RS-232c 통신을 위한 교재

Ver 0.1 / 2000. 05. 12
Ver 0.2 / 2000. 06. 28
Ver 0.3 / 2001. 10. 05
Ver 0.4 / 2001. 11. 22

무한테크놀러지

e-Mail : yongju64@unitel.co.kr Site : http://www.moohantec.com

1 시리얼통신의 이해

1) 용어 설명

① RS-232 (보장사용거리:약 15m)

RS-232C 규격은 미국의 EIA(Electronic Industries Association),현 TIA(http://www.tiaonline.org)에 의해 규격화 된 것으로 정확하게는 EIA-RS232C 규격이라 부르며 전기적/기계적 특성과 인터페이스 회로의 기능 등을 정의하고 있습니다.

RS-232C 는 컴퓨터들과 그 주변의 장치들 간에 비교적 느린 속도의 직렬 데이터 통신을 하기 위한 물리적 연결과 프로토콜에 관해 기술하고 있는 오래된 표준입니다 (현재의 가장 보편적으로 사용하는 버전이 "C"입니다.).

RS-232C 는 컴퓨터가 모뎀과 같은 다른 직렬장치 들과 데이터를 주고받기 위해 사용하는 인터페이스입니다.

RS-232C 가 규격화 되던 시절의 시대적 배경을 생각 해보면 통신의 주 목적은 본사의 컴퓨터와 지사의 컴퓨터를 모뎀을 통하여 서로의 데이터를 주고 받는 것이 주 목적이 아닐까 생각합니다.

컴퓨터로부터 나오는 데이터는 보통 메인보드 상에 있는 UART 칩에 의해 DTE 인터페이스로부터 내장(또는 외장) 모뎀이나 기타 다른 직렬장치 들로 전송이 됩니다.

또한 모뎀이나 다른 직렬 장치와 RS-232C 표준에 입각하여 통신하는 컴퓨터쪽의 DTE 인터페이스는 DCE 인터페이스라고 불리는 보완적인 인터페이스를 가지고 있습니다.

즉,초기에 정의된 RS-232c 의 목적은 DTE(컴퓨터)와 DCE(모뎀)의 연결을 표준으로 하고 있습니다.

뒤에서도 다시 언급이 있겠지만 이런 이유로 인해서 사용하는 케이블의 배선방법이 달라집니다.

DTE 와 DCE 를 연결하려면 1:1 케이블(스트레이트 케이블)을 사용하며 DTE 와 DTE 를 연결 하는 방식에서는 Null Modem Cable(혹은 Cross 케이블)이라고 부릅니다.

컴퓨터 내에 있는 데이터는 병렬회로를 따라 흐르지만 직렬장치 들은 오직 한번에 한 비트씩 만을 처리할 수 있기 때문에, UART 칩이 병렬로 되어 있는 비트들을 직렬 비트 열로 변환시킵니다.

다음의 항목들은 RS-232C 의 전기적 특성입니다.

송신부 - 무부하 출력 : ≤ 25V

송신부 - 부하 출력 : 5V～15V

송신부 - 단락 출력 전류 : ≤500mA

송신부 - 파워온시 특성 : ≥300Ω

수신부 – 입력저항 : 3 kΩ～7 kΩ

수신부 – 히스테리시스 : ±3V

수신부 - 입력 최대전압 : ±25V
최대 케이블 길이 : 15m
최대 전송속도 : 20kbit/sec
신호 논리(부논리) : - TRUE : LOW (-3V 이하)
- FALSE : HIGH(+3V 이상)
- 전송하지 않을 경우 : LOW

위에서 얘기한 전기적 특징은 송/수신측의 전압 레벨과 라인의 입출력 저항 그리고 입력 스레쉬홀드 및 펄스의 상승특성 등을 구성한 것으로 표준 인터페이스의 하나로서 데이터를 직렬로 전송하는 방법을 사용합니다.

따라서 비교적 속도가 느리고 하나의 인터페이스에 대하여 하나의 단말장치 밖에 연결 할수 없으며 전송거리또한 상당히 제한적이긴 하지만 배선수가 적고 접속이 용이하다는 장점때문에 많이 사용합니다.

RS-232C 는 2 개의 송.수신 신호선(TX/RX) 과 5 개의 제어선(CTS/RTS , DTR/DSR , DCD) 그리고 2 개의 접지선 (SG, FG) 이 필요합니다.

참고로 RS-232C 에서 의 232 는 전기공업협회에서 232 번째 제정한 규정이라는 것입니다.(RS-232C : Recommended Standard 232 Revision C)

따라서 직렬통신의 단점을 보완하는 규정들이 422 번째,485 번째 만들어지게 되었고 그들을 RS-422,RS-485 라고 부릅니다.
이들은 RS-232C 의 전송거리가 짧은 이유가 선로상에 발생하는 잡음 및 전송선로에 의한 전압강하 때문이라고 생각하고 잡음에 강한 방식으로 데이터를 보내게 됩니다.

즉 양쪽의 그라운드가 동일한 시점에서의 송수신할 때 잡음이 발생하면 신호가 약해지며 신호를 정확히 판단하지 못합니다. 그래서 송신을 +,- 로 분리해서 부논리를 쌍으로 보내는 방법을 택하여 잡음이 발생할 때 같이 감쇄되어 수신측에서는 보상을 하여 신호로 인식한다는 원리로 전송거리는 약 1.2 킬로 정도 까지 보장을 합니다.

또한 일대일 통신 의 단점을 보완해서 1:N 혹은 N:M 으로 다중통신이 가능하게 만드는 규정도 생겨납니다.

② RS-422 and RS-423 (보장사용거리:약 1.2k)

직렬통신의 단점을 보완하는 규정 422 번째,485 번째 만들어진 것으로 RS-422 와 RS-423 은 EIA 에 의해 승인된 직렬 장치 접속용 표준 인터페이스입니다.

이 표준들은 더 빠른 속도를 제공하면서도 전기적인 간섭에 더욱 강해졌기 때문에, 이전의 표준인 RS-232 를 대체할 목적으로 설계되었습니다.

RS-423 이 오직 점대점 접속만을 지원하는데 비해, RS-422 는 멀티포인트 접속을 지원합니다.

③ RS-485(보장사용거리:약 1.2k)

RS-485 는 멀티포인트 통신회선을 위한 TIA/EIA 표준입니다.

이것은 DB-9 이나 DB-37 과 같은 커넥터들을 지원합니다.

RS-485 는 RS-422 와 비슷하지만, 낮은 임피던스를 가지는 구동기와 수신기를 사용함으로써, RS-422 보다 회선 당 노드 수를 더 많이 허용합니다.

④ 프로토콜

프로토콜 본래의 의미는 외교에서 의례 또는 의정서를 나타내는 말이지만, 네트웍 구조에서는 표준화된 통신규약으로서 네트웍 기능을 효율적으로 발휘하기 위한 협정입니다. 즉, 통신을 원하는 두 개체간에 무엇을, 어떻게, 언제 통신할 것인가를 서로 약속한 규약이라고 볼수있습니다.

컴퓨터 네트웍의 규모가 증가되고 네트웍을 이용한 정보전송 수요가 다양화되며, 소프트웨어와 하드웨어 장비가 계속 증가되는 최근의 환경에서, 효율적인 정보 전달을 하기 위해서는 프로토콜의 기능이 분화되고 복잡해질 수밖에 없습니다.. 따라서 이러한 환경적인 요구를 만족하기 위해서는 프로토콜 계층화의 개념이 필요하게 되었습니다.

프로토콜 계층화의 개념은 마치 구조적 프로그래밍 개념과 비슷한데, 각 계층은 모듈과 같으며 각 계층의 수직적 상하관계는 top-down 구조와 같습니다. 즉, 네트웍의 프로토콜 계층화는 하위계층이 상위계층을 서비스하는 것과 같으며 호출 프로그램과 피호출 프로그램의 매개변수 상호전달 방식 또한 상위계층이 하위 계층의 서비스를 받을 때와 같은 매개변수 전달방식과 같습니다.. 이러한 프로토콜 계층화 개념을 받아들여 상품화한 것이, IBM 사가 1974 년에 내놓은 SNA (system network architecture)입니다.

SNA 의 목적은 IBM 사 제품뿐만 아니라 다른 회사 제품과의 컴퓨터 기기 상호 접속시 발생되는 여러 종류의 호환성 문제를 해결하는 것이었습니다. SNA 이후 다른 회사들도 각자의 네트웍 구조를 내놓았는데, 이들의 목적 또한 네트웍간의 호환성 유지와 정보전송 최소화에 있습니다.

특히 인터넷에서는 TCP/IP 라는 프로토콜을 사용하는데 그 내용은 다음과 같습니다.

TCP (Transmission Control Protocol)는 정보 패킷 차원에서 다른 인터넷 노드와 메시지를 상호 교환하는데 필요한 규칙을 사용하며, IP (Internet Protocol)는 인터넷 주소 차원에서 메시지를 보내고, 받

는데 필요한 규칙을 사용합니다.

결론적으로 프로토콜이라는 단어의 의미는 두 개체간의 통신에 있어서의 약속입니다.
우리가 무전기를 사용할때 마지막에 '오버'라는 말을 붙이는 것 처럼 상대방이 듣고있다가 '오버'라는 말이 나오면 송신이 끝났음을 알고 자신이 할 얘기를 하는것과같은 약속의 의미입니다.

⑤ DTE (Data Terminal Equipment) ; 데이터 단말 장치

컴퓨터 데이터 통신에서 DTE 는 컴퓨터가 모뎀이나 기타 다른 직렬장치를 이용하여 데이터를 교환하기 위한 RS-232C 인터페이스입니다.

⑥ DCE (Data Communication Equipment) ; 데이터 통신 장치

컴퓨터 통신에서의 DCE 는 모뎀이나 다른 직렬장치들이 컴퓨터와 데이터를 주고받기 위해 사용하는 RS-232C 인터페이스를 의미합니다.

⑦ 병렬통신의 종류 LPT (Line Print Terminal)

LPT 는 프린터나 기타 다른 장치에 접속하기 위한 PC 의 병렬포트를 지칭하는 통상적인 호칭입니다.

대부분의 PC 들은 하나 또는 두 개의 LPT 접속을 가지고 있는데, 이들 각각을 LPT1 과 LPT2 라고 부릅니다.. 어떤 시스템들은 세 번째 포트인 LPT3 를 지원하는 것도 있습니다. 그러나, 몇 개가 지원되든 간에 보통 LPT1 이 디폴트값입니다.

사용자는 자신의 컴퓨터에 병렬포트 어댑터를 장착함으로써, 두 번째 프린터나 기타 다른 장치용 병렬포트를 추가할 수 있습니다. LPT 포트는 QuickCam 이나 CU-See-Me 와 함께 쓰이는 비디오카메라 등과 같은 입력장치에도 사용될 수 있습니다.

병렬 접속에는 전통적으로 프린터 통신을 위한 센트로닉스 인터페이스가 사용되어 왔습니다. EPP/ECP 라고 불리는 새로운 표준은 기존의 구형 인터페이스는 물론, 스캐너나 비디오카메라 등을 포함한 다양한 범위의 장치들에 대해 더 빠른 통신을 지원합니다.

프린터포트는 3 개의 주소를 가집니다. 데이터 출력 포트, 상태 포트, 제어 포트입니다.
LPT1 의 경우 기본주소는 378h 입니다. 이때 상태 포트 주소는 379h 가 되며,제어 포트 주소는 37Ah 입니다.
또한 상태 레지스트 포트의 내용은 프린터에러 및 종이 유무 및 기타 작동 상태등을 나타냅니다.

⑧ 병렬통신의 종류 GPIB (General Purpose Interface Bus)

컴퓨터와 주변 기기를 연결하여 정보를 전달하기 위한 외부 버스의 일종입니다.
일반적으로 약어로 불리며,'범용 인터페이스 버스' 라고도 합니다.

원래는 휴렛패커드(HP-현 Agilent)사가 개발하여 자사의 이름을 따서 HPIB 라고 한 것인데, 미국 전기 전자 학회(IEEE)가 1975 년경에 IEEE 488 로 표준화하였으며 이것을 기본으로 하여 국제 전기 표준 회의(IEC)가 IEC 625 로 표준화하였읍니다.

GPIB 는 컴퓨터와 각종 전자 계측 기기, 센서, 자동 시험 기기 등 생산 자동화 관련 기기를 연결하여 정보를 교환하거나 컴퓨터로부터 계측 기기 등을 제어하기 위한 외부 버스의 사실상의 업계 표준 또는 국제 표준이 되어 있지만 일반적인 PC 에서는 많이 사용되지 않습니다.
초기의 GPIB 의 성능은 최대 전송 속도가 초당 1MB, 최대 접속 가능 기기 15 대, 최대 케이블 길이 20m 였으나 점차 데이터 전송 속도를 고속화하고 있는 실정입니다.

GPIB 인터페이스를 사용하기 위해서는 별도의 GPIB Board 를 사용 해야 하며 가장 많이 사용하는 GPIB Board 는 National Instrument 사와 Agilent 사에서 만든 Board 입니다.

2 하드웨어적인 준비.

1) 직렬통신으로 장비를 제어하려고 하거나, 시리얼통신으로 제어가 되는 장비를 만들려고 할때는 다음의 순서에 따라 시험하시기 바랍니다.

① 제어 보다는 모니터링 즉, 수신을 먼저 목적을 가져야 합니다.

② 프로그램을 만들려고 하기 전에 기존의 프로그램으로 충분히 시험합니다.

③ 기존의 프로그램이란 범용 통신 프로그램으로 하이퍼터미널과 같은 프로그램으로 이미 기능상의 검증을 받은 프로그램을 말합니다.

④ 시리얼 포트의 연결 방법은 3 선식만 알아도 99% 가능하므로 다음의 결선 방식을 참고바랍니다.

9 핀	**9 핀**
2 번 (RX)	3 번 (TX)
3 번 (TX)	2 번 (RX)
5 번 (SG)	5 번 (SG)

25 핀	**25 핀**
2 번 (TX)	3 번 (RX)
3 번 (RX)	2 번 (TX)
7 번 (SG)	7 번 (SG)

9 핀	**25 핀**
2 번 (RX)	2 번 (TX)
3 번 (TX)	3 번 (RX)
5 번 (SG)	7 번 (SG)

⑤ 주의점은 통신설정창의 흐름제어를 양쪽의 설정을 동일하게 '없슴'으로 설정하십시오.

⑥ 이와 같은 방법은 초보가 아니라 전문가도 꼭 거쳐야 문제가 적다는 점을 알기 바랍니다.

⑦ RS-422, RS-485 등의 시리얼 통신을 해야 할 경우는 232 TO 422/485 컨버터를 이용하시기 바랍니다.

⑧ 상기의 과정이 완전히 익숙하지 않은 분은 프로그램을 시도하지 않으시는게 좋을 줄로 생각합니다. 반대로 상기과정이 익숙하시면 프로그램으로 통신하는데 일단 모든 걱정을 던져 버리시기 바랍니다.

2) 에코시험 (Loop Back Test)

① 자신의 컴퓨터의 직렬포트에 맞는 케이블을 연결합니다.

② 하이퍼터미널을 실행하여 직접 연결로 COM1 을 설정하는데 나머지는 디폴트로 있는 그대로 하되 흐름 제어는 없음이나 XON/XOFF 로 합니다.

③ 나머지 설정은 스스로 터득하시기 바랍니다.

④ 연결한 케이블 끝단 커넥터의 2 번과 3 번 핀을 찾아내어 쇼트를 안정하게 시켜두고 하이퍼터미널에서 키보드 입력을 했을때만 문자가 에코 된다면 성공입니다.

⑤ 위의 시험이 성공한다면 다른 포트로 옮겨서 다시 시험하시기 바랍니다.

3) 두 포트간의 통신 시험.

① 위의 2 가지 시험이 잘 되었으면 이젠 케이블을 수정하는데 한쪽만 2 번과 3 번 핀의 연결을 바꾸어 연결한다. 이걸 널 모뎀케이블이라고 합니다.(위의 케이블 결선도를 참고 바랍니다.)

② 그럼 COM1 과 COM2 사이에 널모뎀 케이블을 연결하고 두개의 통신프로그램에서 교대로 입력하여 상대 창에 전송되면 시험 종료입니다..

③ 그 상태에서 자주 각종 설정에 따른 변화를 터득하고 기술을 습득하고 프로그램 개발시에도 시험의 무대로 삼기 바랍니다.

4) 널모뎀케이블

초창기에는 데이터 통신이 대부분 모뎀을 통하여 이루어졌습니다.

그러나 세월이 흘러 개인용 컴퓨터의 보급이 엄청나게 늘어나고 이에 따라 모뎀을 이용하지 않고 컴퓨터끼리 통신을 해야할 일의 필요성을 느끼게 되었습니다.

이때 생겨난 것이 널 모뎀 케이블입니다. (위의 케이블 결선도를 참고바랍니다.)

모뎀이 없으면서도 통신을 하게 해주는 케이블인셈이지요.

즉 DTE – DCE 의 통신이 아닌 DTE – DTE 의 통신이 필요한것이지요.

이때 사용하는 케이블은 Direct (Straight) 방식이 아닌 Cross(Twist) 방식의 케이블을 사용합니다.

송신과 수신을 서로 교차되게 연결하고 접지를 같이 연결한 케이블이 널모뎀 케이블입니다.

5) 루프백 커넥터

루프백 커넥터는 루프백 시험을 하기 위한 커넥터입니다.

루프백 시험은 통신 기기가 제대로 동작하는지 알아보기 위한 시험으로서 통신 기기의 송신부와 수신부를 연결하여 자기가 보낸 데이터를 다시 자기가 받아보도록 하여 통신 기기의 이상유무를 판단할 수 있도록 합니다. 이는 다른 기기가 없이 혼자서도 해볼 수 있는 간단한 시험입니다.

6) 모뎀을 이용한 통신

모뎀이라해서 크게 어려운건 없읍니다.

모뎀의 기능은 컴퓨터에서 나오는 디지탈 신호를 아날로그 신호로 바꿔서 멀리가게 하는 기능과 아날로그신호로 들어오는 데이터를 다시 디지털 신호로 바꿔주는 기능이라고 보시면 됩니다.

일반 시리얼 통신과의 차이점은 접속해야하는 과정 뿐입니다. 그이후에는 똑같죠.

