최종보고서

온라인 컴퓨터 건조제어기

Online Computer Dry Controller

연 구 기 관

안동대학교

농 림 수 산 부

제 출 문

농림수산부 장관 귀하

본 보고서를 “온라인 컴퓨터 건조제어기에 관한 연구”과제의 최종 보고서로 제출합니다.

1996 . 11 . 30

주관연구기관명: 안동대학교

총괄연구책임자: 박 세 현

연 구 원: 이 정 환

: 최 병 탁

요 약 문

I. 제목

온라인 컴퓨터 건조 제어기

II.연구개발 목적 및 중요성

농어촌 현장의 컴퓨터 건조기를 온라인 제어하에 둠으로서 마을 공동단위나 중앙감시체제에서 건조기를 효과적으로 운영하게 하는 한편 유지보수를 온라인체계로 구축 가능하게 하려함.

III.연구개발 내용 및 범위

○ 건조기컴퓨터에 온라인 통신기능을 부과하여 컴퓨터 성능을 향상시킴

○ 건조기의 온라인화를 위한 시스템 설계 구현.

○ 자기 진단 기능의 향상.

○ 건조기의 중앙통제 시스템 설계 및 구현.

○ 건조 프로그램을 농가 선로를 통하여 전송가능하도록 설계 및 구현

○ 중앙 통제실에서 상황이 표시되고 건조 절차를 저장, 가공물에 대한 가공 프로그램을 표시 할수 있는 기능 구현.

○ 건조기 시스템 회로의 최적화와 수리보수에 용이.

IV.연구개발결과 및 활용에 대한 건의

본 연구 개발 결과로 설계 구현된 중앙감시컴퓨터 온라인 통신 소

프트웨어, 중앙감시보드, 전용모뎀 및 전용 프린트는 건조기의 온라인을 위해 대단히 필요한 기술이며 농어촌 마을 공동단위의 건조기를 효과적 운영,수리유지보수는 물론 농어촌 정보화에 도움이 되므로 농가에서 조속한 활용을 위해 지원이 필요함.

Summary

I. Title

On-line computer dry controller

II. Purpose and Importance of the Research

Dryer is more effectively operated when it is monitored by a on-line computer. The on-line computer dryer provides not only an effective way for central surveillance of the dryer but also an on-line maintenance of the controller.

III. Contents and Scope of the Research

- Improvement of system performance by addition of the on-line communication between the computer and the dryer.
- Realization of system for on-line control of the dryer.
- Improvement of self diagnosis of the system.
- Design and realization of central surveillance board.
- Design and realization of transmit of dryer programs through the existing phone line.
- Display of system status on the computer.
- Display the function of a dry program among several different dry menu.
- Realization of simplification of system and easy maintenance.

IV. Research Results and Suggestion for the System Application

As results of this research, it was found that the on-line communication software of a central surveillance computer, a central surveillance board, a dedicated modem, and a dedicated printer are essential for the on-line control dryer. The system using these results reduces labor a lot, makes maintenance easy, and help automation of dry processes of agricultural products. To apply of this system to farms, support from the government is required for farmers to adopt the system at a reasonable cost.

Contents

목 차

제 1 장 연구개발의 목적과 범위

1.연구 개발 목적

농어촌 현장에서 사용하고 있는 건조기는 주로 수동식 건조기와 자동식 컴퓨터 건조기로 나뉘는데 수동식 건조기는 농산물 건조를 위해 일일이 조작자에 의해 수작업으로 건조를 시키는데 비하여 컴퓨터 건조기는 자동으로 건조의 전과정을 컴퓨터 제어기에 의해 제어되므로 매우 편리하여 요즈음은 주로 컴퓨터 건조제어기를 많이 사용하고 있다. 그러나 컴퓨터 건조기의 경우 원하는 건조조건과 건조기능이 다를 뿐 아니라 건조기의 신뢰도가 낮아 오히려 건조물을 망치는 경우가 흔히 있어 컴퓨터 건조기의 운영을 위해 여분의 인력이 주기적으로 감시해야 할 뿐 아니라 건조기의 순간적 오동작에 대한 서비스체계의 대비가 없어 농어촌의 커다란 문제점이 되고 있다.

본 연구에서는 이러한 문제점을 해소하고자 개개의 건조기가 Network 상에서 온라인으로 건조기를 제어하고자 하였다. 컴퓨터 건조기를 온라인화하게 되면 건조기를 개개의 사용자가 보유함에 따른 운영과 유지, 보수체계의 문제점으로 인한 건조기의 비효율적 운용을 충분히 보완 할 수 있으며 마을 공동단위 혹은 개별단위 건조기를 중앙 감시 체제에서 온라인으로 효과적인 운영을 할 수 있다는데 그 주된 목적을 두며 그 장점은 다음과 같다.

○ 마을단위, 조합 혹은 서비스 센터의 중앙 제어 컴퓨터에 의해 개별 건조기가 중앙통제하게 하여 인력 절감을 유발시킨다.
○ 중앙통제 컴퓨터에 의해 이상유무가 주기적으로 점검되어 이상 징후

가 나타나면 온라인으로 서비스 센터에 바로 연결되여 빠른 서비스 체계를 구축한다.

○ 미숙한 사용자의 실수로 인한 건조물의 손실을 방지하여 건조물의 신뢰를 높인다.

따라서 건조기의 온라인화는 농.어촌 경쟁력을 높이기 위한 자동화, 정보화를 그 목적으로 한다.

2.연구 개발의 범위

연구 개발은 95년도의 1차 개발과 96년도의 2차로 나누어 1차년도에는 주로 건조기의 온라인화를 위한 하드웨어의 개발을 하고 2차년도에는 1차년도의 하드웨어에 따른 소프트웨어의 개발을 주로 하였다.

가. 1차년도 연구개발

○ 자료조사및 하드웨어의 분석 및 하드웨어 설계

- 세부 계획 수립
- 문헌 및 자료 조사
- 기존 제품의 성능 파악
- 하드웨어 분석

○ 온라인 통신기능이 부과된 시스템 설계 및 구현 온라인 컴퓨터 건조기 시스템 구현

- 시스템의 시험
- 기존 건조기의 온라인화 제어를 위한 표준화된 범용 통신 인터

페이스 하드웨어 개발

- 건조기 전용 모뎀 개발

- 건조기 전용 프린터 개발(시스템 개발시 꼭 필요한 기능이어서 계획상에 없었으나 추가 하였음)

- 마을 공동 단위의 중앙감시보드 개발

나. 2차년도 연구개발

○ 온라인 통신 소프트웨어의 개발

- 중앙 감시컴퓨터의 온라인 통신을 위한 통신 소프트웨어의 개발

- 중앙 감시보드의 온라인 통신 소프트웨어 개발

- 건조기 전용 모뎀의 소프트웨어 개발

건조기 감시 기능, 온라인 무선 호출기능,

중앙감시 컴퓨터 연결 기능

- 건조기 전용 프린터의 소프트웨어 (추가)

건조기 감시 기능, 건조기 결함 하드카피 기능

○ 온라인 통신기능이 부과된 시스템의 종합적 시험

- 온라인 통신기능이 부과된 시스템의 종합적인 시험 및 수정,보완

제 2 장 연구개발 내용.

제 1 절 연구개발 내용.

컴퓨터 건조기의 network상에 온라인 제어를 위해서 크게 2 가지방식의 온라인 건조제어 방식을 구현하였는데 첫째는 건조기의 개별 단위(개별 농가)에서 건조기가 온라인 제어될 수 있게 구성하는 방식이고 , 둘째는 마을 공동단위로 운영을 가능하게 하고 이를 온라인 제어할 수있게 하는 방식이다. 이러한 두가지 제어 방식을 구현하기 위해서는 기존의 컴퓨터 건조기에 간단히 온라인 통신 기능을 부과하고 이를 제어 하여야 하는데 기존 건조기의 온라인화 제어를 위한 표준화된 범용 통신 인터페이스 하드웨어 개발을 하였고(1차년도 보고서 참조), 이러한 표준화된 인터페이스로 건조기를 온라인 제어할 수 있게 IBM PC 상에 운용되는 중앙감시 컴퓨터의 온라인 통신 소프트웨어를 개발하였다. 이 프로그램은 처음 C언어로 구현하였으나 차후의 수리 유지 보수를 위해 비주얼 BASIC으로 구현 하였다. 이 온라인 통신 프로그램(편의상 아리랑으로 명명 하였음)으로 건조기 전용 모템 장비와 건조기 전용 프린터를 접속할 수 있고 마을 공동단위를 위한 중앙감시 보드에도 연결 가능하다.

한편 자기진단 기능과 원거리 통신 기능을 부과하여 건조기의 진단과 감시가 가능하고 원거리 제어를 할 수 있는 건조기 전용 모뎀의 하드웨어(1차년도 보고서 참조)와 소프트웨어를 개발하였다. 특히 건조기의 주기적인 감시와 건조물의 제어를 위해서 계획상에 없었지만 건조기에 직접 인가하여 건조기의 자료를 주기적으로 출력시킬수 있는 프린트가 필요하여 건조기 전용 프린터의 하드웨어(1차년도 보고서 참조)와 소프트웨어를 개발하였다. 또한 현재 컴퓨터건조기가 단독으로 운영되는 것보다 마을

공동단위로 운영되는 것이 필요한 추세이여서 여러대의 개별농가의 건조기들을 통합 관리하는 것이 필요하게 되어 개별컴퓨터 건조기를 통합 관리할 수 있는 마을 공동 단위의 중앙감시보드의 하드웨어(1차년도 보고서 참조)와 온라인 통신소프트웨어를 개발하였다. 이는 개별 단위의 건조방식이 운영의 효율이 낮은데 비하여 공동 건조함으로써 건조기 운용의 효율을 높일수 있고 개별 사용자의 조작상 실수나 시스템의 이상을 공동으로 대치 할 수 있는 장점이 있을 뿐 아니라 노동 인력 절감이라는 차원에서도 효과적이다.

본 연구에서 위에 개발된 여러 하드웨어와 소프트웨어를 컴퓨터 건조기의 network상에 온라인 제어를 위해서 위에서 언급한 바와 같이 2가지 방식의 온라인 건조제어 방식을 구현하였는데 이를 자세히 설명하면 다음과 같다.

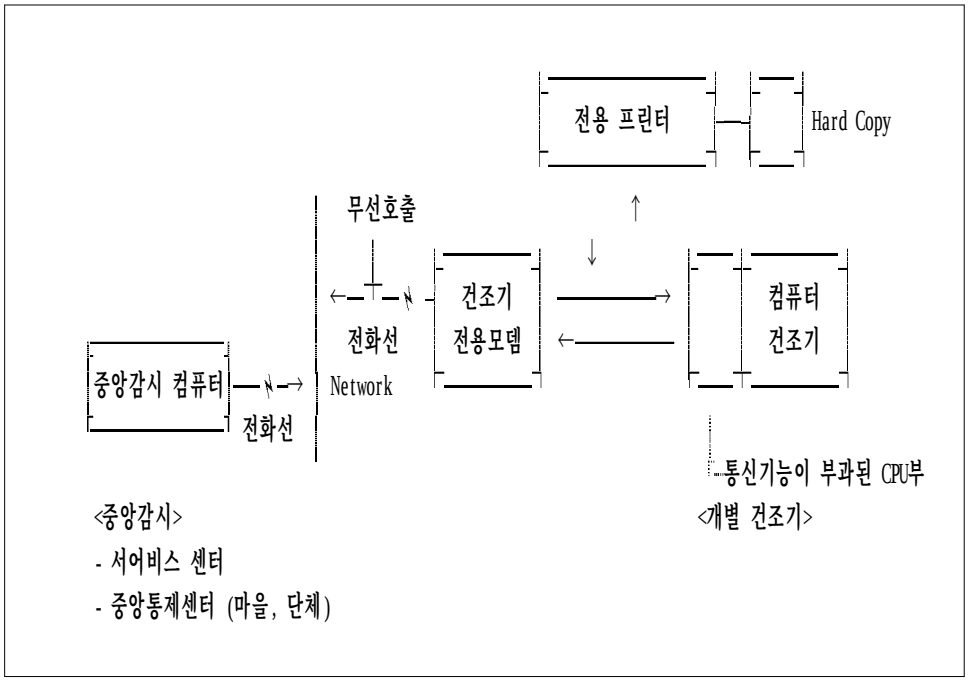

그림1. 개별(개인) 단위에 있어서 온라인 컴퓨터 건조제어기 구성

1. 개별단위에 있어서 온라인 컴퓨터 제어기 설계

개별(개인)단위에 있어서 연구 개발된 온라인 컴퓨터 건조제어기의 구성도는 그림 1과 같다. 개별(개인)단위의 건조제어방식은 단일컴퓨터 건조기의 온라인 방식인데 기존의 컴퓨터 건조기에 기능을 향상시켜 통신 기능을 부가하고 컴퓨터 건조기의 이상 유무를 이 온라인 통신 Port로 온라인상에서 점검한다. 만약 이상이 발생하였을시 전화선로를 통하여 자동적으로 서비스센터나 사용자의 무선호출기에 이상을 알릴 수 있게 한다. 이때 사용자는 건조기의 이상 상태를 자신의 무선호출기를 통하여 알게 되고 서비스 센터는 무선호출이나 혹은 중앙통제 컴퓨터로 하여금 건조기의 이상에 대한 상태를 온라인으로 점검하고 기록하여 서비스 체제에 돌입할 수 있게 된다. 또한 중앙 통제 컴퓨터(IBM PC)는 개별 건조기의 건조 상태를 건조 단계별로 주기적으로 기록하거나 점검할 수 있다. 개별 컴퓨터 건조기에 부착되어 있는 전용 프린터는 개별 사용자가 자신의 건조물에 대한 온.습도 기록이나 건조기의 이상 유무를 프린터 할 수 있으므로 자신의 건조기와 건조물에 대한 상태를 분석, 건조 프로그램 개선등에 이용할 수 있는 자료로 활용 가능하다.

이러한 건조기의 온 라인용 전용모뎀의 이용은 사용자가 건조기와 컴퓨터만 있으면 건조기의 작동 상태를 지역에 관계없이 온라인으로 원격 점검할 수 있으므로 매우 유용하며 건조물에 대한 신뢰성을 높여 준다. 여기서 연구개발된 건조기 전용모뎀은 일반모뎀과 그 구성이 틀린데 일반모뎀은 일반 PC용 컴퓨터의 수동적 장비로서 컴퓨터의 직접적인 입.출력 명령에 의해 작동되고 능동적으로 컴퓨터 시스템을 제어하지 않는 반면 건조기 전용모뎀은 건조기의 내부 상황을 능동적으로 제어하여 컴퓨터 건조기의 내부작동 상태를 전용모뎀에 의해 점검하고 이를 컴퓨터 건조기의 도움없이 외부 Network에 알려주는 역할을 하게끔 설계 하였다. 또한 컴

퓨터 건조기가 고장이 나서 시스템이 멈춘 경우에도 건조기의 멈춤 상태를 외부 Network에 알리는 능동적 기능을 가지고 있다. 한편 연구 개발된 전용 프린터도 능동적 기능을 가지게 하였으며 주기적으로 컴퓨터 건조기의 상태를 전용 프린터가 점검하여 프린터로 Hard Copy하여 건조물과 건조상태의 개선을 위한 중요한 자료를 제공할 수 있게 하였다.

