최 종
연구보고서

농산물 도매시장의 효율적 경매를 위한 화상 시스템이 도입된 전자식 경매장치의 개발

The development of electronic auction apparatus equipped with the picture system for efficient auction at agricultural products wholesale market

연구기관

조 선 대 학 교

농 림 부

제 출 문

농림부 장관 귀하

본 보고서를 "농산물 도매시장의 효율적 경매를 위한 화상시스템이 도입된 전자식 경매장치의 개발" 과제 최종보고서로 제출합니다.

1998 . 11 . 30 .

주관연구기관명 : 조선대학교

총괄연구책임자 : 최 한 수

연 구 원 : 김 영 동

연 구 원 : 송 철

요 약 문

Ⅰ. 제 목

농산물 도매시장의 효율적 경매를 위한 화상시스템이 도입된 전자식 경매장치의 개발

Ⅱ. 연구개발의 목적 및 중요성

UR협상의 타결로 인해 모든 농산물의 수입을 전면적으로 개방하지 않을 수 없게된 이 시점에서 우리의 농업은 위기에 직면하게 되었다. 이와 같은 상황에서 우리의 농업을 살리는 길은 우리 농산물의 국제 경쟁력을 강화시키는 것이며, 이를 위해 생산과정에 필요한 연구개발은 물론 유통과정의 현대화 역시 생산성의 향상을 위해 필요한 요소 중의 하나이다. 농민에 의해 생산된 농산물의 첫번째 유통과정은 도매시장의 경매이며, 다음과 같이 개정된 농안법의 발효로 인해 보다 개선된 전자식 경매시스템의 개발이 요구되어 진다.

[농안법 30조 3항] (1994. 11. 1 부터 시행)

경매 또는 입찰의 방법은 전자식을 원칙으로 하되 필요한 경우 농림수산부 령이 정하는 바에 의하여 거수수지식, 기록식, 서면입찰식 등의 방법으로 행할 수 있으며 공개경매 실현을 위하여 필요한 경우에 농림수산부장관 또는 개설자는 품목별, 도매시장별로 경매방식을 제한할 수 있다.

도매시장에서의 기술적, 경제적 문제점의 해소를 위해 효율적이고 보다 개선된 경매장치를 개발할 목적으로 본 연구를 수행하였다.

Ⅲ. 연구개발 내용 및 범위

1. 단말기의 설계 및 개발

   단말기 key의 기능, 단말기의 회로구성, 단말기의 하드웨어 제어프로그램

2. 단말기-컴퓨터간 인터페이스 개발

   인터페이스 회로설계, 인터페이스 통신프로그램, 컴퓨터의 직렬통신 프로그램

3. TV수상기-컴퓨터간 인터페이스구성

   인터페이스 구성, 구동용 프로그램

4. 측정장치-컴퓨터간 인터페이스 구성

   인터페이스 구성, 인터페이스 제어프로그램 개발, 당도·산도·수분·중량값 디스플레이 프로그램 개발

5. 경매프로그램 개발

   메뉴화면, 경매내용 입력화면, 경매진행 화면, 중매인관리 화면, 출하자 관리 화면, 화면이용 방법

6. 화면구성

   단말기 LCD 화면구성, 컴퓨터 모니터 화면구성 및 TV수상기 화면구성

Ⅳ. 연구개발결과 및 활용에 대한 건의

1. 단말기, 단말기-컴퓨터간 인터페이스 개발결과

가. 단말기와 컴퓨터간 인터페이스 장치를 양방향 통신이 가능하도록 설계하여, 대화식으로 중매인의 고유번호, 비밀번호, 응찰가를 입력할 수 있도록 구성하므로써 사용자의 편의를 도모하였다.

나. 경매를 시작하기 전에 경매할 상품들에 대한 사양 즉 품목, 품종, 산지,

출하자, 수량, 등급 등과 같은 정보를 경매사는 미리 컴퓨터에 입력시키고 이러한 정보들을 단말기의 LCD(Liquid Cristal Display, 표시용량 - 숫자: 20×16, 한글: 10×8, 영문: 20×16)에 나타낼 수 있게 하므로써 경매가 시작되기 전에 구매하고자 하는 품목과 응찰가 등을 결정하기 위한 정보를 중매인에게 제공할 수 있다.

다. 낙찰·유찰의 유무, 낙찰가, 낙찰자 등과 같은 전광판을 보고 알 수 있는 정보를 컴퓨터로 부터 전송받아 단말기의 LCD에 나타나게 하므로써 단말기만으로도 경매진행 상황파악이 가능하도록 하였다.

라. 본 연구에서 개발한 단말기는 정보를 단말기에서 컴퓨터로 전송하는 기능과 컴퓨터에 수록된 정보를 단말기로 전송 받을 수 있는 기능을 갖고 있으며, 표시판으로 숫자와 문자정보를 나타낼 수 있는 LCD를 사용하므로 전광판이 없어도 경매를 진행할 수 있어서 저렴한 가격으로 전자식 경매시스템을 설치할 수 있다.

마. 바퀴가 부착된 구조물 위에 단말기와 컴퓨터(노트북 PC)를 탑재하고 도매시장내에 상품이 놓여있는 곳으로 구조물을 이동하거나, 농산물이 놓여있는 장소로 경매사는 노트북 PC를, 중매인들은 단말기를 지니고 다니면서 설치된 접속단자에 컴퓨터와 단말기를 접속시키면 경매가 가능하기 때문에 고정식은 물론이고 단위 품목당 중매인의 수가 많지 않은 중소규모의 도매시장에 이동식 전자 경매시스템으로 사용할 수 있는 부수적 효과를 얻을 수 있다.

2. TV수상기-컴퓨터간 인터페이스 구성결과

가. 화상시스템은 농산물의 질을 평가할 수 있는 외적인 정보(크기, 색상, 반점 등)를 비디오카메라를 통해 제공하고 화면은 전광판이 아닌 TV수

상기를 사용하기 때문에 많은 양의 문자 및 숫자정보와 고화질의 영상정보를 제공하면서도 설치비용이 저렴하다.

나. 도매시장의 규모에 따라 TV수상기의 설치대수만 조정하면 되므로 소규모 도매시장에서 대규모 도매시장에 이르기까지 모두 적용할 수 있다.

3. 측정장치-컴퓨터간 인터페이스 구성결과

가. 각종 측정장치에 의해 농산물의 내용 즉 당도, 산도, 수분, 무게 등과 같은 내적 정보를 제공하므로써 중매인에게 상품의 질을 판정할 수 있는 기회를 부여한다.

나. 농산물의 품질 확인으로 등급 판정에 대한 객관성을 확보할 수 있다.

4. 경매프로그램 개발결과

메뉴화면, 경매내용 입력화면, 경매진행화면 등과 같은 경매프로그램을 개발하여 경매진행에 대한 효율을 높일 수 있었으며, 중매인 관리화면, 출하자 관리화면을 통해 중매인과 출하자의 성명, 고유번호, 상호, 주소, 전화번호, 주민등록번호 등을 입력하여 경매결과를 전산처리하기 위한 자료를 제공할 수 있다.

5. 화면구성 결과

가. 단말기의 LCD화면을 대화식으로 구성하여 중매인들에게 편의를 제공하였다.

나. 상장번호와 상품에 대한 품목, 품종, 수량, 등급, 출하자, 산지 등을 단

말기의 화면에 나타내줌으로써 단말기를 통해서도 상품의 정보를 알 수 있게하였다.

다. 단말기의 화면에 자신이 입력한 응찰가와 최고응찰가를 나타내고, 경매결과에 대한 정보인 낙찰가와 낙찰자 등을 나타내줌으로써 단말기만을 주시하여도 경매진행이 가능하도록 하였다.

라. 경매사가 사용하는 컴퓨터의 모니터에는 상품정보(상장번호, 품목, 품종, 수량, 등급, 판매단위, 출하자, 산지)와 내적정보(당도, 산도, 수분, 중량)를 수록하여 경매진행의 주최인 경매사에게 다양한 상품정보를 제공하였다.

마. 컴퓨터의 모니터에는 또 다음 경매품목에 대한 사양, 경매진행 시간, 해당 품목에 대해 응찰한 중매인들의 수를 파악할 수 있는 접속율, 중매인들이 입력한 응찰가 중 최고가, 최고응찰가를 입력한 중매인의 고유번호(ID)를 나타내고 경매진행을 위해 경매사가 시작, 낙찰, 유찰, 다음, 종료항을 마우스로 클릭하거나 컴퓨터의 키보드상에 있는 기능키들 중 F1, F2, F3, F4, F5 키를 선택할 수 있도록 구성하므로써 경매사로 하여금 효율성있는 경매진행이 가능하도록 하였다

6. 활용방안

가. 청과 도매시장, 야채 도매시장, 화훼 도매시장, 수산 도매시장, 축산 도매시장 등을 비롯한 모든 농축수산물 도매시장의 전자식 경매시스템으로 활용.

나. 중소형 도매시장에서 이동식 경매 장치로 활용 가능.

SUMMARY

In this research, we develop the new auction system to improve the fairness between consumer and supplier. We develop image electronic auction system which is mainly divided by communication unit, image processing unit and measuring unit.

With communication unit we can collect the price data from the many input devices(keypads). The communication unit consists of the keypad, information interface units and PC system. The auction manager controls the whole program with host PC system and sends the information signal to interface unit. Interface unit transfers the received information to keypad. Through the graphic screen of keypad, the commission merchant can easily see the information like the grade of goods, the current progress status, and so on. And the purchasing price of agricultural products is inputted to keypad.

Measuring unit, which is consists of four measuring instruments (sugar density, acidity, moisture, weight), can improve the practical decision and prevents goods from scratch or injury by commission merchant to define the grade. Measuring devices send informations of goods to main PC. All of the information are collected by PC system and are displayed monitor with image of goods.

Through our research, we can improve the efficiency. So we can protect farmer for damaging by the unjust price . And electronic auction system will also make more reasonable as referring the database file.

CONTENTS

목 차

제 1 장 연구개발의 목적 및 중요성

제1절 연구배경

UR협상의 타결로 인해 모든 농산물의 수입을 전면적으로 개방하지 않을 수 없게된 이 시점에서 우리의 농업은 위기에 직면하게 되었다. 이와 같은 상황에서 우리의 농업을 살리는 길은 우리 농산물의 국제 경쟁력을 강화시키는 것이며, 이를 위해 생산과정에 필요한 연구개발은 물론 유통과정의 현대화 역시 생산성 향상을 위해 필요한 요소 중의 하나이다. 농민에 의해 생산된 농산물의 첫번째 유통과정은 도매시장의 경매이며, 다음과 같이 개정된 농안법의 발효로 인해 보다 개선된 전자식 경매시스템의 개발이 요구되어 진다.

[농안법 30조 3항] (1994. 11. 1 부터 시행)

경매 또는 입찰의 방법은 전자식을 원칙으로 하되 필요한 경우 농림수산부 령이 정하는 바에 의하여 거수수지식, 기록식, 서면입찰식 등의 방법으로 행할 수 있으며 공개경매 실현을 위하여 필요한 경우에 농림수산부장관 또는 개설자는 품목별, 도매시장별로 경매방식을 제한할 수 있다.

1. 수지식 경매의 문제점

현재 우리나라 도매시장에서 주로 사용되어지는 수지식 경매방법은 다음과 같은 문제점을 안고 있다.

1) 경매는 중매인과 경매사 간의 수화에 의해 이루어지는데 농산물의 소

유자인 농민은 수화를 모르기 때문에 그 진행과정을 이해할 수 없다. 따라서 낙찰가에 의구심을 가질 수 밖에 없으며 공정성 여부에 대한 의혹이 제기될 수 있어서, 생산 농민에게 신뢰성있는 경매 상황 및 결과의 전달이 어렵다.

2) 중매인이 도매인을 겸할 수 있는 현행 제도에서는 중매인에 의한 농산물의 가격 담합 가능성에 의한 농민의 피해 또한 예상되어지는 문제점이며, 수화에 익숙하지 못한 매매참가인의 참여가 불가능하다.

3) 경매가 진행되는 동안 경매사의 끊임없는 큰 음성에 의한 소음공해 역시 문제점 중의 하나이며, 음성의 혼선으로 인하여 인접한 장소에서 다른 품목에 대한 경매가 동시에 이루어 질 수 없다.

4) 수지식 방법은 종래의 관계 법규에서는 문제가 없었으나, 다수의 매매참가인을 허용하고 있는 개정된 농안법 하에서는 한계성이 대두된다.

2. 기존의 전자식 경매시스템의 문제점

상술한 바와 같은 수지식의 문제점을 해소하기 위해 전자식 경매시스템을 도입할 필요가 있으며 국내 소수의 도매시장에서 사용되고 있는 기존의 전자식 경매시스템은 다음과 같은 문제점이 있다.

1) 기존의 단말기는 중매인이 입력한 응찰가만을 즉 한줄의 숫자 정보만을 단말기의 표시판에 나타낼 수 있도록 되어있다.

2) 단말기에서 컴퓨터로 중매인의 고유번호와 응찰가를 보내기만하고 컴퓨터에 수록된 정보를 단말기로 전송받지 못하는 일방향 통신만 할 수 있다.

3) 국내의 도매시장에 설치된 전자식 방법은 전광판을 화면으로 사용하고 있다. 전광판은 고가이면서도 나타낼 수 있는 정보량이 매우 적고 정밀한

화상을 나타낼 수 없다.

4) 농산물의 품질 확인 과정에서 객관성을 확보할 수 없다. 즉 농산물의 질을 측정장비에 의해 측정할 수 있는 기능과 농산물의 외관을 나타내는 시설인 화상처리 장치가 없다.

제2절 연구개발의 목적

상술한 바와 같은 도매시장에서의 기술적, 경제적 문제점의 해소를 위해 효율적이고 보다 개선된 경매장치를 개발할 목적으로 본 연구를 수행하였다.

본 연구는 화상 시스템을 이용하여 경매방법의 효율성을 높이는 방안으로써 경매하고자 하는 농산물에 대한 사양-당도, 산도, 수분, 무게, 크기, 색상, 반점, 흠집 등-을 측정기와 비디오카메라로 측정한 내용과, 농산물에 대한 정보-품목, 품종, 수량, 등급, 출하자, 산지 등-를 대형 TV수상기와 단말기의 화면에 나타내면 중매인 또는 매매참가인은 그 정보에 의해 농산물에 대한 내용을 파악하고 경매에 참여하게 된다. 중매인 또는 매매참가인은 응찰가를 단말기를 통해 입력하면 자신의 응찰가와 최고 응찰가격이 TV수상기와 단말기 화면에 나타나게 되고 경쟁적 재응찰을 거듭하여 최고 응찰가를 낙찰가로 결정해 주는 방식이다.

본 연구에서는 다음과 같은 목표로 경매시스템을 개발하므로써 기존의 전자식 경매시스템에 비해 효율성과 경제성을 높이고자 한다.

1) 단말기의 표시부로 넓은 화면을 갖는 LCD(Liquid Crystal Display : 액정화면, 표시용량 - 숫자: 20×16, 한글: 10×8, 영문: 20×16))를 이용하여 많은 문자와 숫자 정보를 수용할 수 있도록 설계한다.

2) 단말기-컴퓨터간 인터페이스를 단말기의 정보가 컴퓨터로 전송되고

컴퓨터의 정보 역시 단말기로 전송하기 위해 양방향 통신이 가능하도록 설계한다.

3) 당도, 산도, 수분, 무게 등을 측정하는 측정장치들을 컴퓨터에 인터페이스하여 측정치들을 컴퓨터의 모니터, 단말기의 LCD, TV수상기에 디스플레이하므로써 농산물의 내용 파악에 대한 공정성을 기할 수 있도록 한다.

4) 경매에 필요한 농산물에 대한 다양한 정보(품목, 품종, 수량, 등급, 출하자, 산지, 당도, 무게 등)를 단말기의 화면인 LCD에 나타내어 중매인에게 응찰가를 결정할 수 있는 정보를 제공하고, 경매진행 중에는 자신의 응찰가와 최고응찰가를 동시에 LCD에 나타내도록 하여 단말기만을 주시하여도 경매진행이 가능하도록 한다.

5) 국내에서 사용 중에 있는 기존의 전자식 시스템은 화면으로 전광판을 사용하고 있는데 전광판은 고가이면서도 나타낼 수 있는 정보량이 매우 적고 정밀한 화상을 나타낼 수 없다. 개발한 화상시스템은 농산물의 질을 평가할 수 있는 외적 정보를 비디오카메라를 통해 제공하고 화면은 전광판보다 저렴한 TV수상기를 사용하므로써 많은 양의 문자 및 숫자정보와 고화질의 영상정보를 제공하면서도 설치비용이 저렴하도록 시스템을 구축한다.

6) 메뉴화면, 경매내용 입력화면, 경매 진행화면, 중매인 관리화면, 출하자 관리화면 등과 같은 경매프로그램을 개발하여 경매진행은 물론 경매결과를 전산처리하기 위한 자료를 제공케 한다.

제 2 장 연구개발 내용 및 범위

제1절 연구개발 범위

연 구 개 발 목 표	연구개발 내용 및 범위
1. 단말기의 설계 및 개발	가. 단말기 key 기능 나. 단말기의 회로구성 다. 단말기의 하드웨어 제어프로그램
2. 단말기-컴퓨터간 인터페이스 개발	가. 인터페이스 회로설계 나. 인터페이스 통신프로그램 다. 컴퓨터의 직렬통신 프로그램
3. TV수상기-컴퓨터간 인터페이스 구성	가. 인터페이스 구성 나. 구동용 프로그램
4. 측정장치-컴퓨터간 인터페이스 구성	가. 인터페이스 구성 나. 인터페이스 제어프로그램 개발 다. 당도,산도,수분,중량값 디스플레이 프로그램 개발
5. 경매프로그램 개발	가. 메뉴화면 나. 경매내용 입력화면 다. 경매진행 화면 라. 중매인관리 화면 마. 출하자관리 화면 바. 화면이용 방법
6. 화면구성	가. 단말기 LCD 화면 구성 나. 컴퓨터 모니터 화면구성 및 TV수상기 화면 구성

제2절 연구개발 내용

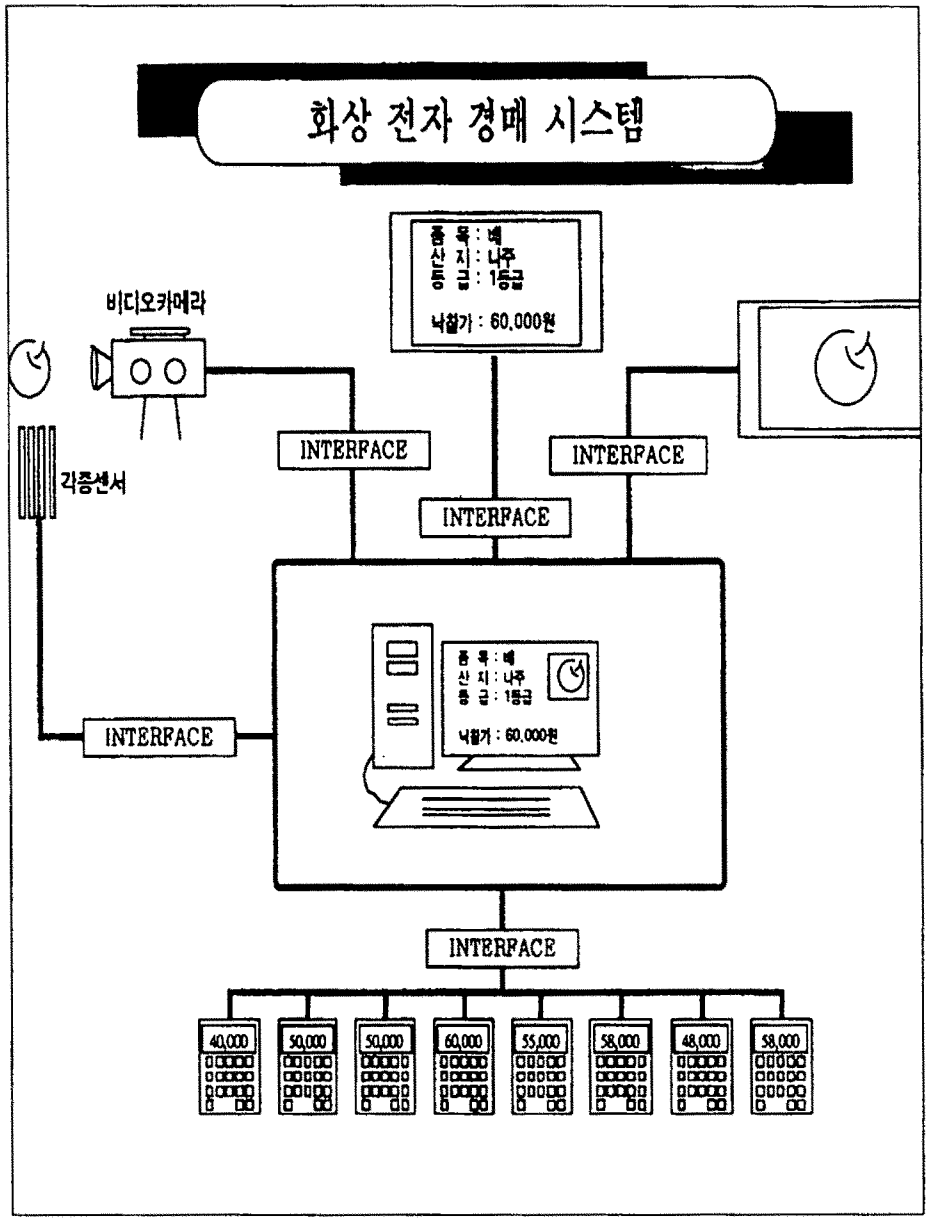


그림 1 화상 전자 경매 시스템의 개요도

본 연구의 전체 시스템은 "화상 시스템이 도입된 전자식 경매 장치"로서 그림 1과 같이 컴퓨터를 중심으로 응찰가를 입력할 수 있는 단말기, 농산물에 대한 내적인 상태(당도, 산도, 수분, 중량)를 측정할 수 있는 각종 센서와 외적인 상태를 보여주기 위한 비디오 카메라, 상술한 내외적 상태와 농산물의 품목, 품종, 산지, 출하자, 수량, 등급, 낙찰·유찰의 유무, 낙찰가, 낙찰자 등과 같은 제반 정보를 나타낼 수 있는 대형 TV수상기, 컴퓨터와 각 장치간의 인터페이스로 구성되어 있다. 이러한 장치들은 경매사의 통제실과 중매인들의 좌석 등과 같은 시설들과 함께 설치되어 전체 시스템을 구성한다.

1. 단말기(keypad)의 설계 및 개발

연구내용은 단말기의 하드웨어와 소프트웨어 부분으로 구분할 수 있으며 단말기 key의 기능, 단말기의 회로구성, 단말기의 하드웨어 제어프로그램 순으로 서술하였다.

가. 단말기 key의 기능

그림 2는 개발한 단말기의 외형과 key의 배치도이며 각 key에 대한 기능은 다음과 같다.

"0~9" Key

중매인의 고유번호와 비밀번호, 응찰가 등과 같은 숫자정보 입력시 사용.

"수정" Key

중매인의 고유번호와 비밀번호, 응찰가에 대한 숫자정보 입력시 중매인의

실수로 인하여 숫자 key를 잘못 사용하였을 경우 "수정" key를 누르면 해당 수치가 취소되고 표시판에 디스플레이된 수치가 삭제된다.

"NEW/END" Key

경매에 참가중인 중매인이 경매를 모두 마쳤을때[END]와 다른 중매인이 경매에 새로 참여할때[NEW] 이미 입력된 고유번호와 비밀번호를 삭제하는 기능을 갖는 key 이다

고유번호와 비밀번호를 삭제하지 않은 상태에서 수치를 입력하면 해당번호의 중매인이 입력한 응찰가로 인식되기 때문에 이러한 오류를 없애기 위함이다. 따라서 같은날 동일 단말기를 다른 중매인이 사용할 경우, 이전에 입력된 번호를 삭제하고 새롭게 시작하기 위해 사용된다.

"보안/해제" Key

응찰가를 입력할때 옆 중매인에게 응찰가의 노출을 방지할[보안] 필요가 있을 경우에 사용한다.

이 key를 한번 누르면 입력한 응찰가는 단말기 내부회로의 메모리에 입력되어 경매는 진행되면서 화면상에만 나타나지 않게되며, 이 key를 한번 더 누르면 응찰가가 화면에 다시 나타나는[해제] 기능을 갖도록 설계한 key 이다.

"상황" Key

경매사에 의해 컴퓨터에 수록된 상품에 대한 정보 즉 품목, 품종, 산지, 출하자, 수량, 등급 등을 중매인이 알고자 할 때 이 key를 누르면 경매 예정 순서 대로 단말기의 LCD 표시판에 나타나게 된다. 중매인은 이 정보에 의해 경매될 품목에 대한 사양을 미리 파악할 수 있으며 구매하고자 하는

품목, 수량, 응찰가를 미리 생각한 후 해당 경매 순서가 되면 경매에 참여할 수 있도록 설계한 key이다.

"000" Key

응찰가를 입력할때 시간단축을 위해 '0' 을 3번 누른 효과를 갖는 단축 key이다. 먼저 누른 숫자에 1000을 곱한 효과를 갖는다.

"0000" Key

"000" Key와 마찬가지로 시간절약을 위한 단축 key이며 '0' 을 4번 누른 효과 즉 먼저 누른 숫자에 10000을 곱한 효과를 갖는다.

" + " Key

단말기는 경매가 진행되는 동안 자기가 입력한 응찰가와 응찰가 중 가장 높은 응찰가가 동시에 단말기의 표시판에 나타나도록 설계되었다. 최고 응찰가를 보고 응찰가를 더 올리고자 할때 " + " Key를 누른 후 숫자 key를 누르면 이전의 응찰가에 가산되는 기능을 갖는다.

"확인" Key

중매인이 고유번호, 비밀번호, 응찰가를 입력한 후 "확인" Key를 누르면 인터페이스 장치를 거쳐 컴퓨터에 정보가 전달된다.

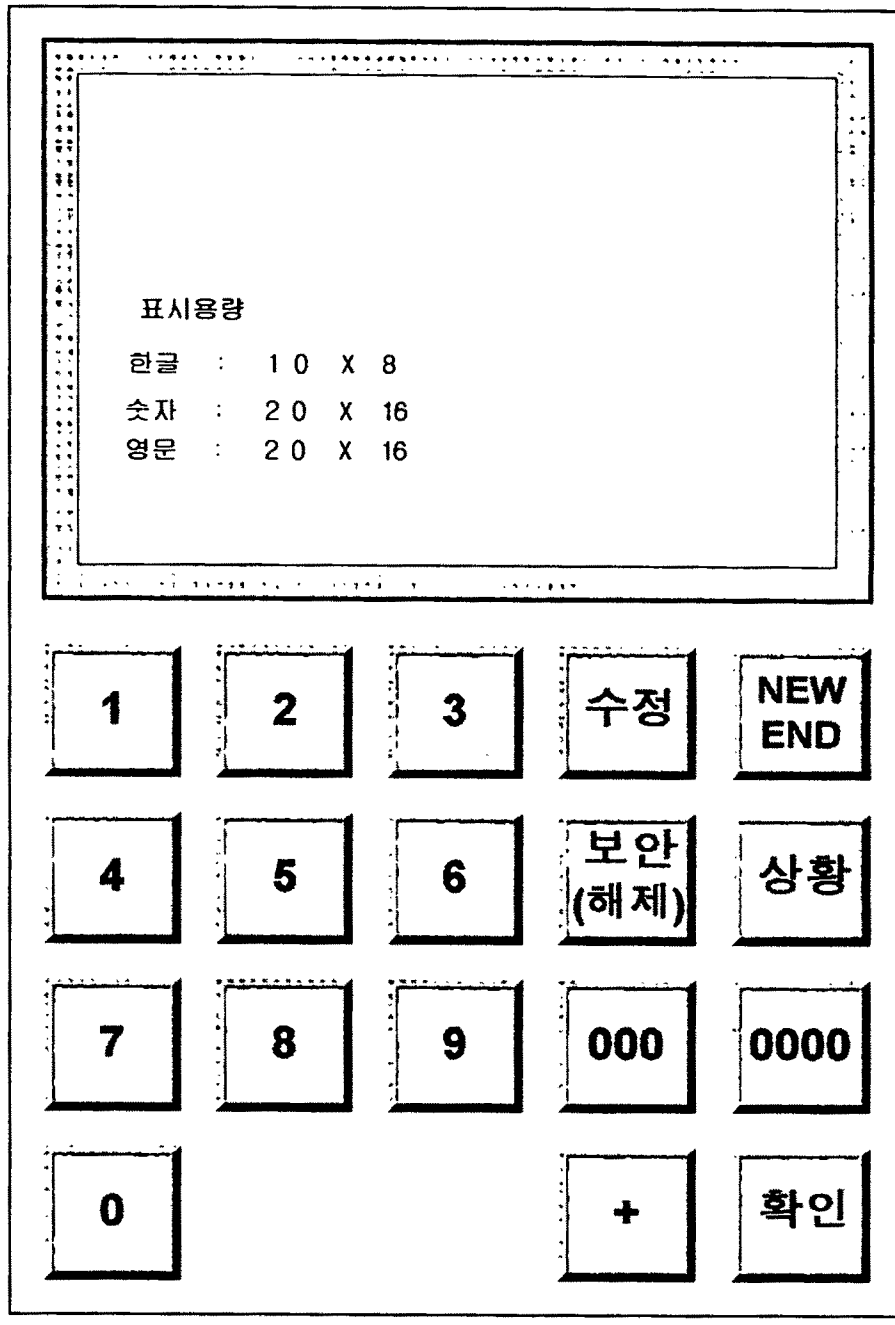


그림 2 Keypad 외형 및 Key 배치도

나. 단말기(keypad)의 회로구성

설계한 단말기 내부회로에 대한 block diagram은 그림 3과 같다. 마이크로 프로세서인 CPU를 중심으로 구성되어 있고 CPU 및 주변 소자들에 대한 역할은 다음과 같다.

▣ CPU

단말기 모듈의 핵심 소자로서 key 입력처리, 메모리와의 데이터처리, 그래픽 LCD제어, 인터페이스와의 통신제어 등을 관장하는 단말기 모듈의 두뇌 역할을 한다.

사용한 CPU는 NEC의 V55PI 이다. 특징으로는 80계열 명령 코드와 호환성을 가진 16bit 마이크로 프로세서이며, 고속 데이터처리, 풍부한 I/O, 우수한 시리얼 인터페이스기능 등 다른 one chip controller에 비해 월등한 기능을 갖고 있다.

▣ MEMORY

ROM (Read Only Memory)

단말기의 구동 프로그램이 기입되며 용량이 64Kbyte인 27C512를 사용하였다. 데이터를 읽을 수만 있고, 데이터를 기입하려면 ROM-writer를 사용해야만 가능하다. 그리고 전원이 끊어진 상태에서도 데이터가 보존된다.

RAM (Random Access Memory)

데이터의 read, write가 가능하고 전원 off시 데이터가 소멸된다. RAM에는 단말기가 동작하는 동안 필요한 변수나 데이터가 저장된다.

용량 128Kbyte의 static RAM인 KM681000을 사용하여 프로그램 개발시 효율을 높게 하였다.

▣ ADDRESS LATCH

CPU인 V55PI의 특성상 필요로 하는 부분으로 V55PI는 어드래스와 데이터 선이 공통 라인으로 되어 있어서 어드래스 신호와 데이터 신호를 분리하여야 한다. 이때 latch와 V55PI에서 제공하는 타이밍 선을 이용하여 어드래스와 데이터를 분리하게 된다. 8bit latch IC인 74HC573을 사용하였다.

▣ ADDRESS DECODER

프로그램 메모리와 데이이터 메모리를 어드래스 신호에 따라 분리할 때 사용된다.

프로그램 메모리인 RAM은 절대번지 00000H~1FFFFH(128KB)에 위치하게되고, 데이터 메모리인 ROM은 절대번지 F0000H~FFFFFH(64KB)에 위치한다, 이때 프로그램 메모리와 데이터 메모리의 선택을 위해 decoder IC를 사용한다.

2→4 디코더 IC인 74HC139를 사용하였다.

▣ KEY INTERFACE

Scan line 4개와 타이밍 신호용 5bit를 V55PI의 포트로 만들어 V55PI에 직결하여 key 신호를 만들며 이 신호는 프로그램에 의하여 기능을 갖게된다.

▣ 통신 INTERFACE

하나의 단말기 모듈은 인터페이스 unit과 통신하여 데이터를 전송하게 된다. 통신은 V55PI의 비동기 직렬 포트 하나를 입력 전용으로, 다른 하나를 출력 전용으로 사용하며 인터페이스 소자로는 RS-485를 사용하였다. RS-485는 공통 라인을 사용하여 30개 까지의 device를 연결할 수 있고 데

이터의 전송속도, 전송거리 등도 RS-232에 비해 성능이 훨씬 우수하다. RS-485의 신호레벨 변환 IC로는 LM75176을 사용하였다.

▣ GRAPHIC LCD

그래픽 LCD는 문자뿐 아니라 그림 정보를 표시할 수 있다는 것이 가장 큰 특징이다. 특히 한글을 표현할 수 있다는 큰 장점이 있다. 그래픽 LCD는 스크린의 구조, 컨트롤러, 백라이트(back light)의 종류에 따라 분류할 수 있으며, 그 종류는 수십 가지에 이른다.

본 연구에서는 160×128픽셀로 구성된 HG16501을 사용하였다. 이 제품은 컨트롤러가 내장된 모듈 형태로서 마이크로 프로세서를 이용해서 제어할 수 있다. 크기는 129×102×11.5mm 이고, 백라이트는 노란색이며, EL 방식이다.

▣ DC-DC CONVERTER

그래픽 LCD 구동 전압을 만들기 위해 AD949 DC-DC converter를 사용하였다.

STN 방식의 LCD에서는 로직용 전원 5[V] 이외에, LCD 자체를 구동하기 위한 음(-)전압 -15[V]도 요구된다. 음 전압을 만드는 방법은 여러가지가 있으나 모듈로된 절연형 DC-DC 변환기를 이용하였다. 5[V] 입력에 출력은 15[V]이나 접지가 분리되어 있으므로 출력단을 반대로 접속하면 -15[V]를 얻을 수 있다. 또 명암을 조정해야 하므로 가변저항으로 출력전압을 조절할 수 있도록 하였다.

▣ INVERTER

그래픽 LCD의 back light 구동용으로 그래픽 LCD 전용 인버터를 사용하

였다.

언급한 백라이트용 전압은 인버터를 이용해서 만든다. 이 인버터는 직류 5[V]를 교류 110[V]로 변환시키는 소자이며 코일과 몇 개의 부품으로 구성된 모듈 형태이다.

▣ POWER

외부에서 12[V] 전압을 입력 받아 마이크로 프로세서 구동전압인 5[V]를 얻기위해 정전압 IC인 LM7805를 사용한다, 단말기 모듈의 총소비 전류는 450㎃ 정도이다.

