선의 상태는 아무 일도 일어나지 않는 정지상태가 논리적으로 1인 상태이다. 시작비트는 논리값 0인데, 정지상태의 1에서 0으로 떨어짐으로써 비트가 시작하고 있음을 알리는 것이다.

데이터 비트

시작비트에 이어서 실제 데이터가 전송되게 되는데, 이 데이터 비트는 5비트, 6비트, 7비트, 8비트일 수 있으며, 5,6비트는 거의 사용되지 않고, 7,8비트가 사용되는데, 데이터 비트에 사용하는 문자 코드는 ASCII이다. 한글을 사용하는 국내 PC 통신업체들은 모두 8비트의 데이터 비트를 사용한다. 왜냐하면, 7비트 데이터 비트를 사용하면 0 ~ 127의 아스키를 사용하는데, 이는 주로, 미국내의 정보표현을 위한 것으로, 영어 알파벳 표현만을 염두에 둔 것이다. 한글의 경우에는 0 ~ 255사이의 아스키가 모두 사용

되기 때문에 국내 PC통신 업체 접속 때에는 데이터 비트를 8로 해야 한다.

패리티 비트

패리티 비트는 오류 검출을 위한 비트이다. 짝수 패리티(Even Parity)는 데이터 비트 내의 1의 수가 짝수이면 0, 홀수이면 1이 된다. 홀수 패리티(Odd Parity)는 데이터 비트 내의 1의 수가 홀수이면 0, 짝수이면 1이 된다. 눈치가 빠른 사람이라면 금방 알아 차렸겠지만, 패리티를 통한 오류 검출에는 중대한 결점이 있다. 한 바이트 내에서 오류가 두 번 일어나게 되면 짝수 패리티든, 홀수 패리티든 모두 오류가 없는 것으로 판단해 버리기 때문이다. 하지만, 이런 일은 자주 일어나는 것이 아니기 때문에 패리티만으로도 오류 확률을 크게 줄일 수 있다. 하지만, 데이터 비트가 8비트인 경우에는 패리티 비트는 사용할 수 없다.

짝수 패리티와 홀수 패리티 외에도 공백(space) 패리티, 마크(mark) 패리티, 패리티 없음 등이 있는데, 공백 패리티는 패리티 비트가 항상 0이며, 마크 패리티는 패리티 비트가 항상 1, 패리티 없음은 아예 패리티 비트가 없다는 의미이다. 공백 패리티와 마크 패리티는 거의 사용되지 않고, 국내 PC 통신 업체에 접속할 때는 데이터 비트를 8비트로 사용해야 하기 때문에 자연, 패리티는 사용되지 않는다.

정지 비트

패리티 비트 다음에 오게 되는 비트로서 하나의 바이트가 완전히 전송되었음으로 알린다. 정지비트는 1, 1.5 또는 2비트가 있는데, 110 보(baud) 레이트 이하인 경우에는 정지비트 2를 사용하고 300 보 레이트 이상인 경우에는 정지비트 1을 사용하게 되기 때문에 거의 모든 경우에 정지비트는 1로 설정한다고 보면 되겠다.

통신 에뮬레이터 프로그램을 보면 데이터 비트, 패리티 비트와 정지 비트를 설정해 주도록 되어 있는데, 선택할 수 있는 것이 여러개이다 보니 어떤 것을 설정해야 할 지 잘 모르는 경우가 있다. 하지만, 거의 대부분의 국내 PC통신업체에서는 데이터비트 8, 패리티 없음, 정지비트 1을 사용하므로, 일단 이렇게 설정해서 사용하면 문제가 없을 것이다. 외국의 경우에는 데이터 비트로 7비트를 사용하는 것이 대부분이고 패리티 비트는 짝수 패리티, 정지비트로는 1비트를 사용한다.

직렬통신용 하드웨어

가장 중요한 직렬통신포트는 UART(범용 비동기 수신 전송 장치:Universal Asynchronous Receiver/Transmitter)이다. PC에서 원래 사용하던 포트는 8250 UART인데, AT이후부터는 16540 UART를 사용하고 있다. 하지만 최근의 대부분의 직렬통신 포트 장치는 16650 UART를 사용한다. 이러한 직렬 포트들은 RS-232 표준을 사용해서 다른 직렬통신 포트와 통신을 한다.

