최 종 보 고 서

다동온실내 복합환경의 컴퓨터 제어시스템 개발

Development of a Computer Hybrid Environmental Control System in the Multi-Greenhouses

연 구 기 관
서 울 대 학 교

농 림 부

제 출 문

농림부 장관 귀하

본 보고서를 “다동온실내 복합환경의 컴퓨터 제어시스템 개발”에 관한 연구 과제의 최종 보고서로 제출합니다.

1996. 12. 15.

주관연구기관명 : 서 울 대 학 교
농업생명과학대학

총괄연구책임자 : 고 학 균

연 구 원 : 권영삼(농촌진흥청 원예연구소)
최창현(성균관대학교)
이대원(성균관대학교)
김재민(성균관대학교)
권지선(농촌진흥청 원예연구소)

요 약 문

Ⅰ. 제목

다동온실내 복합환경의 컴퓨터 제어시스템 개발

Ⅱ. 연구개발의 목적 및 중요성

국내에서 많은 회사가 온실용 환경제어시스템을 개발하고 있지만, 복합환경제어 시스템이 아닌 ON/OFF 제어만 하고 있는 실정이다. 그러나 온실환경과 작물 생장을 통한 생체정보를 동시에 측정하여 온실의 환경을 제어하는 복합환경 제어시스템은 보다 정밀한 온실환경을 제공해 준다. 특히 중소기업의 경우 많은 데이터를 분석한 후 최적화를 위한 프로그램을 개발하기에는 인적 자원과 많은 자본을 투자하기 불가능한 것으로 여겨진다. 따라서 본 연구는 개발된 최적화를 위한 복합환경제어 시스템의 하드웨어를 개발하여 시설재배 농가에 보급하기 위하여 수행하였다.

Ⅲ. 연구개발 내용 및 범위

본 연구에서는 환경요인 측정센서와 생체정보 측정센서를 이용하여 온실의 환경요인 및 작물의 생체정보를 측정하였다. 온실 내의 환경요인을 측정하기 위하여 온도, 습도, 일사량 및 CO2 농도 센서를 2개 동의 온실에

설치하였으며, 외부 기상조건을 측정하기 위하여 일사량, 온도, 습도, 풍향, 풍속 및 강우감지 센서 등으로 구성된 외부 기상관측장치를 지상 5m에 설치하였다.

국내에서는 처음으로 생체정보 측정센서를 이용하여 작물의 생육상태를 측정하였으며, 수집된 자료를 활용하여 작물의 생육상태를 고려하여 데이터로거를 이용한 복합환경 제어시스템을 개발하였다. 2개 동의 온실에는 증산류(stem water flux) 측정센서, 줄기직경(stem diameter) 측정센서, 과일직경(fruit diameter) 측정센서, 엽온(leaf temperature) 측정센서 등 4가지의 생체정보 측정용 센서를 설치하였다.

본 연구를 위하여 설계된 제어시스템은 소형 데이터로거(datalogger)를 이용하여 자료를 수집하고 온실의 작동장치를 제어하고 있다. 온실에 설치된 모든 센서와 작동장치는 데이터로거를 통하여 마이크로 컴퓨터에 연결되어 있다.

농촌진흥청 원예연구소에 설치된 2동의 온실에서 오이와 토마토를 재배하였으며, 데이터 로거 수집장치와 컴퓨터 제어시스템은 서울대학교와 성균관대학교에서 개발하였다.

데이터 로거를 이용한 복합환경 제어시스템의 개발은 시설재배에 관한 연구와 컴퓨터 제어시스템에 관한 연구가 병행되어야 한다. 따라서, 본 연구에서는 최적화를 위한 복합환경 제어프로그램을 개발하기 위하여 환경의 변화와 작물의 생체정보를 측정하여 분석하였다. 복합환경 제어프로그램의 환경측정 자료와 생체정보 자료의 분석은 통계프로그램 SAS를 이용하였으며, 복합환경 제어프로그램에는 농촌진흥청 원예연구소의 시설재배와 작물의 생육에 관한 전문가의 조언을 반영하였다.

Ⅳ. 연구개발 결과 및 활용에 대한 건의

1. 연구결과

국내에서 많은 회사가 온실용 환경제어시스템을 개발하고 있지만, 복합환경 제어시스템이 아닌 온실 작동기의 ON/OFF 제어만 하고 있는 실정이다. 특히, 중소기업이 많은 데이터를 분석한 후 최적화를 위한 프로그램을 개발하기 위하여 인적 자원과 경제적 자본을 투자하기에는 매우 어렵다.

따라서, 본 연구는 복합환경을 제어하기 위한 하드웨어를 개발하기 위하여 수행하였으며, 온실의 복합환경을 제어하기 위하여 실내·외 환경요인을 측정하기 위한 센서와 작물의 생체정보를 수집하기 위한 센서를 설치하였다.

연구결과를 요약하면 다음과 같다.

1. 국내에 설치된 온실의 방문 및 자료수집에 의하여 온실의 현황 및 기술 수준을 분석하였으며, 외국의 온실 현황은 자료를 수집하여 분석하였다.
2. 농촌진흥청 원예연구소에 20평 규모의 실험온실 2동을 확보 (FRP 온실, PET 온실)하였다. 온실 재배용 오이와 토마토의 정식을 위하여 각 온실에 폭 0.5m, 두께 0.025m, 길이 1.2m의 폴리스티렌 베드 7장을 연결하여 폭 0.5m, 깊이 0.2m, 길이 8.4m의 성형 베드를 각 온실마다 3개씩 설치하였다.
3. 각 온실에는 환경요인의 제어를 위하여 측창 개폐장치, 천창 개폐장치, 커튼 개폐장치, 환기장치 등의 온실작동기와 온실작동기의 수동조작 및 자동제어를 위한 동력제어반 설치하였다. 각 온실에 양액을 자동 공급하기 위하여 양액 공급시스템을 설치하였다.

4. 온실의 실내·외 환경요인은 일사량 센서, 실내 온·습도 센서, 외부 온·습도 센서, 풍향센서 , 풍속센서, 강우량 센서를 설치하여 측정하였다. 작물의 생육정보를 측정하기 위하여 줄기직경(stem diameter) 센서, 증산류(stem water flux) 센서, 엽온(leaf temperature) 센서, 과일직경(fruit diameter) 센서와 양액제어를 위하여 pH 센서, EC센서를 설치하였다. 측정된 환경요인과 작물의 생육정보를 디지탈화하기 위하여 중간변환장치를 설치하였으며, 온실 작동기의 제어를 위하여 콘트롤러를 설치하였다.

5. 생체정보 이용 온실 환경관리를 위한 자료분석에서 증산류를 지표로 한 모델은 특히, 작물의 수분관리에 유용할 것으로 판단되었다.

6. 과경과 경경을 지표로 한 모델의 구성을 위해서는 보다 감도가 높은 센서의 이용이 필요했으며, 이들 두 지표의 지연적인 생장과 환경요인들과의 변화 반응시간을 일치시키면 모델 구성이 가능할 것으로 판단되었다.

7. 보다 완벽한 작물 생장모델을 구축하기 위해서는 환경조절이 가능한 그로스챔버를 이용한 실험이 요구되었다.

8. 복합환경의 제어와 하드웨어간의 통신을 위하여 컴퓨터 프로그램은 작성하였다. 프로그램 작성에는 Windows용 Visual Basic, Turbo-C, Assembler어를 사용하였다. 복합환경 제어용 프로그램에는 온실 내·외부 환경요인 및 작물 생육정보 수집부, 환경요인 및 생육정보에 의한 작동기 제어부로 구성하였다.

9. 본 연구에서 개발한 복합환경 제어시스템의 하드웨어와 소프트웨어는 기존의 시스템에 비하여 크기가 작고 개수 및 사용범위, 용도등이 가변적이고 유동적이어서 제어할 온실 및 기타 유사 시설물의 크기와 종류, 개수, 특징에 따라서 시스템을 쉽게 변경하여 사용할 수 있도록 하였다. 또한, 본 시스템은 크기가 작고 사용되는 환경에 따라서 Controller의 개수가 조절이 되기 때문에 가격면에서도 상당히 합리적

이고 경제적일 것으로 사료된다.

10. 하드웨어의 각 입출력부분 및 구동 소프트웨어가 MODEM 및 RS-232, RS-485 통신을 통하여 PC에서부터 전송시켜 줄 수 있기 때문에 사용자가 원할 경우에는 시스템의 규모를 축소 또는 확장시킬 수 있도록 하였다. 또한, 프로그램 및 입출력단자들의 정보를 추가, 삭제 및 수정을 쉽게 할 수가 있도록 구성하였다.
11. 본 연구에서 개발된 복합환경 제어시스템에는 무정전 전원장치를 포함시켜 정전 등에 의하여 전원이 차단될 경우에도 온실의 작동기기를 제어할 수 있도록 하였다.

2. 활용계획

완전자동 컴퓨터제어시스템은 첨단농업을 위한 기초기술을 제공할 것으로 기대되며, 궁극적으로 농업의 생력화, 생산성 및 품질의 향상을 가속화시킬 것이다. 본 연구의 결과는 다음과 같은 개발사업에 응용될 것으로 기대된다.

- 시설원예 환경의 작목별, 지역별 제어 알고리즘 개발 촉진
- 환경요인 및 생체정보를 수집하므로써 새로운 작물의 생장모델 개발에 필요한 기초자료 제공
- 시설원예 관련설비 및 운영 프로그램의 국산화에 필요한 기초기술 제공
- 작물의 성장속도를 고려하여 환경을 제어하므로써 수확시기 및 수확량 조절 가능
- 복합환경 제어시스템의 개발로 시설원예 농가의 경영혁신 및 소득증대 기대
- 시설원예 농민을 위한 첨단기술 교육시설로 활용

SUMMARY

Ⅰ. Title of Research

Development of a Computer Hybrid Environmental Control System in the Multi-Greenhouses

Ⅱ. Conclusions of Research

Growth of plants in greenhouses is increasing in popularity across Korea. Automatic hybrid environmental control technology, including main module, display module, sensor module, input module, output module with one chip CPU controller, EPROM, SRAM, multi task interrupt controller, operating equipment and many sensors, appears to be an attractive alternative to the use of manual labor for accomplishing this task.

However, on/off control systems are most utilized in protected cultivation, but a hybrid environmental control system is not yet. Even though a lot of small company tries to develope a hybrid environmental control system, it is difficult for the company to produce this control system, because a lot of time and money are to be spent to develope the optimal growth programs of plant in the greenhouse. In order to

develope them, many data need to be obtained, analyzed and estimated for very long time.

This research developed both software and hardware for this system. Separate programs for control of the hardware and for optimal analysis of the software, written Visual Basic, Turbo-c and Assembler, are developed to make the automatic hybrid environmental control system operated. Especially, hardware operated out of the electric power by using the UPS with battery. The objective of this research designed and constructed a hybrid environmental control system for the optimal growth of plants within a greenhouse environment.

Based on the results of this research the following conclusions were made:

1. Data of environment factor, such as temperature, humidity, solar irradiate pH and EC were measured and analyzed in the two greenhouse for the optimal growth of both cucumber and tomato.

2. Data of plant growing informative factor, such as leaf temperature, stem flux relative rate, stem diameter variations, stem height increment and fruit increment were measured and analyzed.

3. The automatic hybrid environmental control system, designed and constructed for this project, was adequate for reading data from

many sensors and operating side windows, top windows, curtains and fans in the two greenhouses.

4. The UPS with battery made the automatic hybrid environmental control system operate well out of the electric power. Therefore it is possible for this system to be operated for a limited time under out of the electric power, which would be often happened in Korea.

5. After data were obtained from environment factors and informative factors, they were optimized by analysis of statistics. However, general optimal growth model was not developed, only the model of stem flux model for watering of plant. Because it was not easy to find the plant growth response in time from the hybrid environmental control system.

CONTENTS

목 차

표 목 차

그 림 목 차

제 1 장
서 론

제 1 절 연구배경

WTO체제하에서 농산물의 수입 개방시대를 맞이한 우리나라의 농업은 노동력의 감소와 경제수준의 향상에 따른 식생활 형태의 변화 등으로 인하여 상당한 어려움에 직면해 있다. 특히, 시설재배는 노지재배보다 평균 3 6배에 달하는 550 1,200hr/10a의 노동력이 투입되어야 한다. 이러한 국내외적인 어려움에 처한 상황에서, 국가 기간산업으로서의 농업을 육성시키기 위해서는 토지와 노동력에 의존하던 전형적인 농업의 형태에서 탈피하여 기계화, 자동화된 기술집약형 농업으로 발전하여야 한다.

현재 우리나라에서는 기술집약형 농업의 대표적인 분야로서 온실에서 화훼 및 과채류 등을 집약적으로 생산하는 시설재배가 제시되었으며, 매년 시설재배면적은 증가하고 있는 추세이다. 우리나라의 시설재배 산업은 노동인구의 감소와 노동비용이 증가함에 따라 재래식 온실에서부터 점차 자동화된 온실로의 시설변화가 요구되고 있다. 때맞게 새로운 첨단시설이 국내에 설치되고 있으나 거의 외국, 특히 네덜란드에서 모든 자재, 소프트웨어, 양액, 암면배지까지 전량 수입하여 시공까지 맡기는 실정이다. 이것은 우리의 원예시설 기술수준이 그동안 비닐하우스 중심으로 국내 수요에 부응하는 생산방식에 만족하여 고도의 자본 기술집약 시설에 대한 개발을 등한시하여 왔기 때문이다. 특히, 시설의 환경관리 측면에서 본다면 온실

자동화 시스템의 하드웨어와 소프트웨어의 개발이 미흡하고, 온실자동화에 대한 기초연구의 부족과 환경의 변화에 대한 작물의 반응을 분석하기 위한 자료의 부족 등으로 나타난다.

하드웨어는 생산업자가 성능이 부족한 장치를 무리하게 공급하여 수명을 단축시키고 있다. 측창 및 천창 개폐기, 탄산가스발생기, 온풍 및 난방기, 관수장치 등 온실의 황경제어 및 재배관리용 장치들은 한시적 소모품이 아닌 내구성자재이어야 한다. 그러나 대부분의 장치들은 3년정도를 넘기지 못하고 마모되어 사용할 수 없게 되는 것이 오늘의 현실이다.

온실 환경제어용 소프트웨어의 경우는 온실의 환경제어를 산업용 공조시스템과 동일하게 고려하여 프로그램을 개발하는 경향이 있다. 따라서, 작물을 생육하는 온실에는 적응성이 결여된 프로그램이 개발되어 대농민 신뢰도가 상당히 떨어져 있다. 농민들은 온실제어용 프로그램을 이용하기 보다는 경험과 직관에 의하여 온실환경을 제어하거나, 단순히 제어시간을 설정하여 모든 하드웨어를 ON/OFF제어만을 실행하는 경우가 대부분이다.

국내의 온실경영에서는 난방, 급수, 환기, 탄산가스, 시비, 양액공급의 자동화 요구가 높다. 따라서 제어시스템 개발은 지역별, 작물별, 온실구조별, 제어방법별로 체계화된 연구가 필요하다. 체계화된 연구는 크게 두가지로서, 작물성장을 통한 생체정보의 측정과 측정자료를 사용하여 온실환경을 제어하는 시스템이다.

고도화된 작물 생산시설에 있어서 가장 필수적인 요건 중의 하나가 복합환경의 자동제어 시스템이다. 초기의 환경제어 시스템은 태양광을 이용하는 온실에 국한되어 있었으며, 외부의 극단적인 환경의 변화에 대하여 온실 내부의 광강도, 온도 및 습도를 일정 수준까지 상승시켜 작물의 생육을 촉진시키는 정도의 수준이었다. 이러한 온실제어방법들은 각 환경요소를 독립적으로 측정, 제어하도록 설계되어 개별 단위 장치의 조합적인 형태를

유지하고 있다. 이와 같이 단위장치를 제어하는 것은 복합환경 제어의 개념과는 구별된다. 기존의 환경제어 방법들이 온실 운영상 최적의 경제성을 제공할 수 없었기 때문에 작물의 생체정보를 정량화하여 온실 환경제어에 이용하므로서 가장 안락한 온실 환경을 구현하기 위한 연구가 시도되었다. 그 결과 생체정보를 이용하여 온실의 환경을 복합적으로 제어하는 방법에 'speaking plant'라는 개념이 도입되었다. 이후 온실 환경제어시 환경요인을 측정하여 온실내부의 환경을 결정하는 방법에서 온실 환경 및 작물의 생체정보를 동시에 측정하여 온실의 환경을 제어하는 진정한 의미의 복합환경 제어시스템에 관한 연구가 활발히 진행되고 있다. 특히, 이러한 연구들은 논리적 결정방법에 대한 새로운 이론의 도입과 컴퓨터 기술의 발전 및 보편화를 통하여 복합환경의 최적제어에 관한 연구가 더욱 발전하고 있다.

그러나 지금까지 다양한 연구를 통하여 개발되었던 복합환경의 최적제어 알고리즘이 실제 시설에 적용된 예는 많지 않은 것이 사실이다. 그 원인은 현재까지 제시된 방법들이 환경의 변화에 대한 작물의 생육반응이라는 생물학적 관점과 온실내 복합환경의 자동제어라는 공학적 관점 사이에서 충분한 접근이 이루어지지 못했다고 할 수 있다.

온실 환경의 제어시 고밀도 환경제어가 뒷받침된다면 작물의 생육상태를 관찰하여 특정 환경하에서 단기, 중기, 장기적 생체반응을 조사, 분석할 수 있다. 특히 작물반응 데이터를 가지고 작물의 생장모델과 온실환경을 개발할 수 있으며, 이 모델을 사용하여 기상의 변화에 따라 수확량을 예측하거나, 반대로 원하는 수확량이나 수확시기에 맞추어 온실환경을 조절할 수 있게 된다.

이러한 고정밀 환경을 제공하기 위해서는 온실제어용 하드웨어와 소프트웨어의 개발에 관한 종합적 연구가 필요하다. 특히 소프트웨어의 개발은

온실환경과 기상자료의 온라인 화면제시 및 제어, 정보데이타의 이동 및 교류, 경제분석 프로그램과의 연결, 다동 온실 운영시 종합경영 체제를 목표로 하는 기초 자료가 될 것이다. 또한 축적된 기술은 곧바로 실제 시설원예의 대규모 환경제어 시스템의 실용화와 연결될 수 있다

제 2 절 연구 목적

국내에서 많은 회사가 온실용 환경제어시스템을 개발하고 있지만, 복합환경제어시스템이 아닌 온실 작동기의 ON/OFF 제어만 하고 있는 실정이다. 특히, 중소기업이 많은 데이터를 분석한 후 최적화를 위한 하드웨어의 구성과 제어프로그램을 개발하기 위하여 인적 자원과 경제적 자본을 투자하기에는 매우 어렵다.

따라서, 본 연구는 작물의 생육에 적합한 최적의 환경을 제공하기 위한 복합환경 제어시스템의 하드웨어를 개발하기 위하여 수행되었으며, 구체적인 연구목적은 다음과 같다.

- 다동온실 환경 제어시스템의 자료 수집 및 분석
- 데이터로거를 이용한 복합환경 제어용 하드웨어 개발
- 다동온실의 복합환경 제어를 위한 제어프로그램 개발

이 시스템을 사용하여 얻을 수 있는 자료를 열거하면 다음과 같다.

- 작물에 따른 피복재의 적정 광투과량 선택
- 증산작용으로 생긴 공기내 수분량을 산출하여 적정 환기방법 개발, 관수이용 효과 분석
- 광합성, 호흡, 증산량 등을 주어진 제어조건에 따라 정량적으로 분석하며, 자당의 열매, 뿌리, 잎으로의 이동 및 저장량을 산출하여 그 경제성분석 연구
- CO2 사용 적정시기, 적정량을 산출하여 그 경제성분석 연구

- 난방, 환풍 부하를 계절과 작물 성장단계에 따라 산출
- 온실환경의 여러 가지 제어방식의 비교 연구 기능 즉, 매일 미리 입력된 제어프로그램 방식, 환경의 순간적 변경에 따른 즉각제어 방식, 작물성장기에 따른 제어 방식, 또는 낮의 기상에 따른 야간제어 방식 등을 체계적으로 연구할 수 있다.

제 3 절 연구내용 및 방법

본 연구에서는 환경요인 측정센서와 생체정보 측정센서를 이용하여 온실의 환경요인 및 작물의 생체정보를 측정하였다. 온실 내의 환경요인을 측정하기 위해서 온도, 습도, 일사량 및 CO2 농도 센서를 2개 동의 온실에 설치하였으며, 외부 기상조건을 측정하기 위하여 일사량, 온도, 습도, 풍향, 풍속 및 강우감지 센서 등으로 구성된 외부 기상관측장치를 지상 5m에 설치하였다.

국내에서는 처음으로 생체정보 측정센서를 이용하여 작물의 생육상태를 측정하였으며, 수집된 자료를 활용하여 작물의 생육상태를 고려하여 데이터로거를 이용한 복합환경 제어시스템을 개발하였다. 2개 동의 온실에 설치된 생체정보 측정센서는 증산류(stem water flux) 측정센서, 줄기직경(stem diameter) 측정센서, 과일직경(fruit diameter) 측정센서, 엽온(leaf temperature) 측정센서가 있다.

본 연구를 위하여 설계된 제어시스템은 소형 데이터로거(datalogger)를 이용하여 자료를 수집하고 온실의 작동장치를 제어하고 있다. 온실에 설치된 모든 센서와 작동장치는 데이터로거를 통하여 마이크로 컴퓨터에 연결되어 있다.

현재 국내에서 개발된 대부분의 제어시스템은 컴퓨터를 항상 작동시켜야 시스템이 원할히 동작한다. 그러나, 본 연구에서 개발된 제어시스템은 센서로부터 측정된 자료의 보관 및 테이터로거의 프로그램 수정에 이용되고 있으므로 정전, 컴퓨터의 고장 등으로 컴퓨터가 작동하지 않아도 데이터로거에 저장된 프로그램에 의하여 온실의 환경을 제어할 수 있다.

농촌진흥청 원예연구소에 설치된 2동의 온실에서 과채류를 재배하였으며, 데이터 로거 수집장치와 컴퓨터 제어시스템은 서울대학교와 성균관대

학교에서 개발하였다.

데이터 로거를 이용한 복합환경 제어시스템의 개발은 시설재배에 관한 연구와 컴퓨터 제어시스템에 관한 연구가 병행되어야 한다. 따라서, 본 연구에서는 최적화를 위한 복합환경 제어프로그램을 개발하기 위하여 환경의 변화와 작물의 생체정보를 측정하여 분석하였다. 복합환경 제어프로그램의 환경측정 자료와 생체정보 자료의 분석은 통계프로그램 SAS를 이용하였으며, 복합환경 제어프로그램에는 농촌진흥청 원예연구소의 시설재배와 작물의 생육에 관한 전문가의 조언을 반영하였다.

제 2 장
국내외 원예시설 현황 분석

제 1 절 서 언

우리나라 시설원예는 재배면적면에서 본다면 일본 다음으로 세계에서 두 번째이다. 따라서 양적인 수준 면에서 우리 나라 시설원예의 규모는 결코 작지 않음을 알 수 있다. 그러나 질적인 면에서 그 수준을 두 가지 각도에서 평가해 보면 다음과 같다.

첫째, 자본의 집약도 면에서 시설이 고급화된 유리온실 면적 비율을 검토해 보면 우리 나라는 주로 비닐하우스이고 유리온실은 이제 시작하는 단계로 비교적 하위에 속하는 스페인과 비슷한 수준이다. 앞으로 우리 나라도 유리온실 면적이 크게 증가할 전망이지만 1988년도의 외국과의 비교로도 네덜란드, 프랑스, 벨지움, 독일 등 유럽의 소위 자본•기술 집약 시설원예와는 상당한 거리가 있음을 알 수 있다.

표 2-1. 국가별 유리온실 및 비닐하우스 시설면적 (1988) (단위 : ha)

시설면적	프랑스	이태리	스페인	화 란	이스라엘	일 본	한 국*
유리온실	2,000	1,600	50	8,900	-	2,074	152
비닐하우스	4,000	15,400	12,100	100	1,380	46,737	44,500
계	6,000	17,000	12,150	9,000	1,380	48,811	44,652

* 한국은 '95년 통계임

둘째, 조절농업의 척도로 평가되고 있는 가온 재배면적 비율인데, 우리나라의 '93년 현재 난방기에 의한 가온 재배면적이 꽃 시설재배를 포함하면 약 17%정도 차지하고 있어 어느 정도 조절농업쪽으로 이행되어 있다고 할 수 있다. 그러나 우리나라의 현 상황은 유럽의 시설원예 국가와 비교하면 떨어지며 이탈리아, 그리이스 정도의 난방 보급 수준과 비슷하며 일본의 38% 난방 면적 비율보다도 떨어진다. 물론, 유리온실 보급 면적이나 난방기 보급 면적 자체가 그 나라 시설원예의 객관적인 기술 수준을 평가하는 척도라고는 할 수 없으나 사용되는 환경 조절 장치의 종류나 기능 면에서 볼 때 우리의 기술 수준은 시설원예 선진국에 비해 분명히 한 수 아래이다. 또한 시설면적중 고정시설 면적은 48.1%로 전체의 50%에도 미달하여 주년이용형 시설보다는 답리작 등의 단기이용형 시설이 아직 다수를 차지하고 있다. 따라서 시설의 구조 장치 등에 대한 투자수준이 미흡할 수밖에 없는 형편이다.

우리 나라 시설원예는 그 동안 짧은 기간 동안에 장족의 발전을 거듭하여 1995년 현재 채소 재배 시설면적 40,076ha, 화훼 시설면적 3,054ha로서 전체 면적은 44,652ha에 달하고 있다. 이는 국민 1인당 시설 설치 면적으로 볼 때 8m2 이상이 되며 이는 일본의 4m2보다 2배가 높다.

이와 같이 우리 나라 시설원예가 급격히 신장하게 된 배경은 무엇보다 국민소득 증가에 따른 수요의 증가가 뒷받침되어 왔고, 쌀 농사 중심의 농촌에서 농한기 유휴노동력을 이용할 마땅한 대상이 없었기 때문에 비닐을 이용한 하우스의 설치로 제 철이 아닌 시기에 원예작물을 생산하는 재배형태가 등장하게 된 것이다.

시설원예 초기 단계에서는 어느 나라이든 자연 에너지인 태양열을 이용하므로써 낮에는 햇빛을 하우스 내부로 수용하여 땅속에 열을 저장하였다가 야간에는 보온이라는 수단으로 방열을 최대한 억제시켜 작물에 필요한 온도를 유지하는 이른바 자연 기후 의존형인 수동적 시설원예(Passive

type Protected Horticulture)방식을 선택하기 마련이다.

그러나 수동적 시설원예는 재배 시기의 제한과 재배 환경의 열악으로 품질이 나빠지고 생산 시기가 집중되어, 시설농가가 늘어난 80년대에 들어와서는 노지에서 제 철에 생산한 농산물 가격보다 오히려 떨어지는 기현상까지 나타나게 되었다.

이러한 생산 시기의 집중과 품질 불량, 농촌 인구의 감소에 따른 노동력 부족으로 지금까지 자연 기후에 의존하여 무가온 보온 재배는 필연적으로 보다 적극적인 재배 환경을 조절할 수 있는 환결조절형 시설원예로 나아가지 않을 수 없게 되었다.

최근 WTO 출범에 따른 농산물의 세계화 추세에 따라 품질이나 가격 면에서 국제 경쟁력을 갖추어야 하고, 소비자는 과거와는 달리 품질 선택 면에서 까다로워지고 다양한 품목을 년중 찾게 됨에 따라 지금의 농촌 생산 시설로는 이러한 국내외 여건 변화를 수용하기에는 역부족이기 때문에 시설 현대화를 통해 재배 환경의 개선과 생력자동화 농업으로 전환하여 과거 노동집약 수동적 시설원예에서 기술 집약 적극적 환경 조절 시설원예로 목표 설정을 하기에 이르렀다.

제 2 절 국내 원예시설의 현황

1. 시설원예의 현황

우리나라 시설원예 면적은 '95년 현재 44,652ha로 이중 비닐 하우스 면적이 44,500ha로 99.66%로 절대다수를 차지하고 있다. '91년부터 정부지원사업으로 자동화하우스, 경질판온실, 유리온실이 보급되어 국내 시설원예의 기술 수준 향상과 품질 및 가격 경쟁력을 높이고 있는 실정이다.

그러나 현재 시설면적의 대부분을 차지하고 있는 비닐하우스는 재배관리의 생력화 정도가 낮고 노동 집약적 생산 체계이며 광, 온도, 습도 환경이 불리하여 품질 및 생산성이 떨어지는 실정이다.

이러한 재배환경상의 불리한 조건 때문에 비닐하우스에서는 경쟁력있는 고품질 생산물을 얻기 어려우므로 외국 생산물과 경쟁할 수 있는 생산체제 도입이 불가피한 실정이다.

따라서 시설원예에 대한 전반적인 기술의 선진화를 위해서는 보다 집약적인 시설인 유리온실이나 경질판의 도입이 꼭 필요하다고 판단된다. 반면 내수위주 비닐하우스 재배의 생산체계도 기본적으로 생력화나 재배관리의 효율성면에서 보다 경제적인 최소한의 구조개선이 필요한 실정이다. 또한 국내에 반입될 가능성이 큰 외국 생산물의 가격 경쟁력 때문에 보다 적은 생산비로 양질의 원예작물을 생산할 수 있는 간이시설 재배방법도 선택하지 않을 수 없어 이른바 다원화된 시설원예 생산 방법이 필요한 실정이다.

가. 시설재배 작물별 면적

우리나라의 채소류의 시설재배 면적은 1980년 7,142ha를 기점으로 설치면적이 급진적으로 증가하여 현재 40,076ha의 시설면적을 가지고 있다. 화

훼류 재배면적은 1980년도부터 연간 27.0%가 증가하여 많은 면적이 확대되어 가고 있으며 '95년에는 3,054ha의 면적으로 전체 화훼재배면적의 59.2%를 차지하고 있다(표 2-2). 또한 1980년 이후 전체 면적에 대한 시설화훼 면적이 늘어나 1995년 현재 거의 60%에 육박하여 시설화훼가 차지하는 비중이 매년 높아지고 있다.

나. 시설의 낙후성

채소시설의 경우 시설면적은 '95년말 현재 40,076ha중 밭이 9.5천ha(24%), 논이 30.6천ha(76%)의 시설을 점유하고 있다(표 2-3). 그 중 고정시설면적은 19,276ha(48.0%), 밭 6,721ha(17%), 논이 12,555ha(31%)로 되어 고정시설이 늘어나고 있으나 대형터널이나 아치형으로된 파이프 하우스가 대부분으로 철골로된 지붕형 온실은 보급 초기단계로 시설의 낙후성을 나타내고 있다(표 2-4).

시설재배시 난방방법은 무가온 보온재배 농가가 대부분이고 조절농업의 판단척도가 되는 가온재배면적이 불과 17%에 머무르고 있다(표 2-5).

표 2-2. 시설재배 면적 변화

년도	채소		B/A (%)	화훼		D/C (%)
	전체노지 (A)	시설 (B)		전체(C)	시설(D)	
1970	255,041	763	0.2	-	-	-
1980	368,029	7,142	1.9	1,280	180	14.1
1990	300,298	23,688	7.8	3,503	1,752	50.9
1995	403,000	40,076	9.9	5,156	3,054	59.2

표 2-3. 설치방법별 시설면적 (단위 : ha)

구 분	계			고 정 시 설			이 동 시 설		
	계	전	답	계	전	답	계	전	답
면 적	40,076	9,513	30,563	19,276	6721	12,555	20,800	2,792	18,008
지 수	100	24	76	48.0	17	31	52	7	45

자료 : '95 채소생산 실적(농림부)

표 2-4. 시설유형별 설치면적 (단위 : ha)

구 분	대형터널	아 치 형	지 붕 형	기 타	계
면 적	20,725	18,611	729	11	40,076
지 수	52	46	2	-	100

자료 : '95 채소생산 실적(농림부)

표 2-5. 보온방법별 시설면적 (단위 : ha)

구 분	계	보 온(무 가 온)					가온
		소 계	이중피복	섬 피	부직포	기 타	
면 적	40,076	33,219	22,605	1,801	7,803	950	6,858
지 수	100	83	56	5	19	2	17

* 가온 : 연탄, 석유난로, 온풍난방기, 톱밥, 왕겨등

자료 : '95 채소생산 실적(농림부)

시설내 관수방법은 컴퓨터 환경계측 제어에 의한 현대적 관수방법의 도입은 극히 일부분이며, 주로 점적관수 등을 이용한 관수방법이 주종을 이루고 있다(표 2-6).

다. 시설투자비의 영세성

시설화훼 주산지 농가 8,945농가중 139개 농가를 대상으로 현재 설치되어 있는 시설에 투자금액을 조사해 본 결과 평당 5만원 미만의 시설이 95%로서 아직도 시설투자에 영세성을 나타내고 있다(표 2-7).

앞으로 시설개선시 투자가능 금액을 조사해 본 결과도 평당 100천원 미만의 농가가 95%로 시설투자의 영세성은 시설원예 발전에 큰 저해요인중에 하나라고 생각되어진다(표 2-8).

표 2-6. 관수방법별 시설면적 (단위 : ha)

구 분	계	스프링쿨러	점 적	호 스	살수(분무기)	인 력	기 타
면 적	29,258	791	4,774	18,823	1,461	2,459	950
지 수	100	3	16	65	5	8	3

자료 : '91 채소생산 실적(농림부)

표 2-7. 시설원예농가의 시설비 투자비율

평 당	1만원	2	3	4	5	6	7	8
비율(%)	21.2	39.4	19.7	6.6	8.0	1.5	2.9	1.0

자료 : '91시설원예 현대화 하우스 모델설정 및 재배효과에 관한 연구 (농진청)

표 2-8. 금후 시설개선시 투자 가능 금액 (단위 : 천원/평)

평 당 금 액	25	50	75	100	100이상
비 율 (%)	17.8	60.7	10.4	5.9	5.2

자료 : '91 시설원예 현대화 하우스 모델설정 및 재배효과에 관한 연구 (농진청)

라. 농가시설 경영 현황

(1) 자동화 시설의 도입정도

우리나라에서 시설재배시 환경관리, 재배기술, 수확조제에 대한 자동화 정도는 외국에 비하여 기술수준이 낮은데 그 요인은 다음과 같이 몇가지로 지적할 수 있다.

- o 원예시설의 부대장치에 대한 연구나 기술개발이 미흡했음.
- o 수요가 형성되지 않아 국내 생산 기술수준이 외국에 비하여 미흡하고 도입에 의존하는 부대장치가 많음.
- o 주문생산 공급 또는 도입 설치하는 경우가 많으므로 가격이 고가이므로 농가에서 설치를 원치않음.
- o 국내에서 부대장치 개발이 미흡할 뿐만아니라 성능면에서 낮은 경우가 많음
- o 1990年 이전 대부분의 원예시설에는 부대장치의 설치및 기능을 극대화하기가 어려운 실정이었음.
- o 농가의 기술수준이 미흡함.

(2) 시설유형별 적정규모 추정

농촌진흥청 경영관실에서 시설의 자동화 정도에 따른 적정규모를 분석한

결과, 농촌 노동력이 가족당 2인으로 보고 고용노력 30%를 포함해서 월노동 투하시간을 642시간으로 추정할때 적정규모는 600평이었다. 농가보급형하우스, PET, PC 및 유리온실의 경우도 환기, 난방, CO2시용, 관비시설, 연무방제, 선별작업등을 어느 정도 자동 또는 반자동화할 경우에는 농가당 적정규모를 1,200평 정도로 보고 있다(표 2-10).

2. 원예산업의 국제 경쟁력 분석

우리나라 시설원예의 면적은 일본 다음으로 세계 2위의 위치를 차지하고 있으나 원예시설의 재배환경과 환경관리 기술은 내수 위주의 국내 시장에서 원예작물을 생산하여 왔기 때문에 국제 경쟁력면에서는 열세에 있다 하겠다.

표 2-9. 자동화 PC 및 유리온실의 농가 적정규모 (단위 : 시간)

작 목	월 별	경영규모(평) 별 노동 투하시간			
		900	1,000	1,500	관행대비절감율 (%)
오 이	4월중순 5월상순	438	652	815	48
토 마 토	3월	468	619	738	57
장 미	1월하순 2월중순	426	568	710	59
카테이션	2월중순 3월상순	433	577	721	55

첫째, 원예생산물을 상품이라는 차원에서 품질 경쟁력을 분석해보면 국내시장에서 일등품의 장미가 시범수출 과정에서 일본 동경시장 품질평가로는 중품이하로 떨어진다고 하는 사실이 품질의 경쟁력 상태를 객관적으로 잘 입증해 주고 있다.

둘째, 원예시설과 생산기술의 발전 정도의 척도인 생산성도 과채류의 경우 네덜란드의 1/4 수준, 일본의 1/2 수준으로 떨어져 있다.

셋째 상품의 품질과 생산성을 결정짓는 원예시설의 기술수준은 비교적 하위에 있다.

이와같이 경쟁력이 열세인 그 원인을 종합적으로 분석해 보면, 먼저 기본적인 원인은 지금까지 비닐하우스 중심 시설원예 산업으로 선진국형의 고도의 집약 원예시설에 대한 수요가 없어 개발을 등한시 한점도 있으나 그동안 국내시장의 좁은 울타리에서 국제적인 시야가 부족한 때문으로 해석된다. 그 단적인 예가 철골소재의 고강도 경량화면에서 보면 우리나라에서 사용되는 철골자재는 300평에 10톤이 소요되지만 선진국 시설의 경우 철골자재는 강도가 강하면서 경량화되어 300평에 6톤정도가 소요된다. 뿐만아니라 수요시장의 불확실성으로 온실산업체간의 단결에 의한 알미늄 부재와 같은 온실전용 자재의 규격 표준화를 못시켜 규격품의 대량생산에 의한 가격 경쟁력면에서도 취약한게 현실이다.

3. 온실 농가의 기술수준과 수용태세

비닐하우스의 면적이 대부분인 우리나라 시설원예 농가는 유리온실에서 작물재배 및 온실운영에 대한 경험이 없었으며 온실의 자동화 부분에 대한 조절기술이 취약한 상태이다. 비닐하우스와 비교하여 유리온실이 가지고 있는 광, 온도, 습도의 유리한 면을 고려하여 작물을 선택하여야 한다. 그러나 이에 대한 정확한 지식없이 약광성 작물(동양란)을 재배하는 경우가

있으며, 작물에 대한 경험과 재배 분야의 기술축적도가 높은 농가가 필요에 의해 유리온실을 선택하기보다는 정부에서 50%를 지원하는 점과, 유리온실이면 품질을 향상시킬 수 있을 것이라는 막연한 기대감으로 유리온실을 짓고 있는 실정이다. 따라서 농가에서는 온실에 대한 사전지식인 유리온실의 특성과 기능성, 작물선택을 포함한 유리온실재배 기술능력, 품질차별화에 의한 시장 판매전략에 대한 정확한 분석을 실시한 후 유리온실을 짓도록 유도하여야 한다.

농가 선정시에도 작물재배, 경영능력 평가가 소홀한 반면 오히려 자금담보 능력이 우선되는 경향이 있어 정부지원사업 농가 선정시 현재의 평가기준에 대한 세밀한 검토가 요구된다.

또한 외국형 유리온실이 시공되어지면서 지금까지의 재배방식인 토양재배에서 배지재배로 전환되는 과정에서 선진 시설재배 기술수준과 기존 시설재배와의 기술차이에서 오는 불연속적 현상의 표출로 양액조절에 대한 기술부족, 시설의 주년 이용율 저하, 유리온실의 전용품종 부재, 장기재배시 공간활용기술 저하 등이 심각한 문제점으로 대두되고 있다. 물론 유리온실 사업이 아직 초기 상태로 어느정도의 시행착오는 있을 수 있으나 시설을 시공하여 직접 재배하고 있는 농가로서는 시설운영, 재배작물, 품종의 선택 재배기술에 대한 문제점은 시급히 해결하여야 할 우선 과제이다.

제 3 절 외국 시설원예의 현황과 발전 배경

1. 네덜란드

네덜란드에서는 정밀한 식물생장 조절을 위하여 꾸준히 재배기술을 개발하고 있으며, 최근에는 암면을 이용한 수경재배기술을 이용하여 토마토, 피망, 오이 등을 생산하고 있다. 전체 시설원예 면적은 9,000ha이며 25%이상인 2,500ha가 수경재배를 이용하고 있다.

채소작물은 전체 온실의 52%를 차지하며, 재배면적은 토마토가 가장 많고 그 다음으로 양상추, 오이, 피망, 가지 등이 생산되고 있다. 화훼는 전체 온실면적의 46%를 차지하며, 장미를 주축으로 카네이션, 국화, 프리지어, 카베라 등이 재배된다.

강한 해풍으로 인하여 흔히 벨로형이라 불리는 연동식 온실이 널리 사용되며 모두가 유리 온실로 구성되어 있다. 생력재배의 자동화를 위하여 최근에는 관수량, 액비의 농도, 탄산가스 농도, 온도, 습도, 일사량 등을 자동으로 조정하고 있다. 온실 내부의 환경을 가장 효율적으로 조절하기 위하여 소형의 기상관측소를 외부에 설치하여 온도, 습도, 풍향, 풍속, 일사량, 강수량, 적설량 등을 측정하고 있으며, 이들 자료를 바탕으로 난방, 환기, 기상부 등의 환경제어에 이용되고 있다.

표 2-10. 네덜란드의 주요 수출국의 수출액 (단위 : 백만달러)

분화 및 절화 총수출액	독일	프랑스	영국	미국	일본	기타국
4,572	2,210	546	375	91	81	1,269

2. 프랑스

프랑스의 온실면적은 7,200ha로서 유리온실이 3,150ha, 플라스틱온실이 4,050ha를 차지한다. 야채재배는 유리온실의 41%, 플라스틱온실의 88%를 차지하며 화훼재배는 유리온실의 59%, 플라스틱온실의 12%를 차지한다. 프랑스는 파리근교의 북부와 지중해 근처의 남부의 기후가 상이하여 두 지역에서 서로 다른 작형과 재배기술을 가지고 경쟁하고 있다.

파리 남쪽에 위치한 오를레앙(Orleans)시 주변은 오래된 원예작물 산지로서, 과수, 시설야채, 시설화훼, 노지채소, 묘목생산의 중심지이다. 이 지역은 오이, 토마토, 피망, 양상치를 재배하며, 유리 온실에서는 주로 가온촉성재배(加溫促成栽培)가, 플라스틱온실에서는 무가온재배(無加溫栽培)가 행해지고 있다. 이 지역의 시설형식은 유리온실의 경우 네덜란드와 마찬가지로 벤로형이 주를 이루며, 플라스틱온실은 연동형과 단동형이 병존하고 있다.

남부 프랑스의 원예는 19세기부터 발전하였으나, 파리와 교통편이 좋아지면서 본격적으로 개발되기 시작하였다. 이 지역은 기후가 온화할 뿐만 아니라 관개도 용이하여 프랑스 야채생산의 중요한 비중을 차지하고 있다. 1960년 이전에는 시설원예가 그다지 보급되지 않았으나 그 이후 플라스틱 피복재가 개발되면서 급격하게 플라스틱 온실이 보급되었다. 아비뇽(Avignon)-니스(Nice)지역은 온실면적의 12%가 유리온실이며 나머지는 모두 플라스틱온실이다. 피복재는 자외선 흡수제를 다량 포함한 강화 PE가 주로 사용된다.

지중해 지역인 프랑스는 겨울철이 온화하기 때문에 무가온에 의하여 토마토, 피망, 가지를 재배하고 있다. 무가온 재배의 성패는 일사량과 야간의 온실온도에 좌우되며 야간에 기온이 10℃ 이하로 내려가면 식물성장이 멈춘다. 무가온재배에서 문제점을 해결하기 위하여 2중 플라스틱, 태양열

축열튜브, 보온막 등을 이용한다. 2중 플라스틱은 야간에 복사열 손실을 줄여서 기온의 경우 0.5℃, 지온의 경우 1 2℃의 상승효과를 준다. 태양열을 이용한 난방법은 작물 사이에 지름 30cm정도의 플라스틱 튜브를 설치하고 튜브 안에 태양열 흡수효율을 높이는 첨가제를 섞어서 주간에 태양열을 흡수하고 야간에 방출하도록 한다. 이 방법은 야간기온이나 토양온도를 약 0.5℃ 높이며, 추운 날일수록 효과가 좋다. 보온막은 이미 10여년 전부터 북유럽의 가온온실에서 널리 사용되고 있다. 보온막은 고정형과 가동형이 있으며 고정형은 2중 플라스틱 온실과 같은 효과를 갖는다. 가동형은 주간에 복사열을 최대한 온실 내부로 받아들이고 야간에는 복사열 손실을 차단하므로 야간기온을 2℃까지 올리는 효과가 있으며 작물생산량도 20 50% 증가시키는 효과를 가져온다.

가온온실의 에너지 절약을 위하여 깊이 1,600m 속에 있는 65 75℃의 지하수를 지름 25cm의 강관으로 끌어올려 분당 200m3/s으로 공급한다. 이 분량은 17ha의 온실을 가온할 수 있으며, 지질광산국(BRGM)이 비용을 부담하며 이 사업을 계속적으로 확대해 가고 있다. 또한 원자력 발전소의 뜨거운 폐수를 난방용으로 사용하기 위한 노력도 하고 있으며, 유리온실을 유리 대신에 폴리카보네이트 겹판으로 교체하여 난방비를 줄이고 있다.

가. 프랑스 온실의 표준규범

1972년에 최초의 온실대상 표준규범이 만들어졌다. 그 이후 최초 표준규범은 수차에 걸친 개정보완작업을 거치게 되고 마지막 개정인 耐久性에 관한 표준규범은 1989년과 1991년에 만들어졌다. 1991년에 발효된 여러 형태의 표준규범은 다음과 같은 내용을 규정하고있다.

○ 기술적 용어 및 어휘의 규정 : NFU57001

○ 기계적 저항력 및 내구성에 관한 규정 : NFU57060, NFU57063, NFT57064

○ **시방서에 관한 규정** : NFU57010, NFU57013

○ **치수에 관한 규정** : NFU57020

○ **방수에 관한 규정** : NFU57060, NFU57063, NFU57064

표 2-11. **지역별 설계설하중**

구분	지역	시설형태	설계 설하중 (daN/㎡)
A	프랑스 서부, 남서부	대형아치형 연동 쌍터널형 터널형	42 38 30
B	프랑스 북부, 북동부, 중부, 중동부	대형아치형 연동 쌍터널형 터널형	53 45 38
C	프랑스 남동부	대형아치형 연동 쌍터널형 터널형	63 54 45
D	피레네산맥 동부 (Perpignan 지역)	대형아치형 연동 쌍터널형 터널형	91 73 65

(1) 하중에 관한 표준규범

설 하 중 : 대형아치형, 쌍터널형, 터널형 시설의 부하계산을 위한 표준규범으로 NFU 57063 과 NFU 57064가 있다. 프랑스 전체를 4개의 지역(A, B, C, D)으로 설정하고 각 지역별로 설하중을 규정해 놓고 있다.

풍 하 중 : 풍하중은 표준규범 NFU57063, 57064에 규정되어 있다. 설하중의 경우와 마찬가지로 프랑스 전역을 3개의 지역(Ⅰ, Ⅱ, Ⅲ)으로 구분해 각 지역별로 설계풍속을 규정해 두었다.

단, Perpignan지역이 특별히 취약한 곳이고, Rhone강 유역 전체, Vienne에서 Marseille 까지, Nimes에서 Toulon까지의 지역도 취약한 지역이다.

표 2-12. 지역별 설계풍속

구분	시설형태	설계풍속 (km/h)
Ⅰ	대형아치형 연동	91
	쌍 터 널 형	86
	터 널 형	81
Ⅱ	대형아치형 연동	108
	쌍 터 널 형	102
	터 널 형	95
Ⅲ	대형아치형 연동	122
	쌍 터 널 형	115
	터 널 형	108

(2) 온실 설치 현황

AGRESTE, 1995.1. #19에 의하면 프랑스에는 약 7,500ha의 온실이 설치되어 있고 이 중 반수이상이 남쪽과 남동쪽에 위치하고 있다. 유리온실은 15년전부터 화란에서 수입되어 사용되었으나 현재는 노후한 상태로 거의 사용되지 않는다. 터널형, 플라스틱 온실의 경우는 전체시설면적의 10%를 차지하고 있으며, 터널형, 쌍터널형, 대형아치형 연동온실이 프랑스 온실의 대부분을 차지하고 있으며 온실 총면적의 약 70%(5,050ha)가 채소생산에 이용되고 있다. 화훼온실은 2,150ha로서 온실 총면적의 30%정도이다.

3. 이스라엘

이스라엘의 온실 폭은 7 9m의 것이 대부분이나, 장미는 폭 22m의 대형 온실에서 재배되기도 한다. 시설의 피복재는 대부분 폴리에틸렌 필름으로서 수명은 1년 정도에 불과하다. 이밖에 폴리카보네이트 복층판, 유리, 파이버글라스 등이 사용되고 있다.

시설원예의 보급초기에 온실난방은 주로 네덜란드에서 유래된 온수난방에 의존하였으나 현재는 거의 사라지고 온풍기를 이용한 난방방식이 대부분을 차지하고 있다. 한편, 온실의 프레임 부분은 태양열에 의해서 과열되기 쉽기 때문에 피복재 필름의 약화를 방지하고자 백색으로 도색하고 있다.

표 2-13. 이스라엘 원예작물 수출현황('93) (단위 : 백만달러)

총 계	화훼류	과채류	채소류	오렌지	기 타
1,677	222	416	466	201	372

유럽에서는 겨울에 일사량이 충분하지 못하여 대규모의 장미재배를 하지 않으나, 이스라엘의 경우 겨울철에 장미의 집중출하를 목적으로 난방에 의한 가온재배를 실시하고 있다. 또한 사막지대의 기후특성에 의해서 겨울의 외기온은 0 5℃ 정도이므로 작물의 생육특성에 따라 보온 또는 난방장치를 설치하여 내부환경을 조절하고 있다. 여름에는 실내온도가 35℃ 이상의 고온이 되므로 환기에 의한 냉방을 실시한다.

남북동 온실은 양측면에 환기창을 설치하나 동서방향의 측벽에는 설치하지 않는다. 온실 내 상부와 하부의 온도차는 3℃ 정도로서 그다지 크지 않기 때문에 여름철의 실내온도를 전체적으로 저하시키기 위한 방안이 모색되고 있다.

이밖에 지하수를 이용한 난방과 관개의 실용화에 관한 연구가 시도되고 있다. 지하 300m의 깊이에서 끌어 올려진 40℃ 정도의 지하수는 온실의 토양을 가온시키는데 이용되며 궁극적으로 시설 내의 관개에 이용된다.

4. 일본

일본의 시설원예 규모는 유리 온실과 플라스틱 온실의 면적 합계가 48,812ha에 이르고 있다. 작목별 설치면적은 채소가 69.3%, 과수 19.3%, 화훼 11.4%로 나타나고 있다. 온실의 피복재는 염화비닐 필름이 널리 이용되고 있으며, 온실의 형태는 최근에 와서 비가림을 목적으로 한 간이형태의 무가온 파이프 온실과 고도의 자동화 장치를 설비한 온실로 양분되어 설치되고 있다. 채소재배에서 시설원예가 차지하는 비율은 오이, 토마토, 딸기, 피망 등에서 높게 나타나고 있다. 시설과수 면적은 포도가 6,200ha로서 대부분을 차지하며, 그 다음으로 귤, 오렌지 등이 있다.

최근들어 온실의 환경 제어기술이 현저하게 향상되어 온실내 환경을 제어하기 위한 장치가 점차 복잡해지고 있다. 이러한 장치 가운데는 일사량

에 따라 실내온도 뿐만 아니라 습도, CO2 농도를 제어하고 풍향과 풍속에 의해서 환기창의 개폐정도를 제어하는 장치가 포함되어 있다.

또한, 디지털형 복합환경 제어장치가 개발되어 난방, 지온, 차광, 환기, 관수 및 지중열 교환 온실의 환경 제어에 이용되고 있다.

전체 유리온실의 85.5%가 가온설비를 갖추고 있으며, 난방방법은 온풍난방이 45.9%, 온수난방이 43.9%를 차지하고 있다. 가온설비는 에너지 절약형 주요 설비로서 가온시설의 65.2%가 변온관리장치를 갖추고 있다. 기타 부대시설로서는 자동 관수장치, 보온커튼, 자동 천창 및 측창 개폐장치 등의 설치가 증가하고 있다. 플라스틱 온실의 경우는 31.7%가 가온설비를 갖추고 있다. 난방방법은 82.7%가 온풍기를 사용하고 있으며 이 가운데 51%가 변온설비를 갖추고 있다.

표 2-14. 일본의 시설 유형별 시설면적(1992) (단위: ha당)

전체면적	유리온실	비닐하우스			
		경질판	경질필름	염화비닐	기타
47,166	2,132	603	954	40,256	3,221

제 4 절 우리나라 시설원예의 발전방향

1. 연구개발에 의한 기술의 축적

시설원예의 발전 원동력은 심대한 시험연구시설에 의한 조직적인 연구개발 결과이다. 네덜란드의 경우 관련 연구기관이 3개소로서 많은 유리온실 시설과 연구를 볼 수 있었으며, 일본도 2개소의 국립연구소와 현마다의 연구소에서 시설원예에 대한 집중연구를 하고 있다. 따라서 우리나라에서도 시설원예연구를 위한 과감한 시설과 장비를 투자하여 연구시설기반 정비가 최우선 과제이다. 연구시설기반 정비이후 시설구조, 장치, 관리시스템, 및 재배법등의 시설원예 종합연구로 우리나라의 기술수준을 구미의 농업선진국과 대등한 수준으로 올려놓는 것이 시급하다.

2. 시설원예 교육 및 연구지도 강화

연구결과의 교육 및 현지지도 강화로 발전된 기술을 신속히 농민에게 보급시켜 생산성을 향상시키는 것이 중요하다. 따라서 우리실정에 맞는 보다 나은 시설들의 조속한 개발 및 관련 연구사업 확대 추진이 필요하다. 또한 지역단지중심 작목반 조직의 활성화로 단지중심의 자발적 토론회를 통한 정보와 기술의 상호 교환을 이루는것도 중요하다.

3. 시설원예 관련산업의 육성

현재의 다양한 시설골격자재 및 시설자재를 규격화된 제품으로 생산토록 함으로써으로 제품의 가격인하와 이용을 보다 합리적으로 하게 하여야 한다. 그러기위해서는 품질보증을 위한 시설자재의 품질 및 규격표시를 의무화하고 표준규범을 제정할 필요가있다. 아울러 시설장비(생력장치 포함) 제

조회사를 적극적으로 육성하여 관련업체의 경쟁력을 높이고 산·학·연 협동연구로 기술개발을 촉진하여야 한다.

4. 유통조직과 시설의 현대화

단지중심의 집하와 선별 및 포장시설의 공동이용으로 판매상의 유리점과 기계 및 시설이용의 저렴성을 추구하여야 한다. 또한 선도유지를 위한 유통시설의 개선과 도매시장의 경매방법을 현대화하며 유통의 신용거래가 정착되어야 하겠다.

제 3 장
온실 환경제어 연구 현황

제 1 절 온실 환경의 특성

흔히 온실효과(geen house effect)라 하는 것은 지상에 도달하는 복사에너지에 비해 천공으로 역복사되는 에너지의 차인 순복사가 많아지는 현상을 가리킨다. 지구의 온난화 현상도 이와 같은 맥락에서 이해될 수 있으며 온실의 대표적인 특성으로 꼽을 수 있다.