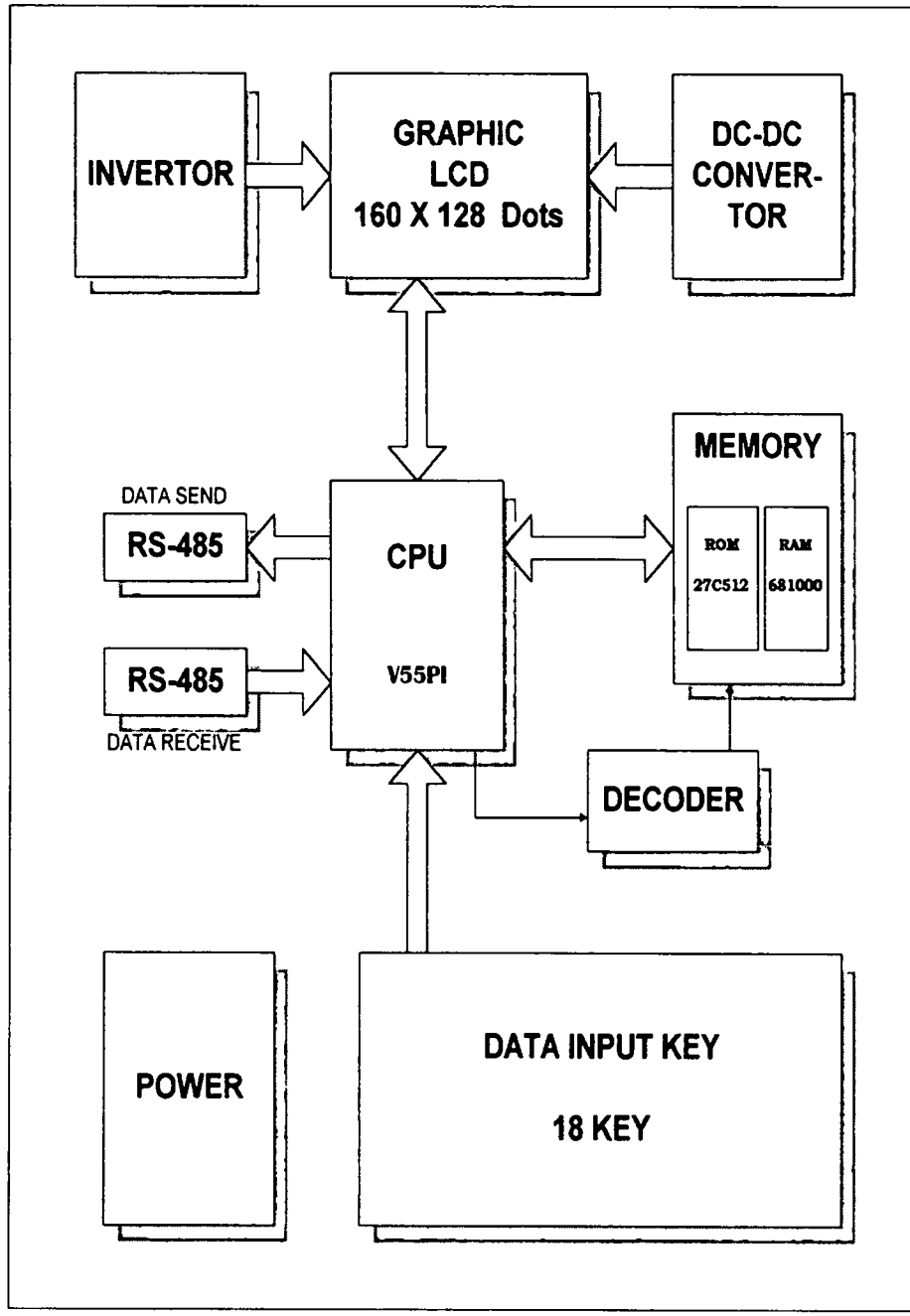


그림 3 Keypad block diagram

다. 단말기의 하드웨어 제어 프로그램

단말기 하드웨어 제어 프로그램은 시스템초기화 및 key 입력처리 프로그램, 그래픽 LCD 제어 및 한글출력 프로그램, key 기능 및 화면구성 프로그램, 인터페이스와의 통신 프로그램 등으로 구분할 수 있다.

1) 시스템초기화 및 key 입력처리 프로그램

시스템 초기화 프로그램은 단말기 시스템이 경매를 수행할 수 있도록 하드웨어를 초기화한다.

전원이 인가되면 가장 먼저 CPU인 V55PI를 초기화하게 되며 초기화 내용은 프로세스 레지스터, 메모리 리프레시 레지스터, 인터럽트 벡터테이블 레지스터, 인터럽트 초기화 레지스터, wait제어 레지스터, 포트제어 레지스터, 시리얼제어 레지스터, 카운터제어 레지스터 등이다.

Key 입력처리 프로그램은 단말기의 key 중 몇 번째 출력과 입력 포트에 연결되어 있는 key가 눌러졌는가를 판단하는 부분이다. 단말기의 key부분 하드웨어는 key 입력을 위해 전용 IC를 사용하지 않고 key를 mtrix 형태로 배선하여 V55PI 포트에 연결한 다음 몇 번째 출력과 입력인가를 검사하는 scan 방식의 프로그램을 사용하여 key입력을 처리한다. 여기서는 어떤 key가 눌러졌는지를 판단할 뿐 key에 기능을 부여하지는 않는다.

2) 그래픽 LCD 제어 및 한글출력 프로그램.

그래픽 LCD에서 한글구현 원리와 출력방법 그리고 그래픽 구현방법 및 그래픽 함수에 대하여 다음과 같이 구현하였다.

▣ 한글의 구현.

완성형은 음절 단위의 독립적인 코드를 할당하여 한글을 표현한 것이며,

조합형은 한글 제자원리와 동일하게 음소별로 초성(19), 중성(21), 종성(27)의 67개 코드로 할당하는 효율이 좋은 방법이다. 본 연구에서는 조합형 한글을 사용하였다.

▣ 한글 출력.

한글은 2바이트(16비트)로 한 글자를 표현한다. 16비트 중 1비트는 한글/영문 판단을 하며, 나머지 15비트를 초성/중성/종성으로 나누어 5비트씩 할당한다. 이렇게 3등분한 코드로 해당 문자를 찾아 겹치기하여 글자를 구현한다. 즉 임시 메모리에 초성, 중성, 종성을 OR 연산후 화면에 뿌리는 방법이다. 초성의 경우 8벌중 어떤 것을 선택할 것인가 하는 알고리즘이 필요하나, 본 연구에서는 간단히 코드 참조표를 이용한다.

▣ 그래픽의 구현

그래픽 LCD는 비트 맵이 아닌 바이트 맵을 가지고 있다. 다시 말하면 본 연구에 사용하는 LCD는 160×128 픽셀인데, 그 중 X=100, Y=20인 그래픽 좌표계에 점을 찍는 경우를 생각해 보면, 우선 Y축은 20번째로 맞추면 된다. 그리고 X축에 대해서는 우선 바이트 단위로 바꾸기 위해 100을 8로 나누면 몫이 12, 나머지는 4가 된다. 따라서 그래픽 메모리번지 20×12의 4번째 bit를 조작하면 된다.

이때 이 값을 바로 출력하면 바이트 셀에서 기존의 값이 지워져 버린다. 이러한 문제를 해결하기 위해서는 비디오 메모리가 필요하며 기존의 값을 읽어 OR연산 후 출력하면 해결할 수 있다. 또 LCD에 장착된 메모리를 일일이 읽고 연산하고 다시 기록하는 식의 프로그램은 효율이 너무 낮아 사용할 수 없었고, 구성한 단말기 프로그램에 화면 크기에 해당하는 4096바이트의 버퍼를 설정하여 사용하였다.

▣ 그밖의 그래픽 함수

선, 상자, 원에 대한 알고리즘은 비교적 간단히 구현되었다.

□ 사용 함수 목록

void gplot(unsigned char gx, unsigned char gy,unsigned char _color);

// 원하는 좌표상에 한점을 찍는다. color=0이면 지우고, 1이면 그린다.

void gotoxy(char x, char y);

// 그래픽 좌표를 설정한다.

void engput(char b1);

// 영문 1글자를 출력한다,

void lcd_initial(void);

// LCD를 그래픽 모드로 설정하고, 컨트롤러의 레지스터를 초기화한다

void lcdputch(int b1);

// 1바이트의 패턴을 출력한다.

void L_outstr(unsigned char *msg);

// 문자열을 출력한다.

void L_printf(char *form, ...);

// C언어의 printf 함수와 같은 역할을 한다.(한영 겸용)

void clrlcd(void)l

// 화면을 지운다.

void line (int sx, int sy, int ex, int ey, unsigned char _color);

// 선을 그린다.

void box(int x1, int y1, int x2, int y2, unsigned char color);

// 상자를 그린다.

3) Key 기능 및 화면구성 프로그램

단말기에 배치된 18개 key의 기능은 소프트웨어로 부여하였으며, key의 사용과 화면구성에 대한 단말기 프로그램의 동작 알고리즘에 대한 흐름도(flow chart)는 그림 4에 나타내었다.

단말기에 전원이 인가되면 가장 먼저 시스템 초기화 프로그램이 구동하게 된다. 초기화는 CPU, 그래픽 LCD, 통신 포트의 초기화 등이다. 기본적인 초기화가 끝나면 LCD 화면에 간단한 로고화면이 나타난다. 그리고 내부적으로는 인터페이스 유닛과 통신하면서 하드웨어 ID를 할당받게 된다. 하드웨어 ID 할당이 끝나면 '중매인 고유번호를 입력하십시오' 라는 메시지를 출력하고 중매인 고유번호를 입력받는다. 입력 받은 중매인 고유번호는 인터페이스 유닛을 통하여 컴퓨터에 전송되고 등록된 중매인의 고유번호인가를 판단한다. 이때 등록된 고유번호이면 등록자 이름과 비밀번호를 함께 인터페이스 유닛을 통하여 단말기로 전송하고 다음 순서로 진행되며 등록된 고유번호가 아니면 '재 입력하십시오' 라는 메시지를 출력하고 중매인 고유번호 입력부분으로 가서 재 입력 받게된다. 고유번호가 유효할때 중매인 이름과 비밀번호를 같이 송출하는 이유는 단말기 모듈과 컴퓨터가 통신하는 횟수를 줄이고 간단한 확인처리(비밀번호 확인)는 단말기 모듈 내에서 하기 위함이다.

고유번호 확인이 끝나면 비밀번호를 확인하게 된다. 화면에 '비밀번호를 입력하시오'라는 메시지를 출력하고 4자리 숫자로 구성된 비밀번호를 입력받게 된다. 비밀번호를 입력하면 확인과정을 거쳐 맞으면 다음 단계로 넘어가고 틀리면 '재 입력하십시오' 라는 메시지를 출력하고 다시 비밀번호를 입력받게 된다. 여기서 알 수 있는바와 같이 경매에 참가하려는 중매인은 사전에 등록하고 자기의 고유번호와 비밀번호를 가지고 있어야 경매에 참가할 수 있다.

고유번호와 비밀번호의 확인이 끝나면, 단말기 모듈은 컴퓨터와 통신하여 경매할 품목과 품목에 대한 사양을 LCD화면에 표시하고 '원하는 경매를 선택하십시오' 라는 메시지를 출력한다. 중매인의 상장번호 선택이 끝나고 해당 품목의 경매순서가 되면 응찰가를 입력하여 경매가 이루어진다. 중매인은 원하는 응찰가를 직접 숫자 key를 사용하여 입력할 수 있고 "+" Key나 "000", "0000" Key를 조합하여 응찰가를 입력할 수 있다. 중매인이 원하는 응찰가를 입력하면 유효한 응찰가(전에 입력한 응찰가 보다 값이 낮으면 의미가 없으므로 무시)인가를 판단하여 LCD화면에 표시하고 동시에 컴퓨터에 전송한다. 이때 최고 응찰가와 자신의 응찰가가 LCD 화면에 나타난다. 한편, 경매중 다른 중매인에게 자기 응찰가가 노출되는 것을 막고자 할 때, "보안" Key를 사용하여 응찰가 부분이 LCD화면에서 지워지게 할 수 도있고 "해제" Key를 사용하여 다시 표시할 수 도있다.

경매를 포기하거나 끝냈을 경우 "NEW/END" Key로, 다른 경매에 참가하려면 "상황" Key로 빠져나갈 수 있다. "NEW/END" Key는 경매상태에서 완전히 빠져나가 초기화면으로 가며, "상황" Key는 경매될 품목에 대한 상황을 표시해 주는 상황화면으로 이동하게 된다.

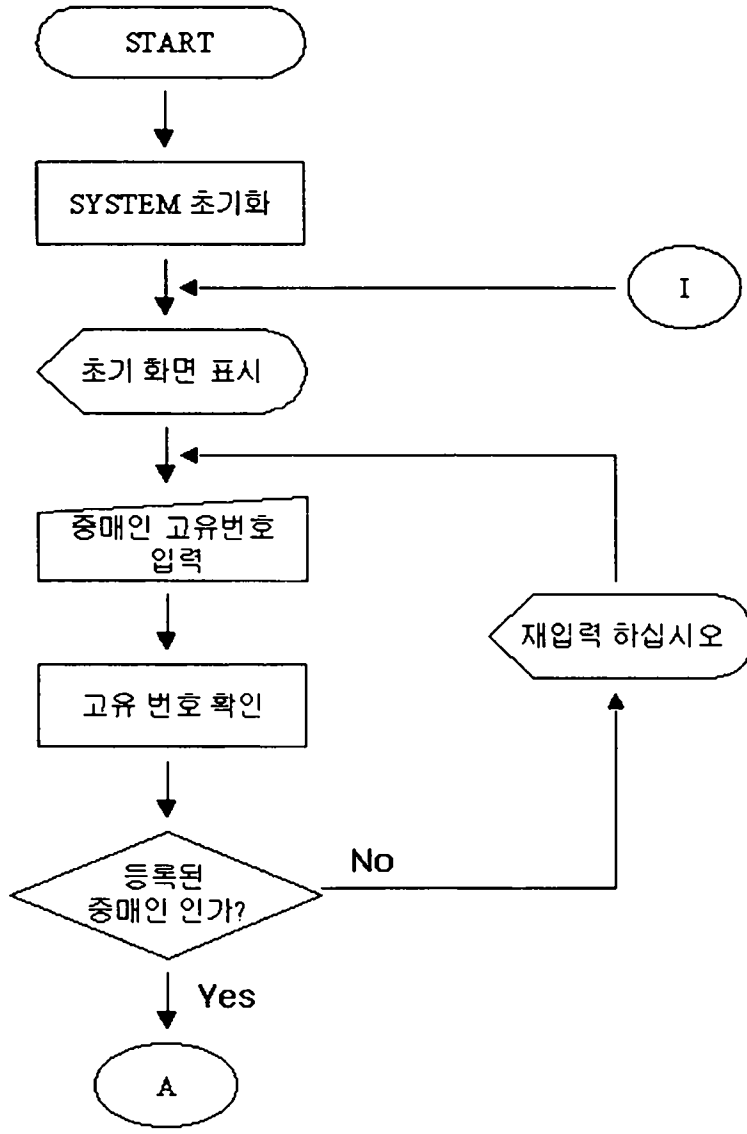
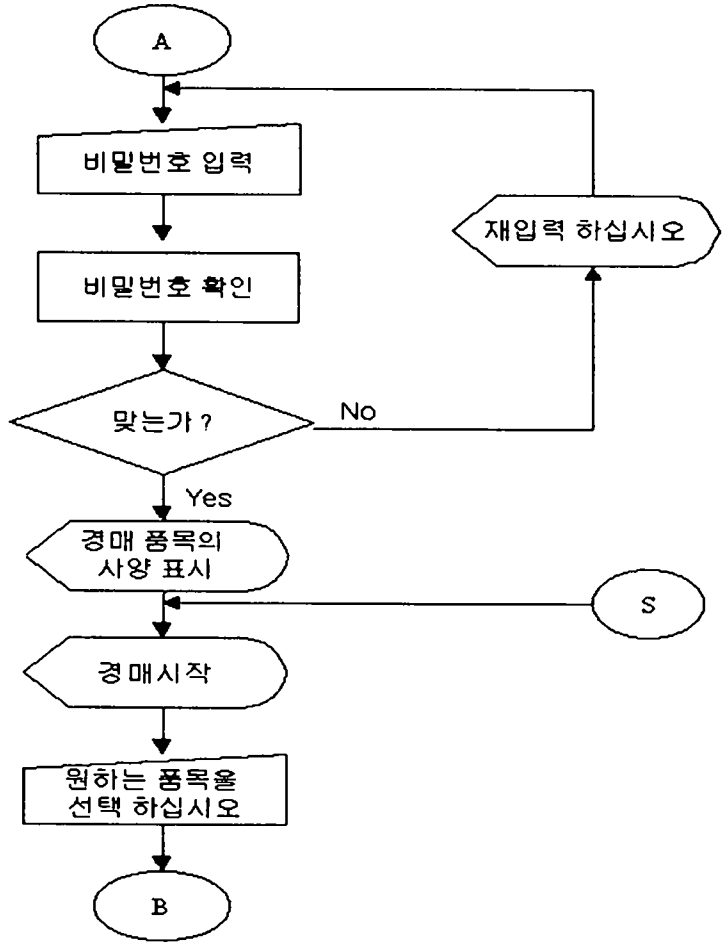
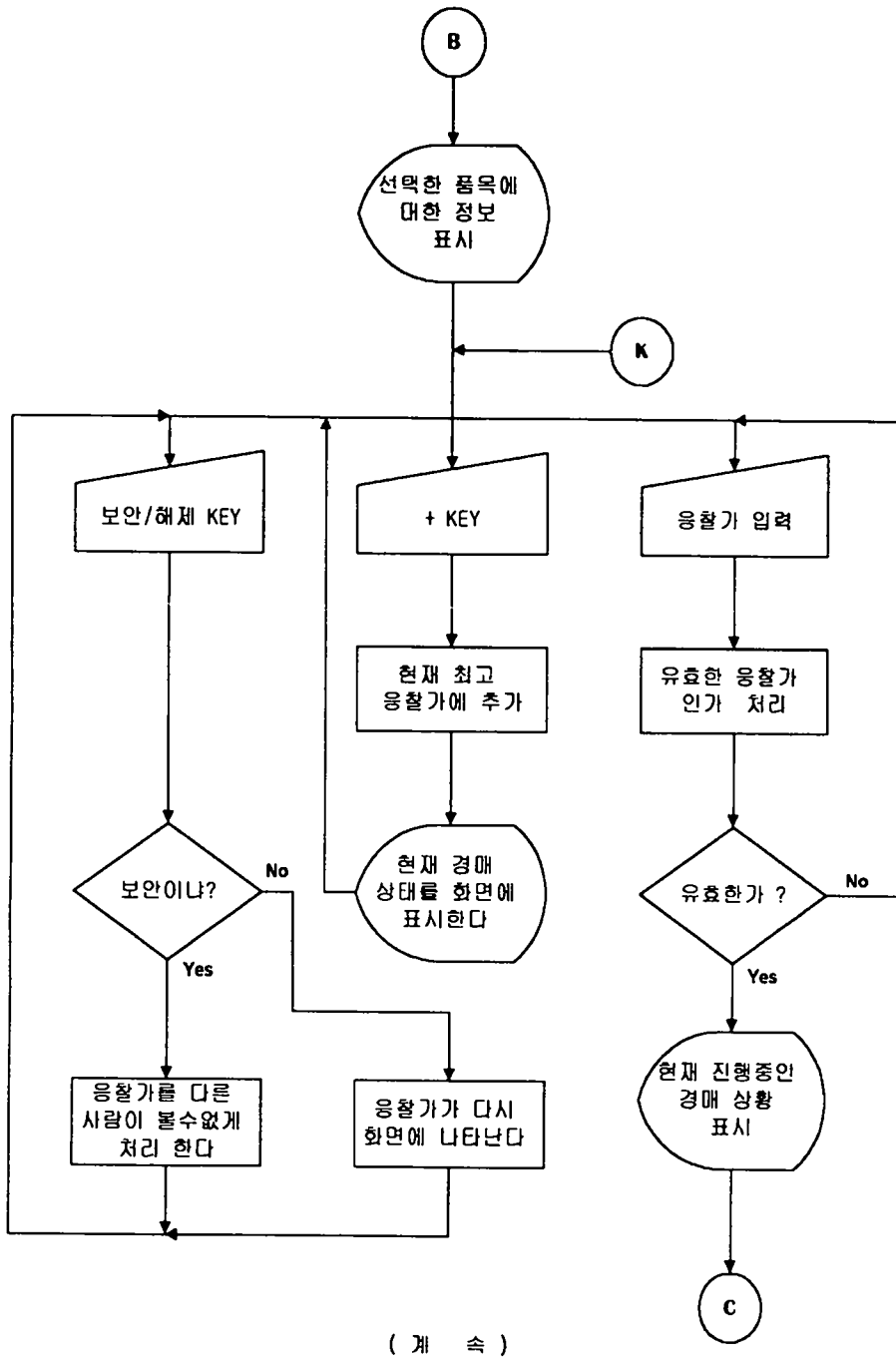


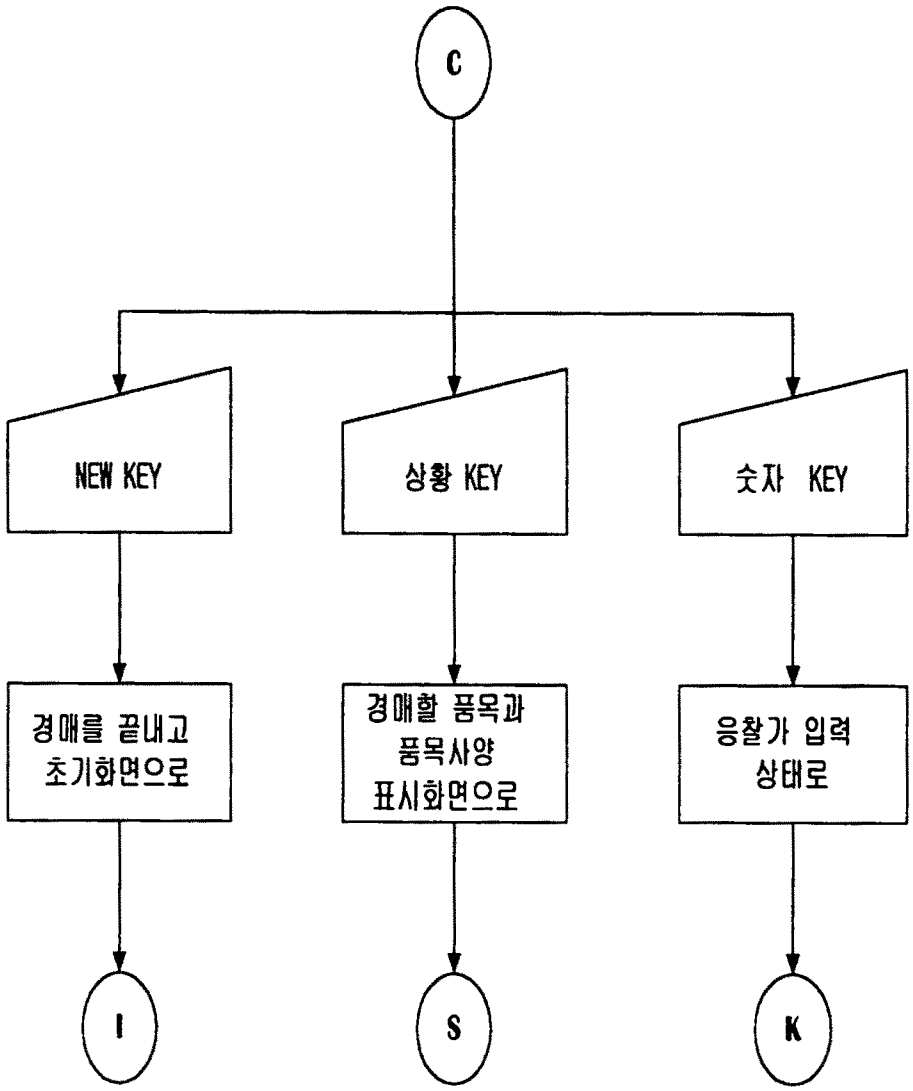
그림 4 단말기의 화면구성 및 동작알고리즘 흐름도


A
비밀번호 입력
비밀번호 확인
맞는가 ?
No
재입력 하십시오
Yes
경매 품목의
사양 표시
S
경매시작
원하는 품목을
선택 하십시오
B

[계 속]


B
선택한 품목에
대한 정보
표시
K
보안/해제 KEY
+ KEY
응찰가 입력
현재 최고
응찰가에 추가
유효한 응찰가
인가 처리
보안이나?
No
현재 경매
상태를 화면에
표시한다
유효한가 ?
No
Yes
Yes
응찰가를 다른
사람이 볼수없게
처리 한다
응찰가가 다시
화면에 나타난다
현재 진행중인
경매 상황
표시
C

(계 속)


C
NEW KEY
상황 KEY
숫자 KEY
경매를 끝내고
초기화면으로
경매할 품목과
품목사양
표시화면으로
응찰가 입력
상태로
I
S
K

(끝)

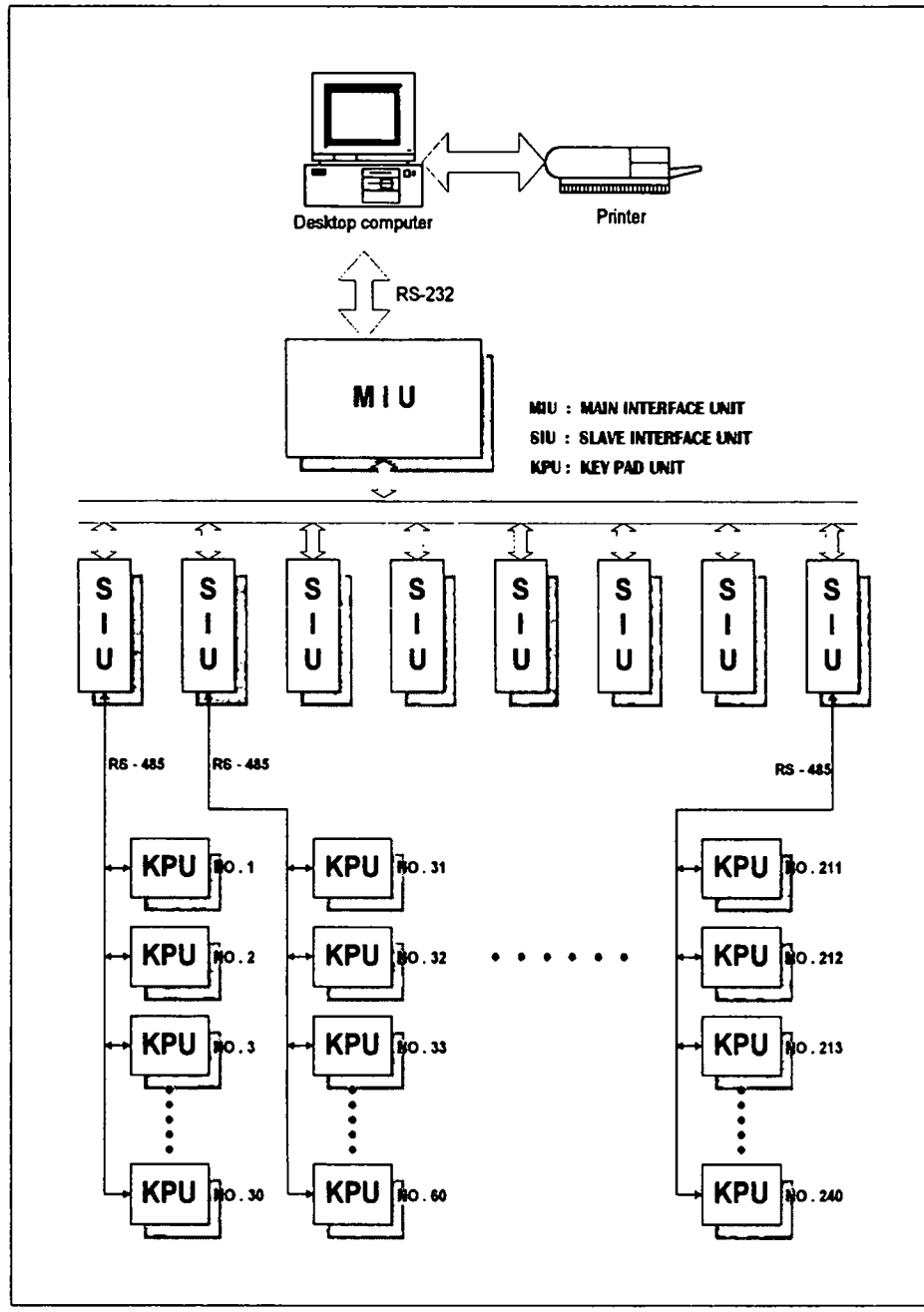


그림 5 Keypad-컴퓨터간 인터페이스 전체구성도

2. 단말기-컴퓨터간 인터페이스 개발

가. 인터페이스 회로설계

단말기-컴퓨터간 인터페이스는 SIU(slave interface unit)와 MIU(main interface unit) 두 개의 card를 개발하였으며 keypad-컴퓨터간 인터페이스의 전체 구성도는 그림 5와 같다.

1) SIU 회로구성

설계한 SIU 회로에 대한 block diagram은 그림 6과 같고, 마이크로 프로세서인 CPU를 중심으로 구성되어 있으며 CPU 및 주변 소자들에 대한 역할은 다음과 같다.

▣ CPU, MEMORY, ADDRESS LATCH, ADDRESS DECODER ·

Keypad의 경우와 동일함.

▣ MIU INTERFACE

MIU와의 인터페이스는 8bit의 data line과 3개의 신호선으로 구성되어 있다.

8bit의 data line은 양방향 전송이 가능하여야 하기 때문에 V55의 입출력 포트인 P4를 사용하여 입출력한다. 그리고 핸드쉐이크를 위한 3개의 제어신호는 SIU가 MIU에 데이터를 전송하거나 MIU로부터 데이터를 전송 받을 때 사용된다.

제어신호선의 구성은 MIU에 데이터 요구를 위한 data request 신호와 데이터가 버퍼에 준비를 알리는 ready 신호, 그리고 데이터의 방향을 나타내는 data direction 신호로 구성하였다.

▣ 통신 INTERFACE

SIU 모듈은 단말기 unit과 통신하여 데이터를 전송하게 된다. 통신은 V55PI의 비동기 직렬포트 하나를 입력 전용으로, 다른 하나를 출력 전용으로 사용하며 인터페이스 소자로는 RS-485를 사용하였다, RS-485는 공통라인을 사용하여 32개까지의 device를 연결할 수 있고 데이터의 전송속도, 전송거리 등도 RS-232에 비해 성능이 훨씬 우수하다. RS-485의 신호레벨 변환 IC로는 LM75176을 사용하였다.

▣ 전류 -> 전압 변환회로

전류 체크는 단말기 전원 line에 기준저항을 직렬로 연결하여 기준저항에 나타나는 전압강하분을 instrument amp를 사용하여 전압으로 변환하여 준다.

기준저항은 0.2Ω 5W를 사용하였고 instrument amp는 TL082를 사용하여 제작하였다.

▣ POWER

외부에서 12V 전압을 입력받아 마이크로 프로세서 5[V]의 구동전압을 얻기 위해 정전압 IC인 LM7805를 사용한다, SIU 모듈의 총소비 전류는 150 [mA] 정도이다.

▣ SWITCHING POWER

단말기에 전원을 공급하기 위하여 12[V], 17[A]를 사용하였다.

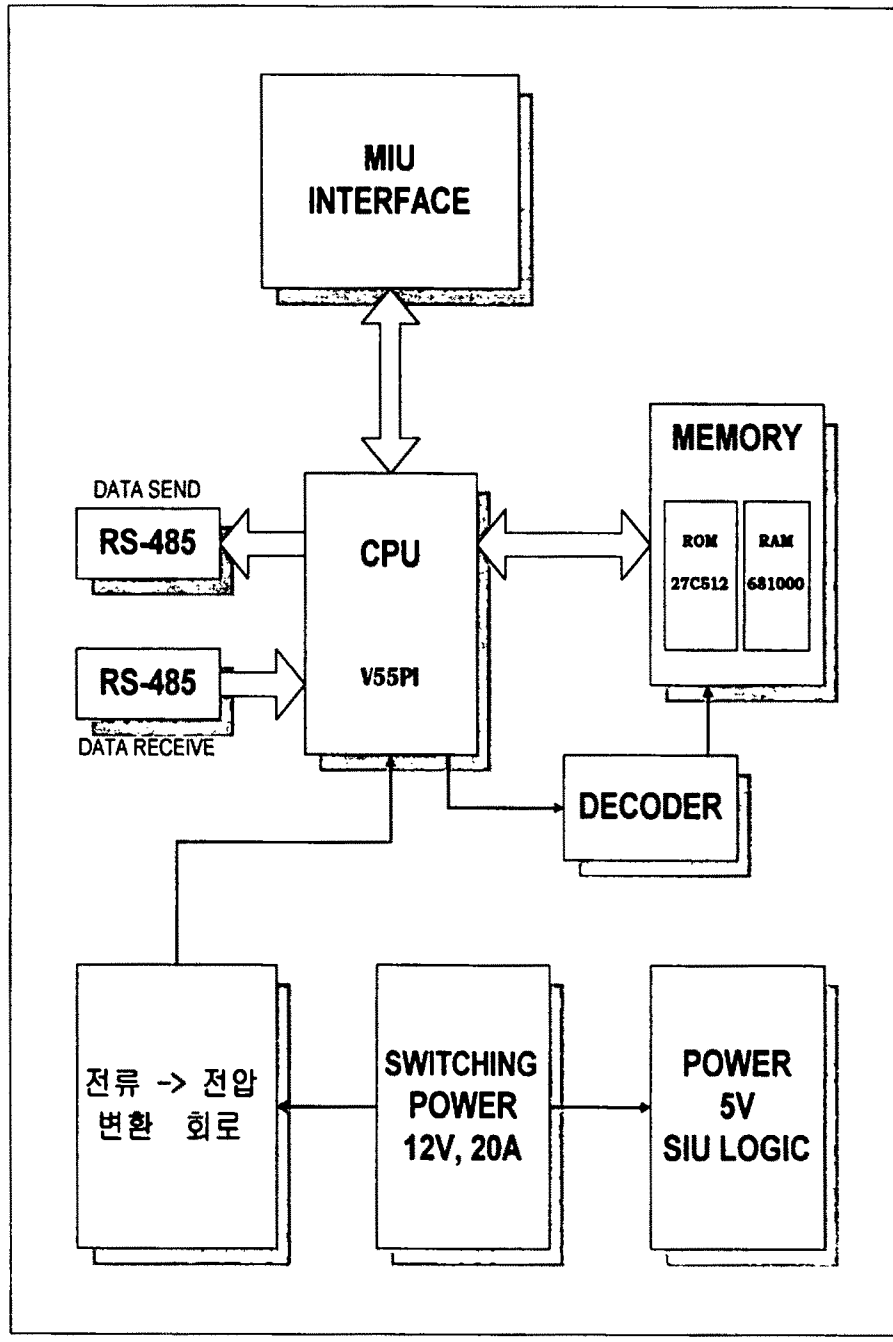


그림 6 SIU의 block diagram

2) MIU 회로구성

그림 7은 설계한 MIU 회로에 대한 block diagram이며, CPU 및 주변 소자들에 대한 역할은 다음과 같다.

▣ CPU, MEMORY, ADDRESS LATCH, ADDRESS DECODER

Keypad의 경우와 동일함.

▣ SIU INTERFACE

MIU에는 8개의 SIU 스롯을 가지고 있으며 내부 전용버스를 통하여 SIU와 통신을 한다.

SIU와의 인터페이스는 8bit의 data line과 3개의 신호선으로 구성되어 있다. 8bit의 data line은 양방향 전송이 가능하여야 하기 때문에 V55의 입출력 포트인 P4를 사용하여 입출력한다. 그리고 핸드쉐이크를 위한 3개의 제어 신호는 MIU가 SIU에 데이터를 전송하거나 SIU로부터 데이터를 전송받을 때 사용된다.

제어신호선의 구성은 SIU가 MIU에 데이터 요구를 위한 data request 신호와 데이터가 버퍼에 준비를 알리는 ready 신호 그리고 데이터의 방향을 나타내는 data direction 신호로 구성하였다.