우리가 일반적으로 사용하는 모뎀은 헤이즈호환 모뎀이라고 합니다.
그렇기 때문에 모뎀에 보내는 명령은 'AT'라는 헤더를 가집니다.
먼저 수신측 프로그램이 가동되면서 모뎀에 'ATS0=1'이라는 명령을 보냅니다.
다음과 같은 코드로 표현됩니다.
MSComm1.Output = “ATS0=1” & vbCrLf

자세한 정보는 모뎀관련 매뉴얼에 자세히 나와있읍니다.

그리고 송신측에서는 접속이 필요한 시점에서 전화를 거는기능이 있어야겠죠?
'ATDT 123-4567'이라고 명령을 주면 됩니다.

그러면 모뎀이 전화를 걸게 되고 수신측에서는 링이 울립니다.

그러면 수신측 모뎀은 응답을 하게됩니다.

그때 수신측/송신측 모뎀은 서로의 조건을 확인합니다.

서로의 조건이 맞다면 양쪽 컴퓨터에 접속완료 메세지가 뜹니다.
이때 응답코드가 문자로 나타날지 숫자코드로 나타날지는 모뎀에 설정한 값에의해 결정됩니다만..
어쨌던 모뎀에서는 결과가 나옵니다.
만약 수신측이 통화중이라면 'BUSY'가 나올테고
정상적으로 접속이 되면 'CONNECT 9600'등이 나옵니다.

이때부터는 일반 직렬통신과 같이 보시면 됩니다.

아래에 간단히 절차를 순서대로 적도록 하겠읍니다. 참고 하시길..

(송) 이라고 한것은 송신측 컴퓨터이고 (수)라고 한것은 수신측 컴퓨터입니다.
물론 양쪽의 포트 속성은 똑같이 맞추셔야 하는건 아시겠지요?

(수) 포트를 연다. Ex) MSComm1.PortOpen = True
(수) 'ATS0=2'를 보낸다 Ex) MSComm1.Output = "ATS0 = 2" & vbcr
(송) 포트를 연다 Ex) MSComm1.PortOpen = True
(송) 전화를 건다 Ex) MSComm1.Output = "ATDT 123-4567" & vbcr
(수) 링이 울린다 Ex) 따르릉~~
(수) 두번째 링이 울린후 빈대떡 부치는 소리가 들린다..
(송) 이에 뒤질세라 해물파전 부치는 소리가 들린다..
(송,수) 거의 동시에 접속 완료 메세지가 나타난다.
문자일경우도 있고 부호일경우도 있다. 일반적으로 'Connect xxx'
----- 여기까지가 접속 과정입니다. ------

(송) 온도를 구하는 명령을 보낸다. Ex) MSComm1.Output = "N0303" & vbcr
(수) 명령을 받은후 현재의 온도를 보낸다..
(송) 수신버퍼를 확인하여 화면에 표시한다..

----- 모든 작업이 끝났다고 가정합니다. -----
(수) 모뎀을 자동응답기능을 해제합니다. Ex) MSComm1.Output = "ATS0 = 0" & vbcr

(수) 전화를 끊습니다. Ex) MSComm1.Output = “ATH0” & vbcr
(송) 전화를 끊습니다. Ex) MSComm1.Output = “ATH0” & vbcr

7) D-Sub 커넥터의 핀 구성

9 핀	25 핀	신호명	입/출력 구분	설명
	1	FG	-	보호용 접지 (Frame Ground)
2	3	RXD	입력	수신 데이타 (Receive Data)
3	2	TXD	출력	송신 데이타 (Transmitted Data)
7	4	RTS	출력	송신 요구 (Request to Send)

8	5	CTS	입력	송신 허가 (Clear to Send)
6	6	DSR	입력	데이타 세트 레디 (Data Set Ready)
5	7	SG	-	신호용 접지 (Signal Ground)
1	8	DCD	입력	수신 캐리어 검출 (Carrier Detect)
4	20	DTR	출력	데이터 단말 레디 (Data Terminal Ready)

3 기본 통신 개통시험.

처음부터 프로그램을 개발하기 보단 정확히 하드웨어적으로 정상적인지를 테스트 해보아야 합니다.

그러기 위해서는 단순해 보이지만 오랜 시간동안 Test 되었고 확실하다 싶은 프로그램으로 Test 를 해보아야 합니다.

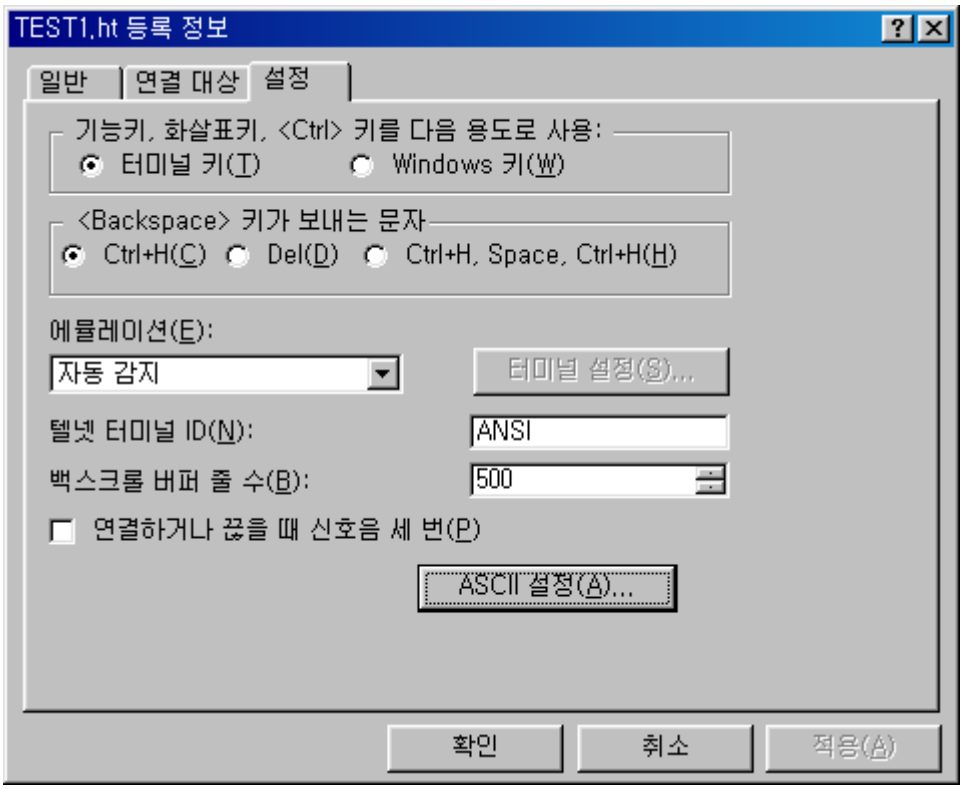
그러한 대표적인 프로그램으로 하이퍼 터미날을 이용하시기 바랍니다.

1) 하이퍼 터미널을 이용한 통신 시험

① 보조프로그램에서 하이퍼 터미날을 선택하여 새로 등록합니다.

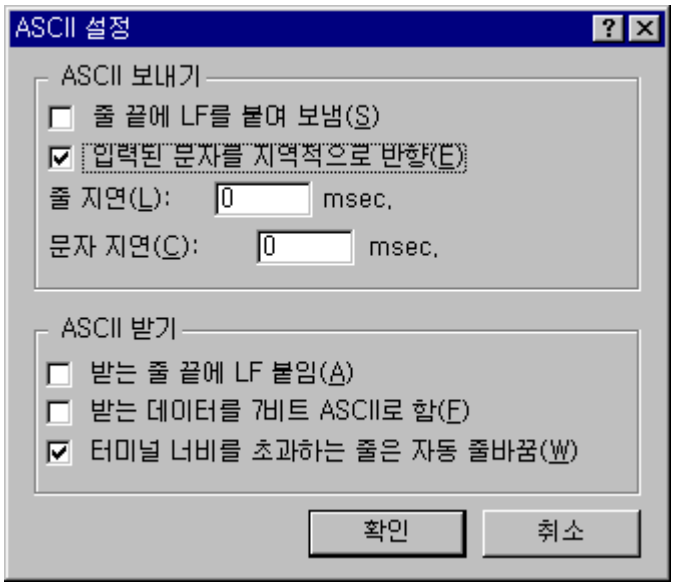
② [주의사항] – 모든 항목에 대해 동일해야 합니다.

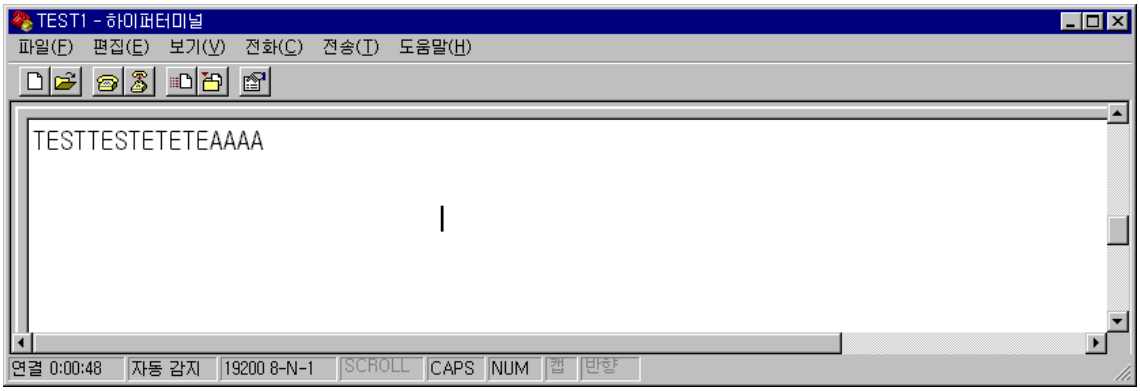
③ 새로 만든 하이퍼 터미날의 등록정보를 엽니다.



④ 연결대상에서 사용할 모뎀 COM Port 를 지정합니다.

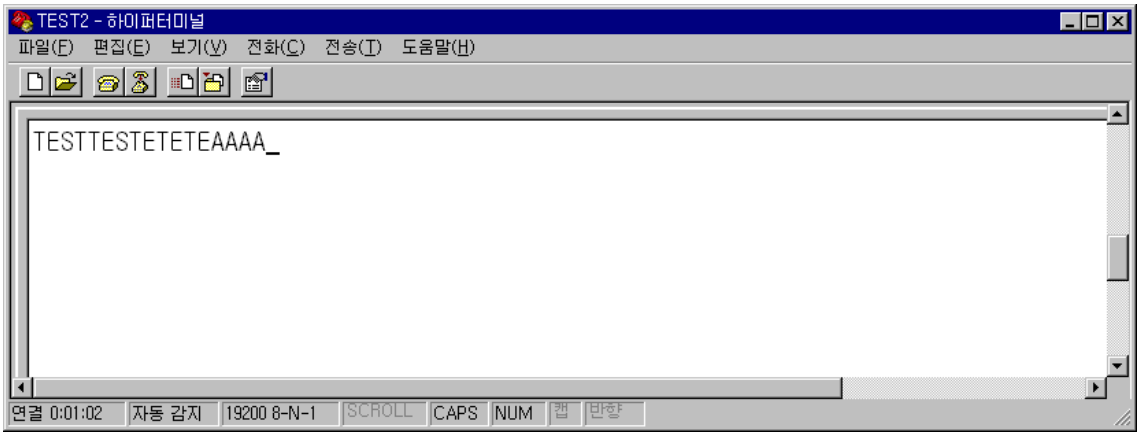
⑤ 설정에서 “ASCII 설정”에서 아래와 같이 지정합니다.





[TEST1 COM1 PORT OPEN]

⑥ TEST1 에서 입력합니다.



[TEST2 COM2 PORT OPEN]

⑦ TEST1 에서 입력한 내용을 확인합니다.

4 비베에서 바라본 시리얼통신.

1) 먼저 알아야 할 내용.

① 통신 속성들

- 전송 속도 는 데이터를 보내는 속도를 얘기합니다.

단위는 bps(bit per second), 초당 전송되는 비트수를 얘기합니다.

그렇다면 우리는 흔히 8 비트 통신을 하게 되는데,

만약 9600bps 라고 설정 한다면 초당 9600 비트이므로

바이트로 환산 하면 9600/8 이 됩니다.

그러면 1200 바이트/초 가 이론적으로는 계산이 됩니다만..

실제 한바이트가 8 비트로 전송이 될때는 다른 비트들도 계산해주어야 합니다.

시리얼 통신으로 데이터를 보내게 되면 다음의 순서로 데이터가 송신 됩니다.

시작비트 – 데이터비트 – 패리티비트 – 종료 비트

이때 시작비트는 항상 1 비트,데이터 비트는 7 비트혹은 8 비트,

패리티를 사용하면 1 비트,사용하지 않으면 0 비트,종료 비트는 선택에 따라 1,1.5,2 비트가 나갑니다.

예를들어 우리가 흔히 사용하는 속성인 9600,n,8,1 를 예를 들어 보겠습니다.

전송속도-9600

패리티 – 없슴

데이터 – 8 비트

종료비트 – 1 비트

이렇게 되면 1 바이트의 데이터 전송될 때 실제 나가는 비트수는 총 10 비트입니다.

시작 1 비트 + 데이터 8 비트 + 종료 1 비트 = 10 비트

따라서 9600 보레이트의 경우 바이트로 환산 하면 960 바이트가 1 초에 전송이 되는 것이지요.

이렇게 계산 해보면 1 바이트 문자가 전송되는데는 약 1 밀리초가 소요됩니다.

이 내용은 수신시에도 그대로 적용이 됩니다.

우리가 흔히 범하기 쉬운 오류는 데이터를 송신 하고는 바로 수신 버퍼를 읽어온다는 것입니다.

우리가 데이터를 보내면 오로지 데이터를 보내는데만 걸리는 시간이 있으며,

수신측에서는 그 데이터를 받아서 처리하는데 걸리는 시간도 있을것이며,

수신측에서는 그에 응하는 응답을 보내게 될 때 그 데이터 길이에 따른 수신완료시의 시간이 있다는 것을 흔히 무시하는 경우가 있습니다.

다시 예를 들어 보겠습니다.

컴퓨터에서 ‘Hello?’라는 명령을 보내면 상대측에서는 곧 바로 ‘I am here.” 라는 응답을 보낸다는 가정을 해보도록 하겠습니다.

아까 얘기한 송신후 바로 수신하는 경우의 코드는 다음과 같습니다.

```
MSComm1.Output = “Hello?” & vbCr
```

```
    MsgBox MSComm1.Input
```

이 코드는 데이터를 보낸후 바로 수신버퍼에서 읽고 끝냅니다.
그러나 이 경우 데이터가 정상적으로 수신되기 힘듭니다.
Output 명령 이후 Input 명령까지의 시간 간격이 적어도 20 밀리초 이상이 소요되는 초울트라수퍼캡 저속 컴퓨터라면 가능합니다.
하지만 실제 상황은 그렇지 못합니다.

따라서 이 경우 데이터가 수신이 될때까지 기다리기 위해서 다음의 같은 방식으로 처리를 해주려고 합니다.

```
MSComm1.Output = “Hello? “ & vbCr

Do
    Doevents
    If MSComm1.InBufferCount > 0 Then Exit Do
Loop
MsgBox MSComm1.Input
```

이 코드는 처음 코드에 비해서는 기능이 향상 되었지만 그래도 문제점은 남아있습니다.
다시 생각해보면 명령을 내보내고 수신버퍼에 데이터가 들러오기 시작하면 루프를 빠져 나갑니다.
그리고 데이터를 수신버퍼에서 읽어옵니다.
이 경우의 문제는 수신버퍼에 있는 데이터를 읽으려는 시점이 모든 데이터가 다 수신완료된 시점이 아닐수도 있다는 문제입니다.
즉,데이터를 수신중인데도 수신완료로 생각해버린다는 문제이지요.

따라서 이 경우는 종료부호를 이용하여 통신을 한다면 종료부호를 체크해서 수신완료로 보는 방법과 수신될 바이트수를 알고 있는 경우라면 그 수신하고자 하는 만큼의 데이터가 들어왔는지를 확인 하는 방법과 데이터 수신중에 발생하는 수신 블록간의 갭을 이용하여 수신 완료로 보는 등의 방법을 강구해야 합니다.

다음의 코드를 참고하시기 바랍니다.
이 경우는 종료부호를 확인 하는 방법입니다.

```
MSComm1.Output = “Hello?” & vbCr

RBuf$ = “”
```

```
Do
    DoEvents
    If MSComm1.InBufferCount > 0 Then
        Rbuf$ = Rbuf$ & MSCOmm1.Input

        If Instr(Rbuf$,vbCr)>0 Then Exit Do
    End If
Loop
```

② QUEUE

큐는 순서를 가지고 있는 선형리스트 구조입니다.

요소의 삽입과 인출(삭제)가 다른 구조로 되어있으며 매표소에서 입장권를 사기 위해서 줄을 서서 기다리는 사람들의 모습을 생각하시면 됩니다.(물론 새치기는 허용이 되지 않습니다.)

여기서는 먼저 들어간 요소가 먼저 인출이 되므로 선입선출,혹은 FIFO(First In First Out)이라고도 합니다.

그리고 요소의 삽입이 일어나는 곳을 꼬리(Rear)라고 하며 인출이 일어나는 곳을 머리(Front)라고 합니다.

③ STACK

스택은 한쪽 끝이 막혀있는 구조로서 요소의 삽입과 인출이 한쪽끝에서만 일어나는 선형 리스트 구조입니다.

밑이 막혀있는 통을 세워놓은 구조를 생각하시면 됩니다.

여기서는 나중에 들어간 요소가 먼저 인출이 되게 되어있는 구조이므로 후입선출,혹은 LIFO(Last In First Out)이라고도 합니다.

요소의 삽입,인출이 일어나는 곳을 스택의 상위(Top)이라고 부르며 요소를 스택에 넣는 것을 푸쉬(Push)라고 부르며 요소를 인출하는 것을 팝(Pop)이라고 합니다.

스택은 서브루틴 호출이나 인터럽트 처리시 돌아갈 주소를 저장하는 용도로 많이 쓰이기도 합니다.

④ CIRCLE QUEUE

비베에서 가지는 통신용 버퍼에는 두가지가 있습니다.