2. 공동(마을)단위에 있어서 온라인 컴퓨터 제어기 설계

마을 공동단위에 있어서 연구개발된 온라인 컴퓨터 제어기의 구성은 여러대의 개별농가의 건조기들을 가까운 중앙 감시보드에 온라인으로 연결하여 중앙감시를 하는 것이다. 이는 개별 단위의 건조방식이 운영의 효율이 낮은데 비하여 공동 건조함으로써 건조기 운용의 효율을 높일수 있고 개별 사용자의 조작상 실수나 시스템의 이상을 공동으로 대처할 수 있는 장점이 있을 뿐 아니라 노동 인력 절감이라는 차원에 서도 효과적이

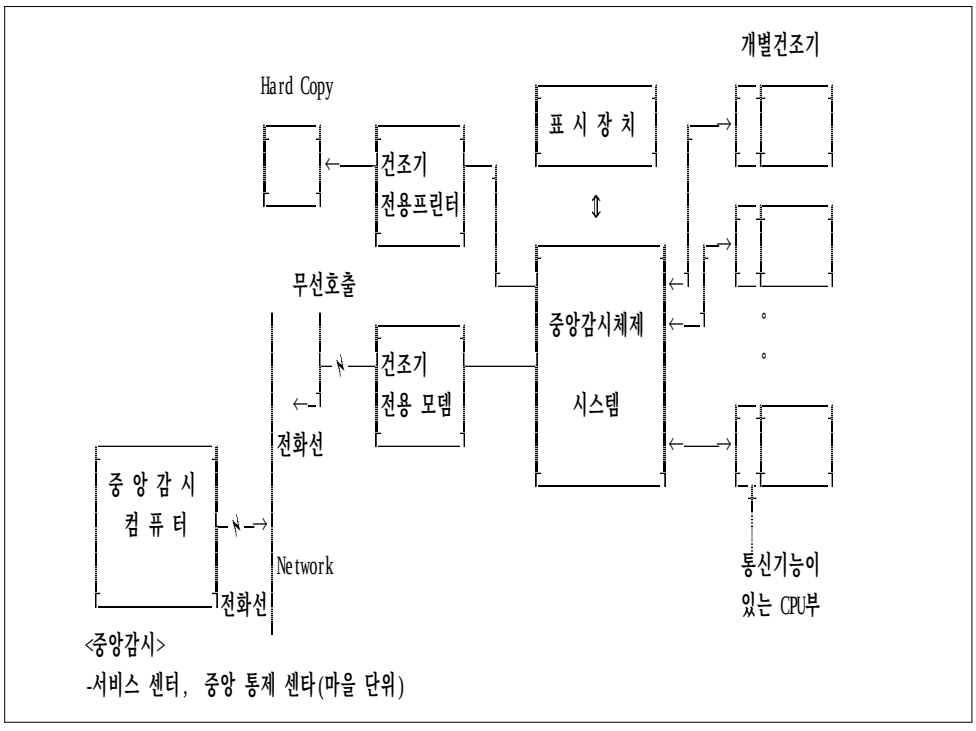

그림2. 공동(마을)단위에 있어서 온라인 컴퓨터 건조제어기 구성

다. 따라서 많은 건조기의 제작자가 중앙 감시 체계의 건조시스템을 지향하고 있으나, 중앙 통제 방식의 온라인 기능이라기 보다는 개별 건조기의 온도센서를 연장하여 중앙 감시표시장치에 나타내게 하는데 불과한 실정이다. 그 중 개량된 방식에서 조차 이러한 표시 장치의 연장을 bus 방식으로 구현할 뿐 온라인 기능은 없다. 특히 이 경우 bus 구조를 가지고 있기 때문에 임의 선로의 장애가 오면 그 뒤 라인의 건조기의 상태를 점검 할 수 없게 되어있다. 본 연구의 공동(마을)단위에 있어서 온라인 건조기의 설계는 이러한 방식을 개량하여 Star방식으로 구현하여 개별 건조기가 자신의 선로로 직접 중앙감시시스템과 연결되어 있어 임의 건조기의 통신장애가 다른 건조기의 통신장애에 영향을 미치지 않게 설계하였으며, 기존의 방식이 단순한 개별건조기의 표시 연장에 불과한 방식에 비하여 본 건조기의 방식에는 진정한 온라인 기능을 부과하였다. 설계 구현된 중앙 제어 보드의 채널(0-15개 채널)에 건조기 전용모뎀, 전용프린터, 개별 건조기 연결을 자유롭게 연결되게하여 통신선로 Interface의 표준화를 이루었으며, 중앙감시보드의 각 채널 상태(개별건조기 상태)가 온라인상으로 중앙통제컴퓨터에 연결 가능하게 하였다. 만약 개별 건조기의 이상이 있을시 중앙 감시 보드는 전용모뎀을 통하여 중앙 통제 컴퓨터나 무선호출기를 통하여 사용자나 서비스 센터에 연결하게 되어있다. 또한 주기적으로 각 채널의 상태를 전용 프린터에 의해 프린터 되게 하는 등 중앙 감시표시장치에 표시되게 설계 구현 하였다.

제 2 절 결과.

1, 2 차 최종 결과는 아래 와 같으며 온라인 컴퓨터 건조제어기의 사업 계획에서는 1차년도 Hardware개발과 2차년도 Software개발로 나누었

으나 1차년도 Hardware개발을 위한 Software 개발도 병행하여야 하므로 2차년도 Software 개발을 앞당겨 일부 구현한 것도 있으며, 특히 온라인 컴퓨터 건조기 전용 프린터의 설계와 제작은 온라인 컴퓨터 건조 제어기의 1, 2차년도 개발 사업계획에 예정되어 있지 않으나 사업 수행 중에 시스템 구성상 매우 필요한 기능이어서 컴퓨터 전용프린트의 설계와 제작에 착수하였다. 각각 세부적 최종 결과 내용을 설명하면 다음과 같다.

	최 종 결 과 내 용	수행된 진도	
		1차 년도	2차 년도
1	기 존 컴 퓨 터 분 석	완료	없음
2	기존 CPU보드에 온라인 표준화된 통신 기능회로 설계와 중앙감시 컴퓨터의 온라인 통신소트트웨어	Hardware완료	Software 완료
3	온라인 컴퓨터 건조기 전용모뎀	Hardware 완료	Software 완료
4	온라인 컴퓨터건조기 전용프린터	Hardware 완료	Software 완료
5	중 앙 감 시 System 보 드	Hardware 완료	Software 완료
6	Hardware Softerware Test & Debug	test 완료	test 완료

1 기존의 컴퓨터 건조기의 분석

대부분 기능적으로 미비한 8bit CPU를 사용하고 온라인 중앙 통제 방식이 없음. 국내 개발된 컴퓨터 건조기의 경우 (1차년도 보고서 침조) 컴퓨터 온라인에 대한 기능이 없거나 있다 하더라도 단순한 중앙 감시용 표시장치일 뿐 전용모뎀 기능이나, 전용프린트의 기능 혹은 온라인 개념이 없어서, 무선 호출기능이나 중앙 통제 컴퓨터에 의한 온라인 제어 기능이 없는 실정이며 중앙 제어 컴퓨터에 의한 원격 제어 프로그램의 구성에 대해서는 발표된 바가 없다. 특히 단순한 중앙 표시 장치인 경우 에도 대부분 bus로 구성하여서 하나의 통신선로에 이상이 있는 경우 그 뒷

단의 모든 라인의 건조기가 통신장애가 오게 되어 있어 매우 불편하며 원격 통신기능이 없이 개별 건조기의 각각의 온도센서가 중앙 표시 장치에 연결되어 있는 방식을 채용하고 있는 실정이다.

2. 통신회로 설계와 중앙감시 컴퓨터의 온라인 통신 소프트웨어

기존의 컴퓨터 건조기를 Network상에서 온라인으로 제어하기 위해 필요한 표준화된 통신 Interface 회로를 설계하여 전용 모뎀, 전용 프린터, 중앙감시 CPU보드가 이 표준화된 통신 회로에 상호 호환되게 접속할 수 있게 하였고(1차년도 보고서 참조), 중앙 감시 컴퓨터 온라인 S/W를 개발하였다.

기존 건조기가 온라인 제어되기 위해서 갖추어야 할 기본 조건은 하드웨어적으로는 Photo Coupler 2개와 저항 2개면 충분하며 소프트웨어적으로는 위의 Protocol을 만족시켜야 한다. 위의 Protocol을 만족시키기 위한 Block화된 소프트웨어 흐름도는 다음과 같다.

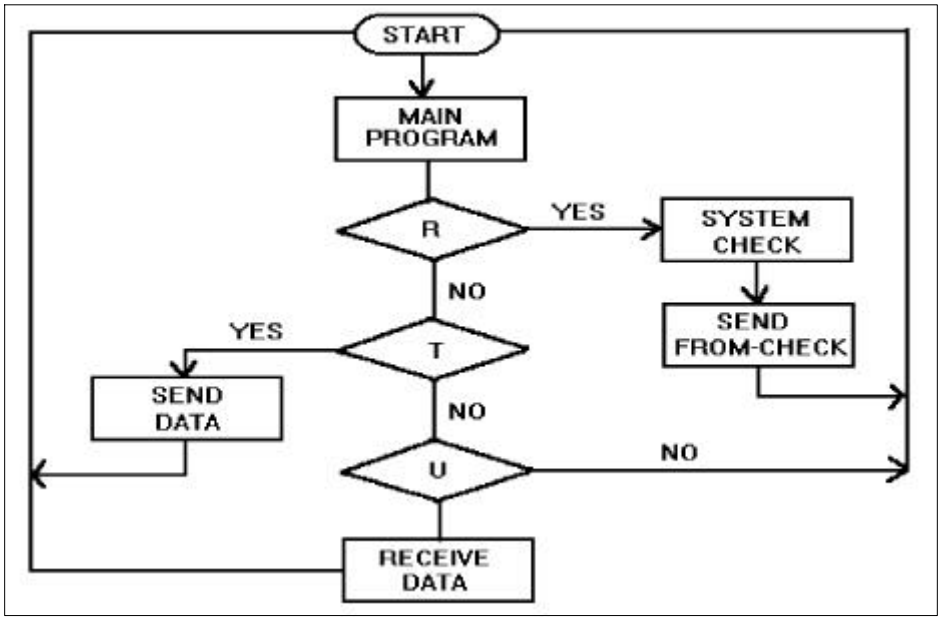

그림 3.건조기 통신 프로그램 흐름도.

구현된 Protocol은 'R','T','U' 명령이며 'R'명령인 경우 시스템의 내부 사항을 온라인으로 통하여 감시제어가 가능하다. 'T'와 'U'명령의 경우는 건조기와 중앙 제어 장치 사이의 File의 이동에 관한 부분이다. 기존 건조기를 온라인화하여 시스템 기능을 향상시키는 방법은 다른 대규모 시스템과 마찬가지로 가장 적은 하드웨어 변화를 주면서 소프트웨어를 개선시키는 방식이다. 하드웨어의 수정은 경제적, 시간적으로 매우 어려운 작업이지만 소프트웨어의 수정은 간단히 시스템내의 EPROM만 교체하면 되기 때문에 기존의 건조기를 온라인 제어하려는 본 연구 방식은 타당하다 하겠다. 따라서 기존 건조기의 회로 설계에 하드웨어적으로 경미한 변화를 줌으로써 외부적으로 전용 모뎀과 중앙 감시 보드에 의해 적절히 운용되게 소프트웨어(EPROM)를 변화시킬 수 있으므로 기존의 시스템 온라인화가 간단히 이루어 질 수 있다.

중앙 감시 컴퓨터의 온라인 통신 소프트웨워는 건조기와 중앙 감시 보드에 전용 통신 프로그램과 통신을 통해 중앙감시 컴퓨터가 개별 건조기의 각종 자료를 수집하고 디스플레이 하는 프로그램이다. 각종 커멘드의 실행을 위한 프로그램의 개발이 완료되었으며. 초기에는 C 언어로 구현 하였으나 차후에 수리, 보수를 위하여 비쥬얼 베이직으로 개발하였다.

다음은 중앙 감시 컴퓨터의 온라인 통신 소프트웨어 프로그램 요약하여 설명하였다.

이 프로그램은 Visual Basic(이하 VB)을 이용하여 구현하였다.

온라인 통신 소프트웨어 프로그램명 : 아리랑 ver 2.0

가. 기본구조

아리랑은 VB의 전용 통신 콘트롤인 MSComm을 이용하여 시리얼 포트를

제어하며, 직접 건조기와 통신을 하거나 모뎀을 통하여 건조기와 원거리 통신을 할 수 있다. 건조기와의 통신은 2진 코드로 이루어지며 이 2진코드를 10진코드로 변환하여 컴퓨터 화면에 아리랑을 통하여 표시하는 것이 가장 기본이자 핵심구조이다. 그리고 publicof.bas파일에 아리랑을 구현하기 위해 사용된 함수와 변수를 포함하고 있다.

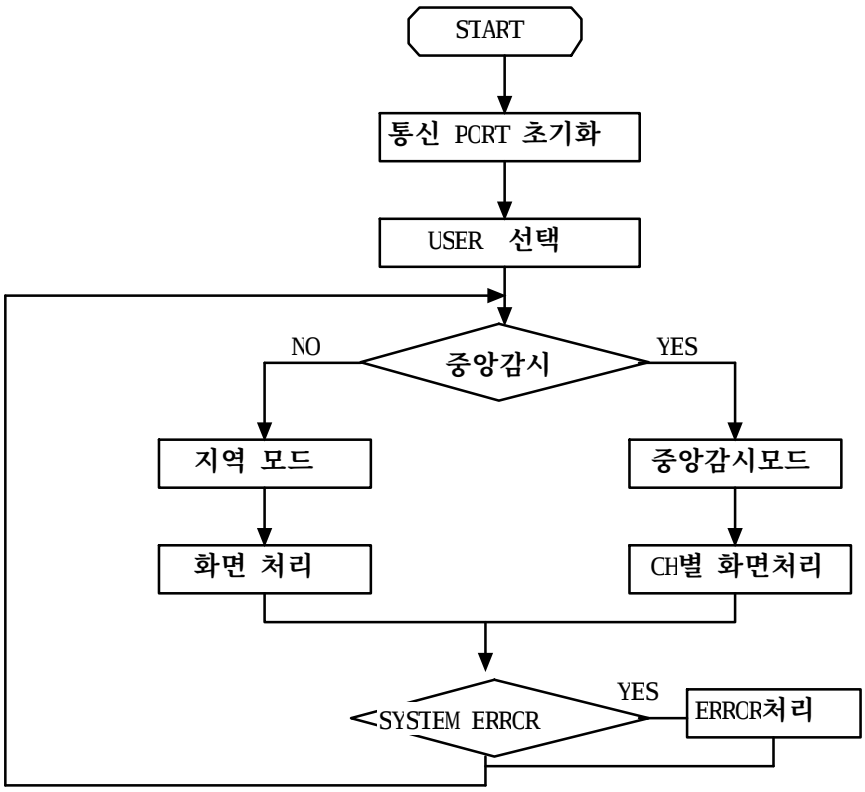

그림.4 온라인 통신 소프트웨어 흐름도

나. 아리랑의 화면 설명

다음 그림 5는 아리랑의 초기 화면이다.

아리랑 초기화면(그림5)는 아래와 같은 메뉴구조를 가지고 있다.