▣ 통신 INTERFACE

MIU와 PC간은 RS-232 통신규격으로 통신한다. RS-232 통신규격을 사용한 이유는 PC의 직렬포트가 RS-232를 사용하기 때문이다.

통신은 V55PI의 비동기 직렬포트 하나를 사용하여 입·출력을 행하며 RS-232 신호레벨 변환 IC로는 MAX232C를 사용하였다.

▣ BUFFER

전류 드라이버 능력을 크게 하여 전송신호가 노이즈를 타는 것을 방지하고 tri-state 버퍼를 사용하므로써 MIU와 SIU가 동시에 데이터전송이 있을 때 전기적인 충돌을 방지한다.

버퍼 IC로는 8bit tri-state 버퍼인 74HC245를 SIU 버퍼로 사용하였다.

▣ SIU CHECK

SIU 시스템을 효율적으로 관리하고 통신하기 위하여 SIU가 MIU에 몇 개가 접속되어 있는지 알아야 한다. 그래서 SIU 가 MIU 기판상에 접속될 때 SIU 패턴상에 5V를 연결하여 두고 MIU는 SIU 체크시 이 레벨을 감지하므로써 SIU의 연결을 알 수 있다.

Level check IC로는 8bit demultplexer인 74HC151을 사용하였다.

▣ POWER

외부에서 12V 전압을 입력받아 마이크로 프로세서 구동전압인 5[V]를 얻기 위해 정전압 IC인 LM7805를 사용한다, MIU 모듈의 총소비 전류는 250 ㎃ 정도이다.

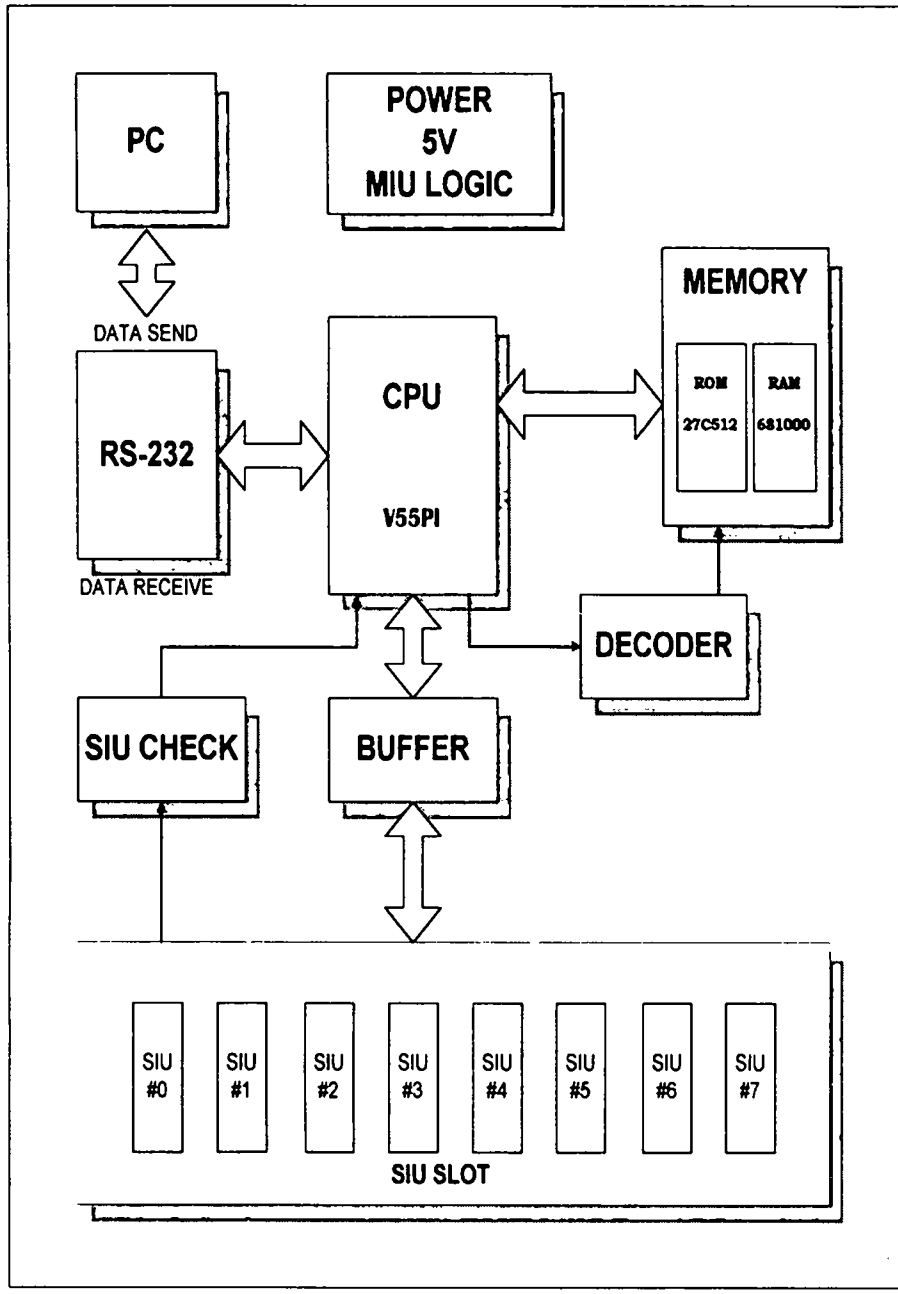


그림 7 MIU의 block diagram

나. 인터페이스 통신 프로그램

단말기와 컴퓨터간 데이터의 전송을 위한 각 단계별 즉 SIU-단말기간, SIU-MIU간 , MIU-PC간의 통신방법은 다음과 같다.

1) SIU - 단말기간 통신 방법

SIU와 단말기 통신은 기본적으로 message polling 방식으로 행하여 진다. 단말기 모듈에 전원이 인가되면 power on reset이 되면서 단말기 모듈을 초기화하고 통신 패킷에 의해 통신이 이루어진다.

가) 단말기 모듈 초기화

단말기에 전원이 인가되면 먼저 단말기 모듈내의 CPU, RAM, 그래픽 LCD 등의 초기화가 끝나고 SIU로 부터 단말기의 고유 ID를 할당받는다. 단말기 모듈의 초기화와 ID 할당에 관한 순서도는 그림 8과 같고 ID의 할당은 다음과 같이 이루어진다.

SIU의 RS-485 케이블에 새로운 단말기가 연결되면 SIU전류 감지기가 전류를 감지함으로써 새로운 단말기가 접속되었다는 것을 알 수 있다, 이때 SIU는 새로운 단말기를 찾아내기 위하여 새로 접속된 단말기로 부터 초기화 완료 코드를 수신할 때까지 대기한다. 이때 일정 시간 경과 후까지 초기화 완료 메시지가 수신되지 않으면 지금까지 설정되어 있는 ID를 제외한 모든 ID에 에러 메시지를 전송한다. 이때 단말기는 수신 버퍼 인터럽트를 enable하고 수신 버퍼를 감시하면서 에러 메시지가 들어오면 LCD 화면에 "접속 에러입니다. 직원에게 문의하십시오." 라는 메시지를 띄운다. 일정시간 내에 SIU에 단말기로 부터 초기화 완료 메시지가 수신되면 SIU는 여유 ID인 '31'을 데이터 패킷에 맞추어 전송하고 새로 접속된 단말기 모듈이 '31'을 수신한 다음 단말기 송신 패킷에 맞추어 ROM에 기록되어 있는 고유

넘버를 전송하게 된다. SIU는 이 넘버를 받아서 단말기 모듈에 새로운 ID를 송출하게 된다. 이렇게 하여 ID 할당이 끝나면 이후 부터 이 단말기 모듈과의 통신은 이 ID를 이용하게 된다.

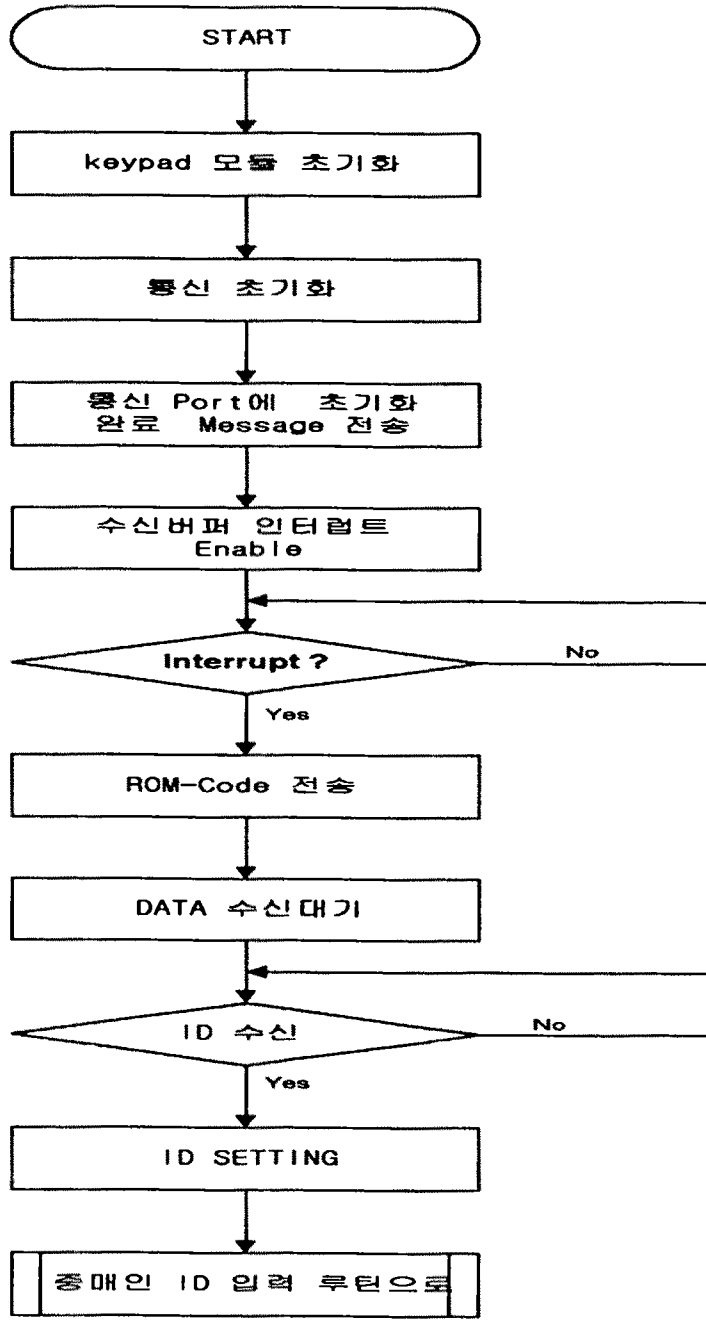


그림 8 Keypad모듈의 초기화와 ID 할당 순서도

나) Data packet

▣ 단말기 => SIU

· 초기화 완료

STA	INITEND	END	CRC

· ROM 고유 넘버 전송

STA	INIT-ID	END	CRC

· CRC ERROR

STA	CRC-ERROR	END	CRC

▣ SIU => KEYPAD

· Keypad ID 할당 1

STA	ID 할당	31	END	CRC

· Keypad ID 할당 2

STA	ID 할당	ROM-NO	END	CRC

· Error message

STA	ERROR	ID	END	CRC

· CRC error

STA	CRC-ERROR	END	CRC

다) 단말기 -> SIU data 전송

단말기 모듈에서 SIU로 데이터전송은 SIU에서 hard ID를 순차적으로 보내고 그 ID에 해당하는 단말기가 ID를 polling한 다음 ID를 전송하게 된다.

단말기에서 SIU에 보내야 할 경매 데이터는 중매인 ID, 비밀번호, 경매

가, 경매정보 요구 등이 있다.

□ 통신 Packet

· 중매인 ID

STA	BROKER-ID	4Byte DATA	END	CRC

· 비밀번호

STA	S-CODE	4Byte DATA	END	CRC

· 경매가

STA	PRICE	8Byte DATA	END	CRC

· 수량

STA	QUANTITY	6Byte DATA	END	CRC

· 경매정보

STA	INFO	END	CRC

· 정보없음

STA	EMPET	END	CRC

라) SIU -> 단말기 데이터 전송

SIU에서 단말기로의 데이터 전송은 다음의 두 가지 모드가 있다.

하나는 단말기 모듈이 hard ID를 지정하고 SIU에 데이터 전송이 있는지 검사하기 위한 polling 모드이고, 다른 하나는 SIU에 특정 단말기 또는 SIU에 연결된 모든 단말기 모듈이 데이터를 전송할 때 사용하는 data write 모드이다.

▣ Data polling 모드

순차적으로 hard ID를 증가하면서 단말기로 부터 데이터를 받는다.

▣ Data write 모드

□ 통신 packet

· 중매인 확인됨

STA	PASS	Hard-ID	중매인 이름	END	CRC

· 중매인 확인되지 않음

STA	NOPASS	Hard ID	END	CRC

· 최고 경매가

STA	PRICE	8Byte DATA	END	CRC

· 수량

STA	QUANTITY	6Byte DATA	END	CRC

· 경매정보

STA	INFO	경매정보	END	CRC

2) SIU - MIU간 통신방법

SIU와 MIU의 데이터 전송은 내부 버스를 이용한 인터럽트 핸드쉐이크 방식의 병렬통신에 의하여 이루어진다.

▣ SIU 초기화

SIU에 전원이 인가되면 먼저 SIU내의 CPU, RAM 등의 초기화가 끝나고 SIU는 단말기에 고유 ID를 할당한다. ID할당은 다음과 같이 이루어진다.

SIU에서 접속된 단말기의 갯수의 확인을 위해 단말기 모듈에 유입되는 전류를 감지하여 단말기가 SIU에 몇 개 연결되어 있는지를 판단하는 방법으로 첨가된 단말기에 hard ID를 할당한다.

전류 체크는 단말기 하나가 동작하기 위해서는 500[㎃]정도 필요하므로

현재 전류보다 전원 라인에 몇 ㎃ 더 추가 유입되는가를 체크하면 단말기 몇 개가 새로 추가되어 있는지 알 수 있다. 전류 체크는 먼저 전류를 전압으로 변환한 다음 V55 CPU내부에 있는 타이머에 일정 시간을 설정한 후 이 기간이 경과할 때마다 인터럽트 루틴에서 AD변환기를 사용하여 전압을 측정하며 이것을 프로그램 내에서 이전 전류와 비교하여 500[mA]이상 변화가 있을 때 단말기모듈의 접속이나 해제를 판단하고 단말기에 새로운 hard ID를 부여하거나 접속이 끊어진 단말기의 hard ID를 삭제한다.

▣ SIU -> MIU 데이터 전송

SIU가 MIU에 데이터를 전송하려면 먼저 MIU로부터 data bus 사용권을 얻어야 한다. Bus 사용권을 얻은 후 SIU는 MIU에 1byte의 데이터를 전송한다.

데이터 전송권은 다음과 같은 절차에 의해 얻는다.

SIU의 data request 신호선의 레벨을 low에서 high로 만들면 MIU는 data request 신호선을 감시하고 있다가 high 레벨이 되면 현재 처리하고 있는 일을 모두 완료한 후 data direction 신호선을 SIU에서 MIU로 설정한다. 그러면 SIU는 data direction 신호선이 SIU에서 MIU로 전환된 것을 감지하여 data port를 출력으로 설정한 후 1byte의 데이터를 data port에 전송한다 data request 신호선의 레벨을 low 레벨로 만든다. 이때 MIU는 data request 신호선이 low 레벨이 되면 data bus로 부터 1byte의 데이터를 읽어가며 읽어간 데이터가 END code가 아니면 END code를 수신할 때까지 MIU가 SIU에 데이터를 요구하게 된다. 이때 다른 SIU에서 MIU에 데이터 전송은 보류되며 SIU번호가 메모리에 저장된다. 데이터 요구는 SIU에 인터럽트를 발생시켜 SIU 데이터 포트에 데이터가 준비되면 데이터를 읽어오는 방법으로 END 코드가 수신될 때까지 반복한다.

□ 통신 packet

SIU에서 MIU에 보내야 할 경매 데이터는 중매인 ID, 비밀번호, 경매가, 수량이다.

· 중매인 ID

BROKER-ID	Hard-ID	4Byte DATA	END

· 비밀번호

S-CODE	Hard-ID	4Byte DATA	END

· 경매가

PRICE	Hard-ID	8Byte DATA	END

· 수량

QUANTITY	Hard-ID	8Byte DATA	END

▣ MIU -> SIU 데이터 전송

Data direction을 MIU에서 SIU로 설정한 다음 MIU 데이터 버스에 데이터를 셋팅 하고 SIU에 인터럽트를 발생시킨다. 이때 SIU인터럽트 루틴에서 data direction 신호선을 체크하여 신호방향이 MIU에서 SIU이면 data port를 입력으로 바꾸고 데이터를 1byte 읽는다. 그후 MIU에 데이터를 읽었다는 신호를 전송한다.

SIU의 통신 패킷 구성시 Hard-ID는 PC에서 넘어온 Hard-ID ÷ 30 한 결과 몫이 SIU slot 번호이고 나머지가 Hard-ID 이다.

□ 통신 packet

MIU에서 SIU에 보내야 할 경매 데이터는 중매인 이름, 비밀번호 통과여부, 최고 경매가, 수량, 경매정보이다.

· 중매인 확인됨

PASS	Hard-ID	중매인 이름	END

· 등록되지 않은 중매인이거나 비밀번호가 틀림

NOPASS	Hard-ID	END

· 최고 경매가

PRICE	8Byte DATA	END

· 수량

QUANTITY	8Byte DATA	END

· 경매정보

INFO	경매정보	END

3) MIU - PC간 통신방법

MIU와 PC간 데이터 전송은 RS-232 통신방법을 이용하여 이루어진다.

▣ MIU 초기화

MIU에 전원이 인가되면 먼저 MIU내의 CPU, RAM 등이 초기화된 후 SIU 유무를 체크한다.

SIU 시스템을 효율적으로 관리하고 통신하기 위하여 SIU가 MIU에 몇 개가 접속되어 있는지 알아야 한다. 그래서 SIU가 MIU에 접속될 때 SIU패턴상에 5[V]를 연결하여 두고 MIU는 SIU 체크시 이 레벨을 감지하므로써 SIU의 연결을 알 수 있다.

▣ MIU -> PC data 전송

MIU에서 PC에 보내야 할 경매 데이터는 중매인 ID, 비밀번호, 경매가, 구매수량이다.

□ 통신 packet

MIU의 통신 패킷 구성시 Hard-ID는 SIU에서 넘어온 Hard-ID + SIU slot 번호 × 30 이다.

· 중매인 ID

STA	BROKER-ID	Hard-ID	중매인 ID	END	CRC

· 비밀번호

STA	S-CODE	Hard-ID	비밀번호	END	CRC

· 경매가

STA	PRICE	Hard-ID	경매가	END	CRC

· 수량

STA	QUANTITY	Hard-ID	수량	END	CRC

▣ PC -> MIU 데이터 전송

PC에서 MIU에 보내야 할 경매 데이터는 중매인 이름, 비밀번호 통과여부, 최고 응찰가, 수량, 경매정보이다.

□ 통신 packet

· 중매인 확인됨

STA	PASS	Hard-ID	중매인 이름	END	CRC

· 등록되지 않은 중매인이거나 비밀번호가 틀림

STA	NOPASS	Hard-ID	END	CRC

· 최고 응찰가

STA	PRICE	최고응찰가	END	CRC

· 수량

STA	QUANTITY	수량	END	CRC

· 경매정보

STA	INFO	경매정보	END	CRC

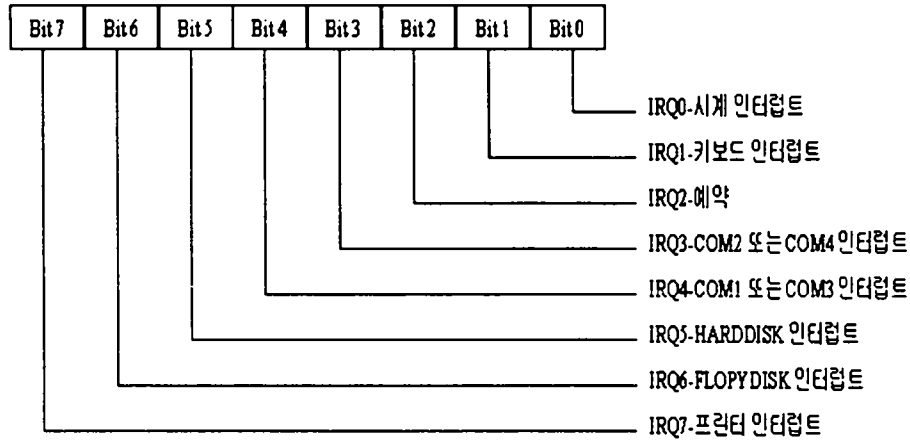
다. 컴퓨터의 직렬통신 프로그램

1) 직렬 통신 인터럽트 루틴

UART의 인터럽트를 사용 가능하게 하려면 우선 UART의 인터럽트 가능 레지스터(IER)에서 적당한 비트를 선택함으로써 적당한 인터럽트를 선택해야 한다. 예를 들어 UART의 수신 버퍼 레지스터(RBR)에 사용 가능한 한 문자가 있을 때마다 UART가 인터럽트를 생성하도록 하려면, 인터럽트 가능 레지스터의 비트 0을 1로 세트해야 한다. 그런 다음 프로그램은 UART의 모뎀 제어 레지스터(MCR)에 들어 있는 GPO2 비트를 1로 세트해야 한다. 그러나 현재 UART 한 문자가 RBR에 놓이는 모든 경우에 인터럽트 생성을 시작할 완전한 준비가 되어 있음에도 불구하고 이러한 인터럽트는 아직은 CPU에 의해 인식되지 않을 것이다. UART의 인터럽트가 완전히 사용 가능하도록 하려면 해당 컴퓨터의 8259 주변장치 인터럽트 컨트롤러(PIC)가 UART의 인터럽트를 인식하도록 세트되어야 한다. 그리고 일단 UART의 인터럽트를 인식하도록 셋업되면 8259는 적당한 CPU 인터럽트를 생성한다. 따라서 CPU는 현재의 작업을 멈추고 원하는 인터럽트 서비스 루

틴(ISR)을 호출한다.

주변 장치 인터럽트 컨트롤러(PIC)가 UART의 인터럽트를 인식하도록 세트하려면 직렬통신 프로그램이 PIC의 인터럽트 매스크 레지스터(IMR)에 들어 있는 적당한 IRQ 라인을 사용 가능하도록 해야한다. 인터럽트 매스크 레지스터는 포트 21H로부터 읽거나 쓰도록 주소가 지정된다. 아래에 인터럽트 매스크 레지스터(IMR)에 의해 세트되는 IRQ 라인을 나타내었다.



8259의 인터럽트 마스크 레지스터(IMR)

이 그림에서 COM1과 COM3는 IRQ4를 공유하며 COM2와 COM4는 IRQ3를 공유한다는 것을 알 수 있다.

일단 직렬 통신 프로그램이 알맞은 인터럽트 매스크 레지스터(IMR) 라인을 세트하였으면 그 직렬 포트의 인터럽트는 완전히 사용 가능해진다. 즉 COM1 또는 COM3에서 인터럽트가 발생할 때마다 CPU는 INT 0 CH의 IRQ4 ISR 루틴을 호출한다. 또한 COM2 또는 COM4에서 인터럽트가 발생할 때마다 CPU는 INT 0BH의 IRQ3 ISR 루틴을 호출한다. 그리고 ISR 루틴으로의 입력에 따라 직렬 통신 프로그램은 UART의 인터럽트 인식 레지스터(ISR)를 검사하여 어떤 종류의 인터럽트가 발생했는지를 알아낸다.

Serial 도구상자의 경우에는 인터럽트 인식 레지스터(ISR)가 UART의 수신 버퍼 레지스터(RBR)에 준비되어있는 한 문자를 인식하고 이 문자를 읽어들여서 알맞은 버퍼 영역에 저장한다. 이때 serial 프로그램에서는 그 밖의 다른 모든 인터럽트는 무시한다.

3. TV수상기-컴퓨터간 인터페이스 구성

가. 인터페이스 구성

컴퓨터의 모니터상에 나타난 화면을 TV수상기에 나타내기 위해 컴퓨터와 TV수상기 사이에 인터페이스장치를 구성한다.

컴퓨터 모니터에 표현된 내용을 대형 TV화면이나 빔프로젝터를 사용하여 크게 표현할 수 있어 프리젠테이션, 애니메이션 구현, 그래픽 녹화 및 디스플레이 구현, 게임용 화면표현 그리고 멀티미디어 동영상의 구현을 위해 주로 사용되는 TV wider를 이용하여 인터페이스를 구성하였다.

TV wider는 크게 외장형과 내장형으로 나뉠 수 있고 TV wider card의 형태에 따라 다르다. 본 시스템에는 외장형인 TVW-200을 사용하였다.

TV wider의 구성요소는 AV컨넥터, S-Vidio 컨넥터, VGA입력 컨넥터, VGA출력 컨넥터, 전원 잭, UP 버튼, DOWN 버튼, LEFT 버튼, RIGHT 버튼, RESET 버튼, DIP SWITCH 등이다.

나. 구동용 프로그램

구동용 소프트웨어는 최적의 화면을 얻고 화면을 정렬하는 역할을 한다.

1) DOS에서 화면정렬

원하는 기능을 수행하기 위해 hot key를 사용하며 기능은 다음과 같다.

HOT KEY	화면정렬
CTRL+ALT+UP ARROW	SCTOLL DISPLAY UP
CTRL+ALT+DOWN ARROW	SCTOLL DISPLAY DOWN
CTRL+ALT+LEFT ARROW	SCTOLL DISPLAY LEFT
CTRL+ALT+RIGHT ARROW	SCTOLL DISPLAY RIGHT
CTRL+ALT+F	TOGGLE DISPLAY FRONT
CTRL+ALT+U	SELECT UNDERSCAN MODE
CTRL+ALT+O	SELECT OVERSCAN MODE
CTRL+ALT+T	TOGGLE DISPLAY TYPE

2) Windows에서 화면정렬

윈도우즈에서 화면을 정렬 하기위해 프로그램을 사용하면 다음 그림과 같은 대화상자가 나타난다. 프로그램 그룹의 TV wider 아이콘을 더블 클릭하고, 원하는 화면을 정렬하기 위해 화살표 버튼을 클릭하면 화면을 정렬할 수 있다.

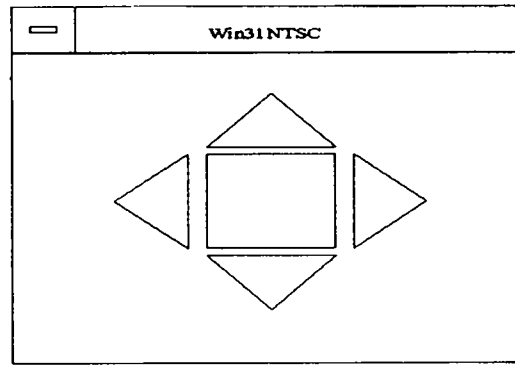


그림 9 화면정렬 버튼

4. 측정장치-컴퓨터간 인터페이스 구성

경매하고자 하는 농산물의 내적사양인 당도, 산도, 수분, 중량값들을 측정하여 상품의 질을 평가하기 위한 객관적인 자료를 중매인에게 제공하기 위해 측정장치와 컴퓨터간에 인터페이스 장치를 구성하였다. 네 개의 측정장치를 컴퓨터에 부착된 하나의 I/O port에 접속하기 위해 멀티 포트로 하드웨어를 구성하므로써 측정장치에 발생되는 자료인 물리적인 데이터를 전기적인 데이터로 변환시켜 컴퓨터의 경매진행 화면에 전송하였다.

가. 인터페이스 구성

중매인들에게 구입하고자 하는 품목을 평가할 수 있도록 농산물에 대한 당도, 산도, 수분, 무게 등과 같은 내적 정보를 제공해야 한다. 당도와 산도는 즙을 내고 수분은 얇은 조각을 내어 해당 측정장치로 그 농도를 측정하고, 중량은 전자저울로 무게를 측정한 후 데이터들을 컴퓨터로 전송한다. 컴퓨터에 전송된 데이터는 경매진행 화면의 해당 란에 표시된다. 컴퓨터의 모니터상에 나타난 내용은 중매인들이 볼 수 있는 화면인 TV수상기로 전송되고 이것에 의해 중매인은 농산물의 질을 파악할 수 있게된다.

이 연구의 목표는 측정장치에 의해 측정된 데이터를 컴퓨터에 전송할 수 있도록 각종 측정장치와 멀티포트간 인터페이스를 구성하고 멀티포트를 PC에 접속하여 측정장치-컴퓨터간 인터페이스의 하드웨어를 구성하고, 관련 프로그램을 작성하는 것이다.

1) 무게 측정장치 인터페이스

가) 접속

무게측정기(EB-32KS, SHIMADZU, JAPAN)에는 RS-232C 인터페이스를

위한 25핀 커넥터가 마련되어 있으며, 그림 10은 멀티포트와 무게측정기와의 결선도이다.

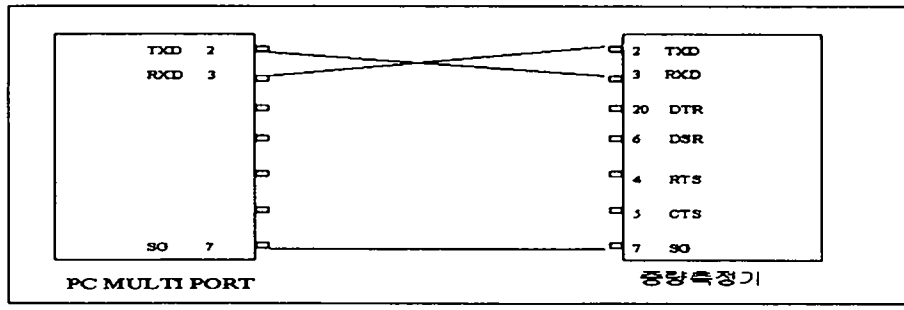


그림 10 무게측정기 결선도

나) RS-232C 커넥터

(1) RS-232C 커넥터

멀티포트 커넥터 25개, 무게측정장치 커넥터 25개의 PIN중 각각 3개 PIN만을 사용한다.

핀번호	신호	I/O	의 미
2	TXD	출력	데이타 출력
3	RXD	입력	데이타 입력
7	SG		GROUND

(2) 데이터 형식

▣ 아스키(JIS)코드

▣ 1200BPS, Non parity, 8-bit length, 1 stop bit

아래의 "_"는 공백(space), "C/R"은 carriage return을 나타낸다.

▣ 출력 데이터 포맷

-	_	_	9	4	.	5	6	7	g	_	C/R

+ 극성 : 표시안함　- 극성 : '-'를 표시함

▣ OL 출력 포맷

-	_	_	_	_	O	L	_	_	_	_	C/R

▣ 입력 데이터 포맷

명령번호	C/R

(3) 명령어 코드

명령 코드	기　능	내　용
T	용기 제거	'TARE'키에 해당
D05	인쇄(1회 출력)	'PRINT'키에 해당
D06	자동 인쇄	SETTING 시간에 따라 출력
D01	연속 출력	저울의 데이터를 연속 출력
D02	안정시연속 출력	안정시 저울의 데이터를 연속출력
D09	출력 정지	자동인쇄 및 연속 출력 해제

2) 산도 측정장치 인터페이스

가) 접속

산도측정기(HI-9017, HANNA, ITALY)에는 RS-232 인터페이스를 위한 25핀커넥터가 마련되어 있으며, 해당핀만을 결선하여 멀티포트와 접속한다.

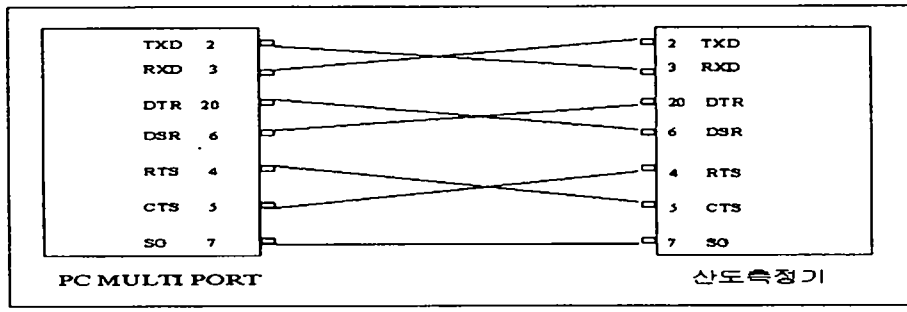


그림 11 산도측정기 결선도

나) RS-232C 커넥터

(1) RS-232C 커넥터

멀티포트 커넥터 25개, 산도 측정장치 커넥터 25개의 PIN중 각각 7개 PIN만 사용한다.

핀번호	신호	I/O	의 미
2	TXD	출력	데이타 출력
3	RXD	입력	데이타 입력
4	RTS	출력	전송요구
5	CTS	입력	전송해제
6	DSR	입력	데이터 세트 준비완료
7	SG		GROUND
20	DTR	출력	DTR 준비완료

(2) 데이터 형식

▣ 아스키(JIS)코드

▣ 1200BPS, Non parity, 8-bit length, 1 stop bit

(3) 명령어 코드

명령코드	기 능
PHR	PH범위 설정
MVR	mV범위 설정
PH?	PH값 요구
MV?	mV값 요구
TM?	온도값 요구

3) 수분 측정장치 인터페이스

가) 접속

수분측정기(AD-4712, A&D, U.S.A.)에는 그림 12와 같이 RS-232C 인터페이스를 위한 25핀 커넥터가 마련되어 있으며 그에 맞는 결선을 하여 멀티포트와 접속한다.

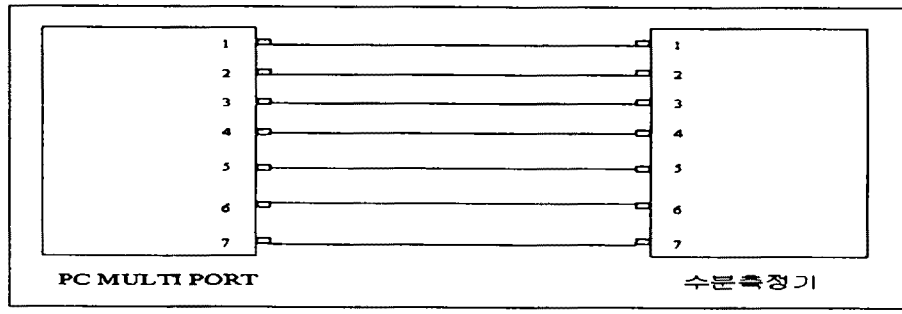


그림 12 수분측정기 결선도

나) RS-232C 커넥터

(1) 결선방법 - 25개의 PIN중 7개의 PIN만을 결선하여 사용한다.