UART(Universal Asynchronous Reciever Transmitter)

직렬 비트 열로 데이터를 쓰는 일은 사실 PC입장에서 보면 아주 피곤한 일이다. 각 비트들을 출력 비트 열에 넣기 위해 위치로 이동시켜야 하고 이런 일은 모든 비트를 다 쓸 때까지 계속해야 한다. 입력되는 비트들을 읽어내는 일은 더더욱 어려운 일이다. 입력 열은 아주 높은 비율로 샘플링 되어야 한다. 프로세서는 시작 비트가 잡음으로 인해 잘못되지 않았는지, 등등을 계속 살펴보고 있어야 한다. 프로세서가 이렇게 직렬통신에 너무 많은 시간을 소비하면 다른 작업을 할 수 없기 때문에 모든 PC들은 UART, 즉 범용 비동기 송수신기 칩(chip)을 통해 직렬 자료 전송을 수행한다. 사용자 입장에서 보면 UART는 다음의 두가지 일을 하는 일종의 블랙박스이다. 첫째, 바이트들을 RS-232C선을 통해 직렬 데이터로 변환하는 일과 둘째, 직렬 데이터를 읽어 컴퓨터가 읽을 수 있도록 바이트 형태로 변환하는 작업이다.

PC안에는 수많은 서로 다른 제작자가 만든 수많은 서로 다른 칩들이 존재하지만 8250만은 그렇지 않다. 8250은 완전히 원래의 8250과 호환되는 것만이 존재하는데 이렇게 된 데에는 다음과 같은 전설이 내려오고 있다.

IBM과 마이크로소프트가 최초의 IBM-PC의 바이오스를 설계할 당시에 직렬 포트에서 들어오는 입력에 대해 인터럽트 기반의 방식을 지원하지 않았다. 단지 폴링 방식(polled-mode)의 입출력만을 지원했는데, 폴링 방식의 출력은 초기의 직렬 프린터에서는 잘 동작했다. 300 보(baud) 레이트 정도의 속도의 터미널 에뮬레이터에서는 잘 동작했지만, 곧 1200 보(baud) 모뎀이 싼 가격에 나오게 됨으로써 문제가 되기 시작한 것이다. 통신 프로그램이 실제 보 레이트의 속도로 동작하도록 프로그래밍하려면 8250 UART에 대한 인터럽트 기반의 드라이버를 작성해야만 하게 된 것이다. 이것은 프로그래머가 레지스터 수준으로 칩을 프로그래밍해야 한다는 것으로 응용프로그램이 특정한 하드웨어에서만 동작하게 된다는 것을 의미하는 것이었다. 만약 IBM과 마이크로소프트가 인터러브 기반의 디바이스 드라이버를 제공했더라면 문제는 크게 달라졌을 것이다. 그렇게 되었더라면, 하드웨어 제작자들은 어떤 UART든지 선택해서 사용할 수 있었을 것이며 그것에 대한 MS-DOS 디바이스 드라이버만 제공하면, 응용프로그램들은 곧바로 문제없이 수행될 수 있었을 것이다.

이런 이유로 해서, 어떠한 새로운 직렬 하드웨어라도 8250 UART를 제어하기 위해 설계된 명령에 제대로 응답해야만 했다. 직렬 통신 하드웨어가 개선되기는 하였지만 그건 단지 초기의 원래 PC와 호환되도록만 개선되게 된 것이다.

송수신기에 16바이트의 내부 버퍼를 가진 16550 UART가 나오긴 했지만 기능적으로는 8250과 똑같다.