<초기화면 메뉴 구조>

주메뉴 ▶	시스템	사용자	감시체제	데이터
서브메뉴▶	-아리랑이란	-사용자등록	-지역감시체제	-기록상황
	-통신환경설정	-사용자선택	-중앙감시체제	
	-전화걸기	-사용자삭제		
	-전화끊기	-사용자보기		
	-끝내기	-전체사용자		
		-현재사용자		

그림5 아리랑 초기 화면.

아리랑 초기화면으로 C로 구현한 도스용 아리랑보다 차후 업그래이드가 용의한 VB로 윈도우버전으로 구현하였다. 그래서 Ver2.0으로 했다.

초기화면 메뉴 설명

(1) 통신환경설정.

그림 6. 통신환경 설정 화면

통신환경 설정화면으로 모뎀포트나 시리얼 포트를 설정한다. 통신에 앞서 반드시 미리 설정해주어야만 한다. VB의 MSComm으로 포트를 제어하며 올바른 포트를 설정해야만 통신을 할수 있다..

(2) 전화걸기, 전화끊기.

그림 7. 전화번호 입력 화면

원거리 모뎀통신을 원할 경우 전화 걸기 메뉴를 선낵하면 로드 되는 화면이다. 통신하고자 하는 전화번호를 입력해서 OK버튼을 선택하면 전화 걸기가 시작된다. 만약 전화걸기가 실행이 된지 않으면 통신환경설정에서 통신 포트가 제대로 설정되었는지 살펴본다.

(3) 사용자 등록.

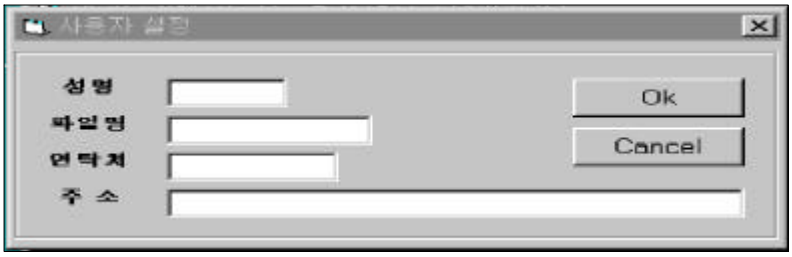

그림8. 사용자 등록 화면

사용자가 처음 사용할 때 자신과 저장되기 원하는 파일명을 등록한다. 이름, 파일명, 전화번호, 그리고 주소로 구성되어 있으며 특히 파일명을 반드시 등록해야만 파일로 저장 된다.(성명과 파일명은 반드시 등록)

(4) 사용자 선택, 사용자 삭제.

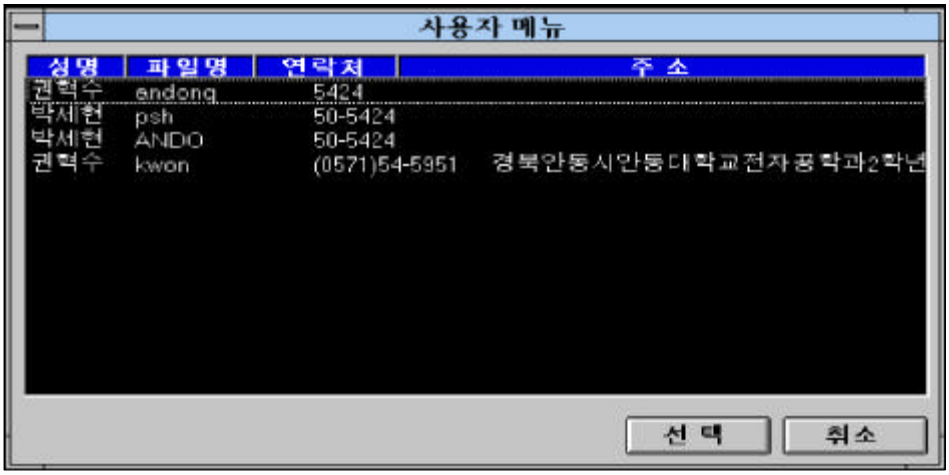

그림 9. 사용자 선택 화면

사용자가 등록되어 있을 경우, 사용자로 선택하거나 불필요한 등록된 사용자를 삭제할수 있다. 단 현재 사용자(선택된 사용자)는 삭제할 수 없으며 삭제하고자 할 때는 현재사용자를 변경하여야 한다.

(5)중앙감시 모드 및 지역감시 모드

그림10. 중앙 감시 모드 화면

이 화면은 중앙 감시 모드 선택시 로드되는 화면으로 10개의 채널의 상황을 살펴볼수 있다. 그리고 가동중인 체널은 회색으로 표시가 되며 흰색으로 표시된 비가동 채널과 쉽게 구별할수 있다. 또한 가동중인 채널 바탕을 더블 클릭 하면 자세한 상황을 볼 수 있다. 더블 클릭한 화면은 지역 감시 모드(그림12)와 같은 자세한 건조기 상태를 살펴볼 수 있는 화면이 로드되어 자세한 체널상황을 살펴볼 수 있다. 그리고 다시 중앙 감시모드로 복귀할 수 있으며 이때, 복귀했을 경우는 반드시 통신초기화를 해주어야 한다.

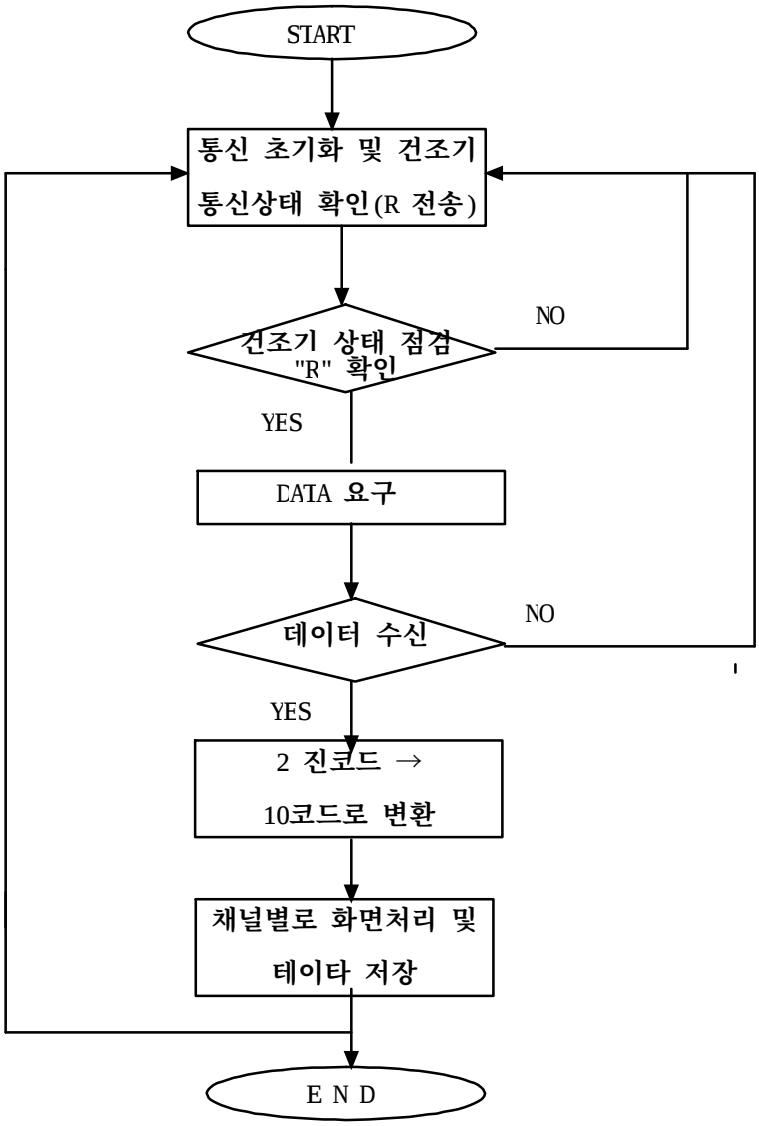

그림11. 중앙감시모드 흐름도

그림 12. 지 역 감 시 모 드

지역 감시 체제를 선택했을 경우와 중앙 감시 모드에서 채널을 더블클릭했을 경우의 화면이다. 이 화면에서는 건조기의 채널번호와 단계시간 및 각종 배습기 상황 그리고 점검사항의 세부 항목까지 확인할수 있는 기능을 가지고 있다. 이 화면에서 전화면메뉴를 선택하면 중앙 감시 모드에서 들어왔을 경우 중앙 감시 모드로 복귀하구 그 외의 경우(초기 화면에서 들오왔을 경우)는 초기화면으로 복귀한다.

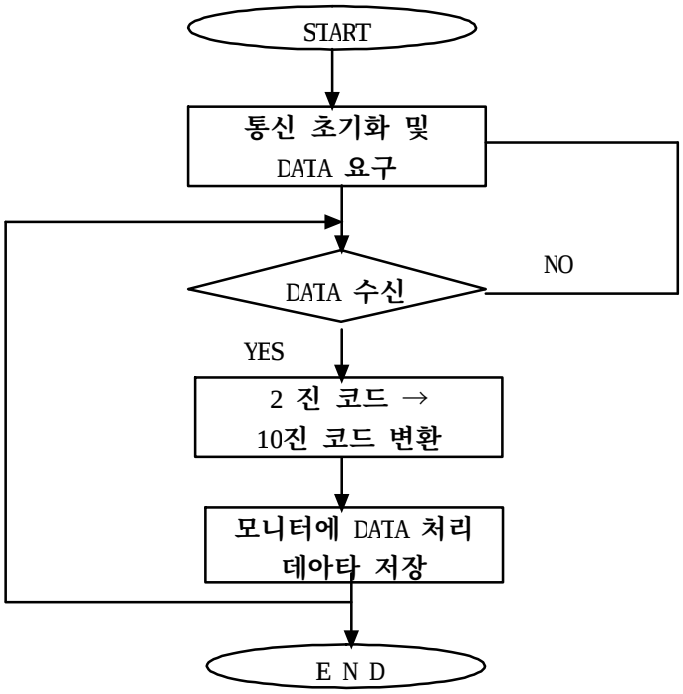

그림13. 지역 감시 모드 흐름도

(6)기록상황.

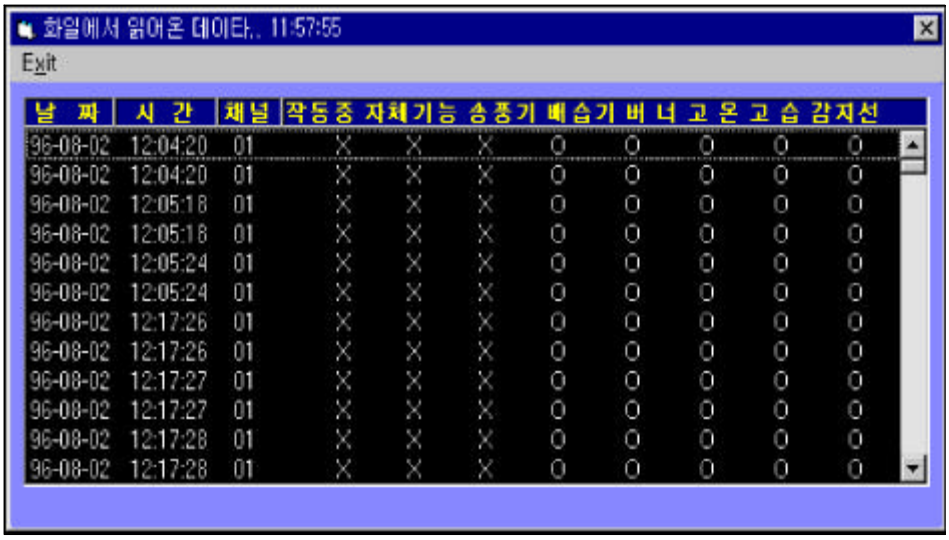

그림 14. 테이터 로드 화면.

기록상황 메뉴에서 화일이름을 입력하면 건조기의 시간별 상태를 확인

할 수 있다.

위와 같이 대략적인 아리랑의 화면모양 및 화면 기능을 알아보았다. 간략하게 사용방법을 요약하자면 먼저 통신 포트를 선택하고 그리고 사용자 선택 및 사용자 사용화일을 선택한다. 그리고 모뎀일 경우는 전화를 걸어서 건조기와 접속을 한다. 주변에서 흔히 사용하는 통신 소프트웨어의 사용방법과 유사하므로 사용상 어려움은 없을 것이다.

다.. 아리랑의 내부 함수 설명

아리랑에 사용된 변수와 함수는 publicof.bas 파일에 대부분이 포함되어 있다.

사용자 지정 화일에서 변수로 데이터를 로드하여 윈도우에 출력할 때 사용되는 함수이다.

```
Sub FileToAllData()
  -- Dim R%
     Dim MyStr As String * 13
     On Error Resume Next
     FileName = App.Path + "\" + gFileName '파일명 저장하기
     RecordLen = Len (Item)     '레코드 길이 계산
     FilePoint = FreeFile  '사용가능한 파일번호 획득
     Open FileName For Random As FilePoint Len = RecordLen

     ...

     LastRecord = FileLen(FileName) / RecordLen  '파일의 마지막 레
```

코드 번호 파악.

```
    If LastRecord = 0 then
       Close FilePoint
       MsgBox "화일에 데이터가 없습니다", 0 + 64, "테이터 리스트"
       Exit sub
    End If
    ...
' 16진수를 10진수로 변환하는 함수
Function HexToDecimal(dat As String) As Integer
If Len(dat) = 2 Then    '2자리
    ' 상위 4비트
      tmp = Asc(Mid(dat, 1, 1))
       If tmp >= 48 And tmp < 58 Then
        HexToDecimal = (tmp - Asc("0")) * 16
    Else
        HexToDecimal = (tmp - Asc("A") + 10) * 16
    End If
    '하위 4비트
       tmp = Asc(Mid(dat, 2, 1))
       If tmp >= 48 And tmp < 58 Then
       HexToDecimal = HexToDecimal + tmp - Asc("0")
    Else
        HexToDecimal = HexToDecimal + tmp - Asc("A") + 10
    End If
Else ' 1자리
```

```
    tmp = Asc(Mid(dat, 1, 1))
    If tmp >= 48 And tmp < 58 Then
        HexToDecimal = HexToDecimal + tmp - Asc("0")
    Else
        HexToDecimal = HexToDecimal + tmp - Asc("A") + 10
    End If
End If
```

사용자 파일을 읽어서 사용자 배열변수에 저장한다.