(2) 데이터 형식

▣ 아스키(JIS)코드

▣ 2400BPS, Even parity, 7-bit length, 1 stop bit

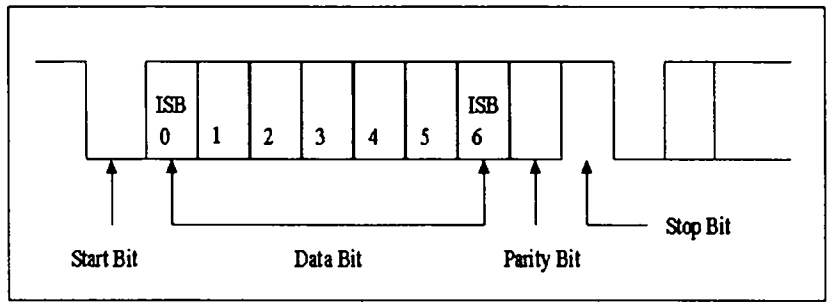


그림 13 수분측정기 데이터 규격

▣ 출력 데이타 포맷

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
*	S	e	t		T	E	M	P			4	0	0	°	C
*	W	e	t		W	T			0	9	.	7	9		g
*	D	r	y		W	T									g
	0	0	m		3	0	s				0	3	9	°	C
											0	0	.	0	%

4) 당도 측정장치 인터페이스

가) 접속

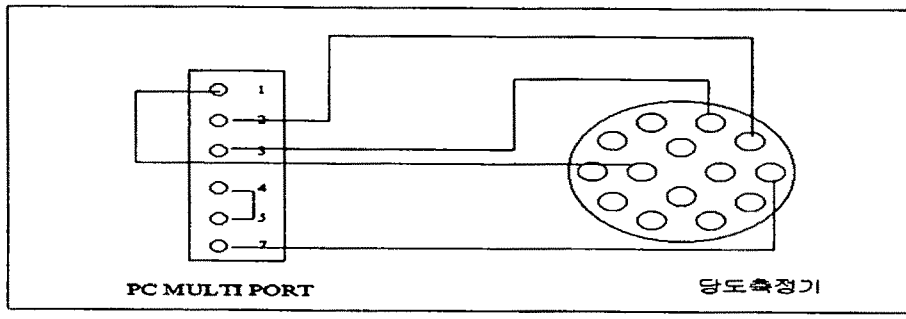


그림 14 당도측정기 결선도

당도측정기(RX-5000, ATAGO, JAPAN)에는 RS-232C 인터페이스를 위한 14핀 커넥터가 마련되어 있으며 해당 핀만을 결선하여 멀티포트와 접속한다.

나) RS-232C 커넥터

(1) 결선방법 - 멀티포트 25개 당도측정기 14개 PIN 중 4개 PIN만을 결선하여 사용한다.

(2) 데이터 형식

▣ 아스키(JIS)코드

▣ 1200BPS, Non parity, 8-bit length, 1 stop bit

▣ 출력 데이타 포멧

YYMMDD, HHMM, nD = *.*****, Brix = **.**,

nDt=*.*****, UUUUUUUU = XXXXXXX, TT.T C/RL/F

YYMMDD : 년 월 일

HHMM : 시간과 분

*.*****　　: 측정값

UUUUUUUU　: 10문자 내의 USER SCALE 명칭

XXXXXXX　　: USER SCALE에서 측정값

TT.T　　: 온도

C/RL/F　　: 캐리지가 돌아감

▣ 수신 데이터 형식

컴퓨터의 지시대로 RX-5000이 영점 설정, 측정시작과 같은 작동을 하게 하려면 컴퓨터에서는 다음과 같은 명령을 사용한다.

* ZERO C/CR/F : "0점 설정"

* START C/RL/F : "측정시작"의 경우

다) 당도기 통신 setup

RX-5000에는 다양한 기능설정을 위한 설정 모드가 있다. 측정 모드의 디스플레이에서 MENU key(SW1)를 눌러 SETUP MENU로 가서 원하는 setup mode를 선택한다.

SETUP MENU의 item은 아래와 같다.

(1) PRINTER SET　　: 프린터로 프린트할 item설정

(2) RS232C SET　　: 통신 mode 설정

(3) TIME SET　　: 날짜와 시간 지정

(4) WAIT TIME SET　: 다음 측정을 위한 대기시간 설정

(5) CONTRAST SET　: 디스플레이의 콘트라스트 조절

(6) nDt SET　　: nDt측정을 위한 reference temperature와 온도 보정 계수 설정

(7) USER SCALE SET : user scale 설정

측정 데이터를 처리하거나 측정 작동을 조절하기 위하여 외부 장치(개인용 컴퓨터)와 RX-5000을 연결할 때에는 두 매체 사이의 정보 전송을 위한 통신모드 설정이 필요하다. 이 메뉴에서는 그러한 통신모드를 설정할 수 있다.

SETUP MENU 디스플레이에서 커서로 "2 RS-232C SET" 을 선택하고 ENTER key를 누르면 화면은 통신모드 설정을 위한 RS-232C SET 메뉴로 바뀐다. 디스플레이의 오른쪽 ITEM 박스에서는 각 item의 현재 상황을 볼 수 있고, 왼쪽의 MENU 부분에서는 커서로 선택할 수 있는 4가지 item을 볼 수 있다. 현재 설정되어있는 ITEM 박스내의 상황 중 바꾸려는 item이 있으면 ↓와 ↑key로 item을 선택한 후 ENTER key를 누른다.

(1) BAUD RATE

(가) 커서를 MENU 부분의 BAUD RATE로 옮기고 ENTER key를 누른다.

(나) 화면의 MENU 부분에 4가지의 baud rate이 나타난다.

(다) ITEM 박스내에서 지정된 값은 QUIT key를 눌러서 설정 메뉴로 돌아가도 유효하다.

(2) Data 길이

(가) MENU 부분에서 ↓와 ↑key로 DATA LENGTH를 선택하고 ENTER key를 누른다.

(나) 왼쪽 부분에 선택할 수 있는 두 종류의 데이터 길이가 나타난다.

커서를 7bit 또는 8bit를 선택하고 ENTER key를 누르면 선택한 데

이터의 길이가 ITEM 박스내에 표시된다. 설정된 값은 현재 유효하다.

(다) ITEM 박스내에서 지정된 값은 QUIT key를 눌러서 설정 메뉴로 돌아가도 유효하다.

(3) PARITY

(가) 디스플레이 MENU에서 ↓와 ↑key로 PARITY를 선택하고 ENTER key를 누른다.

(나) 왼쪽 부분에 선택할 수 있는 3가지의 parity 조건이 나타난다. 커서로 원하는 조건을 선택하고 ENTER key를 누르면 선택한 parity의 조건이 ITEM 박스에 나타난다. 설정된 값은 현재 유효하다.

(다) ITEM 박스내에서 지정된 값은 QUIT key를 눌러서 설정 메뉴로 돌아가도 유효하다.

(4) STOP BIT

(가) 디스플레이의 MENU에서 ↓와 ↑key로 STOP BIT를 선택하고 ENTER key를 누른다.

(나) 왼쪽 부분에 선택할 수 있는 2가지의 stop bit이 나타난다.
커서로 원하는 stop bit를 선택하고 ENTER key를 누르면 선택한 stop bit이 ITEM 박스내에 나타난다. 설정된 값은 현재 유효하다.

(다) ITEM 박스내에서 지정된 값은 QUIT key를 눌러서 설정 메뉴로 돌아가도 유효하다.

5) 멀티포트 인터페이스 구성

당도, 산도, 수분, 중량 측정장치를 하나의 컴퓨터 I/O port에 접속하기 위해 여러개의 직렬통신이 가능한 멀티포트를 사용하였다.

포트의 숫자 제약으로 시리얼 포트의 경우는 3개 이상의 포트를 함께 사용할 수 없게 되어있다. 이것은 PC의 슬롯이 적기 때문에 슬롯을 더 추가하여 사용해야 한다. 본 연구에서는 4개의 계측기로 부터 발생되는 RS232C 형의 데이터를 사용하기 때문에 포트 제약을 받고 있으며 이런 포트의 제약을 없애기 위해 멀티포트를 사용하였다.

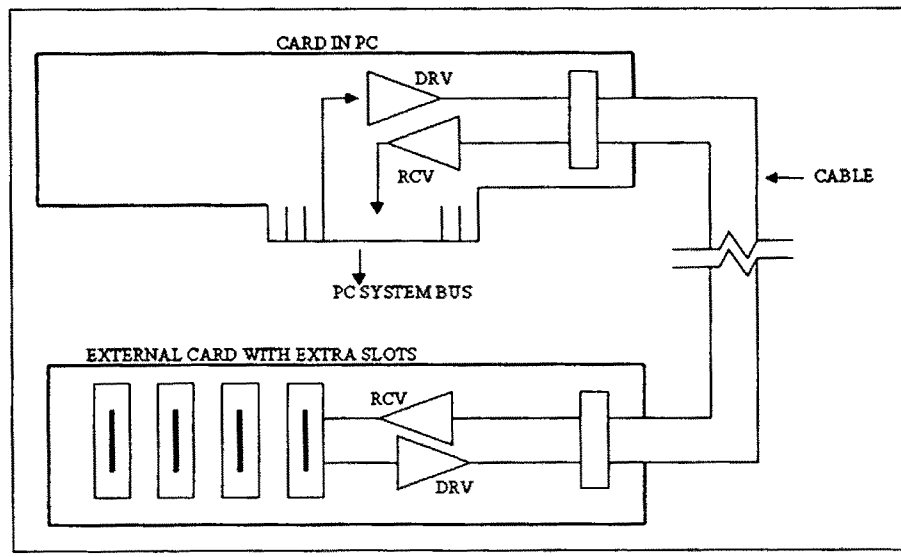


그림 15 multiport 블록도

멀티포트는 본체의 슬롯에 꼽히는 카드와 본체 외부의 확장슬롯용 카드 두 부분으로 이루어진다. 그림 15는 설계에 대한 개념과 구성요소를 개략적으로 나타내었다. 확장 포트는 시스템 유니트 카드, 출력을 드라이브 및 수신하는 카드 그리고 새로운 카드 슬롯을 포함한다.

PC 버스는 단순하기 때문에 기본개념은 전원의 재 공급과 출력 신호를 케이블로 구동시키는 것이다. 드라이브 회로의 주된 기능은 케이블의 정전용량을 나누어 시스템 버스에 신호의 지연과 왜곡을 막는다. 맞은편 케이블

에는 신설 확장카드가 있어서 여기에서도 전원의 재 공급과 신호를 구동한다. 마찬가지로 수신회로는 새로운 슬롯에 부담을 주지 않고 케이블의 정전용량을 나눈다. 버스 파워공급을 위해 TI사의 SN74S 240과 SN74LS 240과 같은 버퍼와 드라이버를 이용하여 회로를 구성하고 있다.

나. 인터페이스 제어프로그램 개발

측정 대상에 대한 당도, 산도, 수분, 무게의 값을 측정 장치들로 부터 측정하여 발생된 데이터는 표준 통신규약인 RS232C를 사용하여 컴퓨터에 전송된다.

RS232C는 오래 전부터 데이터 통신 장비와 데이터 처리장치간의 접속에 보편적으로 사용된 규격이다. 터미널, 프로터, 논리분석기, 테이프 드라이버, 프린터 등 각종 계측기의 신호발생기 장비들은 대체로 이 규격을 갖추고 있다. 응용분야에 RS232C를 이용하면 TTL전위와 TTL전위가 아닌 경우에 이 규칙이 필요하다. IBM PC에 사용되고 있는 시리얼 카드가 이 규칙의 좋은 예이다.

RS232C신호선의 특성으로는 50ft에서 한 방향, 단일 단말에 대해서 최대 20Kbps까지 보장해 두고 있다. 논리"1"은 5~15[V]이며, 논리"0"은 0~-15[V]로 정의한다.

그림 16은 RS232C에서 사용하는 전위사이의 변환 회로이다.

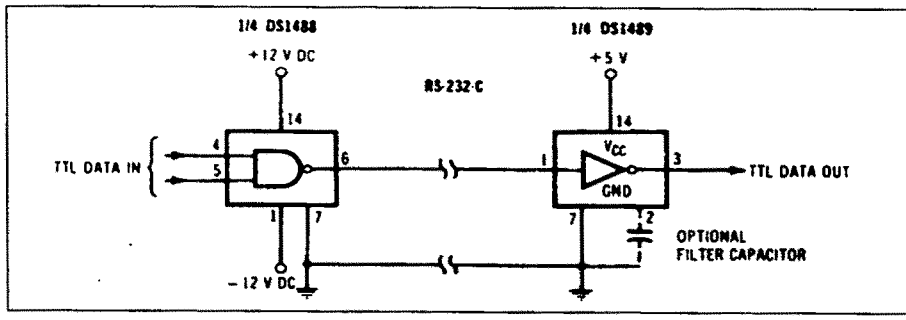


그림 16 RS232C 전위변환 회로

직렬 데이터는 한 번에 한 비트씩 전송되므로 데이터 터미널장치(DTE)와 데이터 통신장치(DCE) 사이의 데이터전송 조작을 동기화 (synchronize)할 방법이 있어야 한다. 따라서 먼저 해야 할 일은 데이터의 전송 속도를 맞추는 것이다. 직렬 통신 라인의 속도는 보드율(baud rate)로 나타낸다. 보드율은 데이터 터미널 장치와 데이터 통신장치 사이에 전송되는 데이터의 초당 비트 수이다. 그리고 전송되는 데이터들을 동기화하기 위해 두 직렬 장치는 각 데이터 문자에 대하여 시작 비트, 데이터 비트, 패리티 비트, 정지 비트를 사용한다.

1) 시작 비트

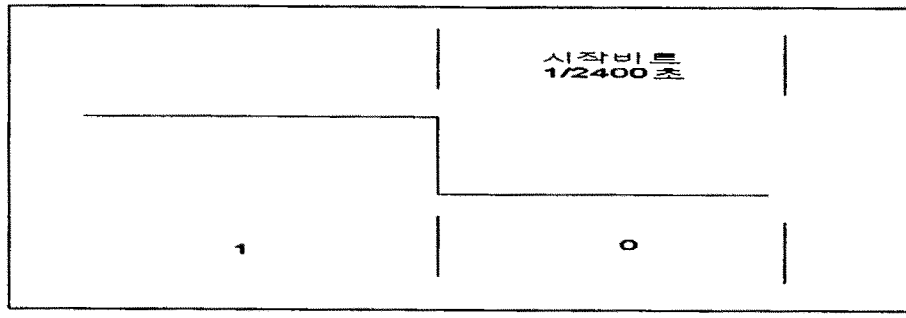


그림 17 시작비트

직렬 장치가 전송중이 아닐 때에는 TD 라인이 1로 세트된다. 문자 데이터의 전송을 시작하려면 직렬 전송 장치는 시작 비트를 전송하며, 시작 비트는 논리적 0으로 보드율의 타임 프레임 동안 지속된다. 시작 비트를 보냄으로써 데이터 통신장치(DCE)는 단순히 데이터 터미널장치(DTE)의 TD 라인이 논리적 1의 상태에서 논리적 0인 상태로 변화하는 것을 기다리기만 하면 전송기를 동기화할 수 있다. 그런 다음 데이터 통신장치는 1비트 타임의 절반 시간만 멈춰 있다가 모든 데이터 전송이 끝날 때까지 1비트 타임마다 한 번씩 들어오는 데이터의 샘플링을 시작한다. 비트 시간의 절반 시간 동안 멈춰 있으면 데이터 통신장치는 들어오는 데이터 스트림을 각 비트의 중간 정도로 계속 샘플링하여 동기화를 유지할 수 있다. 만약 데이터통신장치가 이러한 멈춤을 제공하지 않는다면 데이터 터미널 장치와 데이터 통신장치의 미세한 클럭 속도의 차이 때문에 들어오는 비트 스트림에 혼란이 생긴다. 그림 17은 2400보드로 전송되는 시작 비트를 나타낸 것이다. 이 그림에서 알 수 있듯이 시작 비트는 1/2400초(1타임 프레임) 동안 지속된다

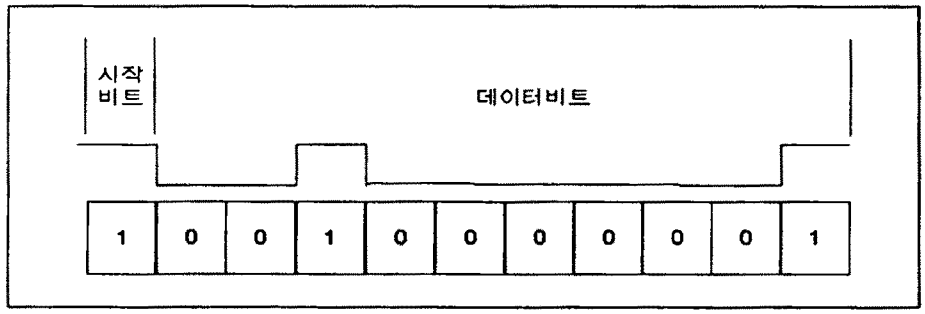


그림 18 데이터 비트

2) 데이터 비트

직렬 장치로 시작 비트를 전송한 다음에는 5~8개의 데이터 비트를 전송한다. 더 많은 데이터 전송으로는 7~9개가 전송되기도 한다. 그림 18은 문자 A(41H)가 8개의 데이터 비트를 사용하여 전송되는 것을 나타낸 것이다.

3) 패리티 비트

데이터 링크에는 에러 검색을 위해 패리티 비트가 필요하다. 패리티의 다섯 가지 기본적인 형태는 없음(NONE), 짝수(EVEN), 홀수(ODD), 표시(MARK), 공백(SFACE)이다.

없음(NONE) 패리티는 패리디 비트 없이 전송되는 문자를 의미한다. 이것은 8개의 데이터 비트가 전송될 때의 일반적인 세팅이다.

짝수(EVEN) 패리티는 직렬 장치를 통해 전송되는 데이터 비트들 안에 들어 있는 논리적 1의 갯수가 홀수이면 그 데이터 비트 다음에 1이 전송된다. 또한 그 데이터 비트들 안에 들어 있는 논리적 1의 갯수가 짝수이면 그 데이터 비트 다음에 0이 전송된다. 따라서 짝수 패리티는 각 문자에 들어 있는 논리적 1의 갯수를 언제나 짝수로 만들어줌으로, 수신 장치에서 논리적 1의 갯수가 홀수인 문자가 수신되면 에러가 발생한다.

홀수(ODD) 패리티는 직렬 장치를 통해 전송되는 데이터 비트들 안에 들어있는 논리적 1의 갯수가 짝수이면 그 데이터 비트 다음에 1이 전송된다. 또한, 그 데이터 비트들 안에 들어 있는 논리적 1의 갯수가 홀수이면 그 데이터 비트 다음에 0이 전송된다. 따라서 홀수 패리티는 각 문자에 들어 있는 논리적 0의 갯수를 언제나 홀수로 만들어 주므로, 수신 장치에서 논리적 1의 갯수가 짝수인 문자가 수신된다면 에러가 발생한다.

표시(MARK) 패리티는 해당 문자의 데이터 비트들 다음에 논리적 1의 패리티 비트가 전송된다. 공백(SPACE) 패리티는 해당 문자의 데이터 비트들

다음에 논리적 0의 패리티 비트가 전송된다. 그림 19는 직렬 장치가 7개의 데이터 비트와 짝수(EVEN) 패리티를 사용하여 문자 G(4SH)를 전송하는 방법을 나타낸 것이다. 그림에서 알 수 있듯이 이 전송에서는 짝수 패리티를 유지하기 위해 패리티 비트에 논리적 1이 전송되어야 한다.

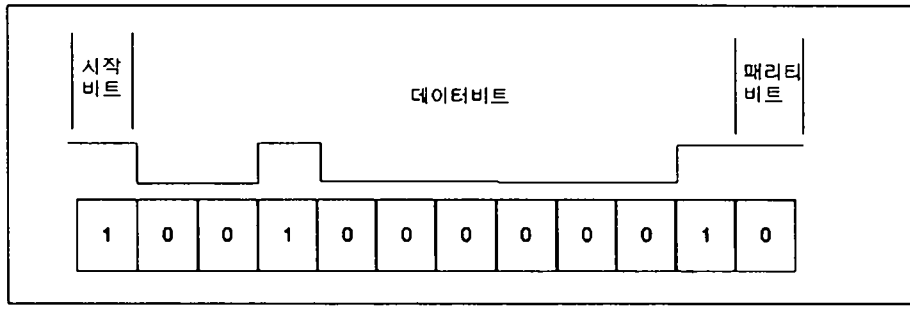


그림 19 패리티 비트

4) 정지 비트

시작 비트가 한 문자의 시작을 지정하듯이 정지 비트는 한 문자의 끝을 지정한다. 직렬 전송 장치가 전송하는 정지 비트는 언제나 논리적 1이다. 이것은 또한 TD 라인을 세트한다. 그러므로 직렬 인터페이스는 정지 비트에 의해 적당한 아이들(idle) 상태가 된다. 그림 20은 문자 G(4SH)가 패리티 없이 여덟 개의 데이터 비트를 사용하여 완전하게 전송되는 방법을 나타낸 것이다

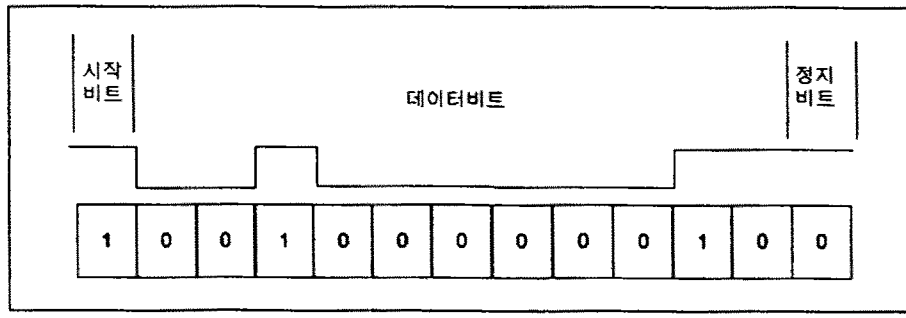


그림 20 정지비트

다. 당도, 산도, 수분, 중량값 디스플레이 프로그램 개발

당도, 산도, 수분, 중량값의 측정을 위한 측정기와 PC간에는 RS232C 통신을 하며 PC에 내장된 RS232C 통신을 위한 포트는 2개이다. 본 연구에서는 네 종류의 측정기와 MIU를 PC에 연결해야 하므로 총 5개의 포트가 필요하다. 따라서 포트의 제한을 극복하기 위해 멀티포트를 사용하였으며, 멀티포트와 PC간 통신을 위한 프로그램을 개발하였다.

다음은 DBXDoc 클래스의 생성자 부분이며 경매프로그램(DBX)을 실행하면 아래와 같이 프로그램이 실행된다.

```
CDBXDoc::CDBXDoc()
{
    m_pDataForm = NULL;
    m_pDataGmjh = NULL;
    m_pDataGmi  = NULL;

    opn_wnd  = FALSE;
```

```
    miu_port    = 2;
    sugar_port  = 4;
    ph_port     = 5;
    mos_port    = 6;
    weight_port = 7;

    xcom        = new CComm(miu_port);
    sugarCom    = new CComm(sugar_port,9600);
    phCom       = new CComm(ph_port,1200);
    mosCom      = new CComm(mos_port, 2400, EVENPARITY, 7);
    weightCom   = new CComm(weight_port,CBR_1200);
}
```

다음은 각 계측기에 대한 멀트포트를 설정해 주는 부분이며 사용한 멀티포트는 8개의 포트를 가진다.

sugar_port=4; 당도계와 RS232C통신을 위한 포트는 4번 포트로 설정.

ph_port= 5; PH계와 RS232C통신을 위한 포트는 5번 포트로 설정.

mos_port= 6; 수분계와 RS232C통신을 위한 포트는 6번 포트로 설정.

weight_port=7; 중량계와 RS232C통신을 위한 포트는 7번 포트로 설정.

다음은 각 계측기의 포트를 사용할 수 있는 권한을 얻기 위해 PC에 메모리 영역을 설정하는 부분이다. 각 포트에 대한 메모리영역을 설정했고, 9600, 1200, 2400, CBR_1200 등은 각 계측기의 특성에 의해 PC와 계측기간의 통신 속도를 나타낸다.

```
    sugarCom    = new CComm(sugar_port,9600);
    phCom       = new CComm(ph_port,1200);
```

```
mosCom    = new CComm(mos_port, 2400, EVENPARITY, 7);
weightCom = new CComm(weight_port,CBR_1200);
```

경매프로그램을 실행시키면 위의 데이터는 초기값으로 설정되며, 경매프로그램에서 각 포트간의 연결을 위한 프로그램은 아래와 같다.

```
if (!GetCom(pDoc->sugar_port)->bConnected)
        OnConnect(pDoc->sugar_port);
if (!GetCom(pDoc->ph_port)->bConnected)
        OnConnect(pDoc->ph_port);
if (!GetCom(pDoc->mos_port)->bConnected)
        OnConnect(pDoc->mos_port);
if (!GetCom(pDoc->weight_port)->bConnected)
        OnConnect(pDoc->weight_port);
```

위의 프로그램은 경매진행화면 상의 GM Load 버튼을 클릭했을 때 당도, 산도, 수분, 중량계와 PC간의 RS232C 통신을 위한 포트가 연결되어 있지 않을 경우 계측기와 포트를 연결하기 위해 void CDBXView::OnConnect (int port) 함수를 사용하여 각 포트를 연결하게 된다.

void CDBXView::OnConnect(int port) 함수는 멀티통신을 위한 준비가 됐을 경우 true를 각 계측기간의 thread를 이용하여 각각의 계측기에서 데이터를 얻어올 수 있는 준비를 한다. 하지만 연결에 실패했을 경우와 thread를 생성치 못했을 경우는 error를 발생시킨다.

```
if (GetCom(pDoc->sugar_port)->bConnected) {
```

```
        OnDisconnect(pDoc->sugar_port);
    }
    if (GetCom(pDoc->ph_port)->bConnected) {
        OnDisconnect(pDoc->ph_port);
    }
    if (GetCom(pDoc->mos_port)->bConnected) {
        OnDisconnect(pDoc->mos_port);
    }
    if (GetCom(pDoc->weight_port)->bConnected) {
        OnDisconnect(pDoc->weight_port);
    }
```

위 프로그램은 경매진행화면 상의 시작 버튼을 클릭했을 경우에 대한 것이다. 시작버튼을 클릭하면 경매가 시작된 상태이므로 이미 계측기의 데이터를 모두 받아서 경매진행 화면 상에 디스플레이 되어있으므로 더 이상 계측기의 데이터를 받을 필요가 없다. 이러한 경우에 void CDBXView::On Disconnect(int port)의 함수를 이용하여 모든 계측기의 포트는 닫아 주고 MIU와 통신을 위한 준비를 하게 된다.

이런 상황은 경매가 진행되는 동안에 계속 나타나게 되는데 경매중에 다음 버튼을 클릭해서 다음 경매로 넘어갈 경우 PC는 경매에 필요한 계측기의 데이터를 얻기 위해 void CDBXView::OnConnect(int port)의 함수를 이용하여 다시 포트들을 연결시켜 각 계측기의 데이터를 전송받아 다음 경매를 준비하게 되고, 경매가 종료될 경우나 완전히 프로그램을 종료시켰을 경우 void CDBXView::OnDisconnect(int port)의 함수를 사용하여 모든 포트의 사용권한을 박탈한다.

```
CComm* CDBXView::GetCom(int port)
{
    CDBXDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);

    if(port == pDoc->miu_port)
                    return pDoc->xcom;
    else if(port ==  pDoc->sugar_port)
                    return pDoc->sugarCom;
    else if(port == pDoc->ph_port)
                    return pDoc->phCom;
    else if(port == pDoc->mos_port)
                    return pDoc->mosCom;
    else if(port == pDoc->weight_port)
                    return pDoc->weightCom;
    else
            return pDoc->xcom;
}
```

위 프로그램은 MIU와 계측기의 데이터를 얻기 위해 그 포트의 포인터를 얻어오는 함수로서 포트로 데이터가 들어오면 Thread로 잡아두었던 각 포트간의 통신을 계속하면서 어느 포트로 데이터가 들어왔는지 확인하는 단계이다. 포트로 들어온 데이터는 여기서 어느 계측기에서 데이터가 들어왔는지를 확인하여 그 데이터가 들어온 포트의 포인터를 얻어서 프로그램이 처음 시작될 때 설정해 두었던 CDBXdoc클래스안의 메모리 영역에 각 계

측기의 데이터를 메모리 영역에 넘겨 주게 된다.

```
LRESULT CDBXView::OnReceiveData(WPARAM wParam, LPARAM
lParam)
{
    char temp[50];
    CDBXDoc* pDoc = GetDocument();
    memset(temp,0,50);
    int port = (int)wParam;

    if(port == pDoc->miu_port)
            InpterpreterMiuData();

    else if(port ==  pDoc->sugar_port) {
            strcpy(temp, m_sugar);
            InterpreterSugarData(GetCom(pDoc->sugar_port)->rxbuf,
            temp);
            m_sugar = temp;
    }

    else if(port == pDoc->ph_port) {
            strcpy(temp,m_ph);
            InterpreterPhData(GetCom(pDoc->ph_port)->rxbuf, temp);
            m_ph = temp;
    }
```

```
        else if(port == pDoc->mos_port){
                strcpy(temp, m_mosture);
                InterpreterMosData(GetCom(pDoc->mos_port)->rxbuf,
                temp);
                m_mosture = temp;
        }

        else if(port == pDoc->weight_port) {
                strcpy(temp,m_weight);

InterpreterWeightData(GetCom(pDoc->weight_port)->rxbuf,temp);
                m_weight = temp;
        }

        UpdateData(FALSE);
        return 0;
  }
```

위 프로그램은 각 포트에 데이터를 받게 되면 그 데이터 값을 디스플레이 시키기 위해 각 계측기의 상황에 맞게 분류해 주는 역할을 한다.

void CDBXView::InterpreterSugarData(char *src, char *dest

당도계에서 넘어온 데이터를 디스플레이시켜 준다.

void CDBXView::InterpreterPhData(char *src, char *dest)

PH계에서 받은 데이터를 디스플레이시켜 준다.

void CDBXView::InterpreterMosData(char *src, char *dest)

수분계에서 받은 데이터를 디스플레이시켜 준다.

void CDBXView::InterpreterWeightData(char *src, char *dest)

중량계에서 받은 데이터를 디스플레이시켜 준다.

위의 디스플레이 함수들은 함수형은 비슷하나 실행루틴은 약간씩 차이가 있는데 그것은 각 계측기가 넘겨주는 데이터가 각각 다른 형식으로 넘겨주기 때문에 다른 형식의 데이터를 main에서 확연히 볼 수 있도록 하기 위해 실행루틴에 약간의 차이가 있다.

5. 경매프로그램 개발

본 프로그램은 GIU(Graphic Interface Unit) 환경에 부합할 수 있도록 window환경에 맞도록 작성하였다. Window환경에서 프로그램을 작성할 경우 OLE(Object Linking Embedding)방식이 가능하므로 프로그램의 고기능화 및 간결화가 가능하였다. 개발한 경매프로그램은 메뉴화면, 경매내용 입력화면, 경매진행 화면, 중매인관리 화면, 출하자관리 화면이며 그 내용은 다음과 같다

가. 메뉴화면

경매진행을 위해 경매에 필요한 제반사항을 입력하고 관리할 수 있는 데이터베이스 프로그램을 구축하여야 한다. 프로그램에 대한 사용이 편리하도록 상단에 한글 메뉴화면을 나타내었다. 메뉴에는 주메뉴와 서브메뉴가 있으며, 이중 필요한 메뉴를 선택하여 경매에 활용할 수 있도록 구성하였다. 주메뉴와 서브메뉴에 관한 내용은 다음과 같다. 프로그램을 실행하면 그림

21과 같은 초기 화면이 나타난다.

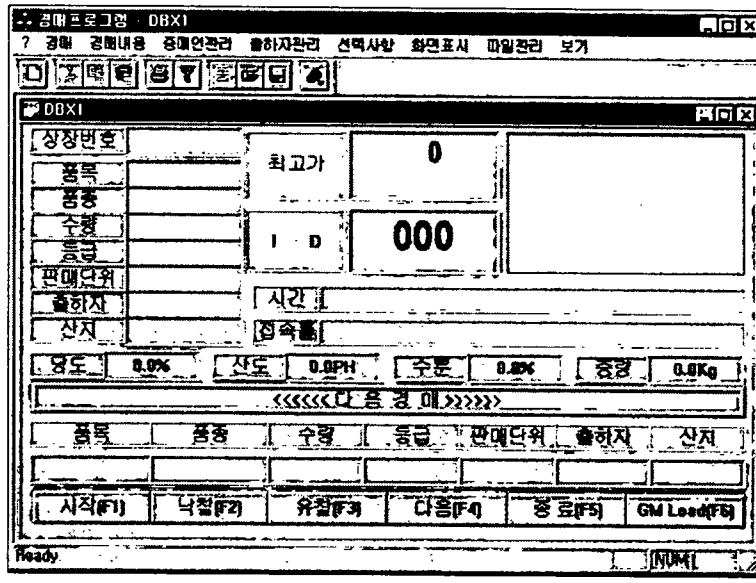


그림 21 초기경매 메인화면

1) ?

주메뉴 "?"에 대한 서브메뉴 "화상경매란?"을 선택하면 본 경매 프로그램에 대한 도움말을 볼 수 있다. 마우스를 사용하여 메뉴를 클릭하거나 F1을 누름으로써 도움말을 볼 수 있다. 도움말은 프로그램에 대한 간단한 설명과 각종 기능키의 역할을 설명한다.

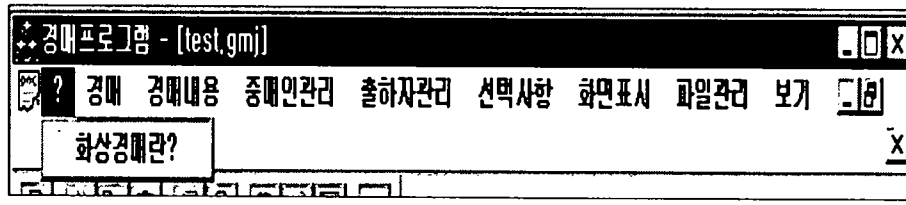


그림 22 ? 메뉴

2) 경매

이 메뉴에서는 경매에 대한 진행화면을 볼 수 있다. 경매 진행화면을 보기 위하여 "경매"에 대한 서브메뉴인 "경매진행"을 선택한다. 이 화면은 단말기에서 입력되는 정보들과 각종 센서로 부터 입력되는 수치등 경매에 대한 정보를 한눈에 볼 수 있다. 선택방법은 마우스를 사용하여 "경매진행"을 클릭하거나 F2를 누르면 된다.

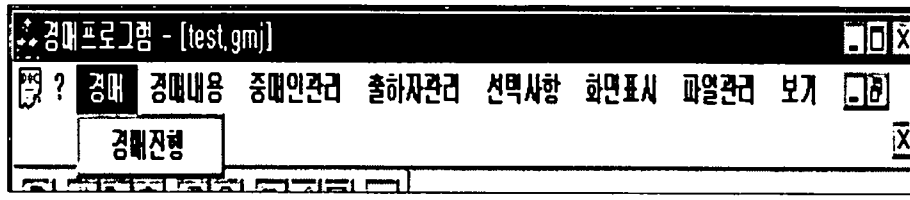


그림 23 경매 메뉴

3) 경매내용

이 메뉴에서는 경매할 품목에 대한 내용을 입력하고 조회할 수 있다. 경매진행을 위해 경매에 필요한 품목에 대한 자료를 입력하여야 한다. 여기에서 입력된 정보는 "경매진행" 화면에도 수록된다. 주 메뉴는 "경매내용"이며 서브메뉴는 "경매내용 입력 및 조회", "화일읽기", "화일저장" 등이 있다.

그림 24 경매내용 메뉴

4) 중매인 관리

중매인 관리 메뉴에서는 중매인에 대한 정보를 입력하고 조회한다. 경매

에 참여하는 중매인들은 반드시 중매인 관리화면에 등록해야 한다. "중매인 관리" 메뉴는 중매인에 대한 등록, 조회, 수정작업을 위한 메뉴이다. 주메뉴의 이름은 "중매인관리"이고 서브메뉴는 "중매인입력 및 조회", "화일읽기", "화일저장" 등이 있다.