8250 범용 비동기 송수신기(UART)

8250 UART는 RS-232C 비동기 통신 어뎁터를 관리하는 IC이다. RS-232C를 통해 입출력되는 모든 정보는 이 8250 UART의 간섭을 받는데, 8250 UART는 그 내부의 레지스터를 통해 자료를 송수신하거나 관련 정보를 제공한다. 원래의 IBM-PC에는 직렬 포트에 대해 두 개의 인터럽트 라인만을 할당해서, 실제로 사용가능한 것은 COM1과 COM2로 제한된다. 8250은 다음 그림과 같은 비동기 출력 비트열을 만들어 낸다. 각 출력 바이트는 시작 비트로 시작하고 5,6,7또는 8비트의 데이터 비트가 이어지며 1, 1.5 또는 2의 정지비트가 그 뒤를 따른다. 마지막으로 패리티 비트가 붙을 수 있는데 패리티 비트는 EVEN, ODD, MARK, SPACE 중 하나이다. 보통 PC기반의 통신 응용 프로그램에서는 7이나 8을 데이터 비트로 쓰고, EVEN, ODD또는 NO 패리티, 그리고 하나의 정지비트를 쓰는 것이 일반적이다.

8250은 보 레이트(baud-rate) 발생기를 내장하고 있는데, 이것은 입력 클럭을 정수로 나눈 보 레이트를 발생시킨다. IBM-PC에서 보 레이트 발생기는 초당 115,200번 진동하는 수정 발진자를 사용한다. 보 레이트 발생기는 이 출력을 16비트 정수로 나눠어 1에서 65536사이의 값을 갖는다. 8250의 특성 때문에 입출력 보 레이트는 같아야 한다.

번호	COM1	COM2	이름
0	3F8	2F8	데이터 레지스터 (THR : Transmit Holding Register) (RBR : Receive Buffer Register)
1	3F9	2F9	인터럽트 인가 레지스터 (IER : Interrupt Enable Register)
2	3FA	2FA	인터럽트 식별 레지스터 (IIR : Interrupt Identification Register)
3	3FB	2FB	선로 제어 레지스터 (LCR : Line Control Register)
4	3FC	2FC	모뎀 제어 레지스터 (MCR : Modem Control Register)
5	3FD	2FD	선로 상태 레지스터 (LSR : Line Status Register)
6	3FE	2FE	모뎀 상태 레지스터 (MSR : Modem Status Register)

데이터 레지스터(THR/RBR:Transmit Holding Register, Receive Buffer Register)

데이터 레지스터는 말 그대로 데이터의 입출력에 이용되는 레지스터이다. 이 레지스터는 입력 출력 두 경우에 모두 사용되는데, 하나의 문자를 받거나 보내기 위해 이를 잠시 보관하는 역할을 한다. 모뎀을 통해 데이터를 주고 받는 다는 것은 이 레지스터에 값을 써넣거나 읽어내는 것을 말하는 것이다. 데이터 레지스터는 입출력 모두에서 사용되기 때문에 출력에 사용할 때는 THR, 입력에 사용할 때는 RBR이라고 불리기도 한다. 이처럼 하나의 레지스터를 입출력 모두에서 사용하기 때문에 입력이나 출력을 할 때, 현재 다른 작업이 진행중인지를 먼저 검사해야 한다. 예를 들어, 문자를 전송하려면 먼저 선로 상태 레지스터를 읽어 준비가 된 경우에만 보내도록 한다.

인터럽트 인가 레지스터(IER : Interrupt Enable Register)

직렬 입출력이 인터럽트 구동 방식으로 처리될 수 있도록 인터럽트가 발생하는 조건을 설정할 때 사용하는 레지스터이다. 인터럽트 구동방식이란 입력 포트에 어떤 변화가 일어나면 이를 인터럽트를 발생시킴으로써 알리는 방식이다. 인터럽트 구동방식을 사용하지 않는다면, 계속해서 포트를 감시하고 있어야 하며, 또한 혹시 약간 시간이 걸리는 다른 일을 하는 동안 입력 포트에 새로운 문자가 들어오게 되면, 이를 잃어버리

게 될 수도 있게 된다.

다음 표는 인터럽트가 발생하는 조건에 대한 레지스터 값이다.