사용자는 최대 20명까지 등록된다.

true - **성공적으로 파일을 읽음.** false - **파일읽기 실패**

```
Function UserFileRead() As Boolean
   Dim i%
   On Error Resume Next
   FileName = App.Path + "\" + "User.dat"
```
- **사용자 정의 데이타 파일**

```
   RecordLen = Len(User(1)
```
- **레코드 길이 계산**

```
   FilePoint = FreeFile
```
- **사용가능한 파일번호 획득**

```
   '랜덤파일 오픈이나 생성
   Open FileName For Random As FilePoint Len = RecordLen
```
- **랜덤파일 오픈이나 생성**

```
UserMax = Int(FileLen(FileName) / RecordLen)
```
- **파일의 마지막 레코드 번호 파악**

```
   If UserMax = 0 Then
      UserFileRead = False
```

```
        Exit Function
    End If
    For i = 1 To UserMax
        Get #FilePoint, i, User(i)  - 파일에 한 레코드 저장
     Next i
    Close FilePoint                - 파일을 제거 한다.
    UserFileRead = True
```

사용자 파일에 사용자 변수의 내용을 입력한다.

true - 사용자 파일에 쓰기 성공. false - 파일에 쓰기 실패

```
Sub UserFileWrite(tmp As UserInfo)
On Error Resume Next
'사용자 정의 데이타 파일
FileName = App.Path + "\" + "User.dat"
'레코드 길이 계산
RecordLen = Len(tmp)
'사용가능한 파일번호 획득
FilePoint = FreeFile
'랜덤파일 오픈이나 생성
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
UserMax = Int(FileLen(FileName) / RecordLen)
' 한 레코드 증가를 위해 변수의 값 증가
UserMax = UserMax + 1
' 파일에 한 사용자 내용을 쓴다.
```

```
Put #FilePoint, UserMax, tmp
' 파일 닫기
Close FilePoint

' 중앙감시모드에서 변수의 값을 채널별로 화면에 출력한다.
'해당 채널명은 파란색의 전경색을 나타낸다.
Sub VarToChannelData()
   Dim MyVar As Integer
   '텍스트 박스에 값을 채운다.
   ' 채널별로 TCha#(0 ~ 6) 의 변수명을 가진다.
   If Item.Item(1) = "0A" Then '채널번호
    MyVar = 10
   Else
       MyVar = Val(Item.Item(1))
' 변수에서 파일로 데이터 구조체 변수를 저장한다. 현재 수신데이터를
파일에 추가하는 것이다.
Sub VarToFile()
   On Error Resume Next
   '파일명 지정하기
   FileName = App.Path + "\" + Trim(User(UserNumber).FileName)
   '레코드 길이 계산
   RecordLen = Len(Item)
   '사용가능한 파일번호 획득
   FilePoint = FreeFile
   '랜덤파일 오픈이나 생성
```

```
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
LastRecord = FileLen(FileName) / RecordLen
'파일이 생성되었을 경우
If LastRecord = 0 Then
    LastRecord = 1
Else ' 데이터의 한 레코드 추가
    LastRecord = LastRecord + 1
End If
' 파일에 한 레코드 저장
Put #FilePoint, LastRecord, Item
'파일을 제거 한다.
 Close FilePoint
```

■ 중앙감시 컴퓨터 온라인 통신 프로그램-아리랑(첨부1 - P.47)

3. 전용 모뎀 개발.

컴퓨터 건조기의 온라인 기능을 가지기 위해서 건조기 전용 모뎀 장치를 개발하였다. 기존의 일반 모뎀과 차별화된 건조기 전용 모뎀의 특징은 컴퓨터 건조기의 온라인 기능에 필요한 재반 기능을 가지게 하였다. 이 기능의 주요한 특징은 건조기의 내부 작동 상태를 전용 모뎀이 능동적으로 점검하여 건조기 작동 상태를 진단하는 기능을 가지고 있으며 건조기의 이상이나 중앙 통제 컴퓨터의 요구가 있을시 건조기의 내부상태를 알려주는 역할을 한다. 특히, 무선호출 기능을 내장시켜 이상 발생 즉시 원거리에 있는 사용자의 무선 호출기나 서비스 센터로 건조기의 이

상 상태를 전송하도록 하여 즉각적인 서비스 체계를 구축할 수 있으므로 건조물에 대한 신뢰성을 높일 수 있게 하였다. 다음 그림 16은 전용 모뎀의 구현된 보드의 실제 사진이다. 전용 모뎀의 소프트웨어적 운용은 첫째, 모뎀 제어 장치의 동작 감시와 건조기 시스템에 주기적 'R'명령을 줌으로써 건조기 시스템의 결함을 온라인으로 중앙감시 컴퓨터에 접속 시키거나 무선호출을 하도록 한다.(1차년도 보고서 참조)

다음 그림15와 그림16은 각각 건조기 전용 모뎀의 사진과 제어흐름도이다. 주기적인 데이터를 건조기로 부터 요구하며 건조기 시스템의 결함이 그 테이터로 통해 CHECK 된다. 이 모뎀은 컴퓨터 건조기에 직접 연결하여 사용하며 건조기의 진단기능이 내장되여 있으며 건조기에 이상이 있을 시 무선호출 기능을 지니고 있으며 중앙감시 컴퓨터에 의해 자료를 주고 받을 수 있다.

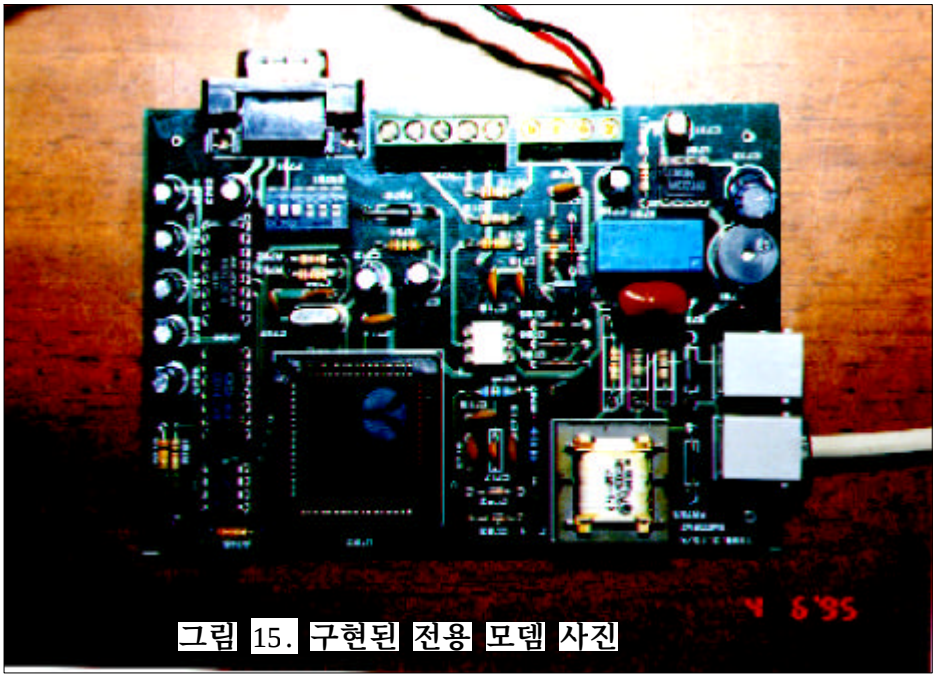

그림 15. 구현된 전용 모뎀 사진

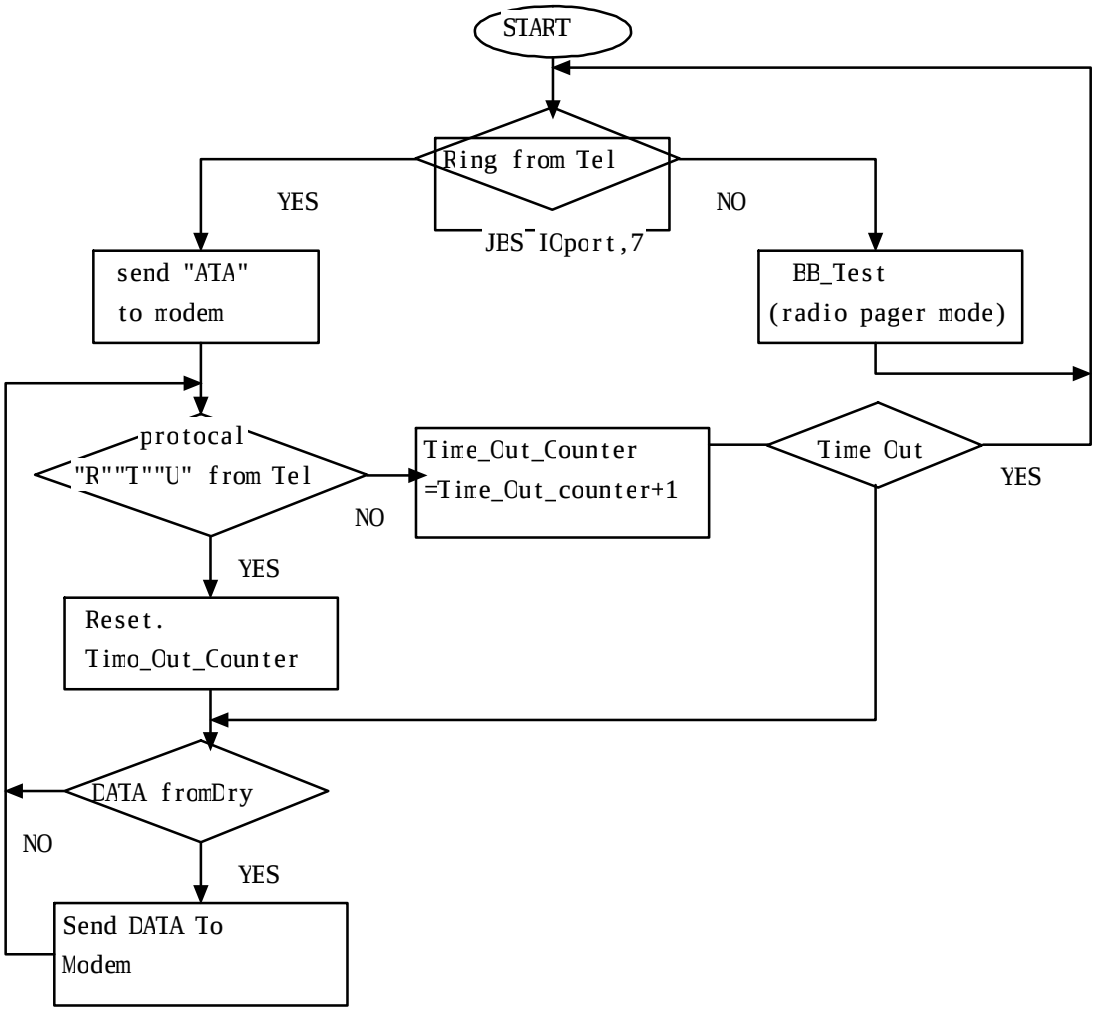

그림16. 건조기 전용모뎀 소프트웨어 흐름도

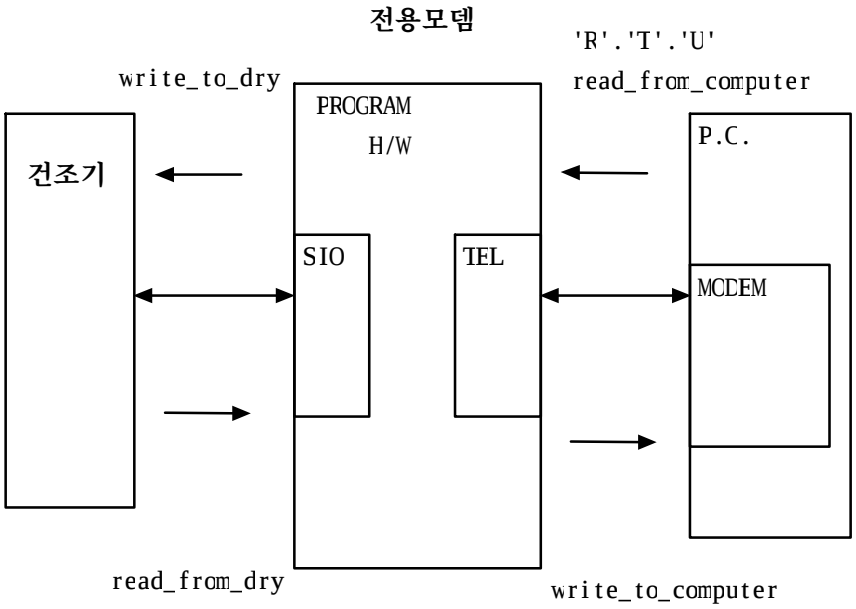

그림 17. 전용 모뎀 프로그램 구성도.

그림17은 건조기 전용모뎀 소프트웨어를 요약 한 것이다. 전용 모뎀프로그램의 구조는 크게 4부분으로 나누어져 있는데 Read_from_computer, Write_To_Dry, Read_Form_Dry 및 Write To Computer 이다. 각각의 루틴의 설명은 다음과 같다.

Read_from_computer .

①Modem_RX_Status : 모뎀의 상태를 읽음.

②“R", "T", "U" 프로토콜 확인.

③Write_To_Dry 수정.

④No Carrier나 Error가 있다면 초기화 루틴으로 간다.

Write_To_Dry

①모뎀으로(Tel) 온 DATA를 건조기로 전송한다.

②Timer_Out_Counter를 Reset시킨다.

Read_Form_Dry

①RX_Status : SIO의 상태를 읽음.

②DATA가 있을시 건조기의 전송자료를 읽는다.(RX_DATA)

③Write_To_Computer수행

Write To Computer

①Modem_TX__DATA을 수행하여 건조기의 전송자료를 Tel을 통하여 Computer로 전송 함.

■ 건조기 전용모뎀 소프트웨어 (첨부 2 - P.79)

4. 전용 프린터 개발.

건조기용 온라인 전용프린터 설계 제작은 계획상에는 없었으나, 온라인 컴퓨터 건조 제어기의 구성상 반드시 필요한 기능이어서 연구 개발을 착수하였으며 컴퓨터 건조기용 전용 프린터는 건조기 건조 상태를 프린트하여 특정 건조물의 건조상태와 온도, 습도를 건조기의 건조 상태에 기록하여 건조 프로그램과 건조 제어 상태의 자료를 차후 활용 할 수 있게한다. 특히 건조기의 전용 프린터기는 일반 프린터와 같은 데이타 출력 기능을 가지고 있을 뿐 아니라 건조기의 내부 출력 상태나 건조기의 일시적 시스템 다운까지 점검 가능하게 하였다.(1차년도 보고서 참조)

그림 20는 구현된 보드의 실제 사진이다. 전용 프린터의 소프트웨어적 운용은 전용 모뎀과 비슷하며 그 차이는 전용 모뎀이 건조기 결함과 상태를 온라인 중앙 감시 보드에 전송시키는데 비하여 전용 프린트는 건조기의 이상 상태를 프린트하게 하였다. 그림 18은 건조기 전용프린트의 흐름도이다. 컴퓨터 건조기용 전용 프린터는 건조기 건조 상태를 프린트하여

특정 건조물의 건조상태와 온도, 습도를 건조기의 건조 상태에 기록하여 건조 프로그램과 건조 제어 상태의 자료를 차후 활용할 수 있게 한다. 특히 건조기의 전용 프린터와 같은 데이터 출력 기능을 가지고 있을 뿐 아니라 건조기의 내부 출력 상태나 건조기의 일시적 시스템 다운까지 점검 가능하게 하였다. 다음은 건조기 전용 프린터의 흐름도이다. 주기적인 DATA를 건조기로 부터 요구하며 ERRCR 발생 시나, 주기적인 기간동안 건조기의 각종 자료를 PRINT하게 된다. 그림 19는 구현된 건조기 전용프린트의 출력상태이다.