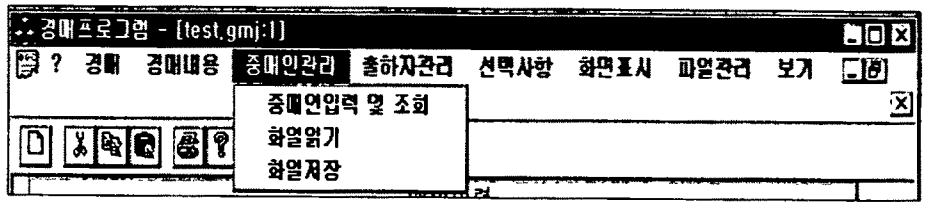


그림 25 중매인관리 메뉴

5) 출하자관리

이 화면은 출하자를 관리하기 위한 화면이며 출하자에 대한 정보를 입력 및 조회한다. "출하자관리" 메뉴는 출하자에 대한 등록작업, 조회 및 수정을 위한 메뉴이다. 주메뉴의 이름은 "출하자관리" 이며 서브메뉴는 "출하자입력 및 조회", "화일읽기", "화일저장" 등이 있다.

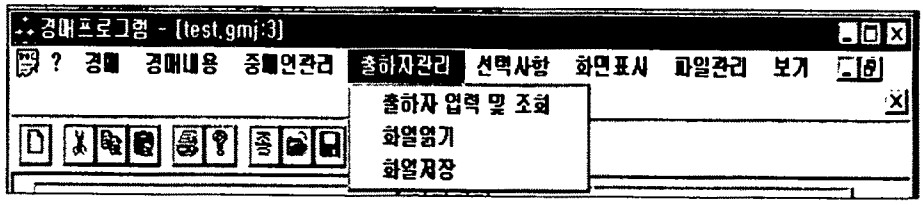


그림 26 출하자관리 메뉴

6) 기타 환경설정 및 유틸리티(선택사항, 화면표시, 파일관리, 보기)

경매에 필요한 부수적인 작업 프로그램으로 "선택사항" 메뉴는 경매 진행의 통신을 설정하며, "화면표시" 메뉴는 여러화면의 이동 또는 화면배치의 기능을, "파일관리" 메뉴는 윈도우 파일에 대한 정보를 관리하는 기능을 갖는다. "보기" 메뉴는 상태바와 툴바에 대한 화면표시를 제어한다.

메뉴를 선택하기 위해서는 마우스를 이용하거나 기능키(F10)와 화살표 키를 이용하여 선택할 수 있으며, 내용선택의 경우는 마우스를 더블 클릭하거나 enter 키를 누르면 된다.

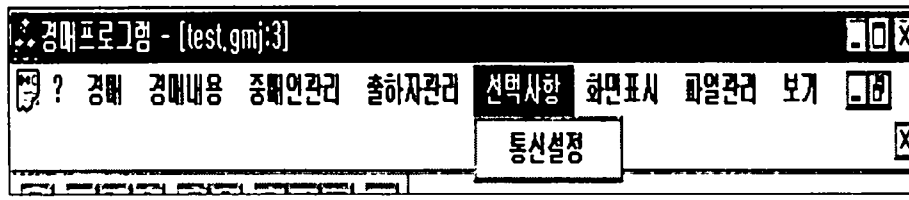


그림 27 선택사항 및 기타 메뉴

나. 경매내용 입력화면

1) 경매내용 - 화일읽기

이미 입력된 경매내용 화일을 읽어오는데 사용한다. 그림 28에서 볼 수 있는 바와 같이 경매진행 관련 화일의 확장자는 .GMJ이다. 주메뉴 "경매내용"에서 서브내용 "화일열기"를 실행하면 화일명이 화면에 표시된다.

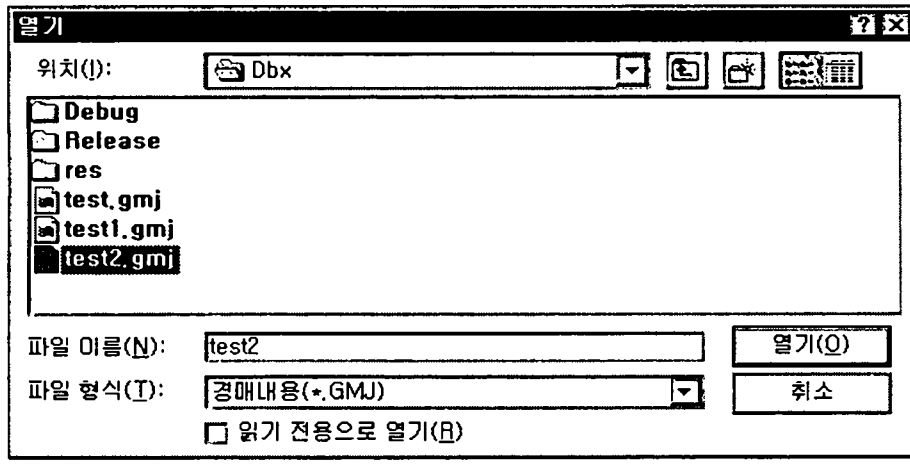


그림 28 경매내용 화일읽기

만약 불러오기를 취소하고자 할 경우에는 "취소" 버튼을 누른다. 화일이 다른 곳에 있을 경우 위치를 바꾸어가며 화일의 위치로 경로를 설정한다.

2) 경매 내용 입력 및 조회

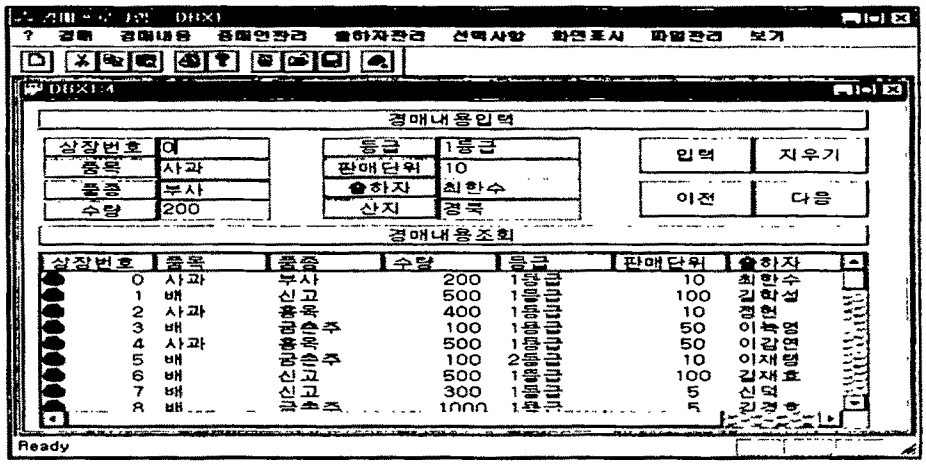


그림 29 경매내용 입력 및 조회

경매할 상품들에 대한 자료 입력과 입력된 자료를 조회할 수 있다. 각각의 상장번호에 대한 품목, 품종, 수량, 등급, 판매단위, 출하자, 산지 등의 정보를 입력하고 "입력" 버튼을 누르면 그림 29에서 볼 수 있듯이 경매내용 조회란에 기록된다. 다음 품목에 대한 내용의 입력 및 수정을 위해 "다음" 버튼을 누른다. 이전 내용에 대한 수정 및 조회를 할 경우는 "이전" 버튼을 누르며, 내용을 지울 경우는 "지우기" 버튼을 누른다. 조회에서는 원하는 상장번호를 선택한 후 클릭하면 그 상장번호에 대한 자료가 경매내용 입력 화면에 나타난다.

3) 경매내용 - 화일저장

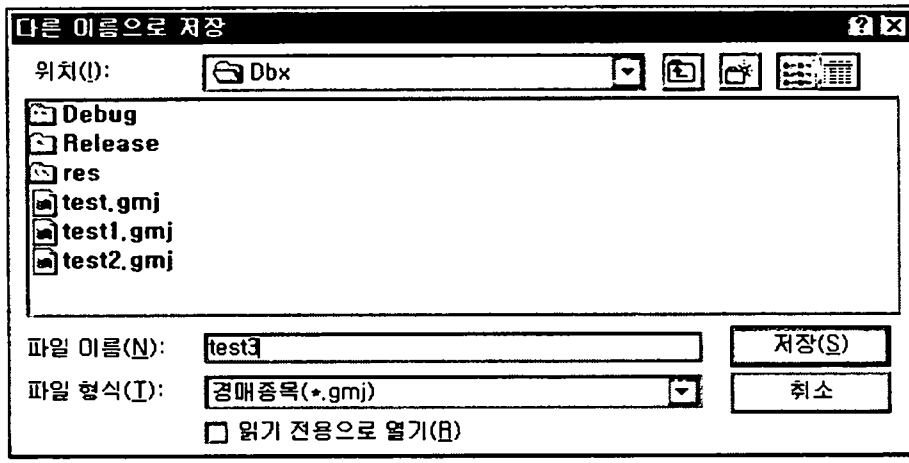


그림 30 경매내용 저장

경매 내용 입력 및 조회에서 작성된 자료는 필요시에 사용할 수 있도록 화일로 저장한다. 주메뉴 "경매내용"의 서브메뉴인 "화일저장"을 선택하면 그림 30과 같은 화면이 나타나며 작성된 자료는 화일로 저장된다.

다. 경매진행 화면

경매내용 입력 및 조회에서 작성한 경매 품목에 대한 자료가 이 화면에 나타나며 이 정보를 이용하여 본 화면에서는 경매를 진행한다. 먼저 GM Load(F6)를 눌러서 경매할 품목을 로드한다. 로드된 자료는 그림 31과 같이 표시된다. 또한 각종 측정장치에 의해 당도, 산도, 수분, 중량 등 품목에 대한 내적 정보를 읽어 표시하게 된다. 그리고 중매인들은 품목에 대한 데이터를 파악한 후 경매에 참여하게 된다. 경매는 시작(F1) 버튼에 의해 시작되며, 소요시간이나 접속률 등은 해당난에 막대 그래프로 나타난다. 입력된 응찰가는 순위에 따라 각각의 관리코드에 기록된다. 그 중 최고가는 최고가란에 나타내진다.

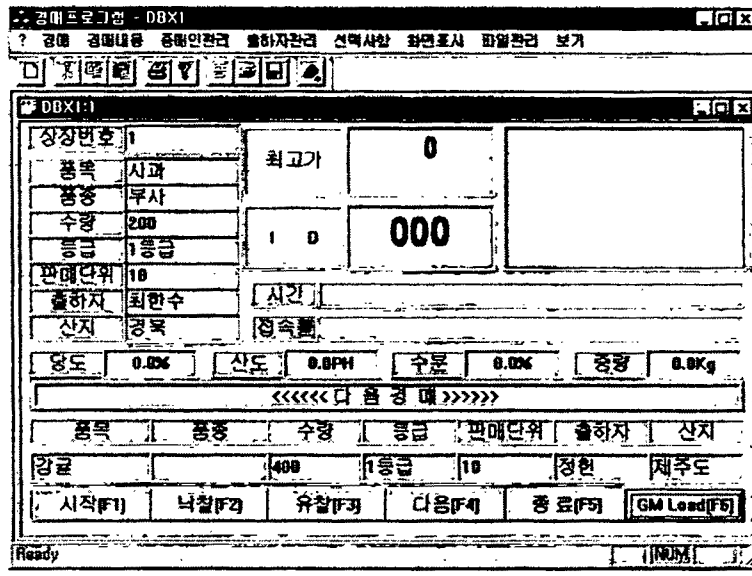


그림 31 GM Load된 main 화면

경매진행중 낙찰(F2), 유찰(F3) 버튼을 사용하며, 다음 품목에 대한 경매를 위해 다음(F4) 버튼을 누른다.

라. 중매인관리 화면

1) 화일 읽기

이미 입력된 중매인 자료 화일을 읽어오는데 사용한다. 그림 32에서 볼 수 있는 바와 같이 중매인관리 화일의 확장자는 .GMI이다. 주메뉴 "중매인관리 화면"에서 서브메뉴 "화일읽기"를 실행하면 그림 32와 같은 내용이 화면에 나타난다. 만약 불러오기를 취소할 경우는 "취소" 버튼을 누른다. 화일이 다른 곳에 있을 경우 위치를 바꾸어가며 화일의 위치로 경로를 설정한다.

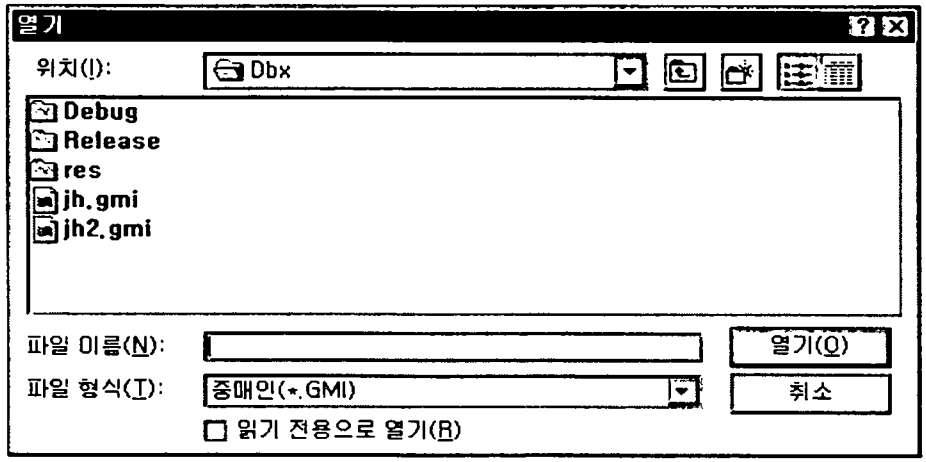


그림 32 중매인자료 읽기

2) 중매인 입력 및 조회

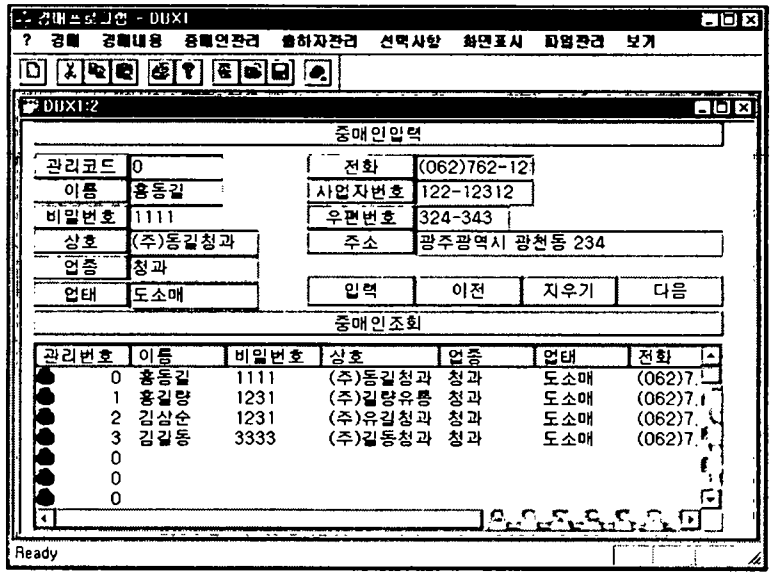


그림 33 중매인 입력 및 조회

본 화면은 중매인에 대한 데이터를 입력하고 조회하기 위한 화면이다. 중매인들의 데이터는 이 화면에 등록해야 한다. 그리고 등록된 중매인들은 각각 관리코드와 비밀번호를 부여받게 된다.

각각의 관리번호에 이름, 비밀번호, 상호, 업종, 업태, 전화, 사업자번호, 우편번호, 주소 등과 같은 정보를 입력하고 "입력" 버튼을 누르면 아래 중매인조회 난에 입력한 내용이 나타난다. 다음 중매인에 대한 정보의 입력 또는 조회를 위해 "다음" 버튼을 누른다. 이전 내용에 대한 수정 및 조회를 할 경우는 "이전" 버튼을 누른다. 내용을 지울 경우에는 "지우기" 버튼을 누른다. 또한 특정 관리번호에 대한 중매인 관리내용을 수정할 경우 중매인 조회 난에서 원하는 관리번호를 마우스로 클릭하면 그 중매인에 대한 자료가 중매인 입력 난에 나타나게 되고 수정하고자 하는 부분을 지우고 재 입력한다.

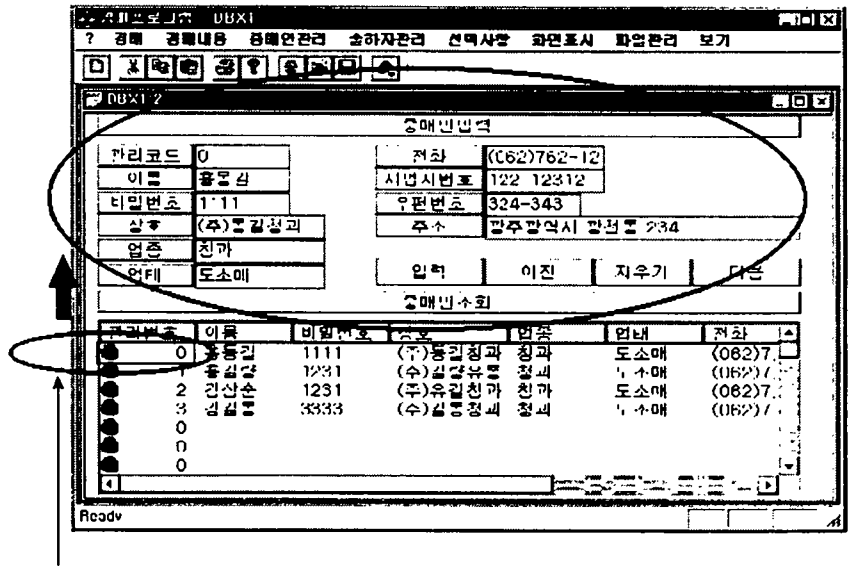


그림 34 중매인자료 로딩방법

3) 화일저장

중매인입력 및 조회에서 작성된 자료는 다음 필요시에 사용할 수 있도록 화일로 저장한다. 주메뉴 "중매인 관리"에서 서브메뉴 "화일저장"을 선택하면 그림 35와 같은 화면이 나타난다. 작성된 자료는 확장자 .GMI로 저장된다.

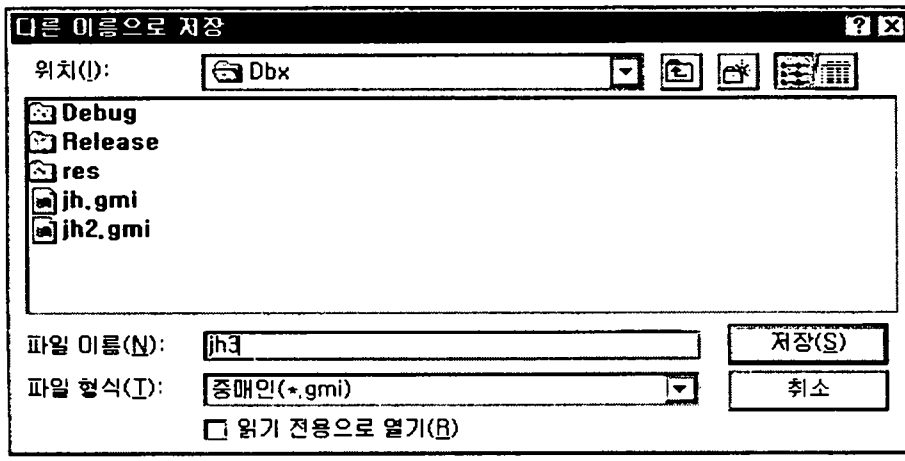


그림 35 중매인화일 저장

마. 출하자관리 화면

1) 화일 읽기

이미 입력된 출하자 자료 화일을 읽어오는데 사용한다. 그림 36에서 볼 수 있는 바와 같이 출하자관리 화일의 확장자는 .CHJ이다. 주메뉴 "출하자관리 화면"에서 서브메뉴" 화일읽기"를 실행하면 그림 36과 같은 내용이 화면에 표시된다. 만약 불러오기를 취소하고 싶을 경우는 "취소" 버튼을 누른다. 화일이 다른 곳에 있을 경우 위치를 바꾸어가며 화일의 위치로 경로를 설정한다.

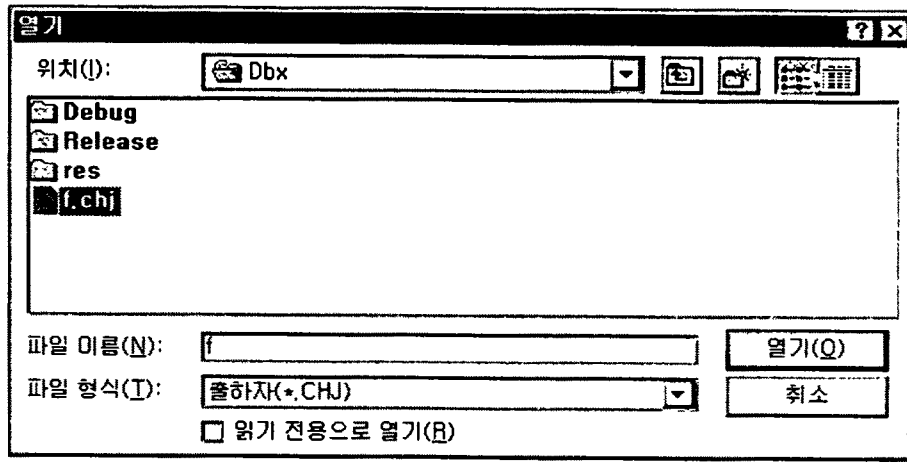


그림 36 출하자화일 읽기

2) 출하자 입력 및 조회

출하자에 대한 정보를 관리하므로써 데이터로 활용할 수 있게된다. 출하자 또한 관리코드로 관리되며 온라인 송금기능의 확장에 대비하여 계좌번호 항목도 설정하였다.

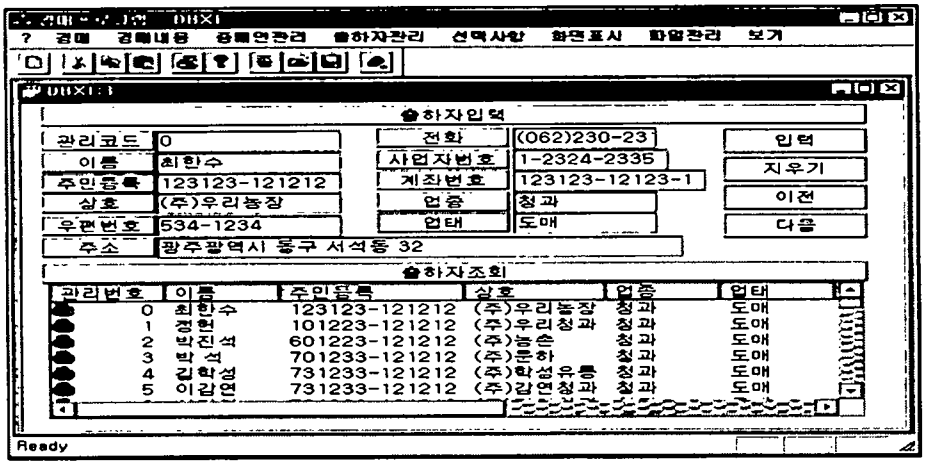


그림 37 출하자 입력 및 조회

3) 파일저장

주메뉴 "출하자 관리"에서 서브메뉴 "화일저장"을 선택하면 그림 38과 같은 화면이 나타나며 작성된 자료는 확장자 .CHJ로 저장된다.

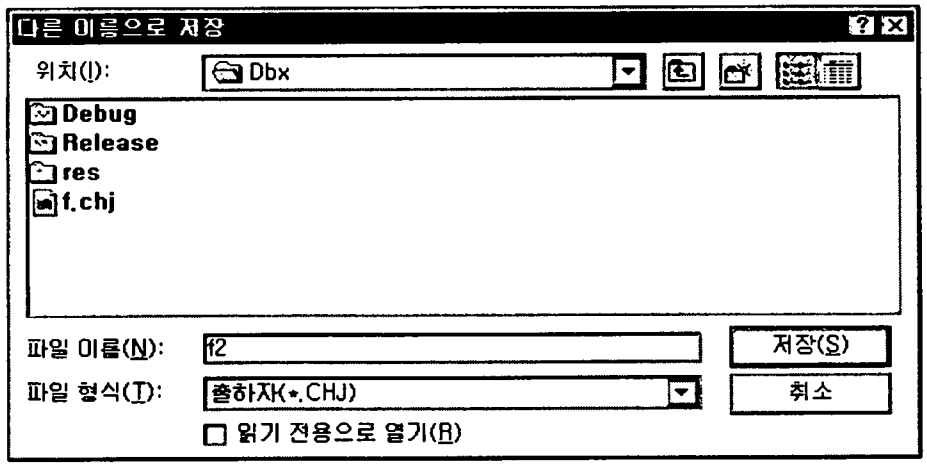


그림 38 출하자화일 저장

바. 화면 이용 방법

1) 메인화면 이용 방법

그림 39에서 볼 수 있는 것과 같이 메인화면의 이용은 팝업 메뉴를 사용하였다. 팝업 메뉴를 마우스로 클릭하면 서브메뉴가 나타나고 다른 곳을 클릭하면 메뉴가 사라지는 방식이다. "경매내용"은 메인 팝업 메뉴이고 그 아래 "경매내용 입력 및 조회", "화일읽기", "화일저장"의 서브메뉴를 볼 수 있다. 마우스가 아닌 키보드를 사용할 경우는 F10과 ←, →키를 사용하여 메뉴를 설정하고 ↑,↓키를 사용하여 서브메뉴를 설정한 후 ENTER키로 메뉴를 선택한다.

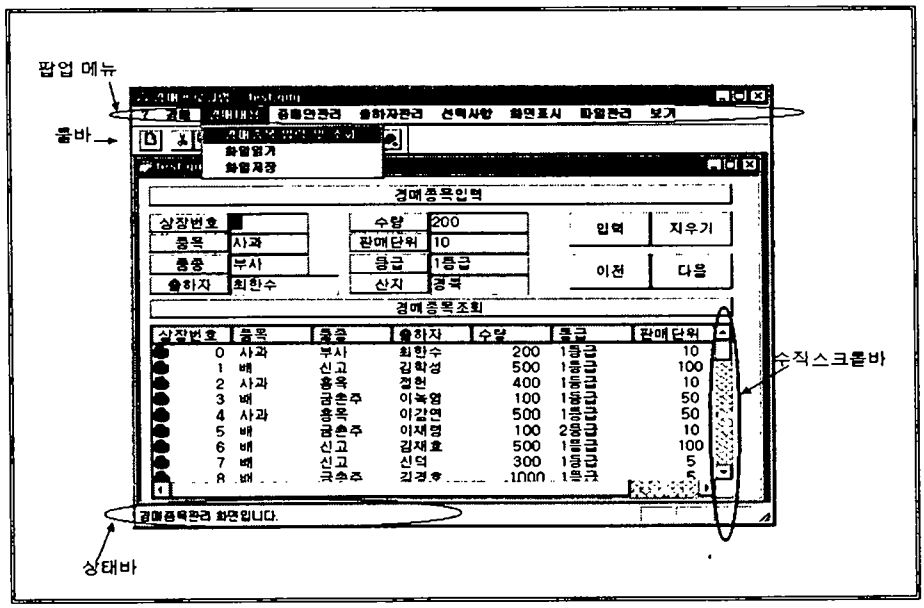


그림 39 메인화면 이용방법

2) 화면 크기조정

그림 39의 오른쪽 상단에 그림 40과 같은 아이콘들을 볼 수 있다. 아이콘

은 작업 표시줄로 이동, 화면크기 변경, 프로그램 종료의 기능을 갖는다. 마우스를 한번 클릭하면 실행된다.

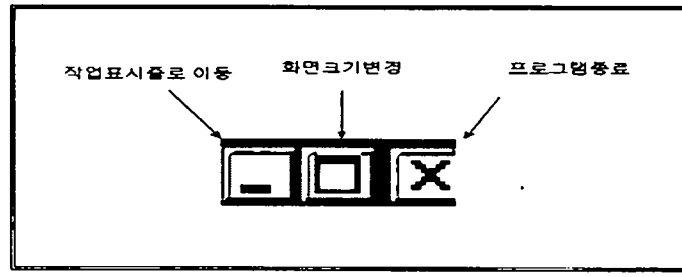


그림 40 화면 크기조정 ICON

3) 툴바

메뉴선택시 메뉴항목에서 선택할 필요없이 툴바를 이용하면 원하는 항목을 간단히 선택할 수 있다. 예를 들어 중매인 관리메뉴를 선택하고자 할 때에는 오른쪽 마지막 아이콘을 마우스로 클릭하면 간단하게 중매인 관리 화면으로 이동시킬 수 있다.

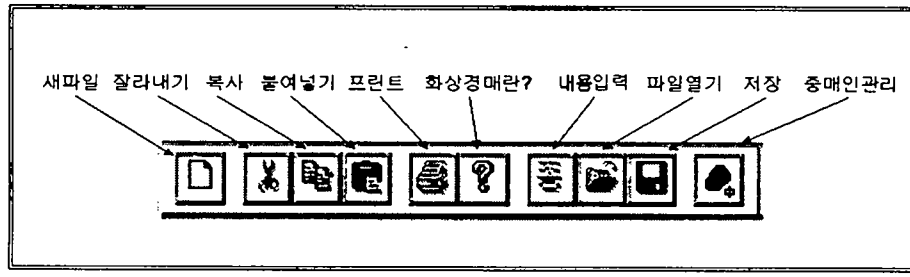


그림 41 툴바 설명

4) 수직 스크롤바, 수평 스크롤바

이것은 한 화면에 전체내용을 나타낼 수 없을 경우 숨김기법을 사용하여 일부만 표현하고 나머지 부분은 좌우 버튼을 통해 내용을 볼 수 있는 방법이다. 마우스로 △▽◁▷키를 눌러 원하는 항목을 찾을 수 있다.

5) 데이터 입력

데이터 입력의 경우 마우스의 포인터를 에디트 박스에 위치한 후 클릭하면 아래 그림과 같이 포인터(커서)의 모양이 바뀌게 된다. 여기에서 키보드를 사용하여 입력한 후 마우스 또는 탭 키를 이용하여 다음 항목으로 이동시킨다.

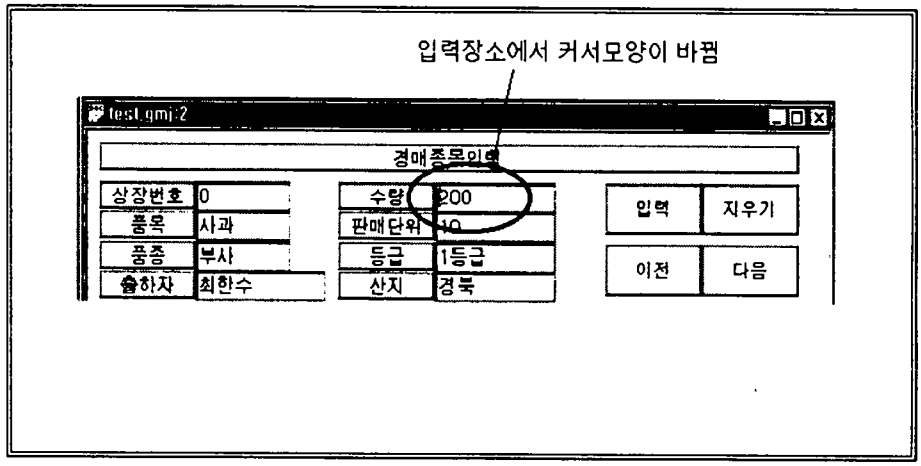


그림 42 자료입력 방법

6. 화면구성

경매시스템의 화면구성은 단말기의 LCD 화면구성과 컴퓨터의 모니터 및 TV수상기의 화면구성으로 구분할 수 있다.

단말기는 중매인이 사용하는 단말장치로서 응찰가를 입력하는 기능외에도 LCD 화면을 통해 경매정보를 중매인에게 제공하여 경매를 원활이 진행할 수 있도록 하였다. 단말기 LCD의 화면에는 한글표현이 10×8, 영문표현이 20×8, 숫자는 20×8의 화면 표현이 가능하며 단말기의 화면에서는 조합형 한글이 사용되었다.

컴퓨터 모니터 및 TV수상기의 화면은 윈도우 95 환경에서 사용하는 visual C++로 프로그램을 작성하였으며, 경매 품목에 대한 정보와 경매 품목의 동영상 그리고 경매사가 경매를 진행하고 통제에 필요한 기능 key들을 나타낼 수 있도록 구성하였다. 또 경매 소요 시간과 해당 품목에 대한 관심도를 파악할 수 있도록 접속율을 나타내는 화면을 구성하였다. 단말기 LCD 화면구성, 컴퓨터 모니터, TV 수상기의 화면 구성은 다음과 같다.

가. 단말기 LCD 화면 구성

단말기의 동작과정은 process라는 변수에 의해 메시지 루프 방식으로 모든 프로그램을 처리하였다. 변수 process의 값은 1에서 9까지 변화되며, 순차적으로 동작을 제어한다. 아래에 process과정에 따른 단말기의 화면 변화 및 동작 내용에 대해 요약하였다.

1) Process 1

전원을 on하면 그림 43과 같은 로고가 표현된다. 로고는 잠시 정지후 다음 과정 수행을 위해 process변수 값이 2로 변환된다.

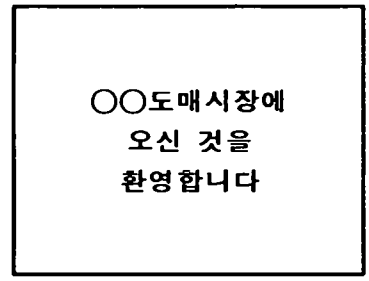


그림 43 단말기 화면 1

2) Process 2

Process의 값이 2가 되면 그림 44와 같은 화면이 출력된다. 화면 출력과 함께 중매인으로 부터 ID와 password를 입력 받는다. 입력 받은 ID와 password는 컴퓨터에 저장되어 있는 중매인의 신상정보와 비교하여 경매 참여 여부를 결정한다. 중매인이 입력한 ID와 password는 단말기에서 패킷 형태로 컴퓨터에 전송된다.

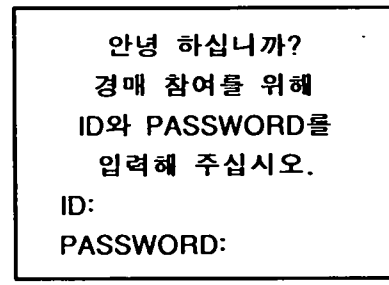


그림 44 단말기 화면 2

단말기에서 컴퓨터에 전송되는 패킷의 형태는 다음과 같다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
102	속성을 나타낸다.(ID확인=102)
006	데이터의 크기를 나타낸다.
001923	실제의 DATA를 나타낸다. (ID=001, PASSWORD=923)
2	데이터의 끝을 나타낸다.

3) Process 3

입력한 ID와 password가 PC에 저장되어 있는 중매인 신상정보와 비교하여 ID와 password가 틀리면 그림 45와 같은 화면을 나타내며 재입력을 요구한다. 틀리면 계속해서 ID와 password의 입력을 요구 받으며, ID와 password가 확인되지 않은 중매인은 경매에 참여할 수 없다. ID와 password가 확인 되면 process의 값이 4로 변환된다.

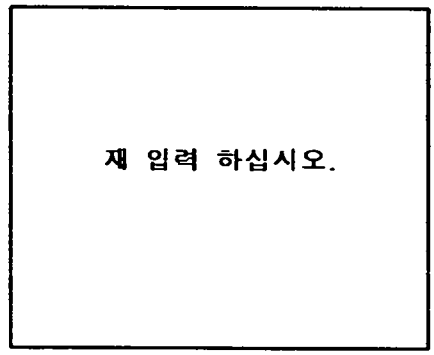


그림 45 단말기 화면 3

4) Process 4

Process 4의 화면이 컴퓨터로 부터 단말기에 전송되어 경매정보(상장번호, 품목, 품종, 수량, 등급, 판매단위, 출하자, 산지)가 그림 46과 같이 나타난다.