IER 비트	발생하는 인터럽트
비트 0	수신 데이터 있음
비트 1	송신 준비
비트 2	선로 상태 변경
비트 3	모뎀 상태 변경
비트 4 ~ 7	사용 안됨

인터럽트 식별 레지스터(IIR : Interrupt Identification Register)

IER에 의해 정해진 조건에 따라 인터럽트가 발생했을 때, 실제 발생한 인터럽트의 종류를 알려주는 레지스터이다. 인터럽트 처리기에서는 이 IIR을 보고 적당한 인터럽트에 대한 처리를 해주면 된다.

IIR 비트	의미
비트 0	현재, 인터럽트 처리 중인지 여부
비트 1 ~ 2	인터럽트 종류 00: 수신 선로 상태 변경 01: 송신기 상태 변경 10: 수신 데이터 있음 11: 모뎀 상태 변경
비트 3 ~ 7	사용안됨

선로 제어 레지스터(LCR : Line Control Register)

데이터 비트, 정지 비트, 패리티 비트, 속도 등을 제어하는 레지스터이다.

LCR비트	기능	비고
비트 0 ~ 1	데이터 비트	00: 5비트 01: 6비트 10: 7비트 11: 8비트
비트 2	정지 비트	0: 1비트 1: 2비트
비트 3 ~ 4	패리티 비트	00: 없음(NONE) 01: 홀수(ODD) 02: 짝수(EVEN)
비트 5	stick parity	
비트 6	set break	
비트 7	Divisor Latch Access Bit	

모뎀 제어 레지스터(MCR : Modem Control Register)

Data Terminal Ready(DTR)와 Request To Send(RTS) 신호를 관리한다.

MCR 비트	기능
비트 0	DTR(Data Terminal Ready)
비트 1	RTS(Request To Send)
비트 2	OUT1: 사용자 정의 인터럽트 요청
비트 3	OUT2: 인터럽트 가능
비트 4	Loop Test
비트 5 ~ 7	사용 안됨

선로 상태 레지스터(LSR : Line Status Register)

직렬데이타가 하나의 바이트 형태로 잘 변환되는지 등에 대한 여러 가지 상태 정보를 갖는다. 비트열이 연속되다가 하나의 문자가 완성되어 입력 포트에 들어가면 비트 0이 1로 바뀐다.

LCR 비트	기능
비트 0	수신 데이터 있음(Data Ready)
비트 1	수신 데이터 넘침(Overrun Error)
비트 2	패리티 비트 오류(Parity Error)
비트 3	정지비트 오류(Framing Error)
비트 4	브레이크 신호(Received Break)
비트 5	THR Empty
비트 6	TSR Empty
비트 7	사용 안함

모뎀 상태 레지스터(MSR : Modem Status Register)

모뎀과의 연결상태나 모뎀의 설정상태등을 알아낼 수 있다.

MCR 비트	모뎀 상태
비트 0	pin 5
비트 1	pin 20
비트 2	pin 22
비트 3	pin 8
비트 4	Clear To Send(CTS)
비트 5	Data Set Ready(DSR)
비트 6	링 검출(Ring Indicator)
비트 7	캐리어 검출(Carrier Detect)

8250/16450 UART

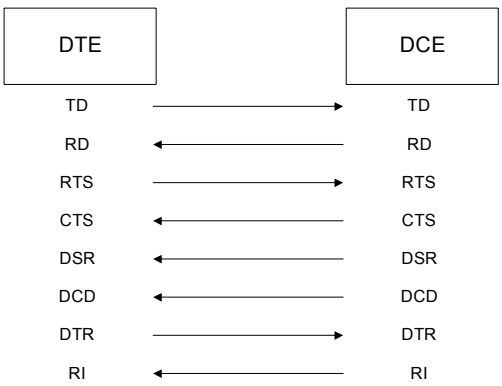
8250 UART은 IBM PC의 표준 직렬 포트 장치이다. 16450 UART는 AT부터 채용되었다. 이 둘은 하드웨어와 소프트웨어 모두에서 호환 가능하다. 16650 UART는 IBM의 PS/2에서 도입되었는데 최근의 대부분의 내장형 모뎀이나 직렬 포트 카드에는 16650 UART을 사용한다. 16650 UART는 8250 UART와 소프트웨어적으로 호환되며 8250 UART보다 확장된 기능을 가지고 있다. 16650 UART이 8250/16450UART보다 좋아진 것 중 가장 중요한 점은 데이타 라인의 전송, 수신에 16바이트 버퍼를 추가했다는 점이다. 윈도우즈 환경에서 통신 에뮬레이터 같은 프로그램을 작성하는 경우에는 UART 내부의 레지스터 이름이나 그 의미 같은 것은 사실 알 필요가 없다. 그냥 그 이