수신버퍼와 송신 버퍼입니다.

이 버퍼의 크기는 각각 MSComm 의 속성중 InBufferSize 와 OutBufferSize 의해서 결정이 됩니다.

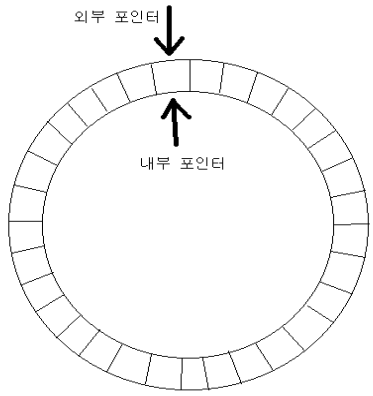
우리가 주로 프로그램에서 많은 고민을 하는버퍼는 수신버퍼입니다.

따라서 지금부터는 수신 버퍼를 기준으로 설명 드리도록 하겠습니다.

수신버퍼는 환형큐라고 부르는 구조로 되어있습니다.

이 환형큐는 기본 구조 및 동작은 일반큐와 같으나 머리와 꼬리가 붙어있는 환형으로 되어있는 것이 다릅니다.

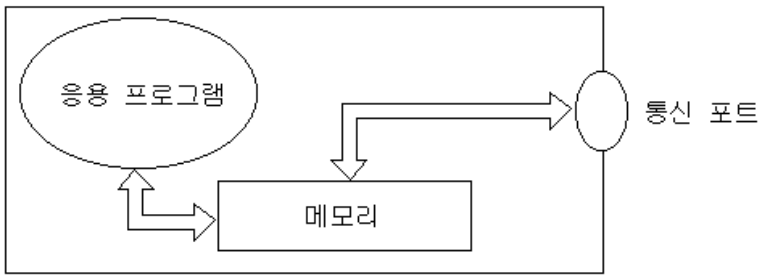
이 환형큐의 특징은 두개의 포인터를 가지고 있습니다.
하나의 포인터는 통신포트에 도착한 데이터를 수신버퍼에 갖다놓은 위치를 가리키고 있으며, 이 포인터를 외부포인터라고 부르겠습니다.
또 하나의 포인터는 응용프로그램에서 가져올수 있는 수신버퍼의 위치입니다.
이 포인터는 내부포인터라고 부르겠습니다.
다음의 그림을 참고 하시기 바랍니다.



Circle Queue
환형 큐

각각의 격자에는 수신되는 데이터가 바이트 단위로 적재가 됩니다.
위의 그림처럼 외부포인터와 내부포인터가 마주 보고있다면 더 이상 버퍼에 남은 데이터가 없는걸로 생각하여 InBufferCount 가 0 이 됩니다.

먼저 다음의 그림을 보시면서 데이터의 흐름을 생각해보시기 바랍니다.

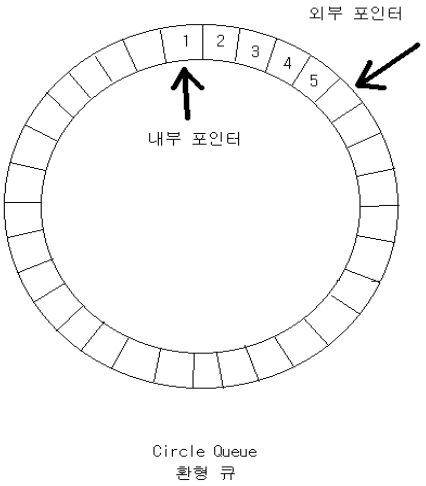


컴퓨터

외부에서 발생한 데이터가 통신 포트에 도착하면 CPU 및 통신드라이브는 그 데이터를 1 바이트씩 묶어서 메모리에 갖다 둡니다.
이때의 메모리는 수신버퍼로 생각하시면 됩니다.
그러면 우리가 만들거나 이미 만들어진 통신 프로그램에서는 수신버퍼에 들어와있는 데이터를 확인 하

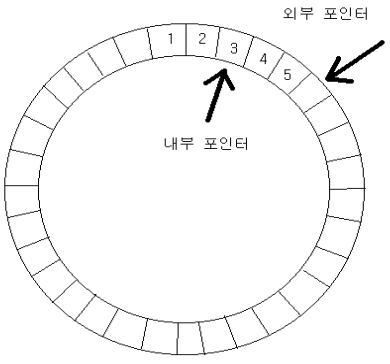
여 메모리로부터 가져오게 됩니다.
즉,우리의 응용프로그램에서는 통신포트로부터 직접 데이터를 가져오는게 아니며 통신 드라이브가 데이터를 바이트 단위로 묶어서 수신버퍼(메모리)에 갖다두면 그때 우리가 데이터를 얻을수 있는 구조로 되어있다는걸 생각하시기 바랍니다.

자, 실제 예를 들어 보겠습니다.
지금 외부에서 보낸 데이터가 있습니다. ‘12345” 라는 문자가 들어왔다고 하겠습니다.
물론 Input 명령은 아직 사용하지 않았습니다.
그러면 다음과 같은 그림이 그려집니다.



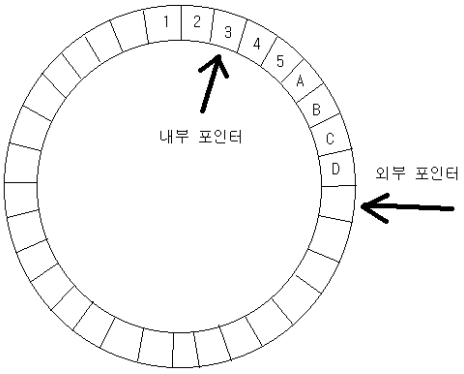
Circle Queue
환형 큐

아직 데이터를 가져가지 않았기 때문에 두개의 포인터는 벗어나 있습니다.
이때 InBufferCount 를 보면 두 포인터의 차이를 계산해서 5 라는 값이 나옵니다.
이때 InputLen 라는 속성을 2 로 하고 Input 을 해보면 다음의 그림과 같이 됩니다.



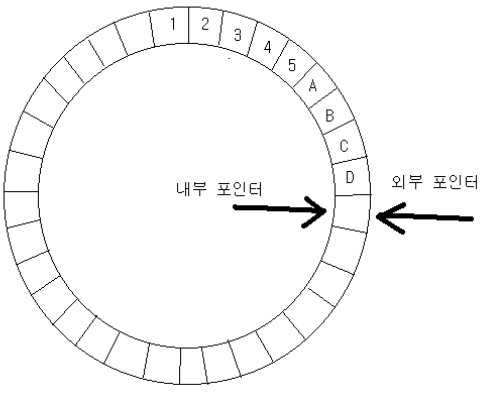
Circle Queue
환형 큐

즉 두바이트를 인출 시켜서 응용프로그램에 돌려주고는 내부포인터의 위치가 변경이 됩니다.
그리고 다시 데이터가 들어왔다고 가정합시다.
이번에는 ‘ABCD” 가 들어 왔다면 다음 그림처럼 데이터가 누적이 됩니다.



Circle Queue
환형 큐

이번에는 InPutLen 속성을 0 으로 주고 다시 읽어봅니다.
물론 MSComm1.Input 으로 읽겠지요?
그러면 버퍼에 남아있던 모든 데이터를 다 되돌려주고 두 포인터는 서로 마주 보게 됩니다.
다음의 그림처럼..



Circle Queue
환형 큐

자, 여기서 정리를 해보겠습니다.
우리가 사용하는 MSComm 의 수신버퍼는 환형큐로 되어있습니다.
이 환형큐보다 더 큰 데이타가 들어오거나 우리가 미처 수신 데이터를 가져가기전에 다른 데이터가 들

어온다면 버퍼내에서는 오버런이 발생합니다.
이때는 이전의 데이터는 없어지면서 그 위에 덮어쓰게 됩니다.
그리고 InputLen 은 한번의 Input 명령에 수신될 바이트를 지정한다는건 이해가 되셨을겁니다.
그리고 InBufferCount 의 속성은 두개의 포인터의 차이를 되돌려주며 이는 수신 버퍼에 남아잇는 데이터 수를 돌려줍니다.
그런데 이 InBufferCount 의 또다른 용도가 있습니다.
MSComm1.InBufferCount = 0 이라고 하면..수신버퍼를 클리어시키는 역할을 합니다.
수신 버퍼에 쓰레기 데이터가 있다고 가정할 때 수신버퍼를 비우는 용도로도 사용되지요.
저번의 강좌에서도 말씀드렸지만 9600 보레이트로 통신을 한다면 통신포트에서부터 수신버퍼까지 데이터가 들어오는 시간이 1 밀리초가 걸립니다.
따라서 수신버퍼에 우리가 원하는 데이터가 다 수신이 되는데는 수신하려는 바이트수 * 1 밀리초이상이 걸리므로 이 시간을 잘 기다려 주셨다가 읽어야만 여러분들께서 원하는 데이터를 얻으실수 있다는걸 유념해 주시기 바랍니다.

⑤ POLLING 과 INTERRUPT

우리가 통신을 한다는 목적은 같은 기종 혹은 다른 기종끼리의 데이터 송수신을 목적으로 하고있습니다.
데이터를 보내는 조건이나 시기는 보내는 쪽에서 대체적으로 결정을 합니다.
송신이 필요할 때 즉시 송신을 하거나 미리 예약을 했다가 정해진 시간에 송신을 하면 되겠죠?
하지만 수신의 경우는 조금 다릅니다.
보내는 쪽에서 언제 보낼지를 모르기 때문입니다.

우리가 지금 다루고 있는 통신의 인터페이스는 RS-232 통신입니다.
이 RS-232 는 비 동기모드 통신 방식이라고 합니다.
여기서 말하는 비 동기 통신 방식이라는건 양쪽의 기기들에게 약속된 동기신호가 없다는 얘기입니다.
만약 정해진 동기신호(클럭)이 있다면 그 동기신호에 맞추어 데이터를 주고 받으면 됩니다만..
우리가 배우고 있는 RS-232 에는 약속된 동기신호가 없습니다.
따라서 비 동기모드 통신이라고 하지요.
그러면 동기신호가 없는데 어떻게 데이터를 인식할 수가 있는지 궁금하실겁니다.
여기서 중요한건 시작비트와 종료비트 입니다.
이 때문에 일본에서는 RS-232 통신을 조보통신(한문으로 쓰는건 잊어버렸습니다..^&^)이라고 하며
영어식으로 표현하면 Start-Stop Transmission 이라고 표현합니다.
이 내용 또한 시작비트와 종료비트로 동기를 맞춘다는 얘기이지요.

송신측에서 '1'을 보낸다고 가정을 하겠습니다.
'1'은 10 진수로는 49 가 되며 16 진수로는 '31'이 됩니다.

이때 송신측에서 문자'1'을 보내려고 하면 그 문자의 아스키코드의 10 진수 형태와 16 진수형태는 큰 의미가 없습니다.
즉,
MSComm1.Output = Chr$(49)
MSComm1.Output = Che$(&h31)
MSComm1.Output = "1"
위의 세가지 방법 모두 같은 데이터가 나간다는 얘기입니다.

어떤 포맷으로 보내든지간에 원하는 문자의 아스키코드에 해당하는 코드값이 8 비트(혹은 7 비트)로 이진 현태로 변환이 되며 그 앞에는 시작비트가 나가고 그 뒤에는 패리티비트(원할때만..) 그리고 그 뒤에는 종료비트가 붙어서 나갑니다.
이걸 단순히 생각해서 1 과 0 으로만 표시 해본다면..(기본 설정인 9600,n,8,1 로 설정 가정하고..)

시작비트 – 데이터 비트(8 비트) – 종료비트 순으로 나갑니다. (패리티는 None 이므로..)
즉, 1 0 0 1 1 0 0 0 1 1 의 순서로 나가지요. (비트간의 간격은 의미가 없습니다.)

그러면 수신측에서는 수신라인을 지켜보고 있다가 시작비트가 확인 되면 그 시점부터 정해진 시간인 100 마이크로 초 (이 시간은 전송속도인 보레이트로부터 계산이 됩니다. 지금의 예는 9600 보레이트이므로 1 바이트에 수신에 1 밀리초가 필요하며 1 비트의 간격은 약 100 마이크로 초입니다) 마다 샘플링을 하여 1 인지 0 인지를 판단합니다.
이렇게 8 개의 비트를 모두 추출하면 이들을 합하여 바이트로 변환시키는 것이지요.
따라서 송신을 할 때 16 진수값전송이나 10 진수값전송에는 큰 의미를 두지 마시기 바랍니다.

자,다시 주제로 돌아가도록 하죠.
위와 같은 방식으로 송수신을 하기 때문에 수신자는 언제 데이터가 들어올지를 예측 할 수는 없습니다.

그래서 필요한 것들이 있습니다. 폴링과 인터럽트입니다.
물론 이들은 데이터의 수신을 그 기본목적으로 하고 있습니다.
단지 데이터의 수신이 아니고 수신의 동기를 맞추는게 더 중요한 목적이 잇다고 하겠습니다.
수신의 동기를 보다 더 정확하게 맞춘다면 데이터를 읽어내오지 않아서 발생할수 있는 오버런을 예방할수도 있으며 기기가 보내주는 중요정보를 바로 사용자에게 알려줄수 있는 효과와 잇점을 제공할수 있기 때문입니다.

폴링과 인터럽트를 다음과 같이 생각하시면 됩니다.

집에 전자오븐렌지가 있습니다.

제가 먹다남은 피자가 있길래 프로그램을 짜다말고 피자를 렌지에 넣고 시간을 맞추고 돌렸습니다.
그리고 다시 프로그램을 짜고 있습니다.
저는 피자가 다 되었는지 아직 덜 되었는지를 생각 할 필요가 없습니다.
젠자렌지에는 조리가 끝나면 '띵~'하는 소리로 조리가 끝났음을 알려주기 때문입니다.
따라서 저는 프로그램을 짜다말고 '띵~'하는 소리가 들리면 ' 아! 피자가 다 데워졌구나..'라고 생각하고 그에 맞는 행동을 합니다.
이것이 인터럽트입니다. 제가 어떤 일을 하더라도 이벤트가 발생하면 프로세서에게 보고가 되어지는 것..이것을 인터럽트라고 부르지요..MSComm 에서는 OnComm 이벤트로 대신합니다.

그런데 어느날 다시 피자를 데우려고 갔는데 전자레인지 문에 쪽지가 붙어있습니다.
"주의 - 이 전자레인지는 고장으로 인하여 조리후 '띵~'소리가 나오지 않음"
이때 저는 고민을 합니다. 그래도 피자를 먹을것인가..
만약 먹고 싶다면 저는 피자를 전자레인지에 넣고 조리를 시작한 다음 부터는 조리가 끝날 때 까지는 가슴을 졸이며 계속 전자레인지를 확인 해야 합니다.
그렇지 않으면 조리가 끝난걸 모르고 있다가 다 식은 피자를 먹어야 할지도 모르기 때문입니다.
이것이 폴링입니다.
어떤 작업이 끝났는지 혹은 어떤 이벤트가 일어났는지의 이벤트를 프로세서가 계속 확인을 하는 방법입니다.

우리는 경우에 따라서 폴링을 사용하거나 인터럽트(이벤트)를 사용하거나 결정을 해야합니다.
만약 두 가지를 다 사용하려고 한다면 우리는 정상적인 데이터를 얻을수 없기때문입니다.

간단하게 비교를 한다면 폴링은 데이터를 보내고 응답을 확인 하면서 기다리는 방법입니다.
다음의 소스를 생각 해보시기 바랍니다.

```
Comm1.InBufferCount = 0
Comm1.Output = "D" & vbCrLf

tmp$ = ""
sTime = Now
Do
    DoEvents
    If Comm1.InBufferCount > 0 Then
        tmp$ = tmp$ & Comm1.Input

        If Right$(tmp$, 2) = vbCrLf Then
            rData = Split(tmp$, ",")
```

```
            Exit Do
        End If

        sTime = Now
    End If

    nTime = Now
    If Abs(DateDiff("s", nTime, sTime)) >= 3 Then Exit Do
Loop
```

위의 경우 명령을 보내고 난 다음 종료부호가 들어오는지 확인을 하면서 수신갭의 시간이 3 초를 초과하는지 확인을 합니다.
이 처럼 데이터를 보내고 원하는 데이터를 계속 확인 하면서 기다리는 방법을 폴링이라고 생각하시면 됩니다.

하지만 인터럽트는 다릅니다.
인터럽트는 필요 시 명령을 보냅니다. 그리고 끝입니다. 왜냐면 데이터가 수신이 되면 이벤트가 발생하라고 했기 때문이지요.
그렇다면 인터럽트를 발생시키려면 어떻게 해야 할까요.
비베에서는 약 10 가지의 오류 이벤트와 7 가지의 이벤트가 있습니다.
우리가 원하는 데이터 수신 이벤트는 그 중의 하나입니다.
따라서 MSComm 의 Oncomm 이벤트 내에서도 현재 발생한 이벤트가 데이터 수신 이벤트인지를 확인하는 기능이 들어가야 논리적 오류를 조금이라도 줄일수 있으리라 생각됩니다.
그렇다면 어떻게 해야 이벤트를 발생 시킬 수가 있을까요?
여기서 해답을 주는 속성이 있습니다.
그것은 바로 Rthreshold 라는 속성이었던 것입니다..(개그 콘서트 버전..)

이 속성은 언제 OnComm 이벤트를 발생시킬 건지를 설정하는 속성입니다.
이 속성에 0 을 설정 하면 이벤트를 사용하지 않을 것 이라는 의미입니다.
즉, 폴링으로 사용하겠다는 의미이지요.
하지만 0 이외의 어떤 값 이 설정되어있다면 설정한 바이트 수 만큼의 데이터가 들어오면 그때 이벤트를 발생시키겠다는 의도로 보시면 됩니다.
일반적으로는 1 을 설정합니다.
이때는 데이터가 들어올때마다 이벤트를 발생된다고 보시면 됩니다.

자,정리를 하도록 하겠습니다.