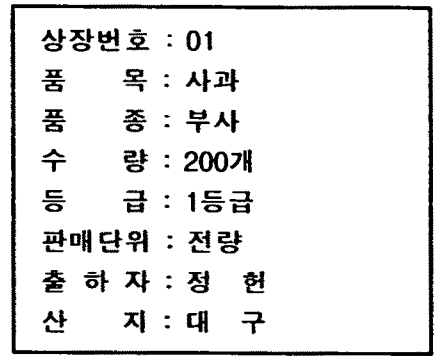


그림 46 단말기 화면 4

경매정보는 패킷형태의 데이터로 되어 있다. 경매정보는 패킷으로 만들어져 컴퓨터--> MIU--> SIU--> KEYPAD로 보내진다. (예 패킷 =1123101039100120000101O정헌 대구2) 보내진 패킷은 다음 표와 같이 나누어 경매정보로 이용된다. 컴퓨터의 경매진행 화면에서 '시작' 버튼이 클릭되면 process의 값이 5로 변환된다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
101	속성을 나타낸다. (경매정보,상장번호=101)
039	데이터의 크기를 나타낸다.
1	마지막 경매 검사 (예 1=마지막 경매)
001	상장번호를 나타낸다. (예 001)
200	수량을 나타낸다. (예 200=200개)
001	등급을 나타낸다. (예 001=1등급)
010	판매단위를 나타낸다. (예 010 = 10개)
name[15]	출하자를 나타낸다. (예 name=정헌)
origin[15]	산지를 나타낸다. (예 origin=대구)
2	데이터의 끝

5) Process 5

컴퓨터에서 '시작' 버튼을 클릭하면 경매가 시작된다.

(ID와 password가 확인되지 않은 중매인은 경매에 참여하지 못한다.)

경매시작은 다음 표와 같은 패킷으로 보내진다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
900	속성을 나타낸다.(경매시작=900)
000	데이터의 크기를 나타낸다.
2	데이터의 끝을 나타낸다.

경매가 시작되면 단말기의 화면에는 그림 47과 같이 출력되며 중매인이 응찰가를 입력하면 응찰가 중 가장 높은값이 '최고가'난에 나타나며 최고가는 계속 갱신된다. 응찰가는 최저와 최고 범위내의 값만 입력되며 범위 밖의 응찰가는 처리되지 않는다.

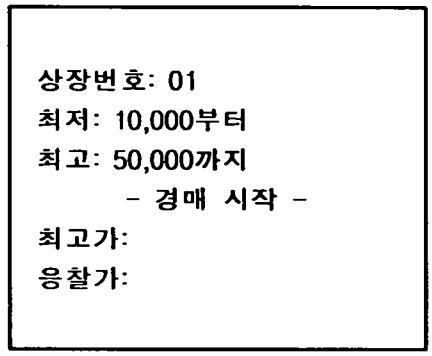


그림 47 단말기 화면 5

가격요구 패킷이 전송되면 단말기에서 다음 표와 같은 패킷 형태로 컴퓨터에 전송된다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
101	속성을 나타낸다.(최고가=101)
000	데이터의 크기를 나타낸다.
000	실제의 데이터를 나타낸다.
2	데이터의 끝을 나타낸다.

6) Process 6

경매사가 컴퓨터에서 '낙찰' 버튼을 클릭하면 단말기에 경매 끝의 패킷을 보낸다. 전송되는 패킷의 형태는 다음 표와 같다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
900	속성을 나타낸다.(경매끝=900)
2	데이터의 끝을 나타낸다.

위의 패킷이 단말기에 전송되면 중매인들의 단말기에 낙찰가와 낙찰자가 그림 48과 같이 표시되어 낙찰을 알리고 한 품목에 대한 경매가 끝난다. 낙찰이 결정되면 낙찰자의 ID 및 password와 함께 품목, 품종, 낙찰가가 컴퓨터에 저장된다.

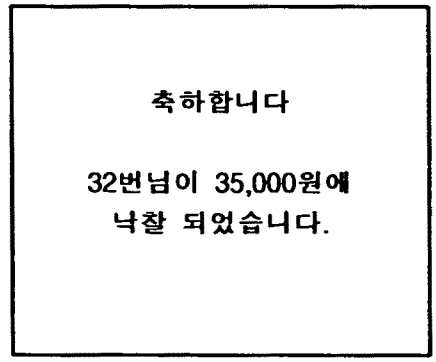


그림 48 단말기 화면 6

7) Process 7

경매사가 컴퓨터에서 '유찰' 버튼을 클릭하면 단말기에 경매 끝의 패킷을 보낸다. 전송되는 패킷의 형태는 다음과 같다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
904	속성을 나타낸다.(유찰=904)
2	데이터의 끝을 나타낸다.

위의 패킷이 단말기에 전송되면 그림 49와 같은 유찰을 알리는 메시지가 나타나게 된다.

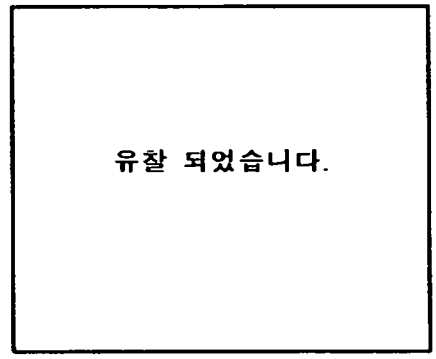


그림 49 단말기 화면 7

8) Process 8

'다음' 버튼이 눌러지면 process의 값이 4로 변화하고 process 4 부터 6 또는 7 까지의 과정이 반복된다.

9) Process 9

경매진행 화면의 '종료' 버튼을 클릭하면 경매를 종료한다. 종료 메시지는 컴퓨터에서 다음 표와 같은 패킷으로 단말기에 전송된다.

패킷의 데이터	패킷의 내용
1	데이터의 시작
123	id를 나타낸다.(miu:1 siu:2 keypad:3)
901	속성(경매 종료=901)
1	마지막 경매 검사(예 1=마지막 경매)
2	데이터의 끝

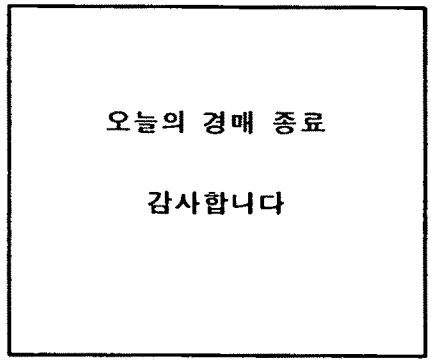


그림 50 단말기 화면 8

나. 컴퓨터 모니터 화면구성 및 TV수상기 화면구성

컴퓨터 모니터와 TV수상기 화면은 동일하게 구성하였으며 그 내용은 다음과 같다.

1) 경매프로그램 실행 (DBX.EXE)

경매프로그램(DBX.EXE)을 실행하면 초기화면이 나타난다. 그림 51의 화면은 경매프로그램의 초기화면으로 toolbar의 첫 번째 아이콘을 클릭하거나 메뉴 file에 있는 new를 클릭하면 경매프로그램으로 들어가게 된다.

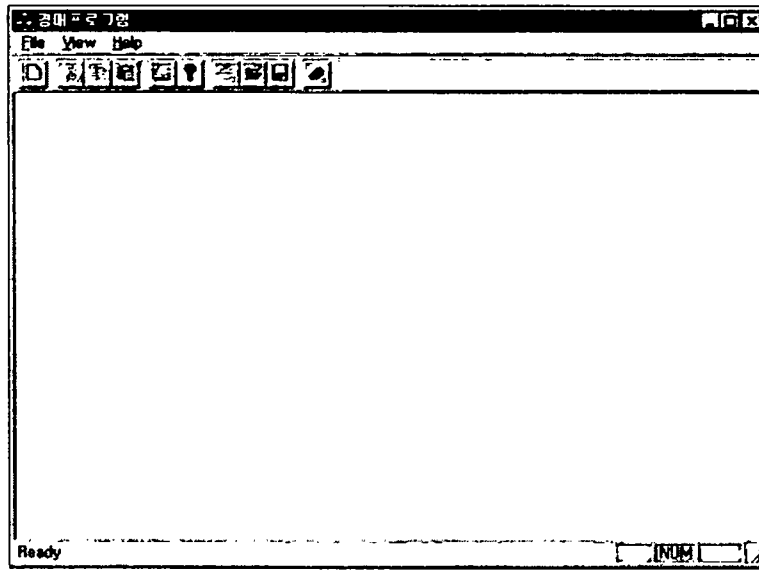
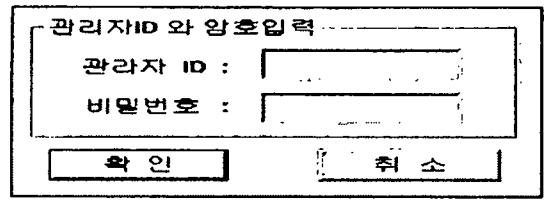


그림 51 경매프로그램 초기화면

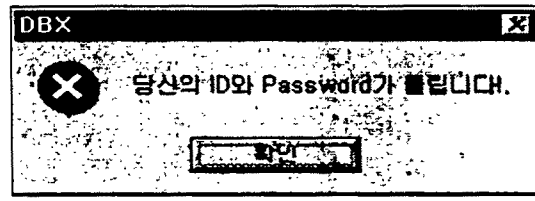
2) 경매시작

위에서 toolbar 또는 메뉴 file의 new를 클릭하면 관리자(경매사) ID와 password를 묻는 dialog box가 실행되게 된다. 다음 그림의 box는 경매프로그램에 들어가기에 앞서 경매프로그램을 관리하는 관리자의 ID와 password를 확인하는 단계로 ID와 password가 맞을 경우 다음 화면으로 넘어가게 된다.



3) ID 확인 및 처리

만약 ID와 password가 틀리면 다음 그림과 같은 message box가 화면상에 표현되고, 3번 틀릴 경우에는 프로그램이 종료되도록 구성하였다.



4) 경매내용 및 조회

관리자(경매사)의 ID와 password가 맞으면 그림 52의 '경매내용 및 조회' 화면으로 넘어가게 된다. 그림 52의 화면은 금일 경매의 내용을 포함하고 있는 file들로 원하는 파일을 선택하여 열기를 실행하면 경매프로그램에 경매내용들이 load된다. 경매내용 파일들은 확장자가 gmj로 정의되어 있다.

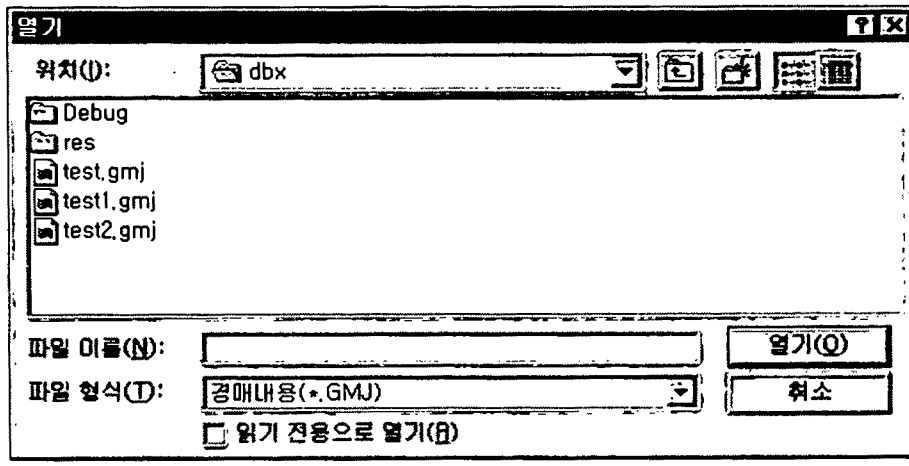


그림 52 경매내용 및 조회화면

5) 경매진행 화면

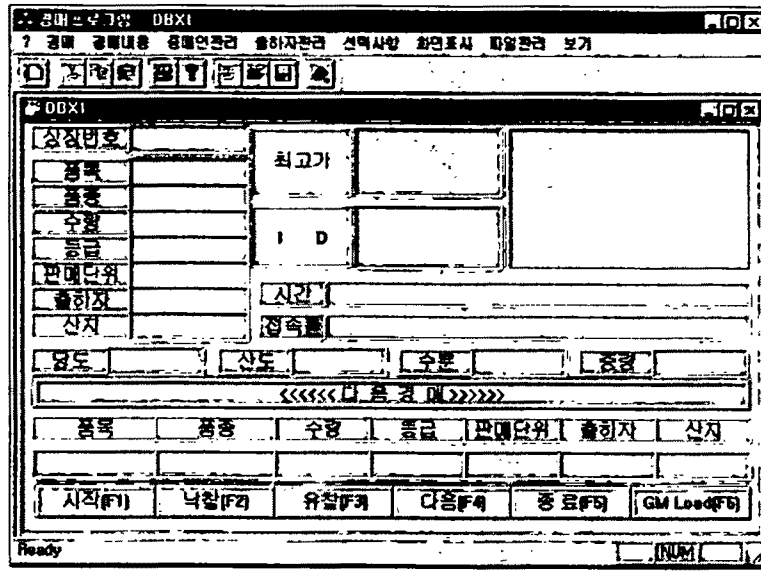
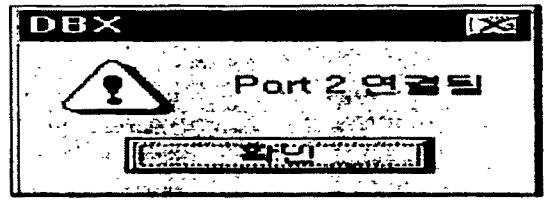


그림 53 경매진행 초기화면

그림 53은 경매진행 화면의 초기화면으로, 경매에 대한 다양한 정보를 나타낼 수 있다. 경매품목의 수량, 등급, 출하자, 산지 등과 같은 정보와 당도, 산도, 수분, 중량과 같은 내적 정보 그리고 영상정보를 제공할 수 있도록 화면을 구성하였다. 이 화면은 경매진행을 위한 정보를 제공하는 기능과 경매를 진행하고 통제할 수 있는 기능을 갖는 화면이다.

6) GM Load

경매진행 화면 중 우측 하단의 버튼 GM Load(F6)를 선택하면 다음 그림과 같은 메시지가 나타난다. 이 메시지는 컴퓨터와 단말기간, 컴퓨터와 측정장치간에 통신이 가능하도록 셋팅되어 초기화면에 각종 정보들을 load한다.



윗 그림에서 message box의 '확인' 버튼을 클릭하면 그림 54와 같은 화면이 나타난다. 이 화면은 상품의 외관을 보여주기 위한 화면으로서 받침대가 회전함에 따라 외관을 고루 보여주기 위해 동영상으로 표현되며 수 초가 경과한 후 경매진행을 위해 경매진행 화면인 그림 55가 나타난다. 이 화면에는 경매를 위한 데이터를 load하고 화상을 뜨게한다. 왼쪽에는 경매 품목에 대한 정보가 나타나 있고 영상의 아래 부분은 중매인들의 단말기 접속(응찰)율과 경매소요 시간 등을 보여 주며, 그 아래 부분에는 당도, 산도, 수분, 중량의 데이터가 표시 되어진다. 중매인들이 단말기를 통해 응찰가를 입력하면 최고가가 갱신되어 나타나며 최고가를 입력한 중매인의 ID가 표시된다. '다음경매' 난에는 다음에 경매할 품목에 대한 정보를 나타내준다.

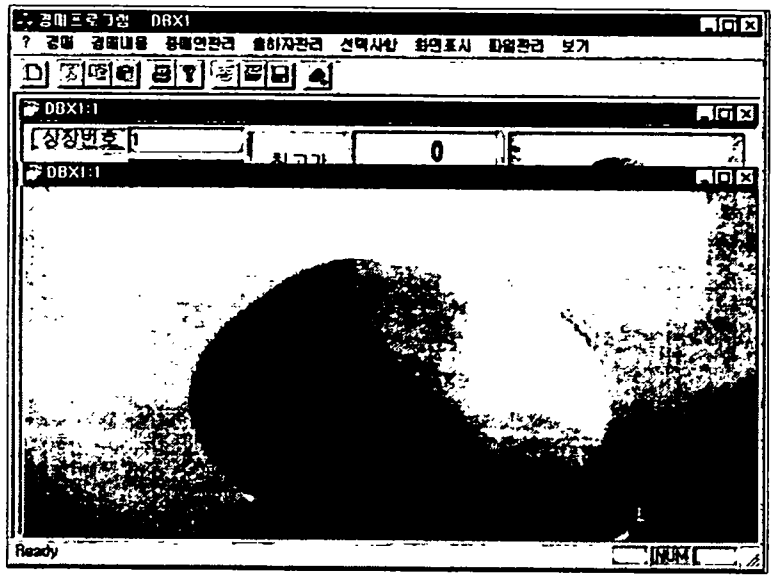


그림 54 화상 화면

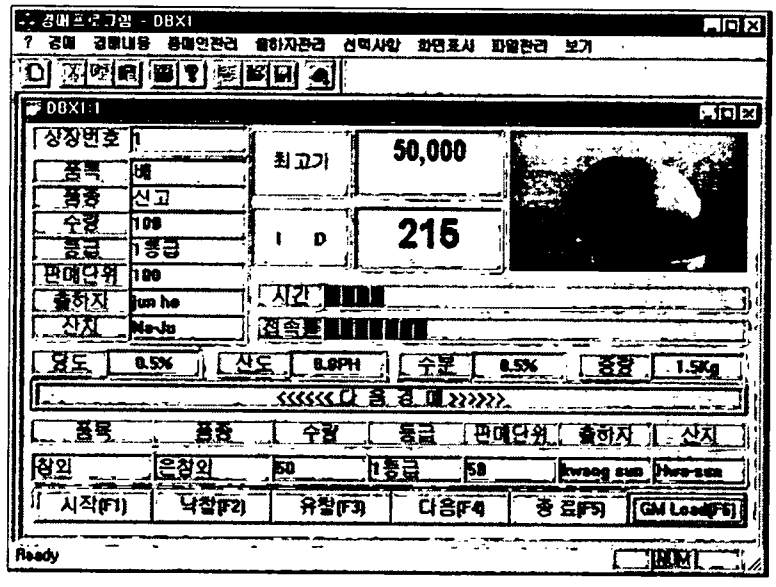


그림 55 경매진행 화면

7) 접속현황

경매 프로그램의 메뉴 중에 옵션항목의 접속현황을 클릭할 경우 그림 56과 같이 현재 경매에 참여하고 있는 중매인과 단말기(keypad)의 고유 ID를 보여 주어 현재 경매에 참여하고 있는 중매인의 상황을 파악할 수 있도록 하였다.

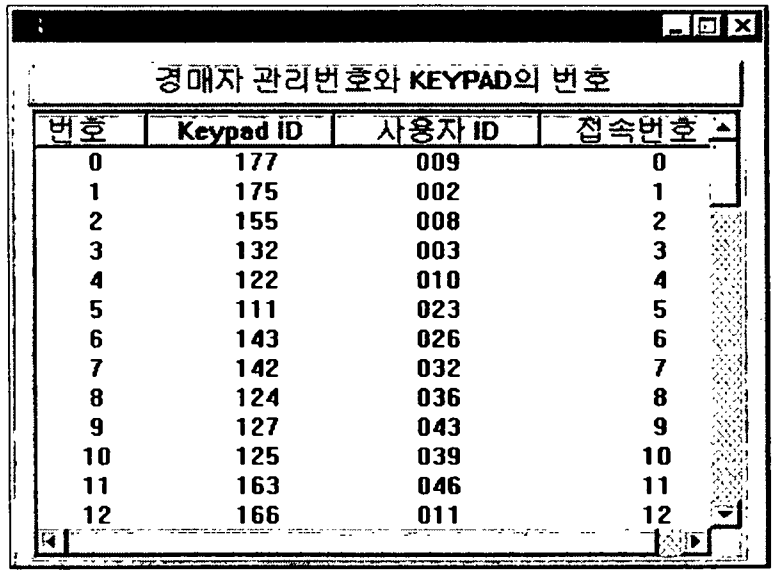


그림 56 중매인 상황화면

제 3 장 연구개발 결과 및 활용에 대한 건의

제1절 연구개발 결과

1. 단말기, 단말기-컴퓨터간 인터페이스 개발결과

국내 몇 곳의 도매시장에서 전자식 경매시스템을 도입하여 사용하고 있다. 기존의 것과 본 시스템은 여러가지 면에서 비교될 수 있으나 단말기(keypad)와 단말기-컴퓨터간 인터페이스 부분에서의 차이점은 크게 다음과 같이 대별할 수 있다.

가. 기존의 단말기는 중매인이 입력한 응찰가만을 즉, 한 줄의 숫자 정보만을 단말기의 표시판에 나타낼 수 있도록 되어있다.

나. 단말기에서 컴퓨터로 중매인의 번호와 응찰가를 보내기만 하고 컴퓨터에 수록된 정보를 단말기로 전송받지 못하는 일방향 통신만 할 수 있다.

반면, 본 시스템은 숫자와 문자정보를 나타낼 수 있는 그래픽 LCD(Liquid Cristal Display, 표시용량 - 숫자: 20×16, 한글: 10×8, 영문: 20×16)를 단말기의 표시판으로 사용하고 양방향 통신이 가능하도록 설계하므로써 다음과 같은 장점을 갖는다.

가. 단말기와 컴퓨터간 인터페이스 장치를 양방향 통신이 가능하도록 설계하여, 대화식으로 중매인의 고유번호, 비밀번호, 응찰가를 입력할 수 있도록 구성 하므로써 사용자의 편의를 도모하였다.

나. 경매될 품목 중 중매인은 구매할 품목에 대한 사양을 알고자한다. 경매를 시작하기 전에 경매할 상품들에 대한 사양 즉 품목, 품종, 산지, 출하자, 수량, 등급 등과 같은 정보를 경매사는 미리 컴퓨터에 입력시키고 이러한 정보들을 단말기의 LCD에 나타낼 수 있게 하므로써 경매가 시작되기 전에 구매하고자 하는 품목과, 응찰가 등을 결정하기 위한 정보를 중매인에게 제공할 수 있다.

다. 낙찰·유찰의 유무, 낙찰가, 낙찰자 등과 같은 전광판을 보고 알 수 있는 정보를 컴퓨터로 부터 전송받아 단말기의 LCD에 나타나게 하므로써 단말기만 으로도 경매진행 상황 파악이 가능하다.

라. 수지식 경매방법에서 전자식 경매방법으로 전환하기 위해 도매시장은 상당한 시설비 부담을 안게 된다. 본 연구에서 개발한 단말기는 정보를 단말기에서 컴퓨터로 전송하는 기능과 컴퓨터에 수록된 정보를 단말기로 전송 받을 수 있는 기능을 갖고 있으며, 표시판으로 숫자와 문자정보를 나타낼 수 있는 LCD를 사용하므로 전광판이 없어도 경매를 진행할 수 있어서 저렴한 가격으로 전자식 경매시스템을 설치할 수 있다.

마. 바퀴가 부착된 구조물 위에 컴퓨터(노트북 PC)와 단말기들을 탑제하여 도매시장 내에 경매할 품목이 놓인 곳으로 구조물을 이동하거나 농산물이 놓여있는 장소로 경매사는 노트북 PC를, 중매인들은 단말기를 지니고 다니면서 설치된 접속 단자에 컴퓨터와 단말기를 접속시킴으로써 경매가 가능하기 때문에 고정식은 물론이고 단위 품목당 중매인의 수가 많지않은 중소규모의 도매시장에 이동식 전자 경매시스템으로 사용할 수 있는 부수적 효과를 얻을 수 있다.

2. TV수상기-컴퓨터간 인터페이스 구성결과

국내에서 사용 중에 있는 기존의 전자식 시스템은 화면으로 전광판을 사용하고 있는데 전광판은 고가이면서도 나타낼 수 있는 정보량이 적고 정밀한 화상을 나타낼 수 없다. 따라서 화면으로 TV수상기를 사용하므로써 다음과 같은 장점을 갖는다.

가. 화상시스템은 농산물의 질을 평가할 수 있는 외적인 정보를 비디오카메라를 통해 제공하고 화면은 전광판보다 저렴한 TV수상기를 사용하기 때문에 많은 양의 문자 및 숫자정보와 고화질의 영상정보를 제공하면서도 설치비용이 저렴하다.

나. 도매시장의 규모에 따라 TV수상기의 설치대수만 조정하면 되므로 소규모 도매시장에서 대규모 도매시장에 이르기까지 모두 적용할 수 있다.

3. 측정장치-컴퓨터간 인터페이스 구성결과

가. 각종 측정장치에 의해 농산물의 내용 즉 당도, 산도, 수분, 무게 등과 같은 내적 정보를 제공하므로써 중매인에게 상품의 질을 판정할 수 있는 기회를 부여한다.

나. 농산물의 품질 확인으로 등급 판정에 대한 객관성을 확보할 수 있다.

4. 경매프로그램 개발결과

메뉴화면, 경매내용 입력화면, 경매진행화면 등과 같은 경매프로그램을 개발하여 경매진행에 대한 효율을 높일 수 있었으며, 중매인 관리화면, 출

하자 관리화면을 통해 중매인과 출하자의 성명, 고유번호, 상호, 주소, 전화번호, 주민등록번호 등을 입력하여 중매인과 출하자를 관리할 수 있으며, 경매결과를 전산처리하기 위한 자료를 제공할 수 있다.

5. 화면구성 결과

가. 단말기의 LCD화면 내용을 대화식으로 구성하여 중매인들에게 편의를 제공하였다.

나. 상장번호와 상품에 대한 품목, 품종, 수량, 등급, 출하자, 산지 등을 단말기의 화면에 나타내줌으로써 단말기를 통해서도 상품의 정보를 알 수 있게 하였다.

다. 단말기의 화면에 자신이 입력한 응찰가와 최고응찰가를 나타내고, 경매결과에 대한 정보인 낙찰가와 낙찰자 등을 나타내줌으로써 단말기만을 주시하여도 경매진행이 가능하도록 하였다.

라. 경매사가 사용하는 컴퓨터의 모니터(그림 55)에는 상품정보(상장번호 품목, 품종, 수량, 등급, 판매단위, 출하자, 산지)와 내적정보(당도, 산도, 수분, 중량)를 수록하여 경매진행의 주최인 경매사에게 다양한 상품정보를 제공하였다.

마. 컴퓨터의 모니터에는 또 다음 경매품목에 대한 사양, 경매경과 시간, 해당 품목에 대해 응찰한 중매인들의 수를 파악할 수 있는 접속률, 중매인들이 입력한 응찰가 중 최고가, 최고가를 입력한 중매인의 고유번호(ID)를 나타내고 경매진행을 위해 경매사가 시작, 낙찰, 유찰, 다음, 종료 항을 마우스로 클릭하거나 컴퓨터의 키보드상에 있는 기능키들 중 F1, F2, F3, F4, F5 키를 선택할 수 있도록 구성하므로써 경매사로 하여금 효율성있는 경매진행이 가능하도록 하였다.

제2절 활용에 대한 건의

1. 개발한 경매시스템의 주요 장치

개발한 화상시스템이 도입된 전자식 경매시스템의 주요 장치와 장치의 역할은 다음과 같다.

가. 단말기(keypad)와 단말기-컴퓨터간 인터페이스

1) 이 장치는 단말기에 LCD를 부착하고 단말기와 컴퓨터간에 양 방향 통신이 가능 하도록 단말기-컴퓨터간 인터페이스 장치를 설계하므로써 컴퓨터에 입력된 경매할 품목에 대한 정보, 중매인이 입력한 자신의 응찰가와 최고 응찰가, 낙찰·유찰 상황, 낙찰가, 낙찰자의 고유번호 등을 단말기의 LCD 화면에 표시할 수 있다.

2) 단말기는 중매인들이 자신의 응찰가를 입력하는 장치이며 본 연구에서 개발한 단말기는 상당량의 숫자와 문자 정보를 수록할 수 있는 LCD를 채용하므로써 상술한 바와 같은 내용들을 LCD 화면에 나타낼 수 있어 단말기만을 주시하여도 경매진행 상황을 파악할 수 있다.

나. 무비카메라 및 인터페이스

농산물의 외관을 촬영하여 TV 수상기와 컴퓨터의 모니터에 나타내줌으로써 농산물의 외적 상태를 확인하는 장치.

다. 당도, 산도, 수분, 중량 측정장치 및 인터페이스

농산물의 내적 상태를 측정하고 측정된 데이터를 인터페이스 장치를 통하여

컴퓨터 모니터와 TV 수상기 화면에 표시할 수 있도록 전송하는 장치.

라. TV 수상기 및 인터페이스

1) 무비카메라로 촬영한 농산물의 외관과 각종 측정장치로 측정한 농산물의 내적정보 그리고 경매 진행과정을 알아볼 수 있도록 하기 위해 최고 응찰가, 낙찰·유찰 상황, 낙찰가, 낙찰자의 고유번호 등을 나타낼 수 있는 화면.

2) 중매인에게는 농산물의 외적 화상을 제공하며, 농민인 출하자에게는 경매 진행과정을 파악할 수 있는 정보를 제공.

마. 컴퓨터 시스템

상술한 장치들을 통합 관리할 수 있는 소프트웨어가 내재된 컴퓨터 시스템.

경매 시작전 경매에 필요한 각종 정보를 입력하고 경매진행을 위해 경매사가 사용.

2. 활용 방안

가. "가, 나, 다, 라, 마" 장치를 모두 사용하는 경우

청과 도매시장의 전자식 경매시스템으로 활용.

나. "가, 나, 라, 마" 장치를 사용하는 경우

내적 정보를 요구하지 않는 청과 도매시장, 야채 도매시장, 화훼 도매시장, 수산 도매시장, 축산 도매시장 등을 비롯한 모든 농축수산물 도매시장의 전자식 경매시스템으로 활용.

다. "가, 라, 마" 장치를 사용하는 경우

1) 농산물의 외관과 농산물의 내적 상태인 당도, 산도, 수분, 중량값 등의 정보를 제공할 필요가 없는 경우의 도매시장에 적용한다.

2) 중매인의 좌석 앞으로 샘플을 이동시키는 화훼 도매시장과 축산 도매시장에 적합.

3) 청과 도매시장과 야채 도매시장 수산 도매시장 등도 샘플을 이동시키면 적용 가능함.

4) 바퀴가 부착된 구조물 위에 단말기, 컴퓨터(노트북 PC), TV수상기를 탑재하고 경매할 품목이 놓여있는 곳으로 구조물을 이동하면서 경매를 수행할 수 있으므로 중소형 도매시장에서 이동식 경매 장치로 사용할 수 있음.

5) 도매시장내에 경매할 장소를 몇 곳 지정해 두고 그 곳에 TV수상기와 단말기, 노트북 PC를 연결할 수 있는 단자를 설치해 두면 경매사는 노트북 PC를 중매인은 단말기를 소지하고 다니면서 단자에 접속하면 경매가 가능하므로 중소형 도매시장에 이동식 경매 장치로 사용할 수 있음.

라. "가, 마" 장치를 사용하는 경우

이동식 경매장치에 TV수상기 설치가 부담스러우면 단말기와 노트북 PC만 으로도 경매가 가능하므로 이 두 장치로 이동식 경매장치를 구성할 수 있다. 단 출하자에게 경매 진행과정을 보여줄 수 없다는 문제점이 있다.