피복 자재로 외부 공기를 차단하고 인위적인 공간을 형성한 것이 온실이므로 밀폐된 상태에서 태양의 단파 방사가 계속적으로 온실 내부에 유입될 경우 복사되는 온실내의 식물체나 지표면에서 에너지가 흡수되면서 파장은 길어지게 되며 이른바 장파 방사인 열선으로 바뀌면서 열이 축적되어 기온이 상승하게 된다. 이 때 환기창 등으로 기온을 조절하지 않을 경우 온실 내부는 외부 기온보다 2.5 3배정도 상승하게 되며, 야간에는 반대로 천공을 향한 역복사가 활발히 진행되는 날에는 보온 자재 등으로 보온하지 않는 외겹 온실의 경우 외부 기온보다 오히려 떨어지게 된다. 이와 같이 환경을 조절하지 않는 온실내에서는 낮에 40 50℃이상 올라가고 야간에는 외부 기온이 낮을 경우 영하로 떨어지게 되는 특성을 가지므로 이를 사막기후로 비유하기도 한다.

온실의 두 번째 특성은 공간내 기온의 수직·수평분포의 불균일성이 매우 심하다는 것이다. 온실은 밀폐된 공간이면서 다습한 토양층과 식물군락층,

상부의 공기층으로 형성되어 있어 온실내의 환경은 일반 건축물과는 전혀 다른 에너지의 이동, 잠열과 현열로의 변환이 일어나기 때문이다.

마지막으로 온실은 독특한 물 순환특성을 가지고 있다는 것이다. 온실내 식물체의 증산과 지면으로부터의 계속적인 증발로 인해 쉽게 과습이 되면서 기온의 하강과 더불어 노점(dew point)에 쉽게 도달하면서 피복면에 결로(結露)현상이 일어나며, 이로 인해 시설내 부대장치들의 내구성을 현저히 단축시켜 주는 원인이 된다.

이처럼 온실은 독특한 환경여건을 지니므로 온실환경을 작물생육에 최적이 되도록 조성해주는 온실 환경제어 방법 또한 다양하게 발전되어 왔다.

제 2 절 온실 환경제어 관련 연구 현황

1. 광 환경 제어

광합성에 필요한 빛 에너지를 인공광원에 의존하고 있는 완전자동형 온실에서는 인공 조명과 광원의 발열을 보상하기 위한 냉방용 전력비가 전체 생산비용의 약 50% 이상으로 매우 높은 비중을 차지하고 있다. 따라서 광원의 선정과 조명 시스템의 설계 및 적정 제어 방법의 도입 등은 매우 중요한 비중을 차지하게 된다(홍, 1990).

광 강도 제어에 관한 연구로, 橋本(1987)은 온실에 이용되는 형광등을 상용 주파수 보다 수백 배의 고주파를 이용하여 간헐적으로 점등시킴으로써, 동일한 광도에서 전력비를 절감하였음을 보고하였다.

작물생육 시스템에 사용되는 인공광원으로는 주로 형광등, 백열등, 나트륨등, 수은등 및 메탈-할로이드계열의 램프가 있다(高 , 1987). 이들은 광원효율과 스펙트럼, 단위면적당 광도에 있어 각기 고유한 특성을 보이므로 실제 설계에 있어서는 이들 광원을 조합하여 구성하는 것이 바람직하다.

김(1994)은 형광등, 텅스텐-할로겐 램프, 메탈-할로이드 랩프를 조합하여, 광합성 유효 스펙트럼과 유사하고 최대 24klux의 조도를 갖는 인공광원 장치를 개발하여 생육시기에 따른 광도 및 광주기의 효과를 실험하였다. 이 실험에 의하여 생육 초기에는 광주기를 길게 설정하고 광도를 낮추는 것이 효과적이며, 생육 후기에 있어는 광주기를 짧게 설정하고 광도를 높이는 방법으로 제어하는 것이 작물의 상대적 생장률을 높일 수 있음을 보고하였다.

2. 온도 및 습도 제어

Takakura 등(1974)은 PDP-8/E 컴퓨터를 이용하여 온습도 환경의 직접 디지털 제어장치와 제어 알고리즘을 개발하여 시설 환경에 대한 컴퓨터 제어의 가능성을 제시하였고, 환경조건과 식물의 생장을 연관시키는 자동제어의 연구 방법을 제시하였다.

Parsons(1980) 등은 환경조건의 변화에 따른 식물의 반응을 실험하기 위하여 온습도 및 탄산가스 환경에 대한 실용적인 데이터 수집장치의 하드웨어와 제어용 소프트웨어를 개발하여 발표하였다.

Tantau(1985)는 온실의 온도 환경을 유지하기 위하여 난방 시스템과 환기장치를 효율적으로 제어하는 시스템을 발표하였다. 여기서는 외기 조건과 내부 온도에 대한 단순화된 모델과 적응제어 알고리즘이 이용되었다.

Saffell(1985)은 PDP-11 컴퓨터를 이용하여 환풍용 팬과 온풍 난방기, 회전 원판식 가습기 등을 제어하는 방법으로, 유리온실의 실용적인 온·습도 제어 시스템을 제안한 바 있다. Mitchell(1986)은 SYM-1 싱글 보드 컴퓨터를 이용한 작물 생육장치의 제어를 통하여 실내 온도와 열교환기 각부의 온도를 5 38℃ 범위에서 ±1℃의 오차로 제어하였으며, 관련 하드웨어를 간략하게 구성함으로써 실용성을 향상시켰다.

서와 민(1990)은 온실 환경제어 시스템 개발의 일환으로 8 bit 개인용 컴퓨터를 이용한 온도 제어 시스템을 개발하여, ±1℃ 이내의 정도로 제어가 가능하다고 보고하였다. 이 등(1992)은 작물생산 자동화를 위한 기초 연구의 단계로, 완전제어형 작물 생육장치의 온도 및 습도 제어를 위한 센서 신호처리 회로와 공조용 작동장치를 구성하고, 마이크로 컴퓨터에 의한 자동조정형 제어 알고리즘을 개발하여 성능시험을 수행한 결과, 외란이 작용하는 경우에 있어서는 온도 10 40℃, 상대습도 50 90%인 설정 범위에서 오차 ±0.3℃, ±5% RH 이내의 정확한 제어가 가능함을 보고하였다.

3. 탄산가스 농도 제어

노출된 대기 환경에서는 탄산가스 농도의 인위적인 제어가 불가능하지만, 일정한 시설 내에서는 그 조절이 가능하기 때문에 작물의 생장을 촉진시키기 위한 탄산가스 시설이 이용화되고 있다.

탄산가스 농도는 항상 일정하게 유지하는 것보다는 작물의 종류와 생육상태, 그리고 생육시기에 따라 변화시키는 것이 합리적이다. 伊東(1976)은 광합성 속도가 광도 및 온도와 밀접한 관련이 있다는 점에 착안하여 광도와 시설의 온도를 감안한 탄산가스의 공급량의 결정 방법에 대한 연구를 수행하였다.

시설 내부의 탄산가스 환경을 제어할 때, 외부 대기와의 농도차로 인한 확산 현상 때문에 Nelson(1991)은 환기 팬의 구동과 천·측창의 개폐를 연계시켜 탄산가스 공급장치의 동작을 제안하는 방법을 제안하였으며, Hartz 등(1991)은 환기의 제약을 최소화하기 위하여 땅속에 묻어 놓은 관수용 파이프를 통한 탄산가스의 공급 방법을 개발하였다.

한편, Hwang(1993)은 아열대 기후에 설치된 온실에서 외기 온도 조건과 연계하여 탄산가스를 공급하는 방법을 개발하여, 오전 중에는 공급 및 환기를 반복하여 수행하고 오후에는 공급하지 않는 것이 유리하다고 보고하였다.

탄산가스 공급의 자동제어에 있어서는 안정성의 증가를 위해 비례이득을 작게 설정하는 것이 일반적이며, 작물의 생육 단계별로 흡수되어야 하는 탄산가스의 양과 온실 용적 등을 미리 계산하여 공급량을 제어하는 피드포워드(feedforward) 방식이 바람직하다(橋本, 1987). 이러한 연구의 일환으로, 진 등(1993)은 CO2 시용을 보다 효과적으로 수행하기 위하여 환경 변화에 따라 민감하게 반응하는 작물의 생장지표인 광합성 속도를 제어의 피드백 신호로 활용하여, 탄산가스 공급량을 대폭 감속시

킨 상태에서 목표한 수확량을 얻어낼 수 있는 제어 알고리즘을 개발한 바 있다.

4. 시설의 환경 모델

작물생산 시설의 환경 모델에 관한 대부분의 연구는 온실 재배시의 열손실을 감소시키고 보온효과를 증진시킴으로써 투입되는 에너지를 절감하려는 목적에서 수행되었다. 이러한 연구는 피복재의 종류에 따른 온실 내부의 열적 특성 변화와 열전달 특성을 고찰하거나, 난방 및 냉방 장치의 효과를 분석하기 위한 것이 대부분이다(Avissar and Mahrer, 1982; Kano and Sadler, 1985; 박·고, 1987). 모델의 접근 방법에 있어서는 동적 모델과 정적 모델, 단일 요인만을 고려한 모델과 복합요인을 고려한 모델, 그리고 분석적인 방법과 수치 해석적인 방법 등이 적용되고 있으며, 온실 내부와 외부 사이의 대류, 전도, 복사, 환기 등에 대한 열전달 이론을 이용한 다양한 모델이 제시되고 있다.

정상상태의 에너지 평형에 기초한 정적 모델은 많은 연구에서 찾아볼 수 있다(Walker, 1985; Garzoli and Blackwell, 1973; Chandra and Albright, 1980; Baily, 1984). 정적 모델에서는 각 상태변수의 시간에 대한 변화가 없는 것으로 보아, 모든 입력을 상수로 취급한다. 이와 같은 상태에서의 에너지 지수에 대한 분석 방법은 온실 환경의 변화 과정을 기술함에 있어서 매우 효과적이며, 온실의 기후를 시뮬레이션 하여 난방 부하를 산출하거나, 각종 제어 장치 및 제어 방법에 대한 성능 평가를 위하여 사용되는 경우가 많다.

반면에 동적 모델에서는 상태변수의 변화율을 고려하게 되는데, 대부분의 온실 시스템은 비선형이기 때문에 분석적인 해를 구하기 어려우므로 컴퓨터에 의한 수치 해석적인 해를 구하게 된다. Takakura 등

(1971)은 최초로 작물 생장과 온실 환경에 대한 현실적인 시뮬레이션 모델을 개발하였다. Jones 등(1984)은 식물생육 챔버에서 가열, 냉각, 탄산가스 공급 프로세스 평형에 대한 컴퓨터 제어를 위하여, 에너지 평형, 수증기 질량 평형, 그리고 탄산가스 평형에 대한 방정식을 설정하여 챔버의 내부 환경에 대한 정상상태 모델을 제안하였다.

최근에 이르러, 손 등(1993)은 임의의 조건에 대하여 시설 내부의 환경 예측과 평가를 위한 다목적 환경 예측 모델을 발표하였다. 또한, 남 등(1993)은 수경 온실의 열 및 물질평형 이론에 기초한 양액 냉각부하 산정용 시뮬레이션 모델을 개발한 바 있다.

그리고 김 등(1995)은 양지붕형 단동하우스의 천창과 좌·우측창의 개방방법에 따른 온·습도의 변화를 규명하고, 하우스내의 평균온도의 변화를 수치화 하여 환기량과 환기시간의 관계를 해석함으로써 환기방법의 개선에 필요한 기초자료를 제시하였다.

송 등(1996)은 겨울철 시설 재배 온실에서 태양에너지와 기타 폐열을 고농축 저장·이용할 수 있는 축열재로 몇가지 파라핀을 선택하였으며, 이를 실용화하기 위한 연구를 수행하였다.

5. 생육환경의 최적제어

실용적인 시설에서, 생육실 내부의 환경요인은 외기 환경조건, 시설의 구조에 관련된 변수들, 그리고 환경제어 장비의 특성 등 다양한 요인에 의하여 결정된다. 이때 시설의 환경은 환경요인의 설정에 소요되는 에너지의 투입을 최소화하면서 수확된 작물의 가치를 최대로 하는 점에서 제어된다.

Udink tem Cate와 Challa(1984)는 작물 생산 시스템의 최적제어 방법으로 단계적 제어 시스템을 제안하였다. 여기에서는 제 1단계를 온실

환경제어, 제 2단계를 작물 생리적 환경 조성, 제 3단계를 작물 생육상태 제어로 분리하여 동적 최적제어를 구현하는 이론을 제시하였다. 또한 高 (1987)은 완전한 인공환경 상태에 있는 식물의 연속 생산 시스템, 즉 식물공장에 대한 연구를 수행하여 온도, 광, CO2 농도 등의 지상부 환경을 인공적으로 제어한 결과, 일반적인 자연 환경에 비해 약 6배에 해당하는 생장촉진 효과를 얻었으며, 파종에서 수확까지의 기간을 ¼로 단축시켰음을 보고하였다.

최근의 연구에서는 최소의 경비를 투입하여 수량을 최대로 하거나 재배기간을 최소화하기 위한 최적제어 이론이 반영되는 추세이다. 그 예로, 퍼지(fuzzy) 및 신경회로망 이론을 이용하거나(Seginer and Sher, 1992), 전문가 시스템이 도입되고 있으며(Jacobson 등, 1987; Jones 등, 1989), 탐색기법(Marsh and Albright, 1991) 등이 응용된 바 있다.

Challa 와 van Stratem(1991)은 실용적인 최적제어 시스템의 실현 가능한 구조에 대하여 제안하였다. 여기에서는 생육과 온실 기상에 관련된 복합적인 지식을 기초로 하여, 환경제어에 대한 단시간 응답의 지표로 광합성 속도를 측정하고 경영적인 개념을 도입하여 목적함수를 만든 후, 판단 기준을 설정하는 최적제어 방법이 사용되었다.

高 (1992)은 완전제어형 식물공장의 환경제어에 있어서 최적화 원리를 제안하여, 최대 생장 조건으로서의 환경 설정보다는 비용성능(cost performance), 또는 이익이 극대화되는 환경조건으로서의 최적화 제어가 바람직하다고 주장하였다.

제 3 절 온실 제어시스템

1. 완전 수동형 제어시스템

인간에 의해 환경을 감지하고 판단하여 난방기, 환기창, 보온커튼 등을 조작하는 것으로 현재도 대부분의 영세 농가에서는 이 방식을 이용하고 있다. 완전 수동형 제어시스템의 구성도는 그림 3-1과 같이 나타낼 수 있다.

이 방식의 문제점은 하우스 재배의 다양한 경험이나 지식이 없으면 작물에 적합한 환경을 최적으로 조절할 수 없으며, 더욱이 관리자가 상주하여야 하는 것이지만 장치비가 가장 저렴한 장점도 지니고 있다.

2. ON/OFF 제어시스템

이 제어계는 인간이 하우스 환경을 감지하는 것을 센서라고 하는 기계적인 전기적 도구를 사용해 하우스 환경을 수치화 정량화하여 측정하고 인간에 의해 설정된 환경조절 기준과 비교하여 작동기를 자동 조절하는 제어계이다. 이 방식은 그림 3-2와 같이 구성되어 정밀한 환경제어가 가능하지만 환경기준의 수시 변동설정이 어렵다.

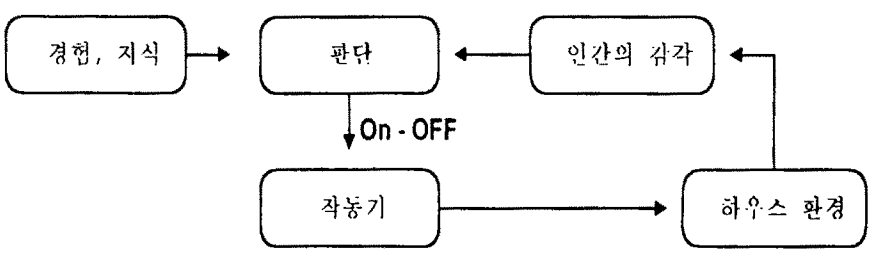


그림 3-1. 하우스환경의 완전 수동형 제어시스템

예를 들어 하우스내의 온도는 작물의 특성에 따라 태양광의 세기에 의해 변동설정되어야 하는데, 이 조작을 위해서는 온도계와 일사계를 구비하고 조작자(사람)가 그때 그때 기준을 변경, 설정하여야 한다.

하지만 인간이 조절기를 대신하는 완전수동형 제어시스템보다는 정밀한 제어와 관리노력이 절감된다고 할 수 있다. 이 방식은 개별제어나 집중제어장치에 자체온도조절식 온풍기, 타이머 부착식 보온 커튼 개폐장치 등이 이에 속한다.

3. 복합환경 제어시스템

ON/OFF 제어시스템에서는 4단 변온 제어를 실시하고자 한다면 태양과의 세기에 따른 1일의 시간별 온도설정(상한, 하한)과, 이 설정 온도 유지를 위해 작동기(환기창, 환개팬, 난방기 등)의 작동순서를 도표화하여 관리자가 하루종일 작동기를 때맞추어 조작하여야 한다. 그러나 복합환경 제어시스템은 그림 3-3과 같이 작물재배에 대한 최적환경조건을 경험과 연구

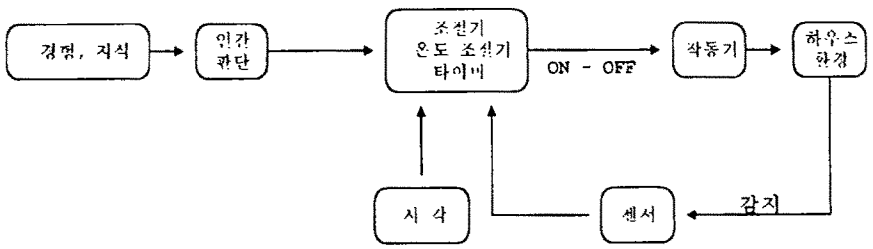


그림 3-2. 하우스 환경의 ON-OFF 제어시스템

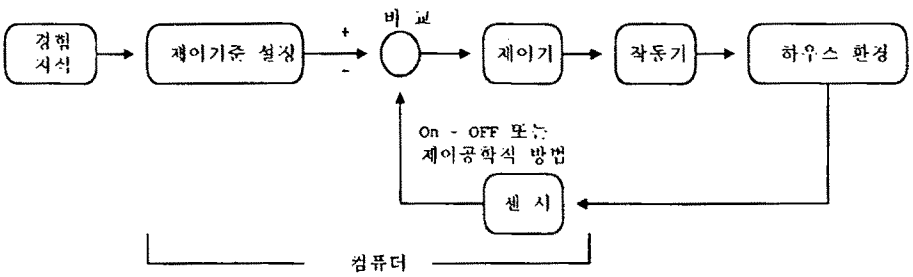


그림 3-3. 복합환경 제어시스템

결과인 지식을 정보화하여 제어기준을 설정하고 하우스환경을 센서로 측정하여 제어기준과 비교하여 하우스 환경이 제어기준에 일치하도록 제어기(controller)를 통해 작동기를 구동한다. 이와 같은 일련의 과정이 반복되면서 하우스환경이 최적상태로 유지되게 할 수 있다.

따라서 컴퓨터를 도입함으로써 전문기술자나 농가의 경험 등의 정보를 컴퓨터에 입력하여 모든 상태를 자동제어 하므로써 관리자를 필요로 하지 않고 하우스재배 경험이 부족한 농가라도 농사를 지을 수 있게 된다. 다만, 장치비가 많이 들게되므로 경제성을 고려한다면 대단위 하우스나 고급시설에 이 장치를 도입하고 있다.

4. 생체정보를 이용한 환경제어

작물에 영향을 주는 환경보다도 환경에 반응하는 작물의 상태를 제어목적으로 하여 직접 측정하면서 환경을 제어하는 것이 보다 합리적이다. 또한 부적절한 환경조건에 대한 재배자의 효율적인 기술적 조치를 위해서는

조치(제어)에 대한 작물의 생리적 반응을 측정하고 알아야만 한다. 이렇게 함으로써 첫째, 작물의 부적당한 상태를 즉시 발견해낼 수 있어 즉각적인 조치가 가능하며, 둘째 재배적 조처에 대한 작물의 생리적 반응을 알 수 있으므로 최대생산성을 획득할 수 있는 기술적 조치들을 적정화할 수 있는 장점이 있다.

생체정보를 이용한 환경제어는 엽온, 도관을 통한 수분이동, 줄기 및 과실비대, CO_2고정 등의 생체정보뿐만 아니라 일사량, 온도, 습도, CO_2 농도, 근권온도, 근권수분 등 온실 내·외부의 기상환경 측정자료도 함께 이용하게 된다. 즉 작물의 주변환경과 환경에 대한 작물의 생체반응을 함께 측정하여 상호 비교, 분석하고 제어를 행하게 된다.

현재는 계측된 생체정보를 재배자가 이해하기 쉽도록 자동적으로 처리하여 나타내어주고 이를 토대로 후속적으로 재배자가 환경의 제어설정치를 수동교정하고 있는 단계에 있다. 계측된 생체정보를 처리하여 나타내주는 형태로서는 다음과 같은 3가지의 형태가 있다. 이들 모두는 측정된 data를 처리하는 software를 통해서 얻어지는 것이다.

① 각각의 항목에 대한 매일의 변화과정

② 관련자료의 분석을 통해 직접 측정하지 않는 식물체의 특성(기공개폐, 팽압, water stress 정도)

③ 현상태에 대한 설명과 경고메시지 등

생체정보를 직접 이용한 환경제어방식은 현재 개발중에 있으며 곧 실현가능해질 것으로 생각되며, 그것의 가장 일반적인 방식은 feedback 제어로서 그림 3-4와 같다.

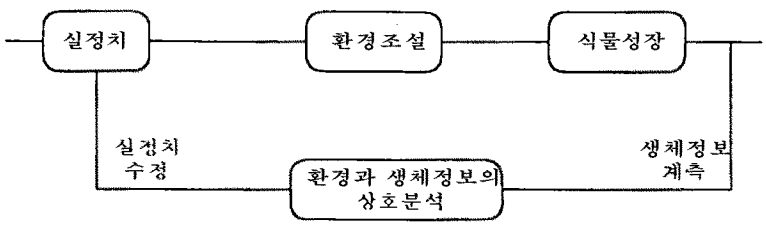


그림 3-4. Bio-Feedback 제어

여 백

제 4 장
실 험 재 료 및 방 법

제 1 절 온실의 구조 및 구동부

1. 온실의 구조

본 연구를 수행하기 위하여 농촌진흥청 원예연구소(수원시 탑동 소재)에 20평 규모의 온실 2동을 사용하였다. 각 온실은 가로 6m, 세로 11m 크기이며, 내부에는 0.5m×9m의 베드를 3열씩 배치하였다.

온실의 피복재로는 유리섬유강화 폴리에스테르(FRP)와 경질 폴리에스테르(PET)를 사용하였다.

온실 피복재는 광투과성과 보온성이 양호해야 하나 이 두가지 조건은 서로 상반되는 물리적 성질로서 동시에 만족시키기는 어렵다. 따라서 피복재는 가능한 한 광투과성이 양호한 동시에 야간의 보온성이 우수하여야 한다. 재배목적에 적합한 피복재의 선택은 냉난방, 작물의 품질및 생산단가 등과 밀접한 관계를 가지고 있으므로 피복재의 물리적 성질, 내구성, 경제성 등을 고려하여 선택하여야 한다.

본 연구에 이용된 온실의 피복재는 유리섬유강화 폴리에스테르(FRP)와 경질 폴리에스테르(PET)이다. FRP는 경질판으로서 불포화 폴리에스테르 수지를 주체로 하여 유리섬유로 보강한 피복재이며, 일반적으로 두께는 0.8 - 1.2mm의 것을 많이 사용하나 본 시험에서는 0.8mm 32파의 파판을 사용하였다. FRP는 유리에 비해 가볍고, 충격에 강하며, 쉽게 성형할 수 있는 장점을 가지고 있다. 파장별 투과성은 380nm이하의 자외선 영역 파장의

빛을 흡수하여 투과시키지 못하므로 가지과나 적자색 계통의 화훼류에 색소발현장애를 주기도 한다.

PET는 경질 필름으로 가소제를 함유하지 않은 두께 0.1~0.2mm의 투명한 염화비닐 또는 폴리에스테르 필름을 말하나 본 시험에서는 0.17mm의 것을 사용하였다. 경질 폴리에스테르 필름은 연질 필름과 달리 장파의 투과가 억제되어 보온성이 우수하며 투광성이 좋고 인장강도가 크다. 또한 먼지 등의 부착에 의한 투과율 저하가 적고, 연소시켜도 유해가스가 발생하지 않는다.

유리섬유강화 폴리에스테르(FRP) 피복 온실(그림 4-1)에는 1,2기작에는 오이를 정식하였고, 경질폴리에스테르(PET) 온실(그림 4-2)에는 토마토를 정식하였다. 표 4-1은 본 연구에 이용된 온실의 구조적 특징을 나타낸다.

두 온실의 동고, 측고 및 상면적은 동일하였으나 피복재고정의 특징상 PET피복온실의 경우 골조율이 FRP 피복온실 보다 5% 높았으며 환기면적율은 천창길이의 차이로 인해 FRP 피복온실에서 다소 높았다.

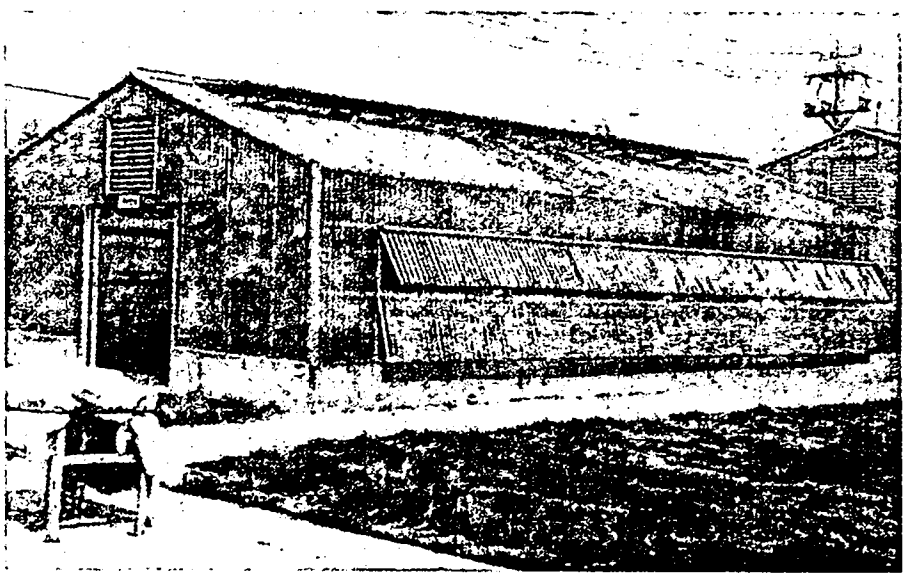


그림 4-1. FRP 온실의 전경

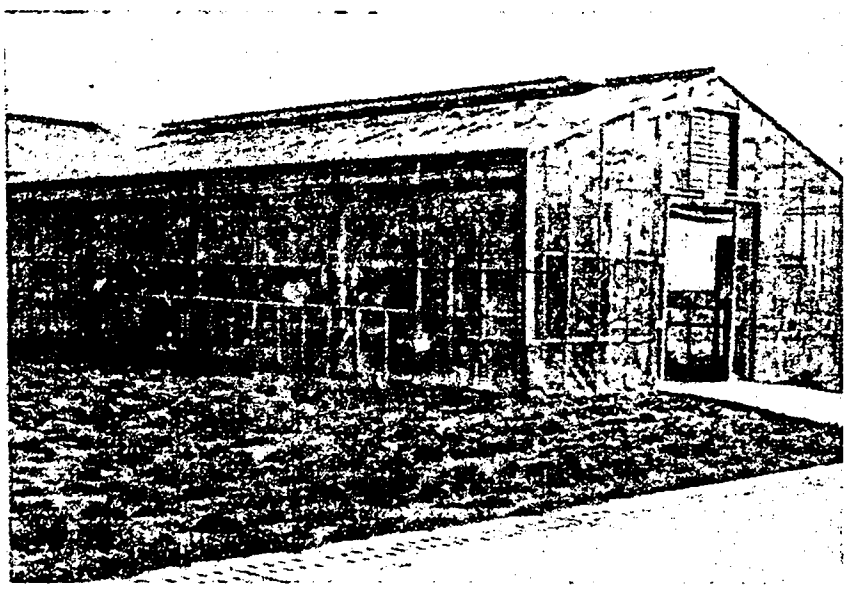


그림 4-2. PET 온실의 전경

표 4-1. 온실의 구조 특성

구 분	PET (Polyethylene terephtalate)	FRP (Fiber Reinforced Plastics)
피복자재	PET (0.5㎜ 필름)	FRP (1.0㎜ 골판)
동고(m)	3.85	3.85
측고(m)	2.3	2.3
폭(m)	6	6
길이	12	12
기둥	□ - 100×50×4.5	□ - 100×50×4.5
<간격(m)>	3	3
트러스	□ - 100×50×4.5	□ - 100×50×4.5
중도리	C-60×50×15×2.0	C-60×50×15×2.0
서가래	알미늄형재(폭3㎝)	-
브레싱(㎜)	φ12	φ12
측창폭(m)	1.4	1.4
측창길이(m)	9.0	9.54
측창수	2	2
천창폭(m)	0.844	0.844
천창길이(m)	9.0	9.54
천창수	2	2
환기선지름(㎝)	66	66
환기선수	2	2
골조율(%)	18.2	13.2
환기면적율(%)	24.77	26.02
용적(㎥)	221.4	221.4
상면적(㎡)	72	72
표면적(㎡)	192.98	192.98
방열비	2.68	2.68
지붕기울기(°)	30.46	30.46
강제환기횟수(회/hr)	81.3	81.3

2. 온실작동기의 설계 및 설치

온실에는 작동기로서 측창, 천창, 커튼 개폐 모터와, 팬이 설치되어 있으며, 복합환경제어실의 제어신호에 따라 작동하도록 하였다. FRP온실의 측창, 천창 개폐용 모터와 PET온실의 천창 개폐용 모터는 AC모터를 이용하였으며, PET온실의 측창 개폐용 모터와 각 온실의 커튼 개폐용 모터는 권취형 모터를 이용하였다. 표 4-2는 AC모터와 권취형 모터의 사양을 나타낸다.

측창, 천창 및 팬은 온실 내의 온도와 습도 조건에 따라 자동으로 작동되도록 하였으며, 커튼은 온도, 습도 및 일사량에 따라서 작동되도록 하였다. 양액·관수시스템의 작동은 독자적인 외부 전원을 이용하며, 복합제어실에서도 제어가 가능하도록 설계되었다.

표 4-2. 모터의 사양

항 목	사 양	
	AC 모터	권취형 모터
모 델	TTH-0024	V-301-140W
정격전압	220V/380V	220V
정격출력	200w	140w
정격회전수	1750 rpm	1750rpm
효 율	53%	52%

겨울철의 온실 난방을 위하여 난방용 보일러와 방열기를 설치하였다. 난방용 보일러는 가정용 석유보일러로 각 온실에 1대씩 설치하였으며, 용량은 25,000kcal의 열량을 발생시킬 수 있다. 본 연구에 이용된 방열기는 콤벡타형 방열기로서 에어로핀 방열기를 사용하였으며, 보일러에서 발생한 열을 온실 내부에 골고루 전달시킬 수 있도록 하였다.

지중온도 유지는 보일러에서 공급되는 온수를 베드 내부로 공급되도록 설계하였다. 베드 내부에는 온수공급용 호스를 2열로 배치하였으며, 보일러의 온수 공급부에는 순환모터를 설치하여 지중온도에 따라 컴퓨터의 신호에 의하여 작동하도록 하였다. 지중온도는 2기작의 경우 정식초기에는 23℃를 유지시켰고 생육중기 이후 20℃를 유지하였다. 표 4-3은 본 연구에 이용된 보일러의 사양을 나타낸다. 그림 4-3는 본 연구에 이용된 보일러이며, 그림 4-4는 온수 순환모터를 나타낸다.

모든 작동기는 온실 내부에 설치된 동력제어반(그림 4-5)에 연결되었으며, 스위치에 의한 수동조작과 컴퓨터에 의한 자동제어가 가능하도록 제작되었다. 우리나라에 공급되는 동력제어반은 기능에 비하여 규모가 크므로 온실 내부에서 동력제어반이 차지하는 면적이 넓고, 가격이 비싼 단점이 있다. 따라서, 본 연구에서는 동력제어반을 단순화하여 소규모, 저비용으로 설치가 가능하도록 하였다.

표 4-3. 보일러의 사양(기름 연소 온수 보일러)

항 목	사 양
연료	등유, 경유
전원	220V, 60Hz
연소방식	압력분무식
가열방식	2회로식
급수방식	시스턴식
난방 사용 최고압력	1 kgf/㎠
연료소비량	3.4 ℓ/h
출력	25,000 ㎉/h
급배기방식	반밀폐식 강제통기형
저탕량	40 ℓ
효율	85% (난방시)
소비전력	점화시 : 88w 연소시 : 74w

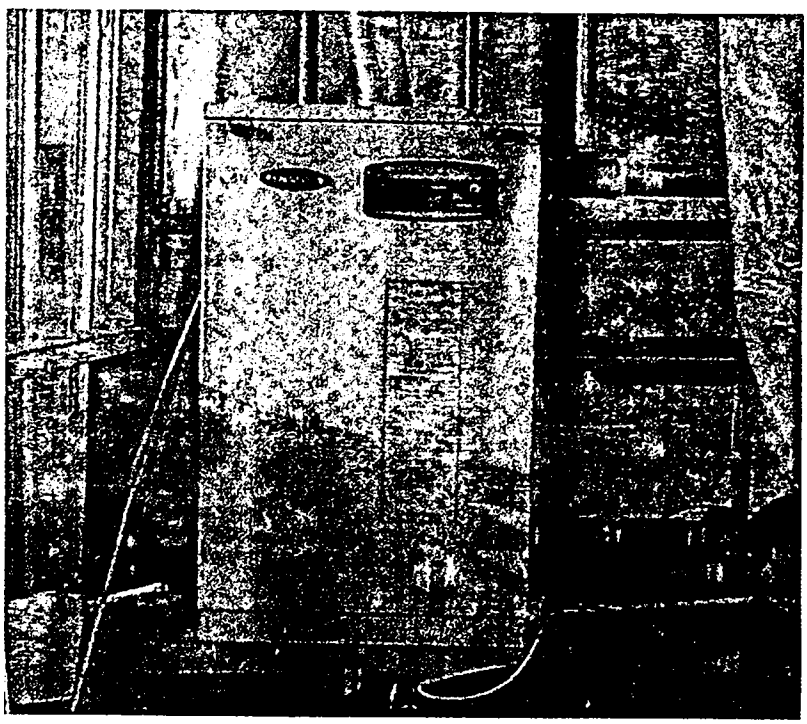


그림 4-3. 보일러

그림 4-4. 온수공급용 순환모터

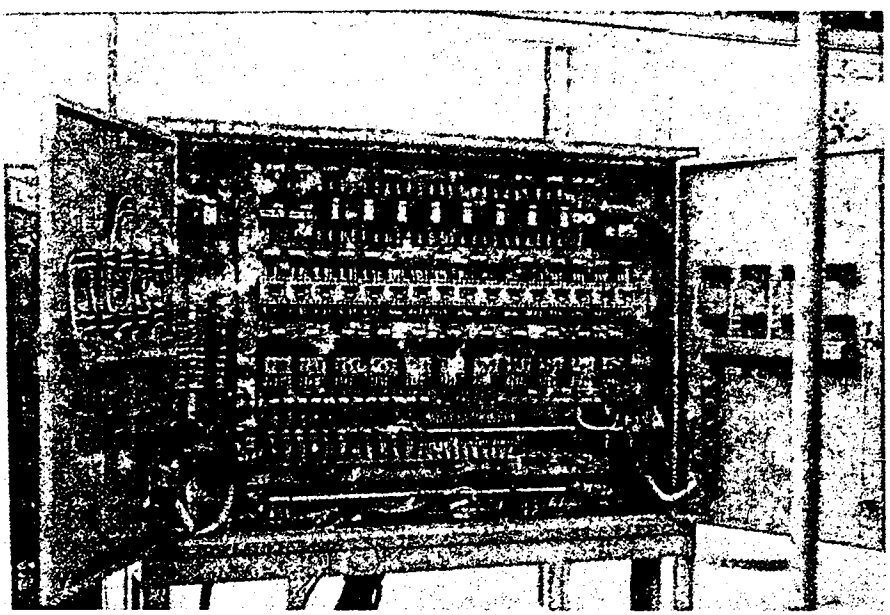


그림 4-5. 동력제어반

제 2 절 양액 공급시스템

양액은 원시표준액을 공급하였으며, 양액자동공급장치(Shany Ray사)를 통하여 오이의 경우 EC는 1.5~2.0, pH는 5.7 내외로, 토마토는 EC 1.0~1.5mS/㎝, pH 5.5~6.0 내외로 생육단계별 조정하면서 공급하였다. 그림 4-6은 본 연구에 사용된 양액공급시스템을 나타낸 것이다. 사용한 양액공급장치는 A, B액의 양액농축원액과 pH 조정을 위한 산이 들어있는 각각의 Tank에서 정량주입 펌프로 흡입하여 설정치대로 희석시켜 온실에 공급하는 방식이었으며 시퀀스제어로서 최대 28곳에의 공급이 가능하고, 1회공급시 최소공급량은 100ℓ였다. 양액 공급은 오이와 토마토 모두 생육전반기에는 1일 주당 0.8ℓ, 생육중반 이후에는 1.7ℓ를 공급하였다. 따라서, 생육기간 동안 공급된 양액은 1기작의 경우 오이는 주당 183ℓ, 토마토는 159ℓ였다. 사용양액의 성분별 조성은 표 4-4와 같다.

양액은 점적식 튜브(Supertyphoon, Netafim)를 베드당 2열 배치하여 이를 통해 공급하였다. 본 연구에서는 배지내 수분센서, 지중온도센서를 이용하여 배지내의 환경을 측정하였다.

양액의 공급시기와 공급량 및 공급양액의 pH, EC는 모두 file로 저장시켜 검색이 가능하도록 하였다.

표 4-4. 사용양액

성 분	NO_3	NH_4	P	K	Ca	Mg
함 량 (me/ℓ)	14	1.0	3	6	8	4

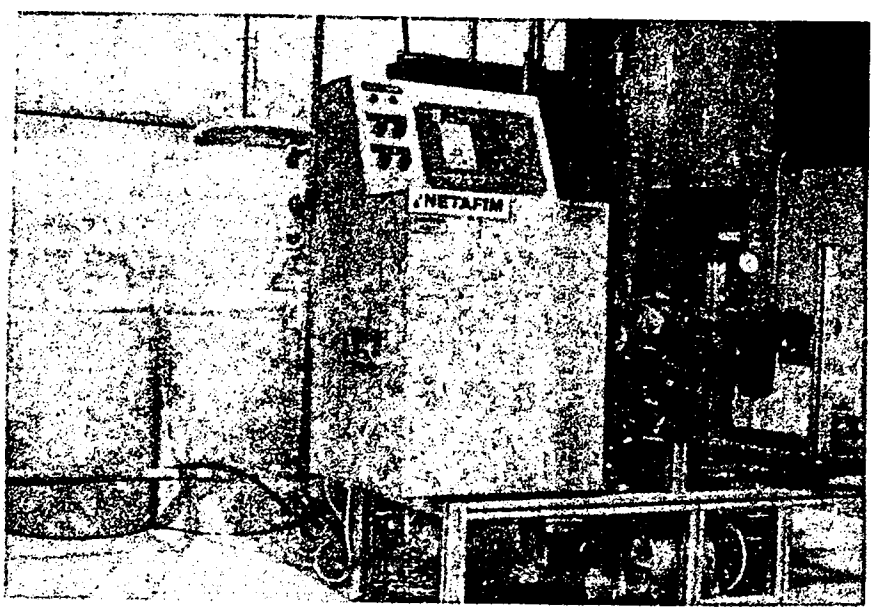


그림 4-6. 양액 공급시스템

제 3 절 공시작물 및 재배방식

1. 공시작물 및 경종개요

본 연구에서는 공시작물로서 토마토, 오이를 이용하였으며, 재식거리는 오이와 토마토 모두 주간 거리를 20cm로 하여 베드당 1열로 정식하였으며, 10a당 재식주수는 총 2500주가 되도록 하였다. 작물은 3회재배하여 실험을 수행하였으며 그 경종개요는 표 4-5와 같다.

시험기간동안 작물의 생육은 정상적이었으며 재배법은 미니토마토의 경우 10단적심재배를, 오이의 경우는 주지착과를 원칙으로 사면유인을 하면서 재배하였다. 2기작 토마토는 '96년 1월 3일 정식하여 6월까지 6개월간 무적심재배하였다. 기타 재배법은 표준영농법에 준하였다.

표 4-5. 오이와 토마토의 경종개요

재배	작물	파종	정식	품종명	비고
1기작	오이	'95. 5. 22	'95. 6. 23	장일반백	
	토마토	'95. 5. 22	'95. 6. 29	빼빼	10단 재배
2기작	오이	'95. 12. 3	'96. 1. 9	겨울살이청장	
	토마토	'95. 11. 11	'96. 1. 3	모모타로	
3기작	오이	'96. 9. 1	'96. 9. 24	입추낙합	

2. 재배방식 및 베드 시설

베드의 모양이나 높이는 작물의 종류에 따라서 작업능률 등을 고려하여 결정한다. 뿌리량이 많고 키가 큰 과채류에서는 근권산소 수준과 완충력이 순수수경에 비하여 높고 지지성이 좋은 배지경이 일반적이다. 따라서, 본 연구에서도 과채류재배를 위하여 배지재배 방식을 택하였으며, 양액은 재순환하지 않고 흘려버림식으로 하였다. 흘려버림식 양액재배는 생산비 증가와 환경오염을 야기시키므로 앞으로는 폐쇄형 순환시스템으로의 발전이 요구되고 있다.

베드는 폭 0.5m, 두께 0.025m, 길이 1.2m의 ㅁ자형 폴리스티렌 베드 7장을 연결하여 폭 0.5m, 깊이 0.2m, 길이 8.4m의 성형 베드를 각 온실마다 3열씩 설치하였다. 그림 4-7은 베드의 단면도를 나타내며, 그림 4-8은 베드의 형태를 나타낸다.

베드 하단에는 배수관을 1열로 배열하고, 한쪽 끝에 배수구를 만들어 배수되도록 하였다. 배수관 위에는 방근투수막을 베드 전체에 씌워 배수관으로 배지와 뿌리가 들어가지 않도록 하여 배수가 양호하도록 하였다. 배수구를 통해 배출된 여액은 집수통에 모이고 일정량이 집적되면 자동수위센서에 감지되어 배수펌프에 의해 배출되도록 하였다.

3. 사용배지

본 연구에 이용된 배지는 퍼얼라이트와 훈탄을 7:3의 비율로 혼합하여 사용하였다. 퍼얼라이트는 배지의 통기성을 증가시키기 위하여 많이 사용되고 있다. 퍼얼라이트의 중요한 장점으로는 무게가 가벼워 수송면에서 유리하다는 것이다. 퍼얼라이트의 무게가 96~128 kg/m^3인데 비해 모래는 1,600 kg/m^3로서 약 20배의 차이가 있다. 퍼얼라이트는 규소 성분을 함유하고 있는 화산암을 잘게 부순 후 약 928℃에서 고온처리하여 흰 입자로

팽창시킨 것으로, 입자의 중간에는 가장자리가 막힌 많은 공극을 가지고 있다. 퍼얼라이트는 고온처리되어 무균상태이고, 화학적으로 불활성이며, 양이온 치환용량(CEC)은 0.15me/100cc로 무시할 수 있는 수준이다.

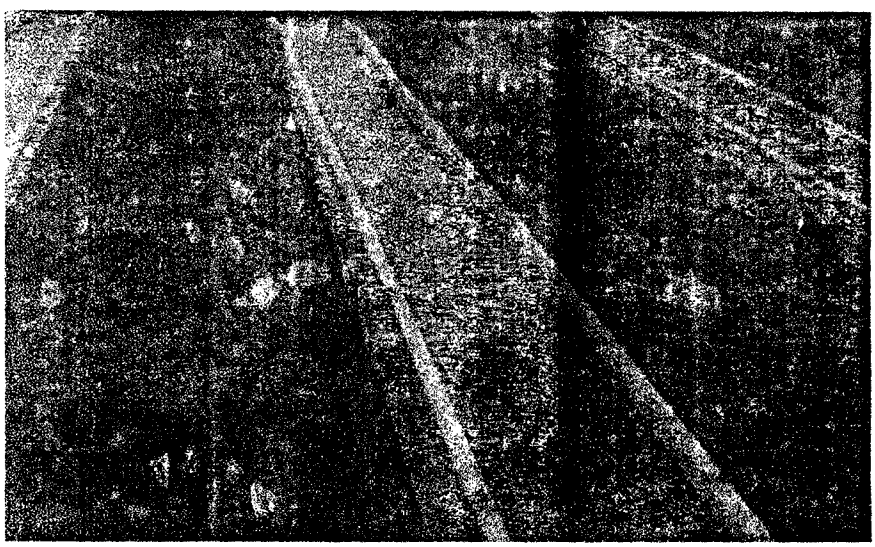


그림 4-7. 베드의 형상

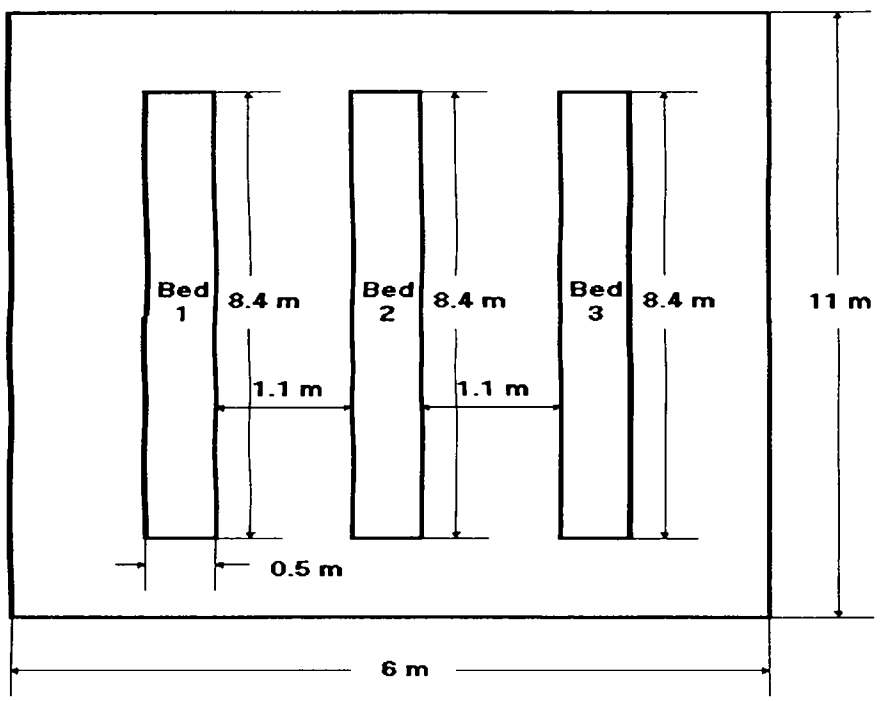


그림 4-8. 베드의 단면도

제 4 절 온실 내·외부 환경 계측시스템

그림 4-9는 본 연구를 수행하기 위하여 각 온실에 설치된 복합환경 제어 시스템이다.

온실 내외부의 환경요인을 측정하는 센서와 작물의 생체정보를 측정하는 센서는 이스라엘 Shany Ray사의 Phytomonitoring 시스템이었다.

온실내의 환경요인을 측정하기 위해서 온도, 습도, 일사량 센서, CO_2 농도 센서를 2개 동의 온실에 설치하였다.

온·습도 센서는 그림 4-10과 같이 일체형으로 이루어져 있으며, 온도의 측정은 정밀 열전대를 이용하고, 습도는 저항형 센서를 이용하였다.

그림 4-11은 온실내부의 일사량을 측정하는 센서를 나타낸다. 일사량 측정센서는 설치 위치에 따라 온실 구조물의 영향을 받게 되므로 태양의 위치에 따라서 수광부의 이동이 용이하도록 하였다.

외부 기상조건을 측정하기 위하여 일사량, 온도, 습도, 풍향, 풍속 및 강우감지 센서 등으로 구성된 외부 기상관측장치를 지상 5m에 설치하였다 (그림 4-12). 풍향 및 풍속 측정센서는 초음파를 이용하여 풍향과 풍속을 측정한다. 발진부에서 일정한 주파수로 초음파를 보내면 바람의 방향 및 속도에 따라 감지부에 감지되는 초음파의 주파수가 변화하도록 설계되어 있다.

온도, 습도 센서 및 강우감지 센서는 일체형으로 구성되어 있다. 온도와 습도의 측정은 온실의 내부에 사용된 센서와 같은 원리를 사용하였으며, 강우량은 빗물이 센서를 통과하는 유속을 감지하여 단위시간당 강우량을 측정할 수 있도록 설계되어 있다.

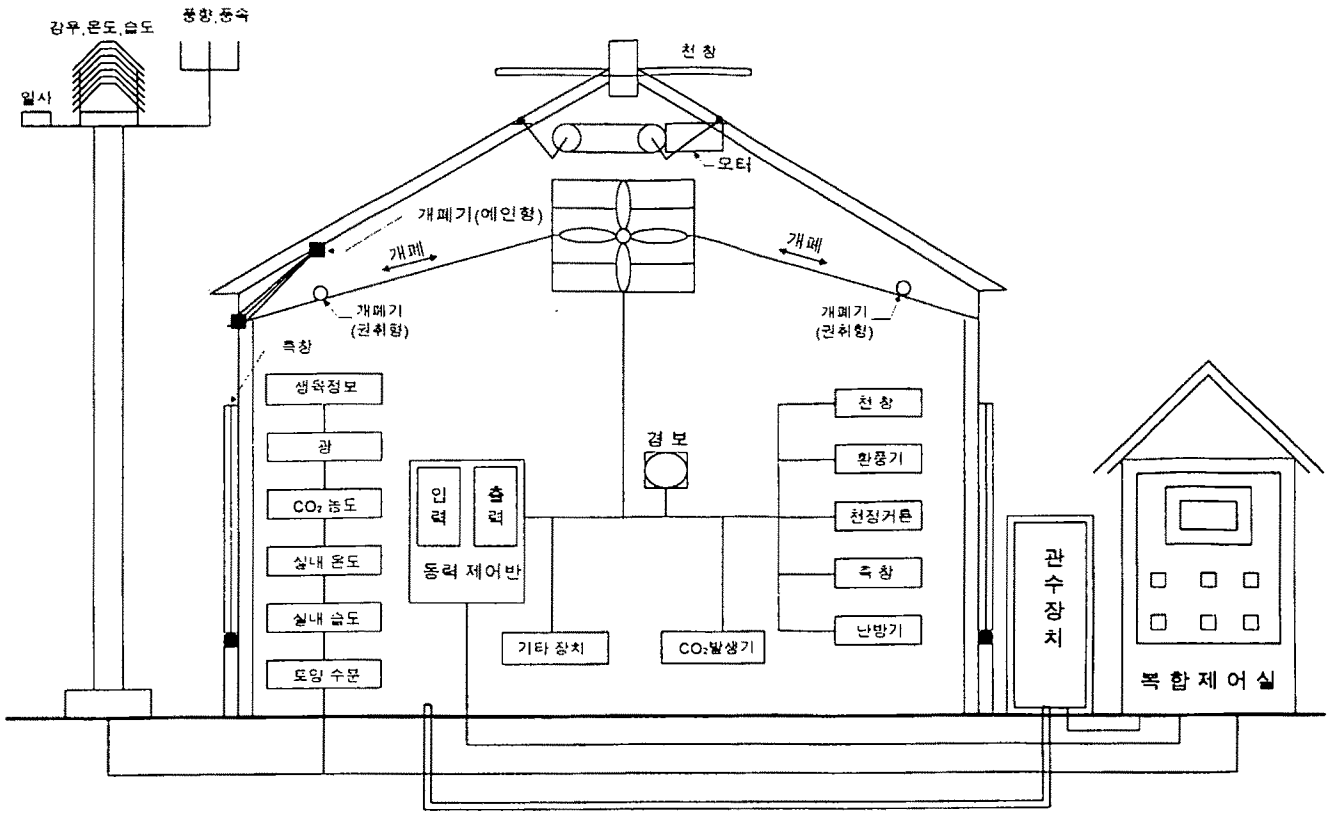


그림 4-9. 복합환경 제어시스템의 구성

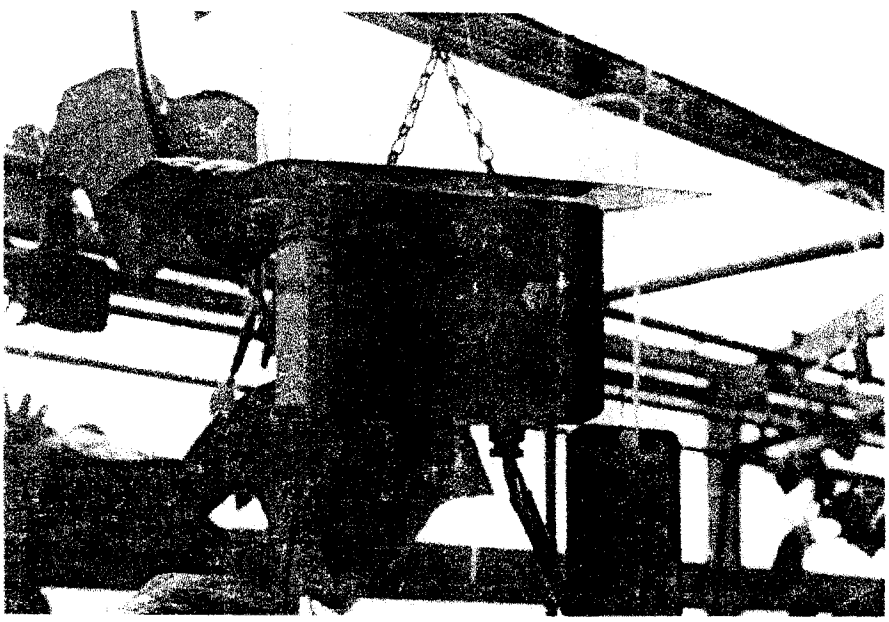


그림 4-10. 실내 온·습도 센서

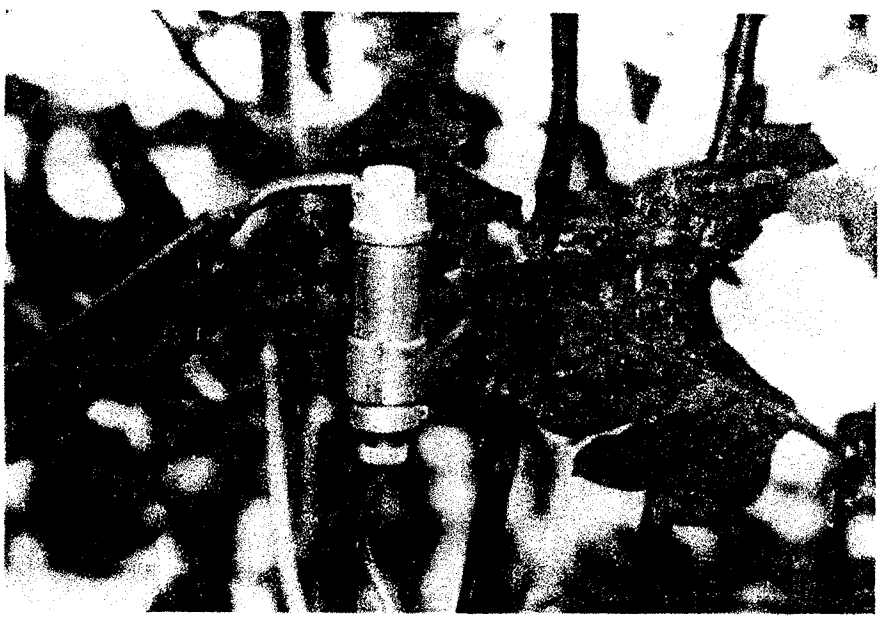


그림 4-11. 실내 일사량 센서

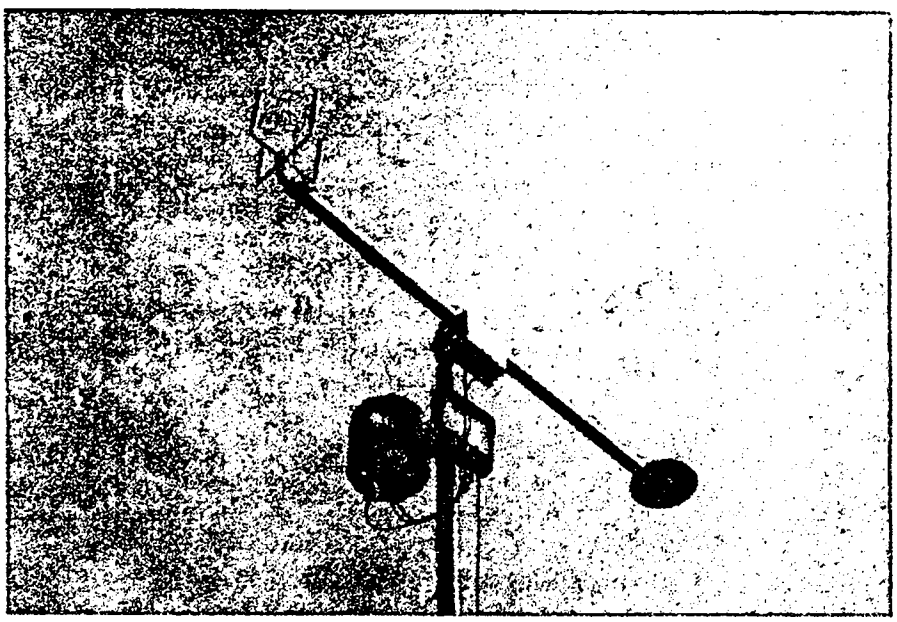


그림 4-12. 외부 기상관측 장치

제 5 절 생체정보 계측시스템

생체정보의 계측방법은 파괴적 방법과 비파괴적 방법으로 구분할 수 있다. 온실의 경우, 복합환경 제어에 이용하기 위해서는 연속적으로 일정시간 이상의 측정이 필요하고, 측정에 의하여 작물의 생장이 영향을 받지 않아야 하므로 비파괴적 계측방법을 적용하여야 한다. 비파괴적 계측방법은 접촉식(contact method)과 비접촉식(non-contact method)으로 나눌 수 있다.