데이터를 안전하게 수신하는 방법에는 폴링과 인터럽트가 있습니다.
폴링은 데이터가 들어왔는지를 스스로 확인 하는 방법이며,
인터럽트는 데이터가 들어왔을 때 프로세서에게 즉각 보고를 하도록 하는 방법입니다.
비베를 이용한 통신을 MSComm 으로 위의 방법을 사용하실때는 RThreshold 라는 속성을 이용하시면 됩니다.
이 속성에 0 이 들어있으면 폴링방식을 사용한다는 의미이며..
0 이외의 값이 들어있으면 설정한 수 만큼의 데이터가 수신되면 그때 이벤트가 발생이 된다는것입니다.

⑥ 바이너리 데이터 처리

우리가 통신을 하는 목적은 같은 기종 혹은 이기종 간의 데이터교환을 목적으로 하고 있습니다.
여기서 말하는 데이터는 컴퓨터가 인식하는 데이터를 의미하며 흔히 아스키코드로 얘기를 합니다.
ASCII(American Standard Code for Information Interchange : 미국 표준 정보 교환 코드) 코드는 컴퓨터내부에서 문자를 표현하는 표준적인 코드 체계입니다.
7 비트로 구성되며 자료의 처리나 통신 장치의 표준코드로 널리 사용되고 있습니다.
기본적인 영문자,숫자 그리고 기호들은 모두 7 비트로 표현이 가능합니다.
그러나 우리나라나 중국 일본처럼 유니코드를 사용하는 나라들은 8 비트를 사용해야 합니다.
최상위비트인 8 번째 비트를 사용함으로서 그 데이터들을 표현할수 있기 때문입니다.

여기서 잠깐 아스키코드 테이블을 살펴보겠습니다.

아스키코드 0 ~ 31 까지의 데이터는 제어코드 영역입니다.
이 영역의 데이터들은 모두 콘트롤키와의 조합으로 사용할수 있습니다.
예를들면 우리가 사용하는 엔터키를 캐리지리턴이라고도 합니다.
아스키코드값으로는 13 (&HD)입니다. 이 값의 명칭은 CR-즉,캐리지 리턴입니다.
그리고 콘트롤키 조합으로는 CTRL-M 입니다.
그렇기 때문에 텍스트 박스같은 곳에서 엔터를 누르지 않고 CTRL 과 M 을 동시에 눌러도 같은 효과가 나는것입니다.
또한 이 영역의 데이터들은 우리가 통신을 할 때 자주 사용하는 코드들이 있습니다.
때로는 데이터 프레임의 선두에 서서 데이터를 이끌고 갈때도 있고 프레임 중간 중간에 섞여서 나갈때도 있으며 또는 데이터의 끝을 장식하는 경우가 있습니다.
흔한 예로 우리는 데이터를 보낼 때 데이터시작을 의미할수 있게 ‘STX’를 보내며,
그리고 데이터의 끝을 알아볼수 있게 ‘ETX’를 붙여서 보낸다고 가정하겠습니다.
그때 우리는 다음과 같은 방법으로 데이터의 프레임을 표현합니다.
STX + Data + ETX
이때 STX 와 ETX 는 제어코드로서 프레임의 형식을 통일시키는 역할을 하지요.
만약 Data 가 “123” 이라는 문자열이라면 실제 전송되는 데이터는 다음과 같습니다.

02 31 32 33 03 (16 진수 형태로 표현하였습니다.)
즉,한 바이트의 제어코드로서 서로의 송수신 프레임에 대한 약속을 하는것입니다.
이러한 용도로 사용하는 것이 제어코드 입니다.
하지만 딱 정해진 규정은 없습니다. 서로간의 약속에 의해서 맞춰 사용하시면 됩니다.

아스키코드 32 ~ 127 까지의 데이터는 일반 문자 영역입니다.
여기에는 각종 부호와 숫자 영문자등이 있습니다.

그리고 또하나의 중요한 영역이 있습니다.
아스키코드 128 부터 255 까지의 영역입니다.
저는 이 영역을 확장 아스키코드 영역이라고 부릅니다.
이 영역은 아까도 말씀드렸지만 우리나라나 중국,일본같은 2Byte 문자를 사용하는 나라에서는 자국의 언어를 처리하는 영역으로 사용됩니다.
MSB(최 상위비트)를 1 로 Set 함으로서 확장된 코드를 사용할수 있습니다.
또한 이 영역은 다른 용도로도 사용됩니다.
통신을 하다 보면 송수신 데이터양을 줄이기 위해서 각 비트별 가중치를 적용하는 경우가 많이 발생합니다.
예를 들면 0 에서 255 까지의 숫자를 보내야하는 장비가 있습니다.
이 장비가 일반 텍스트 형태로 데이터를 보내게 되면 최대 3 바이트가 전송이 되지만 바이너리 형태로 보내게 되면 1 바이트로 전송이 가능합니다.
이런 경우로 인해서 바이너리 데이터 송수신이 자주 일어납니다.

자,지금부터 바이너리 데이터의 송수신에 관해서 본격적으로 설명을 드리겠습니다.

우리가 비베를 통하여 데이터를 주고 받을 때 문제가 발생하는 부분은 확장 아스키코드 영역(128 부터 255 까지의 코드값을 갖는 데이터)의 데이터 송수신입니다.

예전에 비베 3.0 에서는 이런 문제가 전혀 없었습니다.
그런데 비베 4.0 으로 올라가면서 튀어나온 문제가 확장 아스키코드 영역의 데이터 송신시 Null 로 송신되는 문제였지요.
당시에 한국 마이크로소프트와도 많은 얘기를 했지만 결국은 안된다는 결론이었습니다.
MS 에서 제공해준 방법은 API 를 사용하는 방법이었고,그 방법또한 무난하지 않겠다는 판단에 저는 제가 아는 분께 부탁을 드려 델파이로 DLL 을 만들어서 처리를 했지요.
하지만 지금은 바이트 배열이라는 방법을 통하여 문제를 처리하고 있습니다.

여기서도 송신과 수신으로 나누어 생각을 해 보도록 하겠습니다.

먼저 송신시의 바이너리 데이터 처리방법입니다.
보내고자 하는 데이터의 코드값이 128 부터 255 까지의 데이터가 있다면 일반 문자열 전송 방식으로는 정상적인 처리가 되지 않습니다.
이때 사용하는 방법이 바이트 배열 방식입니다.

먼저 전송하고자 하는 데이터의 크기에 맞는 바이트를 배열로 선언 한다음..
각각의 방에 전송하려는 데이터의 아스키코드값을 넣고..
MSComm 의 Output 을 이용할 때 바이트 배열명을 지정하시는 방법과..
방이 하나짜리인 바이트 배열을 만들고..
전송하려는 데이터의 아스키코드값을 그 배열에 넣은다음 그때그때 전송을 하는 방법으로 생각해볼수 있습니다.

전자의 경우는 다음의 소스를 참고 하시기 바랍니다.

```
Dim sBuf(0 to 4)  As Byte

SBuf(0) = 2        'STX
SBuf(1) = &h31   '1
SBuf(2) = &h32   '2
SBuf(3) = &h33   '3
SBuf(4) = 3        'ETX

MSComm1.Output = sBuf
```

이때 주의하실 사항은 Output 할 때 지정한 배열의 모든 첨자가 다 전송이 됩니다.
예를들어 100 개짜리 방을 만들고 그중 5 개의 방에만 데이터를 넣은 다음 원하는 5 개의 데이터만 전송을 할 수가 없습니다.
아무리 5 개의 데이터를 넣고 그 데이터만 전송을 하고 싶어도 100 개의 데이터가 모두 전송이 된다는걸 유념하시기 바랍니다.

후자의 경우는 다음과 같습니다.

```
Dim sBuf(0)    As Byte

SBuf(0) = 2        'STX
```

```
MSComm1.Output = sBuf

SBuf(0) = &h31  '1
MSComm1.Output = sBuf

SBuf(0) = &h32  '2
MSComm1.Output = sBuf

SBuf(0) = &h33  '3
MSComm1.Output = sBuf

SBuf(0) = 3     'ETX
MSComm1.Output = sBuf
```

두 경우의 차이점을 생각해보시고 사용하시려는 용도에 적합한 방법으로 사용하시면 됩니다.

다음은 바이너리 데이터의 수신시 처리방법입니다.
먼저 수신하려는 데이터가 확장 아스키코드 영역의 데이터를 포함하고 있다면..
MSComm 의 속성중 InputMode 를 바이너리로 설정하셔야 합니다.
그리고 Input 을 이용하여 데이터를 수신하는 변수는 Variant 로 선언 하시면 됩니다.
이때 InputLen 의 속성에 따라 수신변수의 상황은 달라집니다.
만약 InputLen 을 1 로 설정 해서 한 바이트씩 데이터를 받아내면 수신 변수는 바이트변수와 비슷합니다.
그런데 InputLen 이 1 을 초과하면 수신 변수는 그 데이터의 크기에 맞는 바이트 배열로 됩니다.
이 속성만 잘 이해하시면 큰 문제는 없습니다.

다음의 소스를 잘 생각해보시기 바랍니다.

```
Dim rBuf   As Variant

MSComm1.InputMode = 1   'Binary Mode
MSComm1.InputLen = 1    '1Byte 씩 수신

If MSComm1.InBufferCount >=1 Then
    RBuf = MSComm1.Input        '이때는 바이트 변수와 같이 동작합니다.
End If
```

```
MSComm1.InputMode = 1   'Binary Mode
MSComm1.InputLen = 0    '한꺼번에 모든 데이터 수신

If MSComm1.InBufferCount >1 Then
    RBuf = MSComm1.Input        '이때는 바이트배열로 동작합니다.
End If                          '만들어진 방의 개수는 UBound 함수로 알아보실수 있습니다.
```

여기서 다시 한번 정리 하도록 하겠습니다.

MSComm 의 속성중 InputMode 는 데이터 수신에만 적용되는 속성입니다.
데이터 송신의 경우는 보내는 방법을 달리 함으로서 바이너리데이타를 보내실수 있기 때문에 InputMode 와는 관계가 없는 것이지요..
송신데이타중 확장 아스키코드영역의 데이터가 있다면 일반 문자열 전송방식으로는 정상적으로 송신이 되지않기 때문에 이때는 바이트 배열을 이용하시면 됩니다.

데이터 수신의 경우는 InputMode 를 바이너리 모드로 하시고 Variant 로 데이터를 받으시면 됩니다.
이때 데이터의 길이에 따라,혹은 InputLen 의 속성에 따라 수신되는 데이터가 바이트 변수가 될수도 있고 바이트배열변수가 될수도 있습니다.

2) MSComm 컨트롤의 많은 중요한 5 가지 속성

① CommPort 통신 포트 번호를 반환하거나 설정합니다.
② Settings 전송 속도, 패리티, 데이터 비트, 정지 비트를 문자열로 반환하거나 설정합니다.
③ PortOpen 통신 포트의 상태를 반환하거나 설정합니다. 또한 포트를 열거나 닫습니다.
④ Input 수신 버퍼에서 문자를 반환하거나 삭제합니다.
⑤ Output 문자열을 전송 버퍼에 기록합니다.

3) 기타 참고 속성

① Break 중단 신호 상태를 설정하거나 해제합니다. 디자인 모드에서는 이 속성을 사용할 수 없습니다.
② CDHolding 반송파 검출(CD) 회선의 상태를 쿼리하여 반송파가 존재하는 지의 여부를 결정합니다. CD 는 모뎀에서 모뎀이 온라인인 지를 나타내기 위해 연결된 컴퓨터로 보내는 신호입니다. 이 속성은 디자인 모드에는 사용할 수 없으며 실행 모드에서는 읽기 전용입니다
③ CommEvent 가장 최근의 통신 이벤트나 오류를 반환합니다. 이 속성은 디자인 모드에는 사용할 수 없으며, 실행 모드에서는 읽기 전용입니다.

④ CommID 통신 장치를 구분하는 핸들을 반환합니다. 이 속성은 디자인 모드에는 사용할 수 없으며 실행 모드에서는 읽기 전용입니다

⑤ CTSHolding Clear To Send(CTS) 회선 상태를 쿼리하여 데이터를 보낼 수 있는지의 여부를 결정합니다. 일반적으로 전송 취소 신호는 전송이 진행될 수 있는 지를 나타내기 위해 연결된 컴퓨터에서 모뎀으로 보내집니다. 이 속성은 디자인 모드에서는 사용할 수 없으며 실행 모드에서는 읽기 전용입니다.

⑥ DSRHolding Data Set Ready (DSR) 회선의 상태를 결정합니다. 일반적으로 데이터 설정 준비 신호는 작업할 준비가 되어 있는 지를 나타내기 위해 연결된 컴퓨터에서 모뎀으로 보내집니다. 이 속성은 디자인 모드에는 사용할 수 없으며, 실행 모드에서는 읽기 전용입니다.

⑦ DTREnable 통신 도중 Data Terminal Ready (DTR)를 가능하게 할 것인지의 여부를 결정합니다. 일반적으로 데이터 터미널 준비 신호는 컴퓨터가 들어오는 전송 신호를 받을 준비가 되어 있는 지를 나타내기 위해 컴퓨터에서 모뎀으로 보내집니다.

⑧ EOFEnable EOFEnable 속성은 입력 도중 MSComm 컨트롤이 End Of File(EOF)를 찾을지 결정합니다. EOF 문자를 발견하면 입력이 중지되고 CommEvent 속성이 comEvEOF 로 설정되면서 OnComm 이벤트가 발생합니다

⑨ Handshaking 하드웨어 초기 접속 신호 프로토콜을 반환하거나 설정합니다.

⑩ InBufferCount 수신 버퍼에서 기다리고 있는 문자의 수를 반환합니다. 이 속성은 디자인 모드에는 사용할 수 없습니다.

⑪ InBufferSize 수신 버퍼의 크기를 바이트 단위로 반환하거나 설정합니다.

⑫ InputLen Input 속성이 수신 버퍼에서 읽는 문자의 수를 반환하거나 설정합니다.

⑬ InputMode Input 속성에서 검색하는 데이터 형식을 반환하거나 설정합니다.

⑭ RThreshold MSComm 컨트롤이 CommEvent 속성을 comEvReceive 로 설정하고 OnComm 이벤트를 발생시키기 전에 수신할 문자의 개수를 반환하거나 설정합니다.

⑮ RTSEnable RTS(전송 요청) 회선을 활성화 시킬지를 결정합니다. 일반적으로 데이터 전송에 대한 허가를 요청하는 RTS 신호는 컴퓨터에서 모뎀으로 보내집니다.

16 Settings 전송 속도, 패리티, 데이터 비트, 정지 비트 매개 변수를 반환하거나 설정합니다.

17 SThreshold MSComm 컨트롤이 CommEvent 속성을 comEvSend 로 설정하고 OnComm 이벤트를 발생시키기 전에 전송 버퍼에서 허용 가능한 최소 문자 개수를 반환하거나 설정합니다.

4) 초기 접속 신호

수신 및 전송 버퍼를 관리할 때는 데이터가 앞뒤로 성공적으로 전송되는지 확인해야 합니다. 예를 들어 데이터가 수신되는 속도는 버퍼 한계를 넘지 않아야 합니다.

초기 접속 신호는 데이터가 하드웨어 포트에서 수신 버퍼로 전송될 때 내부 통신 프로토콜을 참조합니다. 데이터 문자가 직렬 포트에 도달하면 프로그램이 읽을 수 있도록 통신 장치는 문자를 수신 버퍼로 옮겨야 합니다. 초기 접속 신호 프로토콜은 통신 장치가 수신 버퍼로 데이터를 옮기는 것보다 빠르게 포트에 수신되어 포화된 버퍼에서의 데이터 유실을 막아줍니다.

Handshaking 속성을 설정하여 응용 프로그램에서 사용될 초기 접속 신호 프로토콜을 지정할 수 있습니다. 기본적으로 Handshaking 속성값은 없음(comNone)으로 설정됩니다. 그러나 아래 프로토콜 중 하나를 지정할 수 있습니다.

설정	값	설명
ComNone	0	초기 접속 신호 없음(기본값)
ComXOnXOff	1	XOn/XOff 초기 접속 신호
ComRTS	2	RTS/CTS(전송 요청/전송 취소) 초기 접속 신호
ComRTSXOnXOff	3	전송 요청과 XOn/XOff 초기 접속 신호 모두

연결하는 장치에 따라 프로토콜을 선택할 수 있습니다. 이 값을 comRTSXOnXOff 로 설정하면 두 프로토콜을 모두 지원합니다.

대부분의 경우 통신 프로토콜 자체에서 초기 접속 신호를 처리할 수 있습니다. 따라서 이 속성을 comNone 이외의 것으로 설정하면 충돌이 발생할 수 있습니다. 이 값을 comRTS 나 comRTSXOnXOff 로 설정하려면 RTSEnabled 속성을 True 로 설정해야 합니다. 그렇지 않으면 연결하여 데이터를 보낼 수 있지만 받을 수는 없습니다.

5) MS Comm 컨트롤 의 오류 메시지

아래 표는 MSComm 컨트롤에 대해 잡을 수 있는 오류의 목록입니다.

상수	값	설명
ComInvalidPropertyValue	380	속성값이 잘못되었습니다.
comSetNotSupported	383	속성은 읽기 전용입니다.
comGetNotSupported	394	속성은 읽기 전용입니다.
comPortOpen	8000	포트가 열려 있는 동안은 작동이 유효하지 않습니다
	8001	시간 초과 값은 0 보다 커야 합니다
comPortInvalid	8002	포트 번호가 잘못되었습니다.
	8003	속성은 실행 모드에서만 사용할 수 있습니다.
	8004	속성은 실행 모드에서만 사용할 수 있습니다.
comPortAlreadyOpen	8005	포트가 이미 열려 있습니다.
	8006	장치 구별자가 유효하지 않거나 지원되지 않습니다.

	8007 해당 장치의 전송 속도가 지원되지 않습니다.
	8008 지정된 바이트 크기가 유효하지 않습니다.
	8009 기본 매개 변수가 잘못되었습니다.
	8010 다른 장치에서 잠궜기 때문에 하드웨어를 사용할 수 없습니다.
	8011 그 기능은 큐를 할당할 수 없습니다.
comNoOpen	8012 그 장치가 열리지 않았습니다.
	8013 그 장치는 이미 열려 있습니다.
	8014 comm 통지를 활성화할 수 없습니다.
comSetCommStateFailed	8015 comm 상태를 설정할 수 없습니다.
	8016 comm 이벤트 마스크를 설정할 수 없습니다.
comPortNotOpen	8018 포트가 열려 있을 때만 작동할 수 있습니다.
	8019 장치를 사용 중입니다.
comReadError	8020 통신 장치를 읽는 중 오류가 발생하였습니다.
comDCBError	8021 포트의 장치 컨트롤 블록을 검색하는 중에 내부 오류가 발생하였습니다.