부 록

본 연구의 수행 프로그램 중 일부를 수록함.

thread.cpp => 통신을 위한 thread 처리용 프로그램

PortConfig.cpp => port에 대한 환경 설정 프로그램

PacketDlg.cpp => 패킷 처리용 프로그램

MainFrm.cpp => 메인 프레임 클레스 수행 프로그램

dbxsView.cpp => 화면 내용 표현에 대한 프로그램

CDbxsDoc.cpp => application에 대한 클레스 실행 정의 프로그램

v55.c => V55 제어 프로그램

serial.c => 직렬 통신 프로그램

miu.c => 메인 인터페이스 unit 프로그램

glcd.c => 그래픽 LCD 제어 프로그램

```
//////////////////  thread.cpp ///////////////////
//// 통신을 위한 thread 처리용 프로그램 ////


#include "thread.h"

DWORD CommWatch(LPVOID lParam)
{
  DWORD  dwEvtMask, dwTransfer, dwError;
  OVERLAPPED  os;
  COMSTAT    cs;
  BYTE  abIn[MAXBLOCK + 1];
  int       nLength;
  CComm*  com = (CComm*) lParam;

  memset(&os, 0, sizeof(OVERLAPPED));

  os.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);
  PurgeComm(com->hDev ,PURGE_RXCLEAR);

  if (os.hEvent == NULL) {
    MessageBox(NULL,"이벤트 에러","이벤트를 생성 할수 없습니다."
          ,MB_ICONERROR | MB_OK);
    return FALSE;
  }

  if (!SetCommMask(com->hDev, EV_RXCHAR)) {
    MessageBox(NULL, "ERROR", "SetCommMask 불가능", MB_OK)
    return FALSE;
  }

  while (com->bConnected) {
    dwEvtMask = 0;
    if (!WaitCommEvent(com->hDev, &dwEvtMask, NULL)) {
      if (GetLastError() == ERROR_IO_PENDING) {
        GetOverlappedResult(com->hDev, &os, &dwTransfer,FALSE)
```

```
        }
      } else {
        ClearCommError(com->hDev, &dwError, &cs);
        if (((dwEvtMask & EV_RXCHAR) == EV_RXCHAR) &&
          cs.cbInQue) {
          do {
            memset(abIn, 0, MAXBLOCK);
            nLength = com->ReadCommBlock((LPSTR)abIn, MAXBLOCK)
            if (nLength != 0) {
//            if (Win.p.wndWork != NULL)
            Win.p.wndWork->SendMessage(WM_USER + 1, nLength,
            (LONG)&abIn);
              com->PushToQue((LPSTR)abIn, nLength);
            }
          } while (nLength > 0);
        }
      }
    }

    PurgeComm(com->hDev ,PURGE_RXCLEAR);
    CloseHandle(os.hEvent);
    com->hCommWatchThread = NULL ;

    return TRUE;
}

DWORD PacketWatch(LPVOID lParam)
{
    PACKET p;
    DWORD  dwEvtMask, dwTransfer, dwError;
    OVERLAPPED  os;
    COMSTAT    cs;
    BYTE  abIn[MAXBLOCK + 1];
    int       nLength;
    CComm*     com = (CComm*) lParam;
```

```
  memset(&os, 0, sizeof(OVERLAPPED));

  os.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);
  PurgeComm(com->hDev ,PURGE_RXCLEAR);

  if (os.hEvent == NULL) {
    MessageBox(NULL,"이벤트 에러","이벤트를 생성 할수 없습
          니다.",MB_ICONERROR|MB_OK);
    return FALSE;
  }

  if (!SetCommMask(com->hDev, EV_RXCHAR)) {
    MessageBox(NULL, "ERROR", "SetCommMask 불가능", MB_OK)
    return FALSE;
  }

  while (com->bConnected) {
/*  if (cpoint > 3) {
      if (cpoint >= (int)(*(rxbuf + 2) + 5)) {
        p = com->remake_packet();
        if ((p.data_start == DATASTART) && (p.data_end ==
         DATAEND)
          && (p.data_size == strlen(p.data))) {
          SendMessage(HWND,
        } else {
          return FALSE;
        }
      }
    }

    return BUSY;
  }*/


  /*  dwEvtMask = 0;
    if (!WaitCommEvent(com->hDev, &dwEvtMask, NULL)) {
```

```
      if (GetLastError() == ERROR_IO_PENDING) {
        GetOverlappedResult(com->hDev, &os, &dwTransfer,FALSE);
      }
    } else {
      ClearCommError(com->hDev, &dwError, &cs);
      if (((dwEvtMask & EV_RXCHAR) == EV_RXCHAR) &&
          cs.cbInQue) {
        do {
          memset(abIn, 0, MAXBLOCK);
          nLength = com->ReadCommBlock((LPSTR)abIn, MAXBLOCK);
          if (nLength != 0) {
//          if (Win.p.wndWork != NULL)
          Win.p.wndWork->SendMessage(WM_USER + 1, nLength,
          (LONG)&abIn);
            com->PushToQue((LPSTR)abIn, nLength);
          }
        } while (nLength > 0);
      }
    }
*/}

  PurgeComm(com->hDev ,PURGE_RXCLEAR);
  CloseHandle(os.hEvent);
  com->hPacketWatchThread = NULL ;

  return TRUE;
}
```

```
////////////////// PortConfig.cpp //////////////////
//////// port에 대한 환경 설정 프로그램 ////////


#include "stdafx.h"
#include "dbxs.h"
#include "PortConfig.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPortConfig dialog


CPortConfig::CPortConfig(CWnd* pParent /*=NULL*/)
  : CDialog(CPortConfig::IDD, pParent)
{
  //{{AFX_DATA_INIT(CPortConfig)
  m_dtrdsr = FALSE;
  m_rtscts = FALSE;
  m_xonxoff = FALSE;
  m_baud = 9;
  m_data = 3;
  m_port = 1;
  m_stop = 0;
  m_parity = 0;
  //}}AFX_DATA_INIT
}


void CPortConfig::DoDataExchange(CDataExchange* pDX)
{
```

```
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CPortConfig)
  DDX_Check(pDX, IDC_CHECK_DTRDSR, m_dtrdsr);
  DDX_Check(pDX, IDC_CHECK_RTSCTS, m_rtscts);
  DDX_Check(pDX, IDC_CHECK_XON_XOFF, m_xonxoff);
  DDX_CBIndex(pDX, IDC_COMBO_BAUD, m_baud);
  DDX_CBIndex(pDX, IDC_COMBO_DATA, m_data);
  DDX_CBIndex(pDX, IDC_COMBO_PORT, m_port);
  DDX_CBIndex(pDX, IDC_COMBO_STOP, m_stop);
  DDX_CBIndex(pDX, IDC_COMBO_PARITY, m_parity);
  //}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CPortConfig, CDialog)
  //{{AFX_MSG_MAP(CPortConfig)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPortConfig message handlers

void CPortConfig::OnOK()
{
  UpdateData(TRUE);

  CDialog::OnOK();
}
```

```
////////////////// PacketDlg.cpp //////////////////
////////////// 패킷 처리용 프로그램 //////////////


#include "stdafx.h"
#include "dbxs.h"
#include "PacketDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CPacketDlg dialog

CPacketDlg::CPacketDlg(CWnd* pParent /*=NULL*/)
  : CDialog(CPacketDlg::IDD, pParent)
{
  //{{AFX_DATA_INIT(CPacketDlg)
  m_data = _T("");
  m_data_end = 2;
  m_data_start = 1;
  m_update = _T("");
  m_id = 0;
  m_attr = 0;
  m_size = 0;
  //}}AFX_DATA_INIT
}


void CPacketDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CPacketDlg)
```

```
  DDX_Text(pDX, IDC_EDIT_DATA, m_data);
  DDV_MaxChars(pDX, m_data, 250);
  DDX_Text(pDX, IDC_EDIT_DATA_END, m_data_end);
  DDV_MinMaxByte(pDX, m_data_end, 0, 255);
  DDX_Text(pDX, IDC_EDIT_DATA_START, m_data_start);
  DDV_MinMaxByte(pDX, m_data_start, 0, 255);
  DDX_Text(pDX, IDC_EDIT_UPDATE, m_update);
  DDX_Text(pDX, IDC_EDIT_ID, m_id);
  DDX_Text(pDX, IDC_EDIT_ATTR, m_attr);
  DDX_Text(pDX, IDC_EDIT_SIZE, m_size);
  //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CPacketDlg, CDialog)
  //{{AFX_MSG_MAP(CPacketDlg)
  ON_BN_CLICKED(IDC_BUTTON_UPDATE, OnButtonUpdate)
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CPacketDlg message handlers

void CPacketDlg::OnButtonUpdate()
{
  UpdateData(TRUE);

  m_size = strlen(m_data);

  m_update.Format("%c%03d%03d%03d%s%c ", m_data_start, m_id, m_attr
                        m_size, m_data, m_data_end);
  m_update.Format("%s",m_update);
  UpdateData(FALSE);
}


BOOL CPacketDlg::OnInitDialog()
```

```
{
  CDialog::OnInitDialog();

  UpdateData(FALSE);

  return TRUE;  // return TRUE unless you set the focus to a control
                // EXCEPTION: OCX Property Pages should return
        FALSE
}
```

```
////////////////// MainFrm.cpp ////////////////////
////// 메인 프레임 클래스 수행 프로그램 /////


#include "stdafx.h"
#include "dbxs.h"
#include "MainFrm.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
  //{{AFX_MSG_MAP(CMainFrame)
  ON_WM_CREATE()
  ON_WM_DESTROY()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT indicators[] =
{
  ID_SEPARATOR,           // status line indicator
  ID_INDICATOR_CAPS,
  ID_INDICATOR_NUM,
  ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction
```

```
CMainFrame::CMainFrame()
{
  // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
    return -1;

  if (!m_wndToolBar.Create(this) ||
    !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
  {
    TRACE0("Failed to create toolbar\n");
    return -1;      // fail to create
  }

  if (!m_wndStatusBar.Create(this) ||
    !m_wndStatusBar.SetIndicators(indicators,
      sizeof(indicators)/sizeof(UINT)))
  {
    TRACE0("Failed to create status bar\n");
    return -1;      // fail to create
  }

  // TODO: Remove this if you don't want tool tips or a resizeable
        toolbar
  m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
    CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

  // TODO: Delete these three lines if you don't want the toolbar to
```

```
  //  be dockable
  m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
  EnableDocking(CBRS_ALIGN_ANY);
  DockControlBar(&m_wndToolBar);

  return 0;
}

BOOL CMainFrame::OnCreateClient( LPCREATESTRUCT /*lpcs*/,
  CCreateContext* pContext)
{
  return m_wndSplitter.Create( this,
    2, 2,                 // TODO: adjust the number of rows, columns
    CSize( 10, 10 ),      // TODO: adjust the minimum pane size
    pContext );
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
  // TODO: Modify the Window class or styles here by modifying
  //  the CREATESTRUCT cs

  return CFrameWnd::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
  CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
```

```
  CFrameWnd::Dump(dc);
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

void CMainFrame::OnDestroy()
{
  CFrameWnd::OnDestroy();

// ::RemoveProp(GetSafeHwnd(),AfxGetApp()->m_pszExeName);
}
```

```
////////////////// dbxsView.cpp //////////////////
/////// 화면 내용 표현에 대한 프로그램 ///////


#include "stdafx.h"
#include "dbxs.h"

#include "dbxsDoc.h"
#include "dbxsView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDbxsView
IMPLEMENT_DYNCREATE(CDbxsView, CScrollView)

BEGIN_MESSAGE_MAP(CDbxsView, CScrollView)
  //{{AFX_MSG_MAP(CDbxsView)
  ON_COMMAND(ID_MENU_OPT1, OnMenuOpt1)
  ON_COMMAND(ID_MENU_CONNECT, OnMenuConnect)
  ON_COMMAND(ID_MENU_DISCONNECT, OnMenuDisconnect)
  ON_COMMAND(ID_APP_EXIT, OnAppExit)
  ON_MESSAGE(WM_RECEIVE_PACKET, OnReceivePacket)
  ON_MESSAGE(WM_FAULT_PACKET, OnFaultPacket)
  ON_COMMAND(ID_MENU_SEND_PACKET, OnMenuSendPacket)
  ON_WM_CREATE()
  ON_WM_TIMER()
  ON_WM_DESTROY()
  //}}AFX_MSG_MAP
  // Standard printing commands
  ON_COMMAND(ID_FILE_PRINT, CScrollView::OnFilePrint)
  ON_COMMAND(ID_FILE_PRINT_DIRECT, CScrollView::OnFilePrint)
```

```
  ON_COMMAND(ID_FILE_PRINT_PREVIEW,
          CScrollView::OnFilePrintPreview)

END_MESSAGE_MAP()

// THREAD FUNCTION

DWORD CommWatch(LPVOID lParam)
{
  DWORD  dwEvtMask, dwTransfer, dwError;
  OVERLAPPED  os;
  COMSTAT   cs;
  BYTE  abIn[MAXBLOCK + 1];
  int       nLength;
  CComm*     com = (CComm*) lParam;

  memset(&os, 0, sizeof(OVERLAPPED));

  os.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);
  PurgeComm(com->hDev ,PURGE_RXCLEAR);

  if (os.hEvent == NULL) {
    MessageBox(NULL,"이벤트 에러","이벤트를 생성 할수 없습니
                              다.",MB_ICONERROR|MB_OK);
    return FALSE;
  }

  if (!SetCommMask(com->hDev, EV_RXCHAR)) {
    MessageBox(NULL, "ERROR", "SetCommMask 불가능", MB_OK)
    return FALSE;
  }

  while (com->bConnected) {
    dwEvtMask = 0;
    if (!WaitCommEvent(com->hDev, &dwEvtMask, NULL)) {
      if (GetLastError() == ERROR_IO_PENDING) {
```

```
        GetOverlappedResult(com->hDev, &os, &dwTransfer,FALSE);
      }
    } else {
//    ClearCommError(com->hDev, &dwError, &cs);
      if (((dwEvtMask & EV_RXCHAR) == EV_RXCHAR) &&
          cs.cbInQue) {
        do {
//        memset(abIn, 0, MAXBLOCK);
          nLength = com->ReadCommBlock((LPSTR)abIn, 1);
          if (nLength != 0) {
//          if (Win.p.wndWork != NULL)
          Win.p.wndWork->SendMessage(WM_USER + 1, nLength,
          (LONG)&abIn);
            com->PushToQue((LPSTR)abIn, nLength);

//          ::SendMessage(com->pHwnd, WM_RECEIVE_PACKET, 0, 0);

          }
        } while (nLength > 0);
      }
    }
  }

  PurgeComm(com->hDev ,PURGE_RXCLEAR);
  CloseHandle(os.hEvent);
//hCommWatchThread = NULL ;

  return TRUE;
}

DWORD PacketWatch(LPVOID lParam)
{
//PACKET p;
//DWORD  dwEvtMask, dwTransfer, dwError;
  OVERLAPPED  os;
//COMSTAT    cs;
```

```
//BYTE   abIn[MAXBLOCK + 1];
//int      nLength;
  CComm*     com = (CComm*) lParam;

  memset(&os, 0, sizeof(OVERLAPPED));

  os.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);
  PurgeComm(com->hDev ,PURGE_RXCLEAR);

  if (os.hEvent == NULL) {
    MessageBox(NULL,"이벤트 에러","이벤트를 생성 할수 없습니
                              다.",MB_ICONERROR|MB_OK);
    return FALSE;
  }

  if (!SetCommMask(com->hDev, EV_RXCHAR)) {
    MessageBox(NULL, "ERROR", "SetCommMask 불가능", MB_OK);
    return FALSE;
  }

/*while (com->bConnected) {
    if (com->cpoint > 3) {
      if (com->cpoint >= (int)(*(com->rxbuf + 2) + 5)) {
        p = com->remake_packet();
        if ((p.data_start == DATASTART) && (p.data_end ==
          DATAEND)
          && (p.data_size == strlen(p.data))) {
  //        ::PostMessage(com->pHwnd, WM_RECEIVE_PACKET, 0, 0)
        } else {
  //        ::PostMessage(com->pHwnd, WM_FAULT_PACKET, 0, 0);
        }
      }
    }
  }
*/
  PurgeComm(com->hDev ,PURGE_RXCLEAR);
```

```
  CloseHandle(os.hEvent);

  return TRUE;
}

//
/////////////////////////////////////////////////////////////////////////////
// CDbxsView construction/destruction
////  CDbxsView 의 생성자와 소멸자                    /////


CDbxsView::CDbxsView()
{
  m_RefID = 0x00;
  CurX = 0;
  CurY = 0;
  ppoint = 0;

}

CDbxsView::~CDbxsView()
{
}

BOOL CDbxsView::PreCreateWindow(CREATESTRUCT& cs)
{
  // TODO: Modify the Window class or styles here by modifying
  //  the CREATESTRUCT cs

  return CScrollView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CDbxsView drawing

void CDbxsView::OnDraw(CDC* pDC)
```

```
{
//CString str;
  CDbxsDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  // TODO: add draw code for native data here

//str.Format("%d", m_RefID);
//if (m_RefID >= 100) m_RefID = 0;
//else m_RefID++;

//pDC->TextOut(0, 0, str);
}

void CDbxsView::OnInitialUpdate()
{
  CScrollView::OnInitialUpdate();
  CSize sizeTotal;
  // TODO: calculate the total size of this view
  sizeTotal.cx = sizeTotal.cy = 1000;
  SetScrollSizes(MM_TEXT, sizeTotal);

  GetCom()->pHwnd = GetSafeHwnd();
}


/////////////////////////////////////////////////////////////////////////////
// CDbxsView printing

BOOL CDbxsView::OnPreparePrinting(CPrintInfo* pInfo)
{
  // default preparation
  return DoPreparePrinting(pInfo);
}

void CDbxsView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
```

```
  // TODO: add extra initialization before printing
}

void CDbxsView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
  // TODO: add cleanup after printing
}

/////////////////////////////////////////////////////////////////////////////
// CDbxsView diagnostics

#ifdef _DEBUG
void CDbxsView::AssertValid() const
{
  CScrollView::AssertValid();
}

void CDbxsView::Dump(CDumpContext& dc) const
{
  CScrollView::Dump(dc);
}

CDbxsDoc* CDbxsView::GetDocument() // non-debug version is inline
{
  ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CDbxsDoc)));
  return (CDbxsDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CDbxsView message handlers

void CDbxsView::OnMenuOpt1()
{
  if (dlg.DoModal() == IDOK) {
    if (dlg.m_rtscts) {
```

```
//      GetCom()->dcb.fOutxCtsFlow = TRUE;
//      GetCom()->dcb.fRtsControl   =   RTS_CONTROL_HANDSHAKE   ;

    } else {
//      GetCom()->dcb.fRtsControl = RTS_CONTROL_ENABLE ;
//      GetCom()->dcb.fOutxCtsFlow = FALSE;
    }

    if (dlg.m_xonxoff) {
//      GetCom()->dcb.fInX = TRUE;
//      GetCom()->dcb.fOutX = TRUE;
    } else {
//      GetCom()->dcb.fInX = FALSE;
//      GetCom()->dcb.fOutX = FALSE;
    }

    if (dlg.m_dtrdsr) {
//      GetCom()->dcb.fOutxDsrFlow = TRUE;                    // DSR
          흐름 제어함
//      GetCom()->dcb.fDtrControl = DTR_CONTROL_HANDSHAKE;
    } else {
//      GetCom()->dcb.fOutxDsrFlow = FALSE;      //
          DSR흐름 제어 안함
//      GetCom()->dcb.fDtrControl = DTR_CONTROL_ENABLE;
    }

    GetCom()->PortNumber = dlg.m_port + 1;
    GetCom()->BaudRate = BaudTable[dlg.m_baud];
    GetCom()->StopBits = dlg.m_stop;//StopBitsTable[dlg.m_stop];
    GetCom()->ParityBits = dlg.m_parity; //ParityTable[dlg.m_parity];
  }
//  m_RefID = 0x11;
//      Invalidate();

}
```

```
void CDbxsView::OnMenuConnect()
{
  CDbxsDoc* pDoc = GetDocument();

  if (!GetCom()->bConnected) {
    if (GetCom()->OpenComm() == TRUE) {
      hCommWatchThread =
          AfxBeginThread((AFX_THREADPROC)CommWatch
          (LPVOID)GetCom());

      if (hCommWatchThread == NULL) {
        AfxMessageBox("쓰레드 생성 에러");
      }

      EscapeCommFunction(GetCom()->hDev, SETDTR);
      AfxMessageBox("연결됨");
    } else {
      AfxMessageBox("연결 실패");
    }
  } else {
    GetCom()->CloseComm();
  }
}

void CDbxsView::OnMenuDisconnect()
{
  if (GetCom()->bConnected) {
    if (GetCom()->CloseComm() == FALSE) {
      AfxMessageBox("해제 실패");
    }
  }
}

void CDbxsView::OnAppExit()
{
  OnMenuDisconnect();
```

```
}

CComm* CDbxsView::GetCom()
{
  CDbxsDoc* pDoc = GetDocument();

  return &(pDoc->xcom);
}

LRESULT CDbxsView::OnReceivePacket(WPARAM wParam, LPARAM
        lParam)
{
  GetCom()->interpret_miu_packet();
/*char ch;
  int temp1, temp2;
  CString str;
  TEXTMETRIC text;
  CClientDC dc(this);

  dc.GetTextMetrics(&text);

  if (CurX >= 80) {
    CurX = 0;
    if (CurY >= 25) {
      CurY = 0;
      RedrawWindow();
    } else {
      CurY++;
    }
  } else {
    CurX++;
  }

// temp1 = GetCom()->InBufferCount;
// temp2 = GetCom()->cpoint;
// while (temp1--) {
```

```
  ch = GetCom()->rxbuf[GetCom()->cpoint];
  str.Format("%c", ch);
  dc.TextOut(CurX * 12, CurY * text.tmHeight, str);
//}*/

  return 0;
}

LRESULT CDbxsView::OnFaultPacket(WPARAM wParam, LPARAM
        lParam)
{
  return 0;
}

/*void CDbxsView::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
  BYTE c[5];
  c[0] = nChar;

  if (nChar == '\r') {
    GetCom()->WritecommBlock("\r\n", 2);
  } else {
    GetCom()->WritecommBlock((LPCSTR)c, 1);
  }

  CScrollView::OnChar(nChar, nRepCnt, nFlags);
}  */

void CDbxsView::OnMenuSendPacket()
{
  if (p_dlg.DoModal() == IDOK) {
    GetCom()->WritecommBlock((LPCSTR)p_dlg.m_update,
        strlen(p_dlg.m_update));
  }
```

```
  GetCom()->mi.sell_flag = TRUE;

}

int CDbxsView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CScrollView::OnCreate(lpCreateStruct) == -1)
    return -1;

  SetTimer(1, 100, NULL);

  return 0;
}

void CDbxsView::OnTimer(UINT nIDEvent)
{
  char ch;
  char buf[100];
  int temp1, temp2;
  CString str;
  TEXTMETRIC text;
  CClientDC dc(this);

  dc.GetTextMetrics(&text);

  /*  temp2 = GetCom()->mi.miu_que.rear; //GetCom()->rear;
  while (ppoint != temp2) {
    if (CurX > 80) {
      CurX = 0;
      if (CurY > 30) {
        CurY = 0;
        RedrawWindow();
      } else {
        CurY++;
      }
    } else {
```

```
        CurX++;
      }

      ppoint = (ppoint++) % QUE_SIZE;
      ch = GetCom()->mi.miu_que.que[ppoint];
      str.Format("%c", ch);
      dc.TextOut(CurX * 11, CurY * text.tmHeight, str)
    }
  */
    CScrollView::OnTimer(nIDEvent);
}

void CDbxsView::OnDestroy()
{
    CScrollView::OnDestroy();

    KillTimer(1);

}
```

```
////////////////////// CDbxsDoc.cpp ////////////////////////
///// application에 대한 클래스 실행 정의 프로그램 /////


#include "stdafx.h"
#include "dbxs.h"

#include "MainFrm.h"
#include "dbxsDoc.h"
#include "dbxsView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CDbxsApp

BEGIN_MESSAGE_MAP(CDbxsApp, CWinApp)
  //{{AFX_MSG_MAP(CDbxsApp)
  ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    // NOTE - the ClassWizard will add and remove mapping macros
          here.
    //    DO NOT EDIT what you see in these blocks of generated code!
  //}}AFX_MSG_MAP
  // Standard file based document commands
  ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
  ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
  // Standard print setup command
  ON_COMMAND(ID_FILE_PRINT_SETUP, CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CDbxsApp construction
```

```
CDbxsApp::CDbxsApp()
{
  // TODO: add construction code here,
  // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CDbxsApp object

CDbxsApp theApp;

/////////////////////////////////////////////////////////////////////////////
// CDbxsApp initialization

BOOL CDbxsApp::InitInstance()
{
  // Standard initialization
  // If you are not using these features and wish to reduce the size
  //  of your final executable, you should remove from the following
  //  the specific initialization routines you do not need.
  ::CreateMutex(NULL,TRUE,m_pszExeName);
  if (GetLastError()==ERROR_ALREADY_EXISTS) {
    CWnd* pPrevWnd =
          CWnd::GetDesktopWindow()->GetWindow(GW_CHILD);
    while(pPrevWnd) {
      if (::GetProp(pPrevWnd->GetSafeHwnd(),m_pszExeName)) {
        if (pPrevWnd->IsIconic()) {
          pPrevWnd->ShowWindow(SW_RESTORE);
        }
        pPrevWnd->SetForegroundWindow();
        pPrevWnd->GetLastActivePopup()->SetForegroundWindow()
        return FALSE;
      }
      pPrevWnd = pPrevWnd->GetWindow(GW_HWNDNEXT);
```

```
    }

    return FALSE;
  }

#ifdef _AFXDLL
  Enable3dControls();     // Call this when using MFC in a shared DLL
#else
  Enable3dControlsStatic();     // Call
            this when linking to MFC statically
#endif

  LoadStdProfileSettings();  // Load standard INI file options (including
          MRU)

  // Register the application's document templates.  Document templates
  //   serve as the connection between documents, frame windows and
          views.

  CSingleDocTemplate* pDocTemplate;
  pDocTemplate = new CSingleDocTemplate(
    IDR_MAINFRAME,
    RUNTIME_CLASS(CDbxsDoc),
    RUNTIME_CLASS(CMainFrame),        // main SDI frame window
    RUNTIME_CLASS(CDbxsView));
  AddDocTemplate(pDocTemplate);

  // Parse command line for standard shell commands, DDE, file open
  CCommandLineInfo cmdInfo;
  ParseCommandLine(cmdInfo);

  // Dispatch commands specified on the command line
  if (!ProcessShellCommand(cmdInfo))
    return FALSE;

// ::GetProp(pPrevWnd->GetSafeHwnd(),m_pszExeName)
```

```
//::SetProp(pMainFrame->GetSafeHwnd(),m_pszExeName,(HANDLE)1);

  return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
  CAboutDlg();

// Dialog Data
  //{{AFX_DATA(CAboutDlg)
  enum { IDD = IDD_ABOUTBOX };
  //}}AFX_DATA

  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CAboutDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV
          support
  //}}AFX_VIRTUAL

// Implementation
protected:
  //{{AFX_MSG(CAboutDlg)
    // No message handlers
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
```

```
  //{{AFX_DATA_INIT(CAboutDlg)
  //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CAboutDlg)
  //}}AFX_DATA_MAP
}
BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
  //{{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()
// App command to run the dialog
void CDbxsApp::OnAppAbout()
{
  CAboutDlg aboutDlg;
  aboutDlg.DoModal();
}
```

```
////////////////////////////////////////////////////////////////////////////
// CDbxsApp commands
//////////////////  comm. cpp //////////////////
#include "stdafx.h"
#include "comm.h"

CComm::CComm()
{
  hDev     = NULL;
  pHwnd    = NULL;
  bConnected  = FALSE;
  PortNumber = 2;
  BaudRate  = CBR_38400;
  DataBits= 8;
  ParityBits = NOPARITY;
  StopBits= ONESTOPBIT;
  front    = 0;
  rear   = 0;
//BufferR = 0;

  CommTimeOuts.ReadIntervalTimeout = 0;
  CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
  CommTimeOuts.ReadTotalTimeoutConstant = 0xffffffff;
  CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
  CommTimeOuts.WriteTotalTimeoutConstant = 100;
  mi.sell_flag = FALSE;
  mi.miu_cnt = 0;
  mi.sw_job = 0;
}

CComm::~CComm()
{
}

BOOL CComm::OpenComm()
{
```

```
CString szPort = _T("");

// Read, Write event 생성
osRead.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);

if (osRead.hEvent == NULL) {
  AfxMessageBox("READ EVENT CREATE FAILURE",
        MB_OK|MB_ICONERROR);
  return FALSE;
} else {
  osRead.Offset = 0;
  osRead.OffsetHigh = 0;
}

osWrite.hEvent = CreateEvent(NULL,TRUE,FALSE,NULL);

if (osWrite.hEvent == NULL) {
  MessageBox(NULL, "ERROR","WRITE EVENT CREATE FAILURE
        MB_OK|MB_ICONERROR);
  CloseHandle(osRead.hEvent);
  return FALSE;
} else {
  osWrite.Offset = 0;
  osWrite.OffsetHigh = 0;
}

// com port open
szPort.Format("COM%d", PortNumber);

if ((hDev = CreateFile(szPort,GENERIC_READ
   GENERIC_WRITE,0,NULL,OPEN_EXISTING,
                FILE_ATTRIBUTE_NORMAL
        FILE_FLAG_OVERLAPPED,NULL)) == (HANDLE)-1) {
  MessageBox(NULL, "ERROR", "PORT OPEN
        FAILURE",MB_OK|MB_ICONERROR);
  bConnected = FALSE;
```

```
    return FALSE;
  } else {
    bConnected = TRUE;
    SetupComm(hDev, MAXBLOCK, MAXBLOCK);

    CommTimeOuts.ReadIntervalTimeout = 0;
    CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
    CommTimeOuts.ReadTotalTimeoutConstant = 0xffffffff;
    CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
    CommTimeOuts.WriteTotalTimeoutConstant = 100;

    SetCommTimeouts(hDev, &CommTimeOuts);
  }

  // DCB 설정
  dcb.DCBlength = sizeof(DCB);
  GetCommState(hDev, &dcb);

  dcb.BaudRate    = BaudRate;
  dcb.ByteSize    = DataBits;
  dcb.Parity      = ParityBits;
  dcb.StopBits    = StopBits;

    dcb.fOutxCtsFlow  = 0;
  dcb.fOutxDsrFlow  = 0;
  dcb.fRtsControl     = DTR_CONTROL_DISABLE;
  dcb.XonLim      = MAXBLOCK;
  dcb.XoffLim     = MAXBLOCK;
  dcb.XonChar     = ASCII_XON;
  dcb.XoffChar    = ASCII_XOFF;
  dcb.fOutX       = FALSE;
  dcb.fInX      = FALSE;

  dcb.fBinary       = FALSE;
  dcb.fParity     = FALSE;
```

```
  if (SetCommState(hDev, &dcb) == FALSE) {
    MessageBox(NULL, "ERROR", "CONFIGURATION FAILURE",
          MB_OK|MB_ICONERROR);
    return FALSE;
  }

  return TRUE;
}

BOOL CComm::CloseComm()
{
  bConnected = FALSE;

  SetCommMask(hDev, 0);

  EscapeCommFunction(hDev, CLRDTR);
  PurgeComm(hDev,
          PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PUR
          GE_RXCLEAR);
  CloseHandle(hDev);

  return TRUE;
}

int CComm::ReadCommBlock(LPSTR lpszBlock, int nMaxLength)
{
  DWORDdwErrorFlags;
  DWORDdwLength;
  COMSTAT ComStat;
  CString szError;
  DWORDdwError = 0;

  ClearCommError(hDev, &dwErrorFlags, &ComStat);
  dwLength = min((unsigned)nMaxLength, ComStat.cbInQue);

  if (dwLength > 0) {
```

```
  int ret = ReadFile(hDev, lpszBlock, dwLength, &dwLength, &osRead);
  if (!ret) {
    if (GetLastError() == ERROR_IO_PENDING) {

      while(!GetOverlappedResult(hDev, &osRead, &dwLength, TRUE)) {
        dwError = GetLastError();
        if(dwError == ERROR_IO_INCOMPLETE) {
          continue;
        } else {
          szError.Format("<CE-%u>", dwError);
          MessageBox( NULL, szError, "TTY Error!",
            MB_ICONEXCLAMATION | MB_OK ) ;
          ClearCommError(hDev, &dwErrorFlags, &ComStat);
          if (dwErrorFlags > 0) {
            szError.Format("<CE-%u>", dwError);
            MessageBox( NULL, szError, "TTY Error!",
              MB_ICONEXCLAMATION | MB_OK ) ;
          }
          break;
        }
      }
    } else {
      dwLength = 0;
      ClearCommError(hDev, &dwErrorFlags, &ComStat);
      if (dwErrorFlags > 0) {
        szError.Format("<CE-%u>", dwError);
        MessageBox( NULL, szError, "TTY Error!",
        MB_ICONEXCLAMATION | MB_OK ) ;
      }
    }
  }

  return dwLength;
}
```

```
BOOL CComm::WritecommBlock(LPCSTR lpByte, DWORD
        dwBytesToWrite)
{
  BOOL fWriteStat;
  DWORDdwBytesWritten;
  DWORDdwErrorFlags;
  COMSTAT ComStat;
  CString szError;
  DWORDdwError;

  if (bConnected == FALSE) return FALSE;

  fWriteStat = WriteFile(hDev, lpByte, dwBytesToWrite, &dwBytesWritten
        &osWrite);
  if (!fWriteStat) {
    TRACE("GetLastError() = %ld\n", GetLastError());
    if (GetLastError() == ERROR_IO_PENDING) {
      while(!GetOverlappedResult(hDev, &osWrite, &dwBytesWritten,
        TRUE)) {
        dwError = GetLastError();
        if (dwError == ERROR_IO_INCOMPLETE) {
          continue;
        } else {
          szError.Format("<CE-%u>", dwError);
          MessageBox( NULL, szError, "TTY Error!",
          MB_ICONEXCLAMATION | MB_OK ) ;
          ClearCommError(hDev, &dwErrorFlags, &ComStat);
          if (dwErrorFlags > 0) {
            szError.Format("<CE-%u>", dwError);
            MessageBox( NULL, szError, "TTY Error!",
          MB_ICONEXCLAMATION | MB_OK ) ;
          }
          break;
        }
      }
```

```
    } else {
      szError.Format("<Port-%d>", PortNumber);
      MessageBox( NULL, szError, "TTY Error!",
          MB_ICONEXCLAMATION | MB_OK ) ;
      ClearCommError(hDev, &dwErrorFlags, &ComStat)
      return FALSE;
    }
  }
  return TRUE;
}

/*
void CComm::PushToQue(LPSTR str, int count)
{
// ASSERT(count > 0);
// ASSERT(Label != NULL);
  InBufferCount = count;

  while (count-- > 0) {
    rear = (rear++) % MAXBLOCK;
    rxbuf[rear] = *(str++);
//  if (cpoint > MAXBLOCK) cpoint = 0;

  }
}
*/
BOOL CComm::check_miu_buf()
{
  char temp[100];

  memset(temp, 0, 100);

  mi.p.data_start = mi.miu_que.buf[0];
  strncpy(temp, mi.miu_que.buf+1, 3);
  mi.p.id = atoi(temp);
  strncpy(temp, mi.miu_que.buf+4, 3);
```

```
  mi.p.attr = atoi(temp);
  strncpy(temp, mi.miu_que.buf+7, 3);
  mi.p.data_size = atoi(temp);
  memset(mi.p.data, 0, 50);
  strncpy(mi.p.data, mi.miu_que.buf+10, mi.p.data_size);
  mi.p.data_end = mi.miu_que.buf[mi.p.data_size+10];

  memset(mi.miu_que.buf, 0, BUF_SIZE);

  if (mi.p.data_start == 1 && mi.p.data_end == 2) return TRUE

  return FALSE;
}

void CComm::interpret_miu_packet()
{
  switch (mi.p.attr) {
    case BEST_PRICE:
      strcpy(mi.price, mi.p.data);
      break;
  }
}

void CComm::PushToQue(LPSTR str, int count)
{
  while (count-- > 0) {
    if (mi.miu_que.front != mi.miu_que.rear) {
      mi.miu_que.front = (mi.miu_que.front + 1) % QUE_SIZE;
      if (mi.miu_que.que[mi.miu_que.front] == 1) {
        mi.miu_que.buf_pnt = 0;
      }
      mi.miu_que.que[mi.miu_que.front] = *(str++);
      mi.miu_que.buf[mi.miu_que.buf_pnt++]
          mi.miu_que.que[mi.miu_que.front];

      if (mi.miu_que.que[mi.miu_que.front] == 2) {
```

```
            if (check_miu_buf() == TRUE) {