름 정도는 알아두고, 요즘 사용되는 UART가 이전의 것들보다는 오류가 생길 확률이 더 줄어든 성능 좋은 것이라는 것만 알면 된다. 그리고 바로 이 UART라는 것이 8비트 데이타를 받아서, 8개의 비트 스트림으로 변환하고, 거기에 시작비트와 정지비트를 추가하는 기능을 한다는 정도만 알면된다. 이 직렬 포트 장치에도 여러가지 레지스터들이 존재한다. DOS용 통신 프로그램을 개발하고 있다면 이 모든 것을 자세히 알고 넘어 가야 하겠지만 윈도95용 통신 프로그램 작성에는 사실 몰라도 상관 없다. 왜냐하면 윈도에서는 이런 하드웨어 제어 부분은 윈도의 API 내부에서 이루어 지기 때문이다. DOS용 통신 프로그램을 작성하려면 통신 포트의 속도와 데이타의 크기, 패리티 비트, 정지비트의 크기 등을 지정하려면 해당 통신 포트의 주소에 직접 값을 써 주어야 했지만, 윈도 프로그래머라면 더 이상 그럴 필요가 없는 것이다.

RS-232

RS-232표준은 미국 전자 산업 협회(EIA:Electronic Industries Association)에서 제정되었다. RS-232표준 중에 가장 잘 알려진 표준이 바로 RS-232C이다. 이 표준은 데이타 단말 장치(DTE:Data Terminal Equiment)와 데이타 통신 장치(DCE:Data Communication Equipment) 사이의 인터페이스에 대해 정의해 놓은 것이다. 현재 가장 많이 사용되는 RS-232C용 커넥터는 9핀과 25핀이 있는데, 이렇게 핀 수가 다른 커넥터가 존재하는 것은 이 RS-232표준이 사실상 이런 부분에 대해서는 언급이 없었기 때문이다. PC에서는 데이타 단말 장치가 바로 직렬포트이며 데이타 통신 장치가 모뎀에 대응된다. RS-232C에 대해 자세히 다루는 것은 이 책의 범위를 벗어나는 것일 뿐만 아니라 몇가지만 제외한다면 그 자세한 내용은 몰라도 된다. 하지만 다음에 다루는 약간(?)의 용어들과 그 의미를 파악하는 것 정도는 필수이다.

다음은 DTE(PC의 직렬포트)와 DCE(모뎀)의 연결 방식을 보인 것이다.

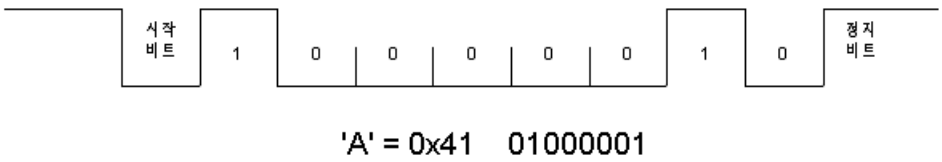


DTE와 DCE라는 용어가 무척 생소하게 느껴지는 사람이 대부분일테니 여기서는 DTE는 PC직렬포트, DCE는 모뎀이라는 용어를 사용해서 설명하겠다.

데이타 라인

피씨 직렬포트와 모뎀 사이의 데이타 전송은 두개의 RS-232 통신 라인 TD와 RD를 통해 이루어 진다. TD 라인은 PC쪽에서 오는 신호를 모뎀 쪽에 전달하는데 사용되고 RD 라인은 모뎀쪽의 신호를 피씨쪽에 전달하는데 사용된다.