6) MS Comm 컨트롤 의 이벤트 리스트

```
Private Sub Comm_OnComm ()
    Select Case MSComm1.CommEvent
        ' 각 case 문 아래에 코드를 위치시켜 이벤트나 오류를 처리합니다.

        ' 오류
        Case comBreak       ' 중지 신호 수신.
        Case comCDTO        'CD (RLSD) 시간 초과.
        Case comCTSTO       'CTS 시간 초과.
        Case comDSRTO       'DSR 시간 초과.
        Case comFrame       ' 프레이밍 오류
        Case comOverrun     ' 데이터 손실.
        Case comRxOver      ' 수신 버퍼 초과.
        Case comRxParity    ' 패리티 오류.
        Case comTxFull      ' 전송 버퍼 꽉 참.
        Case comDCB         'DCB 검색 중 예기치 못한 오류.

        ' 이벤트
        Case comEvCD        'CD 회선 변경.
        Case comEvCTS       'CTS 회선 변경.
        Case comEvDSR       'DSR 회선 변경.
        Case comEvRing      ' 호출음 검출.
        Case comEvReceive   ' 데이터 수신 이벤트 발생
        Case comEvSend      ' 데이터 송신 이벤트 발생
        Case comEvEof       ' 파일 끝
    End Select
End Sub
```

5 응용프로그램을 이용한 통신 시험

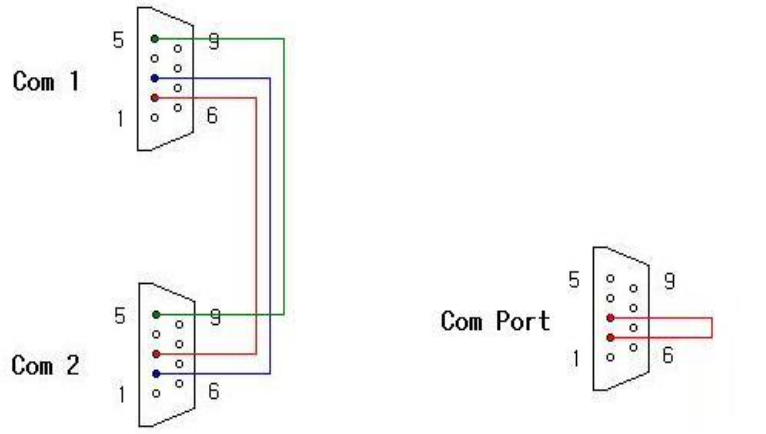
1) 간단한 예제를 통한 포트확인 및 송수신 시험

지금부터 설명하려는 예제의 소스는 제 홈의 자료실에 있습니다.

http://www.moohantec.com 자료실-기초자료실에 “[소스] 간단한 통신 프로그램 예제”라는 이름으로 올려두었습니다.

따라서 여기서는 부분적인 소스만 올려서 설명을 드리도록 하겠습니다.

그리고 다음의 예제를 시험 하기 위해서는 간단한 케이블과 커넥터가 있어야 합니다.



<< 간이형 널 모뎀 케이블 >> << 루프백 커넥터 >>

<< 그림 1 >>

이 케이블은 널모뎀케이블이라는 케이블로서 같은 컴퓨터내의 Com1 과 Com2 를 연결하는 케이블이며 다른 컴퓨터끼리의 연결도 가능하게 해주는 케이블입니다.

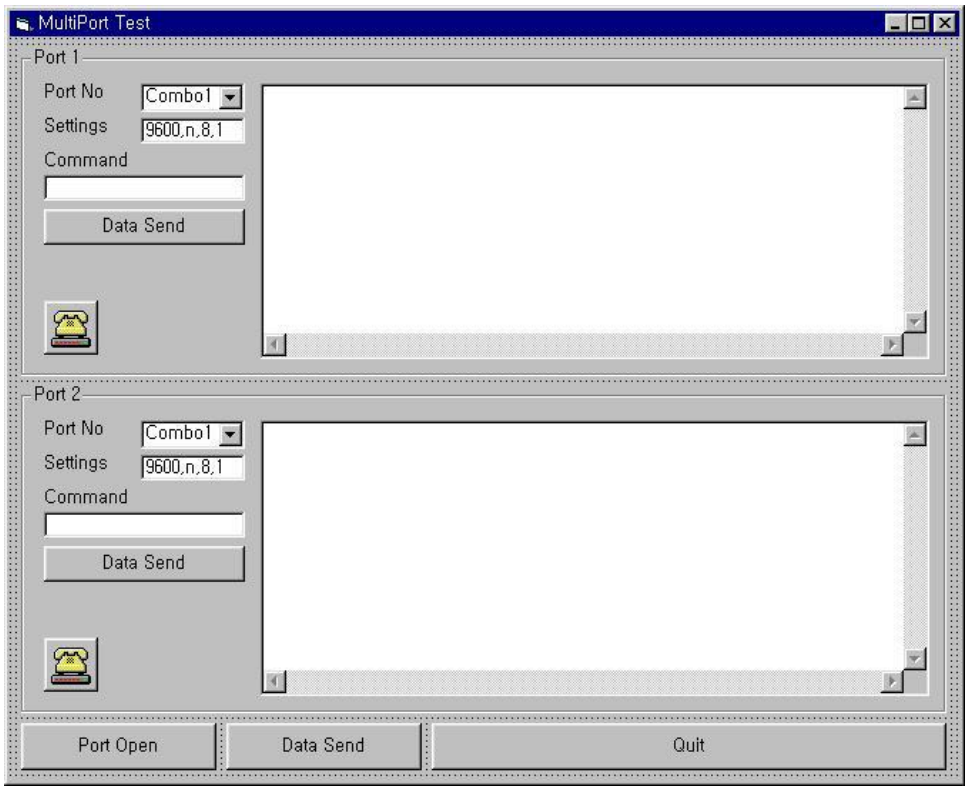
그림에서 보셨듯이 이 방법으로 만든 케이블은 핸드쉐이크를 사용하지 않을경우만 적용이 되며 만약 핸드쉐이크를 사용하는 장비에 연결 할 경우에는 배선이 추가 되어야 합니다.

그리고 그 옆에 있는 커넥터는 루프백커넥터로서 자신의 송신선과 자신의 수신선을 서로 연결하여 송신한 데이터를 그대로 신신하게 만들어서 포트의 이상유무를 확인하게 해주는 커넥터입니다.

먼저 이 케이블과 커넥터를 준비 하시기 바랍니다.

자, 그럼 소스로 들어가겠습니다.

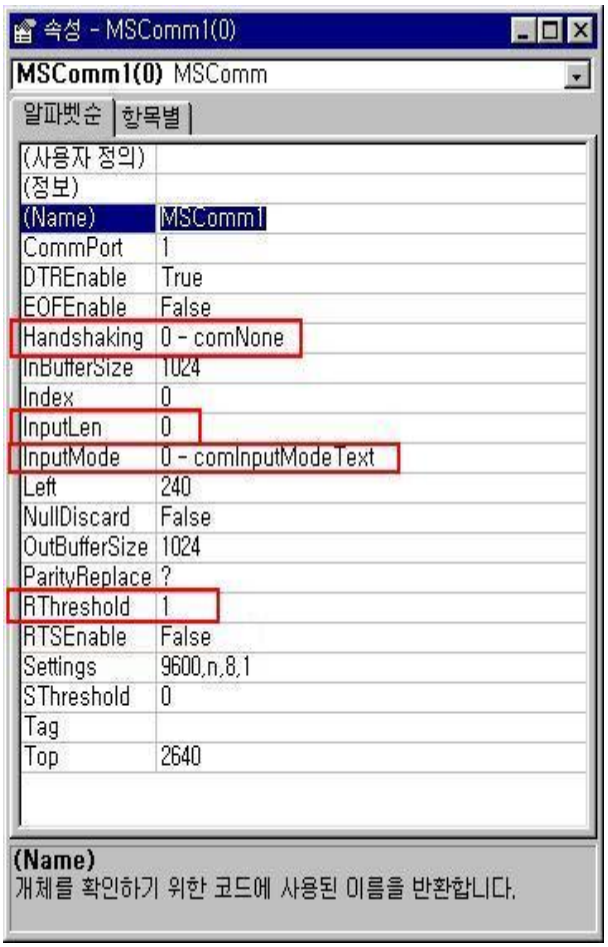
먼저 소스를 다운 받으신 다음 압축을 푸시면 폼 하나가 달랑 나옵니다.
폼을 열어보시면 다음의 그림과 같이 나올겁니다.<< 그림 2 >>



이 화면은 디자인시의 화면입니다.
위의 프레임과 아래의 프레임은 별도의 포트를 이용할수 있게 만들었으며 따라서 각각의 MSComm 콘트롤을 가지고 있습니다.

다른 부분들은 일반 비쥬얼베이직기능이므로 혹시 궁금하신게 있으시더라도 직접 해결하시기 바랍니다.

먼저 MSComm 의 속성을 살펴 보겠습니다.
아래의 그림을 참고 하시기 바랍니다.<< 그림 3 >>



여기서 중요한 속성은 빨간색으로 둘러쳐진 속성들입니다.

- Handshaking 속성은 데이터를 주고 받을 때 손실을 방지하고 원할한 통신을 위해서 사용하는 속성입니다.

말 그대로 악수와 같은 의미이며 내가 손을내밀었을 때 상대방이 손을 잡아주어야만 악수가 성립이 되는것과 같이 송신측에서 데이터를 보내기전 수신측의 허락을 얻는 과정으로 생각하시면 됩니다.
종류에는 하드웨어 방식인 RTS/CTS 방식과 소프트웨어방식인 XON/XOFF 방법이 있습니다.
그러나 이번 과정에서는 사용하지 않도록 하겠습니다.

- InputLen 속성은 한번의 Input 명령으로 받아들일 데이터의 바이트수를 지정합니다.

0 일 경우에는 수신 버퍼에 잇는 모든 데이터를 다 가져온다는 의미입니다.
물론 1 일 경우에는 Input 한번에 1 바이트만 끄집어 내온다는 뜻이겠지요?

- InputMode 속성은 텍스트와 바이너리중 하나를 선택할수 있습니다.

앞 강좌에서도 설명을 드렸지만 텍스트라고 부르는 데이터는 아스키코드 0~127 까지의 데이터를 보통 얘기합니다.이 데이터들은 제어코드 및 일반 문자들의 영역입니다.
쉽게 표현하면 우리는에 보이는 문자들이라고 생각하시면 됩니다.

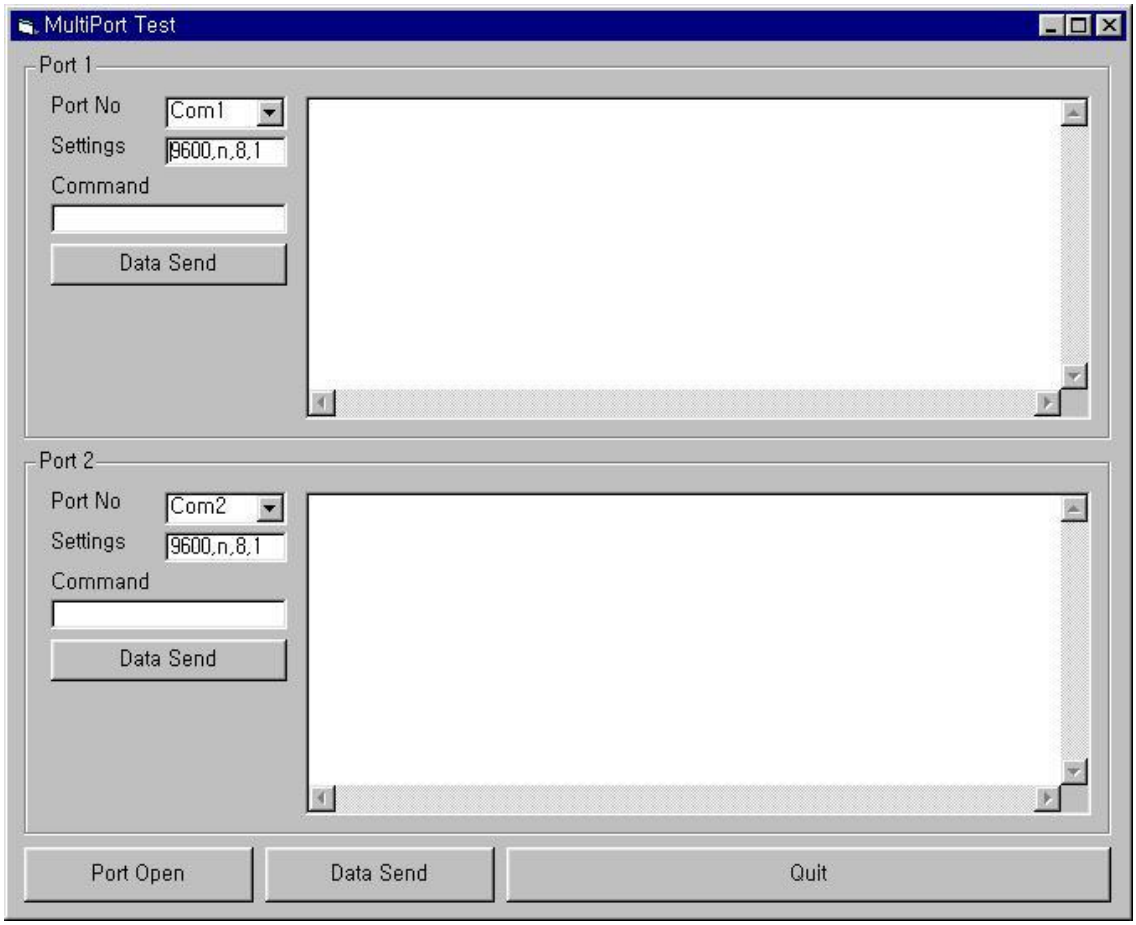
그러나 아스키코드 128 부터 255 까지는 이상한(?) 문자들로 보일수 있는 영역이며 이 영역의 데이터를 흔히 이진데이타 혹은 바이너리 데이터라고 부릅니다.
이 데이터들을 수신할 필요가 있을때는 이 속성은 바이너리로 설정하시면 됩니다.
참! 데이터를 송신할때는 관계가 없다고 했던말 기억하시나요?

- RThreshold 속성은 데이터수신을 인터럽트(이벤트)로 처리할 때 사용하는 속성입니다.

이 속성에 0 이 들어있으면 인터럽트를 사용하지 않고 폴링으로 처리하겠다는 의미이며
이 속성에 0 이 아닌 다른 숫자가 들어있으면 해당 숫자만큼의 데이터가 수신버퍼에 들어오면 그때 이벤트를 발생시키겠다는 의도입니다.
이때 발생되는 이벤트는 MSComm 이 가지고 있는 17 개의 이벤트 중 하나이며,
그 이름은 comEVReceive 입니다.
따라서 우리는 대부분 수신이벤트를 많이 사용하므로 OnComm 이벤트가 발생하더라도 먼저 comEVReceive 이벤트인지를 확인하고 다음 프로시져로 넘어간다는 설명을 다시한번 기억하시기 바랍니다.

이정도면 MSComm 의 기본적이지만 중요한 속성을 미리 설정했습니다.

그러면 다음의 화면을 보시기 바랍니다. << 그림 4 >>

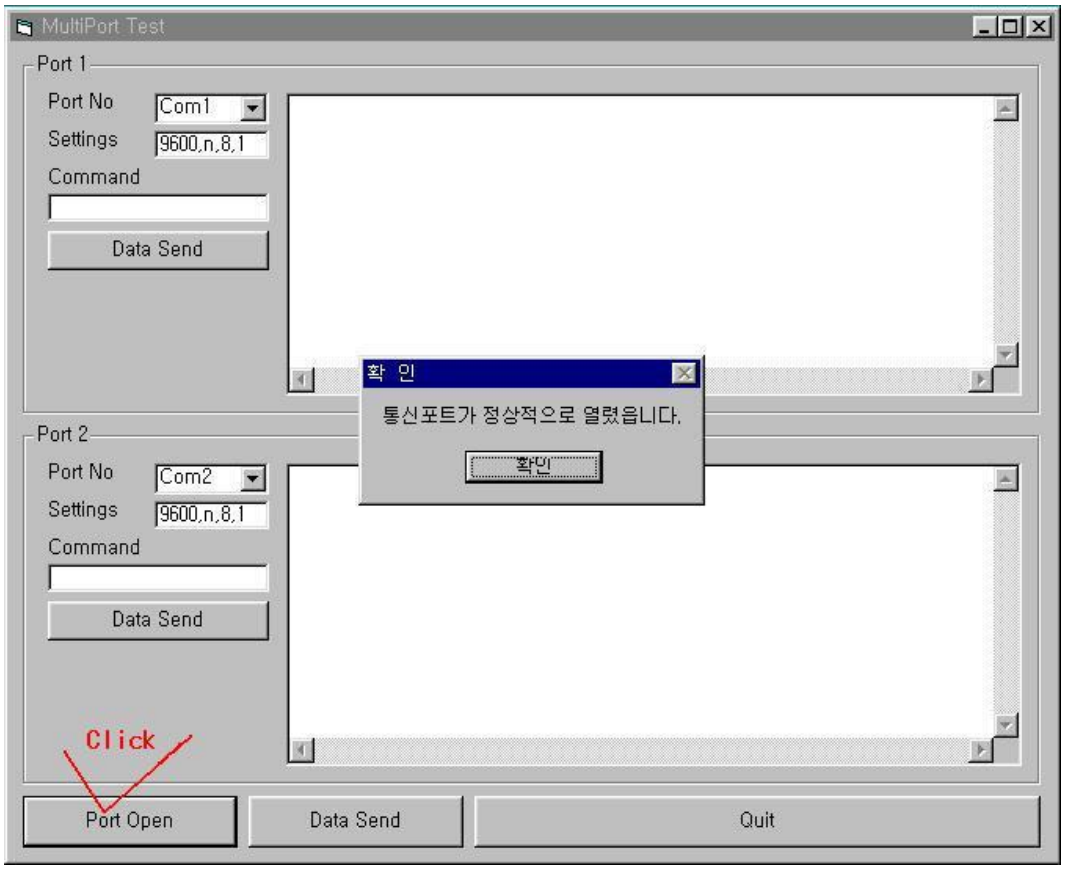


이 화면은 프로그램을 실행 시켰을 때 나타나는 화면입니다.