              ::SendMessage(pHwnd, WM_RECEIVE_PACKET, 0, 0)
//            interpret_miu_packet();
            }
          }
        }
      }
    }
```

```
////////////////////////// v55.c ////////////////////////
/*********** V55 제어 프로그램 ************/


#include <dos.h>
#include <stdarg.h>
#include <stdio.h>
#include "v55.h"
#include "v55-1.h"

#define byte unsigned char

void E_Pulse(void)
{
     setV55reg(P5,4);
     ddelay(1000);
     resetV55reg(P5,4);
}

void L_outch(byte ch)
{
     int i;
     setV55reg(P5,1);
     pokeV55reg(P4,ch);
     E_Pulse();
}


void L_control(byte ch)
{
     int i;
     resetV55reg(P5,1);
     pokeV55reg(P4,ch);
     E_Pulse();
}
```

```
void L_outstr(byte *msg)
{
    int i;
    for(i=0;msg[i];i++) L_outch(msg[i]);
}


void L_printf(char *form, ... )
{
  char buff[128];
  va_list  argptr;
  va_start( argptr, form );
  vsprintf(buff, form, argptr );
  L_outstr(buff);
  va_end( argptr );
}

void L_init(void)  /* LCD 초기화 */
{
    pokeV55reg(PMC5,0);
    pokeV55reg(PM5,0xF0);
    pokeV55reg(PMC4,0);
    pokeV55reg(PM4,0);
    resetV55reg(P5,1);
    resetV55reg(P5,2);
    L_control(0x02);
    L_control(0x38);
    L_control(0x01);
    L_control(0x0e);
    L_control(0x06);
}


void L_cls()
{
```

```
    L_control(0x01);
}

void L_goto( char vx, char vy)
{
  if( vx>15)
    vx = 15;
  if( vy>1)
    vy = 1;
  if( !vy)
    L_control( 0x80+vx);
  else
    L_control( 0xc0+vx);
}

void L_left_shift(void)
{
    L_control(0x18);
}

void L_right_shift(void)
{
    L_control(0x1c);
}


void LCD_ins(void)
{
    L_control(0x1c);
}


void LCD_2nd(void)
{
    L_control(0xc0);
}
```

```
void one_chip_init()
{
    disable();
/*  pokeV55reg(MBC,0x35);*/
    pokeV55reg(PWC0,0xEA);
    pokeV55reg(PWC1,0xAA);
    pokeV55reg(PRC,0xEC);
    pokeV55reg(RFM,0x00);
    pokeV55reg(IMC,0xB0);
}

void scu_init()
{
    pokeV55reg(PMC3,0x0b);
    pokeV55reg(PM3,0x00);
    pokeV55reg(P3,0);
    pokeV55reg(IC26,0x43);
    pokeV55reg(IC28,0x43);
    pokeV55reg(IC30,0x43);
    set_baud(38400);
    pokeV55reg(ASP,0x01);
    pokeV55reg(UARTM0,0xC8);
}
```

```
void set_baud(unsigned int bps)
{
        switch (bps){
        case 9600: {
            pokeV55reg(TXBRG0,163);
            pokeV55reg(RXBRG0,163);
            pokeV55reg(PRS0,2);
            break;
        }
        case 19200: {
            pokeV55reg(TXBRG0,163);
            pokeV55reg(RXBRG0,163);
            pokeV55reg(PRS0,1);
            break;
        }
        case 38400: {
            pokeV55reg(TXBRG0,81);
            pokeV55reg(RXBRG0,81);
            pokeV55reg(PRS0,0);
            break;
        }
        default: {    /* default value : 9600 bps */
            pokeV55reg(TXBRG0,163);
            pokeV55reg(RXBRG0,163);
            pokeV55reg(PRS0,2);
            break;
        }
    }
}
void V_outch(byte c)
{
/*     do {} while( (peekV55reg(UARTS0) & 0x20) == 0); */
    pokeV55reg(TXB0,c);
}
void V_outstr(char *msg)
{
```

```
    int i;
    for(i=0;msg[i];i++) {
        if(msg[i]=='\n')
        {
            V_outch('\r');
            V_outch('\n');
            continue;
        }
        V_outch(msg[i]);
    }
}


void V_printf(char *form, ... )
{
  char buff[128];
  va_list  argptr;
  va_start( argptr, form );
  vsprintf(buff, form, argptr );
  V_outstr(buff);
  va_end( argptr );
}


byte V_inch()
{
    byte i;
/*    do {} while( (peekV55reg(UARTS0) & 0x10) == 0); */
    i = peekV55reg(RXB0);
    return( i );
}
void install (void interrupt (*faddr)(), int inum)
{
    disable();
    poke( 0, inum * 4, faddr);
    poke( 0, (inum*4) + 2, _CS);
```

```
    enable();
}

void OS()
{
    __emit__(0xea,0x00,0x00,0xff,0xff);
}

void abort(){}
void _RealCvtVector(){}
```

```
////////////////////////// serial.c //////////////////////////
/************ 직렬 통신 프로그램 ************/


#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include "serial.h"

#define TX 0
#define RX 0
#define DLL 0
#define IER 1
#define DLH 1
#define IIR 2
#define LCR 3
#define MCR 4
#define LSR 5
#define MSR 6

#define RX_INT 1
#define INT_MASK 7
#define RX_ID 4

#define DLAB 0x80
#define DTR 1
#define RTS 2
#define MC_INT 8

#define RX_RDY 0x01
#define TX_RDY 0x02

#define CTS 0x01
#define DSR 0x20
```

```
#define DCD 0x80

#define IMR 0x21
#define ICR 0x20

#define IRQ3 0xf7
#define IRQ4 0xef
#define EOI 0x20

#define XON 0x11
#define XOFF 0x13

int port_open=FALSE,irq;
void interrupt (*oldvect)();

int port_exist(int port)
{
  return mpeek(0, 0x400+(port-1)*2);
}

void open_port(int port, int inlen)
{
  long i;

  if (port_open) {
    printf("Unable to open port:A port is already open!\n");
    exit(1);
  }
  if (port<1||port>4||!port_exist(port)) {
    printf("Unable to open port:Invalid port number!\n");
    exit(1);
  }
  ilen=inlen;
  if ((inbuff=farmalloc(ilen)==NULL)) {
    printf("Unable to open port:Not enough memory for the buffer!\n")
    exit(1);
```

```
}
foff=(int)((long)ilen*90L/100L);
fon=(int)((long)ilen*80L/100L);
rx_flow=FALSE;

switch (port) {
  case 1:
    base=0x3f8;
    irq=0x0c;
    break;
  case 2:
    base=0x2f8;
    irq=0x0b;
    break;
  case 3:
    base=0x3e8;
    irq=0x0c;
    break;
  case 4:
    base=0x2e8;
    irq=0x0b;
    break;
}
disable();
oldvect=getvect(irq);
setvect(irq,handler);
sibuff=eibuff=0;
outportb(base+MCR,inportb(base+MCR)|MC_INT|DTR|RTS)
outportb(base+IER,RX_INT);
outportb(IMR,inportb(IMR)&(irq==0x0b?IRQ3:IRQ4));
enable();
fifo(4);
set_tx_rts(FALSE);
set_tx_dtr(FALSE);
set_tx_xon(FALSE);
set_rx_rts(FALSE);
```

```
  set_rx_dtr(FALSE);
  set_rx_xon(FALSE);
  port_open=TRUE;
}

void close_port(void)
{
  if (!port_open) return;

  port_open=FALSE;
  disable();
  outportb(IMR,inportb(IMR)&(irq==0x0b?~IRQ3:~IRQ4))
  outportb(base+IER,0);
  outportb(base+MCR,inportb(base+MCR)&~MC_INT);
  setvect(irq,oldvect);
  enable();
  outportb(base+MCR,inportb(base+MCR)&~RTS);
}

void purge_in(void)
{
  disable();
  sibuff=eibuff=0;
  enable();
}

void set_baud(long baud)
{
  int c,n;

  if (baud==0L) return;

  n=(int)(115200L/baud);
  disable();
  c=inportb(base+LCR);
  outportb(base+LCR,(c|DLAB));
```

```
  outportb(base+DLL,n&0x00ff);
  outportb(base+DLH,(n>>8)&0x00ff);
  outportb(base+LCR,c);
  enable();
}

long get_baud(void)
{
  int c,n;

  disable();
  c=inportb(base+LCR);
  outportb(base+LCR,(c|DLAB));
  n=inportb(base+DLH)<<8;
  n|=inportb(base+DLL);
  outportb(base+LCR,c);
  enable();
  return 115200L/(long)n;
}

int get_bits(void)
{
  return (inportb(base+LCR)&3)+5;
}

int get_parity(void)
{
  switch ((inportb(base+LCR)>>3)&7) {
    case 0:
      return NO_PARITY;
    case 1:
      return ODD_PARITY;
    case 2:
      return EVEN_PARITY;
  }
  return -1;
```

```
}

int get_stopbits(void)
{
  if (inportb(base+LCR)&4) return 2;
  return 1;
}

void set_data_format(int bits,int parity,int stopbit)
{
  int n;

  if (parity<NO_PARITY||parity>ODD_PARITY) return

  if (bits<5||bits>8) return;
  if (stopbit<1||stopbit>2) return;

  n=bits-5;
  n|=((stopbit==1)?0:4);
  switch (parity) {
    case NO_PARITY:
      break;
    case ODD_PARITY:
      n|=0x08;
      break;
    case EVEN_PARITY:
      n|=0x18;
      break;
  }
  disable();
  outportb(base+LCR, n);
  enable();
}

void set_port(long baud, int bits, int parity, int stopbit)
{
```

```
  if (!port_open) return;
  set_baud(baud);
  set_data_format(bits, parity, stopbit);
}

int in_ready(void)
{
  return !(sibuff==eibuff);
}

int carrier(void)
{
  return inportb(base+MSR) & DCD ? TRUE : FALSE
}

void set_dtr(int n)
{
  if (n)
    outportb(base+MCR, inportb(base+MCR)|DTR);
  else
    outportb(base+MCR, inportb(base+MCR)&~DTR);
}

void fifo(int n)
{
  int i;

  switch(n) {
    case 1:
      i=1;
      break;
    case 4:
      i=0x41;
      break;
    case 8:
      i=0x81;
```

```
        break;
      case 14:
        i=0xc1;
        break;
      default:
        i=0;
    }
    outportb(base+IIR,i);
}

void set_tx_rts(int n)
{
    tx_rts=n;
}

void set_tx_dtr(int n)
{
    tx_dtr=n;
}

void set_tx_xon(int n)
{
    tx_xon=n;
    tx_xonoff=FALSE;
}

void set_rx_rts(int n)
{
    rx_rts=n;
}

void set_rx_dtr(int n)
{
    rx_dtr=n;
}
```

```
void set_rx_xon(int n)
{
  rx_xon=n;
}

int get_tx_rts(void)
{
  return tx_rts;
}

int get_tx_dtr(void)
{
  return tx_dtr;
}

int get_tx_xon(void)
{
  return tx_xon;
}

int get_rx_rts(void)
{
  return rx_rts;
}

int get_rx_dtr(void)
{
  return rx_dtr;
}

int get_rx_xon(void)
{
  return rx_xon;
}
```

```
////////////////////////// miu.c //////////////////////////
/******* 메인 인터페이스 unit 프로그램 *******/


#include <dos.h>
#include <string.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include "v55.h"
#include "intbank.h"
#include "define.h"

#define  HIGH    0xf0
#define  LOW         0x0f
#define  QUE_SIZE  150
#define  BUF_SIZE  100
#define  SELL_START  900
#define  SELL_END  901
#define  REMAIN_TIME    100
#define BEST_PRICE    101
#define  OTHER_INFO  102
#define INFORM       200
#define A_INFORM     201
#define GM_INFO      601
#define A_GM_INFO   602
#define UserID       106
#define MID           1
#define ALL_MIU       0
#define ALL_SIU       0


/*-----------------------------------------------*/

typedef struct {
  char   data_start;
```

```
  int  id;
  int  attr;
  int  data_size;
  char  data[50];
  char  data_end;
} PACKET;

typedef struct {
  char que[QUE_SIZE+1];
  int  front, rear;
  char buf[BUF_SIZE+1];
  char buf_pnt;
} QUE;

typedef struct {
  int info;
  char idpass[7];
} inform;

typedef struct {
  int   siu_id;
  int   siu_cnt;
  int   siu_idB;
  int   sw_jobB;
  int   sw_job;
  int   id;
  char siu_status;
  char price[20];
  char   best_price[20];
  char   min_price[20];
  char remain_time[5];
  char other_info[30];
  QUE  pc_que, siu_que;
  PACKET p, p1;
  char  sell_flag;
  int inform_id;
```

```
  inform info[9];
  char gm_info[30];
  int price_id;
  char buf[20];

} MAIN_INFO;

MAIN_INFO   mi;

/*********************************************************************
*********/
void txd_pc(char*);
void txd_siu(char*);
void far interrupt rxd_pc_int(void);
void far interrupt rxd_siu_int(void);

/*********************************************************************/

void init_miu(void);
void init_data();
void ddelay(word);

void tx_pc(char*);
void interpret_siu_packet();
void interpret_pc_packet();
int  check_pc_buf();
void pc_read();
void siu_read();
void mi_sw_job();
void push_job();
/*************************************************************/

void init_miu(void)
{
  pokeV55reg(MBC,0x35);
  pokeV55reg(RFM,0x00);
```

```
pokeV55reg(PWC0,0xea);
pokeV55reg(PWC1,0xaa);
pokeV55reg(PRC,0xec);
pokeV55reg(IMC,0x30);

pokeV55reg(PMC2,0x00);
pokeV55reg(PMC3,0x33);
pokeV55reg(PMC4,0x00);
pokeV55reg(PMC5,0x00);
pokeV55reg(PMC7,0x00);
pokeV55reg(PM7, 0x00);
pokeV55reg(P7,0x00);
pokeV55reg(PMC8,0x00);
pokeV55reg(PM0,0xff);
pokeV55reg(PM2,0x00);
pokeV55reg(PM4,0xff);
pokeV55reg(PM5,0x00);
pokeV55reg(PM8,0x01);
pokeV55reg(P2,0x00);
pokeV55reg(P0,0x00);
pokeV55reg(P5,0x00);
pokeV55reg(P8,0x02);

pokeV55reg(ASP,0x03);
pokeV55reg(PRS0,0x00);
pokeV55reg(PRS1,0x00);
pokeV55reg(TXBRG0,81);
pokeV55reg(RXBRG0,81);
pokeV55reg(TXBRG1,81);
pokeV55reg(RXBRG1,81);
pokeV55reg(UARTM0,0xc8);
pokeV55reg(UARTM1,0xc8);

pokeV55reg(IC28,0x40);
SET_VECTOR(INTSR0,rxd_pc_int);
pokeV55reg(IC28,0x00);
```

```
    pokeV55reg(IC29,0x41);
    SET_VECTOR(INTSR1,rxd_siu_int);
    pokeV55reg(IC29,0x01);

/*  pokeV55reg(IC18,0x42);
    SET_VECTOR(INTCM10,data_process)

    pokeV55reg(IC18,0x02);
    pokeV55regw(CM10,0xffff);
    pokeV55reg(TMC0, 0x80);*/

    enable();
}

void init_data()
{
    mi.siu_id = 1;
    mi.sw_job = 0;
    mi.siu_cnt = 0;
    memset(mi.price, 0 , 20);
    memset(mi.other_info, 0, 30);
    mi.pc_que.front = 0;
    mi.pc_que.rear = 0;
    mi.pc_que.buf_pnt = 0;
    mi.sell_flag = FALSE;
    mi.p.attr=1;
    mi.inform_id=1;

}

void ddelay(word time)
{
    word i;
    for(i=0;i<=time;i++);
}
```

```
/*************************************************************************
********/
void far interrupt rxd_pc_int(void)
{
  _DS=0;

  pokeV55reg(IC28,0x40);
  mi.pc_que.rear = (mi.pc_que.rear + 1) % QUE_SIZE;
  mi.pc_que.que[mi.pc_que.rear] = peekV55reg(RXB0);
  pokeV55reg(IC28,0x00);

  fint;
}

void far interrupt rxd_siu_int()
{
  _DS=0;

  pokeV55reg(IC29, 0x41);
  mi.siu_que.rear = (mi.siu_que.rear + 1) % QUE_SIZE;
  mi.siu_que.que[mi.siu_que.rear] = peekV55reg(RXB1);
  pokeV55reg(IC29, 0x01);

  fint;
}

void txd_pc(char *str)
{
  int i;
  pokeV55reg(P8,0x0f);
  for (i = 0; *(str+i); i++) {
    pokeV55reg(TXB0,*(str+i));
    while (peekV55reg(UARTS0) == 0x20);
    ddelay(100);
  }
```

```
  pokeV55reg(P8,0x00);
}

void txd_siu(char* str)
{
  int i;

  for (i = 0; *(str+i); i++) {
    pokeV55reg(TXB1,*(str+i));
    while (peekV55reg(UARTS1) == 0x20);
    ddelay(100);
  }
}

/*************************************************************************
*********/
int check_pc_buf()
{
  char temp[100];

  memset(temp, 0, 100);

  mi.p.data_start = mi.pc_que.buf[0];
  strncpy(temp, mi.pc_que.buf+1, 3);
  mi.p.id = atoi(temp);
  strncpy(temp, mi.pc_que.buf+4, 3);
  mi.p.attr = atoi(temp);
  strncpy(temp, mi.pc_que.buf+7, 3);
  mi.p.data_size = atoi(temp);
  memset(mi.p.data, 0, 50);
  strncpy(mi.p.data, mi.pc_que.buf+10, mi.p.data_size);
  mi.p.data_end = mi.pc_que.buf[mi.p.data_size+10];

  memset(mi.pc_que.buf, 0, BUF_SIZE);

  if (mi.p.data_start == 1 && mi.p.data_end == 2) return TRUE;
```

```
  return FALSE;
}

int check_siu_buf()
{
  char temp[100];

  memset(temp, 0, 100);

  mi.pl.data_start = mi.siu_que.buf[0];
  strncpy(temp, mi.siu_que.buf+1, 3);
  mi.pl.id = atoi(temp);
  strncpy(temp, mi.siu_que.buf+4, 3);
  mi.pl.attr = atoi(temp);
  strncpy(temp, mi.siu_que.buf+7, 3);
  mi.pl.data_size = atoi(temp);
  memset(mi.pl.data, 0, 50);
  strncpy(mi.pl.data, mi.siu_que.buf+10, mi.pl.data_size);
  mi.pl.data_end = mi.siu_que.buf[mi.pl.data_size+10];

  memset(mi.siu_que.buf, 0, BUF_SIZE);

  if (mi.pl.data_start == 1 && mi.pl.data_end == 2) return TRUE

  return FALSE;
}

void push_job()
{
        mi.sw_jobB=mi.sw_job;
        mi.siu_idB=mi.siu_id;
}
void interpret_pc_packet()
{
  char temp[100] ={0};
```

```
int temp_id=0;

if ((int)((mi.p.id)/100)==MID || (int)((mi.p.id)/100)==ALL_MIU)
{

  switch(mi.p.attr) {
    case SELL_START:
      sprintf(temp, P_FORM, 1, ALL_SIU, SELL_START,
                    strlen(mi.p.data), mi.p.data, 2);
      txd_siu(temp);
      memset(mi.best_price, 0, 11);
      strncpy(mi.best_price,mi.p.data,30);
      mi.min_price[10]=0;
      memset(mi.price, 0, 11);
      mi.price[10]=0;
      memset(mi.p1.data,0,20);
      mi.price_id=0;
      ddelay(1000);

/*    txd_pc("I'm ready to sell...");*/
      mi.sell_flag = TRUE;
      mi.sw_job=6;
      mi.siu_id=1;
      break;
    case SELL_END:
      mi.sell_flag = FALSE;
      sprintf(temp, P_FORM, 1, ALL_SIU, SELL_END,
                                strlen(mi.p.data), mi.p.data, 2);
      txd_siu(temp);
/*    txd_pc("OK! sell end...");*/
      mi.sw_job=0;
      mi.siu_id=1;

      break;
    case UserID:
      sprintf(temp, P_FORM, 1, mi.p.id%100 , UserID, strlen(mi.p.data)
```

```
mi.p.data, 2);/*?*/
          txd_siu(temp);
           break;
        case INFORM:
          {
/*          sprintf(temp, P_FORM, 1, mi.inform_id*10 , A_INFORM,
        strlen(mi.info[mi.inform_id].idpass),mi.info[mi.inform_id].idpass , 2);
            txd_pc(temp);
            mi.info[mi.p.id].info=3;*/
          }
        mi.inform_id++;
        if (mi.inform_id>8) mi.inform_id=1;
        break;

      case GM_INFO:
          memset(mi.gm_info,0,30);
          strcpy(mi.gm_info,mi.p.data);
/*        sprintf(temp, P_FORM, 1, MID*100 , A_GM_INFO,
           11,"GM_INFO_OK!" , 2);
          txd_pc(temp);*/
          push_job();
          mi.sw_job=2;
          mi.siu_id=1;
          break;
      default: break;
    }

    if (mi.sell_flag) {
      switch (mi.p.attr) {
        case BEST_PRICE:
          memset(mi.best_price, 0, 11);
          strncpy(mi.best_price,mi.p.data,11);
          mi.best_price[10]=0;
          sprintf(temp, P_FORM, 1, MID*100, BEST_PRICE, strlen(mi.price)
 mi.price, 2);
          txd_pc(temp);
```

```
            break;
        case REMAIN_TIME:
            strcpy(mi.remain_time, mi.p.data);
            break;
        case OTHER_INFO:
            strcpy(mi.other_info, mi.p.data);
            break;
        default :
            break;
      }
    }
    }
}

void interpret_siu_packet()
{
/*int i;                    */
  char temp[100] ={0};
  char temp1[100]={0};

  switch (mi.p1.attr) {
    case BEST_PRICE:
/*
sprintf(mi.buf,"(S%d_%s>P%d_%s)",mi.p1.id,mi.p1.data,mi.price_id,mi.price);
          txd_pc(mi.buf);*/

        if (atol(mi.p1.data)>atol(mi.price))
        {
          strcpy(mi.price, mi.p1.data);
/* ??????????? */ strcpy(mi.best_price, mi.price);
          mi.price_id=mi.p1.id;
          sprintf(temp, P_FORM, 1, MID*100+mi.price_id, BEST_PRICE
strlen(mi.price),mi.price, 2);
          txd_pc(temp);

/*        sprintf(mi.buf,"___%ld",atol(mi.price));
```

```
          txd_pc(mi.buf);*/
        }
        break;
      case A_INFORM:
/*      txd_pc("!");*/
        mi.info[mi.p1.id/10].info=2;
        strncpy(mi.info[mi.p1.id/10].idpass,mi.p1.data,6);
        mi.info[mi.p1.id/10].idpass[6]=0;
        if (mi.p1.data_size)
        {
          sprintf(temp, P_FORM, 1, MID*100+mi.p1.id, A_INFORM,
 strlen(mi.p1.data),mi.p1.data, 2);
/*      sprintf(temp1, "SIU %d
 %s",MID*100+mi.p1.id,mi.info[mi.p1.id/10].idpass);
        sprintf(temp, P_FORM, 1, 0, INFORM, strlen(temp1), temp1, 2);*/
          txd_pc(temp);
        }
        break;
      default : break;
    }
}

/*****************************************************************/

void main()
{
  char buf[10];
  char check_flag;
  init_miu();
  init_data();
  mi.inform_id=0;

/*pokeV55reg(P8, 0x00);*/ ,

  for (;;)
  {
```

```
        pc_read();
        mi_sw_job();
        siu_read();
    }

}
void pc_read()
{
        if (mi.pc_que.front != mi.pc_que.rear)
        {
          mi.pc_que.front = (mi.pc_que.front + 1) % QUE_SIZE
          if (mi.pc_que.que[mi.pc_que.front] == 1) {
            mi.pc_que.buf_pnt = 0;
          }

          mi.pc_que.buf[mi.pc_que.buf_pnt++]
mi.pc_que.que[mi.pc_que.front];

          if (mi.pc_que.que[mi.pc_que.front] == 2) {
            if (check_pc_buf() == TRUE) {
              interpret_pc_packet();
            }
          }
        }
}

void siu_read()
{
  if (mi.siu_que.front != mi.siu_que.rear) {

    mi.siu_que.front = (mi.siu_que.front + 1) % QUE_SIZE;

    if (mi.siu_que.que[mi.siu_que.front] == 1) {
        mi.siu_que.buf_pnt = 0;
    }
```

```
    mi.siu_que.buf[mi.siu_que.buf_pnt++]=mi.siu_que.que[mi.siu_que.front];
    if (mi.siu_que.que[mi.siu_que.front] == 2)
     {
        if (check_siu_buf() == TRUE)
        {
          interpret_siu_packet();
          mi.siu_cnt = 0;
        }
     }
   }
}
void mi_sw_job()
{
   char buf[50];
   char temp[7];

      if (mi.siu_cnt <= 0)
      {
        switch(mi.sw_job)
        {
          case 6:
            sprintf(buf, P_FORM, 1, mi.siu_id, BEST_PRICE,
 strlen(mi.best_price), mi.best_price, 2);
            txd_siu(buf);
/*          sprintf(buf,".");
            txd_pc(buf);*/
            mi.siu_id = mi.siu_id+1;
            mi.siu_cnt = 1000;
            if (mi.siu_id > 8)
            {
               mi.siu_id = 1;
               mi.sw_job = 6;
            }
            break;
          case 7:
            sprintf(buf, P_FORM, 1, ALL_SIU, REMAIN_TIME, 4, "1234
```

```
2);
                txd_siu(buf);
                mi.siu_cnt = 10;
                mi.sw_job = 8;
                break;

              case 8:
                sprintf(buf, P_FORM, 1, ALL_SIU, OTHER_INFO, 4, "3421
2);
                txd_siu(buf);
                mi.siu_cnt = 10;
                mi.sw_job = 6;
                break;
              case 0:
                sprintf(buf, P_FORM, 1, mi.siu_id*10, INFORM, 4, "WWWW
2);
                txd_siu(buf);
/*              sprintf(buf,".",mi.siu_id);
                txd_pc(buf);*/
                mi.siu_id = mi.siu_id+1;
                mi.siu_cnt = 1000;
                if (mi.siu_id > 8)
                  {  mi.siu_id=1;
                     mi.sw_job = 0;
                  }
                break;
               case 1:
                strncpy(temp,mi.info[mi.siu_id].idpass,6);
                temp[7]=0;
                sprintf(buf, P_FORM, 1, mi.siu_id*10, OTHER_INFO,
 strlen(temp), temp, 2);
                txd_siu(buf);
                mi.siu_id = mi.siu_id+1;
                mi.siu_cnt = 1000;
                if (mi.siu_id > 8)
                {
```

```
                mi.siu_id = 1;
                mi.sw_job = 0;
            }
            break;
          case 2:
            sprintf(buf, P_FORM, 1, mi.siu_id*10, GM_INFO
 strlen(mi.gm_info), mi.gm_info, 2);
            txd_siu(buf);
/*          txd_pc(buf);*/
            mi.siu_id = mi.siu_id+1;
            mi.siu_cnt = 1000;
            if (mi.siu_id > 8)
            {
                mi.sw_job=mi.sw_jobB;
                mi.siu_id=mi.siu_idB;
            }
            break;

        default : break;
      }
    }
    mi.siu_cnt = mi.siu_cnt - 1;

}
```

```
////////////////////////// glcd.c //////////////////////////
/******** 그래픽 LCD 제어 프로그램 ********/


#include <v55.h>
#include <stdio.h>
#include <stdarg.h>
#include <dos.h>
/*#include "bw.h"*/
#include "font.c"

/* typedef   unsigned char  byte; */
#define  MAX_X   159
#define  MAX_Y   127
#define  COLUMN  20

#define   WHITE   0
#define   BLACK   1

#define   CRW       3
#define   ARD       4
#define   AWR       8
#define   SPC       0x40

#define  _SC_Y  20*16

void gotoxy(char,char);
byte          Color;

const int _LCD_Y[15] = {
   0, _SC_Y, 2*_SC_Y, 3*_SC_Y, 4*_SC_Y,5*_SC_Y,6*_SC_Y, 7*_SC_Y,
   8*_SC_Y,9*_SC_Y,10*_SC_Y,11*_SC_Y,12*_SC_Y,13*_SC_Y,14*_SC_Y
};

const char table[3][32] = {
    {  0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,
```

```
      15,16,17,18,19, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
    {  0, 0, 0, 1, 2, 3, 4, 5, 0, 0, 6, 7, 8, 9,10,11,
       0, 0,12,13,14,15,16,17, 0, 0,18,19,20,21, 0, 0 }
    {  0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,
      15,16, 0,17,18,19,20,21,22,23,24,25,26,27, 0, 0 }
};

typedef struct { char i; char j; } _LCD_T ;

union  {
    _LCD_T  c;
      int     i;
} _LCD_ADD;

unsigned char video[3000]={0,};
unsigned char hanimgbuf[32];
unsigned char *src1, *src2, *src3;
unsigned char _LCD_gx,_LCD_gy,_LCD_i,_LCD_j;
unsigned char _LCD_a1 = 0x80;


void LCDC(unsigned char lcdc)
{
    pokeV55reg(P0,0x50);
    pokeV55reg(P4,lcdc);
    pokeV55reg(P0,0xf0);
}

void LCDD(unsigned char lcdd)
{
    pokeV55reg(P0,0x40);
    pokeV55reg(P4,lcdd);
    pokeV55reg(P0,0xf0);
}

unsigned char LCDRD(void)
```

```
{
    unsigned char retval;
    pokeV55reg(PM4,0xff);
    pokeV55reg(P0,0x80);
    retval = peekV55reg(P4);
    pokeV55reg(P0,0xf0);
    pokeV55reg(PM4,0x00);
    return(retval);
}

void STATUS(unsigned char check) {
    pokeV55reg(PM4,0xff);
    pokeV55reg(P0,0x90);
    while(peekV55reg(P4)&check!=check);
    pokeV55reg(P0,0xf0);
    pokeV55reg(PM4,0x00);
}

void data_WR1(byte data,byte command) {
/*      STATUS(CRW);*/
     LCDD(data);
/*    STATUS(CRW);*/
     LCDC(command);
}

void data_WR2(byte data_low,byte data_high,byte command) {
/*      STATUS(CRW);*/
     LCDD(data_low);
     /* STATUS(CRW); */
     LCDD(data_high);
     /* STATUS(CRW); */
     LCDC(command);
}


void aps(byte low,byte high) {
```

```
    /* STATUS(CRW); */
    LCDD(low);
    /* STATUS(CRW); */
    LCDD(high);
    /* STATUS(CRW); */
    LCDC(0x24);
}

void clrlcd(void)
{
   int a;
   aps(0,0);
   for (a=0; a<20*128; a++)
       data_WR1(0,0xc0);
   for(a=0;a<2560;a++) video[a]=0;
   gotoxy(0,0);
}

void lcd_initial(void)
{
    pokeV55reg(PM0,0x0f);
    pokeV55reg(PMC4,0);
    pokeV55reg(PM4,0);

    data_WR2(0,0,0x42);
    data_WR2(20,0,0x43);
    /* STATUS(CRW); */
    LCDC(0x88);
    /* STATUS(CRW); */
    LCDC(0x9b);
    /* STATUS(CRW); */
    LCDC(0xa0);
    clrlcd();
   _LCD_a1 = 0;
   _LCD_gx=0,_LCD_gy=0,_LCD_i=0,_LCD_j=0
}
```

```
void gplot(unsigned char gx, unsigned char gy, unsigned char _color)
{
    int cursor;
    unsigned char temp,temp1;
    cursor = (gy*20) + (gx/8);
    temp=video[cursor];

    /* Set cursor address */
    aps(cursor,cursor>>8);
    temp1=gx % 8;
    temp1=7-temp1;
    temp1=1<<temp1;
    if(_color) {
  data_WR1(temp|temp1,0xc4);
  video[cursor]=(temp|temp1);}
     else {
  data_WR1(temp&(~temp1),0xc4);
  video[cursor]=(temp&~(temp1));}
}

void gotoxy(char x,char y) {
   _LCD_gx = x;  _LCD_gy = y;
   _LCD_ADD.i = _LCD_Y[_LCD_gy]+_LCD_gx;
   aps(_LCD_ADD.i,_LCD_ADD.i>>8);
}

void engput(char b1) {
    src1 =(byte *)EF[b1];
    for(_LCD_i=0;_LCD_i<16;_LCD_i++) {
       _LCD_ADD.i = ((_LCD_Y[_LCD_gy])+_LCD_gx) + COLUMN *
 _LCD_i;
       aps(_LCD_ADD.c.i,_LCD_ADD.c.j);
       if(Color == BLACK) data_WR1(*src1,0xc4);
    else data_WR1(~(*src1),0xc4);
```

```
        src1++;
    }
    _LCD_gx++;
}

void hanput(unsigned char b2)
{
   unsigned char first,mid,last,ii=0;
   first = (_LCD_a1 & 124) >> 2;
   mid   = ((_LCD_a1 & 3) * 8) + (b2 >> 5);
   last  = (b2 & 31);
   first = table[0][first];
   mid   = table[1][mid];
   last  = table[2][last];

   if ( mid==1||mid==3||mid==10 )     _LCD_j = 0;
   else if ( mid==9||mid==13||mid==14||mid==18||mid==19 )_LCD_j = 3;
   else  if ( mid==2||mid==4||mid==6||mid==8||mid==11||mid==16)  _LCD_j
2;
   else if ( mid==9||mid==13||mid==14||mid==18||mid==19)  _LCD_j = 3;
   else  _LCD_j = 1;

   if (first==1 || first==16 ) b2 = (last==0) ? 0 : 2;
   else  b2 = (last==0) ? 1 : 3;
   if (mid==14||mid==18 )   _LCD_i = (last==0) ? 2 : 6;
   else if (mid==9||mid==13||mid==19)  _LCD_i = (last==0) ? 1 : 6;
   else if (mid==15||mid==16||mid==17 )  _LCD_i = (last==0) ? 4 : 7;
   else if (mid==10||mid==11||mid==12||mid==20 ) _LCD_i = (last==0) ? 3
7;
   else  _LCD_i = (last==0) ? 0 : 5;
   src1 = (byte *)HF1[_LCD_i][first];
   src2 = (byte *)HF2[b2][mid];
   src3 = (byte *)HF3[_LCD_j][last];
   for (_LCD_i=0;_LCD_i<32;_LCD_i++)
      hanimgbuf[_LCD_j] = *src1++ | *src2++ | *src3++;
```

```
    for (_LCD_i=0; _LCD_i<32; _LCD_i+=2) {
     _LCD_ADD.i = ((_LCD_Y[_LCD_gy])+_LCD_gx) + COLUMN * ii++
     aps(_LCD_ADD.c.i,_LCD_ADD.c.j);
     if(Color == BLACK) data_WR1(hanimgbuf[_LCD_i],0xc4);
     else data_WR1(~(hanimgbuf[_LCD_i]),0xc4);
    }
    _LCD_gx++;
    ii=0;
    for (_LCD_i=1; _LCD_i<32; _LCD_i+=2) {
     _LCD_ADD.i = ((_LCD_Y[_LCD_gy])+_LCD_gx) + COLUMN * ii++
     aps(_LCD_ADD.c.i,_LCD_ADD.c.j);
     if(Color == BLACK) data_WR1(hanimgbuf[_LCD_i],0xc4);
    else data_WR1(~(hanimgbuf[_LCD_i]),0xc4);
    }
    _LCD_gx++;
}


void lcdputch(int b1)
{
    if (_LCD_a1==0x80)  lcd_initial();
    if (_LCD_a1) { hanput(b1);_LCD_a1 = 0;}
    else {
        if (b1>0x80) _LCD_a1 = b1;
        else {
        if ((b1=='\n') || (b1=='\r')) gotoxy(0,_LCD_gy+1);
        else  engput(b1);
        }
    }
}

void L_outstr(unsigned char *msg)
{
     int i;
     pokeV55reg(PM0,0x0f);
     for(i=0;msg[i];i++) lcdputch(msg[i]);
```

```
}


void L_printf(char *form, ... )
{
  char buff[128];
  va_list  argptr;
  va_start( argptr, form );
  vsprintf(buff, form, argptr );
  L_outstr(buff);
  va_end( argptr );
}


void symmetry8(int xx,int yy,int x,int y,char color)

void circle(int xx,int yy,int radius,char color)
{
    int y=radius;
    int d=3-y*2;
    int x=0;

    while (x<y) {
        gplot(xx+x,yy+y,color);
                symmetry8(xx,yy,x,y,color);
        if (d<0) d+=(x*4+6);
        else {
            d+=((x-y)*4+10);
            y--;
        }
        x++;
    }
        if (x==y) gplot(xx+x,yy+y,color);
}

void symmetry8(int xx,int yy,int x,int y,char color)
```

```
{
    gplot(xx-x,yy+y,color);
    gplot(xx+x,yy-y,color);
    gplot(xx-x,yy-y,color);
    gplot(xx-y,yy+x,color);
    gplot(xx+y,yy-x,color);
    gplot(xx-y,yy-x,color);
    gplot(xx+y,yy+x,color);
}

void line (int sx,int sy,int ex,int ey,unsigned char _color){
    int x=0,y=0,buf_x,buf_y,incy;
    int np;  /* //np:number of byte in same y law*/
    register int t,delta_x,delta_y;
    delta_x=ex-sx;
    delta_y=ey-sy;
    if (delta_x<0){
        buf_x=sx;buf_y=sy;sx=ex;sy=ey;ex=buf_x;ey=buf_y;delta_y*=-1
    }
    if (delta_y<0) incy=-1;
        else incy=1;
    delta_x=abs(delta_x);
    delta_y=abs(delta_y);
        if (delta_x<delta_y)
                    {
                    x=delta_x+delta_y/2;
                    for (t=1;t<=delta_y+1;t++){
                        x+=delta_x;
                        gplot(sx,sy,_color);
                        sy+=incy;
                        if (x>delta_y){
                            x-=delta_y;
                            sx++;
                        }
                    }
            }
```

```
            else{
                y=delta_y+delta_x/2;
                for (t=1;t<=delta_x+1;t++){
                    y+=delta_y;
                    gplot(sx,sy,_color);
                    sx++;
                    if (y>delta_x){
                        y-=delta_x;
                        sy+=incy;
                        }
                 }
             }
}
void box(int x1,int y1,int x2,int y2,unsigned char color)
{
        line(x1,y1,x1,y2,color);
        line(x1,y1,x2,y1,color);
        line(x2,y1,x2,y2,color);
        line(x1,y2,x2,y2,color);
}

void prn_xy(int x,int y,char *msg)
{
   gotoxy(x,y);
   L_printf("%s",msg);
}

void clr_line(int line)
{
   gotoxy(0,line);
   L_printf("                          ");
   gotoxy(0,line);
}
void prn_msg(int tt,char *msg)
{
   if (tt==0) {
```

```
        clr_line(6);
        prn_xy(0,6,msg);
    } else if (tt==1) {
        clr_line(7);
        prn_xy(0,7,msg);
    }
}
```