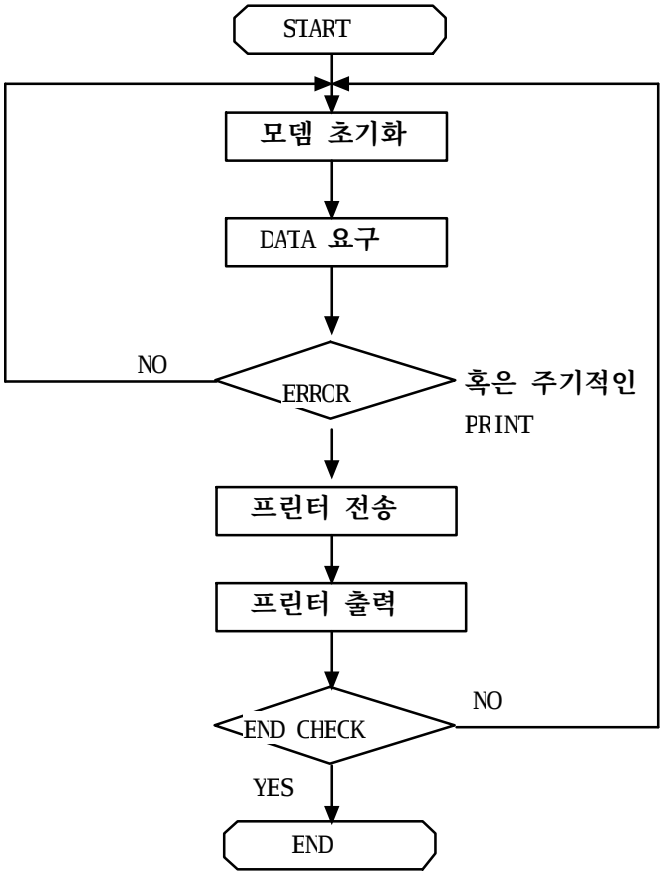

그림18 건조기 전용 프린터 흐름도

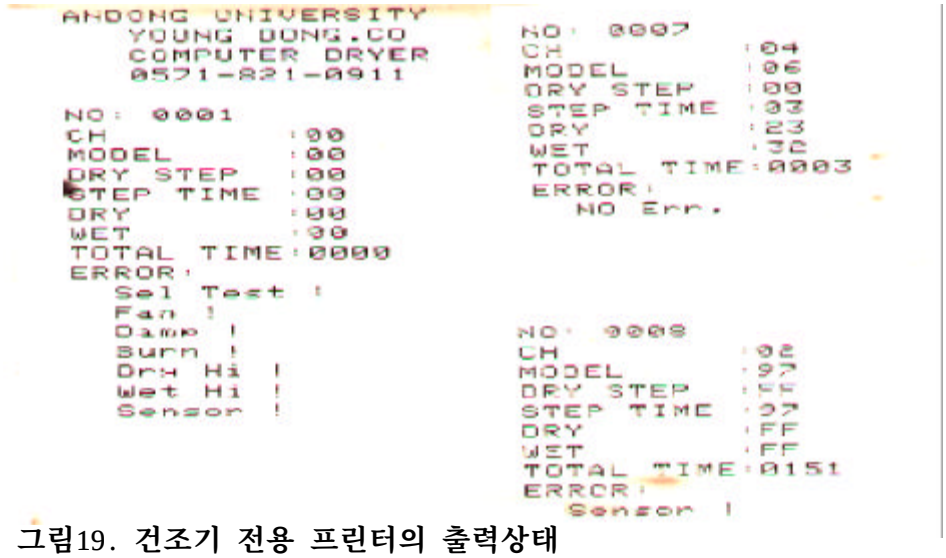

그림19. 건조기 전용 프린터의 출력상태

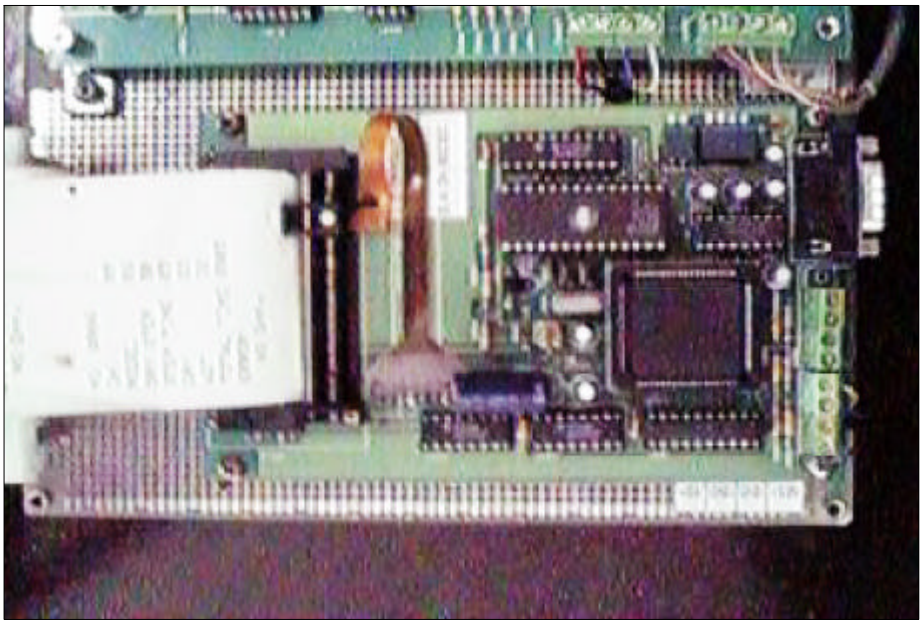

그림 20. 구현된 전용 프린트 사진

■건조기 전용 프린터의 프로그램(첨부 3 - P.106)

5. 중앙감시 시스템 보드 설계.

연구 개발된 중앙감시 시스템보드는 마을 공동 단위의 건조장치에 여러대의 건조기를 연결하여 중앙 감시장치에 각 건조기의 진행 상황을 표시하는 장치이며 시중에 개발된 두어개의 장비가 있다. 그러나 이들의 중앙감시 보드는 단순히 개별 건조기 표시 장치를 연장하여 중앙 표시 장치에 표시하는 것일뿐 온라인 기능이 없다. 특히 접속 방식이 일반적 Bus 방식으로 되어 있어 임의 선로가 끊어지면 그 뒷부분은 동작 하지 못하는 단점이 있다. 따라서 본 개발자는 Bus연결방식이 아닌 Star 방식으로 개별 건조기를 연결하였다.(1 차년도 보고서 참조) 중앙 감시 보드 소프트웨어는 마을 공동 단위의 개별 건조기들을 감시하기 위한 것이다. 온라인 컴퓨터 중앙 제어용 소프트웨어 개발은 컴퓨터와 최대 16대의 개별 건조기와 건조기 사이에 데이터를 수집하고 결함을 검출하며 결함이 생겼을 시 전용 프린터나 전용모뎀을 통하여 에러 메시지를 전송할 수 있을 뿐 아니라 컴퓨터와 직접 연결이 가능 하게끔 프로그램을 수정, 완료하여 실용화를 가능하게 하였으며, 구체적으로 영동농기구 회사에 기술이전 계약을 하여 96년도 현재 생산 판매하여 농민들에 의해 가동중이다.

아래 그림 21.은 중앙 감시 보드의 온라인 통신 소프트웨어 흐름도이다. 우선 중앙 감시보드의 각 장치의 초기화를 시키고 각 16 채널의 임의 건조기에 대한 채널의 자료가 전송되어 왔는지를 확인한다. 만약 채널의 자료가 전송되어 왔다면 이를 해당 표시판에 표시하고 중앙 감시 컴퓨터와 전용모뎀 및 전용 프린터의 데이터 전송요구를 확인후 이 채널의 자료를 전송할 지를 결정 한다. 만약 전송되어 온 채널의 자료를 통해 건조기의 이상을 발견 하였을 시는 중앙 감시보드는 경고음을 내고 대당 표시판에 경고 신호를 표시한다. 한편 중앙 감시보드로 부터 체널의 자료를 다운 중앙 감시 컴퓨터나, 전용모뎀 혹은 전용프린트는 자체의 점검 절차

에 의해 외부에 경고 메시지를 보낸다.

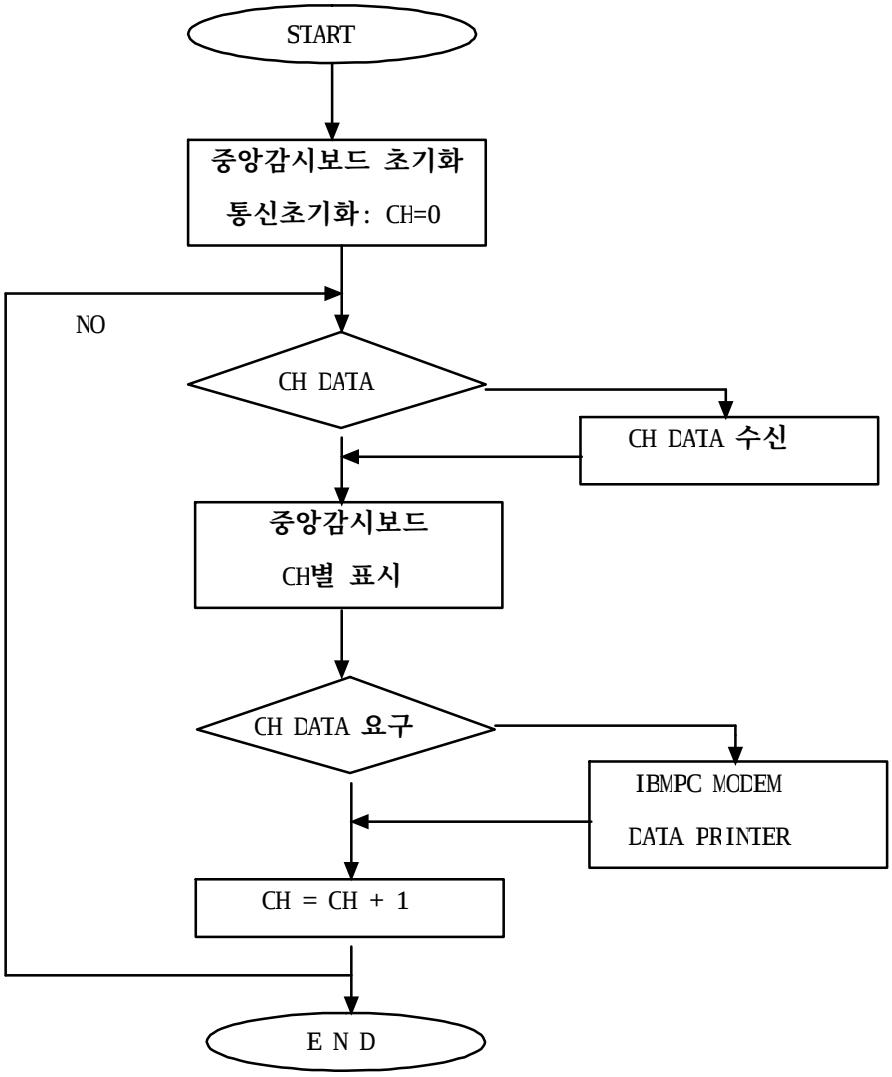

그림 21. 중앙 감시 보드의 소프트웨어 흐름도.

■ 중앙 감시 보드 소프트웨어 (첨부 4 - P.141).

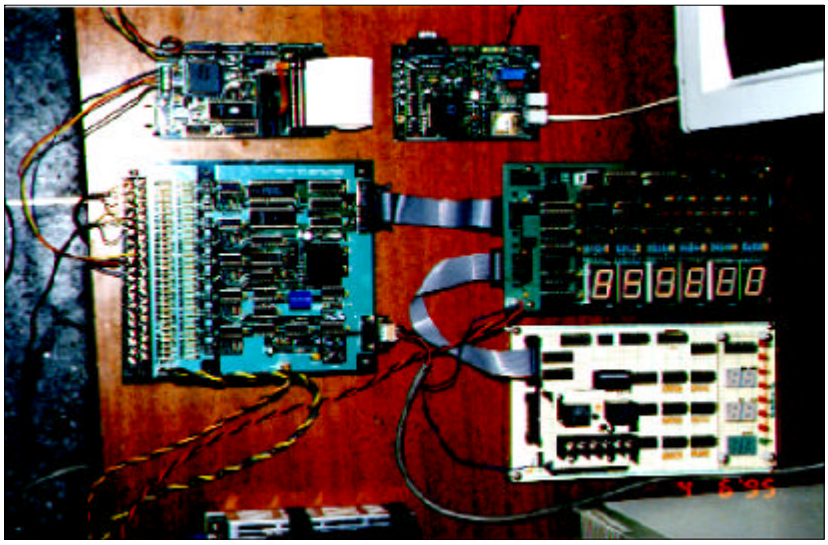

그림 22(a). 구현된 중앙 감시 보드 사진

또한 그림 22(a)는 구현된 중앙 감시 보드를 실험실에서 Test하는 과정이며 그림 22(b)는 10대의 건조기를 동시에 감시하는 완성된 중앙 감시 시스템이다.

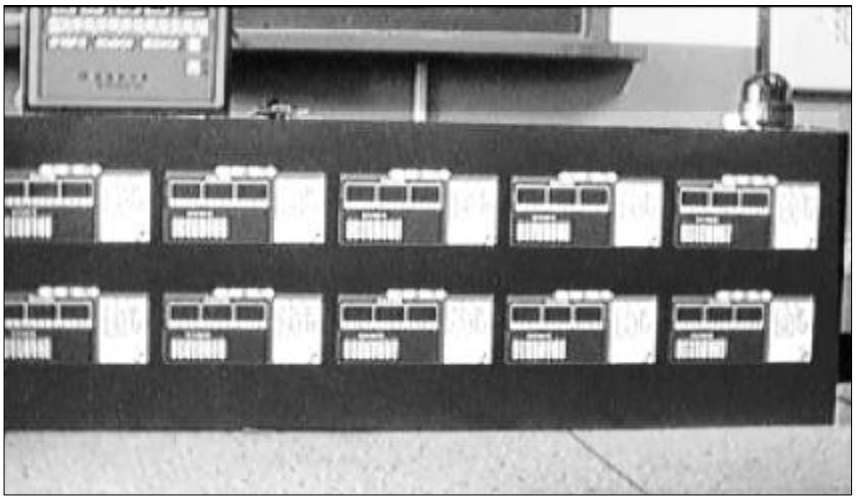

그림 22(b). 완성된 중앙 감시 시스템.

제 3 장 결론

온라인 통신기능이 부과된 시스템의 종합적인 시험을 위해 연구실내에 중앙감시 보드 및 프린터 전용모뎀 및 컴퓨터를 설치하여 종합적인 프로그램을 수행하였고 그 결과 정상적인 동작을 확인하였고 본 연구개발 과제의 결과를 테스트하는 과정에서 건조기 생산관계자들과 의견 교환한 결과 본 연구 개발 사업 성과에 대한 실용화에 대해 매우 긍정적 반응을 얻었다.

본 연구의 성과는 농업분야의 전자 기술 산업 부분이어서 본 연구 개발 사업의 성과를 농어촌에 널리 보급하기 위해서는 농업부분의 중소 기업의 도움을 받아야 하는 농업기계 기술이다. 따라서 온라인 컴퓨터 건조제어기의 연구성과를 활용하기 위해 1995년 6월 부터 본 기술의 활용과 장점을 홍보한 결과 농산물 건조기 생산업자인 영동농기구에서 4차에 걸쳐 기술 자문과 기술의 활용을 위한 협의를 하였고 농림수산부 및 농수산 기술관리센터의 협조를 얻어 1996. 6.30일 농림수산기술개발사업 성과 활용에 따른 기술실시 계약서를 영동농기구와 체결하게 되었다.

계약서의 주요내용은 다음과 같다.