접촉식은 센서를 직접 식물체에 접촉시켜 생체정보를 측정하는 방법으로서 Cate등에 의하여 'Speaking plant' 기법으로 정의되었으며, 이를 이용한 식물생장의 on-line process control이 발달되고 있다. 그러나 접촉식 생체정보 계측방법에서는 전식물을 측정대상으로 삼을 수 없으므로 측정대상 군락에서 군락의 평균치를 잘 나타내는 대표식물로서 몇 개를 선정하는 것이 어렵다.

비접촉식은 이와 반대로 식물체와 접촉없이 생체정보를 측정하는 방법으로서 Takakura에 의해 'Reading plants faces' 기법으로 불리어지고 있다. 일반적으로 비접촉식은 CCD카메라에 의한 image process와 gas balance 법 등을 사용하는 방법을 말한다.

본 연구에서는 온실의 환경에 따른 작물의 생체정보를 측정하기 위하여 접촉식 생체정보 측정센서를 설치하였다. 2개 동의 온실에 설치된 생체정보 측정센서는 증산류(stem water flux) 측정센서, 줄기직경(stem diameter) 측정센서, 과일직경(fruit diameter) 측정센서, 엽온(leaf temperature) 측정센서가 있다. 각 센서에서의 입력신호는 그림 4-13와 같은 중간변환단계를 거쳐 복합환경 제어실의 컴퓨터에 입력되도록 설계되었다.

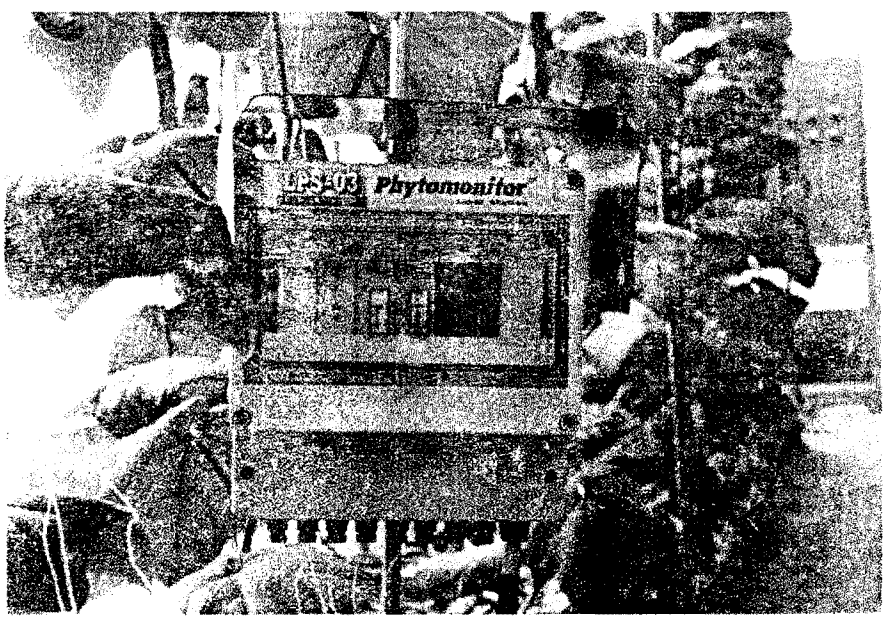


그림 4-13. 생체정보 센서의 중간변환 장치

증산류 측정센서(모델 IVP-5)는 직경이 1~10mm인 작물의 증산류를 측정할 수 있는 센서이다(그림 4-14). 이 센서는 센서의 중앙에 열을 가하고 열원으로부터 동일지점 상하부의 온도차를 측정하여 수분이동정도를 측정할 수 있도록 설계되어 있다. 열원에 요구되는 전력은 30mW이며, 시간당 4~5ml의 수분 이동량을 측정할 수 있다.

줄기직경 측정센서와 과일의 크기를 측정할 수 있는 과일직경 측정센서는 여러 작물의 줄기와 과일의 기준값에 대한 변화량의 미세한 측정이 가능하다. 센서는 LVDT(Linear Variable Differential Transformer)의 기본원리를 이용하여 줄기와 과일직경의 변화량을 측정한다. 줄기직경센서(모델 SD-5)의 측정범위는 0~2,000mcm이며, 측정 가능한 줄기의 직경은 4~10mm이다(그림 4-15). 과일직경 측정센서(모델 FD-3, FD-4)는 과일의 가로 절단면이 원형인 과일의 크기를 측정하는 센서이다. 모델 FD-3은 과일의 직경이 20~100mm인 구형(round type)과일의 크기를 측정하는 센서로서 토마토의 크기를 측정하는데 사용하였으며(그림 4-16), 측정범위는 0~5,000mcm이다. 모델 FD-4는 직경이 10~40mm 범위의 오이와 같은 원통형(cyliderical type) 과일의 크기를 측정하는데 사용하였으며(그림 4-17), 측정범위는 0~5,000mcm이다. 각 센서의 정확도는 ±0.5%로서 매우 정밀한 측정이 가능하였다.

엽온측정센서(모델 LT-1)는 일정한 환경조건 하에서 작물 잎의 표면온도를 측정하도록 설계되었다.(그림 4-18). 엽온측정센서는 반도체 서미스터(thermister)가 잎 표면에 부착되어 엽온을 직접 측정할 수 있다. 엽온측정센서의 측정범위는 5℃~45℃이며 정확도는 ±0.5%이다.

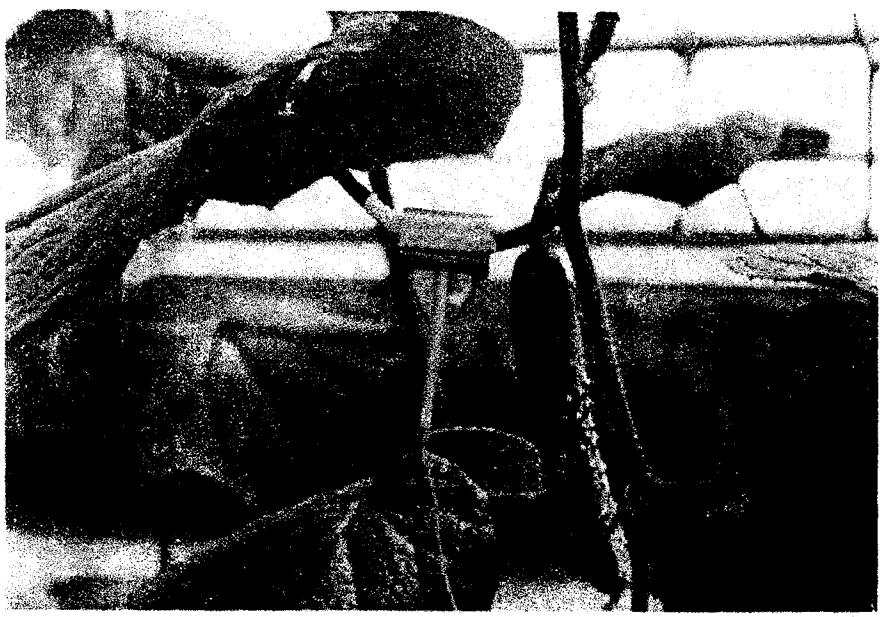


그림 4-14. 증산류 측정 센서

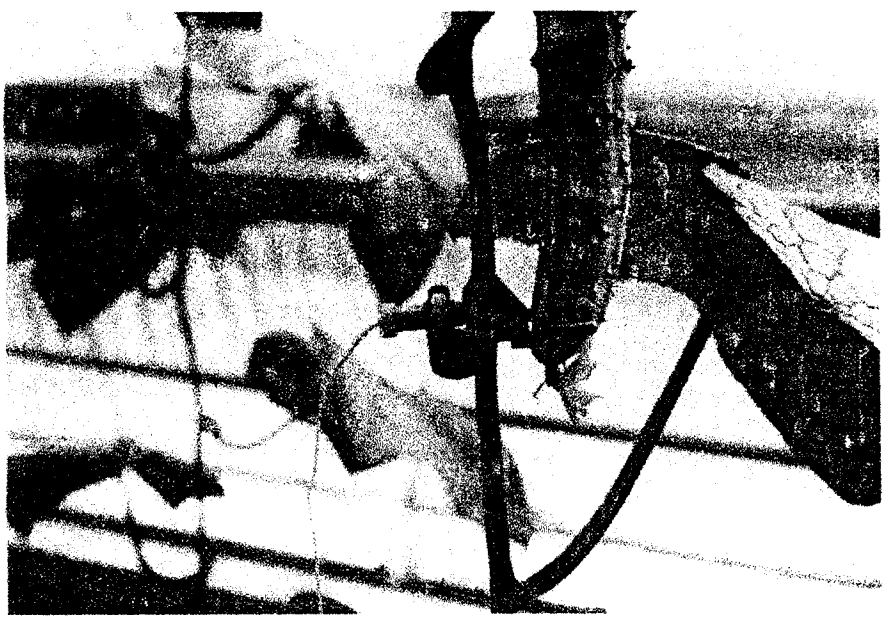


그림 4-15. 줄기직경 측정 센서

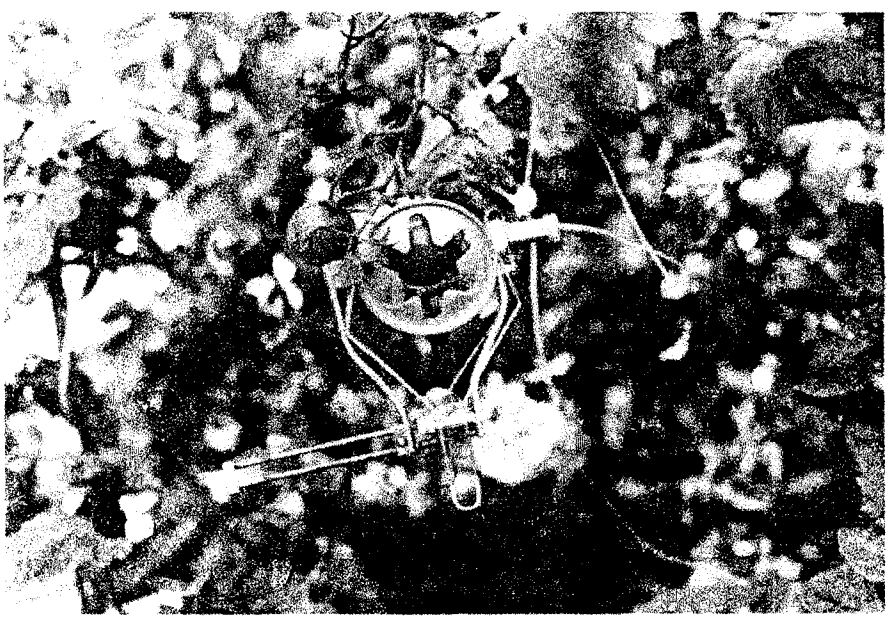


그림 4-16. 구형과일(토마토) 직경 측정 센서

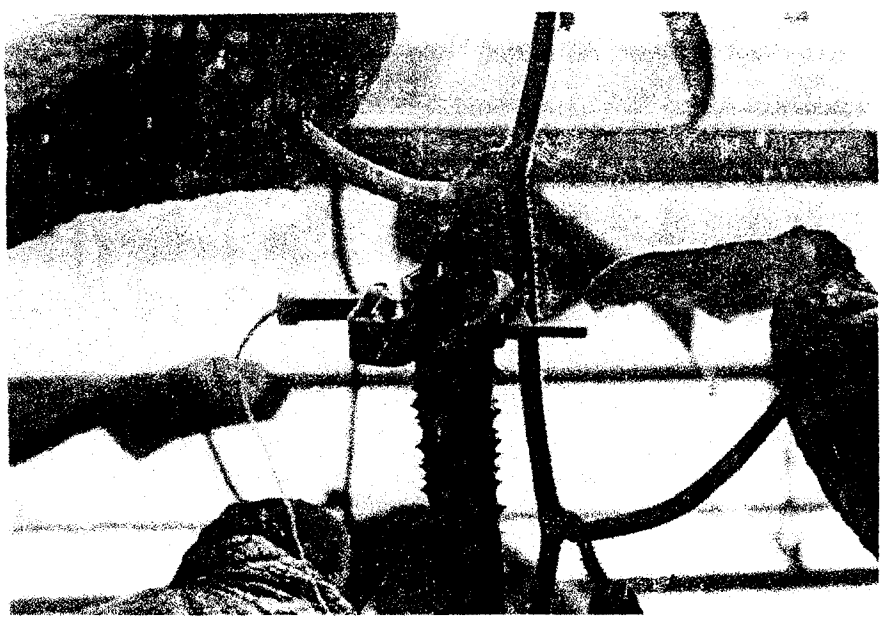


그림 4-17. 원통형 과일(오이) 직경 측정 센서

그림 4-18. 엽온 측정 센서

제 6 절 복합환경 제어시스템

1. 하드웨어

하드웨어는 각각의 가능에 따라서 모듈로서 분리를 해 놓았다. 각 모듈들은 각각의 독립적인 회로를 이용하여 개별적으로 구동이 되도록 하였다. 각 모듈들의 완전독립적인 구동을 위하여 각각의 모듈안에 설치되어있는 회로에는 원칩을 부착시켰다. Main 모듈의 경우에는 80c196 칩을 이용하였고 입력 모듈, 출력 모듈, 디스플레이 모듈, 센서 모듈 등의 Sub 모듈들에는 80c51 칩을 이요하였다.

각각의 칩의 구동을 위한 프로그램은 C언어를 주로 이용하였으며, Main 모듈의 경우에는 필요에 따라서 Assembler 언어를 이용하였다. PC를 이용한 원칩구동용 프로그램은 ROM Compiler(96 Compiler 등)와 ROM Writer를 이용하여 원칩에 저장을 시켜서 사용하였다.

원칩구동용 프로그램의 작성시에는 프로그램 에러나 버그 등을 찾아내기에 상당한 어려움이 있기 때문에 PC 두 대를 이용하여 반복실험을 시행한 후 원칩에 Load시켜 원칩구동의 실험을 실시하였다.

Main 모듈은 80c196 칩과 전원을 ON 시켰을 때 시스템의 부팅을 위한 Booting ROM, Booting RAM을 장착시켰으며, PC에서 지정을 해주는 각종 예약값들을 저장시켜놓을 수 있는 RAM과 수집된 자료의 저장을 위한 Gathering RAM을 포함시켰다. Gathering RAM은 순환방식을 이용하여서 일정분량의 자료를 수집한 후에 RAM 의 용량이 다 채워지면 가장먼저 수집된 자료들을 지우고 새로운 자료들을 입력시키는 형태로 제작하였다. 자료의 수집과정에 있어서는 사용하지 않거나 필요하지 않은 자료들은 제외하고 수집을 함으로써 RAM의 용량을 최대한 사용할 수 있도록 하였다.

Main 모듈에는 PC(또는 다른 Main 모듈)와의 통신과 연결되어 있는

Sub 모듈과의 통신을 원할히 할 수 있게 하고 통신과정에서의 간섭을 줄이기 위하여 시리얼 통신포트를 2개씩 장착하였다. 통신은 RS232C 와 485 통신를 이용하였으며, 에러체크는 CRC와 BCC를 이용하였다. 하드웨어상에서의 통신은 C 언어를 이용하여 프로그램하였다.

하드웨어의 동작을 표시해주기 위하여 상황표시 LED Lamp를 이용하여 작동의 유무를 체크할 수 있게 하였다. 센서값 등의 아날로그 신호를 입력받기 위하여 12 bit의 분해능을 갖는 A/D Converter를 사용하였고, 작동기기들의 구동을 위한 디지털신호의 입력과 출력에는 한 개의 모듈에 24개의 포트를 설치하여 사용하였다.

시스템의 구동시에 정전 등에 의해서 갑작스럽게 전원이 차단되는 경우에 대비하여서 UPS 기능을 추가하여 비상시에 전원을 일정기간 동안 공급할 수 있게 하였으며, 비상시에 UPS의 구동시에는 평상시와는 다른 제어 Logic을 이용하여 전력의 소비를 최소화 하였다.

2. 소프트웨어

본 시스템의 구동을 위한 PC용 프로그램을 위하여서 Windows상에서 작동할 수 있고 멀티 테스킹(multi-tasking)을 가능하게 하기 위하여 마이크로 소프트사에서 나오는 Visual Basic을 이용하였다.

본 소프트웨어의 제작은 MS Windows 3.1을 기준으로 하였으며, 통신의 사양은 하드웨어와 동일하게 하였다. 시스템 구동용 프로그램의 구성은 Main화면과 현재상황 모드, 자료수집 모드, 수동 모드, 환경설정 모드, 하드웨어 설정 모드 등의 5개의 블록으로 구성하였고, 이들은 동시에 작업이 가능하도록 하였다.

온실내 환경의 제어를 위하여서 기존의 작동기기별로의 타이머와 온도설정에 의한 제어가 아니고, 온실내의 온도와 습도를 기준으로 작동기기들의

구동을 순차적으로 제어를 해 줄 수 있게 하여서 온실내 환경을 전체적으로 조절할 수 있게 하였다.

본 소프트웨어의 구동은 16bit 이상의 IBM PC 를 이용하였다. 본 소프트웨어의 제작에서는 Windows의 원활한 구동과 통신의 속도장애를 최소화 하기 위하여서 486급 이상의 PC를 이용하였다.

제 5 장
온실 환경 제어방법 개선을 위한 생체정보 분석

제 1 절 생체정보 수집

일반적으로 복합환경제어는 환경요인만을 계측하여 이를 토대로 환경요인 자체를 제어한다. 그러나, 본 연구에서는 작물의 생육반응, 즉 생체정보를 분석하여 환경을 제어하므로 재배자가 의도하는 최적의 작물생육을 이룰 수 있는 진정한 의미의 복합환경제어를 구현하고자 작물생체정보 이용 온실환경관리 소프트웨어 구축을 위한 기초 조사·분석을 하였다. 이를 위하여 작물의 생육에 영향을 미치는 환경요인과 작물의 생육상황을 나타내는 생체정보를 수집하고, 온실내 환경요인의 변화에 의한 작물의 생체반응을 모델링하였다.

1. 자료 수집 장치

본 연구에 이용한 자료수집 장치는 이스라엘 Shany-Ray사의 Phytomonitoring 장치였으며, 전체적인 구성은 자료 수집부, 제어부, 자료분석부로 이루어져 있다.

자료 수집부는 온실 내·외부에 설치된 각 센서들의 측정신호를 콘트롤러를 통하여 컴퓨터로 입력하도록 설계되었으며, 수집된 자료는 데이터 베이스화되어 저장되고 자료는 외부 출력기를 통하여 프린트할 수 있도록 구성되었다. 본 연구에서는 이와 같이 작물의 전생육기간에 걸쳐서 24시간 온라인으로 자료 수집부에 저장된 데이터들을 환경변화에 따른 생체반응을

모델링하는데 이용하였다.

제어부에는 온실의 각 작동기와 양액 및 관수시스템을 제어하기 위한 제어 조건을 입력하고, 입력된 제어조건과 자료수집부에서 측정된 온실의 환경 조건에 따라 제어신호를 출력하도록 되었다.

자료분석부에서는 수집된 자료를 바탕으로 온실 내·외부의 환경 변화와 작물의 생육상황을 분석할 수 있도록 프로그램되어 있다. 사용자가 분석하고자 하는 온실, 날짜와 시간, 분석 항목을 설정하게 되면 온실 내·외부의 환경 변화와 작물의 생육현황을 통계분석표와 그래프를 통하여 분석 결과를 보여주며, 필요에 따라 분석 결과를 프린트로 출력할 수 있다.
따라서 본 연구에서는 자연적 기후조건 아래에서 일반적인 온실환경관리 조건을 부여하고 이때 생장한 토마토, 오이의 각종 생체정보를 시스템을 통하여 측정, 수집하였다. 그리고 수집된 환경요인들과 상호분석하여 환경요인에 따른 작물생체반응의 기본 모델링을 시도하였다.

2. 자료 수집 항목

가. 환경 요인

측정한 환경 요인은 실내·외 온습도, 실내·외 일사량, 풍향, 풍속, 강우, 지중 온도등이었다. 본 연구에서는 시설작물은 주로 제어된 시설내에서 생장하므로 외부 환경 요인들 즉, 실외 온습도, 풍향, 풍속, 강우 등은 분석항목에서 배제하였고, 실내 온·습도, 일사량을 주요 환경요인으로하여 분석에 이용하였다.

나. 생체 정보

지금까지 생장정보 즉 식물체의 초장, 엽면적, 건물 중량, 생체중 등의 측정은 대부분 개체를 대상으로 수작업에 의한 파괴적인 방법으로 수행되어 왔다.

따라서 본 연구에서는 비파괴적이고 연속적으로 온실의 환경에 따른 작물의 생체정보를 측정하기 위하여 생체정보 측정 센서를 설치하고 데이터들을 수집했다. 온실에 설치된 생체정보 센서는 증산류(stem water flux) 측정센서(모델 IVP-5), 줄기직경(stem diameter) 측정센서, 과일직경(fruit diameter) 측정센서, 엽온(leaf temperature) 측정센서(모델 LT-1)등이다. 이 센서들로부터 계측된 값들은 자료 수집 장치에 실시간 온라인으로 저장되었다.

각각의 생체정보측정 센서들은 작물이 자람에 따라 적당한 위치로 이동시켜 설치하였는데, 특히 오이과일직경 센서는 센서자체의 총 계측범위가 오이 과일의 1일 비대량에 못미쳐 매일 측정범위를 재조정 하였다.

증산류 측정센서는 센서크기에 맞추어 엽병에 설치하여 측정하였고, 줄기직경 센서는 7~10일 간격으로 식물체 상부로 이동시켜 계측하였다.

표 5-1은 생체정보 수집 대상 공시작물인 토마토와 오이의 생육 및 수량을 나타낸다.

표 5-1. 생체정보 수집대상 토마토, 오이의 생육 및 수량

재배	작물	초장 (cm)	엽수 (매/주)	엽면적 (㎠/주)	경경 (mm)	생체중 (g/주)	과중 (g/주)
1기작	오이	940.7	80	5504.3	7.36	566.7	2458
	토마토	271.3	35	7618.3	9.30	763.3	1750
2기작	오이	707.7	86	-	-	523.3	2656
	토마토	565.0	53.7	-	-	1636.6	2788

제 2 절 환경과 생체반응

다동온실내 복합환경의 컴퓨터제어 시스템 개발을 위한 본 연구에서는 그것의 하드웨어 개발에 주력하면서도 기존의 온실 환경관리 개념보다 한 차원 앞선 생체정보를 이용한 온실환경관리 소프트웨어 구축의 기초로서 환경요인과 생체반응의 관련성을 분석하였다.

측정된 자료는 모두 수집되어 분석에 이용하였으며, 환경요인은 온실내 온·습도, 외부일사량의 3가지를 생체정보와 비교 분석하였다. 온실 내부 일사량은 온실골조 등의 그늘로 말미암아 센서값이 달라져 작물이 받는 수광량을 정확히 파악하기 곤란하여 외부일사량을 이용하였다. 분석은 통계분석 프로그램 SAS를 이용하였으며, 이들 생체정보와 환경요인의 관계는 직선적인 관계가 있는 것으로 보아 다중회귀분석을 Stepwise법으로 하였다. 다중회귀분석시 변수상호간의 공선성을 파악하기 위해 다중공선성도 아울러 분석하였다.

1. 오이 생체정보 분석

오이 생체정보분석시 이용한 변수의 개요는 표 5-2와 같았다. 오이 생체정보 분석은 '96년 반촉성재배 작형에서 수집된 자료를 바탕으로 하였으며, 품종은 청장계통의 겨울살이청장(홍농종묘)이었다. 표 5-3은 환경요인과 생체정보간의 상관을 나타낸다.

엽온은 기온에 가장 큰 영향을 받았는데 대체로 기온보다 낮은 경향이었고 특히 한낮의 고온시에는 잎표면의 증산작용에 의한 증발냉각효과로 엽온이 기온보다 3℃ 이상 낮았다(그림 5-3). 과경과 경경은 기타의 요인들과 상관관계가 매우 낮거나 아예 통계적 유의성이 없는 상관을 나타내었다.

표 5-2. 오이생체정보 분석시 이용한 변수 개요

변 수 명	최 소 치	최 대 치	측정기간
기 온 (℃)	14.7	32.5	'96.3.20 ~ '96. 4.4
습 도 (%)	27	99	
엽 온 (℃)	13.1	29	
1회측정 과경 (mcm)	0	5,000	
1회측정 경경 (mcm)	0	2,000	
증 산 류	1.29	5.0	
일 사 량 (W/m²)	0	752	

표 5-3. 환경요인과 생체정보간의 상관

	습 도	엽 온	과 경	경 경	증산류	일사량
기 온	-0.276	0.915	0.069	-	0.540	0.649
습 도		-0.111	-	0.310	-0.188	-0.084
엽 온			0.109	0.130	0.367	0.557
과 경				-	-0.186	-0.152
경 경					-0.454	-0.166
증산류						0.863

이는 자료분석시 과경과 경경의 변화는 오랜 시간에 걸쳐서 일어나지만 기타 요인들의 변화는 단시간에 즉각적으로 나타나므로 상호간의 변화 양상에 대한 시간적 간격이 너무 컸기 때문으로 판단된다(그림 5-5, 5-6).

따라서 과경과 경경의 변화 반응시간을 무시한 무조건적인 상호비교는 이들 항목에 대해서는 큰 의미가 없는 것으로 보이며 정확한 상관관계를 파악하기 위해서는 변화 반응시간의 상호일치화가 필요한 것으로 생각되었다. 오이 과일의 경우 생장속도가 빠를 경우 20시간 내외에 센서의 최대 감지 범위인 5000mcm(micron meter, 5㎜)까지 과경증대가 일어났다. 이 기간 동안에도 과경은 지속적인 변화가 있었지만 10분이라는 기간동안의 변화량은 센서가 감지할 수 없었고, 센서의 최소 분해능(resolution)이 20mcm이었으므로 그만큼의 증대가 일어날 기간동안의 변화는 측정할 수 없어 그 기간은 변화가 정체된 것으로 여겨지기 때문에 상관분석의 결과가 아주 낮게 나타난 것이었다.

따라서 과경 또는 경경과 기타 요소간의 상관 등을 정확히 분석하기위해서는 변화에 반응하는 시간의 상호일치화 또는 미세한 변화의 폭까지도 감지해 낼 수 있는 감도높은 센서의 사용이 전제되어야 할 것이다.

증산류와 관계깊은 항목으로는 일사량, 온도, 경경 순이었는데 일사량과는 상관계수 0.863으로서 일사량이 많을수록 증산류 또한 높아지는 것으로 나타냈다. 반면 증산류와 경경 간에는 상관계수 -0.454의 부의상관을 보여 한낮에 증산류가 활발할수록 경경은 감소하는 경향이었다(그림 5-6).

증산류(엄밀히 말하면 증산작용에 의한 증산류+근압에 의한 수분이동 포함)는 대부분 증산작용에 의한 것인데 증산작용은 일사량에 가장 크게 영향을 받으므로 증산류와 일사량의 상관계수가 높게 나타난 것으로 여겨진다(그림 5-1). 따라서 증산류의 정도로서 작물 수분관리가 가능할 것으로 판단되었으며 실제 증산류를 이용한 작물 수분관리 자동화가 일부 선진

외국에서는 실용화되고 있는 실정이다. 증산류는 또한 온실내 포화수증기압차와 매우 밀접한 관계를 보였으며(그림 5-1), (기온-엽온)과도 유사한 경향을 나타냈다(그림 5-3). 즉 그림 5-4에서 나타난 바와 같이 온실내 포화수증기압차와 (기온-엽온)과는, 양자의 관계가 증산작용을 유도하고(포화수증기압차) 증산작용에 의한 엽온하강(기온-엽온)이라는 원인결과의 관점에서 밀접하게 나타났다. 증산류와 습도는 반대의 경향을 보였고 당연한 결과로 포화수증기압차와 습도 또한 반대의 경향이었다(그림 5-2).

엽온, 과경, 경경 그리고 증산류의 4가지 생체정보 중에서 증산류가 가장 생체변화를 즉각적으로 표현하고 또한 수분스트레스 정도를 포함하여 작물체의 생육상태를 가장 잘 표현하는 것으로 판단되었다. 따라서 증산류를 종속변수로한 직선회귀모델의 구성이 가능하였으며, 구성한 회귀모델을 온실 환경제어 소프트웨어에 이용할 수 있을 것으로 판단되었다.

표 5-4는 오이의 과경추정 회귀모형을 나타며, 표 5-5는 Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력을 나타낸다.

과경을 추정할 수 있는 다중회귀식을 도출할 수 있었으나 과경과 기타 요인들간의 상관이 매우 낮고, 앞서의 이유들처럼 비교시 근본적인 차이가 있으므로 회귀모델의 결정계수는 매우 낮아 통계적인 유의성은 존재하는 것으로 나타났으나 현실성은 거의 없었다.

경경의 경우 과경보다는 모델의 결정계수가 높아 증산류, 일사량, 습도 3요인을 독립변수로 하였을 경우 0.43으로 나타났다. 그러나 이 또한 회귀식의 전체변수 설명력이 43%에 불과한 것이어서 현실성은 낮은 것으로 보였다. 특히 경경은 일정한 크기까지는 증대하나 그것의 일일 변화양상은 증산류의 이동이 많은 한낮에는 약간 감소하고 야간에는 증대하는 것이 전형적인 변화양상이었다(그림 5-6).

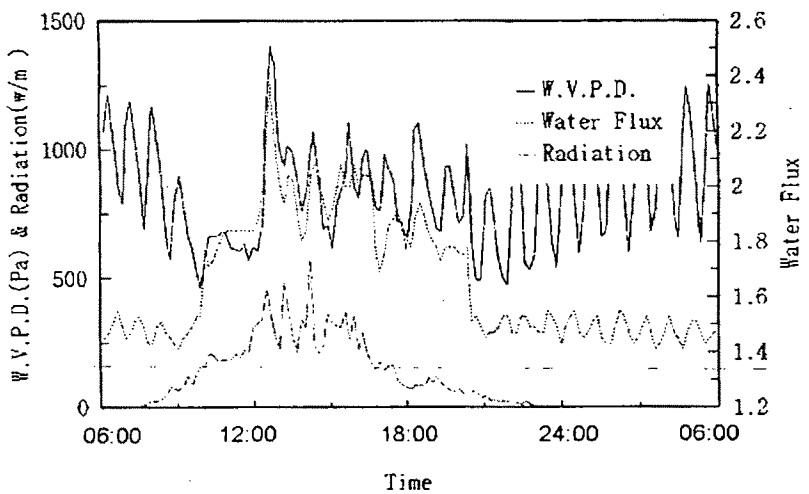


그림 5-1. 온실내 포화수증기압차 (Water Vapor Pressure Defficient) 오이 증산류(water flux) 및 일사량의 일변화

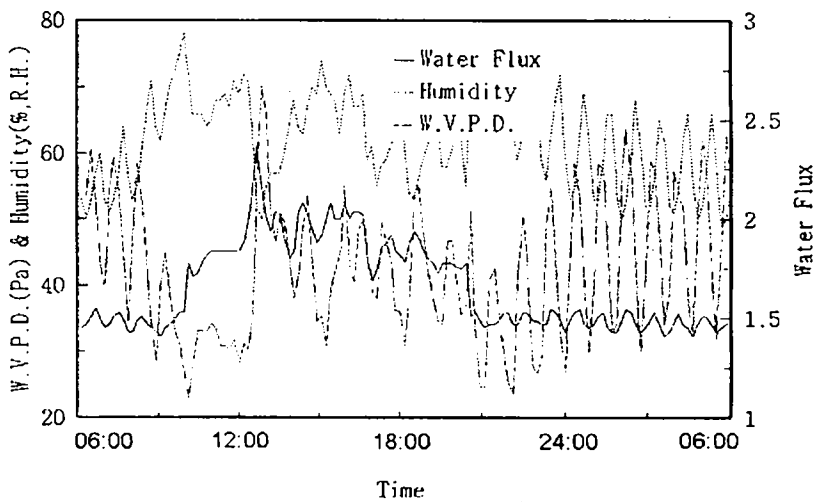


그림 5-2. 온실내 포화수증기압차(Water Vapor Pressure Defficient) 습도 및 오이 증산류의 일변화

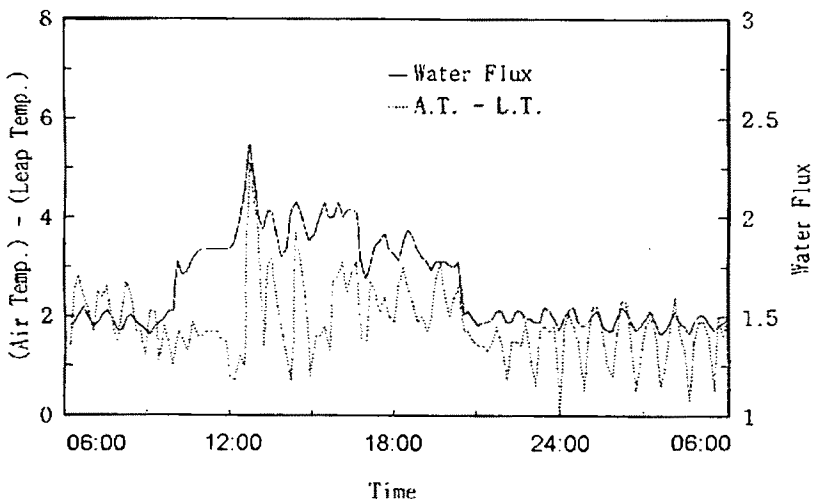


그림 5-3. 오이의 증산류와 (기온-엽온)의 일변화

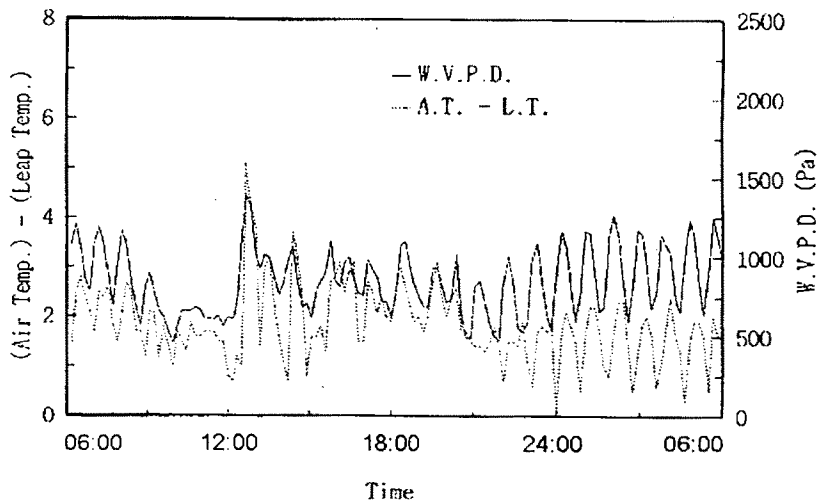


그림 5-4. 오이재배시 포화수증기압차와 (기온-엽온)의 일변화

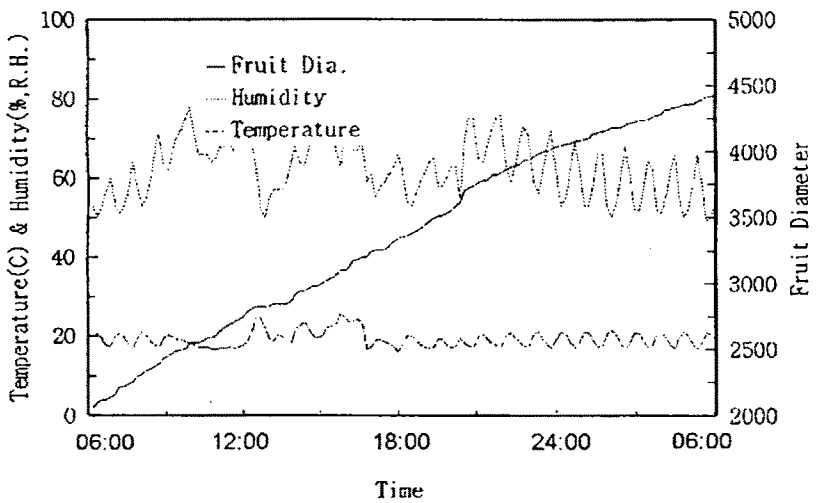


그림 5-5. 오이의 과경, 습도 및 기온의 일변화

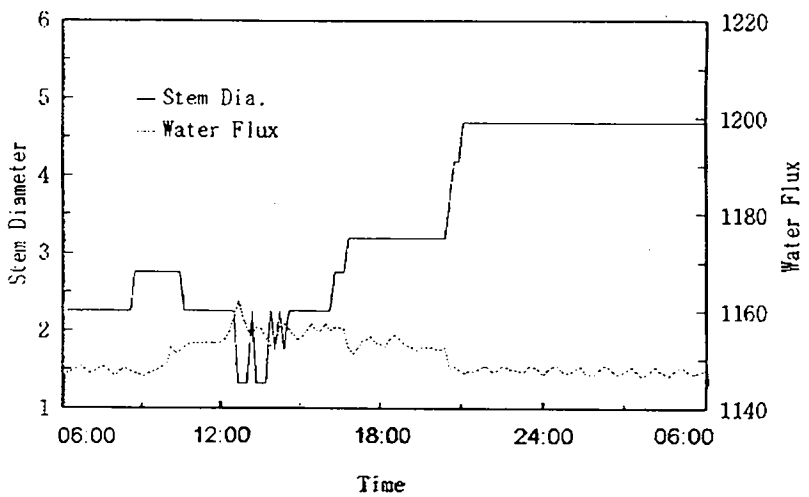


그림 5-6. 오이의 경경과 증산류의 일변화

표 5-4. 과경추정 회귀모형

요인수	회 귀 식
1	과경 = 3217.0 − 328.73 × 증산류
2	과경 = 1143.8 + (125.69 × 기온) − (556.64 × 증산류)
3	과경 = 478.5 + (148.37 × 기온) − (342.57 × 증산류) − (1.238 × 일사량)

표 5-5. Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력

Step	삽입변수	Partial R^2	모델 R^2	$C_{(P)}$	Prob > F
1	증 산 류	0.0346	0.0346	50.26	0.0001
2	기 온	0.0404	0.075	11.95	0.0001
3	일 사 량	0.0058	0.0808	8.17	0.0166

표 5-6. 경경추정 회귀모형

요인수	회 귀 식
1	경경 = 1204.2 − 188.2 × 증산류
2	경경 = 1565.8 − (505.04 × 증산류) + (1.538 × 일사량)
3	경경 = 1300.9 + (3.929 × 습도) − (473.86 × 증산류) + (1.449 × 일사량)

표 5-7. Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력

Step	삽입변수	Partial R^2	모델 R^2	$C_{(P)}$	Prob > F
1	증 산 류	0.2057	0.2057	378.28	0.0001
2	일 사 량	0.1992	0.4049	56.75	0.0001
3	습 도	0.0259	0.4308	16.60	0.0001

표 5-8. 증산류추정 회귀모형

요인수	회 귀 식
1	증산류 = 1.325+0.0036×일사량
2	증산류 = 2.0137−(0.0008×경경)+(0.0033×일사량)
3	증산류 = 2.073−(0.0013×습도)−(0.00075×경경) +(0.0034×일사량)

표 5-9. Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력

Step	삽입변수	Partial R^2	모델 R^2	$C_{(P)}$	Prob > F
1	일 사 량	0.7454	0.7454	582.5	0.0001
2	경 경	0.0988	0.8442	4.74	0.0001
3	습 도	0.0005	0.8446	3.95	0.0952

증산류를 추정할 수 있는 다중회귀모델은 높은 설명력을 보여 일사량 단요인을 독립변수로 하였을 경우 결정계수 0.745, 2요인 이상에서는 0.84로 나타났다. 3요인을 독립변수로 하였을 경우에는 2요인 삽입시와 대동소이하여 Stepwise 2단계에서의 추정식이 현실성이 있는 것으로 판단되었다.

2. 토마토의 생체정보 분석

토마토의 생체정보 분석은 '95년 억제재배 작형에서 수집된 자료를 바탕으로 하였는데, 이때의 토마토 품종은 미니토마토인 빼빼(다끼이종묘)였다.

표 5-10은 토마토 생체정보 분석시 이용한 변수의 개요이며 표 5-11은 환경요인과 생체정보간의 상관을 나타낸 것이다. 표 6-11에서와 같이 엽온은 오이에서와 마찬가지로 기온과 가장 큰 정(+)의 상관을 보였는데 그 정도는 오이보다는 낮았으며, 습도와는 -0.536의 부(-)의 상관을 보여 오이와는 다른 경향이었다.

과경, 경경과 기타 요인들간의 관계는 오이와 동일하게 매우 낮거나 유의성 있는 관계를 보이지 않았는데 그것은 오이의 경우처럼 변화반응시간의 상호불일치 때문에 단순 비교자체가 무리한 때문이었다(그림 5-11, 5-12). 토마토 과경의 경우 오이보다는 과경증대속도가 느렸으며, 지속적인 증대경향을 보였으나 낮동안에는 정체되는 경향이었고 야간에 증대속도가 빨랐다(그림 5-11). 토마토 경경은 오이에서처럼 낮동안 증산류의 이동이 활발한 시기에는 감소하는 경향이었다(그림 5-12).

증산류의 경우 최고의 상관을 보인것은 습도로서 상관계수 -0.82의 부의 상관이었고, 다음으로는 일사량과 0.785의 정의 상관이었다.

오이에서와 같이 측정 생체정보 가운데 증산류는 환경변화에 매우 빠르게 반응하고, 또 작물생리 상태를 가장 잘 표현하여 토마토에서도 증산류를 지표로 삼아 온실환경제어를 할 수 있을 것으로 판단되었다.

표 5-10. 토마토 생체정보 분석시 이용한 변수 개요

변 수 명	최 소 치	최 대 치	측정기간
기 온 (℃)	11.0	30.2	'95.9.13 ~ '95.10.12
습 도 (%)	26	99	
엽 온 (℃)	12.1	32.2	
1회측정 과경 (mcm)	0	5,000	
1회측정 경경 (mcm)	0	2,000	
증 산 류	1.04	5.0	
일 사 량 (W/㎡)	0	1035	

표 5-11. 환경요소와 생체정보간의 상관

	습 도	엽 온	과 경	경 경	증산류	일사량
기 온	-0.844	0.745	0.173	0.074	0.709	0.77
습 도		-0.536	0.037	-0.071	-0.820	-0.823
엽 온			-0.081	0.234	0.495	0.519
과 경				-0.134	-0.107	-
경 경					0.080	0.076
증산류						0.785

토마토의 증산류 또한 오이와 같은 경향으로 온실내 포화수증기압차, 일사량과 밀접한 관계를 나타냈으며(그림 5-7), 아울러 (기온-엽온)과도 변화양상이 유사하였다(그림 5-9). 토마토의 증산류 또한 습도와는 반대의 경향이었다(그림 5-8). (기온-엽온)은 포화수증기압차와 유사한 변화양상을 보여 두 요인의 원인 결과관계를 입증할 수 있었는데, 이시기 토마토 엽온은 한낮에는 기온보다 낮았지만 야간에는 오히려 더 높은 경향이었다(그림 5-10). 이것은 이시기가 억제재배 작형으로 야간에 무가온이었기 때문으로, 앞서의 오이 반촉성재배의 야간가온시 항상 엽온이 기온보다 낮았던 것과는 다른 결과였다.

오이의 경우, 과경을 추정할 수 있는 회귀모델은 경경에 비해 그 적합성이 매우 낮았으나 토마토에서는 반대로 경경에 비해 과경의 회귀모델이 보다 적합성이 높은 것으로 나타났다. 그러나 근본적으로 과경변화의 반응시간과 기타 요인들과의 반응시간이 너무 상이하여 3요인 대입시 모델 결정계수는 0.32로 낮았다. 따라서 과경에 대한 회귀모델도 오이에서처럼 토마토에서도 착과기 이후에는 매우 중요한 생육지표 가운데 하나가 될 수 있으므로, 이를 위해서는 앞서의 언급처럼 요인들간의 변화 반응시간을 합리적으로 일치화시키거나, 감도높은 센서를 사용하여야 할 것으로 생각된다. 특히 토마토의 경우에는 과경증대속도가 오이 보다도 훨씬 늦기 때문에 이 부분의 보다 상세한 연구가 필요할 것으로 생각된다.

증산류를 추정할 수 있는 회귀모델은 습도를 단요인으로 하였을 때 결정계수 0.67이었으며, 일사량을 포함한 2요인 대입시는 0.71이었다. 이것은 오이의 경우보다 추정식의 설명력이 낮은 것으로 보다 상세한 연구가 필요하였다.

표 5-12. 과경추정 회귀모형

요인수	회 귀 식
1	과경 = 1408.3 − 17.684 × 기온
2	과경 = -961.446 + (72.527 × 기온) − (17.845 × 습도)
3	과경 = -1549.737 + (140.246 × 기온) − (23.757 × 습도) −(57.375 × 엽온)

표 5-13. Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력

Step	삽입변수	Partial R^2	모델 R^2	$C_{(P)}$	Prob > F
1	기 온	0.0298	0.0298	1708.9	0.0001
2	습 도	0.1160	0.1458	1063.3	0.0001
3	엽 온	0.1746	0.3204	90.4	0.0001

표 5-14. 증산류 추정 회귀모형

요인수	회 귀 식
1	증산류 = 6.123 − 0.042 × 습도
2	증산류 = 4.756 − (0.028 × 습도) + (0.00134 × 일사량)
3	증산류 = 4.982 + (0.028 × 습도) − (0.00014 × 과경) + (10.0 × 일사량)

표 5-15. Stepwise법에 의한 분석시 Step별 삽입 변수와 모델의 설명력

Step	삽입변수	Partial R^2	모델 R^2	$C_{(P)}$	Prob > F
1	습 도	0.6712	0.6712	548.4	0.0000
2	일 사 량	0.0367	0.7079	77.9	0.0001
3	과 경	0.0058	0.7137	5.3	0.0001

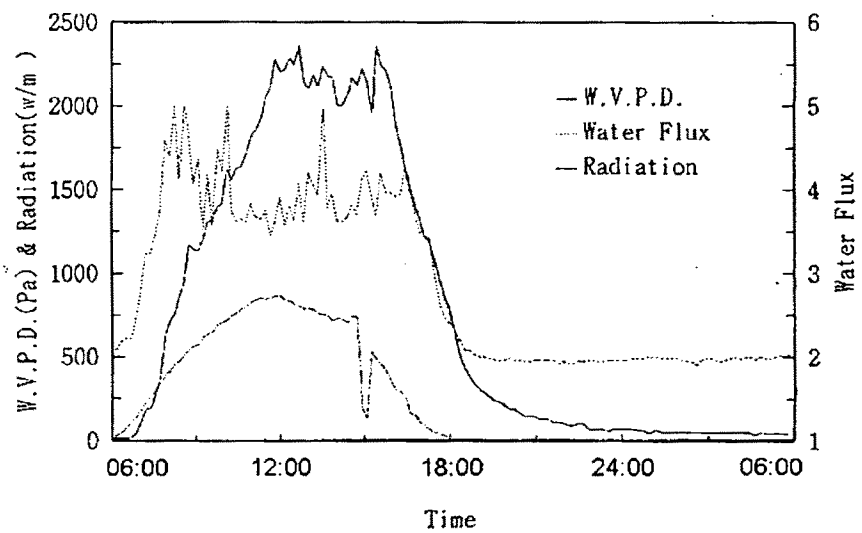


그림 5-7. 온실내 포화수증기압차 (Water Vapor Pressure Defficient), 토마토 증산류(water flux) 및 일사량의 일변화

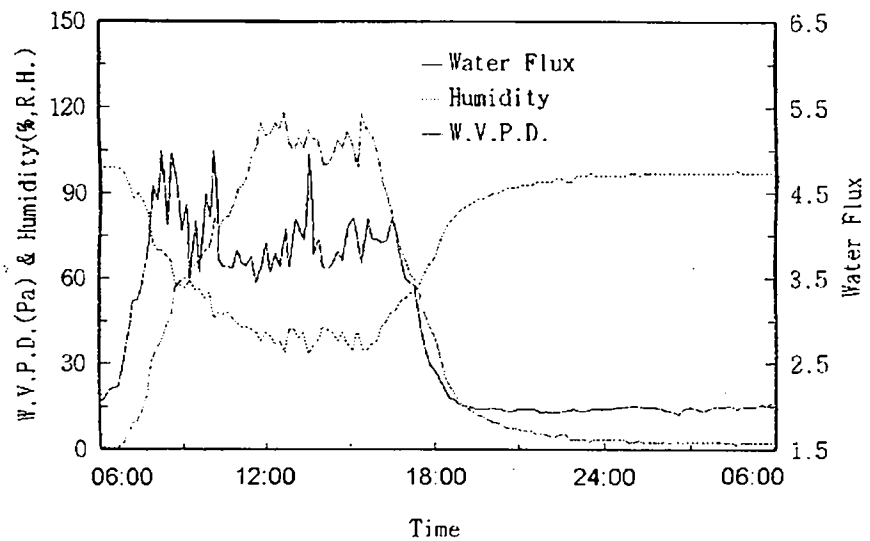


그림 5-8. 온실내 포화수증기압차(Water Vapor Pressure Defficient), 습도 및 토마토 증산류의 일변화

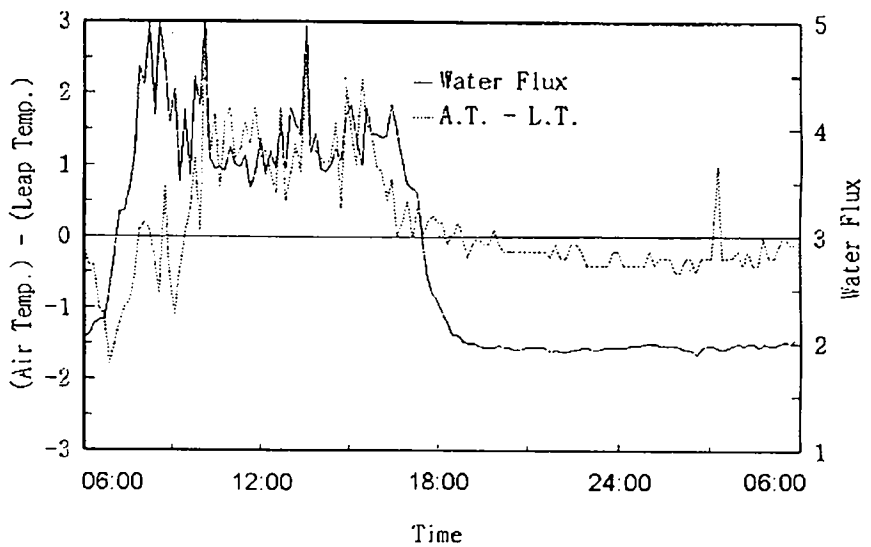


그림 5-9. 토마토의 증산류와 (기온-엽온)의 일변화

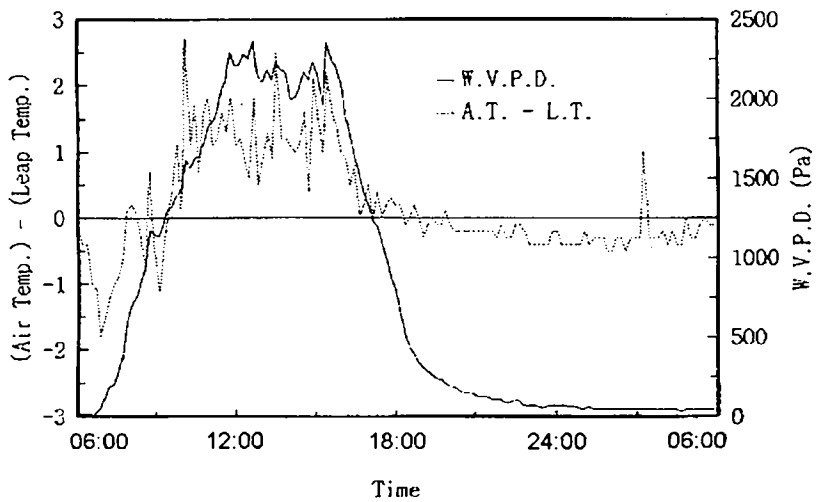


그림 5-10. 토마토재배시 포화수증기압차와 (기온-엽온)의 일변화

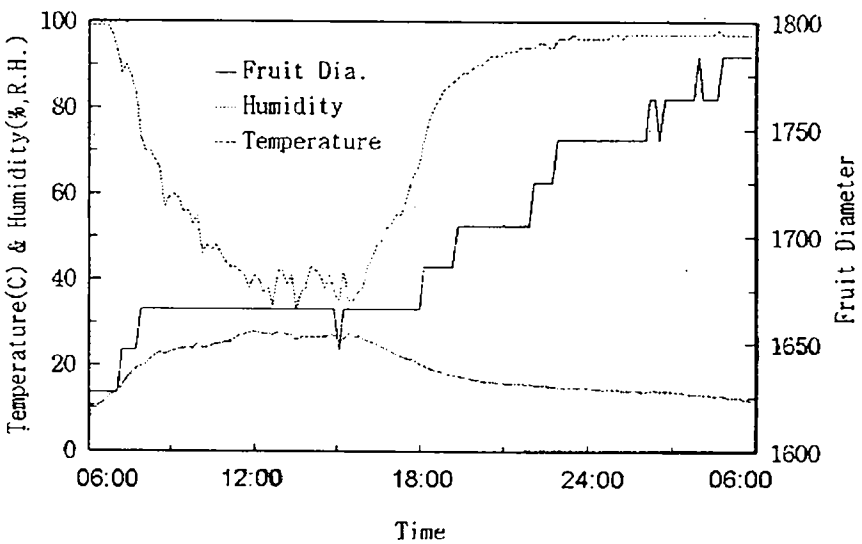


그림 5-11. 토마토의 과경, 습도 및 기온의 일변화

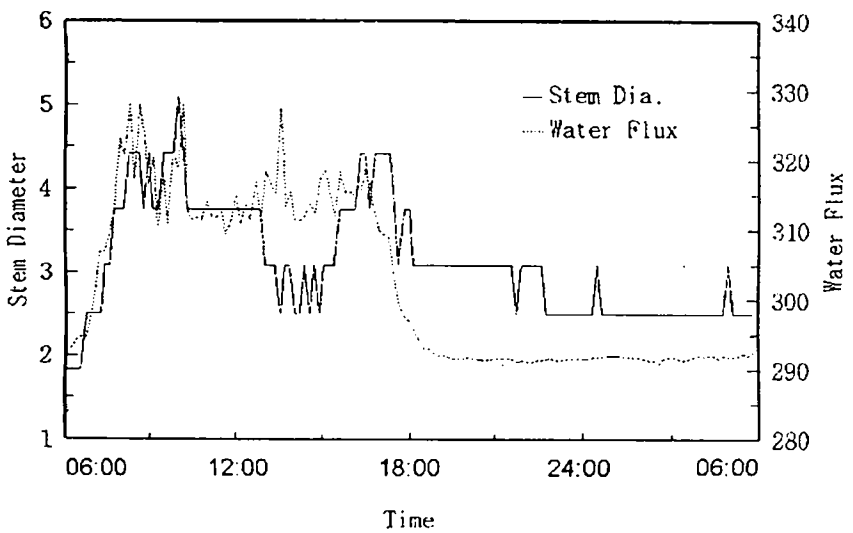


그림 5-12. 토마토의 경경과 증산류의 일변화

제 3 절 생체정보 분석시 문제점과 개선대책

본연구에서 생체정보를 이용한 온실환경관리 소프트웨어를 구축하기 위한 자료 분석결과 실제적용에 이르기에는 여러 문제점들이 도출되었다.