이 프로그램을 실행 시키면 사용할 각각의 포트를 지정해야 합니다.
각각의 콤보박스를 선택해서 사용할 포트를 선택하시고 통신 속성을 설정하시면 됩니다.

기본적으로는 컴퓨터에 Com1 과 Com2 가 있으므로 각각 Com1 과 Com2 가 설정이 됩니다.
그리고 속성은 가장 많이 사용하는 '9600,n,8.,1' 이 기본으로 등록되어있습니다.

먼저 사용할 포트에 이상이 없는지 'Port Open'이라는 단추를 누르면 다음과 같은 화면이 나타납니다.



<< 그림 5 >>

이때는 각각 설정한 통신 포트에 이상이 없음을 알려주며 만약 포트가 열리지 않으면 에러메세지를 보여줍니다. 다음의 소스를 확인 하시기 바랍니다.

```
Private Sub cmdPortOpen_Click()
    Dim i%, eFlag%

    On Error Resume Next

    eFlag = True
```

```
    For i = 0 To 1
        With MSComm1(i)
            .PortOpen = False
            .CommPort = cboPort(i).ListIndex + 1
            .Settings = txtSettings(i).Text

            .PortOpen = True
            DoEvents

            If Not .PortOpen Then
                eFlag = False
                MsgBox "통신 포트가 정상적으로 열리지 않았읍니다.", , fraPort(i).Caption
            End If
        End With
    Next i

    If eFlag Then
        MsgBox "통신포트가 정상적으로 열렸읍니다.", , "확  인"
    End If

End Sub
```

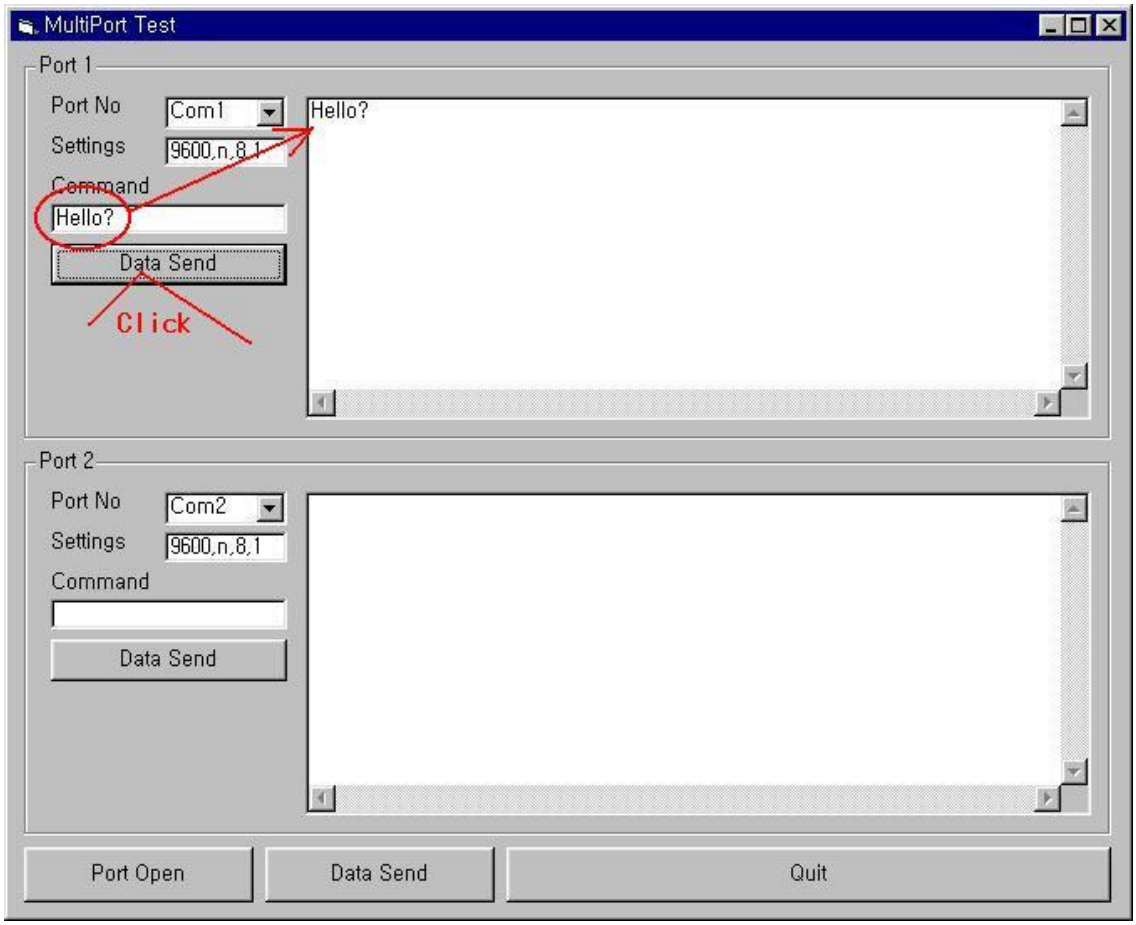
위에서 On Error 문을 사용햇는데 이유는 포트 번호를 잘못 지정하고 포트를 열면 시스템 에러가 발생합니다.이때는 응용프로그램자체가 종료가 되기 때문에 이럴경우를 대비해서 에러를 무시하고 다음으로 보낸다음 포트의 정상적인 상태를 확인하기 위함입니다.

또 여기서 우리가 흔히 범하기 쉬운 오류가 있습니다.
우리는 보통 PortOpen = True 라고 사용함으로서 포트를 여는 일을 다 했다고 생각하기 쉽습니다만..
포트는 연다고 다 열리지 않는 경우가 있다는 것이지요.
따라서 포트를 열라고 명령을 주고나서도 다시 한번 포트가 잘 열렸는지 확인을 하고 만약 열리지 않았으면 사용자에게 메시지를 보여주어 다시 포트를 열도록 해야 한다는 것입니다.

이렇게해서 사용할 포트가 잘 열렸으면 데이터를 보내봅니다.

이때 자신의 포트로 데이터를 내 보내서 자신의 포트로 받는,즉 루프백 시험을 먼저 해 보도록 하겠습니다.
먼저 위에서 설명한 루프백 커넥터를 준비해서 지신의 컴퓨터 Com1 에 꽂아 둡니다.

그리고 Port1 의 Command 에 ‘Hello?’ 라고 입력 한 다음 아래에 있는 ‘Data Send’단추를 누르면 다음과 같은 그림이 됩니다.



<< 그림 6 >>

자신의 포트에서 나간 데이타를 그대로 수신 라인에 연결했기 때문에 다시 자기 자신이 받는 것이지요.

만약 여러분들의 컴퓨터에서 위의 그림과 같이 되지않는다면 다음 두가지의 문제를 생각해보시기 바랍니다.
첫째. 사용하려는 통신 포트에 문제가 있거나
둘째. 방금 사용한 루프백 커넥터를 잘못 만들었을 가능성이 있습니다.

다음은 ‘Data Send’ 단추의 소스입니다.

```
Private Sub cmdSendPort1_Click()

    If Not MSComm1(0).PortOpen Then
        MsgBox "Port1 번의 포트가 열리지 않았읍니다.", , "확  인"
        Exit Sub
    End If
```

```
    txtReceive(0).Text = ""

    MSComm1(0).Output = txtCommand(0).Text

End Sub
```

이때 포트의 상태를 확인 하는 이유는 운용도중 예기치 못한 문제로 포트가 닫힌걸 감지하기 위함입니다. 만약 이렇게 포트상태를 확인하지 않는다면 포트가 닫힌줄 모르고 Output 명령을 사용하게 되며 이때는 시스템 오류가 발생되어 응용프로그램이 종료가 되기 때문입니다.

다음은 MSComm 의 OnCOmm 이벤트의 소스입니다.

```
Private Sub MSComm1_OnComm(Index As Integer)

    If MSComm1(Index).CommEvent <> comEvReceive Then Exit Sub

    Do
        DoEvents
        If MSComm1(Index).InBufferCount > 0 Then
            txtReceive(Index).Text = txtReceive(Index).Text + MSComm1(Index).Input
        Else
            Exit Do
        End If
    Loop

End Sub
```

위에서 확인 하는 것이 현재 발생한 이벤트가 데이터 수신이벤트가 아니면 함수를 빠져 나가게 만드는 부분입니다.
이 구문의 필요성은 MSComm 이 가지는 17 가지의 이벤트중 데이터 수신 이벤트의 경우에만 데이터를 가져오기 위함입니다.
다른 이벤트가 발생하면 실제 데이터는 들어오지 않았을 가능성이 크므로 불필요한 동작을 줄이자는 의도입니다.

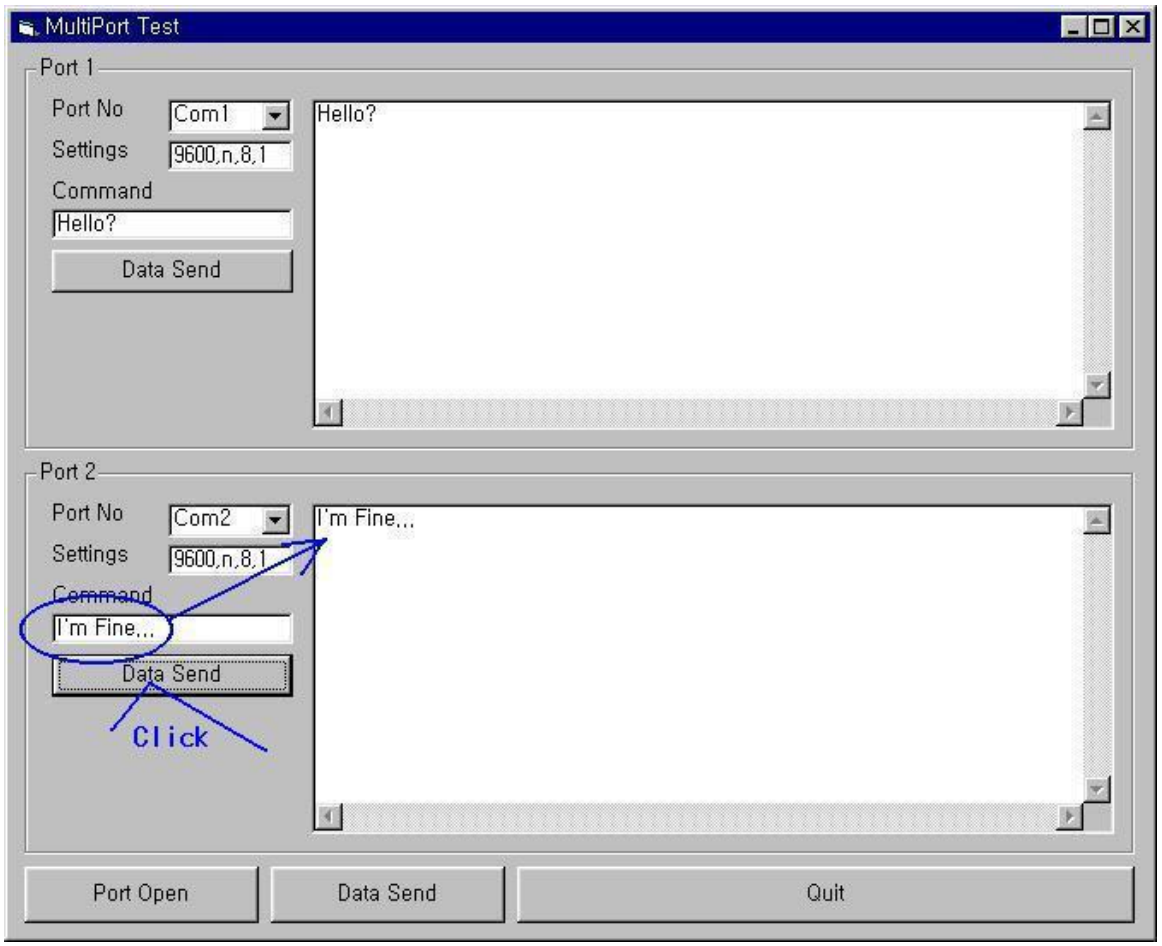
또한 루프를 돌면서 데이터를 가져오는 이유는 데이터가 통신포트로부터 수신버퍼까지 들어오는데 걸리는 시간은 1 바이트당 약 1 밀리초가 걸린다고 설명을 드렸습니다.
따라서 데이터는 한꺼번에 수신버퍼에 들어오는게 아니고 계속 한줄로 줄을 서서 들어오기 때문에 Input 명

령을 사용하는 순간에도 다음 데이터가 들어오고 있다고 봐야되며 이러한 데이터를 모두 가져오기 위한 과정으로 보시면 됩니다.

이번 과정을 성공 하셨다면 이번에는 포트를 옮겨서 시험을 해 보도록 하겠습니다.

아까 Com1 에 연결했던 루프백커넥터를 Com2 로 연결하시기 바랍니다.

그리고 Port2 의 Command 에 'I'm Fine…' 라고 입력 한 다음 아래에 있는 'Data Send'단추를 누르면 다음과 같은 그림이 됩니다.



<< 그림 6 >>

소스는 위 과정과 같으므로 생략 하도록 하겠습니다.

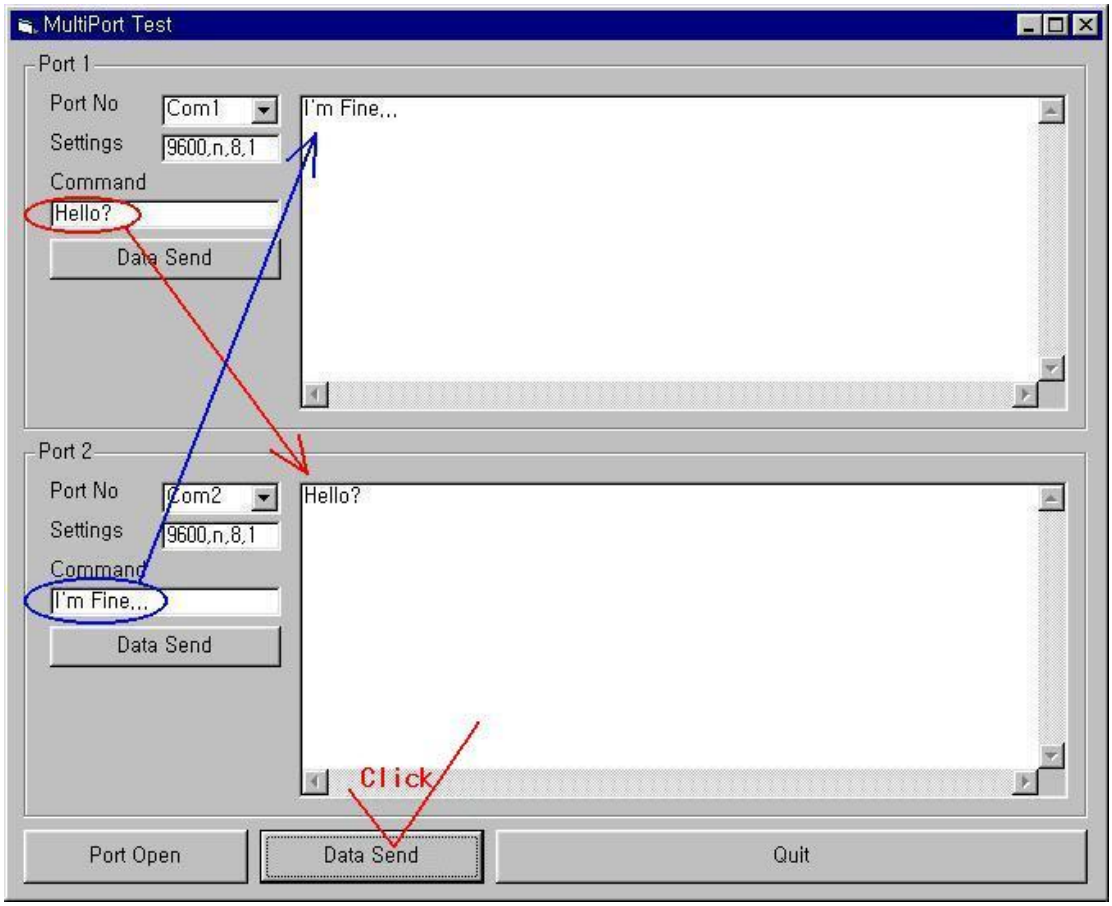
이 두과정을 거치면 각각의 통신포트는 아무런 문제가 없음이 확인된것입니다.

자, 그럼 이번에는 널모뎀 케이블을 이용한 두 포트간 송수신 시험을 해 보도록 하겠습니다.

Com2 에 매달려있는 루프백커넥터를 제거하시고 널모뎀 케이블을 준비하시기 바랍니다.

준비된 널모뎀케이블을 Com1 과 Com2 에 각각 연결하시기 바랍니다.

그리고 이번에는 현재의 화면에서 아래에 있는 'Data Send'단추를 눌러보시기 바랍니다.
그러면 다음과 같은 화면이 됩니다. << 그림 8 >>



그러면 두개의 Command 를 같이 송신하게 됩니다.
물론 의미상으로는 동시이지만 사실은 동시가 아닙니다.
다음의 소스를 보시면..

```
Private Sub cmdDataSend_Click()
    If Not MSComm1(0).PortOpen Then
        MsgBox "Port1 번의 포트가 열리지 않았읍니다.", , "확  인"
        Exit Sub
    End If
    If Not MSComm1(1).PortOpen Then
        MsgBox "Port2 번의 포트가 열리지 않았읍니다.", , "확  인"
        Exit Sub
    End If
```

```
    txtReceive(0).Text = ""
    txtReceive(1).Text = ""

    MSComm1(0).Output = txtCommand(0).Text
    MSComm1(1).Output = txtCommand(1).Text

End Sub
```

각각의 포트 상태를 확인 한 다음 수신창을 클리어시키고나서 Output 명령이 순차적으로 전송이 됩니다.
따라서 두번째 포트에 나가는 데이터는 첫번째 데이터가 나가고 나서 전송이 되므로 약간의 시간지연이 됩니다.
이때 첫번째 포트의 데이터가 통신포트를 통하여 모두 전송이 된이후는 아닙니다.
Output 명령은 송신 버퍼까지의 데이터를 보내는 것으로 목적을 다합니다.
송신버퍼에 들어온 데이터는 통신 드라이브가 한 바이트씩 끄집어내와서 통신 포트에 올려줍니다.
즉,Output 명령이 끝났다고 해서 모든데이타가 통신 포트를 떠난걸로 생각 하지 마시라는 얘기입니다.
Output 명령은 끝났더라도 통신 포트에서는 계속 데이터가 송출될수도 있습니다.