연구개발과제명: 온라인 컴퓨터 건조제어기

총 연구 개발비: 37,783,000원

기술료 징수액: 11,334,900원 (정부지원금의 30 %)

기술 사용 기간: 5년 (1996년 - 2000년)

계약 당사자 :

(갑) 주관연구기관 : 안동대학교 안동대학교 총장 이 진설

(을) 실시기업 : 영동농기구 제작소 대표 권 삼석

계약당사자인 영동농기구는 이미 96년 7월에 일차적으로 본 기술을 활용하여 제품 생산을 하고 제품시험및 시험운영을 해본 결과, 만족할 만한 결과를 얻었으며 온라인 제어를 할 수 있는 건조기를 수백대 시판하였다. 또한 영동농기구는 97년도 이후에 본격적으로 본 연구 개발 성과를 활용할 계획이다.

참고로 현재 영동농기구에서 관심을 갖고 생산하려 하고 농어촌 관련 중소 기업체에서 본 현장애로기술 성과를 활용하여 상품화에 기여할 수 있는 기술은 다음의 세 가지이다.

첫째. 중앙 감시 컴퓨터 온라인 소프트웨어
둘째. 중앙감시보드 설계기술
셋째. 건조기 전용모뎀 설계기술
네째. 건조기 전용프린터 설계기술

아래 그림 23, 그림 24, 그림 25는 온라인 통신기능이 부과된 시스템의 종합적인 시험을 위해 연구실내에 중앙 감시 보드및 프린터 전용모뎀및 컴퓨터를 설치하여 종합적인 프로그램을 수행하고 있는 사진과 실제 영동농기구에서 마을 단위 공동건조장에서의 중앙감시보드의 사진 및 실제 본 기술을 운용하고 있는 공동건조장의 모습과 공동건조장 내부 모양이다..

그림 23. 실험실 테스트 과정.

본 기술개발로 직접 간접으로 유발될 수 있는 기대되는 효과.는 다음과 같다.

1. 기대 효과

가. 개별 단위로 건조기를 사용할 경우 사용자의 조작상 실수나 건조기 시스템의 이상으로 인한 건조물의 피해가 생길 수 있었으나 컴퓨터 건조기를 온라인화 할 경우 마을 단위나 조합 혹은 서비스 센터의 중앙통제 컴퓨터에 의해 즉각적인 서비스 체계를 구축 할 수 있으므로 건조물의 피해를 최소화할 수 있어 건조물에 대한 신뢰를 높일 수 있다.

나. 건조기를 온라인화 함에 따라 각 지역에서 사용하고 있는 특정 건조물의 건조를 위한 건조 프로그램을 정보화하여 다른 지역에서도 공유할 수 있다.

다.각 지역 건조물에 대한 자료의 집계가 가능하므로 건조물에 대한 종합적 통계 지표로 활용할 수 있고 정보화가 가능하다.

그림 24. 본 연구 성과가 활용되고 있는 공동 건조장 모습

그림25. 제작 설치된 중앙감시 보드 장치와 온라인화된 개별 건조기.

라. 단위지역, 혹은 전국 단위의 사용중인 건조기와 사용하지 않는 건조기의 중앙 집계가 가능하므로 건조기의 효율적 활용에 이바지 할 수 있다.

마.개별 사용자가 개인용 컴퓨터에 의해 지역에 무관하게 자신의 컴퓨터 건조기에 접근하여 건조 진행 상황을 알 수 있으므로 농어촌의 정보

자동화 산업에 이바지할 수 있다.

2.연구개발사업 성과에 대한 활용 방안.

본 연구의 개발 성과를 활용하기 위해서 다음과 같은 요건이 만족할 때 상품화와 실용화 가능성이 있다고 생각됨.

○ 상품화 및 실용화가 가능한 개발 성과를 조속한 시일에 농어민이나 농업 부분의 영세 중소기업체가 성과에 대한 기술실시를 할 수 있게 정부의 재정적 보조가 필요함.

○ 본 과제의 성과가 농가에 매우 필요한 기술(농가공 분야)이나 농가에 널리 보급하기 위해서는 생산업체의 기술을 빌려야하는 전자기술 부분이며 또한 영세한 농업부문의 중소기업의 기술력 강화에 도움을 주는 기술이므로 개발된 기술을 농가에 확산, 실용화 시키는 조치가 마련 되어야 함.

○ 본 과제의 성과에 대한 실용화에 앞서 대량생산을 위한 생산 기술과 실증 시험이 필요한 과제임. 참고로 본 과제의 성과를 활용하기 위해서는 중소기업의 경우(연간 500대의 건조기 생산의 경우) 다음의 세 가지 품목을 생산 하는데 대해 필요한 생산 시설 및 실증 시험 재원은 다음과 같다.

생 산 품	생산설비	실증시험
1중앙 감시 컴퓨터 온라인 소프트웨어 2. 중앙 감시 보드 3. 건조기 전용모뎀 4. 건조기 전용프린터	8,000만원	3,300만원

따라서 생산설비및 실증시험을 위한 부대 비용(약 11,300만원)에 따른 투자로 미루어 실용화된 생산제품에 따른 초기이익이 많지 않을 것이므로 영세 농기구회사의 경우 기술실시료를 면제 하는 것이 타당하다고 본다.

○ 본 성과 중 건조기 전용모뎀과 건조기 전용프린터의 개발은 국내외 소개된 바가 없는 기술이므로 조속한 상품화가 필요함.

특히 현장관계자들의 말에 의하면 중국의 농수산 관계자들이 국내 건조기 사업에 매우 높은 관심을 보이고 있다고 하며 국산 건조기의 중국 진출이 희망적이라고 한다. 따라서 본 성과의 실용화및 상품화는 농가의 소득 증대, 정보화 및 자동화에 이바지하며 상품의 외국 수출에도 고부가가치를 창출할 것이라 본다.

○ 농수산기계의 정보화, 자동화가 필요

본 성과는 건조기의 Network화를 지향하고 있기 때문에 타 농가공분야의 on line 원격제어에 활용화할 수 있으며 농가공 기술의 정보화에 도움을 줄 수 있는 기술임