1. 생체정보 수집항목

본실험에서 이용한 생체정보는 엽온, 경경, 과경 그리고 증산류 등 4가지에 불과하여 전반적인 작물의 생체반응을 파악하기 위해서는 보다 많은 생체정보 항목이 필요하였다.

최적생육지표로서 주로 증산류를 분석하였는데 증산류의 경우 수분스트레스의 파악에는 매우 유효하지만 그것이 곧 작물의 최대생산성을 나타내주는 지표는 아니므로 앞으로는 여기에 특히 CO_2 동화량을 계측할 수 있는 장비가 추가될 필요가 있었다. CO_2 동화량과 증산류를 결합시키면 작물의 최적생육지표로서 매우 유용하게 사용될 수 있고 재배자가 원하는 방향으로의 작물생육유도에 적합할 것이다. 여기에 초장 또는 엽면적 항목이 가미되어 외향적인 생장량이 계측되어야 보다 완벽한 작물 생육상태를 파악할 수 있고 이에 따른 환경제어도 적합할 것으로 생각된다.

2. 센서의 문제

접촉식 생체정보 측정 방식으로는 군락을 대표하는 개체의 선정이 어려워 한 두 개의 측정치로는 만족할 만한 군락대표치를 얻기 어려우므로 보다 많은 측정치가 필요하게 된다. 이는 곧 비용의 증가와 관리운용상의 어려움을 수반하므로 비접촉식 생체정보 측정방식이 적극 도입될 필요가 있다.

증산류 센서의 경우 토마토, 오이 등 생장이 빠르고 왕성한 과채류에 적

합하도록 측정 Range를 증가시킬 필요가 있었다. 본 연구에 사용된 증산류측정센서는 크기가 작아 오이의 경우 상위엽의 엽병에, 토마토의 경우 2차엽병에 설치하여 측정하였는데도 잎의 최대증산시에는 센서가 더 이상 계측하지 못하는 오류가 있었다.

오이에서 과일직경센서의 경우, 센서의 최대 감지 범위가 0~5,000mcm인데 비하여 오이과일의 1일 생장량은 이것을 초과하여 매일 재조정을 하여도 최대생장량 감지가 어려웠다. 또한, 오이는 '96년 춘계재배시 19~21시간이면 5,000mcm(5mm)의 과경증대가 일어났다. 따라서 앞으로 센서 개발시 이들 센서들은 보다 감지범위를 증대시킬 필요가 있었다.

3. 생체정보 분석시 문제점

생체정보를 이용한 온실 환경관리는 기본적으로 생체정보에 의해서 온실환경을 관리하는 것이므로, 측정된 생체정보가 온실 환경제어에 이용될 수 있도록 처리되어야 한다. 이를 위해서는 환경과 생체정보간의 관계를 수학적 모델로서 정립하는 것이 매우 효과적인데 정밀 환경조절이 어려운 자연조건 온실에서의 실험으로는 만족할만한 결과를 얻기 어려웠다. 따라서 적합한 모델을 구축하기 위해서는 정밀 환경조절이 가능한 생육상에서의 연구가 필요하였다. 이때의 생장상은 일사량은 자연조건으로 두고 나머지 환경요인들만 제어되도록 하는 것이 도출 모델의 실제이용시 유리할 것으로 생각된다.

앞서 언급한 바와 같이 과경과 경경을 종속변수로 한 회귀모델 구축에는 근본적으로 문제가 있었다. 따라서 이들과 여타의 다른 항목들과의 변화 흐름을 어떻게 접근시켜 상호관계를 구명할 것인가를 보다 심도 있게 연구할 필요가 있었다. 과채류의 경우 재배자가 의도하는 최적의 생육은 수확량 즉, 과일수량을 최대로 하는데 있으므로 과일착과기 이후에는 과경

이 중요한 생육지표가 될 수 있다. 과경 즉 과일의 생장은 광합성과 동화산물의 과일로의 전류에 큰 영향을 받는데 광합성은 또 온도, 일사량, CO_2 농도 그리고 양·수분 조건에 큰 영향을 받으므로 과경의 증대정도에는 많은 요인들이 복합적으로 작용하고, 그것들의 효과 또한 누적적이고 지연적으로 나타난다고 볼 수 있다.

따라서 여러 생리작용들의 결과물인 과경변화를 수학적으로 모델화하기 위해서는 정밀 환경조절이 가능한 생육상에서 보다 심도있는 연구가 필요할 것으로 여겨진다. Jones등(1991)은 일사량을 제외한 나머지 환경요인들을 조절할 수 있는 생육상에서의 실험을 통해 토마토 생장 및 수량 모델을 작성하였는데 이것은 온실환경제어에 유용하게 이용할 수 있었다고 하였다.

생체정보 분석에는 실측자료 뿐만 아니라 실측자료를 가공한 자료들의 분석도 필요하다. 작물 수분소비모델의 경우 증산류 단독으로 뿐만 아니라 앞서 그림에 제시 되었던 (기온 - 엽온), 포화수증기압차 등 계측자료 처리에의한 2차적 data들을 활용하므로써 보다 설명력 높은 모델구축이 가능할 것으로 판단된다.

여 백

제 6 장
복합환경 제어시스템의 설계 및 제작

제 1 절 서 언

일반적으로 온실의 환경제어는 환경요인만을 제어하는 것을 의미하나, 본 연구에서는 작물의 생육 상황을 분석하여 환경을 제어하므로써 진정한 의미의 복합환경 제어를 구현하고자 수행하였다. 최근까지 국내에서는 이러한 연구가 시도된 적이 없으며, 네델란드 등과 같은 원예농업의 선진국에서 이러한 연구가 일부 시도되어 상품화된 예가 있으나, 현재로서는 활발한 연구가 이루어지지 않고 있다.

온실의 환경제어를 위한 하드웨어의 개발과정에서 보면, 지금까지 연구, 개발되는 하드웨어들은 데이터의 입력과 출력과정에서 각 센서들의 단자들과 각 Relay 단자들이 하드웨어의 일정한 장소에 고정적으로 장착되어 사용되므로써, 각각의 센서나 Realy단자들의 고장시에 교환하거나 수리하는 과정에서 적지 않은 어려움이 있다. 또한 같은 시스템을 가지고 환경측정요인, 작동기기의 종류, 동작상태가 다른 온실이나 또는 그로스챔버(growth chamber)등의 생육시스템에 적용시킬 경우 각각의 센서의 단자들과 Relay 단자들의 용도변경이 필요하다. 따라서 기존의 시스템들을 가지고 이를 수행할 경우에는 시스템의 상당부분을 변경시켜주어야 하는 어려움을 가지고 있다.

소프트웨어의 개발과정도 마찬가지로 기존의 시스템에서 조그마한 변화

나 용도변경이 필요할 경우에는 입출력 및 디스플레이의 각 부분에 걸친 많은 부분의 수정 및 추가, 삭제가 필요하다. 특히 일반 사용자들이 소프트웨어를 구동시킬 때, 소프트웨어상의 변경이 필요하게 되면 기존 시스템의 경우에는 사용자들이 직접 수정을 한다는 것은 거의 불가능하다.

이러한 단점들을 보완하기 위해서 각 하드웨어의 입출력부분을 탈착과 부착이 용이하고 교체 및 수리가 용이하도록 모듈화시키고, 각각의 모듈에 독립적인 CPU를 부착시켜 다른 부분들의 용도변경 및 수리시에 영향을 받지 않도록 설계하는 것이 필요하다. 소프트웨어도 마찬가지로, 기존의 방식으로 하드웨어의 고정적인 입출력단자에 연결시키는 방법을 탈피하여 독립적인 하드웨어에 맞추어서 프로그램되어야 한다. 즉, 소프트웨어를 하드웨어의 각 모듈에 연결시켜서, 하드웨어의 각 모듈이 용도와 사용범위가 변할 때 그 변화에 맞추어서 각 모듈의 용도와 범위, 개수, 사용방법 등을 유동적으로 지정해 줌으로써 쉽게 응용할 수 있게 하여야 한다.

본 연구에서 개발된 각 하드웨어는 각각의 단자들이 고유한 목적을 가지고 작동하는 것이 아니고, 각 모듈자체의 고유주소만을 가지고 있고, 소프트웨어상에서 각 고유주소에 해당하는 모듈의 사용방법 및 용도, 범위 등을 지정하여 사용함으로써 온실의 작동환경이 변화하거나 또는 조건이 다른 온실에 응용할 수 있도록 하였다. 또한 고장 및 용도변경의 요구가 있을 시에 시스템 상에서 요구되는 부분의 교체, 수리를 쉽게 하기 위하여 하드웨어상의 각 모듈을 슬롯(slot)형태로 제작하여 탈부착이 용이하도록 설계, 제작 하였다.

또한, 각각의 시스템이 모듈화되어 있고 각각의 모듈은 자신의 고유 CPU를 가지고 있어 용도변경뿐 아니라, 확장에도 용이하도록 설계를 하였다. 각 Controller는 여러 개의 연결포트를 가지고 있어서 PC와 연결할 수 있는 포트와 각 입출력 블록의 연결되는 포트, 그리고 확장이 필요할 때

다른 Controller와 연결될 수 있는 포트를 가지고 있다. 특히 다동온실을 제어할 때에는 다른 Controller와 연결 할 수 있는 포트들을 연결시켜서 각 Controller의 고유번호만 지정을 해 준다면, 그중 하나가, Main Controller 가 되고 나머지 Controller들은 자신들의 고유번호를 통해서 PC 또는 다른 Controller들과 통신을 할 수 있게 하였다(그림 6-1).

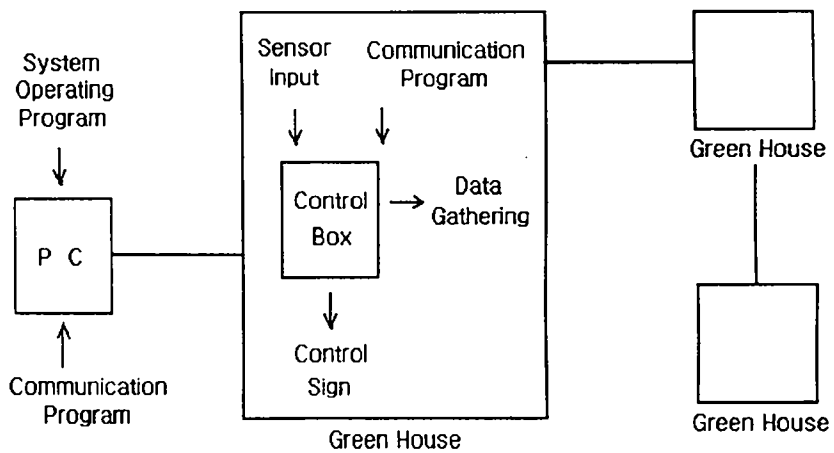


그림 6-1. 시스템의 개략도

제 2 절 복합환경 제어시스템의 구성

본 연구에서는 데이터의 입출력과 작동기기들을 작동시키는 Controller가 부착된 하드웨어와 사용자가 PC를 이용해서 각 하드웨어의 설정치를 지정 및 변경할 수 있는 제어용 소프트웨어로 구성하였다. 또한, 제어용 소프트웨어는 수집된 데이터를 받아보고 저장할 수 있으며, 직접 수동으로써 작동기기들을 제어해 줄수 있는 시스템 구동 소프트웨어, 그리고 PC와 모듈, 모듈과 모듈사이의 통신을 하여 데이터를 주고 받을 수 있는 통신프로그램으로 구성하였다.

본 시스템의 물리적인 구조는 그림 6-2와 같으며, 그림 6-2에서와 같이 각각의 독립적인 모듈구조로 되어있다. 하나의 Main 모듈에 여러개의 Sub 모듈을 연결할 수 있도록 구성하였다. 기본적으로 하나의 Main 모듈에 각 8개의 Sub 모듈을 연결시킬 수가 있으며, 그 Sub 모듈이 어떠한 모듈인가는 상관없이 각각의 모듈들은 Slot 형태로 제작하여 확장 및 탈부착이 용이하도록 하였다.

Main 모듈의 CPU는 80c196를 사용하였으며, Booting시에 필요한 Booting ROM과 수집된 자료를 일시 저장하는 RAM이 있다. 자료 저장용RAM은 환경설정치(Environmental Configuration)와 각 모듈들의 특성을 나타내는 각 입출력 단자들의 모듈내 번지(Address)가 입력되어있는 RAM과 각종 데이터를 수집해주는 Data Gathering RAM들로 구성하였다. 본 시스템에는 통신을 주관하는 SIO 통신칩을 부착시켜서 PC 또는 각 모듈과의 통신을 원활하도록 하였다. Main 모듈과 각 Sub 모듈의 종류에 따른 용도 및 사용된 프로그램언어, CPU의 종류들은 표 6-1에 나타내었다.

각 Main 모듈과 그에 연결되어 있는 각종 Sub 모듈은 하나의 Control 블록으로 분리시킬 수 있도록 하였다.

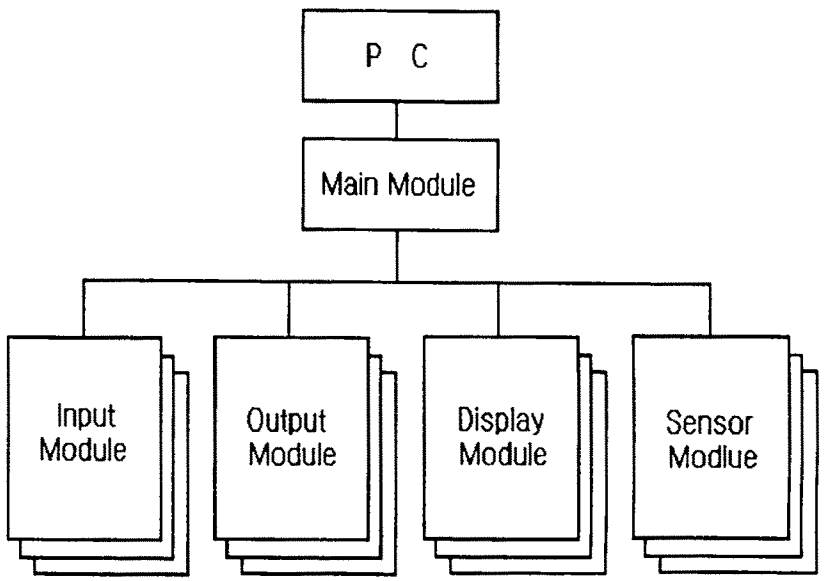


그림 6-2 시스템의 물리적인 기본구조

표 6-1. 모듈의 종류

Module의 종류	CPU	Language	비 고
Main module	80c196	C, Assembler	PC와의 연결 및 다른 모듈과의 통신등 기타 모든 상황을 주관한다.
Display module	87c51	C	환경설정 및 기기동작상황을 볼 수 있다.
Sensor module	87c51	C	각종 센서들의 입력을 담당한다.
Input module	87c51	C	각종 동작기기들의 입력을 담당한다.
Output module	87c51	C	각종 기기들의 전원공급 및 가동을 담당한다.

특히 본 연구에서 개발된 시스템은 다동온실의 제어 또는 다량의 센서입력과 작동기기들의 출력이 필요로 할 때는 그림 6-3과 같이 개수에 큰 제한을 받지 않고 쉽게 확장시킬 수 있도록 하였다.

소프트웨어는 사용자가 보기 좋고 사용의 편리성을 위하여 Windows용 프로그램을 사용하였다. 프로그램구성은 현재의 온실상황을 한 화면에 모두 보여줄 수 있는 실시간 환경상황 모드와 수집된 자료를 시간과 종류별로 열람 및 저장할 수 있는 자료수집모드, 그리고 환경설정모드, 하드웨어 설정모드, 수동모드로 구성 하였다.

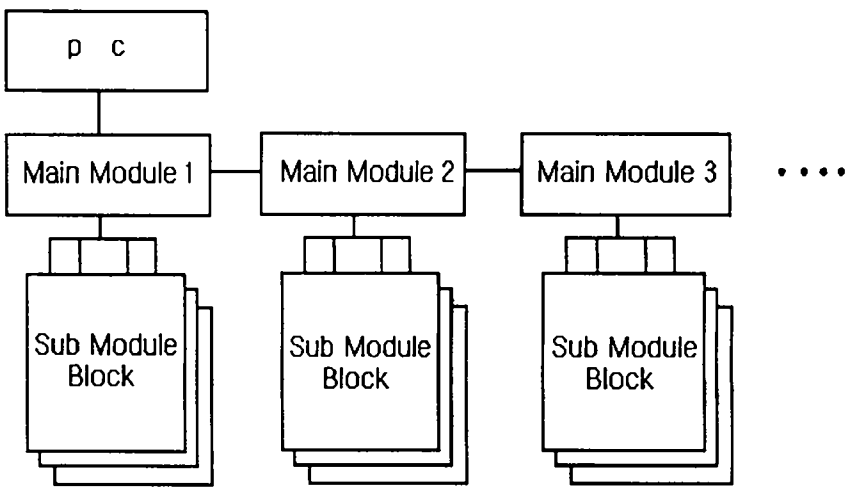


그림 6-3. 모듈의 연결과 확장

하드웨어 설정모드에서는 각각 고유의 번지만을 가지고 있는 각종 모듈들의 종류와 사용범위, 사용방법 등을 설정해 줄 수 있으며, 하나의 모듈만을 확장, 교환 또는 용도변경시에는 이 하드웨어 설정모드에서 변경이 요구되는 모듈의 번지만을 지정하여 변경을 해주면 되도록 설계하였다.

표 6-2는 소프트웨어상의 각각의 모드의 종류와 그 모드에서 사용할 수 있는 항목들을 간단히 나타내었다.

표 6-2. 소프트웨어상의 각 모드의 종류와 기능

선택 모드의 종류	기 능
실시간 환경 상황	현재의 온실 상황을 3초 내지 5초정도의 간격으로 각종 센서값 및 작동기기의 동작상황 등을 표시
자료수집	현재까지 일정기간동안 모아두었던 각종 센서값들과 작동기기의 동작과정을 표시해주고 저장 및 저장된 자료의 열람
환경설정	온실의 적정 온습도 등의 환경을 사용자가 직접 설정
하드웨어 설정	자료수집 시간간격, 동작기기의 1회 작동시간, 비상시 설정온도 등의 설정
입출력 단자 설정	하드웨어상의 고유번지(address)를 가지고 있는 각 입출력단자들의 용도 및 사용범위를 설정
수동모드	사용자가 PC에서 직접 작동기기 제어
도움말	프로그램 전반적인 설명

본 시스템에서는 제어 Logic을 RAM에 입력시켜 환경설정치에 따라서 각종 작동기기들을 자동으로 동작시키도록 하였다. 기존에는 각종 작동기기별로 환경설정치들을 부여하여 각 작동기기들이 자신들만의 고정적인 환경값들을 갖고 제어가 되는 방식이었다. 그러나 본 연구에서 설계한 제어 Logic은 작동기기별로의 환경설정이 아닌 온실 전체의 온도와 습도 및 지중온도 등의 환경 예약값들을 설정하여 온실내의 온도조정시에 시퀀스 제어를 하도록 구성하였다.

프로그램된 제어 Logic들은 각각의 독립된 Main 모듈에 저장되어 각 온실마다 다른 환경 설정값들과 다른 입출력 인자들에 구애를 받지 않고, 독립적인 환경제어를 할 수 있도록 하였다. 그림 6-4는 본 연구에서 개발한 제어 Logic의 기본 구성을 나타낸 것이다.

PC와 Main 모듈간의 통신은 통신프로그램의 작성이 쉬운 RS232 통신을 사용하였고, 모듈과 모듈사이의 통신은 통신거리에 거의 영향을 받지않는 485 통신을 사용하였다.

각각의 Main 모듈에는 SIO통신칩이 장착되어 있으며, 다동온실을 제어하는 경우에 어느 곳의 모듈이라도 PC와 연결되는 모듈이 Main Control 모듈로 사용되어질 수 있도록 하였다. Main 모듈과 Main 모듈과의 연결순서는 상관이 없으며, 각각을 직렬로 연결시키고 각각의 모듈의 고유 번호만 지정해 주면 PC와 연결되어 있는 Main 모듈을 통해서 각 모듈의 자료들을 받아 볼 수 있도록 하였다. Main Control 모듈은 연결선의 절약과 통신상의 노이즈(noise)를 줄이기 위하여 PC와 가장 가까운 곳에 위치한 Main 모듈을 Main Control 모듈로 사용하였다.

그림 6-5는 PC와 Main Control 모듈, 각각의 Main 모듈과 Main 모듈들간의 통신관계를 블록형태로 표시해 주고 있다.

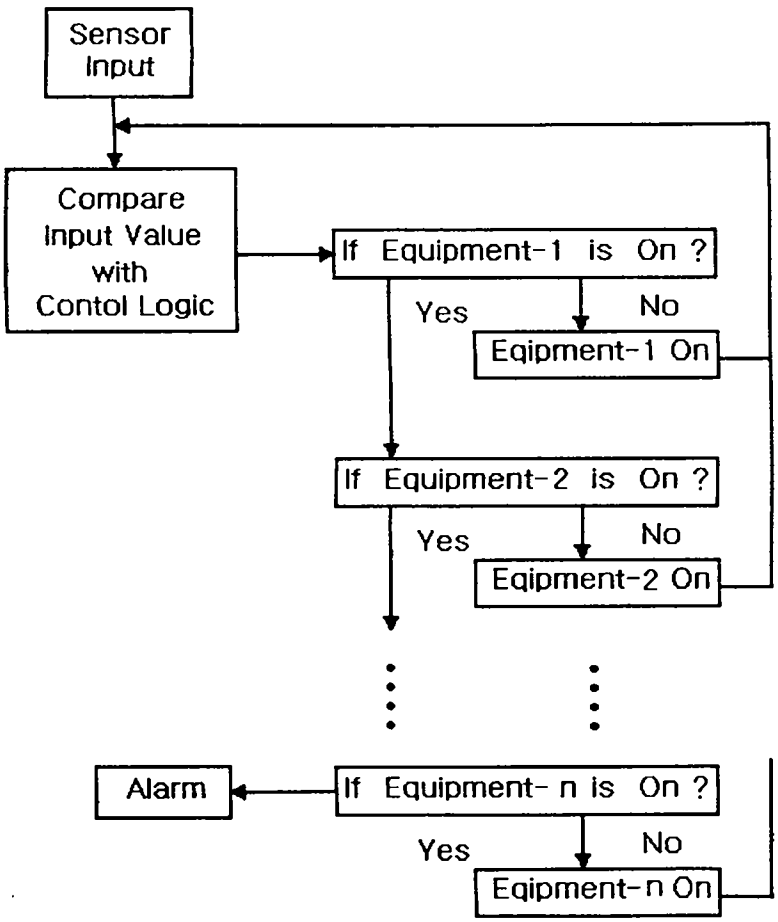


그림 6-4. 제어 logic의 기본 구성도

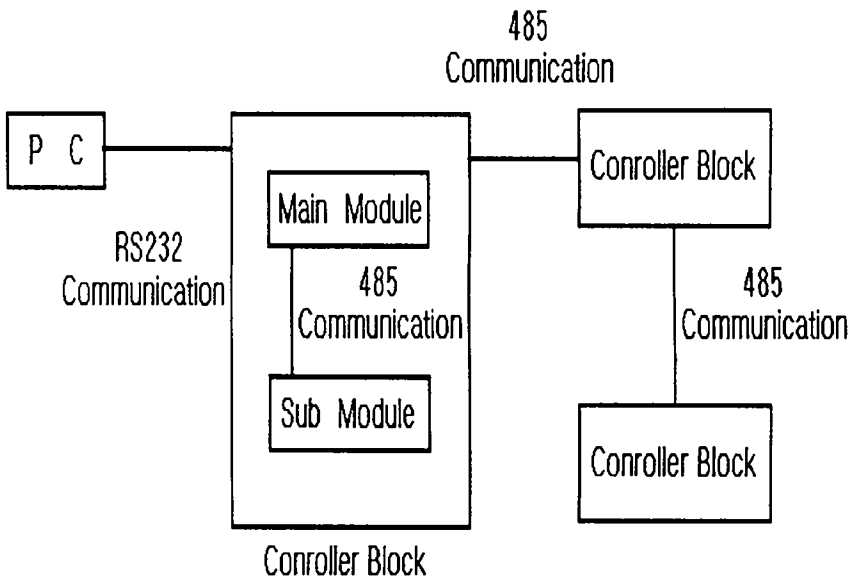


그림 6-5. 통신프로그램의 종류와 연결

제 3 절 복합환경 제어시스템의 소프트웨어

복합환경 제어시스템의 소프트웨어는 넓은 의미에서 PC를 통한 제어시스템 구동에 필요한 PC용 소프트웨어와, 하드웨어의 RAM에 writing시켜 놓는 하드웨어 구동 및 하드웨어와 하드웨어 사이의 통신(모듈과 모듈사이의 통신)을 할 수 있는 하드웨어용 소프트웨어, 그리고 PC와 Main Control 모듈사이, 모듈과 모듈사이의 통신을 할 수 있는 통신 프로그램으로 나눌 수 있다. 하드웨어용 소프트웨어는 제 4절 복합환경 제어시스템의 하드웨어에서, 통신 프로그램은 제 5절 복합환경 제어시스템의 통신 프로그램에 나타내었다. 여기에서는 PC를 통한 시스템의 구동에 필요한 전반적인 PC용 소프트웨어의 구성과 환경제어를 위한 제어 Logic의 구성 및 순서도에 대하여 나타내었다.

1. PC용 소프트웨어의 기본 구성

PC용 소프트웨어는 보기에 좋고 사용자의 편리성을 위하여 Windows용 프로그램을 사용하였다. 본 연구에 사용된 PC용 소프트웨어는 Visual Basic Language를 사용하였다. Windows용 프로그램은 보기좋은 디자인이 보다 쉽게 설계 및 작성, 수정이 용이하며 사용자가 마우스 하나로 거의 모든 동작을 행할 수가 있기 때문에 시각적 효과가 뛰어나며 프로그램 작성상에 있어서 시간을 절약할 수 있는 잇점이 있다. 온실의 환경제어용 프로그램을 작성할 때에는 PC의 처리속도에는 많은 영향을 받지 않기 때문에 직접 PC를 제어해 주는 기존의 Language를 사용하지 않고 Windows용 Language를 충분히 사용할 수가 있다. 또한 일반 사용자들이 현재 PC를 사용함에 있어서 Windows의 보급율이 점점 높아지고 있고, 그에 따른

소프트웨어의 보급도 늘어감에 따라 이러한 Windows용 소프트웨어들이 사용자들의 눈에 좀더 많이 적응되어가고 있는 실정이다.

이러한 여러 가지 이점들 때문에 앞으로 연관분야에서의 소프트웨어의 제작시에 Windows용 소프트웨어의 사용이 증가할 것으로 판단된다.

본 연구에서는 시스템의 제작시 소프트웨어 상으로는 총 4개까지의 온실을 동시에 제어할 수 있도록 구성하였으며, 필요에 따라서는 소프트웨어의 간단한 변형과 하드웨어를 직렬 연결함으로써 4개 이상의 더 많은 온실을 동시에 제어할 수 있도록 하였다. 만약 4개 이하의 온실제어를 할 경우에는 본 소프트웨어로써 충분히 사용되어질 수 있다. 본 연구에서 제작한 PC용 소프트웨어의 기본 구성은 그림 6-6과 같다.

가. 초기화면

소프트웨어를 구동시키면, 시리얼 포트를 선택하는 옵션이 나온다. 사용자가 현재 사용하고 있는 PC의 사용가능한 시리얼 포트를 선택하면 되며, 마우스나 기타 다른 장치로 사용되고 있지 않은 시리얼 포트로 시스템의 하드웨어와 연결시킨후 그 포트를 선택하면 된다.

시리얼 통신포트를 선택하면 Main화면이 나오며, 여기에 현재상황, 환경설정, 하드웨어설정, 자료수집 등의 명령선택 버튼이 나온다. 이들 버튼 중 하나를 선택하여 마우스로 클릭해주면 사용자가 원하는 옵션으로 들어가 여러 가지 환경설정, 현재의 온실상황 및 수집되어지고 있는 자료들을 열람할 수 있다. Main 화면은 그림 6-7에 나타내었다.

나. 현재상황 모드

현재상황 모드는 현재 온실의 환경 및 생체정보자료 그리고 작동기기들의 동작상태들을 한 화면에서 모두 볼 수 있다. 그림 6-8은 현재의 온실상황을 실시간으로 보여주는 실시간 환경상황 화면을 나타낸다.

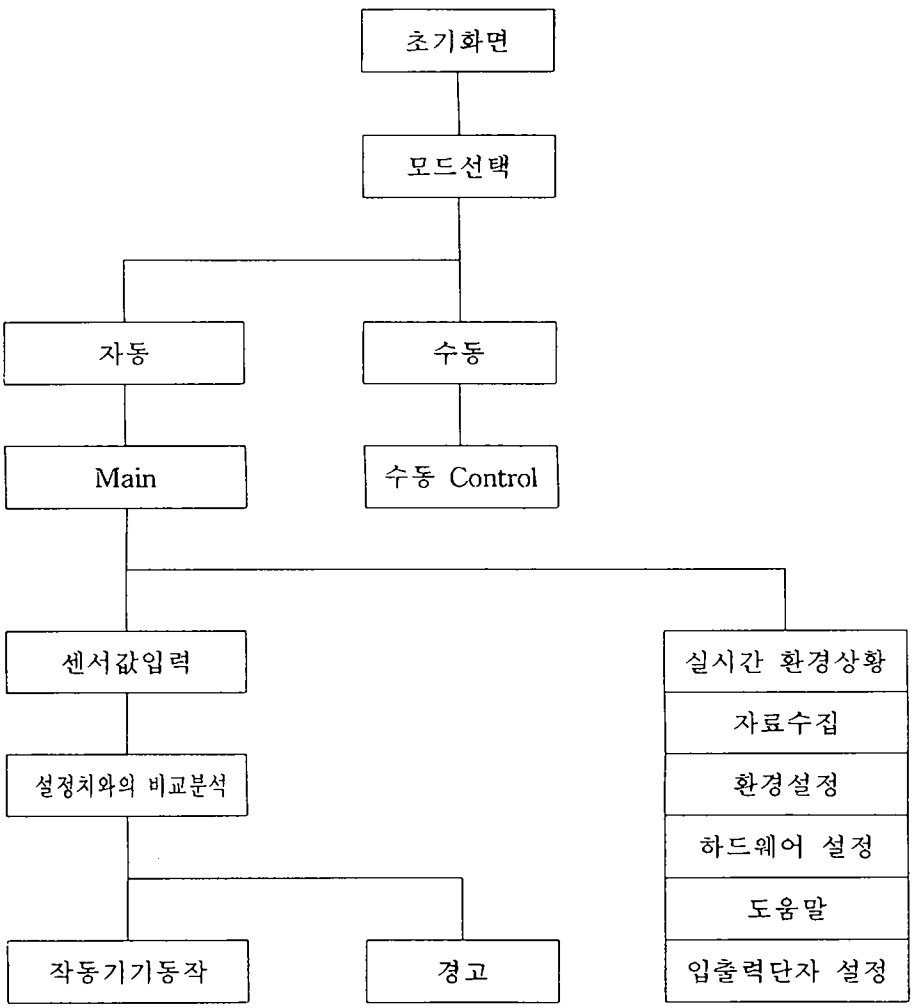


그림 6-6. 소프트웨어의 기본구조

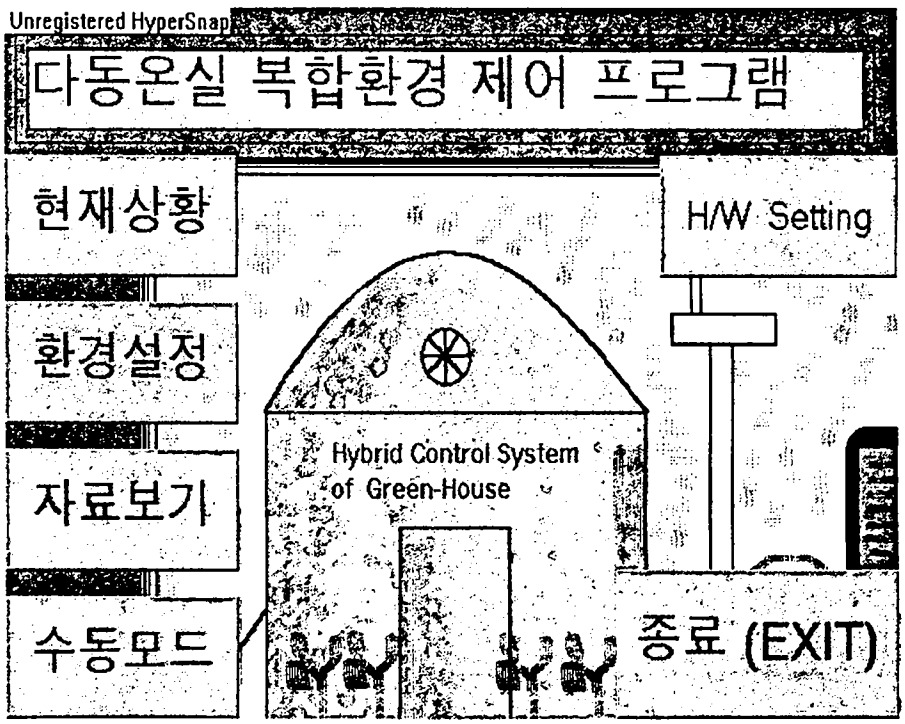

Unregistered HyperSnap
다동온실 복합환경 제어 프로그램
현재상황
환경설정
자료보기
수동모드
H/W Setting
Hybrid Control System
of Green-House
종료 (EXIT)

그림 6-7. 초기화면

그림 6-8과 같이 온실내부에는 온실내의 각종 센서 입력값들이 표시되어 있으며, 온실 외부에서는 외부의 환경 입력값들이 표시되어 있다. 센서값은 온실의 환경제어에 사용되는 모든 종류의 센서의 값들이 한 화면에 나타나도록 구성하였다. 현재상황 모드에서는 내부 환경인자로서 실내온도, 실내습도, 일사량, 지중온도, CO_2농도를 표시해주고 있고, 외부 환경인자로는 외부온도, 외부습도, 일사량, 풍향, 풍속, 강우량 등을 표시해 준다. 또한 생체정보 자료로서 도관류, 엽온, 과실직경, 줄기직경을 표시해 준다. 내부 일사량은 외부일사량을 환산하여 표시해 주며, 외부의 강우량은 현재 비가 오는지 않오는지의 의미 이외의 정확한 강우량은 큰 의미를 두고 있지 않다.

온실상태 상자에서는 현재의 온실상태를 표시해주며, 정상적인 상태에서는 "정상"이라는 표시가 나타나며, 온실작동기기의 이상이나, 제어 Logic의 범위를 벗어나서 가지고 있는 Logic에 의한 제어가 불가능할 때 어느부분이 이상이 있는지를 표시해 주게된다.

작동기기 동작상태는 창 종류와 커튼등 개폐의 동작을 하는 것들은 "열림" 또는 "멈춤" 으로 표시를 해 주고 나머지의 작동기기들은 "동작" 또는 "멈춤"으로 동작상태를 설명해 주도록 하였다. 동작기기의 동작상태를 나타내주는 message box의 색은 두가지로 표시하였으며, 각각의 동작기기가 자동모드 상태이면 빨간색으로, 수동모드의 상태이면 파란색으로 표시를 하여 각각의 동작기기가 제어 Logic에 따라서 움직이는 자동모드상태인지 아니면, 사용자가 임의로 동작시키고 있는 수동모드인지를 나타내도록 하였다.

또한 현재상황 모드에서는 작동기기들의 동작상황을 Text로만 나타내주는 것이 아닌 그림으로서 단순화하여 표시해 줌으로써 시각적인 효과와 더불어 한눈에 현재 상황을 쉽게 알아 볼 수 있도록 하였다.

화면 우측 하단의 온실번호 선택버튼을 이용하여 총 4개까지의 서로 다른 온실의 상황을 사용자가 원하는 대로 번갈아 가면서 볼 수가 있으며, Windows의 크기를 조절하여 동시에 여러 온실의 상황을 한 번에 볼 수도 있다.

다. 환경예약 모드

기존에 있는 온실환경 제어시스템의 환경설정은 각각의 작동기기별로 온도와 시간을 설정해 주는 방식이었다. 그러나 본 연구에서는 작동기기별로의 환경예약이 아닌 순수 온실내 환경을 기준으로 환경설정을 하도록 구성하였다. 온실내의 환경은 온실내의 온도와 습도를 기준으로 제어하도록 구성하였으며, 제어인자로서는 온실내 최고온도, 최저온도, 최고습도, 최저습도가 있다. 환경설정모드에서 설정이 되는 이러한 예약값들은 환경 제어 Logic에 의하여 온실내 온도는 최고온도와 최저온도사이, 습도는 최고습도와 최저습도 사이에서 유지될 수 있도록 하였다.

최대허용풍속은 천창을 포함하는 모든 창종류의 개폐가능 여부를 판단해 주는 인자로서 온실외부 풍속이 최대허용풍속의 설정치보다 세게 불면, 모든 창문을 닫도록 하였다. 만약 제어 Logic에 의하여 창문을 열 필요가 있을때에는 다른 방법을 찾게 되며, 그래도 제어가 불가능하게 되면, 경보를 내도록 구성하였다. 측창개폐허용풍속은 측창의 개폐가능성을 판단하는 기준이며, 예약값보다 바람이 세게 불면 양쪽 측창을 닫도록 하였다. 그러나 측창 이외의 창들은 제어 Logic에 따라서 개폐시킬 수 있다.

강우시 허용풍속은 풍속이 측창개폐허용풍속보다 약하게 불 때라도 비가 오는 경우에는 비가 온실내부로 들어올 염려가 있기 때문에 이를 방지해 주기 위하여 비가 올때의 창문개폐허용풍속을 별도로 지정해 주도록 하였다.

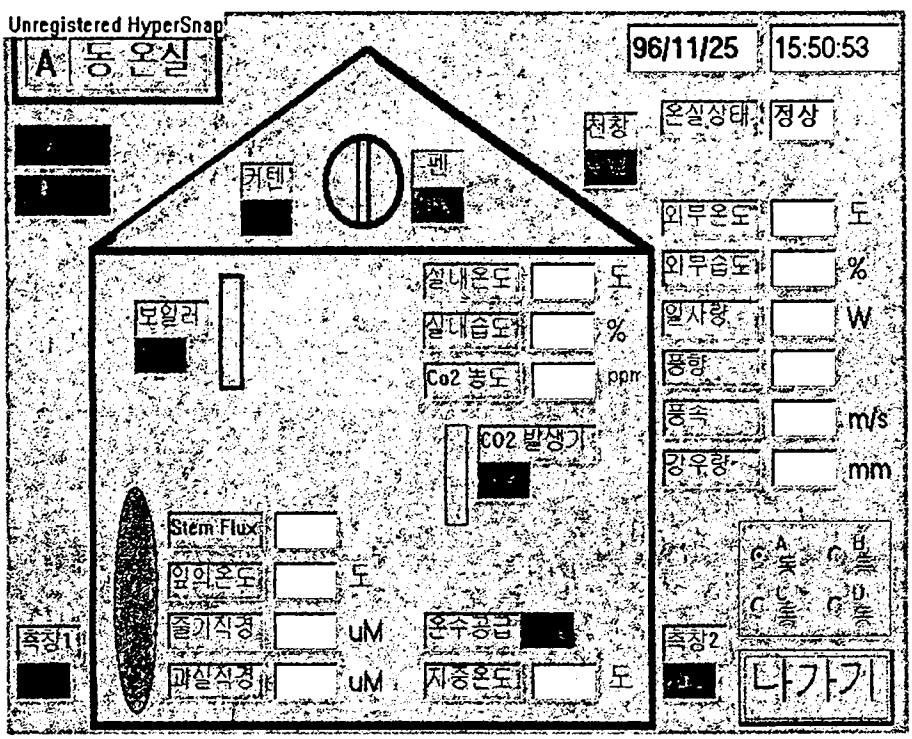


그림 6-8. 실시간 환경상황 화면

강우시 허용풍속도 다른 창문개폐 가능 풍속과 마찬가지로 비가 올 때, 풍속이 그의 허용풍속을 초과할 때에는 다른 제어 Logic을 수행하게 되며, 제어가 불가능할 때에는 경보를 표시해 줌으로서 사용자가 알 수 있게 하였다.

지중온도 설정은 온실내 작물뿌리가 있는 부분에 지중온도 센서를 부착시켜서 그 온도를 측정하게 하였으며, 설정치 보다 지중온도가 낮을 경우에는 지중온수 보일러를 가동시켜서 근권부온도를 올려주도록 하였다.

생체정보 센서들의 설정치들은 본 연구에서는 사용되지 않았으며, 앞으로 사용중에 생체정보의 값에 따라서 온실의 환경을 제어해 줄 필요성이 있거나 아니면 현재 가지고 있는 환경인자외의 다른 환경인자를 추가시켜서 환경을 제어해 줄 필요가 있을때에 추가적으로 사용할 수 있는 일종의 보조 message box로 이용하였다.

환경설정 모드에서 예약값들을 입력시킨 후에 설정 버튼을 누르면, 해당하는 온실에 있는 controller의 RAM에 저장이 되어 제어 Logic에 의하여 온실의 환경이 제어된다. 그림 6-9는 환경설정 모드를 나타낸다.

라. 자료보기 모드

자료보기 모드에서는 온실에 사용되고 있는 내부환경 측정센서, 외부환경 측정센서, 생체정보 측정센서등 온실의 환경제어에 사용되고 있는 모든 센서의 값들을 하드웨어 설정에서 설정해 주는 시간 간격으로 표시하도록 하였다.

한 화면에 표시되지 않은 부분들은 수직 스크롤바와 수평 스크롤바를 이용하여 원하는 위치에 있는 자료를 빠르게 탐색하여 열람할 수가 있다.

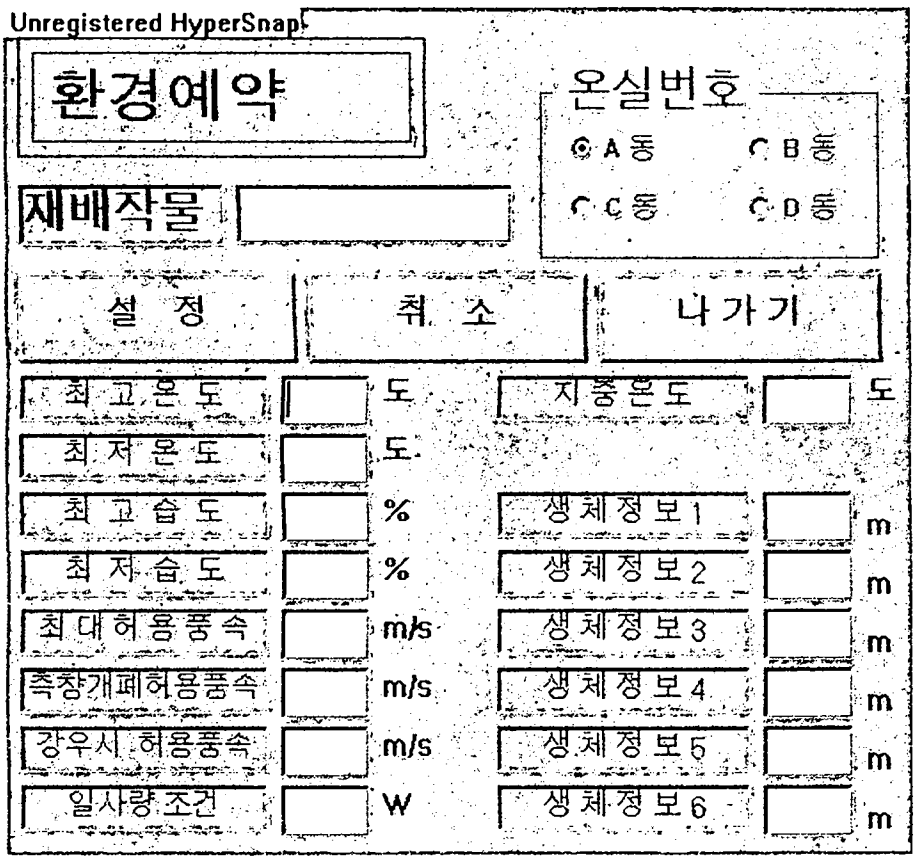


그림 6-9. 환경예약 모드 화면

온실의 상황이 제어 Logic의 범위를 벗어나서 제어 Logic만으로는 제어가 불가능한 상황이거나 그 이외의 정전 등의 특정 상황에 의하여 온실내의 경보가 발생하였을 때 그 경보의 유무를 저장하였다가 자료보기 모드의 최 좌측 message box에 표시를 하여 각각의 시간에서의 온실의 정상 또는 비정상(경보)상태를 알 수 있게 하였다.

자료보기 모드에는 하부모드가 있으며, 하부모드에는 구간보기 모드와 동작상황 모드, 자료저장 모드, 파일보기 모드등 하부 모드가 있다.

(1) 구간보기 모드

처음에 자료보기 모드를 들어가면 지정된 시간간격으로 일정한 간격 전부터(특별한 옵션이 없을 경우에는 24시간 전으로 함) 현재 시각까지의 자료들이 자동으로 로드(load)된다.

현재까지의 하루동안의 자료이외에 그 전의 자료 또는 하루중에서 일부의 자료만을 보고 싶을 때 구간보기 모드를 이용하며 원하는 구간을 시작되는 때의 년, 월, 일, 시, 분과 원하는 구간의 끝날 때의 자료를 입력시키면, 원하는 부분의 자료들을 열람할 수가 있다. 분까지 입력을 하지않고 시까지만 입력이 된다면, 그 시의 마지막 측정된 자료까지 읽어올 수가 있다.

한 번에 6만 라인의 자료를 읽어올 수가 있기 때문에 자료수집 간격을 10분으로 하였을 때, 약 8개월간의 자료를 한 번에 열람할 수가 있다.

그림 6-10은 구간보기 모드를 나타낸다.

(2) 동작상황 모드

동작상황 모드에서는 지정해준 기간동안(구간보기 모드를 들어가기 전에는 현재이전의 24시간 전부터의 구간)의 센서값 이외의 작동기기의 동작상황을 나타내준다. 각 측정시간으로부터 측정되는 간격별로 작동된 작동기기명과 그 작동기기의 동작상태를 표시해 준다.

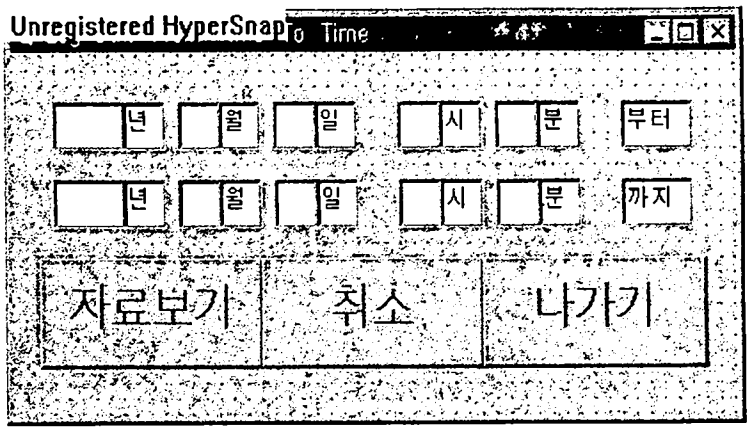


그림 6-10. 구간보기 모드의 화면

이러한 동작기기의 작동상황들과 각종 센서값들의 자료들을 비교, 검토함으로써 온실의 상황이 사용자가 원하는 대로 유지, 작동되고 있는지를 확인할 수 있다. 또한, 사용자가 원한다면 제어 Logic을 간단히 수정할 수 있다.

(3) 자료저장 모드와 파일보기 모드

자료저장 모드는 현재 화면에 나타나 있는 자료를 파일로 저장할 수 있는 모드이다. Directory box와 File box를 이용하여 자신이 원하는 드라이브의 원하는 디렉토리에 자료의 파일을 저장할 수 있으며, 파일의 형식은 '*.dat' 나 '*.txt'파일로 저장이 가능하도록 하였다.

파일보기 모드는 사용자가 이미 저장해 놓았던 자료파일을 불러서 화면상으로 나타내 줄 수 있는 모드이며, 자료보기 모드와 마찬가지의 형식으

로 일반 센서값들과 작동기기 동작상황들을 열람할 수 있다.

자료들을 날짜별로 저장해두면 편리하게 데이터 검색이나 분석을 행할 수 있다. 구간보기에서 오래 지나간 시간의 데이터는 하드웨어의 RAM에 없으므로 미리 자료저장 모드에서 자료들을 자신이 원하는 데로 저장시켜 놓은 후 파일보기 모드에서 파일을 열어 볼 수 있다. 그림 6-11은 자료저장 모드를 나타낸다.

마. 수동모드

수동모드에서는 제어 Logic에 따라서 제어를 할 수 없는 경우거나 사용자의 임의에 의해서 PC를 통해서 온실의 작동기기를 작동시켜야 할 필요가 있을 경우에 이용할 수 있다.

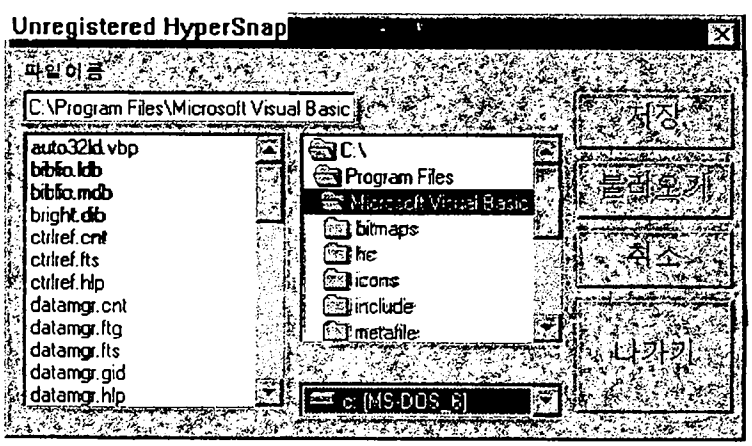


그림 6-11. 자료저장 모드의 화면

본 연구에서는 각종 창과 보일러, Fan 등 현재 실험용 온실에서 사용되는 작동기기 8가지를 직접 제어할 수 있도록 구성하였다.

측창과 천창 그리고 커튼은 멈춤과 열림, 닫힘으로 구분하여 열리거나 닫히는 중에도 사용자가 적당하다고 여겨질 때에 멈출 수가 있다. 그 이외의 작동기기(보일러, Fan, 지중온수 공급용 보일러, CO_2발생기)들은 동작과 멈춤의 옵션버튼이 있어서 사용자가 PC앞에서 직접 작동기기들을 작동시킬 수가 있다.

창이 수동 동작으로 작동될 때에는 열림 상태에서는 닫힘 버튼을, 닫힘 상태에서는 열림 버튼을 사용할 수 없게 해서 갑작스러운 모터의 반전을 방지하였다. 또한 하드웨어적으로는 각종 창에 리미터스위치를 장치하여 수동 작동시에 창이 끝까지 여리거나 닫힌상태에서 닫힘 버튼을 누르지 않았거나 다시 재동작 시킬때에도 자동으로 멈추거나 멈추어 있게 하였다.

그림 6-12는 수동모드를 나타낸다.

바. 하드웨어 setting 모드

하드웨어 setting 모드에서는 제어 Logic에 사용되는 자료나 온실내의 작동기기 동작에 필요한 간단한 옵션들을 설정해 줄 수가 있다. 또한 하드웨어가 가지고 있는 고유 번지(address)에 따른 용도와 사용범위, 사용방법, 개수 등을 지정할 수 있게 하였다.

정전시 온실내의 최고온도와 최저온도를 설정하여 비상동력을 이용하여 온실을 제어할 때에 정상적인 상태보다 작은 온도폭을 가지고 최소한의 작동기기를 동작할 수 있게 하여 전력의 소비를 최소화 하였다.