이 경우 여러분들께서 확인 하실수 있는 방법은 다음과 같습니다.

```
MSComm1.Output = String$(1024,"A")

Do
    DoEvents
    Debug.Print MSComm1.OutBufferCount

    If MSComm1.OutBufferCount=0 Then Exit Do
Loop
```

위의 소스를 실행해 보시면 아시겠지만 Output 이 끝나도 Outbuffercount 가 계속 바뀌고 있음을 보실겁니다.
즉 Output 명령은 끝났지만 송신버퍼에서 통신포트로 데이터는 계속 송출이 되고 있다는 얘기이지요.

자,이상으로 여러분들은 RS-232c 를 비베로 주물러 보셨습니다.

물론 지금까지는 맛보기에 불과 합니다.

실제 현업은 이정도의 프로그램으로는 만족할수 없습니다.

하지만 여러분들은 이제 통신포트를 마음대로 가지고 놀기 위한 문을 열고 들어섰으며 남은건 여러분들의 꾸준한 노력으로 이루어 나가셔야 한다는 것입니다.

2) 범용으로 사용가능한 소스설명

지금부터 설명하려는 예제의 소스는 제 홈의 자료실에 있습니다.

http://www.moohantec.com 자료실-기초자료실에 "통신포트관련 소스입니다."라는 이름으로 올려두었습니다.

따라서 여기서는 부분적인 소스에 의한 설명을 드리도록 하겠습니다.

먼저 통신 관련 소스를 설명드리기 전에 이 예제를 사용할 때 필요한 몇가지의 함수에 대한 설명을 드리겠습니다.

소스에 보시면 Comm_Mod.Bas 라는 파일에 있는 함수들입니다.

- 첫번째는 시간 지연 함수입니다.

지연 시키고싶은 시간을 초단위로 지정하시면 됩니다.

만약 2 초를 지연시킬경우 다음과 같이 사용하시면 됩니다.

```
Call Delay(2)
```

그리고 함수 내부를 보시면 함수를 호출하는 시점이 우연히도 자정에 걸쳐서 사용되는 경우를 대비한 코드가 있습니다.

Timer 라는 함수는 하루를 초단위로 환산 시켜 줍니다. 그 값은 0 에서 86399.99 초 입니다.

만약 자정을 통과할 경우 그 값은 다음의 순서로 바뀝니다.

… 86398 , 86399 , 86399.99 , 0 , 1 , 2 , 3 …..

즉, 저녁 11 시 59 분 59 초는 타이머값이 86399 초 입니다.

이때 2 초간의 지연시간을 원할경우 86401 이 될때까지 기다리게 되는데,실제 Timer 함수에는 86401 이라는 수가 리턴될 수가 없으므로 무한루프에 빠지게 됩니다.

이러한 부분을 보완하는 기능을 적용했습니다.

```
Sub Delay(ByVal dTime!)
    Dim oTime!

    oTime! = Timer
    Do
        If Timer < oTime! Then oTime! = oTime! - 86400
        sTime! = Timer - oTime!
        DoEvents
    Loop Until dTime! < sTime!
End Sub
```

- 두번째는 한글 관련 함수입니다.

이 함수들은 한글 문자열 관련 함수들입니다.

문자열의 길이를 구하는 Len 이나 문자열을 자르는 Mid, 그리고 문자열을 검색하는 Instr 함수들이 한글과 영문이 혼합된 상태에서는 정상적인 결과를 돌려주지 않기 때문에 별도로 만든 함수입니다.

```
Function HanLen(Src$) As Integer
    HanLen = LenB(StrConv(Src$, vbFromUnicode))
End Function

Function HanInstr(st%, Src$, chk$) As Long
   HanInstr = InStrB(st%, StrConv(Src$, vbFromUnicode), StrConv(chk$, vbFromUnicode))
End Function

Function HanMid(Src$, ss%, ll%) As String
    Dim cc%, i%, chk$
    Dim tmp$, ll2%

    ss2 = ss - 1
    ll2 = ss2 + ll
    tmp$ = ""
    cc = 1
    ln = 0
    Do
        chk$ = Mid$(Src$, cc, 1)
        If chk$ = "" Then Exit Do
        If Asc(chk$) < 0 Then
            ln = ln + 2
        Else
            ln = ln + 1
        End If

        If ln > ss2 Then
            If ln = ll2 Then
                tmp$ = tmp$ & chk$
                Exit Do
            ElseIf ln > ll2 Then
                tmp$ = tmp$ & " "
                Exit Do
            Else
                tmp$ = tmp$ & chk$
            End If
```

```
        End If
        cc = cc + 1
    Loop

    HanMid = tmp$

End Function
```

- 세번째는 문자열에서 원하는 구분자의 개수를 알아내는 함수입니다.

일정한 구분자를 포함하는 메시지 프레임이 여러 개 일 경우 메시지 개수를 파악하는데 필요한 함수입니다.
만약 캐리지리턴(vbCr)을 구분자로 사용해서 수신된 메시지가 있습니다.
이 메시지에서 캐리지리턴의 개수를 알아내면 루프를 몇번 돌아야 하는지를 알수 있기 때문에 로직이 단순화 될수 있습니다.
여기서 주의하실 사항은 구분자의 개수만 가져오기 때문에 실제 프레임의 개수는 그보다 하나가 많은 수가 될수도 있습니다. 물론 마지막 프레임은 구분자가 없기 때문에 미완성프레임으로 생각하시면 됩니다.

```
Function CountSep(srcData$, chkData$) As Integer
    On Error Resume Next

    cnt% = 0

    g% = 1
    Do
        g% = InStr(g%, srcData$, chkData$)
        If g% > 0 Then
            cnt% = cnt% + 1
            g% = g% + 1
        Else
            Exit Do
        End If
    Loop

    CountSep = cnt%

End Function
```

-네번째는 문자열에서 지정한 구분자를 기준으로 지정한 위치의 데이터를 인출하는 함수입니다.

이 함수의 리턴되는 값은 인출한 자료가 되며 사용되는 인자는 다음과 같습니다.

srcData$ - 원본 문자열

chkData$ - 지정 구분자

PickPoint% - 인출 할 위치

다음의 사용예를 보시기 바랍니다.

SrcStr$ = “123,4,56,7890”

Debug.Print Pickup(srcStr$,”,”,3)

이 코드는 위의 문자열에서 컴마를 기준으로 세번째 블록을 가져오는 예제입니다.

이때 리턴되는 값은 ‘56’이 됩니다.

```
Function Pickup(srcData$, chkData$, PickPoint%) As String
    On Error Resume Next

    srcData1$ = srcData$ & chkData$
    g% = 1
    For i% = 1 To PickPoint% - 1
        g% = HanInstr(g%, srcData1$, chkData$) + 1
    Next i%
    If PickPoint% > 1 And g% = 1 Then
        Pickup = ""
        Exit Function
    Else
        g2% = HanInstr(g%, srcData1$, chkData$)
        Pickup = Trim$(HanMid$(srcData1$, g%, g2% - g%))
    End If
End Function
```

- 마지막으로 문자열 대체 함수입니다.

이 함수의 기능은 지정한 문자열에서 특정문자열 들을 변경하는 함수입니다.

리턴되는 값은 변경된 문자열이며 사용되는 인자는 다음과 같습니다.

Src$ - 원본 문자열

Chr1$ - 변경될 문자열

Chr2$ - 변경할 문자열

이 함수의 용도는 원본 문자열에서 어떤 문자를 다른 문자로 바꾸는 기능이며 중복된 문자를 단일한 문자로 바꾸는 기능도 있습니다.

예를 들어 공백이 겹쳐져 있는 문자열에서는 공백을 구분자로 사용하기가 힘이 듭니다.

따라서 이럴 경우는 먼저 중복 사용된 공백을 단일한 공백으로 치환을 시킨 다음 공백을 구분자로 해서 잘라내면 문제가 해결이 됩니다.
만약 이런 문자열이 있는데.. “123 45 6 789 012”
이 문자열에서는 공백이 중복 사용되었으므로 구분자로 활용할 수는 없지만 다음의 방법으로 치환을 하시면 공백을 구분자로 사용할 수 있는 다음과 같은 문자열이 됩니다.
Debug.Print Replace(SrcStr$,” “,” “)
“123 45 6 789 012”

```
Function Replace$(Src$, chr1$, chr2$)
    Dim Gap%

    If chr1$ = chr2$ Then
        Replace$ = Src$
    Else
        Do
            Gap = HanInstr(1, Src$, chr1$)
            If Gap < 1 Then Exit Do
            If Gap = 1 Then
                Src$ = chr2$ & HanMid$(Src$, Gap + HanLen(chr1$), HanLen(Src$))
            Else
                Src$ = HanMid$(Src$, 1, Gap - 1) & chr2$ & HanMid$(Src$, Gap + HanLen(chr1$), HanLen(Src$))
            End If
        Loop

        Replace$ = Src$
    End If

End Function
```

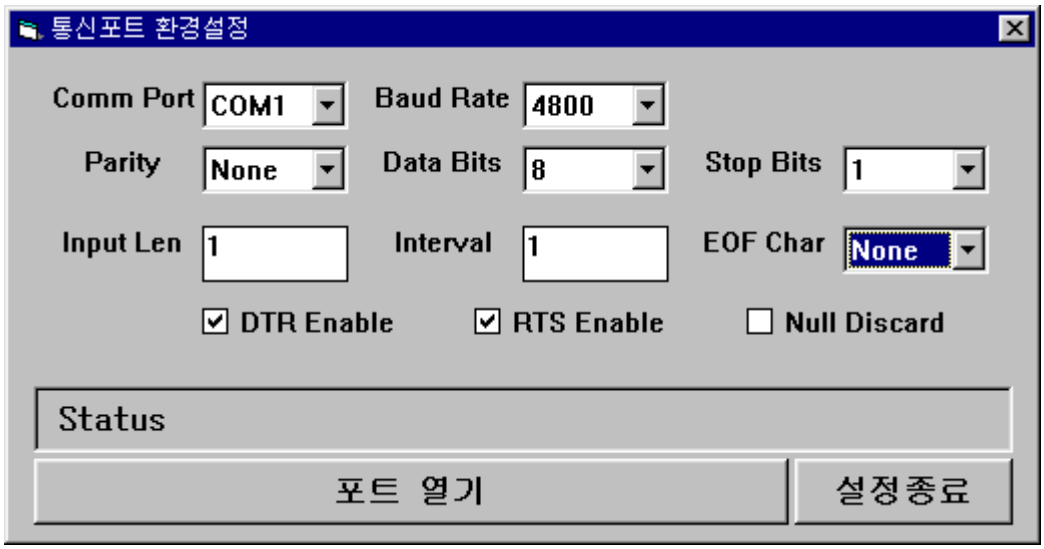
이상으로 공통 모듈의 함수에 대한 설명은 마치도록 하겠습니다.

다음은 이번 강좌에서 설명을 드리려는 소스에 대한 설명을 드리도록 하겠습니다.
먼저 다운 받은 자료를 보시면 폼 두개와 모듈 두개가 있습니다.

frmComm 은 이 프로그램의 기본 폼입니다.
frmSetup 은 통신포트 설정용 폼입니다.

CommTest_Mod 는 위에서 설명드린 공통함수를 가지고 있는 모듈입니다.
CommTest 는 그외의변수 선언용으로 사용할 모듈입니다.

먼저 통신 포트 설정용 화면입니다.



통신포트 및 각종 속성들을 설정할수 있습니다.

핸드쉐이크를 위한 DTR,RTS 신호를 지정할수 있으며 데이터 송/수신시 널(Null)문자를 사용할지의 여부도 지정할수 있습니다.

여기서 설정한 데이터는 레지스트리에 저장을 하여 다시 프로그램이 실행이 되면 이전에 설정 했던값으로 자동 설정되며 폼 로드시에는 포트 열기가 자동 수행 됩니다.

다음의 소스는 포트 열기를 선택할때의 함수입니다.

```
Public Sub cmd232Set_Click()
    ‘포트를 열 때 각종 속성을 설정합니다.
    ‘그러나 포트를 열수 없을때나 속성을 잘못 지정하여
    ‘발생될수 있는 시스템오류를 막기 위한 방법입니다.
    On Error GoTo Error232

    Dim x As Integer
    Dim st$, ef$

    ‘먼저 포트가 열려져 있으면 포트를 닫습니다.
    If frmComm!Comm1.PortOpen = True Then
```

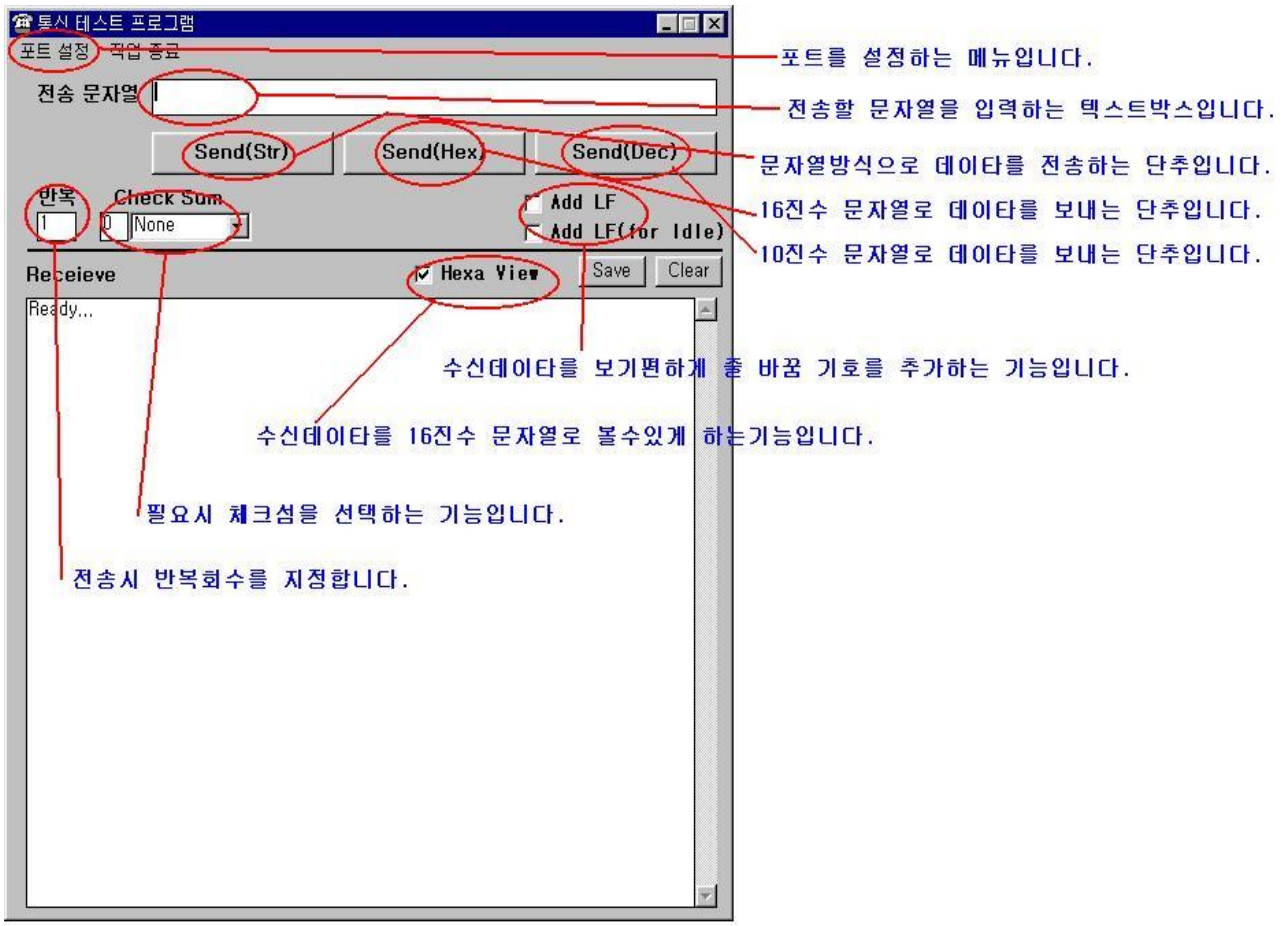
```
    frmComm!Comm1.PortOpen = False
End If

'각종 속성을 설정합니다.
frmComm!Comm1.DTREnable = chk232(0).Value
frmComm!Comm1.RTSEnable = chk232(1).Value
frmComm!Comm1.NullDiscard = chk232(2).Value

frmComm!Comm1.InputLen = Val(txt232(0).Text)
frmComm!Comm1.Interval = Val(txt232(1).Text)

frmComm!Comm1.CommPort = cbo232(0).ListIndex + 1

'Settings 이라는 속성에 들어갈 문자열을 만듭니다.
Select Case cbo232(1).ListIndex
    Case 0: st$ = "19200"
    Case 1: st$ = "9600"
    Case 2: st$ = "4800"
    Case 3: st$ = "2400"
    Case 4: st$ = "1200"
    Case 5: st$ = "600"
End Select
Select Case cbo232(2).ListIndex
    Case 0: st$ = st$ & ",N"
    Case 1: st$ = st$ & ",E"
    Case 2: st$ = st$ & ",O"
    Case 3: st$ = st$ & ",M"
    Case 4: st$ = st$ & ",S"
End Select
Select Case cbo232(3).ListIndex
    Case 0: st$ = st$ & ",8"
    Case 1: st$ = st$ & ",7"
    Case 2: st$ = st$ & ",6"
    Case 3: st$ = st$ & ",5"
    Case 4: st$ = st$ & ",4"
End Select
```

```
Select Case cbo232(4).ListIndex
    Case 0: st$ = st$ & ",1"
    Case 1: st$ = st$ & ",1.5"
    Case 2: st$ = st$ & ",2"
End Select

‘프로그램에서 사용할 종료부호를 결정합니다.
‘이 종료부호는 MSComm 의 속성에는 없으므로 전역변수로 기억하여 데이터를 보낼 때 사용합니다.
Select Case cbo232(5).ListIndex
    Case 1
        eofChar = vbLf
    Case 2
        eofChar = vbCr
    Case 3
        eofChar = vbCrLf
    Case Else
        eofChar = ""
End Select

설정한 각 속성을 레지스트리에 저장합니다.
Call SaveSetting("COMMTEST", "PORT", "DTR", chk232(0).Value)
Call SaveSetting("COMMTEST", "PORT", "RTS", chk232(1).Value)
Call SaveSetting("COMMTEST", "PORT", "NULL", chk232(2).Value)

Call SaveSetting("COMMTEST", "PORT", "LEN", txt232(0).Text)
Call SaveSetting("COMMTEST", "PORT", "INT", txt232(1).Text)

Call SaveSetting("COMMTEST", "PORT", "NUMBER", cbo232(0).ListIndex)
Call SaveSetting("COMMTEST", "PORT", "BAUD", cbo232(1).ListIndex)
Call SaveSetting("COMMTEST", "PORT", "PARITY", cbo232(2).ListIndex)
Call SaveSetting("COMMTEST", "PORT", "DATA", cbo232(3).ListIndex)
Call SaveSetting("COMMTEST", "PORT", "STOP", cbo232(4).ListIndex)
Call SaveSetting("COMMTEST", "PORT", "EOF", cbo232(5).ListIndex)

‘포트 열기를 시도합니다.
frmComm!Comm1.Settings = st$
```

```
    frmComm!Comm1.PortOpen = True
    pan232Status.Caption = "통신포트가 열렸읍니다."

end232:
    '만약 포트가 잘 열렸으면 종료합니다.
    Exit Sub

Error232:
    '포트가 열리지 않으면 사용자에게 메시지를 보여주고 다시 시도하게 합니다.
    pan232Status.Caption = "통신포트를 열수가 없읍니다."
    Resume end232

End Sub
```

다음은 이 프로그램의 주 화면입니다.