■첨부1) 중앙감시컴퓨터 온라인 통신프로그램(아리랑)

```
 ---------publicof.bas--------------
Public Const None = 0
Public Const Center = 1
Public Const LocalForm = 2
Public Flag As Integer
Public NewPort As Integer
' 16진수를 보관한다. 1바이트는 문자2자리로 표현
Public ItemTemp(1 To 11) As String * 2
Type SELFTESTINFO    ' 자체점검
    Working As Boolean '작동중bit7
    SelfFunc As Boolean '자체기능bit6
    SendWind As Boolean '송풍기bit5
    CutMos As Boolean '배습기bit4
    Berner As Boolean '버너bit3
    HotLevel As Boolean '고온bit2
    HighMos As Boolean '고습bit1
    Senser As Boolean '감지선bit0
End Type

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 각 시간의 데이타 구조
' 11바이트의 데이타를 화일저장시에는
' 바로 저장하고..
' 읽어들일 때는 자체점검을 분석해야한다.
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Public Type ItemInfo
    NowDate As String * 8 ' 날짜
    NowTime As String * 8 '시간
    Channel As Integer '채널
    Item(1 To 11) As String '11바이트짜리 16진수
    SelfTest As SELFTESTINFO
End Type

'~~~~~~~~~~~~~~~~~~~~~~
' 변수 선언
Public Item As ItemInfo
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 데이타 화일명은 arirang.dat
' 데이타 저장경로는 실행화일이 있는 곳
' 레코드 단위로 자료 I/O를 하기 위해
' 랜덤화일을 사용한다.
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Public FilePoint As Integer
Public RecordLen As Long
Public CurrentRecord As Long
Public LastRecord As Long
Public FileName As String

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자 데이타 포맷
' 사용자와 품목 자료의 파일간에
' 연결이 된다. 즉, 사용자 설정시에
' 데이타 파일명을 지정한다.
' 사용자 파일은 User.dat
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Type UserInfo
    Name As String * 6 '이름
    FileName As String * 15 ' 파일명
    Phone As String * 13 '전화번호
    Address As String * 40 ' 주소
End Type

'~~~~~~~~~~~~~~~~~~~~
' 사용자 변수 지정
'~~~~~~~~~~~~~~~~~~~~
Public User(1 To 20) As UserInfo
Public UserNumber As Integer '현재 사용자 번호
Public UserMax As Integer ' 총 사용자 수

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자 변수에서
' 설정 = , 선택 = 2, 삭제 = 3, 보기 = 4
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Public Const UserSet = 1
Public Const USERSEL = 2
Public Const USERDEL = 3
Public Const USERVIEW = 4
Public UserMenu%

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 호출 함수를 알아내기 위한 변수
Public Const FromMenu = 10
Public Const FromForm = 20
Public OldFrom%
Public FromIs As Byte
Public ChannelHexa As String * 2 '채널번호
```

```
Public gFileName As String
Public gEnd As Integer ' 윈도우 닫기
' 타임 체크에 사용되는 변수
' 데이타 초기화 버튼 클릭시 작동하는 함수에서...
Public gPreT As Long
Public gNowT As Long
' 화면의 중앙에 위치
Sub CenterForm(Frm As Form)
Frm.Left =(Screen.Width-Frm.Width)/2:Frm.Top=(Screen.Height-Frm.Height)/2
End Sub
Sub VarToCeLocal()

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 화면에 변수의 11바이트 출력
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Dim i%, tmp$
FLocal.t1.Text = Item.Item(1)
FLocal.t2.Text = Item.Item(2)
FLocal.t3.Text = Item.Item(3)
FLocal.t4.Text = HexToDecimal(Item.Item(4))
FLocal.t5.Text = Item.Item(5)
FLocal.t6.Text = Item.Item(6)
FLocal.t7.Text = Item.Item(7)    '자체점검
For i = 0 To 7                  '작동중
    FLocal.t70(i).Value = False
Next i
If HexToDecimal(Item.Item(7)) And 1 Then
    FLocal.t70(0).Value = False
    Item.SelfTest.Working = False
Else
    FLocal.t70(0).Value = True
    Item.SelfTest.Working = True
End If
If HexToDecimal(Item.Item(7)) And 2 Then   '자체기능
    FLocal.t70(1).Value = False
    Item.SelfTest.SelfFunc = False
Else
    FLocal.t70(1).Value = True
    Item.SelfTest.SelfFunc = True
End If
If HexToDecimal(Item.Item(7)) And 4 Then   '송풍기
    FLocal.t70(2).Value = False
    Item.SelfTest.SendWind = False
Else
```

```
    FLocal.t70(2).Value = True
    Item.SelfTest.SendWind = True
End If
If HexToDecimal(Item.Item(7)) And 8 Then   '배습기
    FLocal.t70(3).Value = False
    Item.SelfTest.OutMos = False
Else
    FLocal.t70(3).Value = True
    Item.SelfTest.OutMos = True
End If
If HexToDecimal(Item.Item(7)) And 16 Then  '버너
    FLocal.t70(4).Value = False
    Item.SelfTest.Berner = False
Else
    FLocal.t70(3).Value = True
    Item.SelfTest.Berner = True
End If
If HexToDecimal(Item.Item(7)) And 32 Then  '고온
    FLocal.t70(5).Value = False
    Item.SelfTest.HotLevel = False
Else
    FLocal.t70(5).Value = True
    Item.SelfTest.HotLevel = True
End If
If HexToDecimal(Item.Item(7)) And 64 Then   '고습
    FLocal.t70(6).Value = False
    Item.SelfTest.HighMos = False
Else
    FLocal.t70(6).Value = True
    Item.SelfTest.HighMos = True
End If
If HexToDecimal(Item.Item(7)) And 128 Then  '감지선
    FLocal.t70(7).Value = False
    Item.SelfTest.Senser = False
Else
    FLocal.t70(7).Value = True
    Item.SelfTest.Senser = True
End If
FLocal.t80.Text = Item.Item(8)                '배습기
tmp = Item.Item(8)
For i = 0 To 2
    FLocal.t8(i).Value = False
Next i
FLocal.t81.Value = False
```

```
' 열림
If Not (Val(Asc(Right(tmp, 1))) And 6) Then FLocal.t8(0).Value = True
'반열림
If (Val(Asc(Right(tmp, 1))) And 2) And Not Val(Asc((Right(tmp, 1))) And 4)
Then FLocal.t8(1).Value = True
'닫힘
If Val(Asc(Right(tmp, 1))) And 6 Then FLocal.t8(2).Value = True
'버너on
If Val(Asc(Right(tmp, 1))) And 1 Then FLocal.t81.Value = True Else
FLocal.t81.Value = False
FLocal.t9.Text = Item.Item(9)                    '버너off 출력상태
FLocal.t10.Text = HexToDecimal(Item.Item(10))    '총경과시간
FLocal.t11.Text = Item.Item(11)                  '합계-16진수로 출력
End Sub

' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 현재시간을 출력한다.
Function Clock(Title As String) As String
Clock = Title + Format$(Now, "hh:mm:ss")
End Function

Static Sub ComToVar(Cta As Variant)
    On Error Resume Next
Dim Txt$

' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 임시변수의 데이타를 구조체 변수에 저장
Item.NowDate = Format(Date, "yy/mm/dd")
Item.NowTime = Format(Time, "hh:mm:ss")
Item.Channel = Format$(ItemTemp(1), "00") '채널번호
' ~~~~~~~~~~~~~~~~~
'나머지 자체점검 비트 속성의 저장은
' 화면에 출력하는 부분에 나타난다.
End Sub

' 현재 시간 변경
Sub CurChange()
' 20초를 더한다.
CurS = CurS + 20
If CurS >= 60 Then
    CurS = 0: CurM = CurM + 1
    If CurM = 60 Then
        CurM = 0: CurH = CurH + 1
        If CurH = 24 Then
```

```
            CurH = 0
        End If
    End If
End If
End Sub
' 하루를 증가시킨 날짜 반환한다.
Function DateDec() As String
Const jan = 31
Const feb = 28
Const mar = 30
Const apr = 31
Const may = 31
Const jun = 30
Const jul = 31
Const agu = 31
Const sep = 30
Const oct = 31
Const nov = 30
Const dec = 31
Dim Y As Integer, M As Integer, D As Integer
Dim FebD%
Y = Val(Left(BeginDate, 2)) '년
M = Val(Mid(BeginDate, 4, 2)) '월
D = Val(Right(BeginDate, 2)) '일
' 윤년검사
If M <= 2 Then
    If Y Mod 4 = 0 And Y Mod 100 <> 0 And Y Mod 400 Then
        FebD = 29
    Else
        FebD = 28
    End If
End If
'날짜감소
D = D - 1
Select Case M
Case 1
    If D > jan Then M = 2: D = 1
Case 2
    If D > FebD Then M = 3: D = 1
Case 3
    If D > mar Then M = 4: D = 1
Case 4
    If D > apr Then M = 5: D = 1
Case 5
```

```
    If D > may Then M = 6: D = 1
Case 6
    If D > jun Then M = 7: D = 1
Case 7
    If D > jul Then M = 8: D = 1
Case 8
    If D > agu Then M = 9: D = 1
Case 9
    If D > sep Then M = 10: D = 1
Case 10
    If D > oct Then M = 11: D = 1
Case 11
    If D > nov Then M = 12: D = 1
Case 12
    If D > dec Then M = 1: D = 1: Y = Y + 1
End Select
DateDec=Format$(Y, "00")+"-" + Format$(M, "00")+"-"+Format$(D, "00")
End Function
' 하루를 증가시킨 날짜 반환한다.
Function DateInc() As String
Const jan = 31
Const feb = 28
Const mar = 30
Const apr = 31
Const may = 31
Const jun = 30
Const jul = 31
Const agu = 31
Const sep = 30
Const oct = 31
Const nov = 30
Const dec = 31
Dim Y As Integer, M As Integer, D As Integer
Dim FebD%
Y = Val(Left(BeginDate, 2))                    '년
M = Val(Mid(BeginDate, 4, 2))                  '월
D = Val(Right(BeginDate, 2))                   '일
If M <= 2 Then                                 '윤년검사
    If Y Mod 4 = 0 And Y Mod 100 <> 0 And Y Mod 400 Then
        FebD = 29
    Else
        FebD = 28
    End If
End If
```

```
D = D + 1                                   '날짜증가
Select Case M
Case 1
    If D > jan Then M = 2: D = 1
Case 2
    If D > FebD Then M = 3: D = 1
Case 3
    If D > mar Then M = 4: D = 1
Case 4
    If D > apr Then M = 5: D = 1
Case 5
    If D > may Then M = 6: D = 1
Case 6
    If D > jun Then M = 7: D = 1
Case 7
    If D > jul Then M = 8: D = 1
Case 8
    If D > agu Then M = 9: D = 1
Case 9
    If D > sep Then M = 10: D = 1
Case 10
    If D > oct Then M = 11: D = 1
Case 11
    If D > nov Then M = 12: D = 1
Case 12
    If D > dec Then M = 1: D = 1: Y = Y + 1
End Select
DateInc=Format$(Y, "00")+"-" + Format$(M, "00")+"-" + Format$(D, "00")
End Function

'~~~~~~~~~~~~~~~~~~~~~~~~~~
' 화일에서 변수로
'~~~~~~~~~~~~~~~~~~~~~~~~~~
Sub FileToAllData()
Dim R%
Dim Mystr As String * 13
On Error Resume Next
'파일명 지정하기
FileName = App.Path + "\" + gFileName     '레코드 길이 계산
RecordLen = Len(Item)                      '사용가능한 파일번호 획득
FilePoint = FreeFile                         '랜덤파일 오픈이나 생성
Open FileName For Random As FilePoint Len = RecordLen
'에러면 나감
If Err Then
```

```
    MsgBox Error$, 48, "에러"
    Exit Sub
End If
LastRecord = FileLen(FileName) / RecordLen'파일의 마지막 레코드 번호 파악
If LastRecord = 0 Then                          ' 파일이 생성된 경우 자료가 없다.
    Close FilePoint
    MsgBox "화일에 데이타가 없습니다", 0 + 64, "데이타 리스트"
    Exit Sub
End If
Dim Txt$ ', i%
'화일의 시작에서 끝까지 모두 읽어서
' 레코드들을 리스트 박스에 넣어둔다.
For R = 1 To LastRecord
    Get #FilePoint, R, Item
    '날짜와 시간
    Txt = Item.NowDate + Chr(9) + Item.NowTime + Chr(9)
    Txt = Txt + Format((Item.Item(1)), "00") + Chr(9) '채널
    Mystr = (OX(Item.SelfTest.Working))
    Txt = Txt + Mystr  '작동중
    Mystr = OX(Item.SelfTest.SelfFunc)
    Txt = Txt + Mystr '자체기능
    Mystr = OX(Item.SelfTest.SendWind)
    Txt = Txt + Mystr '송풍기
    Mystr = OX(Item.SelfTest.CutMos)
    Txt = Txt + Mystr '배습기
    Mystr = OX(Item.SelfTest.Berner)
    Txt = Txt + Mystr '버너
    Mystr = OX(Item.SelfTest.HotLevel)
    Txt = Txt + Mystr '고온
    Mystr = OX(Item.SelfTest.HighMos)
    Txt = Txt + Mystr '고습
    Mystr = OX(Item.SelfTest.Senser)
    Txt = Txt + Mystr '감지선
    'For i = 1 To 11
    '    Txt = Txt + Item.Item(i)
    'Next i
    AllData.lst.AddItem Txt
Next R
AllData.Show 1
End Sub

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 16진수를 10진수로 변환하는 함수
Function HexToDecimal(dat As String) As Integer
```

```
Dim tmp%
'2자리
If Len(dat) = 2 Then                    ' 상위 4비트
    tmp = Asc(Mid(dat, 1, 1))
    If tmp >= 48 And tmp < 58 Then
        HexToDecimal = (tmp - Asc("0")) * 16
    Else
        HexToDecimal = (tmp - Asc("A") + 10) * 16
    End If
    tmp = Asc(Mid(dat, 2, 1))           '하위 4비트
    If tmp >= 48 And tmp < 58 Then
        HexToDecimal = HexToDecimal + tmp - Asc("0")
    Else
        HexToDecimal = HexToDecimal + tmp - Asc("A") + 10
    End If
Else ' 1자리
    tmp = Asc(Mid(dat, 1, 1))
    If tmp >= 48 And tmp < 58 Then
        HexToDecimal = HexToDecimal + tmp - Asc("0")
    Else
        HexToDecimal = HexToDecimal + tmp - Asc("A") + 10
    End If
End If
End Function
' 640x480 mode
Sub LargeForm(Frm As Form)
Frm.Width = 640 * Screen.TwipsPerPixelX
Frm.Height = 480 * Screen.TwipsPerPixelY
Frm.Left = 0
Frm.Top = 0
End Sub

Sub main()          '초기시작 프로시져
FormMain.StarInit    ' 별의 초기화
FormMain.Show      ' 시작화면을 부른다.
End Sub
Function CX(Domain As Boolean) As String
If Domain = True Then
    CX = "C"
Else
    CX = "X"
End If
End Function
```

```
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 팩형식의 수를 16진술로 변환
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Function PackToHex(dat As String) As String
Dim Hexa As String
Hexa = Format(Hex(Asc(dat)), "00")
PackToHex = Hexa
End Function
' 시간간격을 체크한다.
' 시간간격은 초단위로 빼거나 더한다.
Function TimeGap(ScrTime As String, Gap As Integer) As String
Dim HTime%, MTime%, STime%, ScrH%, ScrM%, ScrS%
' 원시분초
ScrH = Val(Left(ScrTime, 2))
ScrM = Val(Mid(ScrTime, 4, 2))
ScrS = Val(Right(ScrTime, 2))
'분 간격
HTime = Gap / 3600
MTime = (Gap Mod 3600) / 60
STime = Gap Mod 60
' 계산
ScrS = ScrS + STime
ScrM = ScrM + MTime
ScrH = ScrH + HTime
If ScrS < 0 Then '초간격
    ScrM = ScrM - 1 + ScrS \ 60: ScrS = 60 + ScrS Mod 60
ElseIf ScrS >= 60 Then
    ScrM = ScrM + ScrS \ 60: ScrS = ScrS Mod 60
End If
If ScrM < 0 Then ' 분 간격
    ScrH = ScrH - 1 + ScrM \ 60: ScrM = 60 + ScrM Mod 60
ElseIf ScrM >= 60 Then
    ScrH = ScrH + ScrM \ 60: ScrM = ScrM Mod 60
End If
If ScrH < 0 Then ScrH = 24 + ScrH
If ScrH >= 24 Then ScrH = 0
TimeGap = Format$(ScrH, "00") + ":" + Format$(ScrM, "00") + ":" +
Format$(ScrS, "00")
End Function

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자 파일을 읽어서 사용자 배열변수에 저장한다.
' 사용자는 최대 20명 까지 등록된다.
' true  성공적으로 파일을 읽음 false  파일읽기 실패
```

```
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Function UserFileRead() As Boolean
Dim i%
On Error Resume Next
FileName = App.Path + "\" + "User.dat"   '사용자 정의 데이타 파일
RecordLen = Len(User(1))                '레코드 길이 계산
FilePoint = FreeFile                     '사용가능한 파일번호 획득
'랜덤파일 오픈이나 생성
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
UserMax = Int(FileLen(FileName) / RecordLen)
If UserMax = 0 Then
    UserFileRead = False
    Exit Function
End If
For i = 1 To UserMax                     ' 파일에 한 레코드 저장
    Get #FilePoint, i, User(i)
Next i
Close FilePoint                           '파일을 제거 한다.
UserFileRead = True
End Function

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자 파일에 변수의 내용을 작성한다.
' true  사용자 파일에 쓰기 성공 false 파일에 쓰기 실패
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Sub UserFileWrite(tmp As UserInfo)
On Error Resume Next
FileName = App.Path + "\" + "User.dat"   '사용자 정의 데이타 파일
RecordLen = Len(tmp)                    '레코드 길이 계산
FilePoint = FreeFile                     '사용가능한 파일번호 획득
'랜덤파일 오픈이나 생성
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
UserMax = Int(FileLen(FileName) / RecordLen)
' 한 레코드 증가를 위해 변수의 값 증가
UserMax = UserMax + 1
Put #FilePoint, UserMax, tmp    ' 파일에 한 사용자 내용을 쓴다.
Close FilePoint                  ' 파일 닫기
End Sub

'---------------------------
' 중앙감시모드에서 사용한다.
' 변수의 값을 채널별로 화면에 출력한다.
```

```
Sub VarToChannelData()
Dim MyVar As Integer
'텍스트 박스에 값을 채운다.
' 채널별로 TCha#(0 ~ 6) 의 변수명을 가진다.
If Item.Item(1) = "0A" Then '채널번호
    MyVar = 10
Else
    MyVar = Val(Item.Item(1))
End If
' 채널별로 데이타를 출력한다.
'해당채널의 캡션을 파란색으로..
If MyVar < 0 Or MyVar > 10 Then Exit Sub
If Flag = Center Then
    FCenter.Fcha(MyVar).BackColor = &HC0C0C0
    FCenter.TCh2(MyVar).Text = Item.Item(2) '모델번호
    FCenter.TCh3(MyVar).Text = Item.Item(3) '건조단계/
    FCenter.TCh4(MyVar).Text = HexToDecimal(Item.Item(4)) '단계경과시간