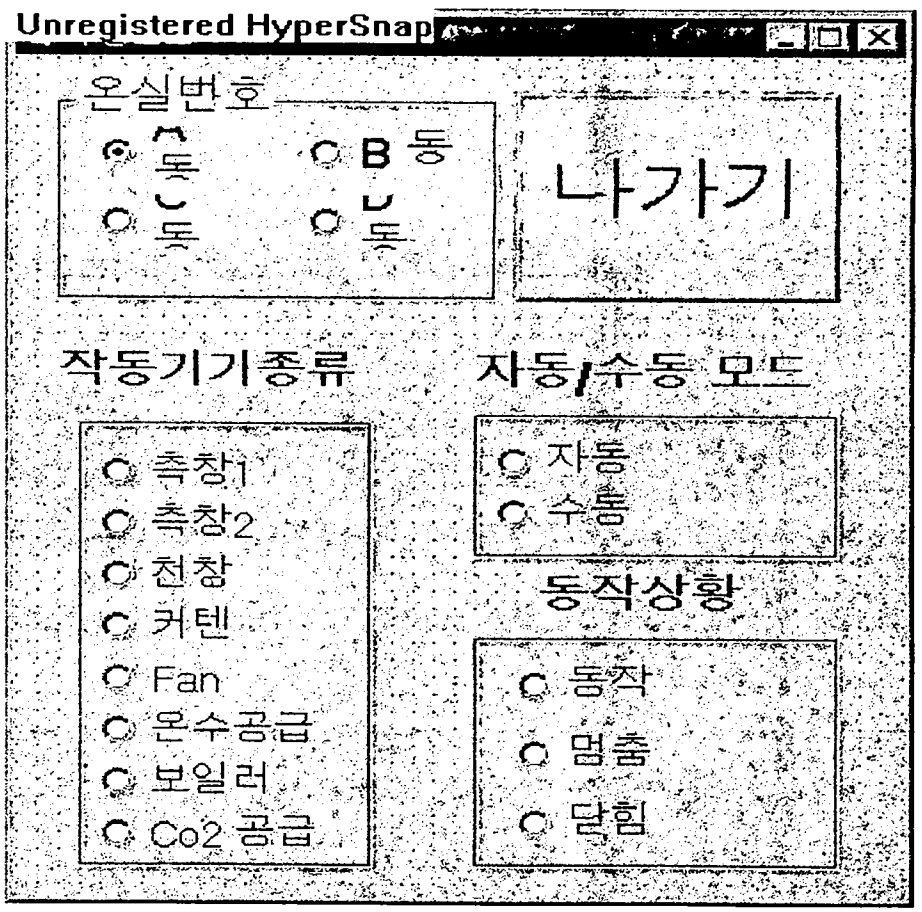


그림 6-12. 수동모드의 화면

창의 작동시에 한 번 움직일 때의 작동시간을 표시하여서 온실내의 환경을 조절할 때 좀더 점진적이고 효과적인 제어가 진행되어 질수 있도록 하였다. 겨울철과 같이 온실 내부와 외부 사이의 온도차이가 심한 경우에는 온실의 환경제어시 특히 온도가 갑작스럽게 큰 폭으로 변하게 되면 작물에게 심각한 장애를 줄 수 있기 때문에 제어 Logic상에서 창이 작동되는 경우에 있어서 갑작스럽게 큰 폭으로 작동되는 것을 방지하였다. 각 창들의 1step 작동시간을 설정하기 위해서는 각각의 창에 설치되어 있는 모터의 제원 및 속도를 알아야 한다.

창의 제어시, 온실 내·외부의 온도차가 10℃미만과 10℃에서 20℃사이, 그리고 20℃이상으로 구분하여 제어 Logic을 한 번 수행할 때 필요한 시간을 설정해주었다. 일반적으로는 온실의 실내외 온도차이가 심할수록 제어주기를 짧게 설정하여야 한다. 온도차이가 심할수록 적은 창개폐로도 온도차이가 적을 때보다 더 빠른 온도변화를 가져올 수가 있기 때문이다. 따라서 온실내의 갑작스러운 온도의 변화를 적당히 조절하기 위해서 창종류의 1step 작동시간과 더불어 상당히 중요한 부분이라 할 수 있다.

온실내 작물의 광합성은 빛의 영향이 가장 크다. 따라서, 본 연구에서는 일출 시간과 일몰 시간을 일정한 날짜 간격으로 미리 입력을 시켜 광조건이 맞지 않더라도 적당한 시간에 커튼을 열어서 작물이 광합성작용을 하는데 도움을 주도록 하였다. 현재에는 기본적으로 보름간격을 두고 계절에 맞게 적당한 시간을 설정해 주고 있으며, 작물의 종류와 특징에 따라서는 사용자가 임의로 설정치를 조절하여 사용할 수가 있다.

자료수집주기를 사용자가 임의로 설정하게 하여 필요에 따라서는 주기를 짧게 설정하여 짧은 시간동안 세밀한 변화를 관찰하게 할 수도 있고, 반대의 경우로 장기간에 걸친 온실의 전반적인 상황을 한눈에 알아볼 수 있게 하였다. 기본값은 10분(600초)으로 설정하였다.

하드웨어 setting 모드에서는 하드웨어상의 각종 입출력단자들의 번지(address)별로 그에 따른 용도와 사용범위, 사용방법 등을 설정하게 하였다. 이 부분이 하드웨어와 소프트웨어를 연결시켜주는 연결고리부분이 되며, 통신을 통해서 PC가 하드웨어로부터 어떤 값을 입력받았을 때 그 값이 어느 부분의 값이고 그 수치가 무엇을 의미하는지를 알 수가 있다. 하드웨어의 입출력단자 setting 부분은 기계적인 요소가 많이 가미되어 있고, 복잡하기 때문에 온실내 하드웨어의 특별한 변화가 있기 전에는 기존의 설정치를 가능한 바꾸지 않는 것이 좋다. 또한, 주사용자 이외의 사람들에 의한 돌발적인 설정치 변경으로부터 보호하기 위하여 이 모드에는 암호를 설치하여 쉽게 설정치들을 바꾸지 못하게 하였다.

각종 시간에 관련된 설정치들은 단위를 초로 하여 보다 정밀하고 점진적인 제어가 이루어지도록 하였으며, 환경설정 모드와 마찬가지로 각 입력치들을 입력한 후에 설정 버튼을 클릭하면 각종 설정치의 값들이 하드웨어상의 RAM에 저장되어 시스템 제어에 사용된다. 하드웨어 setting 모드는 그림 6-13에 나타내었다.

2. 제어 Logic의 구성과 내용

제어프로그램은 온실내 복합환경 제어시스템을 직접 구동시키는 Logic이며, PC로 프로그램하여 하드웨어상의 RAM에 저장시켜서 작동을 시킨다. 주기는 소프트웨어상의 하드웨어 setting에서 설정해 주는 제어주기에 맞추어서 주기에 한 번씩 제어 Logic loop를 실행한다.

기존의 제어방식이 각종 작동기기별로의 시간과 온도에 의한 개별제어 시스템인데 반하여, 본 연구에서 사용되는 제어방식은 온실내의 순수 온도와 습도를 조절해 주는 일종의 시퀀스제어 방식이다.

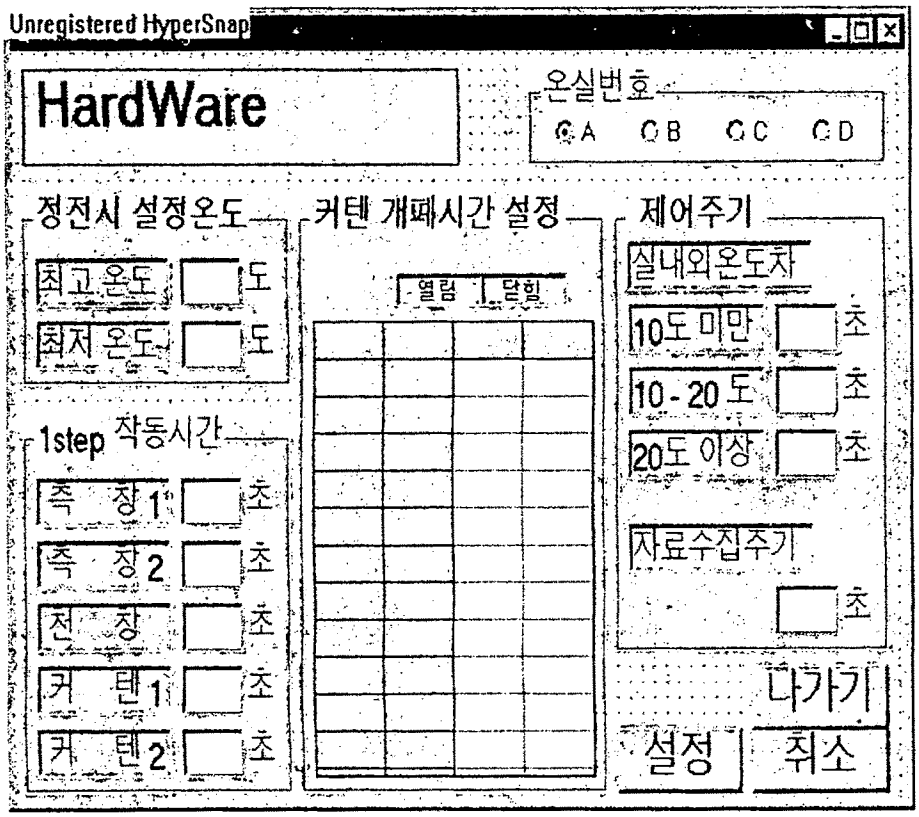


그림 6-13. 하드웨어 설정모드의 화면

소프트웨어상의 환경설정모드에서 설정해 주었던 최고온도, 최저온도, 최고습도, 최저습도가 RAM에 입력이 되어서 그 인자들을 이용하여 적정온도와 적정습도가 유지될 수 있도록 구성하였다. 1회 제어에 작동되는 창들의 개폐정도 및 작동시간은 하드웨어 설정모드에서 지정해 주는 시간에 따라 작동하게 된다.

온실의 복합환경 제어시스템의 제어 Logic은 Main Logic 부분과 그에 따르는 몇가지의 Sub Logic으로 나눌 수가 있다. Main Logic 부분에서는 기본적으로 온도와 습도 등 제어에 필요한 모든 환경인자들을 각종 센서로부터 읽어들이며, 센서로 부터 입력받은 현재의 온실상태값들을 환경설정에서 설정해주었던 설정값들과 비교를 하여 각종 제어를 실행하게 하였다.

센서로부터 입력받은 실제값들을 설정치들과 비교를 해주어서 온도와 습도 모두가 적정치에 있으면 그대로 제어 Logic을 빠져나가도록 하였다. 제어의 기본 비교개념은 크게 나누어서 온실내의 온습도 조건이 고온고습일 때, 저온일 때, 적온저습일 때, 적온고습일때로 구분하였다. 고온저습일 때는 물기를 뿌려서 온도를 낮추고 습도를 높여줄 수 있도록 하였다. 고온적습일때와 저온일때는 Sub Logic로 들어가서 각각 Cooling 과 Heating을 해 줄 수 있도록 하였다. 적온고습일때는 Fan을 작동시켜서 온실내의 습도를 낮출 수 있도록 하였다.

이상의 제어에도 온실상태가 원하는 적정한 상태가 되지 않을 때에는 경보를 발생시키며, 이를 사용자가 알 수 있게 하여 필요한 조치를 취할 수 있도록 한다.

지중온도 비교에 의한 지중온도의 가온과 일출 및 일몰시간에 따른 커튼의 개폐에 관한 Logic은 온실내의 온도, 습도에 의한 제어 Logic과는 별도로 해서 제어를 행하도록 하였다.

그림 6-14~그림 6-18은 Main Logic 과 각종 Sub Logic, 그리고 일출,

일몰에 의한 커튼의 개폐에 관한 Flow-Chart를 나타내는 것이다. 각 Flow-Chart에서 물음표("?")는 경보를 나타내는 것이며 이는 이 제어 Logic 상으로는 더 이상 온실의 제어를 할 수 없음을 뜻한다. 이럴 때는 사용자가 직접 수동 모드를 통해서 작동기기를 수동으로 동작시키거나 다른 별도의 조치를 취해야 하며, 작동기기가 이상없이 작동되고 있는가도 살펴보아야 한다.

다음은 Sub Logic의 Flow-Chart들의 설명이다. 크게 가열과 냉각으로 나눌수가 있으며, 각각은 작동기기별로 순위를 두어서 시퀀스 제어를 해주고 있다. 한가지 작동기기가 모두 동작하였을 때에도 온실내의 환경이 적정수준에 이르지 못했다면 다음 작동기기의 동작으로 넘어가게 된다. 제어 Logic상의 모든 작동기기들이 모두 작동되었을 때에도 불구하고 온실이 적정 상태에 이르지 못하면, 제어시스템은 경보를 발생하도록 하였다.

가. Heating Folw-Chart

측정된 온실내 온도센서의 값이 환경설정 모드에서 설정된 온실내 최저 온도보다 낮을 경우에는 습도의 정도에 상관없이 일단 가열을 하도록 하였다. 온실내의 온도를 높여줄 때에는 갑작스러운 온실내의 온도변화를 막기 위하여 온도에 영향을 작게 미치는 작동기기부터 작동을 시킨다. 하드웨어 설정모드에서 설정해준 제어주기시간마다 제어 Logic 의 loop를 실행시키며, 하나의 작동기기가 모두 작동이 된 후에도 온실내의 온도가 설정치보다 낮을 경우에는 다음의 작동기기를 작동시킨다.

실내온도가 낮을 때 작동기기들의 작동은 커튼, Fan, 측창, 천창, 보일러를 이용하여 온도제어를 실행하도록 하였다. 작동순서는 온도변화에 비교적 적은 영향을 미치는 커튼을 가장 먼저 체크를 하고, 그 다음으로 Fan, 측창, 천창 등의 순서로 제어를 행하도록 하였다.

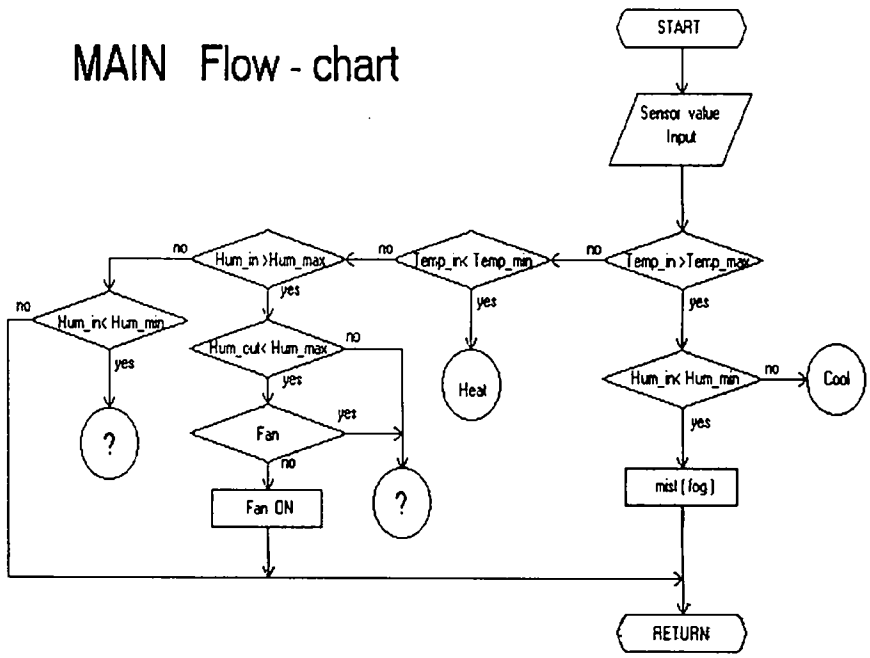


그림 6-14. 환경제어논리의 주 흐름도

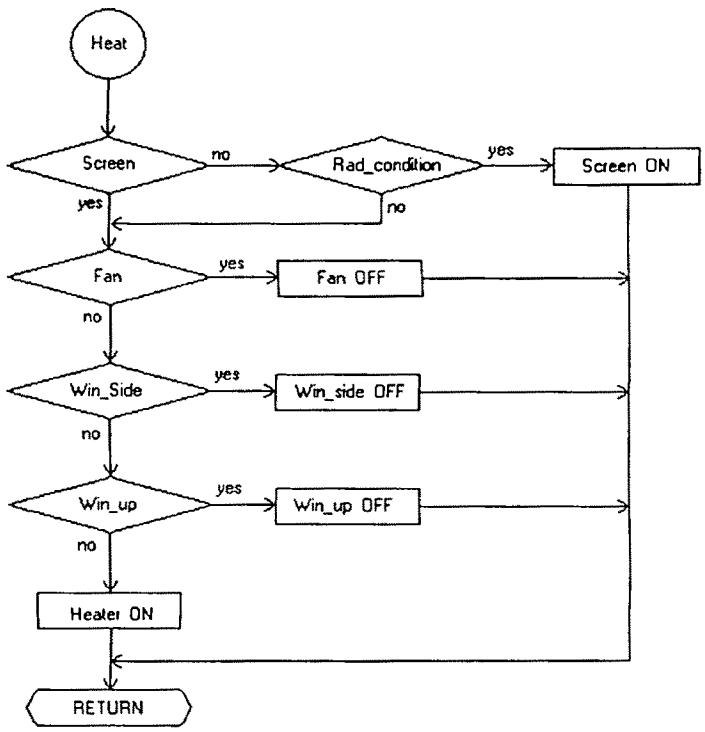


그림 6-15. 실내 온도조절과정의 흐름도(저온일 때)

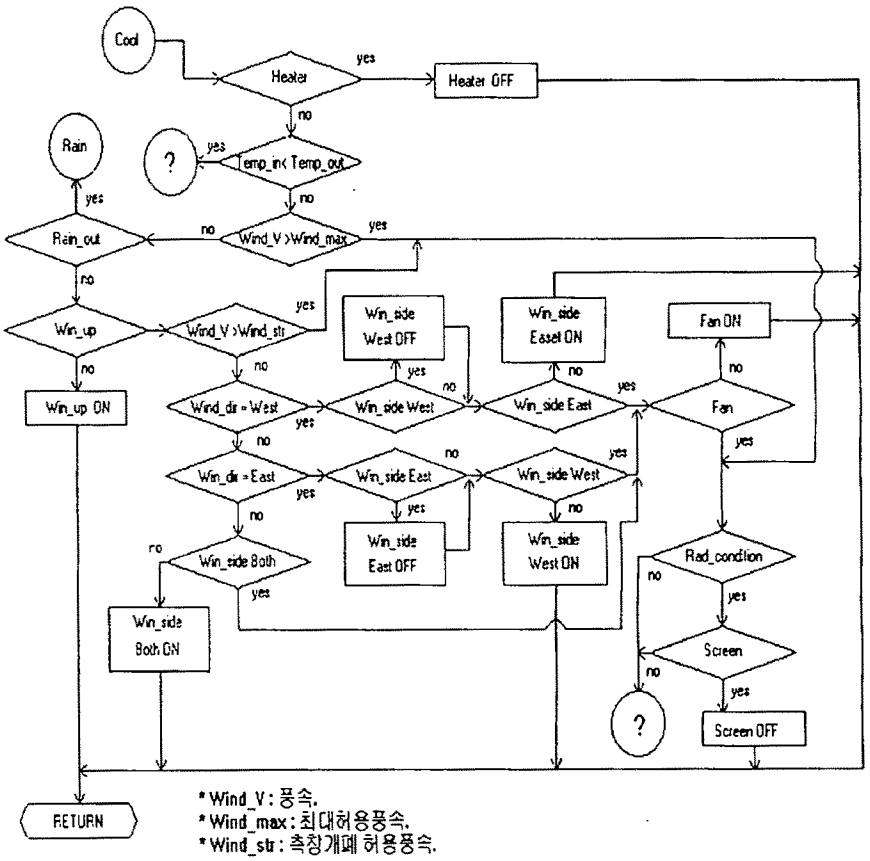


그림 6-16. 실내 온도조절과정의 흐름도(고온일 때)

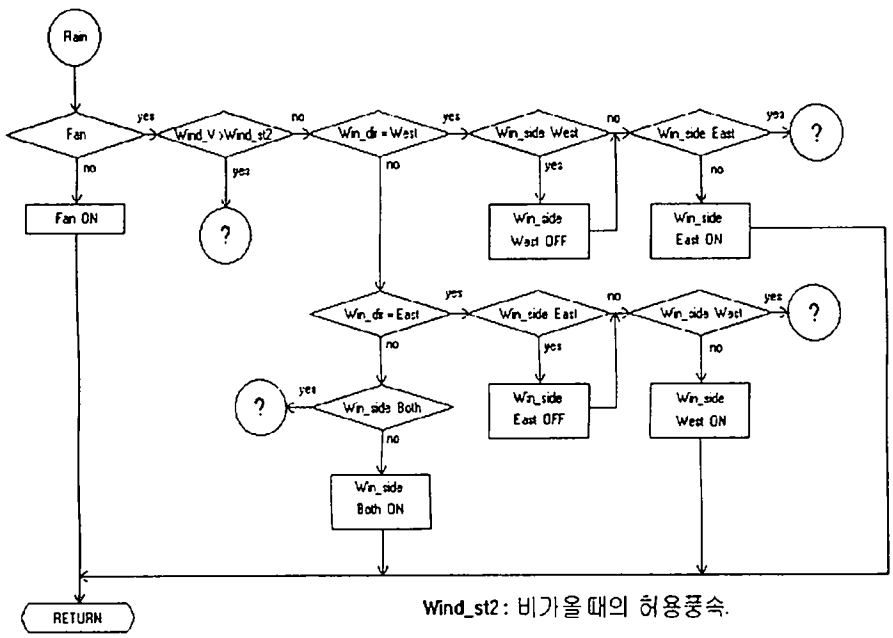


그림 6-17. 실내 온도조절과정의 흐름도(강우시)

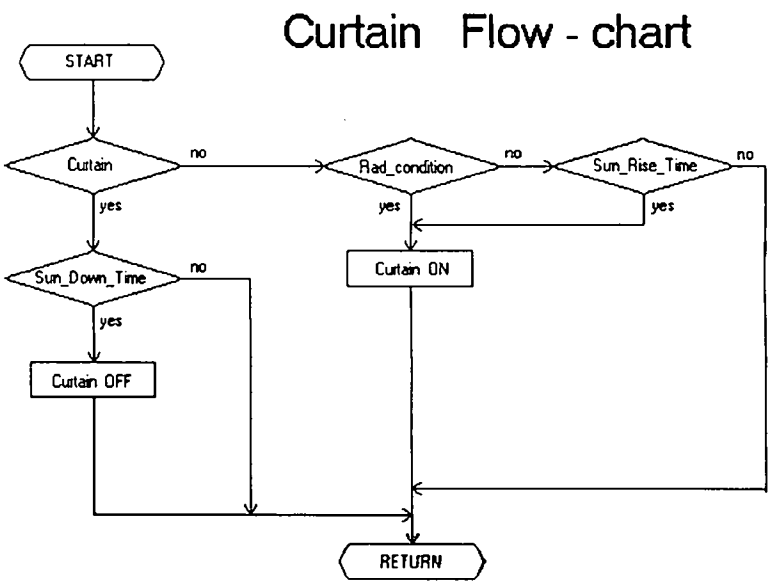


그림 6-18. 커튼동작의 흐름도

위의 모든 작동기기들이 모두 작동을 했을 경우에도 온실의 온도가 설정치 보다 낮게 나타나면 마지막으로 보일러를 가동시켜서 온실 내부를 가열시키도록 하였다. 보일러를 일정시간 이상 가동을 시켰을 경우에도 온실내의 온도가 낮게 나타나면 경보를 발생시키도록 하였다. 실내의 온도를 높일 경우의 순서도는 그림 6-15와 같으며, 자세한 과정은 다음과 같다.

Main Logic에서 실내의 온도를 높일 필요가 있다고 판단이 되면 Heating Flow-Chart로 Logic의 순서가 오게된다. 먼저 온도변화에 가장 영향이 덜 미치는 커튼의 현재 상황을 검색하여 커튼이 닫혀있다면, 커튼을 열어주어서 광에 의한 가열을 하게 된다. 이 때, 커튼을 열기 전에 외부의 광조건을 판단하여서 일정수준 이상의 광조건을 만족을 시켰을 경우에만 커튼을 작동시키게 된다.

커튼이 열려 있을 경우에 가열을 시키기 위해서 다음으로 Fan의 작동상황을 체크해 준다. Fan이 작동하고 있다면 Fan의 동작을 멈추게 하여 외부의 공기가 온실내로 들어오는 것을 막음으로써 보온의 효과를 낼 수 있다. 이 때에 실내와 실외의 온도차이를 비교하여서 실외의 온도가 더 높다면 이 동작을 행할 필요가 없으나, 온실 외부의 온도가 내부의 온도보다 낮을 경우에 온실내 온도가 설정치보다 낮게 나타나는 경우는 적기 때문에 이 경우는 무시하였다.

커튼과 Fan의 작동이 완료되었을 때에도 가열을 해줄 필요가 있을때에는 측창과 천창을 제어함으로서 가열의 효과를 얻도록 하였다. 순서는 측창을 천창보다 우선으로 체크하도록 하였다. 창들이 열려 있다면, Fan과 마찬가지로 창들을 닫아줌으로써 보온의 효과를 얻도록 하였다.

마지막으로 가열을 시키기 위한 동작기기는 보일러이다. 위에서 언급한 바와 같이 보일러는 일련의 시퀀스 제어에서 온실 내부의 온도를 높여주기 위한 마지막 수단이며, 일정기간 이상 보일러를 작동시켜도 온실내의 온도

가 낮게 나타난다면, 제어 Logic으로서는 더 이상의 가열의 효과를 기대할 수가 없으며 경보를 발생하게 된다. 일반적으로 온실이 적절히 밀폐가 되어있고 보일러의 화력과 온실의 크기가 균형을 이루고 있다면, 보일러를 가열시킴으로써 사용자가 원하는 온실내의 온도를 최저온도 이상으로 유지시킬수 있게 된다.

나. Cooling Flow-Chart

Main 제어 Logic에서 측정된 센서값들과 환경설정 모드에서 설정해 준 값들을 비교하여서 온실내 측정온도가 설정된 최고온도보다 높고, 측정된 온실내의 습도가 설정된 온실내 최저습도보다 낮지 않을 경우라고 판단이 될 경우에 Cooling 과정을 수행하게 된다. 온도가 높고 습도가 낮을 경우에는 안개를 발생시켜서 감온가습의 효과를 얻을 수가 있으나, 대부분의 일반적인 경우에서 온실의 온도가 높을 경우에는 온실내의 습도가 상당히 높기 때문에 안개를 발생시켜 주어야 할 경우는 상당히 드물다고 할 수 있다.

감온의 과정은 원칙적으로 가열과 반대과정으로 하였으며 가열의 과정보다는 비교적 온실의 급작스러운 온도와 습도의 변화를 가져올 우려가 높기 때문에 가열과정에서 보다는 신중한 제어 Logic이 필요하다. 작동기기의 동작 순서는 가열과 반대로 보일러, 천창, 측창, Fan, 커튼의 순서이다. 가열과정과는 다르게 감온과정을 수행할 때에는 외부의 공기 조건을 비교하였다. 외부의 온도와 습도를 비교하여 제어를 하였고, 바람의 방향과 강우의 유무를 판단하여 감온 제어를 해 주었다. 특히 강우시에는 비가 오지 않을 때와는 다르게 빗물이 온실 내부로 들어오는 것을 방지해야 하기 때문에 비가 올 때와 비가 오지 않을 때의 작동기기 제어 순서를 다르게 하여 Cooling Logic을 구성하였다.

가열과정과는 반대로 Cooling Logic에 들어오면 가장 먼저 보일러의 작

동 여부를 판단하여 보일러가 작동하고 있다면, 보일러의 동작을 멈추도록 제어 순서를 정하였다.

보일러가 작동되지 않을 때에도 온실내부의 온도가 설정온도보다 높을 때에는 천창과 측창을 체크하여서 온실의 온도를 낮추어야 하는데 천창이 측창보다는 우선적으로 작동하도록 하였다. 따라서, 본 시스템에서는 창이 닫혀 있다면 창을 열어주어서 온실내의 온도를 낮추는 효과를 가져올 수 있도록 하였다.

천창의 열림 동작을 하기 이전에 환경설정 모드에서 설정된 창문개폐허용풍속과 외부의 풍속을 비교하여서 외부의 현재 풍속이 그 설정치를 넘지 않았을 때에 창문을 열도록 하였다. 특히 천창의 경우에는 온실의 위쪽에 설치가 되어있기 때문에 비가 올때에는 온실내부가 비가 들어오는 확률이 높고 작물에 큰 피해를 줄 수도 있기 때문에 강우의 유무를 판단하여서 비가 오는 경우에는 감온과정에서 천창의 열림동작은 제외시켰다.

천창의 작동이 완료된 후에도 온실 내부의 온도가 설정치보다 높을 경우에는 측창을 제어하여 실내의 온도를 낮추도록 하였다. 측창을 열어주기 이전에 먼저 외부의 풍속이 측창개폐허용풍속보다 작은지를 판단하여 측창개폐허용풍속조건이 맞을 경우에는 바람의 방향을 고려하여 양쪽 측창의 개폐순서를 정하도록 하였다. 측창의 방향과 바람의 방향을 비교하여 측창중 한쪽 방향의 창과 바람의 방향이 일치하는 경우에는 그쪽의 측창을 먼저 열도록 하였으며, 바람의 방향이 측창의 양쪽 방향 사이에 있을 경우에는 양쪽 측창을 동시에 열도록 하였다. 바람의 방향을 고려하여 양쪽 측창의 개폐순서를 정해 줌으로써 온실 내부의 급작스러운 온도의 변화를 최소화하도록 하였다.

보일러와 창의 작동이 완료된 후에도 온실내부의 온도가 설정치보다 높을 경우에는 Fan의 작동상황을 체크하여서 Fan이 동작하지 않을 경우에는

Fan을 동작시켜서 온실의 온도를 낮추도록 하였다. Fan이 동작하고 있는 경우에도 온도가 적정수준이 되지 못하면, 광조건과 커튼의 동작상황을 비교하여 커튼이 작동하도록 하였다. 위의 제어로써도 온실의 온도가 적정수준으로 제어가 되지 못한다면 경고를 발생하여 사용자가 알 수 있게 하여 사용자가 임의로 다른 조치를 취할 수 있도록 하였다.

감온과정에서는 창의 열림과정이 필요하기 때문에 비가 올 때와 오지 않을 때로 구분하여 비가 올 때는 온실내로 빗물이 들어오는 것을 방지하도록 하였다. 또한, 본 시스템에서는 비가 올 때와 오지 않을 때에 따라서 제어 Logic상의 순서와 동작되는 작동기기의 종류를 다르게 하였다. 비가 올 때 감온과정의 경우에서는 천창의 열림과정을 생략하였으며, 측창의 개폐시에도 비가 오지 않을 때는 허용풍속보다 낮은 강우시 측창개폐허용풍속을 고려하여 제어하였다.

다. Curtain Flow-Chart

커튼의 제어 Logic은 가열이나 감온시에 작동되는 것 이외에 별도로 작물의 광합성과 일사량과의 밀접한 관계를 고려하여 설계하였다.

광조건이 맞지 않을 경우에라도 일출 한 시간 후에 커튼을 열리게 하였고 일몰 두 시간전에 커튼을 닫음으로써 작물의 광합성 작용이 최대한이 될 수 있도록 하였다. 광이 미약한 상태일지라도 작물은 일정한 시간이 되면 광합성 작용을 하기 때문에 이러한 과정은 반드시 필요하다 할 수 있으며 단순히 광도 조건만을 가지고 커튼을 제어해 주게 되면 해가 떠있는 낮 시간에 온실내의 온도와 습도 조건이 적절한 상황에도 불구하고 커튼이 닫혀있어서 작물의 광합성 작용에 악영향을 끼칠 우려가 있기 때문이다.

일정한 시기별로의 일출시간과 일몰시간은 기본적으로 15일을 기준으로 (한달에 2번- 매월 1일과 15일 기준) 기상대의 자료를 이용하도록 하였다. 그러나 그 시간이 아주 정밀할 필요는 없으므로 해가 뜨고 적당한 시간이

지난 후에 커튼을 열어주고 해가 지기 적당한 시간전에 커튼을 닫아주면 큰 무리가 없기 때문에 대략적인 시간을 미리 입력을 시켜주면 큰 무리가 없다.

그림 6-19는 센서의 입력에서부터 가열, 감온 및 강우시의 감온에 있어서의 작동기기들의 동작 순서를 나타낸다.

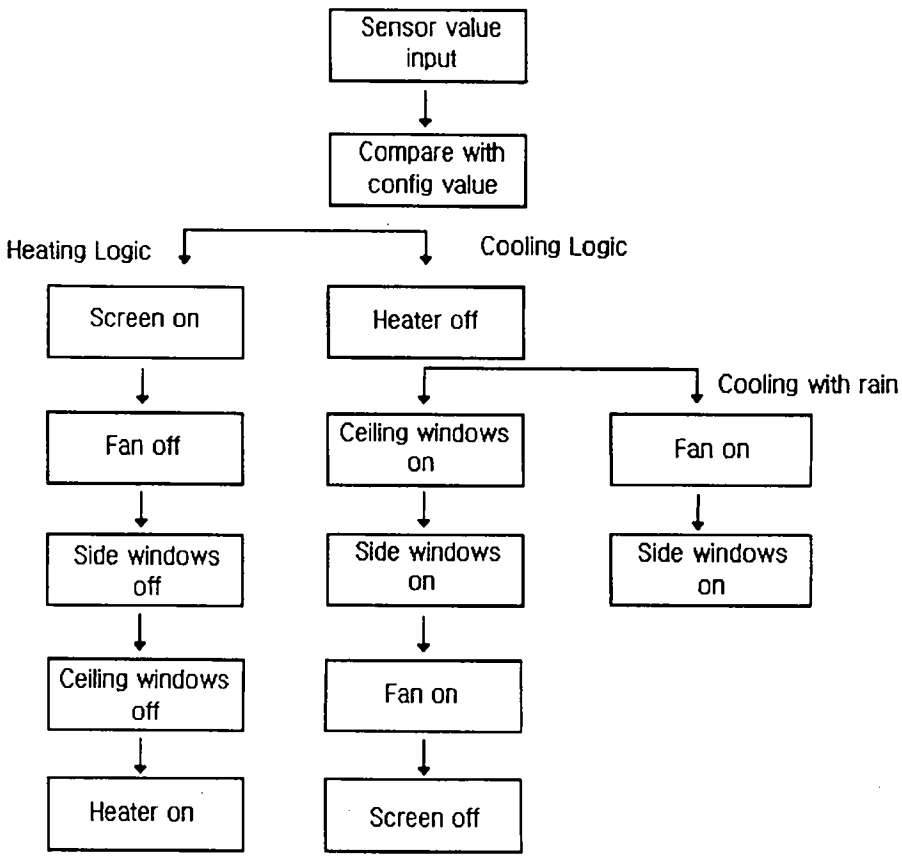


그림 6-19. 온실환경제어의 제어순서

제 4 절 복합환경 제어시스템의 통신프로그램

온실의 복합환경 제어시스템에서 통신프로그램은 필수적이다. 특성상 센서값들과 작동기기 동작신호를 하드웨어상에서 주고 받아야 하며 사용자가 PC에서 하드웨어와 각 온실의 상황들과 수집된 자료의 열람 및 편집을 위해서는 반드시 원할하게 통신이 되어야 한다.

본 연구에서는 통신을 행함에 있어서 빠른 속도를 지니지 않아도 되기 때문에 비동기 직렬(Serial)통신을 사용하였다. 데이터의 송수신은 1BYTE(8bit) 단위로 하였으며, 통신 종류로는 CRC32와 485통신을 이용하였다. 에러체크는 CRC나 BCC를 이용하였다. 통신프로그램을 작성한 Language는 PC의 경우에는 시스템 구동 소프트웨어와 마찬가지로 Windows용 프로그램 언어인 Visual Basic을 사용하였으며, 하드웨어상에서의 통신프로그램은 C언어를 주로 사용하였고, 필요에 따라서는 Assembler어를 사용하기도 하였다.

본 연구에 이용되는 통신프로그램의 Protocol은 Z-MODEM Protocol을 기본개념으로 하였으며, Header의 종류를 나타내는 Header Code와 PC에서 하드웨어로 또는 하드웨어에서 다른 모듈로의 명령을 내리는 Command Code에 충분한 여분을 두어서 나중에 확장이 필요한 경우에 큰 무리가 없이 확장할 수 있게 하였다.

통신 코드의 기본 개념은 Header와 Data Subpacket으로 이루어 지며, Header의 종류에 따라서 뒤에 데이터가 뒤따라 올 수도 있고 데이터가 없이 Header만 전송 되는 경우도 있다.

1. 통신의 기본개념

통신에 사용되는 모든 사양은 Z-MODEM 방식을 기본 개념으로 하여 대부분의 Protocol을 그에 상응하도록 하였다. 어느 경우의 전송에 있어서든지 Header가 먼저 전송이 되고 그 Header의 종류와 Command Code에 따라서 뒤에 데이터가 일정한 분량의 단위로 전송이 되는 경우도 있고 그렇지 않고 데이터의 수반없이 Header자체만이 전송되는 경우도 있다.

Header는 에러체크 BYTE까지 포함하여 기본적으로 12BYTE로 구성이 된다. Header의 앞 2BYTE는 Header의 시작을 알리는 ZPAD와 ZDEL(Z-Delete 문자)가 놓이게 되며, 이는 16진수로는 0x2A와 0x24로 표기가 된다. 그 뒤에 세 번째 BYTE는 에러체크의 종류를 알리는 HEAD BYTE가 오며 그 종류에 따라서 에러체크 방식이 CRC 체크인지 아니면 BCC 체크인지 등을 결정하게 된다. Header의 네 번째 BYTE는 Header의 종류를 알리는 Header Type BYTE가 오며 그 Header Type에 따라서 Header의 종류와 역할이 부여된다. 그 뒤로 수신부의 모듈번지(address)를 알리는 ADDR1 1BYTE와 수신부의 해당 모듈내의 번지를 알리는 ADDR2 1BYTE가 뒤따르게 된다. Header의 일곱 번째는 뒤에 따르는 데이터가 있는지 없는지를 판별해 주는 D-Flag(Data Flag)가 1BYTE, Header의 명령 Code를 담고 있는 Command CODE 1BYTE가 온다. Header의 가장 끝에는 Header가 전송되는 도중에 생기는 손실이나 변형된 자료의 수신을 막기 위하여 HEAD BYTE에 있는 에러체크의 종류에 따른 방식으로의 에러체크를 하여 4BYTE를 보낸다.

HOST(PC)나 RTU(hardware)에서는 이 Header의 TYPE과 Command Code에 대응하는 데이터를 전송시키거나 또는 응답을 하도록 프로그램하였다. 표 6-3은 전송 Header의 구성을 나타낸다.

Header 뒤에는 옵션으로 데이터가 따라오게 되며, 전송도중에 발생하는

전송에러를 최소화하기 위하여 데이터의 양이 많을 경우에는 몇 개의 블록으로 나누어서 연속으로 전송하도록 하였다.

표 6-3. 헤더의 구성

BYTE 수	이 름	16 진수	내 용
1	ZPAD	0x2A	Header 의 시작을 알린다
2	ZDEL	0x24	
3	HEAD	'A', 'B', 'C', 'D'	뒤에 따라올 Error Check의 종류를 알린다.
4	TYPE		Header의 종류를 알린다.
5	ADDR1		수신부 모듈의 고유 번지(Address)를 알린다.
6	ADDR2		수신부의 해당모듈 내에서의 고유 번지(Address)를 알린다.
7	D-Flag	0x1, 0x2	뒤에 따라오는 데이터의 유무를 알린다.
8	CODE		Header가 수신측에 보내는 명령코드를 실는다.
9, 10, 11, 12	Error Check		전송도중에 생긴 Error에 의한 잘못된 신호를 수신하는 것을 방지한다.

현재는 500 BYTE씩으로 나누어서 전송하고 있으며, 전송 선로가 개선되면 데이터의 한 블록의 크기를 더 크게 할 수 있을 것으로 사료된다.

데이터 블로에도 Header에서와 마찬가지로 데이터 블록의 맨 뒤에 4BYTE를 에러체크 BYTE로 하여 에러를 검출해 내게 된다. 데이터의 한 블록에서 전송에러가 검출이 되면 그 에러가 난 블록에서 부터 다시 전송을 시도하게 되며 그 이전에 전송되어진 데이터에는 영향을 주지 않는다.

Header의 상위 3BYTE를 제외하고는 통신에서 사용되는 통신의 고유 Code와 혼선을 피하기 위하여 Escape-Character를 지정하여 그 문자가 전송이 될 때에는 특별한 표식을 하여서 바꾸어 보내고 수신측에서는 그러한 특별한 표식을 전송받았을 경우에는 전송시와 반대의 경로로 그 문자를 해석하여 원래의 자료로 복원을 시킨다. 이는 데이터 블록뿐만 아니라 Header의 상위 3BYTE를 뺀 나머지 부분과 에러체크 4BYTE에도 해당되도록 하였다.

2. 통신사양

기본적인 통신방식은 비동기 직렬통신(serial)통신을 사용하며 1BYTE단위로 송신과 수신을 하도록 하였다. 에러체크 BYTE를 Header 와 Data Subpacket에 곧바로 실어보내기 때문에 Parity Bit는 사용하지 않았다. 본 연구에 사용된 통신사양을 표 6-4와 같다.

자료를 송수신할 때에는 Frame의 구조를 가지고 전송을 하게 되며, 맨 앞에 Header가 전송이 되고 그 뒤를 이어서 Header의 종류에 따라서 Data Subpacket이 한 개 또는 여러 개가 전송되도록 하였다. 전송에 사용되는 Frame의 기본 구조는 그림 6-20과 같다.

표 6-4. 통신의 기본사양

구 분	내 용
Baud Rate	9600 BPS
Parity Bit	None
Data Length	8 Bit
In Buffer Size	500 Byte
Out Buffer Size	500 Byte
RThreshold	1
Stop Bit	1 Stop Bit
Protocol	Z-Modem Protocol
Error Check	CRC or BCC etc.

그림 6-20. 프레임의 구조

가. Header의 종류와 형식

Header는 Header의 시작을 알리는 2BYTE와 Header의 종류와 특성을 나타내는 5BYTE, Header의 Command Code를 표시하는 1BYTE, 그리고 Error Check를 나타내는 4BYTE등 총 12BYTE로 구성되어진다. Header의 구조는 그림 6-21에 나타내었다.

본 시스템의 통신을 위하여 사용되어지는 Header의 개수는 6개이며, 그 종류는 TYPE의 종류와 일치한다. 필요에 따라서는 Header의 개수를 늘릴 수도 있다.

(1) ZPAD 와 ZDLE

ZPAD와 ZDLE가 순서적으로 동시에 쓰이게 되면, 수신측에서는 그 이후의 10BYTE를 Header로 인식하게 되고 그 Header를 해석함으로써 다른 응답을 하거나 또는 필요한 데이터를 올려보낼 수가 있다.

(2) HEAD

현재 전송되어지는 Header와 그에 따르는 데이터들의 에러체크 방식을 나타내 주는 신호로써 본 연구에서는 4가지의 종류를 사용하고 있다. 전송상태와 기대대는 최소한의 정확성, 하드웨어의 특성에 따라 각기 다른 종류의 에러체크를 하고 있으며 32 bit CRC Check와 BCC Check를 많이 사용하고 있다. HEAD의 종류와 그 내용을 표 6-5에 나타내었다.

(3) TYPE

TYPE의 종류에 따라서 Header의 종류와 특성이 규정되어 지는 것이다. 본 시스템에 사용되어지는 Header의 종류는 총 6가지 이며 필요에 따라서는 더 개수를 확장시켜서 사용할 수도 있다.

표 6-6에서는 본 시스템에 사용되어지는 Header의 6가지 종류와 그 성격을 정리한 것이며, 그림 6-22~그림 6-27은 각 Header의 모식도를 나타낸다. 다음은 각 Header의 종류별로의 특징과 간단한 설명을 한 것이다.

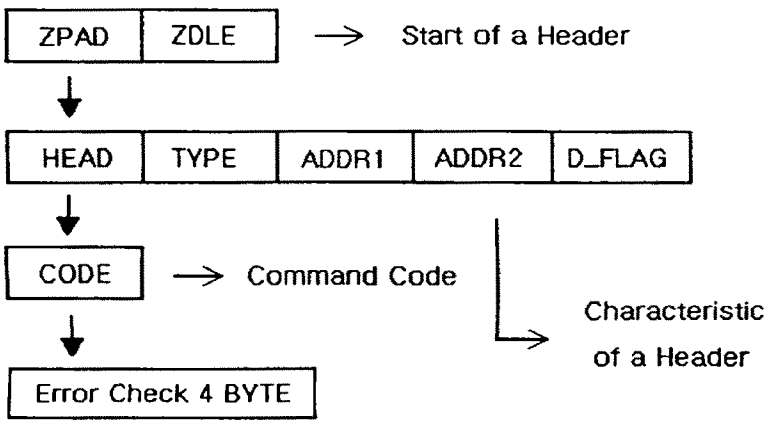


그림 6-21. 헤더의 구조

표 6-5. 에러체크의 종류

HEAD의 종류	내용
"A"	16 Bit CRC 2진 (Binary)
"B"	BCC Check
"C"	32 Bit CRC 2진 (Binary)
"D"	32 Bit CRC RLE PACK 2진 (Binary)

(가) ZRQINIT

PC나 각 하드웨어에서 통신을 처음으로 시도하고자 할 때에 사용되는 Header로써 수신측에서는 통신이 열려 있고 ZRQINIT를 제대로 받았다는 신호로써 ZRINIT를 송신측에 전송하게 된다.

표 6-6. 헤더의 종류

No.	종 류	사 용 목 적	비 고
0	ZRQINIT	통신을 처음으로 시도할 때	Pri. ===> Sec.
1	ZRINIT	ZRQINIT를 성공적으로 수신했을때	Pri. <=== Sec.
3	ZACK	데이터를 성공적으로 수신했을 때	Pri. ===> Sec.
6	ZNAK	데이터의 수신이 성공적이지 못할때	Pri. <==> Sec.
18	ZCOMMAND	송신측과 수신측사이에서 명령을 주고 받을 때	Pri. <==> Sec.
8	ZFIN	통신을 완전히 종료할 때	Pri. ===> Sec.

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	0	ADDRESS			HOUSE NUM	BCC
A, C, D	0			0		CRC-32

그림 6-22. ZRQINIT의 구조

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	1	ADDRESS		ERROR CODE		BCC
A, C, D	1			ERROR	VER	CRC-32

그림 6-23. ZRINIT의 구조

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	3	ADDRESS			CODE	BCC
A, C, D	3	OFF SET			CODE	CRC-32

그림 6-24. ZACK의 구조

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	6	ADDRESS			CODE	BCC
A, C, D	6	0	0	0	CODE	CRC-32

그림 6-25. ZNAK의 구조

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	18	ADDRESS		D-FLAG	CODE	BCC
A, C, D	18			D-FLAG	CODE	CRC-32

그림 6-26. ZCOMMAND의 구조

HEAD	TYPE	ZP0	ZP1	ZP2	ZP3	CRC OR BCC
B	8	ADDRESS				BCC
A, C, D	8	0	0	0	0	CRC-32

그림 6-27. ZFIN의 구조

TYPE Number는 '0'이며, 신호를 보내고자하는 온실의 번호를 7번째 Bit에 표시한다. ADDRESS 부분에 해당하는 온실에 있는 수신측의 고유의 모듈번지와 모듈내번지를 3번째와 4번째 Bit에 실어보낸다.

(나) ZRINIT

송신측에서 보내온 ZRQINIT신호를 성공적으로 수신하였을 때 답신하는 Headerd이며 아대부터 실제적인 데이터를 전송할 수 있게 된다. 7번째 Bit에 송신측에서 보내온 온실의 번호를 전송함으로써 송신측에서는 ZRINIT Header를 받아봄으로써 자신이 원하는 온실의 번호인지를 확인할 수가 있다.

TYPE Number는 '1'이며 ZRQINIT에서 전송에러등으로 인하여 수신이 제대로 되지 않았을 경우에는 ERROR Code를 보내어서 송신측에서 보내온 신호가 맞는 것인지를 송신측에서 확인해 볼수 있도록 한다.

(다) ZACK

Header는 물론이고 Data Subapcket의 수신을 할 때 전송에러가 없이 제대로 받았다고 송신측에 전하는 신호이다. 이 신호는 송신측에서 Header 및 데이터를 전송할 때에 ZACK신호의 응답을 원하는 경우가 있는데 송신측에서 원하는 경우에만 전송을 해주면 된다. TYPE Number는 '3'이다.

(라) ZNAK

ZACK신호와 상응하는 신호로써 송신측에서 보내온 신호가 전송에러 등의 이유로 제대로 수신이 되지 않았을 경우에 신호를 제대로 받지 못했다는 것을 송신측에 알리는 Header이다. 이 Header는 직전에 전송되어왔던 신호(Header 또는 Data Subpacket)를 다시 한 번 전송해 달라는 요청을 송신측에 하는 것이기도 한다.

만약 송신측에서 전송되어온 신호가 Command Code 를 가지고 있었다면 그 Code를 8번째 비트에 실어 보냄으로써 송신측에서는 어떠한 명령전

송에서 전송에러가 발생하였는지를 알 수 있다.

송신측에서는 ZNAK신호를 받았을 때 직전에 전송시켰던 Header나 Data Subpacket을 다시 한 번 보내주어야 한다. 만약 계속해서 ZNAK신호로 응답을 한다면 연결선로에 이상이 있거나 하드웨어적인 결함이 있는 것이기 때문에 이를 살펴보아야 한다. TYPE Number는 '6'이다.

(마) ZCOMMAND

수신측에 특수한 목적의 데이터나 수동모드에서의 작동기기의 수동동작 신호를 보낼 때 사용 되는 Header이다. TYPE Number는 '18'이며 8번째 Bit에 Command Code를 실어 보냄으로써 송신측에서 보내는 명령이 어떤 것인지를 수신측에서 판단을 할 수가 있다.

실제적인 데이터 전송이나 작동기기의 동작 및 온실의 환경제어에 영향을 미치는 Header이며 가장 많이 사용이 된다. 특히 송신측에서 수집된 자료의 전송등 실제 자료의 전송을 원할때에는 수신측에서는 같은 ZCOMMAND Header와 함께 송신측에서 원하는 자료를 전송해주게 된다. Command Code가 필요에 의해서 확장 시킬때에는 Command Code를 추가 시켜서 사용할 수가 있다. Command Code를 표 6-7에 나타내고 있다.

(바) ZFIN

한 단위의 전송이 모두 끝났을 때 송신측에서 수신측으로 통신을 끊기 위해 보내는 Header로써 그에 대한 응답은 필요가 없으며 수신측에서는 ZFIN Header를 전송받았을 때에는 통신선로를 끊으면 된다.

마찬가지로 송신측에서도 ZFIN Header를 전송시킨 후에는 응답을 기다릴 필요가 없이 바로 통신을 끊으면 된다. TYPE Number는 '8'이다.

표 6-7. 명령코드의 종류

Code	내 용	비 고
0x30	Date Send	Host ==> Receiver
0x31	Date Correct	Host <== Receiver
0x32	Date Mismatched	Host <== Receiver
0x40	Normal Data Request	Host ==> Receiver
0x41	Normal Data Send	Host <== Receiver
0x42	Event Data Request	Host ==> Receiver
0x43	Event Data Send	Host <== Receiver
0x44	Normal Data (From Time To Time) Request	Host ==> Receiver
0x45	Normal Data (From Time To Time) Send	Host <== Receiver
0x46	Event Data (From Time To Time) Request	Host ==> Receiver
0x47	Event Data (From Time To Time) Send	Host <== Receiver

표 6-7. (계속)

0x48	Current Data Request	Host ==> Receiver
0x49	Current Data Send	Host <== Receiver
0x4A	Control Data Send	Host ==> Receiver
0x54	Hardware Configuration Request	Host ==> Receiver
0x55	Hradware Configuration Send	Host <== Receiver
0x60	Configuration Request	Host ==> Receiver
0x61	Configuration Send	Host <== Receiver
0x65	Manual-Mode Data Request	Host ==> Receiver
0x66	Manual_Mode Data Send	Host <== Receiver

(4) ADDR1 , ADDR2

ADDR1과 ADDR2는 하드웨어상에서 사용할 부분에 대한 고유의 번지(Address)를 지정해주는 것이다. ADDR1에서는 필요한 부분의 모듈의 번호를 지정해주는 것이고 ADDR2는 그 해당하는 모듈안에서의 각 입출력 단자의 번지이다. 이론적으로는 한 Controller에서 256개(2^8개)까지의 모듈을 연결시켜 사용할 수가 있고 각각의 모듈에는 256개의 단자를 연결시킬 수가 있다.

여기서 사용되는 모듈번지와 모듈내의 번지는 구동 소프트웨서상의 하드웨어 입출력 단자 번지지정에서 설정해 놓은 설정치와 일치하는 번지수이다.

하드웨어상의 각각의 입출력단자들이 고정적인 목적으로 사용되는 것이 아니고 이처럼 하드웨어 설정에서 지정해주는 각 번지들과 통신 프로그램상의 Header에서 지정해주는 Address를 일치시켜 줌으로써 시스템의 변형과 수리, 확장을 하는데 편리하게 이용될 수 있다.

(5) D-FLAG

Header뒤에 Data Subpacket이 추가가 되는지의 여부에 따라 값을 다르게 설정해 준다. 값이 '1'이면 뒤에 데이터가 따라오는 것이고 그 이외의 값이면 뒤에 따르는 데이터가 없이 그냥 Header만 전송되는 것이다.

나. Data Subpacket의 구조

Haeader의 D-Flag(7번째 Bit)가 '1'인 경우에 그 뒤에 따라가는 실제적인 데이터 블록의 형식은 그림 6-28에 나타내고 있다.

통신과정에 있어서 또는 자료의 수집이나 현재 입력되고 있는 센서값들이나 작동기기 동작에 관한 Digital Input/Output 값들을 전송하는 부분이다. 통신에서의 실제적인 정보가 담기는 부분이며 Header에 비하여 정보의 중요성이 상대적으로 클 뿐만 아니라 한 블록의 크기도 크기 때문에 각

각의 블록마다 에러체크를 반드시 해 주어야 하며 가능하면 ZACK신호를 요구하여 송수신이 잘 되고 있는 지를 확인하면서 전송시키는 것이 바람직하다.

블록의 크기는 1000 Byte 까지가 일반적으로 사용되어지는 범위의 최대값이며, 그 이상이 되어도 상관은 없으나 데이터의 블록이 커지면 전송시 에러가 발생할 확률이 높고 에러발생 후에 재전송할 때의 시간적인 손실이 따르게 된다. 본 시스템에서는 한 블록당 500Byte를 기본으로 하여 데이터의 송수신을 하고 있다.

일정 분량의 데이터가 전송되면 그 바로뒤에 데이터의 한 블록이 (또는 전체 블록이) 끝났다는 신호를 보내주어야 한다. ZDLE 신호와 Data End 신호를 함께 붙여서 보냄으로써 수신측에서 데이터 블록이 끝이 났음을 알 수 있게 한다. Data End Sign의 종류에 따라서 뒤에 다른 데이터 블록이 있는지의 유무와 ZACK를 요구하는지를 판단하게 된다.

Data Subpacket의 각 부분에 대한 설명과 Data End Sign의 종류와 내용을 표 6-8과 표 6-9에 나타내었다.

다. 에러체크와 특수문자의 전송

Header와 데이터를 전송할 때 전송선로등에서 생길수 있는 전송에러들에 의한 오류를 방지하기 위하여 CRC나 BCC 에러체크를 많이 이용하였다.

비교적 프로그램이 간단한 CRC 에러체크는 PC용 소프트웨어에서 많이 이용하였고, 긴 거리에서 비교적 에러가 적은 BCC 체크를 하드웨어에 주로 이용하였다.

소프트웨어에 사용된 CRC32 에러체크는 미리 계산되어져 있는 256개의 8자리 16진수들을 배열에 저장시켜놓고 각 BYTE마다 계산을 해 주어서 4BYTE의 에러체크 BYTE를 전송하게 된다.