다음은 RS-232를 통해 글자가 전달되는 예이다.



'A' = 0x41 01000001

제어 라인

RTS/CTS(Request To Send/Clear To Send)

주로 하드웨어 핸드쉐이킹에 사용한다. 핸드쉐이킹이란 말 뜻 그대로 '악수'이다. 통신에서 데이타를 전송하려고 할 때, 상대방의 상태가 데이타를 전송받을 수 있는 상태인지를 확인 한 후 데이타를 보내야 한다. 이 때, 상대방의 상태가 데이타를 전송 받을 수 있는 상태인지 알려면 먼저, 전송하려는 쪽에서 "그쪽... 말이죠 제가 데이타를 전송하려고 하는데 지금 받을 수 있나요?"라는 신호를 전송받으려는 쪽에 보내면 전송받는 쪽에서 "물론이죠, 지금 당장 보내세요", 또는 "어 안돼요, 지금 다른 일을 좀 하느라고요"라는 응답을 전송하려는 쪽에 보내게 된다. 이러한 일련의 확인 과정을 핸드쉐이킹이라고 한다. RS-232라인에 연결된 장치는 입력 버퍼가 찼다는 것을 알라기 위해 핸드쉐이킹을 이용한다. 다음 예를 보자. 피씨의 직렬포트가 모뎀을 통해 데이타를 받고

있을 때, 디스크를 읽는다거나 하는 일로 해서, 포트로부터 데이타 읽어오는 일이 잠시 중단되는 경우, 데이타는 모뎀을 통해 직렬 포트로 계속 들어오기 때문에 직렬포트의 입력버퍼가 가득 차 버릴 수 있다. 이렇게 버퍼가 가득차면 직렬포트는 RTS 신호를 0으로 떨어뜨림으로써 이런 상태를 모뎀에 알린다. 버퍼 데이타가 모든 읽히고 나면 RTS가 한번 더 보내지게 되는데 이것은 모뎀에게 피씨가 데이타를 처리할 준비가 되었다는 것을 알린다.

DTR/DSR(Data Terminal Ready/Data Set Ready)

한때, 통신포트는 모뎀을 off-hook(전화기의 훅 스위치를 생각해 보라, 이 스위치가 off, 즉 눌리지 않았으니 전화기가 들린 상태, 통화를 하려고 시도하려는 상태이다)상태로 만드는데 이 DTR신호를 사용했었다. DSR은 모뎀이 연결되어 있다는 것을 나타내는데 사용되었지만 요즘의 대부분의 모뎀들은 DSR를 항상 high로 유지한다. 언제라도 통신포트 쪽에서 모뎀에 접속할 수 있도록 하기 위해서이다. 반면 DTR은 여전히 중요한 제어선으로 사용되고 있다. 대부분의 모뎀들은 여전히 DTR을 low로 떨어뜨리면 전화가 끊긴다.

CD(Carrier Detect)

이것은 다른 것들에 비해 상대적으로 그 의미 파악이 쉽다. 접속이 되어 있다는 것을 알리기 위해 사용된다. 접속중이라면 CD값은 언제나 high이다.

RI(Ring Indicator)

전화가 걸려오면 모뎀에서 통신포트로 이 신호가 보내진다.

너무 어렵다고? 사실 이런 내용을 모두 잘 알고 있어야 하는 것은 아니지만 그 의미 정도라도 이해하고 있어야 통신 프로그램의 작성에 도움이 될 것이다. 실제 프로그래밍에서는 이런 것들을 직접 다루지는 않고 주로 모뎀에서 발생되는 응답들을 해석해서 작업을 하게 된다. 굳이 꼭 알아 두어야 할 것이라면 RTS/CTS, DTR/DSR이 무엇의 약어인지 어떤 의미인지 정도만 알아두면 앞으로 통신 프로그래밍 하는데 별 어려움이 없을 것이다.