각각의 콘트롤에 대한 기능과 내용은 다음과 같습니다.



지금 설명드리려는 소스는 데이터를 전송하는 세개의 단추와 데이터 수신시의 처리 부분에 대한 설명 입니다.
그 외의 소스는 이전 강좌에서 기본적으로 다루었으므로 이번에는 그런 내용들은 생략을 하도록 하겠습니다.

먼저 Send(Str)이라는 단추의 내용입니다.

```
Private Sub cmdSend_Click()
    Dim tmp%
    Dim sBuf$, tBuf$

    ‘전송시 반복회수를 가져옵니다. 만약 지정되지 않았으면 기본적으로 1 회 전송합니다.
    Repeat = Val(txtRepeat.Text)
    If Repeat < 1 Then Repeat = 1

    ‘전송할 문자열을 가져옵니다.
    sBuf = txtSend.Text

    If Trim$(sBuf) <> "" Then
        ‘아래의 소스블럭은 전송 문자열중 제어코드를 이용하기 위한 기능입니다.
        sBuf = Replace$(sBuf, "#STX", STX)
        sBuf = Replace$(sBuf, "#EOT", EOT)
        sBuf = Replace$(sBuf, "#ENQ", ENQ)
        sBuf = Replace$(sBuf, "#ACK", ACK)
        sBuf = Replace$(sBuf, "#NAK", NAK)
        sBuf = Replace$(sBuf, "#ETX", ETX)
        sBuf = Replace$(sBuf, "#ESC", ESC)

        sBuf = Replace$(sBuf, "#TAB", vbTab)
        sBuf = Replace$(sBuf, "#CR", vbCr)
        sBuf = Replace$(sBuf, "#LF", vbLf)
        sBuf = Replace$(sBuf, "#CRLF", vbCrLf)

        ‘체크섬을 지정한 경우 체크섬 데이터를 만드는 과정입니다.
        Select Case cboCheckSum.ListIndex
            Case 0    'None
            Case 1    'Add 1Byte
```

```
                cs = 0
                For i = Val(txtCSPos.Text) To Len(sBuf)
                    cs = cs + Asc(Mid$(sBuf, i, 1))
                Next i
                cs = cs And &HFF
                sBuf = sBuf & Chr$(cs)
            Case 2    'Add 2Byte
            Case 3    'XOR 1Byte
                cs = 0
                For i = Val(txtCSPos.Text) To Len(sBuf)
                    cs = cs Xor Asc(Mid$(sBuf, i, 1))
                Next i
                cs = cs And &HFF
                sBuf = sBuf & Chr$(cs)
            Case 4    'XOR 2Byte
        End Select

        '미리 정의된 종료부호를 추가하여 반복 회수 만큼 데이터를 전송합니다.
        For r = 1 To Repeat
            Comm1.Output = sBuf & eofChar
        Next r

    End If

End Sub
```

이 함수에서 사용한 기능중 제어코드사용에 대해서 설명 드리겠습니다.
문자열 방식으로 데이터를 보내기 때문에 'Hello?'라든지 'I am a Korean.'같은 일반 Ascii Code 영역의 데이터를 입력 하시면 됩니다.

그러나 시험하시려는 장비가 특정 프레임을 요구 하고 있다면,그것도 제어코드를 사용해서 통신하기를 원하고 있다면 문자열 전송 방식으로는 문제가 발생합니다.
STX(Ascii Code 2)나 ETX (Ascii Code 3) 과 같은 코드를 문자열형식으로 입력할수 없기 때문입니다.
따라서 이 프로그램에서는 이런 제어코드를 미리 정의하고 사용할수 있는 기능으로 되어있습니다.
만약 송신 프레임이 STX + Data + ETX 구조로 되어있고 Data 에 Hello? 라고 보내야 한다고 가정하도록 하겠습니다.

먼저 송신할 데이터의 데이터를 16 진수 문자열로 보면 다음과 같습니다.
02(STX) + 48(H) + 65(e) + 6C(l) + 6C(l) + 6F(o) + 3F(?) + 03(ETX)
이때 Hello?라는 일반 문자는 입력이 쉽지만 아스키코드 2 와 3 에 해당하는 STX,ETX 의 처리가 곤란해집니다.
이때는 다음과 같이 입력하시면 됩니다.
#STXHello?#ETX

그러면 함수의 초기에 있던 변환루틴에서 #으로 시작하는 문자중 약속된 제어코드를 해당 아스키코드로 변환시키므로 원하는대로 데이터가 전송이 됩니다.
이때 여러분들께서 원하는 다른 제어코드가 있으면 얼마든지 추가해서 사용하실수 있습니다.

그리고 체크섬의 경우 간단한 체크섬의 경우만 예를들었습니다.
단순하지만 이런식으로 체크섬을 만든다는 참고용으로 생각하시면 되겠습니다.

다음은 Send(Hex)라는 단추의 내용입니다.

```
Private Sub cmdHexSend_Click()
    ‘먼저 첨자가 하나인 바이트 배열을 선언합니다.
    ‘이 바이트 배열이 필요한 이유는 앞의 강좌에서 충분히 이해 하셨으리라 생각됩니다.
    Dim oBuf(0) As Byte

    Dim tmp$
    Dim ttt$, i%, J%, K%
    Dim cnt%
    Dim sBuf$

    On Error Resume Next

    ‘반복 전송회수를 가져옵니다.
    Repeat = Val(txtRepeat.Text)
    If Repeat < 1 Then Repeat = 1

    If Trim$(txtSend.Text) <> "" Then
        ‘이번함수는 공백을 기준으로 입력한 문자열을 16 진수 아스키 코드로 보기 위한 함수입니다.
        ‘따라서 공백의 중복 사용에 따르는 문제를 미리 해결하기 위해서
        ‘ 이중 공백을 단일 공백으로 치환합니다.
        tmp$ = Replace$(txtSend.Text, "  ", " ")
```

```
        For r = 1 To Repeat
            ‘전송 문자열중 구분자로 사용되는 공백의 개수를 카운트합니다.
            ‘카운트값에 더하기 1 을 하는건
            ‘구분자의 개수는 구분된 블록보다 하나가 작게 나오기 때문입니다.
            cnt = CountSep(tmp$, " ") + 1

            sBuf = ""

            ‘공백으로 구분된 코드 블록만큼 루프를 수행합니다.
            For i = 1 To cnt
                ‘전체 전송 문자열중에서 현재위치의 코드를 추출합니다.
                ttt$ = Pickup$(tmp$, " ", i)

                If ttt$ = "," Then
                    ‘만약 추출한 코드가 컴마이면 약 0.2 초의 지연시간을 갖습니다.
                    ‘이는 모뎀 이용시 내선에서 국선으로 전환할 때 사용하는 컴마의 기능을 이용한방법
입니다.
                    Call Delay(0.2)
                ElseIf Left$(ttt$, 1) = "#" Then
                    ‘만약 추출한 코드가 #으로 시작하면 그 데이터는 코드가 아닌 문자열을 의미합니다.
                    ‘따라서 각 바이트를 추출하여 해당 아스키코드를 변환하여 전송합니다.
                    For J = 2 To Len(ttt$)
                        oBuf(0) = Asc(Mid$(ttt$, J, 1))
                        Comm1.Output = oBuf
                    Next J
                ElseIf Len(ttt$) <> 0 Then
                    ‘추출한 데이터는 16 진수 아스키 코드이므로 숫자화 시켜서 바이트배열에 담고 전송
합니다.
                    oBuf(0) = Val("&h" & ttt$)
                    Comm1.Output = oBuf
                End If
            Next i
        Next r
    End If
```

End Sub

이 함수는 기본적으로는 전송할 데이터의 아스키코드를 16 진수 문자열형태로 입력을 하기 위해서 만들었습니다.

즉,&hF1 부터 &hFF 까지 전송을 원하시면 다음과 같이 입력하신다음 전송하시면 됩니다.
F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

그런데 전송 데이터중 일반 문자열이 길게 있다면 이들의 아스키값을 일일이 찾아서 16 진수로 만들어서 입력을 해야하는데 이방법이 귀찮을때가 있습니다.
다시 위의 예제로 확인 해보겠습니다.
STX + Hello? + ETX 의 구조로 데이터를 보내시려면 다음과 같이 보내시면 됩니다.
02 48 65 6C 6C 6F 3F 03
하지만 이 경우 문자열의 아스키 코드를 일일이 찾지 않고 다음과 같이 입력하셔도 결과는 같습니다.
02 #Hello? 03
첫번째 블록과 세번째 블록은 아스키 코드값이므로 바로 데이터가 전송이 됩니다.
그러나 두번째 블록은 #으로 시작하므로 #다음부터의 문자열을 바이트단위로 아스키코드를 추출해서 데이터를 전송하므로 결과는 같아지게 됩니다.

다음은 Send(Dec)의 기능입니다.

```
Private Sub cmdDecSend_Click()
    Dim oBuf(0) As Byte

    Dim tmp$
    Dim ttt$, i%, J%
    Dim cnt%

    On Error Resume Next
    Repeat = Val(txtRepeat.Text)
    If Repeat < 1 Then Repeat = 1

    If Trim$(txtSend.Text) <> "" Then

        tmp$ = Replace$(txtSend.Text, "    ", " ")

        For r = 1 To Repeat
```

```
            cnt = CountSep(tmp$, " ") + 1
            For i = 1 To cnt
                ttt$ = Pickup(tmp$, " ", i)

                If ttt$ = "," Then
                    Call Delay(0.2)
                ElseIf Left$(ttt$, 1) = "#" Then
                    For J = 2 To Len(ttt$)
                        oBuf(0) = Asc(Mid$(ttt$, J, 1))
                        Comm1.Output = oBuf
                    Next J
                Else
                    oBuf(0) = Val(ttt$)
                    Comm1.Output = oBuf
                End If
            Next i
            Comm1.Output = eofChar
        Next r

    End If

End Sub
```

이 함수는 위의 16 진수 문자열 전송과 거의 같은 기능입니다.

경우에 따라서는 전송하려는 데이터가 16 진수로 변환하기 귀찮은 데이터의 경우 10 진수 코드열로 만들어서 전송할수 있도록 구성한 것입니다.

16 진수문자열과 10 진수 문자열의 차이점은 단지 해당 코드를 진법에 맞게 변환시키는 과정뿐입니다.
16 진수 코드로 구성되어있을 경우는 다음과 같이 변환하시면 됩니다.

```
oBuf(0) = Val("&h" & ttt$)
```

그리고 10 진수 코드값으로 구성되어있을 경우는 다음과 같이 사용하시면 됩니다.

```
oBuf(0) = Val(ttt$)
```

두 경우의 차이점은 아시리라 생각됩니다.

다음은 MSComm 의 데이터 수신부분입니다.
이 예제에서는 폴링이 아닌 인터럽트 방식을 사용하였고 따라서 MSComm 의 속성중 RThreshold 가 1 로 설정되어있습니다.
그리고 데이터 수신시 1 바이트씩 받도록 구상했으므로 포트를 열 때 속성중 InputLen 은 1 로 설정 하셔야 합니다.

```
Private Sub Comm1_OnComm()
    Dim tmp$, a$, b$
    Dim rBuf      As Byte
    Dim i, J

    ‘지금 발생한 이벤트가 데이터수신이벤트가 아니라면 함수를 빠져나갑니다.
    If Comm1.CommEvent <> comEvReceive Then Exit Sub

    ‘지금부터 데이터를 처리하는 동안에는 이벤트가 발생되지 않도록 지정합니다.
    Comm1.RThreshold = 0

    ‘데이터 수신시 수신블럭간의 갭을 무시하기 위해서 루프를 수행합니다.
    ‘하지만 이 과정은 사용되는 컴퓨터의 프로세서마다 속도가 달리작용이 되므로
    ‘그리 좋은 코드로 볼 수는 없습니다.
    ‘그러나 이런식으로도 데이터를 받을수 있다는 참고용으로 생각 하시기 바랍니다.
    For i = 1 To 5
        DoEvents
        ‘수신된 데이터가 있으면…
        If Comm1.InBufferCount > 0 Then
            ‘화면에서 16 진수 보기가 체크되어있으면..
            If chkHex.Value = 1 Then
                ‘수신된 데이터수만큼 루프를 실행합니다.
                For J = 1 To Comm1.InBufferCount
                    ‘수신버퍼에서 한바이트 데이터를인출합니다.
                    rBuf = AscB(Comm1.Input)

                    ‘수신창에 수신한 데이터를 16 진수 문자열로 바꾸어 보여줍니다.
                    txtRecv.Text = txtRecv.Text & Right$("00" & Hex$(rBuf), 2) & " "
                    ‘만약 수신한 데이터가 캐리지 리턴이라면
                    If Chr$(rBuf) = vbCr Then
```

```
                    ‘그리고 여러줄 보기가 설정 되었다면..
                    If chkMultiLine.Value = True Then
                        ‘수신데이타창에 줄바꿈기호를 더해줍니다.
                        txtRecv.Text = txtRecv.Text & vbCrLf
                    End If
                End If
            Next J
        Else
            ’16 진수 보기가 체크되어 있지 않은 경우에는..
            ‘수신버퍼에서 데이터를 한바이트 인출합니다.
            rBuf = AscB(Comm1.Input)

            ‘수신창에 수신한 데이터에 해당하는 문자를 표시합니다.
            txtRecv.Text = txtRecv.Text & Chr$(rBuf)

            ‘만약 수신한 데이터가 캐리지 리턴이라면..
            If Chr$(rBuf) = vbCr Then
                ‘그리고 여려줄 보기가 설정 되었다면..
                If chkMultiLine.Value = True Then
                    ‘수신데이타창에 줄바꿈 기호를 더해줍니다.
                    txtRecv.Text = txtRecv.Text & vbLf
                End If
            End If
        End If

        ‘수신창의 커서위치를 맨 마지막으로 지정합니다.
        txtRecv.SelStart = 65535
        DoEvents

        ‘그리고 For 문의 변수를 초기화 합니다.
        ‘ 이 방법은 데이터 수신시 발생할수 있는 수신 블록간의 갭을 처리하기 위한 기능입니다.
        ‘즉,데이터를 수신중이라면 계속 For 문이 처음부터 돌게 됩니다.
        ‘이렇게 되면 루프를 빠져나가는 경우는  데이터가 들어오지 않는 상태에서 루프가 다섯번 돌게되면 빠져나갈수 있습니다.
        i = 1
    Else
```

```
            DoEvents
        End If

        ‘수신데이타를 표시하는 텍스트 박스의 용량 때문에 사용한 코드입니다.
        ‘일반 텍스트박스는 최대 저장용량이 32767 바이트입니다.
        ‘ 그런데 계속 데이터를 누적하다보면 이 용량을 넘게되고 그때는 시스템 오류가 발생합니다.
        ‘따라서 텍스트박스의 용량을 확인 하다가 30000 바이트가 넘으면 앞의 10000 바이트를 제거하여 오류를 미리 막는 기능입니다.
        If Len(txtRecv.Text) > 30000 Then
            txtRecv.Text = Mid$(txtRecv.Text, 20000)
        End If
    Next i

    ‘Idle 시 여러줄 보기가 설정 되어있으면 데이터 수신후 줄바꿈 기호를 추가시켜줍니다.
    If chkMultiLineIDLE.Value = True Then
        txtRecv.Text = txtRecv.Text & vbCrLf
    End If
    ‘그리고 커서의 위치를 텍스트 박스의 맨 마지막에 설정합니다.
    txtRecv.SelStart = 65535

    ‘데이터 수신이 끝났으므로 다시 수신 이벤트를 감지하기 위해서 설정합니다.
    Comm1.RThreshold = 1

End Sub
```

이상으로 이번 강좌는 모두 마치도록 하겠습니다.

이제는 여러분들도 통신 포트에 대해서 나름대로의 자신감을 가지셨으리라 생각됩니다.

다음 강좌는 통신 포트를 이용한 실제 프로젝트를 해볼까 합니다.

많이 기대 해 주시길..