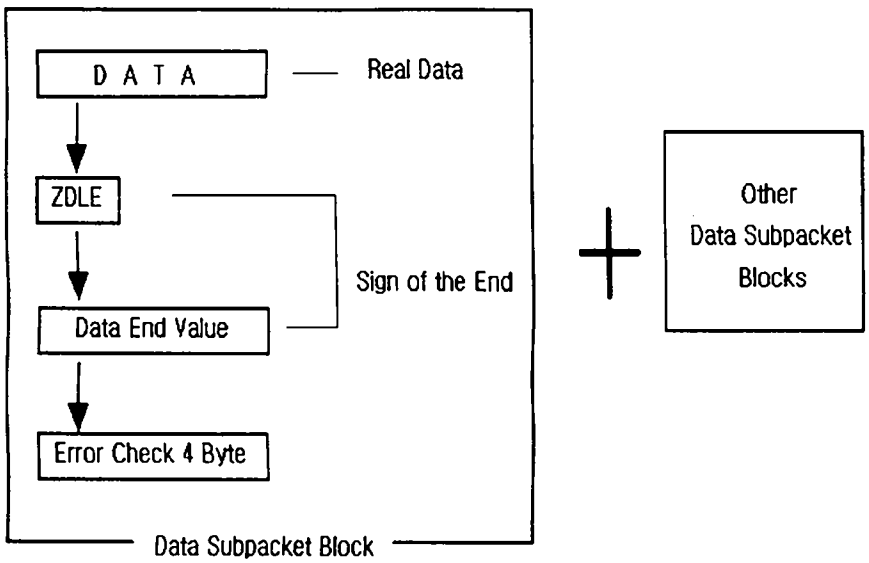


그림 6-28. 데이터 전송블록의 구조와 연결

표 6-8. 데이터 전송블록의 구성

종 류	크 기	내 용
Real Data	500 BYTE 이하	실제 전송하고자 하는 데이터
ZDLE	1 BYTE	데이터 블록의 끝을 알리기 위해 Data End Sign 앞에 붙여줌
Data End Sign	1 BYTE	데이터 블록의 끝을 일리며, 종류에 따라서 ZACK 신호의 유무와 Frame 끝의 유무를 나타냄
Error Check	4 BYTE	CRC나 BCC 에러체크 BYTE

표 6-9. 데이터 전송블록의 종류

Value	Error Check	Next Frame	비고
"h"	○	X	Header Packet
"i"	○	○	Nonstop
"j"	○	○	ZACK Request
"k"	○	X	ZACK Request

Header에서는 앞의 3BYTE를 제외한 4번째 BYTE부터 에러체크를 계산해주며 Data Subpacket에서는 맨 앞 BYTE부터 ZDLE문자와 Data End Sign까지를 계산에 넣어 주어야 한다. 수신측에서는 Header의 경우에는 송신측과 마찬가지로 앞의 3BYTE는 에러체크를 하지 않고 그 신호가 Header인지 아닌지 에러체크방식이 어떤 것인지만을 판단해 준다. 전송되는 Header나 데이터들을 역시 BYTE 단위로 순서대로 에러체크를 해주며 마지막에 전송되는 에러체크 4BYTE도 계산을 해주어서 일정한 값이 나왔을 때 (CRC32의 경우에는 16진수 0xDEBB20E3)가 전송에러가 발생하지 않은 경우가 된다. 만약 이 일정한 값이 나오지 않았을 경우에는 데이터 전송과정에 있어서 에러가 발생되었다는 것이고, 이때는 송신측에 ZNAK 신호를 보내주어서 에러가 발생하였음을 알려주어야 한다.

Header의 시작을 알리거나 자료전송의 끝을 알리는 ZDLE 문자나 또는 시리얼 통신 자체에서 사용되어지는 특수문자들은 전송사이에 원하는 의미의 데이터가 아닌 다른 특수신호로 인식되어질 수 있기 때문에 특별한 조치를 취해 주어야 할 필요성이 있다.

송신측에서 데이터들을 전송하다가 이러한 특수문자를 만나면 그 문자의 6번 bit값을 역으로 바꾸고 ZDLE문자 다음에 전송시킨다. 수신측에서는 ZDLE 문자가 수신이 되면 다음의 1BYTE의 데이터를 입력받아서 6번 bit 값을 역으로 바꾸어 해석함으로서 전송되려는 원래의 데이터를 전송받을 수 있다. 데이터의 6번 bit값을 바꾸기 위해서는 16진수 0x40으로 exclusive OR를 취해주면 간단히 계산이 될 수 있다.

특수문자의 변환전송은 Header와 에러체크 BYTE등 전송되는 모든 문자에 해당이 되며, Header가 시작됨을 알리는 ZDLE 문자와 Data Subpacket의 끝을 알릴 때 사용되는 ZDLE 문자는 제외를 시킨다. 수신측에서는 위의 두가지 경우를 제외한 모든 전송문자에 대해서 ZDLE 문자

를 입력받으면 다음 BYTE의 문자를 입력받아서 원래값으로 환원하여 데이터를 처리한다.

특수문자의 종류와 송수신 과정을 표 6-10과 그림 6-29에 나타내었다.

라. 통신의 기본적인 순서와 구성

PC와 하드웨어 사이의 또는 하드웨어와 하드웨어 사이의 통신을 하기 위해서는 몇가지 순서를 지켜야 원활한 데이터전송을 이룰 수가 있다. 모든 신호는 송신측에서 먼저 신호를 보내면 수신측에서 그에 해당하는 응답을 하는 것을 원칙으로 하며, ZACK 신호와 ZNAK 신호는 제외이다.

통신을 하기위해서는 가장 먼저 송신측에서 수신측으로 통신선로가 열려 있는지 그리고 현재 통신을 수행할 수 있는지의 여부를 타진하기 위하여 ZRQINIT Header를 전송한다. ZRQINIT 를 전송받은 수신측에서는 성공적으로 수신이 되고 원활히 통신을 할 수 있을 경우에 ZRINIT Header로 응답을 한다. 다음은 현재의 송신측과 수신측 사이에서 시간이 맞는지를 알아보기 위하여 송신측에서 자신의 타이머가 가지고 있는 시간을 실어 수신측으로 보낸다(이 때 ZCOMMAND Header를 사용).

표 6-10. 특수문자의 종류와 16진수 코드

종 류	16진수
ZDLE	0x18
DLE	0x10, 0x90
XON	0x11, 0x91
XOFF	0x13, 0x93

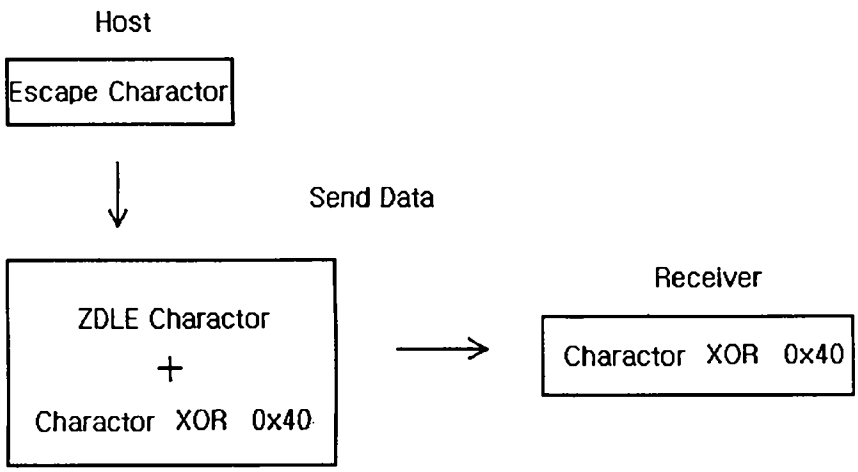


그림 6-29. 특수문자의 송수신 과정

이 신호를 받은 수신측에서는 송신측과 자신이 가지고 있는 시간이 맞는지를 검토하여 시간이 맞으면 ZACK Header를 맞지 않으면 ZNAK Header로 응답한다. 여기까지의 신호교환이 성공적으로 이루어지면 실제 데이터의 송수신이 가능하다고 판단이 되며, 실제로 데이터를 전송하고 전송받는다.

실제 데이터 전송과정에서는 환경설정과 하드웨어 설정부분과 같은 예약값 설정의 모드에서는 송신측에서 수신측으로 데이터가 전송이 되고 현재의 온실상황과 수집자료열람의 모드등에서는 수신측에서 송신측으로 데이터가 전송이 된다. 실제 데이터의 전송이 이루어질 때 그 데이터의 양과 속성을 결정해주고 받아보는 쪽에서 제대로 인식을 하기 위하여 Command Code가 요긴하게 사용이 된다.

데이터의 전송에서 데이터의 양이 많은 경우에는 적당한 크기로 나누어서 전송하는 것이 바람직하며 그는 성공적인 에러체크와 전송에러 발생시에 시간을 절약해 주기 위함이다.

통신을 성공적으로 수행한 후에 통신을 끊기 위해서 송신측에서 ZFIN Header를 보내주면 되며 그에따른 응답을 주거나 받을 필요가 없이 통신을 끊으면 된다.

그림 6-30에서 본 시스템에 사용된 통신프로그램상의 통신 과정을 대략적으로 나타내 주었다.

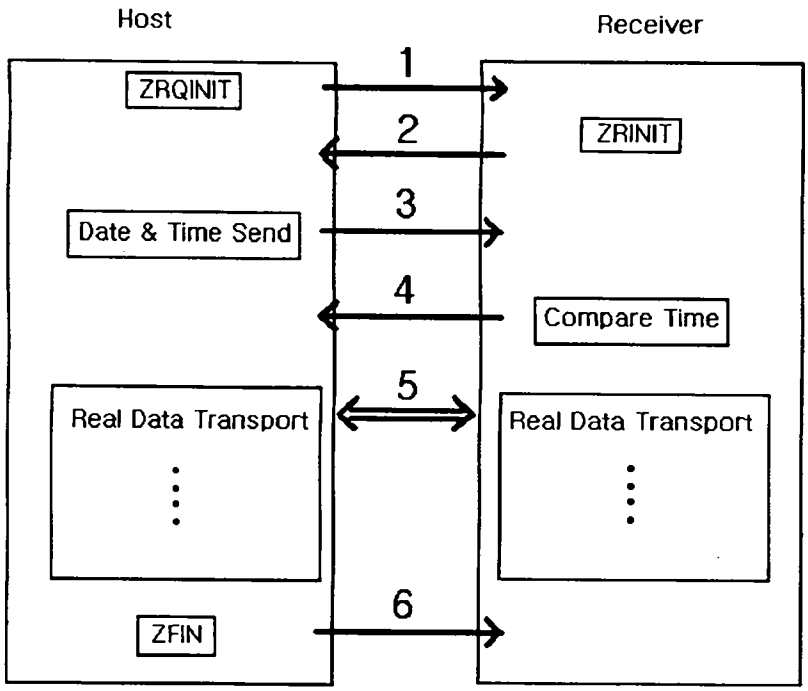


그림 6-30. 통신의 기본순서

제 5 절 복합환경 제어시스템의 하드웨어

본 시스템을 구성하는데에 가장중요한 부분중의 하나가 하드웨어의 제작이라 할 수 있다. 이번에 제작한 하드웨어는 기존에 제작되어왔던 하드웨어보다 크기가 작고 경량이기 때문에 설치 및 이동시에 비교적 편리하다 할 수 있다. 본 연구에서 제작된 하드웨어의 가장 큰 특징은 UPS기능과 탈부착이 용이하도록 제작이 된 것이다. UPS기능을 이용하여서 정전시와 같은 갑작스럽게 전원이 차단될 경우에 어느 정도까지의 비상조치로서 온실의 작동기기를 제어할 수 있게 하였다. 이를 활용함으로써 온도에 민감한 작물의 생육에 있어서 돌발사태로 인한 피해를 줄일 수 있다. 표 6-11은 본 연구에 이용된 무정전 전원장치(UPS)의 사양을 나탄낸다.

또한 하드웨어의 각 모듈을 Slot형태로 제작하여서 일부분만을 교체, 수리 및 보충을 해 줄 때 작업이 용이하며 시간을 절약시켜 줄 수가 있다.

표 6-11. UPS의 사양

항 목	사 양
엔진	10마력
발전기	5마력
출력	3상, 380V/220V
연료	휘발유
출력공차	60Hz ± 1Hz

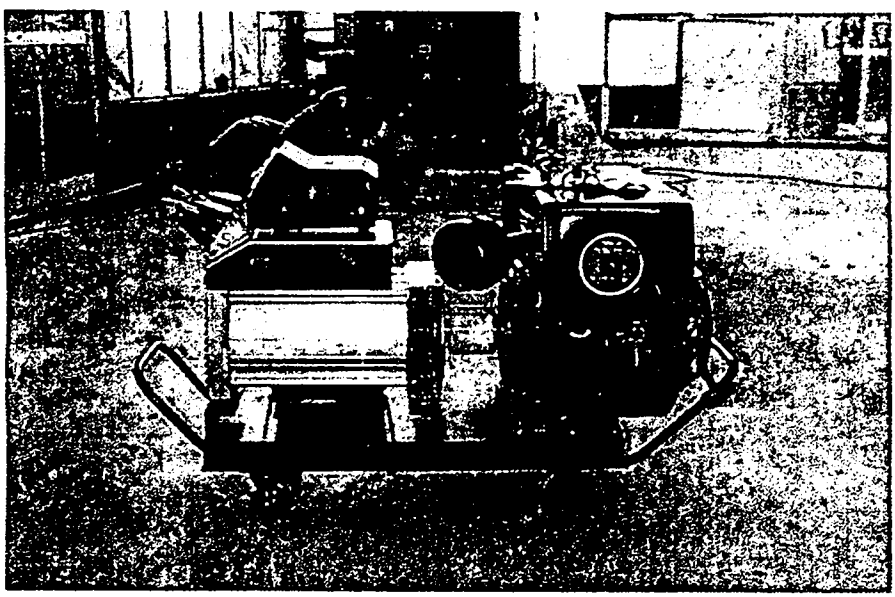


그림 6-31. 무정전전원장치(UPS)

이는 각각의 Sub 모듈뿐만 아니라 각 Controller에 있는 Main 모듈도 마찬가지로 제작이 되어 Main 모듈의 CPU에 있는 정보들의 수정이나 추가, 삭제가 매우 용이하다.

각 모듈에는 독립된 CPU를 장착하여 별도로 운용이 되며 각 모듈사이에서는 통신프로그램을 이용하여 정보들을 입출력시키기 때문에 통신프로그램의 변화가 없다면 각 모듈의 역할과 범위 등의 조절에 있어서 독립성과 확장성이 뛰어나다고 할 수 있다. 하드웨어는 Main 모듈과 그 Main 모듈에 연결되는 여러개의 Sub 모듈로 구성이 된다. Sub 모듈에는 Input 모듈, Output 모듈, Sensor 모듈, Display 모듈이 있으며, 각각의 모듈에는 통신칩이 부착이 되어서 각 모듈사이의 정보교환을 원활하게 한다.

1. 하드웨어의 특징

기존의 온실환경 제어시스템에서는 공업용 PLC를 주로 이용해왔다. 그러나 농업의 특성상 단순한 공업용 Controller으로는 제어하기가 까다로운 경우가 많이 있다. 본 시스템의 제작에서는 이러한 농업전용의 Controller가 되도록 하였다. 특히 작물을 인위적인 환경제어로 생장시키는 온실의 환경제어에서는 특히 이러한 농업전용의 Controller가 필요하다 할 수 있다.

온실 환경 제어용 하드웨어에서는 환경 및 생체정보의 정확한 계측 및 환산과 비교적 많은 양의 데이터를 수집할 수 있어야 한다. 또한 환경을 제어해 줄 수 있는 Logic에 의한 환경제어 Program을 최적으로 수행할 수 있어야 하는 능력이 있어야 하며 그에따른 실시간 제어와 디스플레이가 필수적이다. 본 시스템에서는 이러한 환경정보와 생체정보의 계측값을 입력받기 위한 입력부와 작동기기들을 동작시켜줄 신호를 내보내는 출력부를 사용 및 수리, 교환이 쉽도록 하는데 역점을 두었다. 각각의 입출력 단자

들은 그들을 포함하는 하나의 그룹단위로 구성되어 있으며 각자의 고유번지를 가지고 구동 소프트웨어에서 지정해주는 데로 입출력을 담당한다.

Main 모듈안에는 Data gathering RAM이 별도로 부착이 되어 있어서 비교적 많은 양의 Data를 저장할 수 있게 하였다. 기존의 것들에서는 하드웨어의 각 부분이 제작될때부터 고정적으로 개수와 범위등이 지정이 되기 때문에 사용하지 않는 단자들이 상당수 있다고 해도 데이터의 수집과정에서 그들의 자리를 무시할 수 없다. 따라서 본 시스템에서는 현재사용중인 부분의 데이터들만을 수집 및 저장할 수 있기 때문에 같은 용량의 gathering RAM을 가지고도 기존의 것들에 비해서 상당한 양의 데이터를 더 많이 저장시킬 수가 있다.

하드웨어의 각 부분은 모듈화가 되어있고 각각의 모듈에는 독립된 CPU가 장착되어 있으므로 독립성이 뛰어나며, 부분적인 수리 및 교환, 확장성에 있어서 노동력과 시간을 상당부분 감소시킬수 있는 이점이 있다. 또한 각 부분들이 독립적으로 구성이 되어있고 탈부착이 간편하며 독립적인 통신칩이 부착되어 있기 때문에 시설의 크기나 규모, 용도에 따라서 하드웨어의 크기와 용량이 정해지게 되어있어서 가격면에서 비교적 저렴하다.

각 모듈에 있는 원칩은 완전 독립적이어서 다른 모듈들간의 간섭을 줄일 수가 있다. 각 모듈의 위치와 순서에는 상관이 없이 제작이 되었으며 향후에 하드웨어용 프로그램의 수정 및 추가사항이 있을 때 쉽게 만들어서 원거리로 보낼 수 있는 장점이 있다.

정전시에는 Power Down을 감지하여 전원이 갑작스럽게 끊겼을 경우에 별도로 제작되는 특정제어모드(정전시 제어모드)로 전환이 되어서 반드시 필요한 작동기기만을 제한하여 동작을 시킨다. 온도에 민감한 작물의 생장에 있어서 이러한 정전 등의 비상시에 구동되는 시스템과 특정제어모드는 반드시 필요하다 할 수 있으며 정상적인 제어모드로는 불가능하다.

2. 하드웨어의 구성

하드웨어는 다동온실의 실시간 동시제어에 있어서 특별한 Main Controller가 있는 것은 아니며 여러개중의 하나가 Main Controller가 될 수 있다. 나머지 Controller들은 각 Controller에 있는 Main 모듈들 간의 통신으로 직렬연결되어 사용되어질 수 있다. 각 Controller는 하나의 Main 모듈과 그에 연결되어 있는 Sub 모듈들로 구성되어 있고 각 모듈들간의 통신은 주로 485통신을 이용하였다.

Sub 모듈의 종류는 Input 모듈, Output 모듈, Analog 모듈, Display 모듈이 있다. Input 모듈은 각종 환경계측 및 생체정보 계측용 센서에서 들어오는 신호들을 수신하게 되었으며, 교류 또는 직류 전류의 입력신호를 받을 수 있다. Output 모듈은 모터 및 각종 동작기기들을 작동시킬 수 있는 데이터를 릴레이단자를 통하여 출력시킨다.

Analog 모듈은 12bit의 해상도를 가지고 있으며 전류와 전압 모듈을 각각 제작하여 모든 신호를 수용할 수 있도록 제작하였다. Display 모듈은 각각의 온실의 입력과 출력의 모든상태를 표시하여 준다.

그림 6-32는 Main 모듈의 모식도를 나타낸다.

각각의 모듈은 완전히 독립적으로 제작을 하였으며 Slot형태로 제작을 하여서 탈부착이 용이하도록 하였다. 수리 및 교환이 필요한 모듈만을 탈착시켜서 작업을 할 수 있기 때문에 다른 모듈이나 보드들에는 전혀 영향을 주지 않는다. Main 모듈만 제외한 다른 모듈들은 부착하는 위치와 순서에 제한이 없기 때문에 확장이 필요하거나 여러모듈의 수정작업을 동시에 수행할 경우에도 큰 부담이 없다. 하나의 Controller Box에는 최고 6개까지의 모듈을 끼워넣을 수가 있으며 그 이상의 모듈이 필요한 경우에는 다른 Box를 추가시켜서 확장을 할 수가 있다.

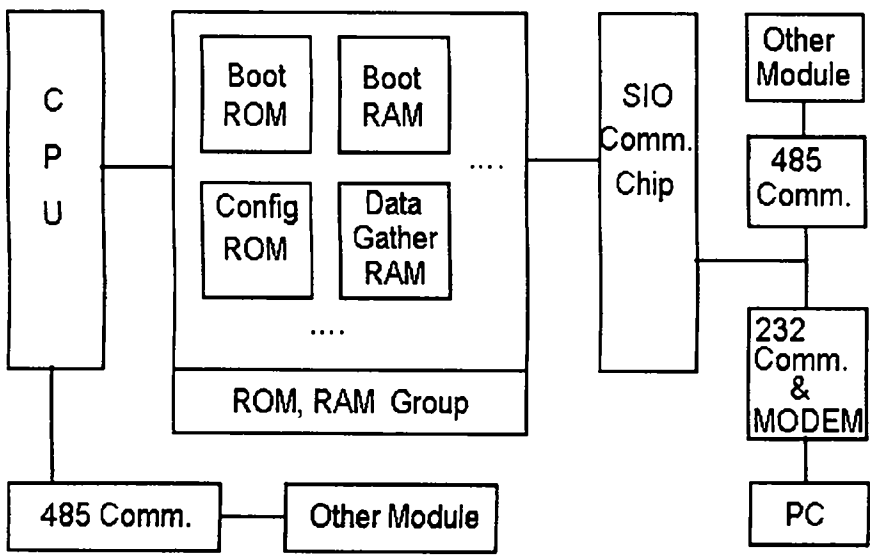


그림 6-32. Main 모듈의 기본구조

각각의 Box에는 4개의 통신포트를 만들어 놓아서 2개의 포트는 PC나 Main 모듈로의 연결을 할 수가 있고 나머지 2개의 포트를 이용하여 다른 Sub 모듈로 연결을하여 사용할 수가 있다.

그림 6-33에서 그림 6-37에 하드웨어상의 Controller Box와 각 모듈의 연결 및 slot의 형태를 나타내주고 있다.

가. 하드웨어의 개요

하드웨어의 주 기능은 각종 센서값의 입력 및 해석을 하고 동작기기들으 동작을 환경제어 Logic에 의하여 작동시키는 것이며, 전원장치 관리 시스템의 Remote 지역에서 정류기 및 UPS를 응용 관리 하는 장치이다.

(1) 블록 다이아그램

본 시스템에 사용된 CPU는 Main 모듈은 80c196, Sub 모듀은 80c96을 이용하였으며 16bit 단위로 데이터를 처리할 수 있다. 전원과 상관없이 사용이 가능한 Booting 시스템등은 EEPROM을 이용하였으며 32K Byte에서 64K의 용량을 저장해 놓을 수가 있다.

수집된 자료의 계속적인 보관을 위한 시스템은 SRAM형식을 사용하였으며 용량은 64K Byte에서 512K Byte로써 한 단위의 데이터를 처리할 수가 있다. Watch Dog 회로를 설치하여서 정전시 등의 비상사태의 유무를 항시적으로 Check, 관리 해줄 수가 있다.

또한 본 시스템에서는 RTC기능을 가지고 있어서 일반적인 시간의 Clock기능과 실시간으로 제어를 해 줄수 있는 Real Time Interrupt기능을 가지고 있다. 인터럽트 컨트롤러(Interrupt Controller)는 On-Line 방식을 채택하였으며 멀티테스킹(Multi Tasking)기능을 가지고 있다.

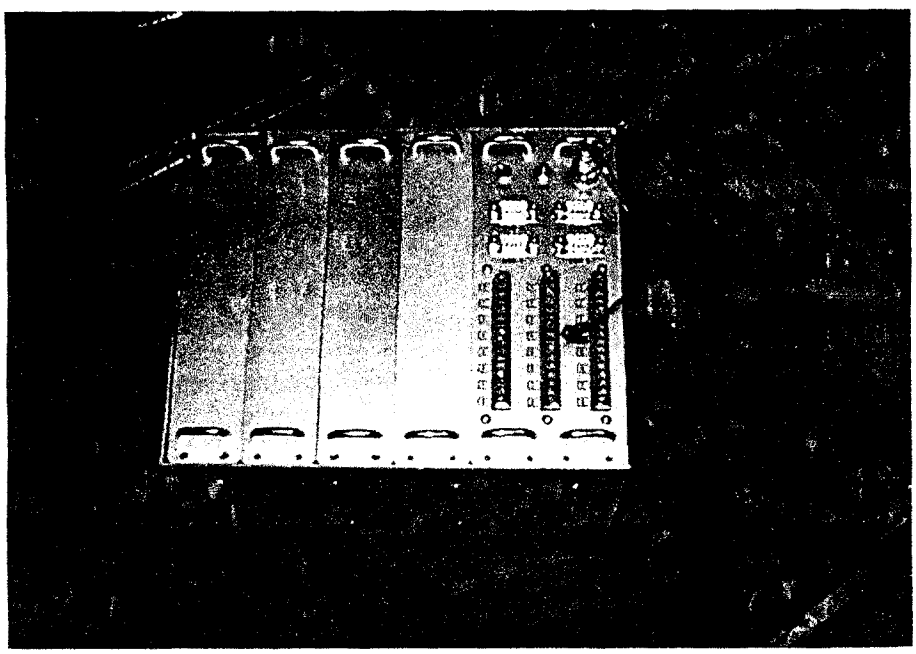


그림 6-33. Main 모듈 및 센서입력 모듈의 Box

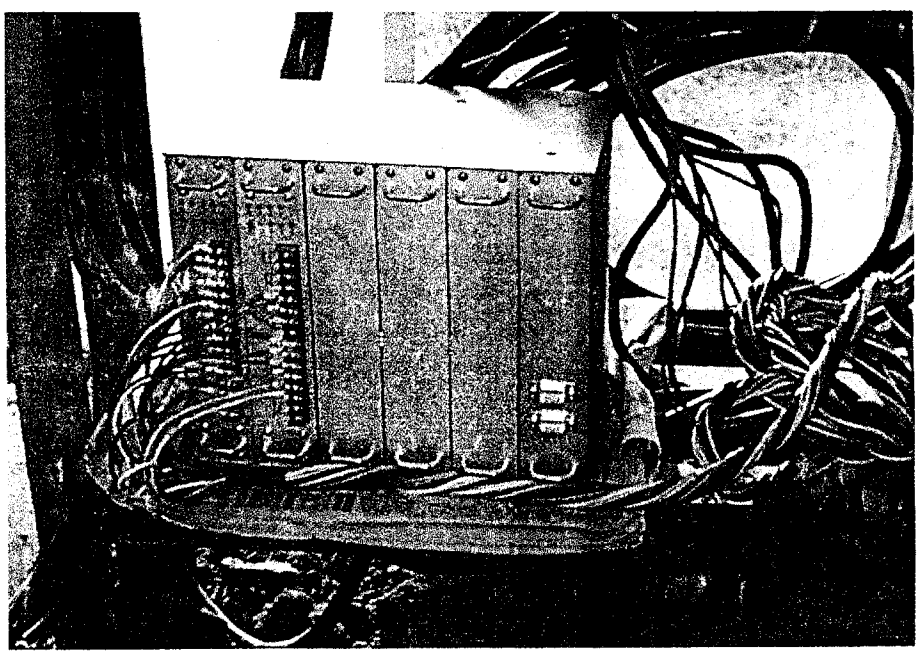


그림 6-34. 출력 및 기타 Sub 모듈

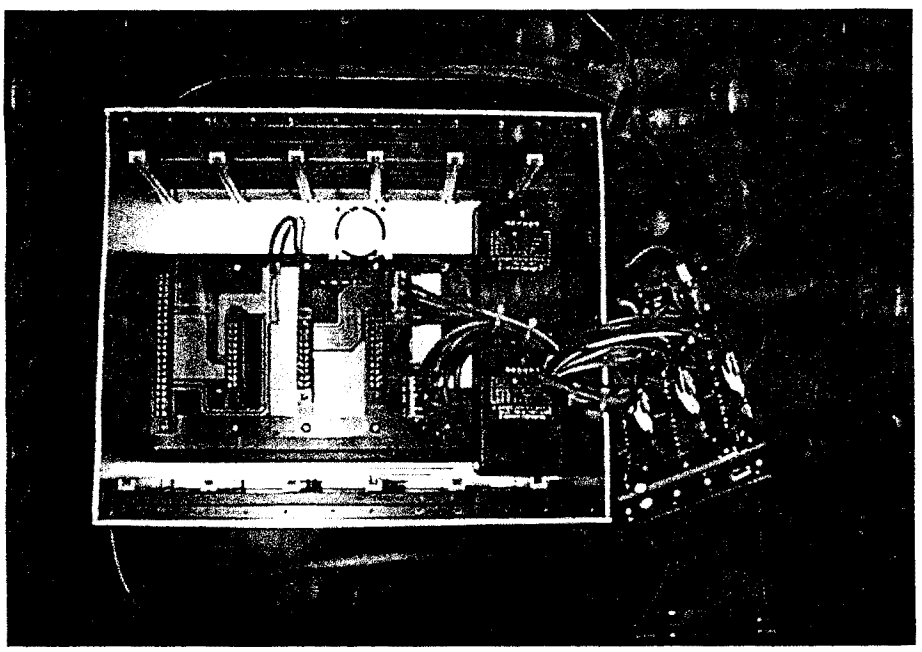


그림 6-35. 제어 Box의 내부

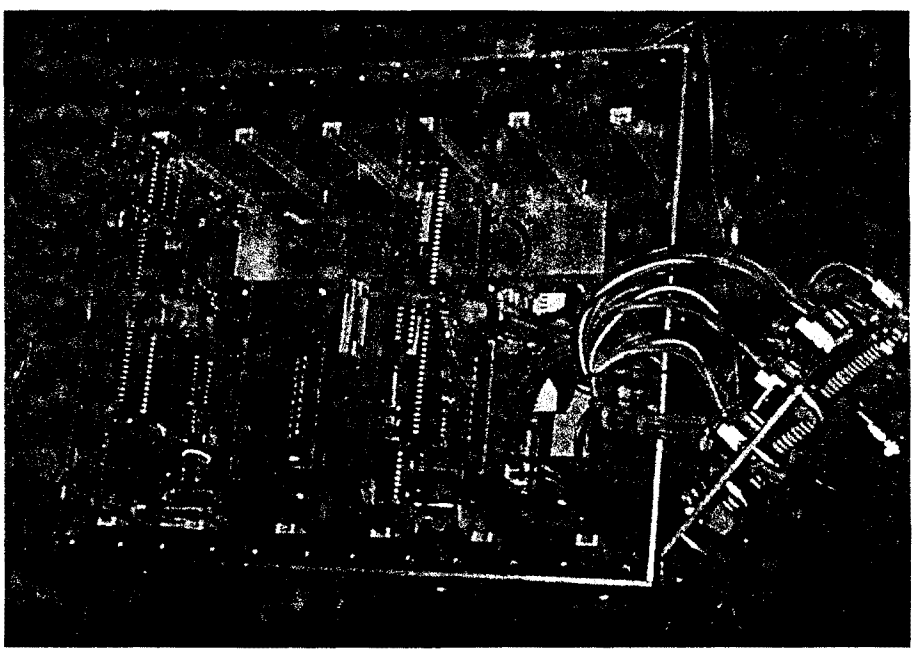


그림 6-36. 모듈을 끼워넣은 모습

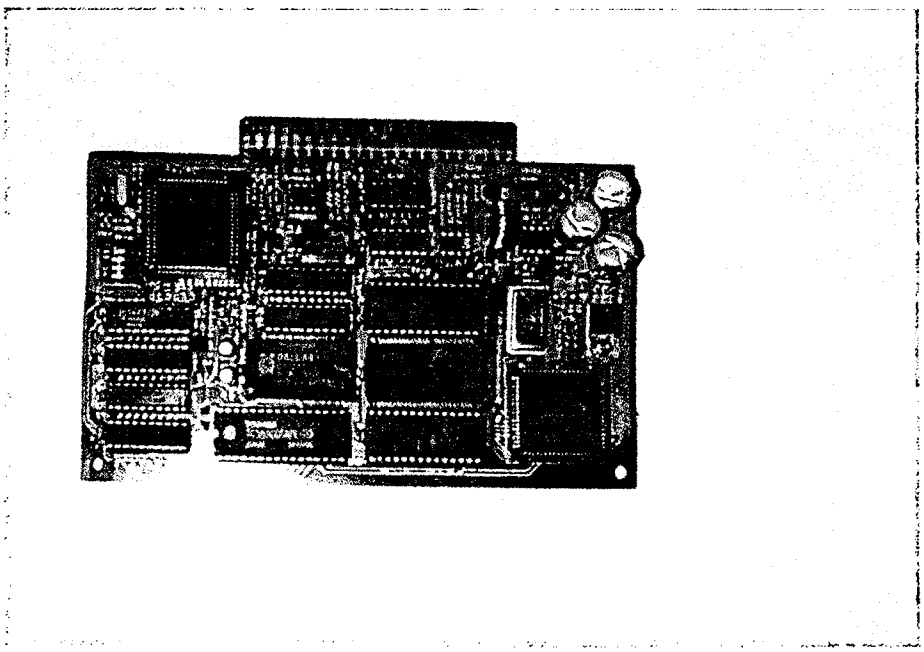


그림 6-37. Main 모듈의 모습

통신 포트로써는 2개의 SIO통신칩과 1개의 PIO 통신칩을 각 Main 모듈에 내장시켰으며 원거리의 Local회로에의 통신도 큰 장애없이 가능하도록 하였다. 전송속도는 하드웨어 끼리의 통신은 14400BPS이며 하드웨어와 HOST(PC)와의 통신의 경우에는 9600BPS를 원칙으로 하였다. 각각의 Local SIO 포트는 RS-232C와 RS-485C의 통신이 가능하게 하였으며 동시에 여러개의 Controller와의 통신을 가능하게 하기 위하여 Multi Drop기능을 가지고 있다.

하드웨어 상의 입력과 출력의 형식은 입력의 경우에 각종 센서의 값을 입력받을 수 있는 Analog Input 부분과 전원의 상태와 각 작동기기의 상태를 입력받는 Digital Input부분이 있고, 각종 동작기기의 구동을 위한 신호로써 Digital Output 부분이 있다. Analog Input은 한 Controller에 기본적으로 32 포트가 제공이 되며 12 bit A/D Converter를 이용하여 센서의 연속적인 신호를 디지털 신호로 변환하여 사용할 수 있게 하였다. Digital Input 과 Digital Output은 한 Controller에 24 포트까지를 기본적으로 사용할 수 있게 하였고 작동기기들의 동작에 있어서 Limit Check와 Relay Out 기능을 가지고 있다. 또한 각 신호들을 감지하는 Sampling Time은 1초이하로 하였으며 상황표시 LED 램프를 설치하여 하드웨어의 각 입출력 신호들의 동작을 시각적으로 표시해 주었다.

그림 6-38에 하드웨어의 기본적인 블록 다이아그램을 나타내었다.

(2) 하드웨어 각 부분들의 기능

하드웨어는 각각의 성격와 임무에 따라 모듈로 나우어지며 크게는 Main 모듈돠 그에 연결되어 있는 Sub 모듈들로 나우어질 수가 있다. 각 모듈에는 독립적인 CPU가 장착이 되어 있다. Main 모듈에는 CPU외에 Booting ROM, EEPROM 및 각종 기능을 담당하는 RAM들이 부착이 되어있다.

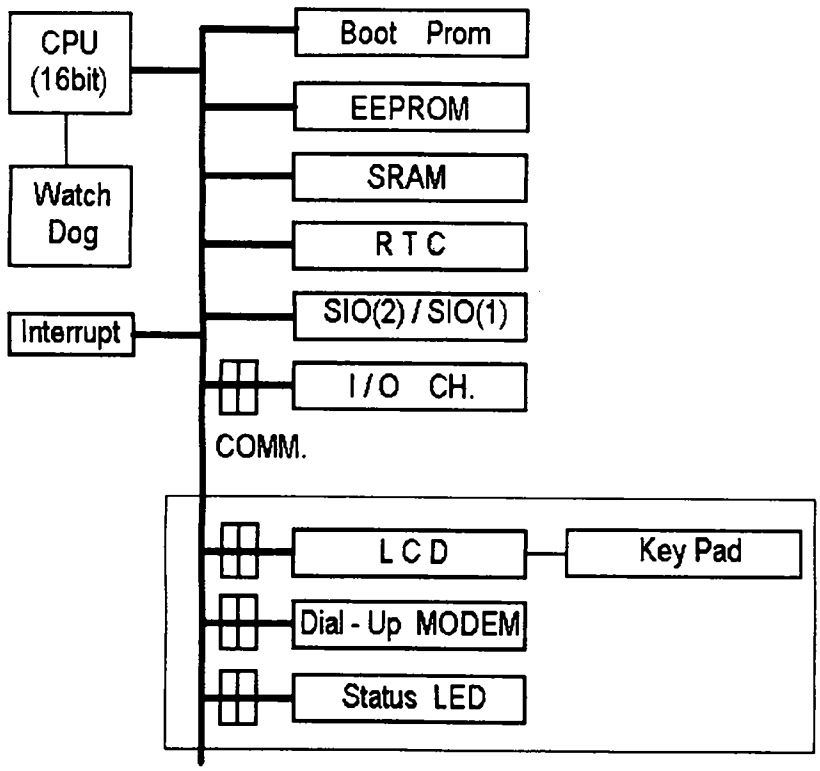


그림 6-38. 하드웨어의 블록 다이아그램

각 Sub 모듈들에는 그에 해당하는 입출력 부분들과 통신포트가 부착이 되어있다. 또한 각 Main 모듈에는 Power를 담당하는 부분이 있어서 각 Controller에 주어지는 Power를 감지하고 제어를 할 수 있도록 하였다.

(가) CPU

CPU를 Reset시키면 Watch Dog기능이 작동을 하게 되고 그에 따라서 Power Failed 및 정전, Battery Out를 감지하여 제어를 해 줄수 있게 하였다. 각 CPU들은 동일 지역에서 여러개의 하드웨어에 연결시켜서 사용할 수 있도록 하였으며 CPU는 한 단위에서 16개(CPU ID : 0 - 15) 까지를 기본적으로 사용할 수 있게 하였다.

(나) Booting ROM

각 Contrller에 탑재되어 있는 CPU의 초기화 Routine이 저장되어 있으며 자체내의 Diagnostic(Power On Self Test)기능을 가지고 있다. 또한 Booting ROM에서는 EEPROM 의 Up/Down Routine을 저장하고 있다.

(다) EEPROM

하드웨어 상에서 작동이 될 각종 프로그램 및 각종 Parameter들이 저장이 되어있다. 하드웨어와 PC, 하드웨어와 하드웨어 사이에서 데이터의 원활한 교환을 할 수 있게 해주는 각종 통신프로그램들이 내장이 되어있으며 환경제어를 위한 Logic에 해당하는 환경제어 프로그램을 내장하고 있어서 실시간으로 환경을 제어할 수 있게 하였다.

PC에서 설정을 해주어서 하드웨어로 보내어진 각종 Parameter들의 Config값들이 저장이 되어있다. 하드웨어의 각 모듈과 그 모듈에 있는 입출력단자들의 고유번지에 해당하는 기능과 성격, 사용범위, 사용개수, 사용방법 등의 데이터가 저장이 되어있고 환경제어를 위한 환경설정치의 값들이 저장되어서 환경제어를 해 줄 때 이용한다. 또한 각 하드웨어의 Define Config값들이 저장이 되어있고 제어 위치(Position)값들이 저장되어

있다.

(라) RAM

하드웨어에서 현재 사용중인 각종 프로그램들 및 입력, 출력 데이터들을 저장하여 사용한다. 환경제어 및 하드웨어의 각 부분의 정보를 담고 있는 각종 Config값들을 EEPROM으로 부터 전송받아서 사용한다. 전원이 들어왔을때 EEPROM으로부터 각종 데이터들을 전송받기 위한 Down Load Buffer가 내장되어있다. 각종 입출력 단자로부터 송수신되는 Input/Ouput 데이터들을 보통상태일 경우(normal data)와 비상상태(event data)로 구분하여 입출력한다. 또한 RAM의 운용을 위한 전원들 비축할 수 있는 Backup Battery기능을 갖고 있다.

(마) RTC

하드웨어의 시간을 유지시켜줄 수 있는 하드웨어 Clock을 제공한다. 실시간제어 및 Watch Dog기능에 필요한 Real Time Interrupt를 발생시켜서 적당한 시간에 각 제어모드를 실행 시킬수 있게 한다. 외부명령에 의해서 시간을 재설정시킬 수 있는 기능이 있어서 하드웨어의 이상이나 장기간의 무전원시에도 시간을 새롭게 설정시켜서 사용할 수 있게 하였다.

(바) Input/Output

데이터의 입력과 출력시에 On-Line Diagnostic 기능을 탑재하여 실시간의 제어중의 데이터의 입력이나 출력의 상황이 있을 경우에 그 상태를 점검할 수 있게 한다. 입출력 부분의 On-Line Diagnostic 수행은 Running Time의 10% 이하로 한다.

(사) 통신포트

원거리의 통신은 MODEM을 사용하는 것을 원칙으로하여 환경제어 프로그램이나 하드웨어 및 각종 환경설정치들의 변경 및 추가사항이 있을 경우에 원거리까지 전송을 할 수 있도록 한다. Local 통신은 RS-232C 및

RS-485 방식을 사용하며 Local Host 및 인접한 하드웨어에 연결하여 사용한다.

나. 하드웨어용 소프트웨어

각종 하드웨어의 전원 장치 관리 시스템의 제어를 해주게 되며 각 모듈의 구동을 위한 프로그램들로 구성되어지는 근거리 및 원거리 하드웨어의 운용 프로그램이다. 하드웨어의 초기화 프로그램이 선행이 되어야 하며 그 후에 각종 자료들의 수집, 제어, 가공, 저장, 전송, 관리 등의 Sub 프로그램들로 구성되어 진다.

그림 6-39에서 그림 6-45에 하드웨어용 소프트웨어의 기본구성 다이아그램과 Sub 프로그램들의 다이아그램을 나타내었다.

(1) 초기화 Routine

자체내에서 자기 진단 Routine을 수행하게 하였다. Watch Dog 및 전전시 등에 의한 비상사태에 의한 초기 동작시에는 Data RAM을 초기화 하지는 않도록 구성하였다. Data RAM의 초기화가 필요한 경우에는 외부명령에 의해서만 가능하도록 하였으며 자체내에서 임의적으로 초기화하는 것을 방지하여 사용자가 원하지 않는 데이터의 변경 및 유실을 방지하였다.

자기 진단 과정을 수행한 후에 EEPROM의 자료들이 유효할 경우에는 EEPROM의 각종 자료들을 RAM에 복사하여 실시간 제어를 수행할 수 있도록 하였다.

EEPROM의 데이터중의 일부가 또는 전체가 손상 및 유실이 되었을 경우에는 경보 표시를 하여주고 통신포트를 통해서 그 손실된 자료 및 운용 프로그램들을 Down-Load하여 다시 사용할 수 있도록 하였다. 프로그램의 복사가 성공적으로 완료가 되면 운용권을 RAM으로 넘겨서 그 이후로는 RAM에 의하여 자료수집 및 실시간 환경제어 등을 수행하도록 하였다.

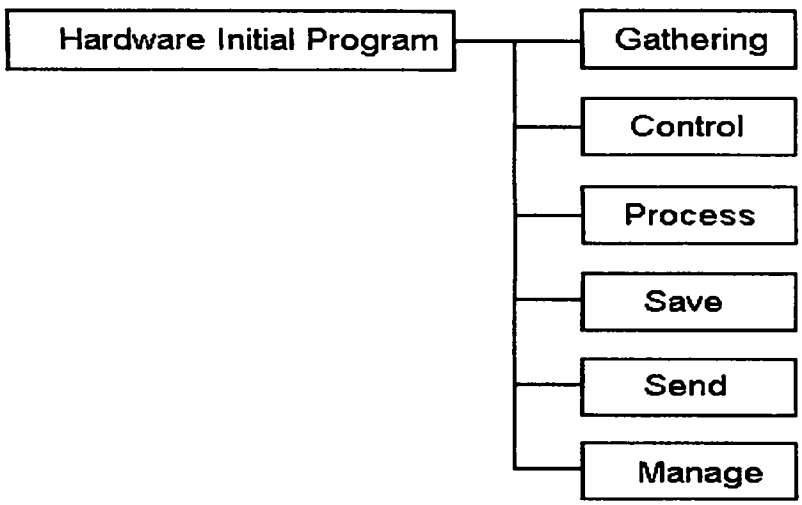


그림 6-39. 하드웨어용 소프트웨어의 블록 다이아그램

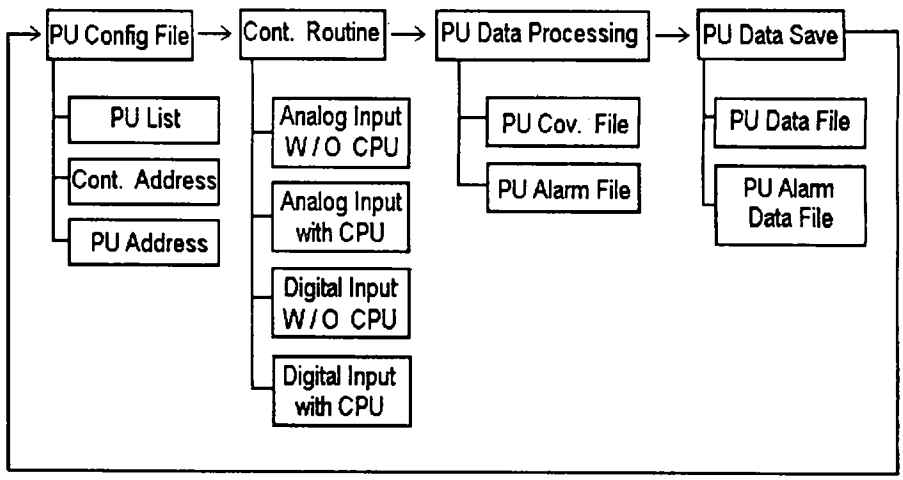


그림 6-40. 하드웨어용 소프트웨어의 수집과정

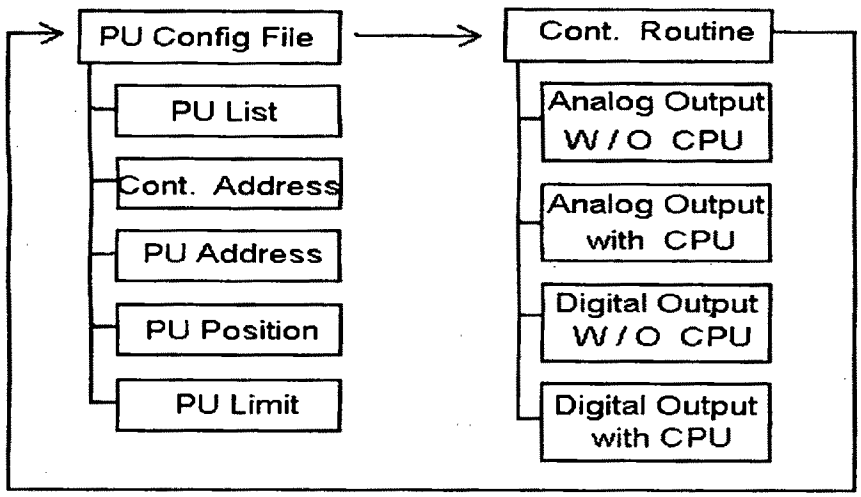


그림 6-41. 하드웨어용 소프트웨어의 제어과정

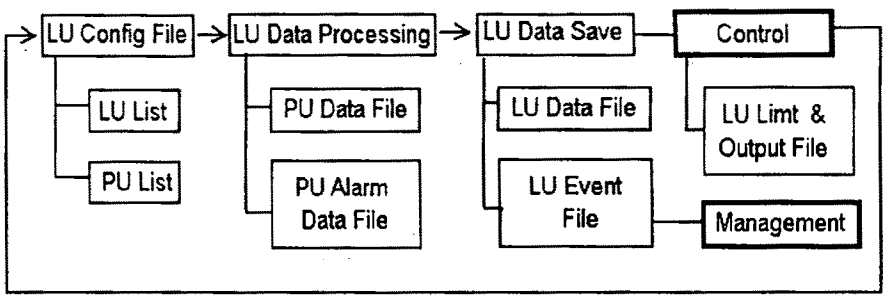


그림 6-42. 하드웨어용 소프트웨어의 가공과정

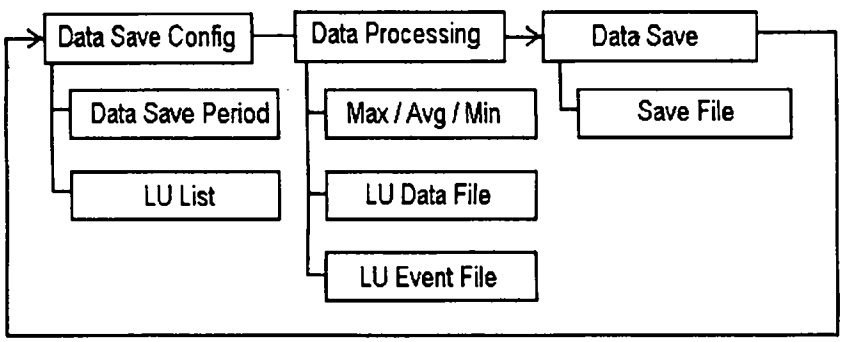


그림 6-43. 하드웨어용 소프트웨어의 저장과정

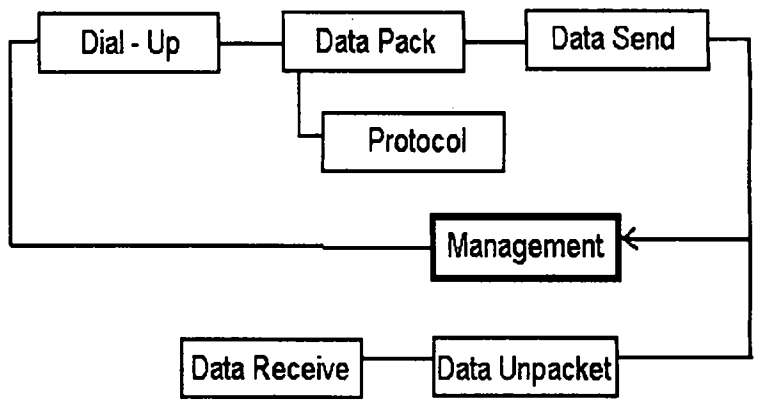


그림 6-44. 하드웨어용 소프트웨어의 전송과정

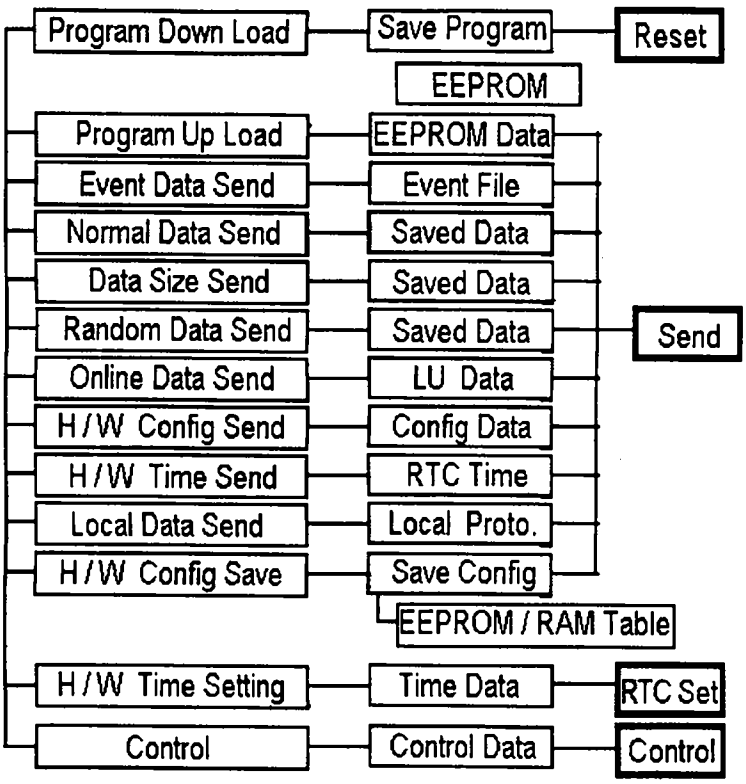


그림 6-45. 하드웨어용 소프트웨어의 관리과정

주변의 입력 및 출력의 Config값과 각종 데이터들의 초기화를 시키며 CPU, Controller 및 PU(Physical Unit)는 자동적으로 Configuration될수 있는 기능을 선택사항으로 수행하도록 하였다.

(2) PU(Physical Unit)

각종 단자들로부터 수집한 데이터들을 실제 사용할 수 있고 계산을 할 수 있는 데이터들로 변환하여서 LU(Logical Unit)로 전송시켜준다. 수집한 데이터에 이상이 발생할 경우에는 별도의 Alarm File을 만들어서 그곳에 저장을 한다. 각 Controller Type에 따른 Sub Routine을 사용하며 각 PU들의 List Table들을 운영한다. 제어포트는 현제 제어위치값을 EEPROM에 저장시켜 놓으며 PU는 다중 CPU에 할당이 가능하다.

한 단위에서 사용가능한 CPU의 개수는 16개 이며 0 에서 15 까지의 CPU ID.를 갖는다. 각 Controller는 0에서 32 까지의 번지값(Address)을 가질 수가 있으며 Controller P/N, S/N Type이다. 각 포트의 이름은 최대 16문자로 표현이가능하며 0부터 64까지의 번지값을 가질 수가 있다. 각 포트에서는 그 포트의 물리적인 해상도 범위와 센서들의 해상도 범위 그리고 사용되어지는 데이터들의 계산치의 범위와 경보시에 경보의 발생범위들을 가진다. 경보에는 지역경보와 원격경보로 나누어서 그 상황에 부합되는 경보의 정도 및 종류를 판단하여 효율적인 경보시스템의 작동을 유도한다. 각 포트의 설정값들은 PC의 시스템 구동 프로그램상의 하드웨어 설정 및 입출력단자 설정모드에서 설정해주는 기본설정치를 가지고 운용이 된다.

(3) LU(Logical Unit)

PU(Physical Unit)에서 수집한 자료들을 가공 및 제어하는 부분이다. 가공한 자료들은 Buffer에 임시로 저장을 해 두며 각종 자료들의 저장 및 On-Line 전송에이 사용한다. 가공한 자료들능 기존의 LCD 및 상태표시

LED에 표시를 하여준다. 이상이 발생된 자료들은 감시체제로 Monitoring을 해주며 즉시 Host에 전송할 수 있게 한다.

각 LU에서는 LU List Table을 운영하며 Host에서의 자료처리는 LU를 기초단위로 하여 처리한다. 각종 LU들은 다중 PU와의 연결이 가능하도록 하여서 각 하드웨어 상호간의 효율적이고 유기적인 세스템구동을 할 수 있게 하였다. LU의 감시 및 제어는 동시에 여러개의 정보들을 처리할 수 있게 다중처리방식을 체택하였으며 각 LU들은 다중 CPU에 할당되어진다. LU에서의 출력자료들은 PU 포트를 제어해줄 수 있게 한다.

LU들의 설정이름은 최대 16문자까지 가능하도록 하였으며 최대 5개씩의 입력 PU 함수들과, 출력 PU 함수들이 있다. 함수는 +, -, x, /, Avg, And, Or, Xor, Not 등의 대부분의 사칙연산과 논리연산 등으로 이루어질수가 있다.

(4) 데이터의 저장

데이터는 사용자가 PC를 통해서 설정을 해주는 데이터 저장 주기를 가지고 저장을 시킬 수 있도록 하였다. 데이터의 저장주기내의 자료들은 Max / Min / Avg 등의 값으로 가공처리되어서 저장한다. 경보상태는 데이터들의 저장주기를 가변하며, 저장 주기는 사용자의 정의에 의한다.

데이터들의 저장은 Circular Queue 방식으로 저장이 되며 Queue의 처음과 끝 및 전송 Point의 정보들을 가지고 운용시킨다. 제어명령에 의하여 임의의 범위에 있는 데이터들을 전송 할 수 있돌고 하였다. 저장된 데이터들은 Local Serial 포트를 이용하여서 Down-Load 기능을 수행할 수가 있다.

(5) 전송

데이터들의 전송은 Remote 전송과 Local 전송으로 구분하여 전송함으로써 시간과 자료전달의 정확성에서 최소한의 오류로 데이터들을 전송시켜줄

수 있도록 하였다.

Remote 전송의 경우에는 Point to Point Interface 방식으로 데이터들을 전송시켜주며 이때는 MODEM을 사용하여 데이터를 전송시킨다. MODEM의 양식은 ZMODEM Protocol을 기본으로 하여 제작하였으며 데이터의 전송에러는 CRC 32 Error Correction을 이용하여 검사한다.

Local 전송의 경우에는 Multi Drop 을 사용하였으며, RS-232C 및 RS-422를 사용할 수가 있다. Local Host 및 인접한 하드웨어와의 Interface를 담당한다.

(6) 다중 하드웨어 방식

Master CPU는 CPU ID. 0번을 기본으로 하며 이는 다동온실의 환경제어에 있어서 온실의 위치 및 종류와 환경에 따라서 가변적일 수가 있다. Master CPU(기본적으로는 0번)를 제외한 나머지 15개의 CPU들이 Local Host가 되어 시스템을 제어하게 된다. 다중 하드웨어의 사용할 때에는 CPU들간의 주기적인 통신을 이용하여서 각 하드웨어의 Down 상태를 항상 감시 보고하도록 되어있다. 통신 Time Out가 발생될 때에는 해당하는 CPU가 Down된 것으로 간주하며 CPU간의 통신의 주도는 Master CPU가 담당하게 된다.

만약 일정기간 동안이상 통신이 이루어지지 않을 경우에는 Master CPU를 다음의 CPU ID.를 갖는 하드웨어로 전환하여 시스템을 구동시킨다.

하드웨어 프로그램의 설정은 Local/Remote Down-Load 및 Key-Pad에 의해서 설정이 된다. Config값의 Down-Load시에는 전체의 하드웨어에 동일한 데이터로써 저장을 한다.

제 6 절 결 언

본 연구에서 제작되어진 하드웨어와 소프트웨어는 기존의 시스템들에 비하여 크기가 작고 개수 및 사용범위, 용도등이 가변적이고 유동적이어서 제어할 온실 및 기타 유사 시설물의 크기와 종류, 개수, 특징에 따라서 시스템을 쉽게 사용할 수 있도록 하였다. 본 시스템은 크기가 작고 사용되는 환경에 따라서 Controller의 개수가 조절이 되기 때문에 가격면에서도 상당히 합리적이고 경제적일 것으로 사료된다.

하드웨어의 각 입출력부분 및 구동 소프트웨어가 MODEM 및 RS-232, RS-485 통신을 통하여 PC에서부터 전송시켜 줄 수 있기 때문에 사용자가 원할 경우에는 시스템의 규모를 축소 또는 확장시킬수가 있다. 또한, 프로그램 및 입출력단자들의 정보를 추가, 삭제 및 수정을 쉽게 할 수가 있어서 경제적, 시간적 측면에서 상당히 효율적이라 할 수가 있다.

시스템 구동 소프트웨어의 제작시에 하드웨어의 각 부분들의 정보를 정밀하게 설정해줄 필요가 있기 때문에 구동 소프트웨어의 작성시에 상당한 노력과 정밀성이 요구가 된다. 그러나 본 시스템을 이용한 환경제어시에 상당한 확장성과 유동성을 가지고 있기 때문에 정밀한 구동 소프트웨어가 한 번 작성이 되면 시스템을 개수와 용도에 그리 큰 제약을 받지않고 운용을 시킬 수가 있을 것으로 사료된다.

하드웨어는 각각의 기능별로 완전한 독립구조의 모듈형태로 제작이 되었기 때문에 어느 제한된 부분의 고장수리나 추가되는 정보를 입력시킬 필요가 있을 때에는 비교적 간단하게 작업을 수행할 수가 있다. 물리적인 측면에서 보면 하드웨어의 각 입출력부분 및 Main 모듈들을 탈부착이 간단한 Slot 형태로 제작하였기 때문에 수리 및 수정과정에서 몇 개의 나사만 풀어주면 쉽게 그 부분을 빼내어 점검을 할 수가 있어서 시간적으로도 큰

이득이 되며 이동성에 있어서도 큰 장점이 있다고 할 수 있다.

입출력단자의 제어측면에서는 각 입출력단자들의 기능 및 개수, 범위, 사용방법 등을 구동 소프트웨어의 하드웨어설정 및 하드웨어 입출력단자 설정모드에서 지정해 주는 데로 입력신호와 출력신호들을 처리하고 제어해 줄수가 있기 때문에 상황에 따른 기능의 변동 및 추가가 상당히 용이하다. 입력과 출력신호들의 처리를 위한 함수도 소프트웨어적으로 설정을 해 줄 수가 있기 때문에 하드웨어의 기능변동 및 기능추가시에 특별한 사항이 없을때에는 하드웨어를 변화시키지 않고도 PC를 통해서 하드웨어의 원하는 부분의 설정치만을 변경 및 추가해 줌으로써 해결이 가능하다.

하드웨어는 각 온실별 또는 하나의 제어단위별로 작은 Controller Box로 구성이 되어있고 각각의 Controller Box들에는 통신포트가 부착되어 있어서 하드웨어의 추가가 필요한 경우에는 통신포트와 연결선로를 이용하여 쉽게 확장시킬수가 있다. 하드웨어 적으로는 같은 구동제어 Logic을 갖는 독립적인 CPU가 각 모듈별로 내장되어 있어서 이러한 하드웨어의 확장 및 축소시에 다른 하드웨어에는 전혀 영향을 미치지 않는다.

본 시스템에 사용되고 있는 환경제어 Logic은 기존의 시스템에서 사용되었던 타이머를 이용한 작동기기별로의 제어가 아닌 온실전체의 온도와 습도를 위주로 하여 시퀀스 제어를 하도록 구성하였다. 이때에 한 번에 하나의 작동기기 제어로는 그 상황에서 원하는 만큼의 제어가 이루어지지 않을 경우가 있기 때문에 경우에 따라서는 2개 또는 그 이상의 작동기기가 동시에 제어될수 있도록 하는 제어 Logic이 더 필요할 것으로 사료된다. 또한 작물의 생장에 좀더 많은 영향을 미치는 작동기기와 환경요인을 찾아내어서 가능한 작물이 원하는 방향으로의 환경제어가 있어야 될 것으로 사료되며 생체정보와 환경정보를 동시에 계측하고 그를 이용한 환경제어를 해주는 Logic이 개발되어야 한다.

여 백

제 7 장
결론 및 요약

국내에서 많은 회사가 온실용 환경제어시스템을 개발하고 있지만, 복합환경 제어시스템이 아닌 온실 작동기의 ON/OFF 제어만 하고 있는 실정이다. 특히, 중소기업이 많은 데이터를 분석한 후 최적화를 위한 프로그램을 개발하기 위하여 인적 자원과 경제적 자본을 투자하기에는 매우 어렵다.

따라서, 본 연구는 복합환경을 제어하기 위한 하드웨어를 개발하기 위하여 수행하였으며, 온실의 복합환경을 제어하기 위하여 실내·외 환경요인을 측정하기 위한 센서와 작물의 생체정보를 수집하기 위한 센서를 설치하였다.

연구결과를 요약하면 다음과 같다.

1. 국내에 설치된 온실의 방문 및 자료수집에 의하여 온실의 현황 및 기술 수준을 분석하였으며, 외국의 온실 현황은 자료를 수집하여 분석하였다.
2. 농촌진흥청 원예연구소에 20평 규모의 실험온실 2동을 확보 (FRP 온실, PET 온실)하였다. 온실 재배용 오이와 토마토의 정식을 위하여 각 온실에 폭 0.5m, 두께 0.025m, 길이 1.2m의 폴리스티렌 베드 7장을 연결하여 폭 0.5m, 깊이 0.2m, 길이 8.4m의 성형 베드를 각 온실마다 3개씩 설치하였다.
3. 각 온실에는 환경요인의 제어를 위하여 측창 개폐장치, 천창 개폐장치, 커튼 개폐장치, 환기장치 등의 온실작동기와 온실작동기의 수동조작 및 자동제어를

위한 동력제어반 설치하였다. 각 온실에 양액을 자동 공급하기 위하여 양액 공급시스템을 설치하였다.

4. 온실의 실내·외 환경요인은 일사량 센서, 실내 온·습도 센서, 외부 온·습도 센서, 풍향센서 , 풍속센서, 강우량 센서를 설치하여 측정하였다. 작물의 생육정보를 측정하기 위하여 줄기직경(stem diameter) 센서, 증산류(stem water flux) 센서, 엽온(leaf temperature) 센서, 과일직경(fruit diameter) 센서와 양액제어를 위하여 pH 센서, EC센서를 설치하였다. 측정된 환경요인과 작물의 생육정보를 디지탈화하기 위하여 중간변환장치를 설치하였으며, 온실 작동기의 제어를 위하여 콘트롤러를 설치하였다.

5. 생체정보 이용 온실 환경관리를 위한 자료분석에서 증산류를 지표로 한 모델은 특히, 작물의 수분관리에 유용할 것으로 판단되었다.

6. 과경과 경경을 지표로 한 모델의 구성을 위해서는 보다 감도가 높은 센서의 이용이 필요했으며, 이들 두 지표의 지연적인 생장과 환경요인들과의 변화 반응시간을 일치시키면 모델 구성이 가능할 것으로 판단되었다.

7. 보다 완벽한 작물 생장모델을 구축하기 위해서는 환경조절이 가능한 그로스챔버를 이용한 실험이 요구되었다.

8. 복합환경의 제어와 하드웨어간의 통신을 위하여 컴퓨터 프로그램을 작성하였다. 프로그램 작성에는 Windows용 Visual Basic, Turbo-C, Assembler어를 사용하였다. 복합환경 제어용 프로그램에는 온실 내·외부 환경요인 및 작물 생육정보 수집부, 환경요인 및 생육정보에 의한 작동기 제어부로 구성하였다.

9. 프로그램 중, 커튼의 제어 Logic은 가열이나 감온시에 작동되는 것 이외에 별도로 작물의 광합성과 일사량과의 밀접한 관계를 고려하여 설계하였다.

10. 본 연구에서 개발한 복합환경 제어시스템의 하드웨어와 소프트웨어

는 기존의 시스템에 비하여 크기가 작고 갯수 및 사용범위, 용도등이 가변적이고 유동적이어서 제어할 온실 및 기타 유사 시설물의 크기와 종류, 개수, 특징에 따라서 시스템을 쉽게 변경하여 사용할 수 있도록 하였다. 또한, 본 시스템은 크기가 작고 사용되는 환경에 따라서 Controller의 개수가 조절이 되기 때문에 가격면에서도 상당히 합리적이고 경제적일 것으로 사료된다.

11. 하드웨어의 각 입출력부분 및 구동 소프트웨어가 MODEM 및 RS-232, RS-485 통신을 통하여 PC에서부터 전송시켜 줄 수 있기 때문에 사용자가 원할 경우에는 시스템의 규모를 축소 또는 확장시킬 수 있도록 하였다. 또한, 프로그램 및 입출력단자들의 정보를 추가, 삭제 및 수정을 쉽게 할 수가 있도록 구성하였다.
12. 본 연구에서 개발된 복합환경 제어시스템에는 무정전 전원장치를 포함시켜 정전 등에 의하여 전원이 차단될 경우에도 온실의 작동기기를 제어할 수 있도록 하였다.

참 고 문 헌

1. 권영삼. 1995. 국내 원예시설의 유형별 특성 및 발전방향. 한국시설원예연구회 국내외 원예시설의 특성과 시스템에 관한 심포지엄. pp. 109 - 125.

2. 권영삼. 1996. 온실환경조절의 현황과 전망. 영남대 농업기계화 학술세미나.

3.권지선. 1995. 생체정보 계측에의한 온실환경관리. 시설원예연구 8(1). pp. 127 - 130

4. 김영복외. 1992. 환경변화에 대한 실험적 분석(Ⅰ). 농촌열에너지 연구보고 논문집.

5. 김효중. 1994. 완전제어형 작물생육 장치를 위한 인공광원장치부 개발. 서울대학교 석사학위논문.

6. 나우현. 1985. 비닐하우스 채소재배. 오성 출판사

7. 남상운, 김문기, 손정익. 1993. 수경온실의 양액 냉각부하 예측모델 개발. 생물생산시설환경. 2(2) : 99-109.

8. 민영봉, 문성동, 김진영. 1994. 온실의 자동화 설비 구조특성. 『공정육묘 온실의 자동화시스템 개발』에 관한 심포지움 발표문. 경상대학교 시설원예연구소. p55-132.

9. 민영봉, 박중춘, 이상옥, 정태상. 1994. 온실의 복합환경제어시스템 구성. 『공정육묘 온실의 자동화시스템 개발』에 관한 심포지움 발표문. 경상대학교 시설원예연구소. p132-151.

10. 박재복. 1987. 플라스틱 온실의 일사량 분석과 열적 환경의 시뮬레이션에 관한 연구. 서울대학교 박사학위논문.

11. 서원명, 민영봉. 1990. Microcomputer를 이용한 Greenhouse의 온도 제어 System개발에 관한 연구. 한국농업기계학회지 12 : 16-27.

12. 서원명. 1994. 온실의 환경인자 및 환경조절에 관한 기본 지침. 『공정육묘 온실의 자동화시스템 개발』에 관한 심포지움 발표문. 경상대학교 시설원예연구소. p31-54.

13. 손정익, 이동근, 김문기. 1993. 식물생산 시스템의 다목적 환경예측 모델의 개발. 생물생산시설환경. 2(2) : 126-135.

14. 송현갑, 금동혁, 류관희, 이기명, 이종호, 정두호. 1993. 시설원예 자동화. 문운당.

15. 송현갑, 유영선. 1996. 그린하우스 열환경 조절을 위한 파란핀계 화합물(C_nH_{2n+2})의 잠열 축열 특성. 한국농업기계학회지. 21(1) : 84-93.

16. 이규철, 류관희, 노상하, 홍순호. 1992. 완전제어형 실험용 작물생육 장치의 개발(I), 온습도 제어 시스템. 한국농업기계학회지. 17(1) : 55-64.

17. 이석건. 1992. 농업환경조절공학. 교보문고.

18. 진제용, 류관희, 홍순호. 1993. 작물의 생장계측 및 생육제어에 관한 연구(I), 탄산가스 제어 알고리즘 개발. 생물생산시설환경. 2(1) : 27-36.

19. 한국농자재산업협의회. 1992. 『'92 시설원예 기술전 자료집』.

20. 한국원예시설연구회 1995. 『국내외 원예시설의 특성과 시스템에 관한 심포지움』 발표문.

21. 홍순호. 1990. 작물 생산 자동화를 위한 복합환경계측 시스템 개발. 서울대학교 석사학위논문.

22. 김진현, 김철수, 구건효, 이기명. 1995. 마이크로 컴퓨터에 의한 시설재배의 자동화에 관한 기초 연구(Ⅲ), 양지붕형 하우스의 창 개방 방법에 따른 온·습도의 변화. 한국농업기계학회지. 20(2) : 162-172.

23. 이용범. 1995. 시설재배의 주요 환경관리. 새농사. 13 : 38-40.

24. 장연옥. 1989. 남부지방 시설원예의 재배형태 및 특성과 미기상 환경에 관한 조사 연구.

25. 한국과학재단. 1994. 작물의 생장 정보 계측 및 생육 제어에 관한 연구.

26. Avissar, R. and Y. Mahrer. 1982. Verification study of a numerical greenhouse microclimate model. Transactions of the ASAE. 25(6) : 1711-1720.

27. Bailey, B.J. 1984. Limiting the relative humidity in insulated greenhouse at night. Acta Hort. 148 : 411-419.

28. Challa, H. and G. van Straten. 1991. Reflections about Optimal Climate Control in Greenhouse Cultivation. IFAC Mathematical and Control Applications in Agriculture and Hort.. Matsuyama, Japan.

29. Chandra, P. and L.D. Albright. 1980. Analytical determination of the effect on greenhouse heating requirement of using night curtains. Transactions of the ASAE. 23(4) : 994-1000.

30. Garzoli, K.V. and J. Blackwell. 1973. The response of a glasshouse to high solar radiation and ambient temperature. Journal of Agricultural Engineering Research 18 : 205-216.

31. Hartz, T.K., A. Baameur and D.B. Holt. 1991. Carbon Dioxide Enrichment of High-value Crops under Tunnel Culture. J. Amer. Soc. Hort. Sci. 116(6) : 970-973.

32. Hwang, Y.K. 1993. Optimization of Greenhouse Temperature and Carbon Dioxide in Subtropical Climate. Ph.D. thesis. University of Florida.

33. Jacobson, B.K., J.W. Jones and P.H. Jones. 1987. Tomato greenhouse environment controller : Real-time expert systems supervisor. ASAE. paper No. 87-5022.

34. Jones, J. W., E. Dayan, L. H. Allen, H. van keulen, and H. Challa. 1991. A Dynamic tomato growth and yield model(TOMGRO). ASAE 34(2). pp. 663 - 672

35. Jones, P., B.L. Roy and J.W. Jones. 1989. Coupling expert systems and models for the real-time control of plant environment. Acta Hort. 248 : 445-452.

36. Jones. P., J.W. Jones, L.H. Allen, Jr., and J.W. Mishoe. 1984. Dynamic computer control of closed environmental plant growth chambers. Design and verification. Transactions of the ASAE. 27(3) : 879-888.

37. Kano, A. and C.H.M. van Bavel. 1988. Design and test of a simulation model of tomato growth and yield in a greenhouse.

Journal of Japan Society Horticultural Science 56 : 408-416.

38. Marsh, L.S. and L.D. Albright. 1991. Economically optimum day temperatures for greenhouse hydroponic lettuce production, Part I : A computer model. Transactions of the ASAE. 34(2) : 550-556.

39. Marsh, L.S. and L.D. Albright. 1991. Economically optimum day temperatures for greenhouse hydroponic lettuce production, Part II : Result and simulation. Transactions of the ASAE. 34(2) : 557-562.

40. Mitchell, B.W. 1986. Integrated microcomputer-based multiple environmental cabinets. Proceeding of the Agri-Mation 2, ASAE.

41. Nelson, P.V. 1991. Greenhouse Operation and Management. Prentice Hall. Englewood Cliffs.

42. Parsons. J.E., J.L. Dunlap. J.M. Mckinion, C.J Phence and D.N. Baker. 1980. Microcomputer-Based Data Acquisition and Control Software for Plant Growth Chamber(SPAR System). Transactions of the ASAE. 23(3) : 589-595.

43. Saffell. R.A. 1985. The control of temperature, humidity and carbon dioxide concentration n experimental glasshouse. Acta Hort. 174 : 443-447.

44. Seginer, I. and A. Sher. 1992. Neural-nets for greenhouse climate control. ASAE. Technical Paper No. 92-7013.

45. Takakura, T., K.A. Jordan, and L.L. Boyd. 1971. Dynamic simulation of plant growth and environment in the greenhouse. Transactions of the ASAE. 14(5) : 964-971.

46. Takakura, T., T. Kozai, K. Tachibana and K.A. Jordan. 1974.

Direct digital control of plant growth - I. Design and operation of the system. Transactions of the ASAE. 17(6) : 1150-1154.

47. Takakura, T. 1989. Micro-computer use for energy and labor savings in Japaese protected cultivation. Acta Hort., 257. pp. 79 - 85

48. Takakura. T. 1991. Greenhouse production in engineering aspects. IFAC Mathematical and control applications in agriculture and horticulture. pp. 19 - 21

49. Tantau, H.J. 1985. Greenhouse climate control using mathematical models. Acta Hort. 174 : 449-459.

50. Ton Y., and N. Nilov, 1996. Phytomonitoring technology in control of crop growth. International symposium on plant production in closed ecosystems. p. 77

51. Udink ten Cate, A.J. and H. Challa. 1984. On Optimal Computer Control of the Crop Growth System. Acta Hort. 148 : 267-276.

52. Walker, J.N. 1965. Predicting temperatures in ventilated greenhouses. Transactions of the ASAE. 8(3) : 445-448.

53. 高辻正基. 1987. 植物工場入門. オーム社. 東京.

54. 高辻正基. 1992. 限界效用の遞減則と植物工場の最適化原理. 植物工場學會誌. 4(1) : 6-9.

55. 橋本 康. 1987. 植物環境制御入門. オーム社. 東京.

56. 安井秀夫. 1990. 施設栽培學. 川島書店, p21-30.

57. 位田勝久太郎. 1977. 施設園藝の環境と栽培. 誠文堂新光社, p2-10.

부 록

(Appendix)

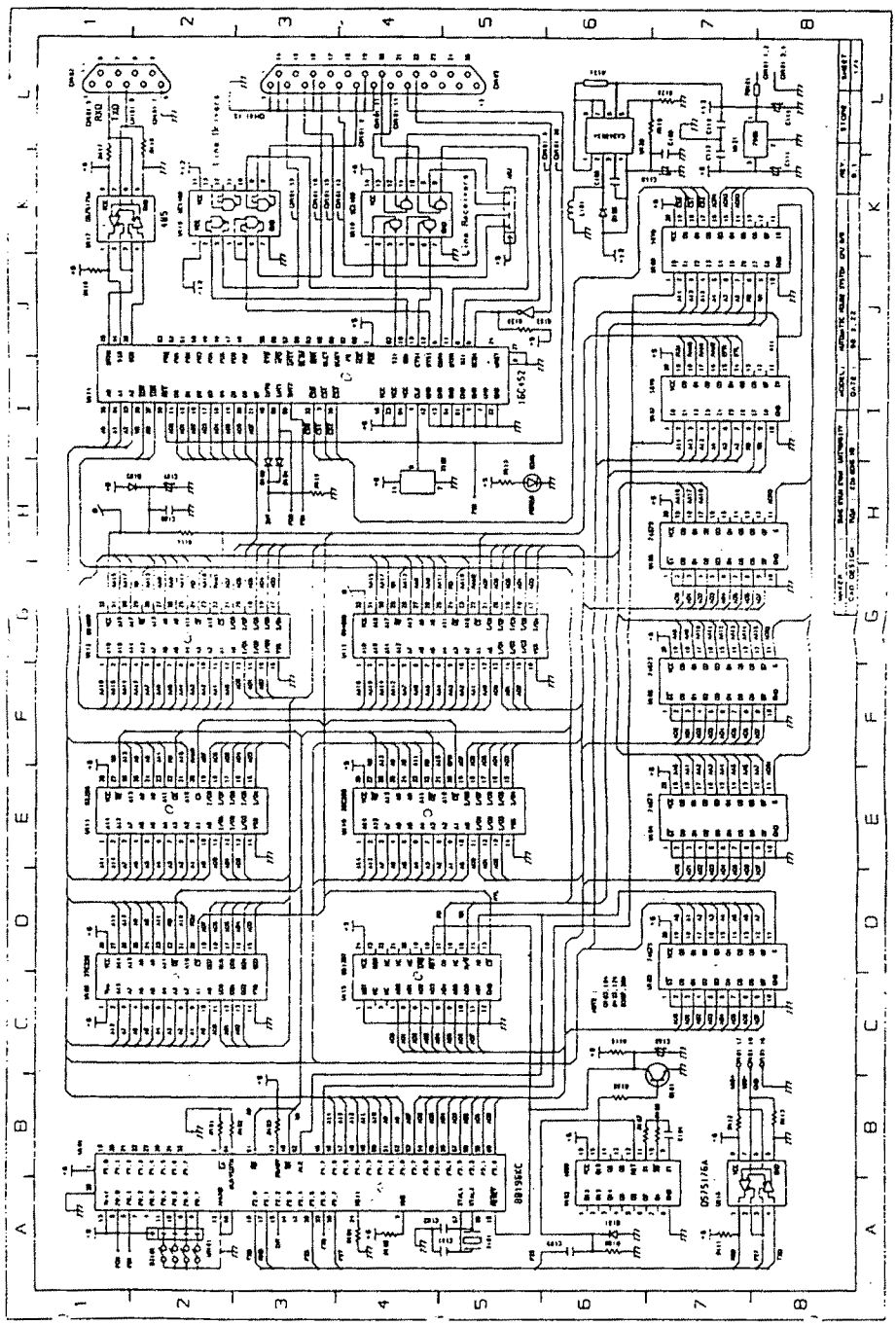
A. Hardware Circuit

B. PC용 프로그램

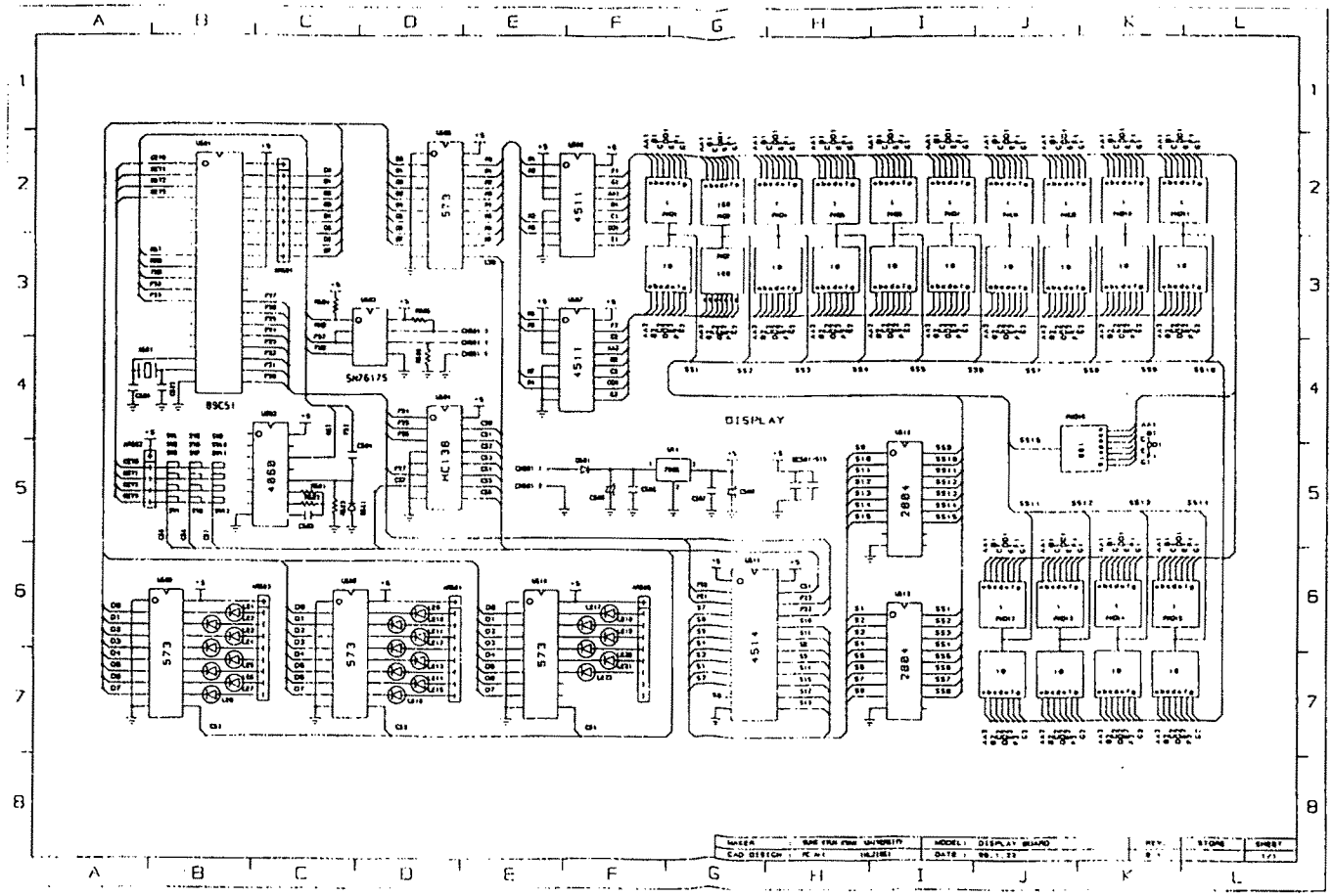
C. 하드웨어용 프로그램

D. CRC32 Table

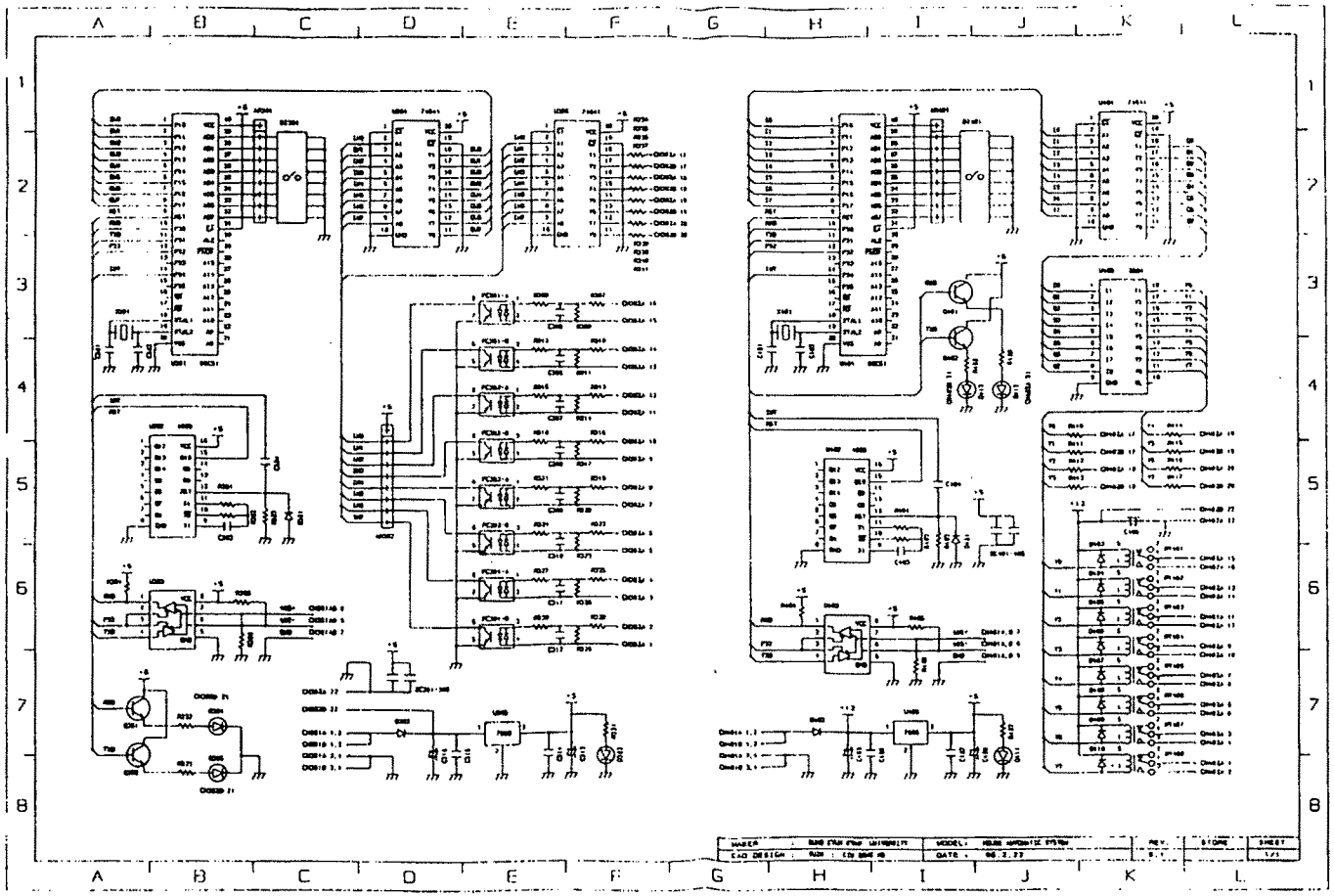
A. Hardware Circuit



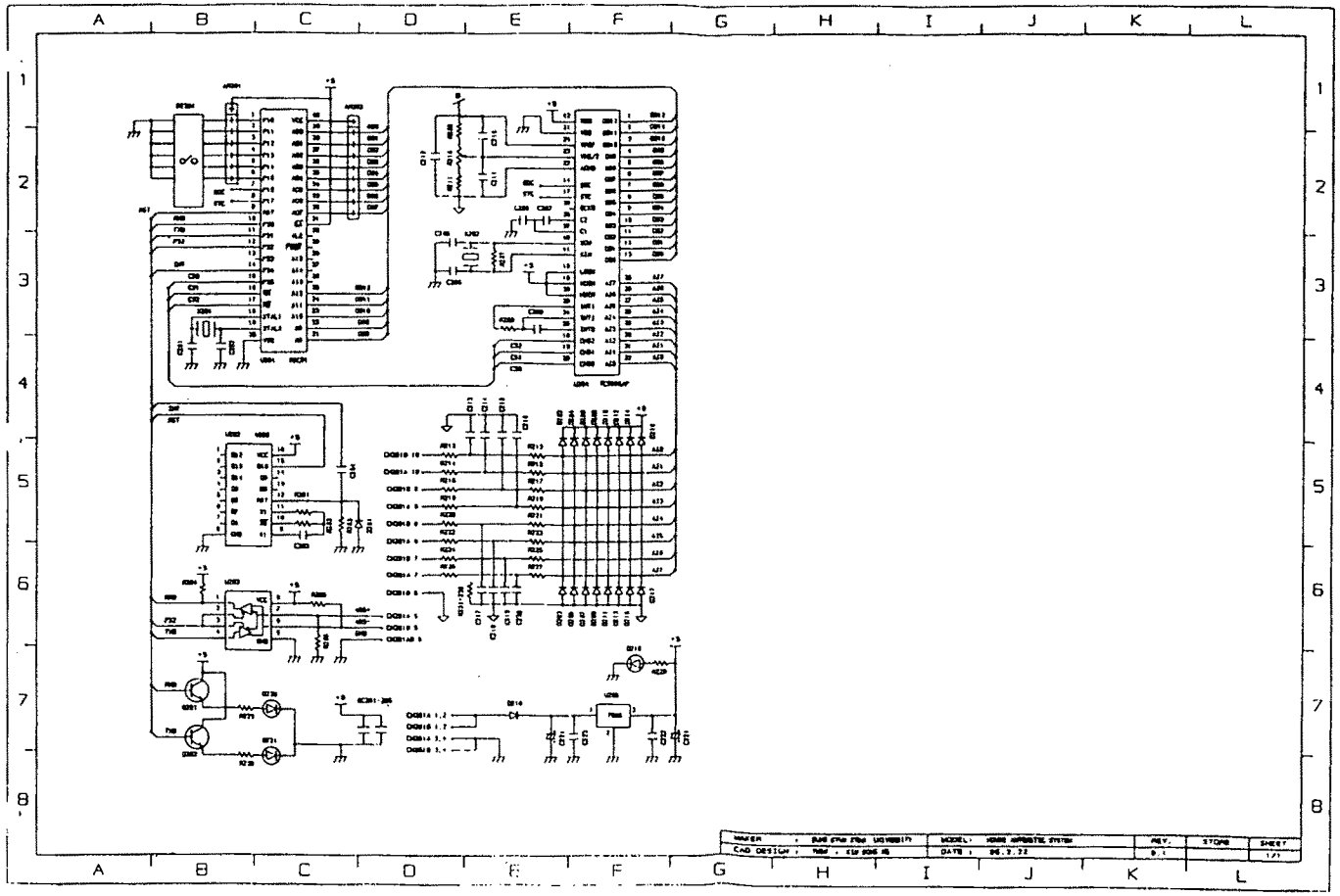
A-1. Main Module 회로도



A-2. Display Module 회로도



A-3. Relay Module 회로도



A-4. Sensor Input Module 회로도

B. PC용 프로그램

B-1. 메인 프로그램

```
Attribute VB_Name = "Main_Mod"

Private Sub Con_But_Click()
    Con_Frm.Visible = False
    Con_Frm.Visible = True
    Con_Frm.Enabled = True
End Sub

Private Sub Exit_But_Click()
    Call ZFIN
        Main_Frm.My_Modem.PortOpen = False
    End
End Sub

Private Sub Form_Load()
    Call Initial_Header
    Call Initial_CRC
    Call Initial_ESC
    Call Initial_Variable
    Main_Frm.My_Modem.InputLen = 1
    Main_Frm.My_Modem.InBufferCount = 0
    Main_Frm.My_Modem.OutBufferCount = 0
    House_Num = 1
    Main_Frm.Visible = False
    Main_1.Visible = True
    Main_1.Enabled = True
End Sub

Private Sub HSet_But_Click()
    HSet_Frm.Visible = False
    HSet_Frm.Visible = True
    HSet_Frm.Enabled = True
End Sub

Private Sub Log_But_Click()
    Log_Frm.Visible = False
    Log_Frm.Visible = True
    Log_Frm.Enabled = True
End Sub

Private Sub My_Modem_OnComm()
    If Main_Frm.My_Modem.PortOpen = False Then Exit Sub
    Select Case My_Modem.CommEvent
        Case comRxOver   ' Receive buffer overflow.
            My_Modem.InBufferCount = 0
        Case comEvReceive    ' Received RThreshold # of chars.
            Call Receive_Data
    End Select
End Sub

Private Sub Non_But_Click()
    Non_Frm.Visible = False
```

```
    Non_Frm.Visible = True
    Non_Frm.Enabled = True
End Sub

Private Sub Real_But_Click()
    Real_Frm.Visible = False
    Real_Frm.Visible = True
    Real_Frm.Enabled = True
End Sub
```

B-2. 환경설정 모드 프로그램

```
Attribute VB_Name = "Con_Mod"

Sub Config_Rcv()
    Close
    Open Save_File_Name For Binary As #1
        For i = 1 To AIData_Num
            Get #1, , AI_Con(i)
            AI_Con(i) = AI_Con(i)
        Next i
    Close #1
        Temp_Max_C(House_Num) = AI_Con(1)
        Temp_Min_C(House_Num) = AI_Con(2)
        Hum_Max_C(House_Num) = AI_Con(3)
        Hum_Min_C(House_Num) = AI_Con(4)
        Wind_Max_C(House_Num) = AI_Con(5)
        Wind_Str_C(House_Num) = AI_Con(6)
        Wind_St2_C(House_Num) = AI_Con(7)
        Rad_Con_C(House_Num) = AI_Con(8)
        T_Ground_C(House_Num) = AI_Con(9)
End Sub

Sub Config_Req(h4 As Byte, h5 As Byte, h6 As Byte, h7 As Byte, h8 As Byte)
    h4 = 18
    h5 = 0
    h6 = 0
    h7 = 0
    h8 = &H60
    Send_Flag = ConfigReq_Flag
    Call Send_File
End Sub

Sub Config_Send()
    Header_Array(4) = 18
    Header_Array(5) = 0
    Header_Array(6) = 0
    Header_Array(7) = 1
    Header_Array(8) = &H62
    Close
    Open Send_File_Name For Binary As #1
    Close #1
    Kill Send_File_Name
    Open Send_File_Name For Binary As 1
        Put #1, , Data_Group
    Close #1
    Send_Flag = ConfigSend_Flag
    Call Send_File
```

```
End Sub

Private Sub A_Option_Click()
    House_Num = 1
    Menu_Option = Configuration
    HNum_Con = House_Num
    Call ZRQINIT
End Sub

Private Sub B_Option_Click()
    House_Num = 2
    Menu_Option = Configuration
    HNum_Con = House_Num
    Call ZRQINIT
End Sub

Private Sub C_Option_Click()
    House_Num = 3
    Call ZRQINIT
End Sub

Private Sub Can_But_Click()
    T_Max_Text.SetFocus
End Sub

Private Sub D_Option_Click()
    House_Num = 4
    Call ZRQINIT
End Sub

Private Sub Exit_But_Click()
    Con_Frm.Visible = False
    Con_Frm.Enabled = False
    Main_Frm.Visible = True
    Main_Frm.Enabled = True
End Sub

Private Sub Form_Activate()
    Menu_Option = Configuration
    HNum_Con = House_Num
End Sub

Private Sub Form_Initialize()
    Menu_Option = Configuration
    HNum_Con = House_Num
    Call ZRQINIT
End Sub

Private Sub Form_Load()
    Menu_Option = Configuration
    HNum_Con = House_Num
    Call ZRQINIT
End Sub

Private Sub OK_But_Click()
    Crop_Name_C(House_Num) = Val(Obj_Name.Text)
    Temp_Max_C(House_Num) = Val(T_Max_Text.Text)
    Temp_Min_C(House_Num) = Val(T_Min_Text.Text)
    Hum_Max_C(House_Num) = Val(H_Max_Text.Text)
    Hum_Min_C(House_Num) = Val(H_Min_Text.Text)
    Wind_Max_C(House_Num) = Val(W_Max_Text.Text)
    Wind_Str_C(House_Num) = Val(W_Open_Text.Text)
```

```
    Wind_St2_C(House_Num) = Val(W_Rain_Text.Text)
    Rad_Con_C(House_Num) = Val(R_Con_Text.Text)
    T_Ground_C(House_Num) = Val(T_Ground_Text.Text)
End Sub
```

B-3. 하드웨어설정 모드 프로그램

```
Attribute VB_Name = "HSet_Mod"

Sub HWSetting_Rcv()
    Close
    Open Save_File_Name For Binary As #1
        Next i
    For i = 1 To 24
        HSet_Frm.Cu_RiDo_Times.Row = i
        For j = 2 To 5
            HSet_Frm.Cu_RiDo_Times.Col = j
            HSet_Frm.Cu_RiDo_Times.Text = Cu_RiDo_Array(House_Num, i, j + 1)
        Next j
    Next i
End Sub


Sub HWSetting_Req(h4 As Byte, h5 As Byte, h6 As Byte, h7 As Byte, h8 As Byte)
    Send_Flag = HWSettingReq_Flag
    Call Send_File
End Sub


Sub HWSetting_Send()
    Close
    Open Send_File_Name For Binary As #1
    Close #1
    Kill Send_File_Name
    Open Send_File_Name For Binary As #1
    For i = 1 To 24
    Next i
    Close #1
    Send_Flag = HWSettingSend_Flag
    Call Send_File
End Sub

Private Sub Can_But_Click()
    JunT_H.SetFocus
End Sub

Private Sub Cu_RiDo_Times_KeyPress(KeyAscii As Integer)
    Cu_RiDo_Times.Text = Cu_RiDo_Times.Text + Chr$(KeyAscii)
    If KeyAscii = 8 Then Cu_RiDo_Times.Text = ""
End Sub

Private Sub Exit_But_Click()
    HSet_Frm.Visible = False
    HSet_Frm.Enabled = False
    Main_Frm.Visible = True
    Main_Frm.Enabled = True
```

```
End Sub

Private Sub Form_Activate()
    Menu_Option = HardWare_Setting
End Sub

Private Sub Form_Load()
    Dim i, j
    Cu_RiDo_Times.Row = 0
    For i = 0 To 5
        Cu_RiDo_Times.ColWidth(i) = 400
        Cu_RiDo_Times.Col = i
        Cu_RiDo_Times.Text = Title(i + 1)
    Next i
    For i = 1 To 23 Step 2
      Cu_RiDo_Times.Col = 0
        Cu_RiDo_Times.Row = i
        Cu_RiDo_Times.Text = " " + Str$((i + 1) / 2)
        Cu_RiDo_Times.Row = i + 1
        Cu_RiDo_Times.Text = " " + Str$((i + 1) / 2)
      Cu_RiDo_Times.Col = 1
        Cu_RiDo_Times.Row = i
        Cu_RiDo_Times.Text = " " + "1"
        Cu_RiDo_Times.Row = i + 1
        Cu_RiDo_Times.Text = " " + "15"
    Next i
    Menu_Option = HardWare_Setting
    Call ZRQINIT
End Sub

Private Sub OK_But_Click()
    Dim i, j
    For i = 1 To 24
        Cu_RiDo_Times.Row = i
      For j = 0 To 5
        Cu_RiDo_Times.Col = j
        Cu_RiDo_Array(House_Num, i, j + 1) =
                              Val(Trim$(Cu_RiDo_Times.Text))
      Next j
    Next i
    Call HWSetting_Send
End Sub

Private Sub Option1_Click()
    House_Num = 1
    Menu_Option = HardWare_Setting
    Call ZRQINIT
End Sub

Private Sub Option2_Click()
    Call ZRQINIT
End Sub

Private Sub Option3_Click()
    House_Num = 3
    Call ZRQINIT
End Sub

Private Sub Option4_Click()
    House_Num = 4
    Call ZRQINIT
End Sub
```

B-4. 현재상황 모드 프로그램

```
Attribute VB_Name = "Real_Mod"

Sub Analysis_Current_Data()
    Dim Auto_Compare As Byte
    Auto_Compare = &H10
    If (DO_Real(1) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Side_Win1_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Side_Win1_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(2) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Side_Win2_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Side_Win2_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(3) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Up_Win_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Up_Win_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(4) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Curtain_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Curtain_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(6) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Fan_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Fan_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(8) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Hot_Water_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Hot_Water_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(9) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Heater_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Heater_Sign.BackColor = Auto_Mode_R
    End If
    If (DO_Real(10) And Auto_Compare) = Auto_Compare Then
        Real_Frm.Co2_Sign.BackColor = NonAuto_Mode_R
    Else: Real_Frm.Co2_Sign.BackColor = Auto_Mode_R
    End If
    Dim Act_Comp As Byte
    Act_Comp = 1
    If (DO_Real(1) And Act_Comp) = Act_Comp Then
        Real_Frm.Side_Win1_Sign = "열림"
        Real_Frm.Side_Win1_Close.Visible = False
        Real_Frm.Side_Win1_Open.Visible = True
    Else
        Real_Frm.Side_Win1_Sign = "닫힘"
        Real_Frm.Side_Win1_Close.Visible = True
        Real_Frm.Side_Win1_Open.Visible = False
    End If
    If (DO_Real(2) And Act_Comp) = Act_Comp Then
        Real_Frm.Side_Win2_Sign = "열림"
        Real_Frm.Side_Win2_Close.Visible = False
        Real_Frm.Side_Win2_Open.Visible = True
    Else
        Real_Frm.Side_Win2_Sign = "닫힘"
        Real_Frm.Side_Win2_Close.Visible = True
        Real_Frm.Side_Win2_Open.Visible = False
    End If
    If (DO_Real(3) And Act_Comp) = Act_Comp Then
```

```
            Real_Frm.Up_Win_Sign = "열림"
            Real_Frm.Up_Win_Close.Visible = False
            Real_Frm.Up_Win_Open.Visible = True
        Else
            Real_Frm.Up_Win_Sign = "닫힘"
            Real_Frm.Up_Win_Close.Visible = True
            Real_Frm.Up_Win_Open.Visible = False
        End If
        If (DO_Real(4) And Act_Comp) = Act_Comp Then
            Real_Frm.Curtain_Sign = "열림"
            Real_Frm.Curtain_Close.Visible = False
            Real_Frm.Curtain_Open.Visible = True
        Else
            Real_Frm.Curtain_Sign = "닫힘"
            Real_Frm.Curtain_Close.Visible = True
            Real_Frm.Curtain_Open.Visible = False
        End If
        If (DO_Real(6) And Act_Comp) = Act_Comp Then
            Real_Frm.Fan_Sign = "동작"
            For i = 0 To 3
            Real_Frm.Fan_On(i).Visible = True
            Next i
        Else:
            Real_Frm.Fan_Sign = "멈춤"
            For i = 0 To 3
            Real_Frm.Fan_On(i).Visible = False
            Next i
        End If
        If (DO_Real(8) And Act_Comp) = Act_Comp Then
            Real_Frm.Hot_Water_Sign = "동작"
            Real_Frm.hot_water_on.FillStyle = 0
        Else:
            Real_Frm.Hot_Water_Sign = "멈춤"
            Real_Frm.hot_water_on.FillStyle = 1
        End If
        If (DO_Real(9) And Act_Comp) = Act_Comp Then
            Real_Frm.Heater_Sign = "동작"
            For i = 0 To 4
                Real_Frm.heater_on(i).Visible = True
            Next i
        Else:
            Real_Frm.Heater_Sign = "멈춤"
            For i = 0 To 4
                Real_Frm.heater_on(i).Visible = False
            Next i
        End If
        If (DO_Real(10) And Act_Comp) = Act_Comp Then
            Real_Frm.Co2_Sign = "동작"
            For i = 0 To 3
                Real_Frm.Co2_On(i).Visible = True
            Next i
        Else:
            Real_Frm.Co2_Sign = "멈춤"
            For i = 0 To 3
                Real_Frm.Co2_On(i).Visible = False
            Next i
        End If
    End Sub

    Sub RealState_Rcv()
        Dim Dir_Sign As String
        Close
```

```
    Open Save_File_Name For Binary As #1
    For i = 1 To 6
        Get #1, , Date_Real(i)
    Next i
    For i = 1 To AIData_Num
        Get #1, , AI_Real(i)
        AI_Real_Single(i) = AI_Real(i) / 10
    Next i
    For i = 1 To DIData_Num
        Get #1, , DI_Real(i)
    Next i
    For i = 1 To DOData_Num
        Get #1, , DO_Real(i)
    Next i
    Close #1
    Call Analysis_Current_Data
    Select Case AI_Real(12)
        Case 1: Dir_Sign = "E"
        Case Else: Dir_Sign = "None"
    End Select
    Real_Frm.Win_Dir_Txt = Dir_Sign
End Sub

Sub RealState_Req(h4 As Byte, h5 As Byte, h6 As Byte, h7 As Byte, h8 As Byte)
    Send_Flag = RealStateReq_Flag
    Call Send_File
End Sub

Private Sub A_Opt_Click()
    House_Num = 1
    Menu_Option = Real_State
    Call ZRQINIT
    House_Num_Txt = "A"
End Sub

Private Sub B_Opt_Click()
    House_Num = 2
    Menu_Option = Real_State
    Call ZRQINIT
    House_Num_Txt = "B"
End Sub

Private Sub C_Opt_Click()
    House_Num = 3
    Call ZRQINIT
    House_Num_Txt = "C"
End Sub

Private Sub D_Opt_Click()
    House_Num = 4
    Call ZRQINIT
    House_Num_Txt = "D"
End Sub

Private Sub Exit_But_Click()
    Real_Frm.Timer1.Enabled = False
End Sub

Private Sub Exit_but_KeyPress(KeyAscii As Integer)
    If KeyAscii = 13 Then End
End Sub
```

```
Private Sub Form_Activate()
    Menu_Option = Real_State
    Current_Data_Period = 0
    Real_Frm.Timer1.Enabled = True
End Sub

Private Sub Form_Load()
    Menu_Option = Real_State
    Current_Data_Period = 0
    Real_Frm.Timer1.Enabled = True
    Call ZRQINIT
End Sub

Private Sub Form_LostFocus()
    Real_Frm.Timer1.Enabled = False
End Sub

Private Sub Timer1_Timer()
   Date_Box.Text = Mid$(Date$, 3, 2) + "/" + Mid$(Date$, 6, 2) + "/" + Right$(Date$, 2)
    Time_Box.Text = Time$
    If Menu_Option = Real_State Then
        Current_Data_Period = Current_Data_Period + 1
    End If
    If Current_Data_Period = 5 Then
        Current_Data_Period = 0
        Call Date_Good_Case
    End If
End Sub
```

B-5. 자료보기 모드 프로그램

```
Attribute VB_Name = "Log_Mod"

Sub DataLog_Analysis(Row_Num As Integer)
    Dim Input_Val As Byte
    Dim a_str As String
        Date_YMD = ""
        Time_HMS = ""
        a_str = ""
        For i = 1 To 3
            Input_Val = DateTime_Log(i)
            a_str = Str$(Input_Val)
            a_str = Trim$(a_str)
            If Input_Val < 10 Then a_str = "0" + a_str
            Date_YMD = Date_YMD + a_str
        Next i
        a_str = ""
        For i = 4 To 6
            Input_Val = DateTime_Log(i)
            a_str = Str$(Input_Val)
            a_str = Trim$(a_str)
            If Input_Val < 10 Then a_str = "0" + a_str
            Time_HMS = Time_HMS + a_str
        Next i
        Date_YMD=Left$(Date_YMD,2)+"-"+Mid$(Date_YMD,3,2)+
                    "-" + Right$(Date_YMD, 2)
```

```
        Time_HMS=Left$(Time_HMS,2)+":"+Mid$(Time_HMS,3,2)+
                                ":" + Right$(Time_HMS, 2)
End Sub

Sub DataLog_Rcv()
    Dim i As Integer
    Dim j As Integer
    Dim n As Integer
    Open Log_AI_File_Name For Binary As #2
    Close #2
    Kill Log_AI_File_Name
    Open Log_AI_File_Name For Binary As #2
    Open Log_DI_File_Name For Binary As #3
    Close #3
    Kill Log_DI_File_Name
    Open Log_DI_File_Name For Binary As #3
    Open Log_DO_File_Name For Binary As #4
    Close #4
    Kill Log_DO_File_Name
    Open Log_DO_File_Name For Binary As #4
    Open Log_Time_File_Name For Binary As #5
    Close #5
    Kill Log_Time_File_Name
    Open Log_Time_File_Name For Binary As #5
    Open Save_File_Name For Binary As #1
    n = LOF(1) / (6 + (AIData_Num * 2) + DIData_Num + DOData_Num)
    Log_Frm.Data_Log.Rows = n + 1
    Call DataLog_Analysis(i)
    Next i
    Close
End Sub

Sub DataLog_Req(h4 As Byte, h5 As Byte, h6 As Byte, h7 As Byte, h8 As Byte)
    Send_Flag = DataLogReq_Flag
    Call Send_File
End Sub

Sub LoadFile_Fun(F_N As String)
    Dim Trans_LogData As Byte
    Open Save_File_Name For Binary As #1
    Close #1
    Kill Save_File_Name
    Open Save_File_Name For Binary As #1
    Open F_N For Binary As #2
    For i = 1 To LOF(2)
        Get #2, , Trans_LogData
        Put #1, , Trans_LogData
    Next i
    Close
    Log_Save.Enabled = False
    Log_Save.Visible = False
    Call DataLog_Rcv
End Sub

Sub SaveFile_Fun(F_N As String)
    Dim Trans_LogData As Byte
    Open F_N For Binary As #1
    Close #1
    Kill F_N
    Open F_N For Binary As #1
    Open Save_File_Name For Binary As #2
    For i = 1 To LOF(2)
```

```
            Get #2, , Trans_LogData
            Put #1, , Trans_LogData
        Next i
        Close
        Log_Save.Enabled = False
        Log_Save.Visible = False
End Sub

Private Sub A_Opt_Click()
        House_Num = 1
        Menu_Option = Data_Logger
        HNum_Con = House_Num
        Call ZRQINIT
End Sub

Private Sub Act_But_Click()
End Sub

Private Sub B_Opt_Click()
        House_Num = 2
        Menu_Option = Data_Logger
        HNum_Con = House_Num
        Call ZRQINIT
End Sub

Private Sub C_Opt_Click()
        House_Num = 3
        Call ZRQINIT
End Sub

Private Sub D_Opt_Click()
        House_Num = 4
        Call ZRQINIT
End Sub

Private Sub Form_Activate()
        Menu_Option = Data_Logger
End Sub

Private Sub Form_Load()
        Dim i, j
        Data_Log.Row = 0
        Menu_Option = Data_Logger
        HNum_Con = House_Num
        Call ZRQINIT
End Sub

Private Sub FT_But_Click()
End Sub

Private Sub Load_But_Click()
End Sub

Private Sub Save_But_Click()
        Log_Save.Load_But.Enabled = False
        Log_Save.Save_But.Enabled = True
        Log_Save.Enabled = True
        Log_Save.Visible = True
End Sub
```

B-6. 수동모드 프로그램

```
Attribute VB_Name = "Non_Mod"

Sub Non_Act_Send()
    If Non_Frm.Auto_Opt.Value = True Then Exit Sub
    Dim a
    a = DO_Non(Non_Index) And 3
    If Non_Index < 5 Then
        Select Case a
            Case 0, 2              If (Non_Frm.Off_Opt.Value = True) Then Exit Sub
            Case 1                 If (Non_Frm.Down_Opt.Value = True) Then Exit Sub
            Case 3                 If (Non_Frm.On_Up_Opt.Value = True) Then Exit Sub
        End Select
    Else
        Select Case a
            Case 1, 3            If (Non_Frm.On_Up_Opt.Value = True) Then Exit Sub
            Case Else            If (Non_Frm.Off_Opt.Value = True) Then Exit Sub
        End Select
    End If
    Dim Non_Data_Value As Byte
    Close
    Open Send_File_Name For Binary As #1
    Close #1
    Kill Send_File_Name
    Open Send_File_Name For Binary As #1
    Close #1
    Send_Flag = NonActSend_Flag
    Call Send_File
    If Non_Index < 5 Then
        If Non_Frm.On_Up_Opt.Value = True Then
            DO_Non(Non_Index) = DO_Non(Non_Index) Or &H3
        ElseIf Non_Frm.Down_Opt.Value = True Then
            DO_Non(Non_Index) = DO_Non(Non_Index) And &HFD
            DO_Non(Non_Index) = DO_Non(Non_Index) Or 1
        Else: DO_Non(Non_Index) = DO_Non(Non_Index) And &HFE
        End If
    Else
        If Non_Frm.On_Up_Opt.Value = True Then
            DO_Non(Non_Index) = DO_Non(Non_Index) Or 1
        Else: DO_Non(Non_Index) = DO_Non(Non_Index) And &HFE
        End If
    End If
End Sub

Sub Non_Or_Auto_Send()
    If (Non_Frm.Auto_Opt.Value = True) And ((DO_Non(Non_Index) And &H10) <>
&H10) Then Exit Sub
    If (Non_Frm.NonA_Opt.Value = True) And ((DO_Non(Non_Index) And &H10) =
&H10) Then Exit Sub
    Dim Non_Data_Value As Byte
    Close
    Open Send_File_Name For Binary As #1
    Close #1
    Kill Send_File_Name
    Open Send_File_Name For Binary As #1
    Put #1, , Non_Index
        If Non_Frm.NonA_Opt.Value = True Then
            Non_Data_Value = &H10
        Else: Non_Data_Value = 0
```

```
        End If
    Put #1, , Non_Data_Value
    Close #1
    Send_Flag = NonOrAutoSend_Flag
    Call Send_File
    If Non_Frm.Auto_Opt.Value = True Then
        DO_Non(Non_Index) = DO_Non(Non_Index) And &HEF
    Else
        DO_Non(Non_Index) = DO_Non(Non_Index) Or &H10
    End If
End Sub

Sub NonAuto_Rcv()
    Dim i As Integer
    Close
    Open Save_File_Name For Binary As #1
    For i = 1 To DIData_Num
        Get #1, , DI_Non(i)
    Next i
    For i = 1 To DOData_Num
        Get #1, , DO_Non(i)
    Next i
    Close #1
End Sub

Sub NonAuto_Req(h4, h5, h6, h7, h8)
    Send_Flag = NonAutoReq_Flag
    Call Send_File
End Sub

Private Sub A_Opt_Click()

    House_Num = 1
    Menu_Option = NonAutomatic_Control
    Call ZRQINIT
End Sub

Private Sub Actor_Click(Index As Integer)
    Non_Index = Index
    Non_Frm.Auto_Opt.Enabled = True
    Non_Frm.NonA_Opt.Enabled = True
    Select Case Index
        Case 1, 2, 3, 4
            Non_Frm.On_Up_Opt.Caption = "열림"
            Non_Frm.Down_Opt.Visible = True
            Non_Frm.On_Up_Opt.Enabled = True
            Non_Frm.Off_Opt.Enabled = True
            Non_Frm.Down_Opt.Enabled = True
            If (DO_Non(Non_Index) And 1) = 1 Then
                If (DO_Non(Non_Index) And 2) = 2 Then
                    Non_Frm.On_Up_Opt.Value = True
                Else
                    Non_Frm.Down_Opt.Value = True
                End If
            Else
                    Non_Frm.Off_Opt.Value = True
            End If
            If (DO_Non(Non_Index) And &H10) = &H10 Then
                If (DO_Non(Non_Index) And 3) = 3 Then
                    Non_Frm.Down_Opt.Enabled = False
                ElseIf (DO_Non(Non_Index) And 1) = 1 Then
                    Non_Frm.On_Up_Opt.Value = False
```

```
            End If
        Else
            Non_Frm.On_Up_Opt.Enabled = False
            Non_Frm.Off_Opt.Enabled = False
            Non_Frm.Down_Opt.Enabled = False
        End If
        Non_Command_Code = &H4B
    Case 6, 8, 9, 10
        Non_Frm.On_Up_Opt.Caption = "동작"
        Non_Frm.Down_Opt.Visible = False
        Non_Frm.Down_Opt.Enabled = False
        Non_Frm.On_Up_Opt.Enabled = True
        Non_Frm.Off_Opt.Enabled = True
        If (DO_Non(Index) And 1) = 1 Then
            Non_Frm.On_Up_Opt.Value = True
        Else: Non_Frm.Off_Opt.Value = True
        End If
        If (DO_Non(Index) And &H10) <> &H10 Then
            Non_Frm.On_Up_Opt.Enabled = False
            Non_Frm.Off_Opt.Enabled = False
        End If
        Non_Command_Code = &H4B
End Select
If (DO_Non(Index) And &H10) = &H10 Then
    NonA_Opt.Value = True
Else: Auto_Opt.Value = True
End If
End Sub

Private Sub Auto_Opt_Click()
    Call Non_Or_Auto_Send
    Non_Frm.On_Up_Opt.Enabled = False
    Non_Frm.Off_Opt.Enabled = False
    Non_Frm.Down_Opt.Enabled = False
End Sub

Private Sub B_Opt_Click()
    House_Num = 2
    Menu_Option = NonAutomatic_Control
    Call ZRQINIT
End Sub

Private Sub C_Opt_Click()
    House_Num = 3
    Call ZRQINIT
End Sub

Private Sub D_Opt_Click()
    House_Num = 4
    Call ZRQINIT
End Sub

Private Sub Down_Opt_Click()
    On_Up_Opt.Enabled = False
    Call Non_Act_Send
End Sub

Private Sub Exit_But_Click()
    Non_Frm.Visible = False
    Non_Frm.Enabled = False
    Main_Frm.Visible = True
    Main_Frm.Enabled = True
```

```
End Sub

Private Sub Form_Activate()
    Menu_Option = NonAutomatic_Control
End Sub

Private Sub Form_Load()
    Menu_Option = NonAutomatic_Control
    Non_Frm.Auto_Opt.Enabled = False
    Non_Frm.NonA_Opt.Enabled = False
    Non_Frm.On_Up_Opt.Enabled = False
    Non_Frm.Off_Opt.Enabled = False
    Non_Frm.Down_Opt.Enabled = False
    Call ZRQINIT
End Sub

Private Sub NonA_Opt_Click()
    Call Non_Or_Auto_Send
    Non_Frm.Off_Opt.Enabled = True
    If Non_Index < 5 Then
        If (DO_Non(Non_Index) And 3) = 3 Then
            Non_Frm.On_Up_Opt.Enabled = True
            Non_Frm.Off_Opt.Enabled = True
            Non_Frm.Down_Opt.Enabled = False
        ElseIf (DO_Non(Non_Index) And 3) = 1 Then
            Non_Frm.On_Up_Opt.Enabled = False
            Non_Frm.Off_Opt.Enabled = True
            Non_Frm.Down_Opt.Enabled = True
        Else
            Non_Frm.On_Up_Opt.Enabled = True
            Non_Frm.Off_Opt.Enabled = True
            Non_Frm.Down_Opt.Enabled = True
        End If
    Else
            Non_Frm.On_Up_Opt.Enabled = True
            Non_Frm.Off_Opt.Enabled = True
    End If
End Sub

Private Sub Off_Opt_Click()
    On_Up_Opt.Enabled = True
    If Non_Index < 5 Then
        Down_Opt.Enabled = True
    End If
    Call Non_Act_Send
End Sub

Private Sub On_Up_Opt_Click()
    If Non_Index < 5 Then
        Down_Opt.Enabled = False
    End If
    Call Non_Act_Send
End Sub
```

C. 하드웨어용 프로그램

C-1. 메인 프로그램

```
void main(void){
        Delay(100);
        Init();
         TimerSerInit();
        RUN_APP_FLAG = 1;
        ClrInterrupt();
        ClrApp();
        AddInterrupt(0,  30, testInt1);
        AddInterrupt(1,  30, testInt2);
          RunApplication();
}

void DipRead(void){
        CpuId = ioport0;    CpuId = ~CpuId;
        CpuId >>= 4;
        CpuId &= 0x0f;
}

void Init(void){
        BYTE i;
        RUN_APP_FLAG = 0;
        TxdStartPoss = TxdEndPoss = 0;
        MsecData1 = MsecData2 = MsecData3 = 0
        RxdStartPoss = RxdEndPoss = 0;
        RamBeginAdr = RamEndAdr = 0L;
        StartNormal = 0;
        DataSendWork = 0;
        rtu_id = 1;
        SyEroChk = 0;
        TxdRxdSelt &= RxdSelt;
   InitModem();
        RunDs1287();
        SenAdrCnt = 0;
        ReadSec = 0;
        SecFlg = 0;
}

void TimerSerInit(void){
         wsr = HWINDOW15;
         timer1 = ONE_MSEC;
         wsr = HWINDOW0;
}

void RunDs1287(void){
   BYTE *LExtAdr;
   int i;
}

void real_timer(void){
        int i, no;
        FP CallFunc;
                }
        }
        if (!(no < 0)) {
```

```
        }
}

void AddInterrupt(int no, int Count , void (*FuncP)(void)){
        } else Result_Int = 0;
}

void DelInterrupt(int no){
        if ((no >= 0) && (no < INT_SIZE)) {
                TiTable[no].TISave  = 0;
                TiTable[no].TICount = 0;
                Result_Int = 1;
        } else Result_Int = 0;
}

void ClrInterrupt(void){
        int i;
        for (i=0; i < INT_SIZE; i++) {
                TiTable[i].TISave  = 0;
                TiTable[i].TICount = 0;
        } }

void AddApp(int pri, void (*FuncP)(void)){
        int i;
        Result_Int = -1;
                } } }

void DelApp(int no){
        if ((no >=0) && (no < KNR_SIZE)) {
                OSTable[no].CurrPri = 0;
                OSTable[no].SavePri = 0;
        } }

void ActApp(int no){
        if ((no >= 0) && (no < KNR_SIZE)) {
                if (OSTable[no].SavePri) {
                        OSTable[no].CurrPri &= 0x7f
                } } }

void ClrApp(void){
        memset(OSTable. 0x0, sizeof(OSTable));
        return value : long

long EncodeLongTime(int hour, int min, int sec){
}

void ReadTime(void){
   BYTE *LExtAdr;
   disable();
   LExtAdr = (BYTE *) CLK_ADR;
   enable();
}

void WriteTime(void){
   BYTE *LExtAdr;
   disable();
   enable();
}

void testInt1(void){
          } } }
```

```
void testInt2(void){
        MsecData1++;
        if(MsecData1 > 10){
        }
        if(MsecData2 > 10){
        }
        if(ReadSec > 33){
           ReadSec = 0;
        } }

void CovRlySub(BYTE *WinKide, BYTE *DoKide, BYTE UpVal, BYTE DowunVal){
   *DoKide &= ~UpVal;    *DoKide &= ~DowunVal;
   if(*WinKide & 2){
           if(*WinKide & 1) *DoKide |= UpVal;
   } else if(*WinKide & 1) *DoKide |= DowunVal;
}

void CovRlySub2(BYTE *WinKide, BYTE *DoKide, BYTE OpenVal){
   if(*WinKide & 1) *DoKide |= OpenVal;
   else * DoKide &= ~OpenVal;
}

void CovRlyData(void){
   BYTE i;
   int Cval;
}

BYTE StartPoint(BYTE Fdata){
        switch(Fdata){
                case 0x18:
                case '*':
                        Fdata = Fdata ^ 0x40;
                        break;
        }
        return(Fdata);
}

void PushRamSave(BYTE SaveData){
}

void RamErase(void){
   if(RamBeginAdr != RamEndAdr) RamBeginAdr++;
   RamBeginAdr %= RAM_END_MAX;
}

int PopRamSave(void){
   int Aval = -1;
   BYTE  *LExtrl;
        }
        return(Aval);
}

void RamRecodr(BYTE Adata, BYTE StDa){
   BYTE i, Aval;
   if(RamBeginAdr == (RamEndAdr + 1) % RAM_END_MAX) RamErase();
   Aval = Adata;
   if(StDa == 0){
           Aval = StartPoint(Adata);
           if(Aval != Adata) PushRamSave(0x18);
   }
   PushRamSave(Aval);
}
```

```
void RamRecodrBlock(void){
   BYTE Aval, i;
   RamRecodr('*', 1); /* data beging */
   for(i=0; i<6; i++) RamRecodr(year_data[i], 0);
   for(i=0; i<16; i++){
            Aval = LuAiData[i];       RamRecodr(Aval, 0);
            Aval = LuAiData[i] >> 8; RamRecodr(Aval, 0)
   }
   for(i=0; i<24; i++) RamRecodr(LuDiData[i], 0);
   for(i=0; i<12; i++) RamRecodr(LuDoData[i], 0);
   RamRecodr('*', 1);
}

int RamReadBlock(void){
   int Aval, Bval;
   BYTE i;
   for(i=0; i < 200; i++){
            Aval = PopRamSave();
            if(Aval < 0) return(-1);
            if(Aval == '*') break;
   }
   for(i=0; i < 250; i++){
            Aval = PopRamSave();
            if(Aval < 0) return(-1);
            if(Aval == '*') return(0);
            if(Aval == 0x18) Aval = PopRamSave() ^ 0x40
            crc_32value = Aval;
            CRC32();
            Bval = ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(0x18);
            PushModem2Txd(Bval);
   }
   return(0);
}

void AiValueData(void){
  int Aval, Bval, i;
         PushModem2Txd('*');
         PushModem2Txd(0x18);
         PushModem2Txd(' ');
   }
   Aval = ActionFlagBog;
   for(i=0; i<16; i++){
            if(Aval & 1) PushModem2Txd('1');
            else         PushModem2Txd('0');
            Aval >>= 1;
   }
   PushModem2Txd('h');
   PushModem2Txd(13);
  while(1){
           Aval = PopModem2Txd();
           if(Aval < 0) break;
           Modem2ByteSend(Aval);
  } }

void SecCmpValue(void){
          if(year_data[5] != SecBo){
             SecBo = year_data[5];
             ActionCmpTime[SYS_CHK_TIM]++;
             ActionCmpTime[GIGI_RUN_TIM]++;
  } }
```

```
void TestDeciaml(int Bval){
        int Aval;
}

BYTE AddRamTime(BYTE RSec, BYTE Rmin, BYTE RHou, int AvalData){
   long Aval, Bval;
   return(RSec);
}

void RunApplication(void){
        BYTE i;
        int Cval;
                control();
                if(ConvFlag) CovRlyData();
          } } }

void Testeciaml(int Bval){
        int Aval;
}

WORD machine(int *SenPoin, WORD SysFlag){
        BYTE i;
        int Aval;
        WORD RunFlag = 0;
        while(1){
          if(*(SenPoin + INSIDE_TEMP)  > *(SenPoin + TEM_CMP_HAIGH)){
         if(*(SenPoin + INSIDE_HUM)  < *(SenPoin + HUM_CMP_LOW)) RunFlag |=
FOG_FLAG; /* Col run*/  else RunFlag |= COL_FLAG; /* fog run */
                break;    }
          if(*(SenPoin + INSIDE_TEMP) < *(SenPoin + TEM_CMP_LOW)){
                        RunFlag |= HEAT_FLAG;
                        break; /* heat run */    }
         if(*(SenPoin + INSIDE_HUM) < *(SenPoin + HUM_CMP_HAIGH)){
         if(*(SenPoin + INSIDE_HUM)  > *(SenPoin + HUM_CMP_LOW)) break;
         RunFlag |= BELL_FLAG;
         break;    }
                if(*(SenPoin    +    OUTSIDE_HUM)        >    *(SenPoin    +
HUM_CMP_HAIGH)){
                        RunFlag |= BELL_FLAG; /* bell run */
                }
                if((SysFlag & FAN_FLAG) == FAN_FLAG){
                        RunFlag |= BELL_FLAG; /* bell run */
                        break;      }
                RunFlag |= FAN_FLAG; /* fan run */
                break;     }
  return(RunFlag);
}

WORD HeatRun(WORD SysFlag, BYTE SenChkVal){
   WORD RunFlag = 0;
        while(1){
   }
  return(RunFlag);
}

WORD RainRuning(int *SenPoin, WORD SysFlag, BYTE RunFlag, BYTE SenChkVal){
   if((SysFlag & FAN_FLAG) != FAN_FLAG){
          RunFlag |= FAN_FLAG;
          return(RunFlag);
   }
   if(*(SenPoin + WIND_SPEED) > *(SenPoin + WIND_STR2)){
```

```
            RunFlag |= BELL_FLAG;
            return(RunFlag);
    }
    if((SenChkVal & WIN_VECTOR_WEST) == WIN_VECTOR_WEST){
             if((SysFlag & WIN_WEST_UFLAG) == WIN_WEST_UFLAG) RunFlag
WIN_WEST_DFLAG;
            if((SysFlag & WIN_EAST_UFLAG) != WIN_EAST_UFLAG){
                    SysFlag |= WIN_WEST_UFLAG;
                    return(RunFlag);
            }
            RunFlag |= BELL_FLAG;
            return(RunFlag);
    }
    if(((SysFlag & WIN_EAST_UFLAG) != WIN_EAST_UFLAG ||
            (SysFlag & WIN_WEST_UFLAG) != WIN_WEST_UFLAG)){
            RunFlag |= WIN_EAST_UFLAG + WIN_WEST_UFLAG;
            return(RunFlag);
    }
    RunFlag |= BELL_FLAG;
    return(RunFlag);
}

WORD ColRun(int *SenPoin, WORD SysFlag, BYTE SenChkVal){
WORD RunFlag = 0;
RunFlag |= WIN_WEST_DFLAG;
                    if((SysFlag & WIN_EAST_UFLAG) != WIN_EAST_UFLAG){
                            RunFlag |= WIN_EAST_UFLAG;
                            break;
                    }
RunFlag |= WIN_EAST_DFLAG;
                    if((SysFlag & WIN_WEST_UFLAG) != WIN_WEST_UFLAG){
                            RunFlag |= WIN_WEST_UFLAG;
                            break;
                    }
    }
    return(RunFlag);
}

void control(void){
    WORD RunFlag, SysFlag, ActionFlag;
    BYTE SenChkVal, Aval;
            SenChkVal = 0;
            SysFlag = ActionFlagBog;
            ActionCmpTime[SYS_CHK_TIM_CMP] = 2;
}

int PushTxdData(int Aval){
    if(TxdStartPoss != (TxdEndPoss + 1)){
            TxdData[TxdEndPoss++] = Aval;
    } else   Aval = -1;
    TxdEndPoss  %= 256;
    return(Aval);
}

int PopRxdData(void){
    int Aval = -1;
    if(RxdStartPoss != RxdEndPoss){
            Aval = RxdData[RxdStartPoss++];
            RxdStartPoss %= 256;
    } else  LcRxdFullFlg = 0;
    return(Aval);
}
```

```
int PushRxdData(int Aval){
   if(RxdStartPoss != (RxdEndPoss + 1)){
            RxdData[RxdEndPoss++] = Aval;
            RxdEndPoss  %= 256;
   } else Aval = -1;
   LcRcvStrtFlg = 1;
   LcRcvWaitTim = 0;
   return(Aval);
}

int PopModem1RxdData(void){
   int Aval = -1;
   if(Modem1TxdStrat != Modem1TxdEnd){
            Aval = Modem1TxdData[Modem1TxdStrat++];
            Modem1TxdStrat %= 1024;
   }
   return(Aval);
}

int PushModem2Txd(int Aval){
   if(Modem2TxdStrat != (Modem2TxdEnd + 1)){
            Modem2TxdData[Modem2TxdEnd++] = Aval;
   }  else    Aval = -1;
   Modem2TxdEnd  %= 1024;
   return(Aval);
}

int PopModem2Txd(void){
   int Aval = -1;
   if(Modem2TxdStrat != Modem2TxdEnd){
            Aval = Modem2TxdData[Modem2TxdStrat++];
            Modem2TxdStrat %= 1024;
   }
   return(Aval);
}

int PopModem2RxdData(void){
   int Aval = -1;
   if(Modem2RxdStart != Modem2RxdEnd){
            Aval = Modem2RxdData[Modem2RxdStart++];
            Modem2RxdStart %= 1024;
   } else  MoRxdFullFlg = 0;
   return(Aval);
}

void SendInter(void){
   int Aval;
   disable();
   Aval = PopTxdData();
   if(Aval >= 0){
            sbuf = Aval;
   } else{
           TxdRxdSelt &= RxdSelt;
   }
   Aval = sp_stat;
   enable();
}

void InitModem(void){
          int i;
}
```

```
void ExtClrSrial(void){
        BYTE Aval;
}

void Modem1Send(char Data){
   BYTE *LExtAdr;
   disable();
}
void Modem2Send(char Data){
   BYTE *LExtAdr;
   disable();
   LExtAdr = (BYTE *) C452_COM2;
   *(LExtAdr + THR) = Data;
   enable();
}

void SubC452Com1(void){
   BYTE *LExtAdr, Aval;
   LExtAdr = (BYTE *) C452_COM1;
   Aval = *(LExtAdr+RBR);
   if(*(LExtAdr+LSR) & 0x1){
           if(Modem1TxdStrat != (Modem1TxdEnd + 1)){
                   RxdData[Modem1TxdEnd++] = Aval;
           }
           Modem1TxdEnd  %= 1024;
   }
   Aval = *(LExtAdr+LSR);
}

void SubC452Com2(void){
   unsigned char Aval, *LExtAdr;
   LExtAdr = (BYTE *) C452_COM2;
   Aval = 0;
   if(*(LExtAdr+LSR) & 1){
           Aval = *(LExtAdr+RBR);
                   if(Modem2RxdStart       !=       (Modem2RxdEnd       +       1))
Modem2RxdData[Modem2RxdEnd++] = Aval;
           Modem2RxdEnd  %= 1024;
   }

   if(Modem2RxdStart > Modem2RxdEnd)  Aval = Modem2RxdStart - Modem2RxdEnd;
   else                               Aval = Modem2RxdEnd - Modem2RxdStart;
   if(Aval > 800) MoRxdFullFlg = 1;
   MoRcvStrtFlg = 1;
   MoRcvWaitTim = 0;
   Aval = *(LExtAdr+LSR);
}

void TSralC452Inter(void){
   unsigned char Aval;
   disable();
        Aval = 0;
   enable();
}

void SralC452Inter(void){
   unsigned char Aval;
   disable();
   Aval = ioport0;
   enable();
}
```

```
void ModemOff(void){
   BYTE *LExtAdr;
   LExtAdr = (BYTE *) C452_COM2;
   *(LExtAdr+MCR) = 0; /* modem disable and modem resiver */
}

void ModemOn(void){
   BYTE *LExtAdr;
   LExtAdr = (BYTE *) C452_COM2;
  *(LExtAdr+MCR) = 11; /* modem enable and modem resiver */
}

void ChkCtsWait(void){
   BYTE *LExtAdr, j, i;
   LExtAdr = (BYTE *) C452_COM2;
  j = 0;
  while((i == 0) && (j < 30)){
          i = *(LExtAdr + MSR); /* read modem sate*/
          i = i & CTS;
          j++;
  }     }

void Chk0LsrWait(void){
   BYTE *LExtAdr, i;
   LExtAdr = (BYTE *) C452_COM1;
   for(;;) {
            i = *(LExtAdr+LSR); /* read modem sate */
            if (i & TEMY) break;
  }     }

void ChkLsrWait(void){
   BYTE *LExtAdr, i;
   LExtAdr = (BYTE *) C452_COM2;
   for(;;) {
            i = *(LExtAdr+LSR); /* read modem sate */
            if (i & TEMY) break;
  }     }

void OpenWriteModem(int *str, int lengh){
  int i;
  for(i=0; i<= lengh; i++){
          ChkLsrWait();
          Modem2Send(*str);
          *str++;
 }     }

void StartTxd(void){
   BuyTxd = 0x55;
   TxdSendCunt = 0;
   sbuf = '*';
}

void SendDataTest(void){
   WORD i;
   BYTE ChekSum;
   ChekSum = 0;
   TxdDaCunt = 0;
   TxdBuffer[TxdDaCunt++] = ZCAN;
   for(i=0; i<200; i++){
            ChekSum += i;
           switch(i){
                   TxdBuffer[TxdDaCunt++] =  ZCAN;
```

```
                    TxdBuffer[TxdDaCunt++] = i ^ 0x40;
                    break;
                    defulit :
                    TxdBuffer[TxdDaCunt++] = i;
                    break;
            }       }
    TxdBuffer[TxdDaCunt++] = ZEND;
    TxdBuffer[TxdDaCunt++] = ChekSum
    StartTxd();
}

void Que_mode(int mode){
        QueMode = mode;
        if (QueMode == 1) que = &EQue;
        else              que = &NQue;
}

void Que_write(char *buff, int len){
        Que_hd_move(len);
        hd.pos = (que->tail + len + sizeof(hd));
        if (hd.pos >= que->size) hd.pos %= que->size;
        hd.len = len + sizeof(hd.date);
        Que_ram_write(que->tail, (char *)&hd, sizeof(hd));
        Que_ram_write((que->tail + sizeof(hd)) % que->size, buff, len);
        que->tail = hd.pos;
        que->recs++;
}

long Que_recpos(long rec){
        long apos;
        long rpos;
        long i;
        apos = que->head;
        for (i=1; i<= rec; i++) {
                Que_ram_read(apos, (char *)&hd, sizeof(hd));
                rpos = apos + hd.len + sizeof(hd) - sizeof(long);
                if (rpos >= que->size) rpos %= que->size;
                if (hd.pos != rpos) {
                        printf("\nRD Error %d/%d/%d", apos, hd.pos, hd.len)
                        return -1;
                }
                apos = rpos;
        }
        return(apos);
}

void Que_hd_move(long len){
        long apos;
        len += sizeof(QHD);
        while (Que_freelen() <= len) {
                apos = Que_recpos(1);
                if ((apos < 0) || (que->recs <= 0)) {
                        /* que error */
                        que->head = que->tail;
                        que->recs = 0;
                        break;
                }
                que->head = apos;
                que->recs--;
        }   }

void Que_ram_write(long pos, char *buff, int len){
```

```
        long apos;
        apos = que->size - pos;
        if (apos >= len) {
                memmove(&QueBuff[pos], buff, len);
        } else {
                memmove(&QueBuff[pos], buff, apos);
            memmove(QueBuff, &buff[apos], len - apos);
        }    }

char *ram_gets(long addr, char *buff, int len){
        char *pbuff;
        int  i;
        pbuff = buff;
        ram_addr(addr);
        for (i = 0; i < len; i++) {
                *(pbuff++) = *(char *) DRAM1;
                if ((addr & 0xff) == 0xff) ram_addr(++addr)
                else *(char *) RADR1 = ++addr;
        }
        return(buff);
}

void main(void){
        Que_mode(0);
        printf("\nQHD Size : %d", sizeof(QHD));
        for (i=0; i < 32; i++) buff[i] = i + 0x41;
        for (i=0; i < 1000; i++) {
                for(j = 1; j < 26; j++) {
                        Que_write(buff, j);
                }
                printf("\n Count : %d:%d", i, que->recs);
        }
        Que_recpos(que->recs-1);
        printf("\n Count : %d", que->recs);
        for(i=0; i < que->recs; i++) {
                j = Que_read(i, rbuff);
                rbuff[j] = 0;
        }    }

void TTelApp(void){
   int Aval, i;
   WORD Ddata;
   BYTE Bval;
   MoRxdFullFlg = 0;
   for(i=0; i<250; i++){
                Aval = PopModem2RxdData();
                if(Aval < 0) break;
                Bval = PopModem2RxdData();
                if(Bval < 0) break;
   }    }

void TelApp(void){
   BYTE i;
   int Aval;
   MoRxdFullFlg = 0;
   while(1){
          if(0 > eksplanation()) break; /* 받은데이타 해석 */
          if(0 > HedExption()) break;
          if(CpuIdOk == 0) break;
          if(StartNormal) NormalDataPop();
          else if(CodeComand >= 0x30) recv_answer();
          break;
```

```
    }
    for(i=0; i<250; i++){
                        Aval = PopModem2RxdData();
                        if(Aval < 0) break;
    }     }

int HedExption(void){
    BYTE i;
    int Aval = 0;
    switch(ARdata[0]){  /* eksplanation command TYPE */
              case 0:
                        if(CpuId == ARdata[3]){
                                  CpuIdOk = 1;
                                  ioport1 = 0xff;
                         } else  CpuIdOk = 0, ioport1 = 0;
                         for(Aval=0; Aval<100; Aval++){
                                  for(i=0; i< 200; i++);
                         }
                         if(CpuIdOk == 1) ZRINIT_answer();
              break;
              case 3: DataSendWork = 3;
              break;
              case 6: DataSendWork = 6;
              break;
              case 8: ModeRun = 0;
                        CodeComand = 0;
                        StartNormal = 0;
                        DataSendWork = 0;
              break;
              case 18: CodeComand = ARdata[4];
                         StartNormal = 0;
                         if(ARdata[3] == 1){
                        ZDLEDelet_data(); /* 데이타 ZDLE 삭제 */
                        if(crc_32 != 0xDEBB20E3){
                            Aval = -1;
                            CodeComand = 0;
                            ZNACK_answer(); /* reciver 헤더 erro */
                        }       }
              break;
    }
    return(Aval);
}

void HeadrSendData(void){
    int Cval;
    BYTE i, Aval, Bval;
    PushModem2Txd(ZPAD);
    PushModem2Txd(ZDLE);
    PushModem2Txd(ZBIN32);
    crc_32 = 0xffffffff;
    for (i=0; i <= 4; i++) {
              Aval = Bval = ModemHder[i];
              crc_32value = Aval;
              CRC32();
              Bval= ZDLEInsert_data(Aval);
              if(Bval != Aval){
                        PushModem2Txd(ZDLE);
                        PushModem2Txd(Bval);
              } else  PushModem2Txd(Bval);
    }
    crc_32 = ~crc_32;
              for (i=0; i < 4; i++) {
```

```
                Aval = Bval = crc_32 & 0xff;
                Bval= ZDLEInsert_data(Aval);
                if(Bval != Aval){
                        PushModem2Txd(ZDLE);
                        PushModem2Txd(Bval);
                } else  PushModem2Txd(Bval);
                crc_32 = crc_32 >> 8;
    }
    while(1){
            Cval = PopModem2Txd();
            if(Cval < 0) break;
            Modem2ByteSend(Cval);
    }    }
BYTE ZDLEInsert_data(BYTE Fdata){
            switch(Fdata){
                        case ZDLE :
                                Fdata = Fdata ^ 0x40;
                                break;
                        case 0x93 :
                                Fdata = Fdata ^ 0x40;
                                break;
            }
            return(Fdata);
}
int Hcheck_crc(void){
    int Aval;
    WORD k=0;
    crc_32 = 0xffffffff;
    while(1) {
                Aval = PopModem2RxdData();
                if(Aval < 0) break;
                if(Aval == ZDLE) Aval = PopModem2RxdData() ^ 0x40
                if(k < 5) ARdata[k] = Aval;
                k++;
                crc_32value = Aval;
                CRC32();
                if(k > 8) break;
    }
    if(crc_32 == 0xDEBB20E3) return(5);
    else                        return(-1);
}

void RealDataSend(void){
     int i;
     BYTE Aval, Bval;
     crc_32 = 0xffffffff;
     for(i=0; i<6; i++){
                crc_32value = Aval = year_data[i];
                CRC32();
                Bval = ZDLEInsert_data(Aval);
                if(Bval != Aval) PushModem2Txd(ZDLE);
                PushModem2Txd(Bval);
     }
     for(i=0; i<16; i++){
                Aval = LuAiData[i];
                crc_32value = Aval;
                CRC32();
                Bval = ZDLEInsert_data(Aval);
                if(Bval != Aval) PushModem2Txd(ZDLE);
                PushModem2Txd(Bval);
                Aval = LuAiData[i] >> 8;
                crc_32value = Aval;
```

```
            CRC32();
            Bval = ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE);
            PushModem2Txd(Bval);
    }
    for(i=0; i<24; i++){
            Aval = LuDiData[i];
            crc_32value = Aval;
            CRC32();
            Bval = ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE);
            PushModem2Txd(Bval);
    }
    for(i=0; i<12; i++){
            Aval = LuDoData[i];
            crc_32value = Aval;
            CRC32();
            Bval = ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE);
            PushModem2Txd(Bval);
    }
    PushModem2Txd(0x18);
    PushModem2Txd('h');
    crc_32value = 'h';
    CRC32();
    crc_32 = ~crc_32;
    for (i=0; i < 4; i++) {
            Aval = Bval = crc_32 & 0xff;
            Bval= ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE)
            PushModem2Txd(Bval);
            crc_32 = crc_32 >> 8;
  }
   while(1){
          i = PopModem2Txd();
          if(i < 0) break;
          Modem2ByteSend(i);
    }     }

void DateSend(void){
    int i;
    BYTE Aval, Bval;
    crc_32 = 0xffffffff;
    for(i=0; i<6; i++){
            crc_32value = Aval = year_data[i];
            CRC32();
            Bval = ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE);
            PushModem2Txd(Bval);
    }
    PushModem2Txd(0x18);
    PushModem2Txd('h');
    crc_32value = 'h';
    CRC32();
    crc_32 = ~crc_32;
    for (i=0; i < 4; i++) {
            Aval = Bval = crc_32 & 0xff;
            Bval= ZDLEInsert_data(Aval);
            if(Bval != Aval) PushModem2Txd(ZDLE)
            PushModem2Txd(Bval);
            crc_32 = crc_32 >> 8;
  }
```

```
  while(1){
          i = PopModem2Txd();
          if(i < 0) break;
          Modem2ByteSend(i);
  }   }

void ChaingSub(void){
  switch(DAbuff[0]){
  }    }

void SendDoDi(void){
  BYTE  Aval, Bval;
  int i;
  crc_32 = 0xffffffff;
  for(i=0; i<24; i++){
          Aval = LuDiData[i];
          crc_32value = Aval;
          CRC32();
          Bval = ZDLEInsert_data(Aval);
          if(Bval != Aval) PushModem2Txd(ZDLE);
          PushModem2Txd(Bval);
  }
  for(i=0; i<12; i++){
          Aval = LuDoData[i];
          crc_32value = Aval;
          CRC32();
          Bval = ZDLEInsert_data(Aval);
          if(Bval != Aval) PushModem2Txd(ZDLE);
          PushModem2Txd(Bval);
  }
  PushModem2Txd(0x18);
  PushModem2Txd('h');
  crc_32value = 'h';
  CRC32();
  crc_32 = ~crc_32;
  for (i=0; i < 4; i++) {
          Aval = Bval = crc_32 & 0xff;
          Bval= ZDLEInsert_data(Aval);
          if(Bval != Aval) PushModem2Txd(ZDLE)
          PushModem2Txd(Bval);
          crc_32 = crc_32 >> 8;
  }
  while(1){
          i = PopModem2Txd();
          if(i < 0) break;
          Modem2ByteSend(i);
  }    }

void SetHardConSave(){
  BYTE *LEepAdr, i;
  LEepAdr = (BYTE*) EEP_HCF_ADR;
  if(DAbuffpos > 200) return;
  for(i=0; i < DAbuffpos; i++){
          *(LEepAdr + i) = DAbuff[i];
          TimeMsec = 0;
          while(TimeMsec < 8);
  }    }

void PopSetHardCon(void){
  BYTE *LEepAdr, Aval, Bval;
  int i;
  LEepAdr = (BYTE*) EEP_HCF_ADR;
```

```
    ZCOMMAND_answer(1, 0x69); /* flag, command */
    }
    PushModem2Txd(0x18);
    PushModem2Txd('h');
    crc_32value = 'h';
    CRC32();
    crc_32 = ~crc_32;
    }
   while(1){
            i = PopModem2Txd();
            if(i < 0) break;
            Modem2ByteSend(i);
    }
}

void recv_answer(void) {
    long Aval, Bval;
    BYTE i;
 }

void NormalPopSub(void){
            BYTE i;
            BYTE Aval, Bval;
             }    }

void NormalDataPop(void){
    if(DataSendWork == 3) StartNormal++;
    if(StartNormal == 1) ZCOMMAND_answer(1, 0x41);
}

void PopLUSet(void){
    BYTE *LEepAdr, Aval, Bval;
    int i;
     }

void RtuLUSet(void){
    BYTE *LEepAdr, i;
    LEepAdr = (BYTE*) EEP_CF_ADR;
            for(i=0; i < MAX_CF_ADR; i++){
             *(LEepAdr+i) = DAbuff[i];
             TimeMsec = 0;
             while(TimeMsec < 8);
    }    }

void ZRINIT_answer(void){
  WORD Aval;
}

void ZACK_answer(BYTE code, WORD offset){
}

void ZCOMMAND_answer(BYTE Dflag, BYTE Code){
}
void main(void){
    int SenSorValue[22], ActionCmpTime[22];
                         ActionFlag = HeatRun(SysFlag, SenChkVal)
            break;
    }
}
```

C-2.. Analog Input 모듈 프로그램

```
void StartInit(void){
        TxdRxdSelt = RxdSelt;
}

void Delay(int DlyData){
        int  i, b;
        for(i=0; i < DlyData; i++){
            for(b=0; b < DlyData; b++);
        }
}

void TimerSrlInit(void){
}

void ChSetAi(void){
        AiCh++;
        if(AiCh > 7) AiCh = 0;
        switch(AiCh){
        }
}

void TC5092Init(void){
        AiCh = 0;
        ChSetAi();
        StartUpConve();
}

int Read5092(void)      {
        unsigned int Aval, Bval;
}

void AdcData(void){
        int Aval;
        Aval = Read5092();
        SensorData[AiCh] = Aval;
        ChSetAi();
        StartUpConve();
}

interrupt [0x0B] void T0_int (void){
}

void main(void){
Loop :
   if(EOC == 1) AdCovBit1 = 0, AdcData();
   if(AdCovBit) AdCovBit = 0, StartUpConve();
   if(RxdDataFull){
            SrlDataChk();
            RxdDataFull = 0;
            RxdEndFlg2 = 0;
            RxdEndFlg1 = 0;
   }
goto Loop;
}

interrupt [0x23] void SCON_int (void){
        BYTE i;
}
```

```
int PopRxdData(void){
   int Aval = -1;
   if(RxdStartPoss != RxdEndPoss){
            Aval = RxdData[RxdStartPoss++];
            RxdStartPoss %= 32;
   } else {
            RxdDataFull = 0;
            RxdEndFlg2 = 0;
            RxdEndFlg1 = 0;
   }
   return(Aval);
}

void SendDataTest(void){
   if(AiAdrLow != Aval) srl_send(ZCAN);
   srl_send(AiAdrLow);
   ChkSum += AiAdrLow;
   srl_send(AiInDataCom);
   ChkSum += AiInDataCom;
   }
   srl_send(ZEND);
   srl_send(ChkSum);
   srl_send(ChkSum);
   srl_send(ChkSum);
   TxdRxdSelt = RxdSelt;
}

void SrlDataChk(void){
   int Aval;
   while(1){
}
```

C-3. Digital Input 모듈 프로그램

```
void StartInit(void){
   TxdRxdSelt  = RxdSelt;
   RxdDataFull = 0;
   SendTimOver = 0;
   RxdStartPoss = RxdEndPoss = 0;
   ChkRcvAdrOk = 0;
   RxdEndFlg1 = RxdEndFlg2 = 0;
}

void TimSerIInt(void) {
 }
interrupt[0x0B] void T0_int (void)
 {
   TR0 = 0;
   ++usec;
}

void SrlSend(BYTE SData){
   SBUF = SData;
   Delay(20);
   TI = 0;
}
```

```
interrupt [0x23] void SCON_int (void){
          BYTE i;
          EA =0;
}

int PopRxdData(void){
   int Aval = -1;
   if(RxdStartPoss != RxdEndPoss){
             Aval = RxdData[RxdStartPoss++];
             RxdStartPoss %= 32;
   } else {
             RxdDataFull = 0;
             RxdEndFlg2 = 0;
             RxdEndFlg1 = 0;
   }
   return(Aval);
}

void SendToMain(void){
          BYTE Bval, Aval;
}

void SendTest(void){
          BYTE Bval;
}

void main(void){
   StartInit();
   TimSerlInt();
loop:
   if(SendTimOver){
             /* SendTest(); */
             SendTimOver = 0;
   }
   if(RxdDataFull){
             SrlDataChk();
             RxdEndFlg2 = 0;
             RxdEndFlg1 = 0;
   }
goto loop; }
```

C-4. Digital Output 모듈 프로그램

```
void StartInit(void){
   P1 = 0;
   TxdRxdSelt  = RxdSelt;
   RxdDataFull = 0;
   SendTimOver = 0;
   RxdStartPoss = RxdEndPoss = 0;
   ChkRcvAdrOk = 0;
   RxdEndFlg1 = RxdEndFlg2 = 0; }
void TimSerlInt(void){
 }

interrupt [0x23] void SCON_int (void){
          BYTE i;
```

```
        EA =0;
}

int PopRxdData(void){
   int Aval = -1;
   if(RxdStartPoss != RxdEndPoss){
            Aval = RxdData[RxdStartPoss++];
            RxdStartPoss %= 32;
   } else {
            RxdDataFull = 0;
            RxdEndFlg2 = 0;
            RxdEndFlg1 = 0;
   }
   return(Aval);
}

void SendToMain(void){
}

void SendTest(void){
}

void main(void){
   StartInit();
   TimSerlInt();
loop:
   if(SendTimOver){
            SendTimOver = 0;    }
   if(RxdDataFull){
            SrlDataChk();
            RxdDataFull = 0;
            RxdEndFlg2 = 0;
            RxdEndFlg1 = 0;    }
   P1 = DoData;
goto loop;
}
```

C-5. 외기환경 입력 모듈 프로그램

```
void StartInit(void){
          char i;
          RxdStartPoss = 0;     RxdEndPoss = 0;       RxdFull = 0;
          Tsec = 0;
          for(i=0; i<8; i++) SensorData[i] = 0;
          AiAdrLow = 3;          AiAdrLow &= 0x3f;     TxdRxdSelt = RxdSelt
          WindSpCunt = RainCunt = 0;     usec = msec = RealSec = 0;
}

void TimerSrlInit(void){
}

interrupt[0x13] void ext1_int (void){
          EX1 = 0;
          ExFlag2 = 0;
          RainCunt++;
          SensorData[1]++;
```

```
            if(SensorData[1] > 30000) SensorData[1] = 0
}

void main(void){
            StartInit();
            TimerSrlInit();
Loop :
    if(RxdDataFull){
                SrlDataChk();
                RxdDataFull = 0;
                RxdEndFlg2 = 0;
                RxdEndFlg1 = 0;
    }
goto Loop;
}

interrupt [0x23] void SCON_int (void){
            BYTE i;
            EA =0;
}

int PopRxdData(void){
    int Aval = -1;
    if(RxdStartPoss != RxdEndPoss){
                Aval = RxdData[RxdStartPoss++];
                RxdStartPoss %= 32;
    } else {
                RxdDataFull = 0;
                RxdEndFlg2 = 0;
                RxdEndFlg1 = 0;
    }
    return(Aval);
}

void SendDataTest(void){
    BYTE i, Aval;
    TxdRxdSelt = TxdSelt;
    srl_send('*');
    srl_send(ZCAN);
    srl_send(AiAdrHigh);
    ChkSum = AiAdrHigh;
    Aval= AddInsertZdle(AiAdrLow);
    if(AiAdrLow != Aval) srl_send(ZCAN);
    srl_send(AiAdrLow);
    ChkSum += AiAdrLow;
    srl_send(AiInDataCom);
    ChkSum += AiInDataCom;
}

void SrlDataChk(void){
    int Aval;
    while(1){
                Aval = PopRxdData();
                if(Aval < 0) break;
```

C-6. 80c196 보드의 회로도 map

```
module hmain1 flag '-r2'
title '
          TITLE     : hmain1
          ABSTRACT : decoder for house board
    AUTHOR(S): park, hung-pyo
          DATE      : 1996. 04. 15
          tuin indestrial company'
HMAIN1    device  'p16v8s';
a2,a3,a4,a11,a12,a13,a14,a15     pin      7,6,5,11,4,3,2,1;
rd,wr                                pin      8,9;
rom,ram                              pin      19,18;
ram1,ram2                            pin      17,16;
epr                                  pin      15;
rtl                                  pin      14;
h,l,x    =    1,0,.x.;
address = [a15,a14,a13,a12, a11,x,x,x, x,x,x,a4, a3,a2,x,x];
end hmain1

ABEL(tm) Version 2.10a  -  Document Generator        18-Sep-96 01:21 PM
        TITLE     : hmain1
        ABSTRACT : decoder for house board
    AUTHOR(S): park, hung-pyo
        DATE      : 1996. 04. 15
        tuin indestrial company
Equations for Module hmain1
Device HMAIN1
- Reduced Equations:

ABEL(tm) Version 2.10a  -  Document Generator        18-Sep-96 01:21 PM
        TITLE     : hmain1
        ABSTRACT : decoder for house board
    AUTHOR(S): park, hung-pyo
        DATE      : 1996. 04. 15
        tuin indestrial company
end of module hmain1

device G16V8 converted from 16V8
PALtoGAL Version 2.5 Copyright 1987 1988 1989 Lattice Semiconductor Corporation
PALtoGAL input file: HMAIN1.JED
ABEL(tm) Version 2.10c FutureNet/Data-IO Corp.  JEDEC file for: P16V8S
Created on: 18-Sep-96 01:21 PM
        TITLE     : hmain1
        ABSTRACT : decoder for house board
    AUTHOR(S): park, hung-pyo
        DATE      : 1996. 04. 15
        tuin indestrial company
*C2231
*9FCA
```

D. CRC32 Table

No.	Value	No.	Value	No.	Value	No.	Value
1	0x00000000	2	0x77073096	3	0xee0e612c	4	0x990951ba
5	0x076dc419	6	0x706af48f	7	0xe963a535	8	0x9e6495a3
9	0x0edb8832	10	0x79dcb8a4	11	0xe0d5e91e	12	0x97d2d988
13	0x09b64c2b	14	0x7eb17cbd	15	0xe7b82d07	16	0x90bf1d91
17	0x1db71064	18	0x6ab020f2	19	0xf3b97148	20	0x84be41de
21	0x1adad47d	22	0x6ddde4eb	23	0xf4d4b551	24	0x83d385c7
25	0x136c9856	26	0x646ba8c0	27	0xfd62f97a	28	0x8a65c9ec
29	0x14015c4f	30	0x63066cd9	31	0xfa0f3d63	32	0x8d080df5
33	0x3b6e20c8	34	0x4c69105e	35	0xd56041e4	36	0xa2677172
37	0x3c03e4d1	38	0x4b04d447	39	0xd20d85fd	40	0xa50ab56b
41	0x35b5a8fa	42	0x42b2986c	43	0xdbbbc9d6	44	0xacbcf940
45	0x32d86ce3	46	0x45df5c75	47	0xdcd60dcf	48	0xabd13d59
49	0x26d930ac	50	0x51de003a	51	0xc8d75180	52	0xbfd06116
53	0x21b4f4b5	54	0x56b3c423	55	0xcfba9599	56	0xb8bda50f
57	0x2802b89e	58	0x5f058808	59	0xc60cd9b2	60	0xb10be924
61	0x2f6f7c87	62	0x58684c11	63	0xc1611dab	64	0xb6662d3d
65	0x76dc4190	66	0x01db7106	67	0x98d220bc	68	0xefd5102a
69	0x71b18589	70	0x06b6b51f	71	0x9fbfe4a5	72	0xe8b8d433
73	0x7807c9a2	74	0x0f00f934	75	0x9609a88e	76	0xe10e9818
77	0x7f6a0dbb	78	0x086d3d2d	79	0x91646c97	80	0xe6635c01
81	0x6b6b51f4	82	0x1c6c6162	83	0x856530d8	84	0xf262004e
85	0x6c0695ed	86	0x1b01a57b	87	0x8208f4c1	88	0xf50fc457
89	0x65b0d9c6	90	0x12b7e950	91	0x8bbeb8ea	92	0xfcb9887c
93	0x62dd1ddf	94	0x15da2d49	95	0x8cd37cf3	96	0xfbd44c65
97	0x4db26158	98	0x3ab551ce	99	0xa3bc0074	100	0xd4bb30e2
101	0x4adfa541	102	0x3dd895d7	103	0xa4d1c46d	104	0xd3d6f4fb
105	0x4369e96a	106	0x346ed9fc	107	0xad678846	108	0xda60b8d0
109	0x44042d73	110	0x33031de5	111	0xaa0a4c5f	112	0xdd0d7cc9
113	0x5005713c	114	0x270241aa	115	0xbe0b1010	116	0xc90c2086
117	0x5768b525	118	0x206f85b3	119	0xb966d409	120	0xce61e49f
121	0x5edef90e	122	0x29d9c998	123	0xb0d09822	124	0xc7d7a8b4
125	0x59b33d17	126	0x2eb40d81	127	0xb7bd5c3b	128	0xc0ba6cad
129	0xedb88320	130	0x9abfb3b6	131	0x03b6e20c	132	0x74b1d29a
133	0xead54739	134	0x9dd277af	135	0x04db2615	136	0x73dc1683
137	0xe3630b12	138	0x94643b84	139	0x0d6d6a3e	140	0x7a6a5aa8
141	0xe40ecf0b	142	0x9309ff9d	143	0x0a00ae27	144	0x7d079eb1

No.	Value	No.	Value	No.	Value	No.	Value
145	0xf00f9344	146	0x8708a3d2	147	0x1e01f268	148	0x6906c2fe
149	0xf762575d	150	0x806567cb	151	0x196c3671	152	0x6e6b06e7
153	0xfed41b76	154	0x89d32be0	155	0x10da7a5a	156	0x67dd4acc
157	0xf9b9df6f	158	0x8ebeeff9	159	0x17b7be43	160	0x60b08ed5
161	0xd6d6a3e8	162	0xa1d1937e	163	0x38d8c2c4	164	0x4fdff252
165	0xd1bb67f1	166	0xa6bc5767	167	0x3fb506dd	168	0x48b2364b
169	0xd80d2bda	170	0xaf0a1b4c	171	0x36034af6	172	0x41047a60
173	0xdf60efc3	174	0xa867df55	175	0x316e8eef	176	0x4669be79
177	0xcb61b38c	178	0xbc66831a	179	0x256fd2a0	180	0x5268e236
181	0xcc0c7795	182	0xbb0b4703	183	0x220216b9	184	0x5505262f
185	0xc5ba3bbe	186	0xb2bd0b28	187	0x2bb45a92	188	0x5cb36a04
189	0xc2d7ffa7	190	0xb5d0cf31	191	0x2cd99e8b	192	0x5bdeae1d
193	0x9b64c2b0	194	0xec63f226	195	0x756aa39c	196	0x026d930a
197	0x9c0906a9	198	0xeb0e363f	199	0x72076785	200	0x05005713
201	0x95bf4a82	202	0xe2b87a14	203	0x7bb12bae	204	0x0cb61b38
205	0x92d28e9b	206	0xe5d5be0d	207	0x7cdcefb7	208	0x0bdbdf21
209	0x86d3d2d4	210	0xf1d4e242	211	0x68ddb3f8	212	0x1fda836e
213	0x81be16cd	214	0xf6b9265b	215	0x6fb077e1	216	0x18b74777
217	0x88085ae6	218	0xff0f6a70	219	0x66063bca	220	0x11010b5c
221	0x8f659eff	222	0xf862ae69	223	0x616bffd3	224	0x166ccf45
225	0xa00ae278	226	0xd70dd2ee	227	0x4e048354	228	0x3903b3c2
229	0xa7672661	230	0xd06016f7	231	0x4969474d	232	0x3e6e77db
233	0xaed16a4a	234	0xd9d65adc	235	0x40df0b66	236	0x37d83bf0
237	0xa9bcae53	238	0xdebb9ec5	239	0x47b2cf7f	240	0x30b5ffe9
241	0xbdbdf21c	242	0xcabac28a	243	0x53b39330	244	0x24b4a3a6
245	0xbad03605	246	0xcdd70693	247	0x54de5729	248	0x23d967bf
249	0xb3667a2e	250	0xc4614ab8	251	0x5d681b02	252	0x2a6f2b94
253	0xb40bbe37	254	0xc30c8ea1	255	0x5a05df1b	256	0x2d02ef8d

첨부 : 개발 기술의 활용

본 연구에서 개발된 복합환경 제어시스템의 하드웨어와 소프트웨어는 산업체에서 보유한 현장기술과 접목하여 다음과 같이 활용할 수 있다.

1. 기존의 온실 환경제어용 컨트롤러는 환경요인만을 계측하여 제어하지만, 본 연구에서 개발된 컨트롤러는 환경요인 뿐만아니라 생체정보도 수집하여 제어에 활용할 수 있도록 구성되어 있으므로, 적정 생장모델이 도출되면 즉각 환경제어에 활용할 수 있다.
2. 규모면에서 소형화하고, 시설물의 크기, 종류, 적용 온실 수에 따라 시스템을 쉽게 변경하여 사용 가능하도록 개발하였다. 따라서, 본 시스템은 크기가 작고 사용되는 환경에 따라서 Controller의 개수가 조절이 되기 때문에 가격면에서도 상당히 합리적이므로 경제적 보제적 보급이 가능하다.
3. 하드웨어의 각 입출력부분 및 구동 소프트웨어는 MODEM 및 RS-232, RS-485 통신을 통하여 PC에서부터 전송이 가능하도록 구성되어 있으므로 원격제어가 가능하다.
4. 본 연구에서 개발된 복합환경 제어시스템에는 무정전 전원장치를 포함시켜 정전 등에 의하여 전원이 차단될 경우에도 온실의 작동기기를 제어할 수 있다.
5. 본 연구에서 개발된 제어용 소프트웨어는 작물의 생육시 필요로로 하는 환경을 가장 적절히 제공할 수 있도록 구성하였다. 프로그램 내용중에는 여러 외부 환경조건에 따른 제어 로직을 별도로 구성하므로써 온실제어의 효율성을 높혔으며 특히, 커튼의 제어 Logic은 가열이나

감온시에 작동되는 것 이외에 별도로 작물의 광합성과 일사량과의 밀접한 관계를 고려하여 설계하였다. 따라서 본 제어로직을 활용한 복합환경 제어시스템을 구축함으로써, 작물의 생육을 최대한으로 도모할 수 있다.