    FCenter.TCh5(MyVar).Text = Item.Item(5) '건구온도
    FCenter.TCh6(MyVar).Text = Item.Item(6) '습구온도
    FCenter.TCh10(MyVar).Text = HexToDecimal(Item.Item(10)) '총경과시간
    FCenter.TCh7(MyVar).Text = Item.Item(7) '자체점검
'    If Item.Item(7) <> "FE" Then
'        FCenter.TCh7(MyVar).BackColor = QBColor(vbRed)
'    Else
'        FCenter.TCh7(MyVar).ForeColor = QBColor(vbBlue)
'    End If
ElseIf Flag = LocalForm Then
    MyVar = Val(Item.Item(1))
    If MyVar = ChannelHexa Then
       VarToCeLocal
    End If
End If
End Sub

'~~~~~~~~~~~~~~~~~~~~~~~~~
' 변수에서 화일로
'~~~~~~~~~~~~~~~~~~~~~~~~~
Sub VarToFile()
On Error Resume Next
'파일명 지정하기
FileName = App.Path + "\" + Trim(User(UserNumber).FileName)
RecordLen = Len(Item)           '레코드 길이 계산
FilePoint = FreeFile             '사용가능한 파일번호 획득
'랜덤파일 오픈이나 생성
```

```
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
LastRecord = FileLen(FileName) / RecordLen
If LastRecord = 0 Then          '파일이 생성되었을 경우
    LastRecord = 1
Else ' 데이타의 한 레코드 하나 추가
    LastRecord = LastRecord + 1
End If
Put #FilePoint, LastRecord, Item   ' 파일에 한 레코드 저장
Close FilePoint                   '파일을 제거 한다.
End Sub

'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 화면에 변수의 11바이트 출력
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Sub VarToScreen()
Dim i%, tmp$
FLocal.t1.Text = Item.Item(1)
FLocal.t2.Text = Item.Item(2)
FLocal.t3.Text = Item.Item(3)
FLocal.t4.Text = HexToDecimal(Item.Item(4))
FLocal.t5.Text = Item.Item(5)
FLocal.t6.Text = Item.Item(6)
'자체점검
FLocal.t7.Text = Item.Item(7)
For i = 0 To 7
    FLocal.t70(i).Value = False
'     FLocal.see(i).BackColor = QBColor(8)
Next i
'작동중
If HexToDecimal(Item.Item(7)) And 1 Then
    FLocal.t70(0).Value = False
'    FLocal.see(0).BackColor = QBColor(8)
    Item.SelfTest.Working = False
Else
 '   FLocal.see(0).BackColor = QBColor(12)
    FLocal.t70(0).Value = True
    Item.SelfTest.Working = True
End If
'자체기능
If HexToDecimal(Item.Item(7)) And 2 Then
    FLocal.t70(1).Value = False
  '  FLocal.see(1).BackColor = QBColor(8)
    Item.SelfTest.SelfFunc = False
```

```
Else
   ' FLocal.see(1).BackColor = QBColor(12)
    FLocal.t70(1).Value = True
    Item.SelfTest.SelfFunc = True
End If
'송풍기
If HexToDecimal(Item.Item(7)) And 4 Then
  '  FLocal.see(2).BackColor = QBColor(8)
    FLocal.t70(2).Value = False
    Item.SelfTest.SendWind = False
Else
 '   FLocal.see(2).BackColor = QBColor(12)
    FLocal.t70(2).Value = True
    Item.SelfTest.SendWind = True
End If
'배습기
If HexToDecimal(Item.Item(7)) And 8 Then
  '  FLocal.see(3).BackColor = QBColor(8)
    FLocal.t70(3).Value = False
    Item.SelfTest.OutMos = False
Else
   ' FLocal.see(3).BackColor = QBColor(8)
    FLocal.t70(3).Value = True
    Item.SelfTest.OutMos = True
End If
'버너
If HexToDecimal(Item.Item(7)) And 16 Then
 '   FLocal.see(4).BackColor = QBColor(8)
    FLocal.t70(4).Value = False
    Item.SelfTest.Berner = False
Else
  '  FLocal.see(4).BackColor = QBColor(12)
    FLocal.t70(4).Value = True
    Item.SelfTest.Berner = True
End If
'고온
If HexToDecimal(Item.Item(7)) And 32 Then
   ' FLocal.see(5).BackColor = QBColor(8)
    FLocal.t70(5).Value = False
    Item.SelfTest.HotLevel = False
Else
'    FLocal.see(5).BackColor = QBColor(12)
    FLocal.t70(5).Value = True
    Item.SelfTest.HotLevel = True
```

```
End If
'고습
If HexToDecimal(Item.Item(7)) And 64 Then
 '   FLocal.see(6).BackColor = QBColor(8)
    FLocal.t70(6).Value = False
    Item.SelfTest.HighMos = False
Else
 '   FLocal.see(6).BackColor = QBColor(12)
    FLocal.t70(6).Value = True
    Item.SelfTest.HighMos = True
End If
'감지선
If HexToDecimal(Item.Item(7)) And 128 Then
 '   FLocal.see(7).BackColor = QBColor(8)
    FLocal.t70(7).Value = False
    Item.SelfTest.Senser = False
Else
 '   FLocal.see(7).BackColor = QBColor(12)
    FLocal.t70(7).Value = True
    Item.SelfTest.Senser = True
End If
'----------
'배습기
FLocal.t80.Text = Item.Item(8)
tmp = Item.Item(8)
For i = 0 To 2
    FLocal.t8(i).Value = False
Next i
FLocal.t81.Value = False
' 열림
If Not (Val(Asc(Right(tmp, 1))) And 6) Then FLocal.t8(0).Value = True
'반열림
If (Val(Asc(Right(tmp, 1))) And 2) And Not Val(Asc((Right(tmp, 1))) And 4)
Then FLocal.t8(1).Value = True
'닫힘
If Val(Asc(Right(tmp, 1))) And 6 Then FLocal.t8(2).Value = True
'버너on
If  Val(Asc(Right(tmp,  1)))  And  1  Then  FLocal.t81.Value  =  True  Else
FLocal.t81.Value = False
'버너off
'출력상태
FLocal.t9.Text = Item.Item(9)
FLocal.t10.Text = HexToDecimal(Item.Item(10))    '총경과시간
FLocal.t11.Text = Item.Item(11)    '합계 - 16진수를 출력한다.
```

```
End Sub

--------------Form alldata---------
Private Sub Exit_Click()
Unload Me
End Sub
Private Sub Form_Load()
'사용자 지정파일에서 내용을 읽어온다.
' 결국 이제까지 통신으로 인해서 기록된
'내용으로서 화면에 나타나게 된다.
Call CenterForm(Me)
Me.Show
Me.Caption = "화일에서 읽어온 데이타.." + Format$(Now, " hh:mm:ss")
Label4.Caption = "작동중 자체기능 송풍기 배습기 버 너 고 온 고 습 감지선"
' 화일에서 자료를 읽어온다.
FileToAllData
End Sub
Private Sub Form_Unload(Cancel As Integer)
'리스트를 모두 비운다.
lst.Clear
'우주여행끄기
FormMain.StarSimulOn
End Sub
Private Sub Timer1_Timer()
Me.Caption = "화일에서 읽어온 데이타.." + Format$(Now, " hh:mm:ss")
End Sub

----------Form config--------------
' Communication Settings Configuration Form
DefInt A-Z
' Cancel button actions.
Private Sub CancelButton_Click()
    Unload Config
End Sub
Private Sub ComPort_Click(Index As Integer)
    NewPort = Index
End Sub
' Initialize and display the configuration form.
Private Sub Form_Load()
   Call CenterForm(Me)
   ' Get the current port.
    Port = MSComm1.CommPort
    Config.ComPort(Port).Value = True   ' Set the option button.
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
'우주여행끄기
FormMain.StarSimulOn
End Sub
' No handshaking option button.
Private Sub NoFlow_Click()
    NewShake = 0
End Sub
' No Parity option button.
Private Sub NoParity_Click()
    NewParity$ = "N"
End Sub

-----form fcenter----
' 중앙감시모드는 R을 주고 받으면서
'초기화를 한다.
Sub InitComm()
Dim InPutStr As String
Dim i As Byte, j As Byte, a%
Dim Count As Integer
If FormMain.MSComm1.PortOpen = False Then
    TRev.Enabled = False
    Exit Sub
End If
InPutStr = ""
FormMain.MSComm1.InputLen = 0
FormMain.MSComm1.Cutput = "R" '데이타를 요구한다.
Do
    a = DoEvents()
    If gEnd = True Then
        Exit Sub
    End If
Loop Until FormMain.MSComm1.InBufferCount >= 1
End Sub
Private Sub Fcha_DblClick(Index As Integer)
' 해당채널의 지역감시모드체제로 제어를 이동한다.
FromIs = FromForm
OldFrom = FromMenu
Me.Hide                 '감추고 인덱스값을 넘겨준다.
'TRev.Enabled = False
ChannelHexa = Index ' 채널의 범위는 1 ~ 10
Flag = LocalForm
TRev.Enabled = False
FLocal.Show '지역감시모드 로딩...
```

```
End Sub
Private Sub Form_Load()
Call CenterForm(Me)
Me.Show
' 초시간 설정
gPreT = Timer
Flag = Center
End Sub
Private Sub Form_Unload(Cancel As Integer)
TRev.Enabled = False: Timer1.Enabled = False
' 포트를 닫는다.
If FormMain.MSComm1.PortOpen = True Then
FormMain.MSComm1.PortOpen = False
If Err Then MsgBox Error$, 48
OldFrom = FromMenu
FormMain.StarSimulOn    '우주여행켜기
Flag = None
End Sub
Private Sub mData_Click()
'타이머를 셋하고 'R"이라는 제어 명령을
' 송신한다.  ' 타이머 인터벌 간격은 5초로 한다.
' 즉, 5초 마다 하나의 모델에 대한' 데이타 11바이트가 온다./
If FormMain.MSComm1.PortOpen = False Then
    FormMain.MSComm1.PortOpen = True
End If
If TRev.Enabled = True Then Exit Sub
If Err Then
    MsgBox Error, 16, "통신에러메시지"
    Exit Sub
End If
Call InitComm '통신초기화 R을 주고 받고 R을 보내고 시작
TRev.Enabled = True        ' 타이머 시작
gEnd = False
End Sub
Private Sub mExit_Click()
gEnd = True
Unload Me
End Sub
Private Sub Timer1_Timer()
' now time print
Me.Caption = Clock("중앙감시체제 ===> 비가동채널 =   흰색")
End Sub
Private Sub TRev_Timer()
' 5초 단위로 데이타를 수신하여 화면과 파일에 나타내거나 저장한다.
```

```
Dim InPutStr As String
Static tempstr As String * 11
Dim i As Byte, j As Byte, a%
Dim Count As Integer
If FormMain.MSComm1.PortOpen = False Then
    TRev.Enabled = False
    Exit Sub
End If
' 3초간격의 데이타 수신을 위한 시간 계산
gNowT = Timer '오늘의 초시간
If Abs(gNowT - gPreT) < 30 Then
    Exit Sub
Else
    gPreT = Timer
End If
InPutStr = ""
FormMain.MSComm1.InputLen = 0
FormMain.MSComm1.Cutput = "R" '데이타를 요구한다.
' REQUEST 'R'
checkiT:
Do
    a = DoEvents()
    If gEnd = True Then Exit Sub
Loop Until FormMain.MSComm1.InBufferCount >= 1
If Right$(FormMain.MSComm1.Input, 1) <> "R" Then GoTo checkiT
FormMain.MSComm1.Cutput = "R" '데이타를 요구한다.
     FormMain.MSComm1.InputLen = 0
'11바이트를 체크하는 부분
Do
    a = DoEvents()
    If gEnd = True Then Exit Sub
Loop Until FormMain.MSComm1.InBufferCount >= 11
tempstr = FormMain.MSComm1.Input
Count = Len(tempstr) ' count는 11이어야 겠죠...
j = 1
For i = 1 To Count
    If IsNull(Mid(tempstr, i, 1)) Then
        Item.Item(j) = "00"
    ElseIf Mid(tempstr, i, 1) = "" Then
        Item.Item(j) = "00"
    Else
        InPutStr = PackToHex(Mid(tempstr, i, 1))
        Item.Item(j) = Left$(InPutStr, 2)
        If Len(InPutStr) > 2 Then
```

```
            j = j + 1
            Item.Item(j) = Right(InPutStr, 2)
        End If
    End If
    j = j + 1
Next i
ComToVar InPutStr '변수에 값을 저장
'화일로 저장한다....화일명은 사용자가 지정한 파일명
VarToFile
VarToChannelData ' 채널별로 데이타를 보여준다.
End Sub

-----form formmain------
Private Type STARINFO   ' 별의 구조체
    Speed As Long '속도
    RGB As Long ' RGB는 색지정값
    X As Long 'x좌표
    Y As Long 'y좌표
    Cldx As Long
    Cldy As Long
End Type
' 별의 선언
Const StarMax = 30
Dim Star(0 To StarMax) As STARINFO
Sub StarInit()
'거리는 100에서 150 사이 로 속도를 빼면된다.
'속도는 1 에서 20사이로 1씩 높아진다.
' x 좌표는 정중앙에서 +-20
' y 좌표는 정중아에서 +-20
' 흰색 RGB가 255,255,255
Dim i%
Randomize Timer
' 별의 속성 초기화
For i = 0 To StarMax
    Star(i).Speed = Rnd * 10
    Star(i).RGB = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
    If i Mod 2 Then Star(i).X = Rnd * 350 Else Star(i).X = -(Rnd * 350)
    Star(i).Y = -300 + Rnd * 600
    Star(i).Oldx = Star(i).X: Star(i).Oldy = Star(i).Y
Next i
End Sub
Sub StarSimul()
' 별의 거리는 100 ~ 150 으로
' 거리 - 속도로 거리가 0보다 작으면 또는
```

```
' 화면의 밖으로 벗어나면 사라진다.즉, 다시시작..
'순서는 먼저 출력한다.    다음 새로 초기화한다.
' 이전의 위치를 흰점으로 찍고 새로 점을 찍는다.
Dim i%, Cx%, Cy%
Cx = FormMain.ScaleWidth / 2
Cy = FormMain.ScaleHeight / 2
Randomize Timer
'별의출력
For i = 0 To StarMax
    ' 별지우기
    FormMain.Line (Cx + Star(i).Cldx - 1, Cy + Star(i).Cldy - 1)-(Cx +
Star(i).Cldx + 1, Cy + Star(i).Cldy - 1), RGB(0, 0, 0)
    FormMain.Line (Cx + Star(i).Oldx - 2, Cy + Star(i).Cldy)-(Cx +
Star(i).Cldx + 2, Cy + Star(i).Cldy), RGB(0, 0, 0)
    FormMain.Line (Cx + Star(i).Cldx - 1, Cy + Star(i).Cldy + 1)-(Cx +
Star(i).Cldx + 1, Cy + Star(i).Cldy + 1), RGB(0, 0, 0)
    '별그리기
    FormMain.Line (Cx + Star(i).X - 1, Cy + Star(i).Y - 1)-(Cx + Star(i).X
+ 1, Cy + Star(i).Y - 1), Star(i).RGB
    FormMain.Line (Cx + Star(i).X - 2, Cy + Star(i).Y)-(Cx + Star(i).X +
2, Cy + Star(i).Y), Star(i).RGB
    FormMain.Line (Cx + Star(i).X - 1, Cy + Star(i).Y + 1)-(Cx + Star(i).X
+ 1, Cy + Star(i).Y + 1), Star(i).RGB
    Star(i).Oldx = Star(i).X: Star(i).Oldy = Star(i).Y
    ' 별의위치 변경, 속도와 거리변경
    If Star(i).X >= 0 Then
        Star(i).X = Star(i).X + (Star(i).X) * 0.06 * Star(i).Speed
    Else
        Star(i).X = Star(i).X + (Star(i).X) * 0.06 * Star(i).Speed
    End If
    If Star(i).Y >= 0 Then
        Star(i).Y = Star(i).Y + (Star(i).Y) * Star(i).Speed * 0.06
    Else
        Star(i).Y = Star(i).Y + (Star(i).Y) * Star(i).Speed * 0.06
    End If
    ' 스피드 업
    Star(i).Speed = Star(i).Speed + 1
    '초기화할 것 선택
    If (Star(i).X) < -(FormMain.ScaleWidth / 2) Or Star(i).X >
FormMain.ScaleWidth Or Star(i).Y < -(FormMain.ScaleHeight) Or Star(i).Y >
FormMain.ScaleHeight Then
        Star(i).Speed = Rnd * 7
        Star(i).RGB = RGB(Rnd * 255, Rnd * 255, Rnd * 255)
        If i Mod 2 Then Star(i).X = Rnd * 350 Else Star(i).X = -(Rnd *
```

```
350)
        Star(i).Y = -300 + Rnd * 600
        'Star(i).CldX = Star(i).x: Star(i).CldY = Star(i).y
    End If
Next i
End Sub
Sub StarSimulOff()
FormMain.Timer.Enabled = False     ' 우주여행을 끄기
End Sub
Sub StarSimulOn()
FormMain.Timer.Enabled = True     '우주여행 재개
End Sub
Private Sub Form_Load()
' 화면의 크기
Call LargeForm(Me)
Call CenterForm(Me)
FormMain.Show
OldFrom = FromMenu
End Sub
Private Sub Form_Unload(Cancel As Integer)
End
End Sub
' 사용자 전체보기
Private Sub mAllUser_Click()
UserMenu = USERVIEW
StarSimulOff           '우주여행끄기
FUser.Show
End Sub
Private Sub mCenter_Click()
FromIs = FromMenu '호출부분
StarSimulOff          '우주 여행끄기
FCenter.Show         '중앙감시제체
End Sub
Private Sub mDel_Click()  '사용자삭제
UserMenu = USERDEL   ' 사용자 삭제 이벤트
StarSimulOff             '우주여행끄기
FUser.Show
End Sub
Private Sub mEnrol_Click()  '사용자 설정
UserMenu = UserSet
StarSimulOff               '우주여행끄기
fUserSet.Show
End Sub
Private Sub mExit_Click()
```

```
Unload Me
End Sub
Private Sub mGraph_Click()
Dim Ret$
Ret = InputBox("데이타 파일명을 입력하세요.", "상황 그래프 보기")
If Len(Ret) = 0 Then
    Exit Sub
End If
gFileName = Ret
FromIs = FromMenu '호출부분
StarSimulOff         '우주여행끄기
End Sub
Private Sub mLocal_Click()
FromIs = FromMenu '호출부분
Call StarSimulCff
FLocal.Show             '지역감시체제로드
End Sub
Private Sub mNowUser_Click()
Dim Msg$
On Error Resume Next
Msg = "사용자 성명 :" + User(UserNumber).Name + Chr(13) + Chr(13) + "기록
파일명:" + User(UserNumber).FileName
If Err Then
    MsgBox "사용자선택을 하지않았습니다", 16, "에러메시지"
    Exit Sub
End If
MsgBox Msg, 48, "현재사용자는..."
End Sub
Private Sub mnuDial_Click()
Dim Mystr$
Mystr = InputBox("", "전화번호를 입력하세요...", "5424")
If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
MSComm1.Cutput = "atdt" + Mystr + Chr$(13)
End Sub
Private Sub mnuoff_Click()
If MSComm1.PortOpen = True Then
    MSComm1.Output = "3" + Chr$(13)
End If
End Sub
Private Sub mReg_Click()    '기록상황보기
Dim Ret$
Ret = InputBox("데이타 파일명을 입력하세요.", "데이타 파일 보기")
If Len(Ret) = 0 Then
    Exit Sub
```

```
End If
StarSimulOff          '우주여행 끄기
gFileName = Ret       '파일을 읽어들인다.
FileToAllData
End Sub
Private Sub mSel_Click()  '사용자 선택
UserMenu = USERSEL
StarSimulOff              '우주여행끄기
FUser.Show
End Sub
Private Sub mSet_Click()
StarSimulOff ' 우주여행 끄기
FromIs = FromMenu
' Show the communications settings form.
Config.Show 1
StarSimulOn '우주여행켜기
End Sub
Private Sub Timer_Timer()
Me.Caption = "아리랑 Ver2.0    오늘은 " + Format$(Now, "yy년mm월dd일") + "
입니다."
StarSimul                 '우주여행
End Sub

----form fuserset------
Private Sub cCancel_Click()
tName.Text = ""           ' 비우기
tFile.Text = ""
tPhone.Text = ""
tAddr.Text = ""
Unload Me
End Sub
Private Sub cCk_Click()
Dim tmp As UserInfo
' 이름과 파일명이 작성되었는지 확인
If Len(Trim(tName.Text)) > 0 And Len(Trim(tFile.Text)) > 0 Then
    tmp.Name = tName.Text
    tmp.FileName = tFile.Text
    tmp.Phone = tPhone.Text
    tmp.Address = tAddr.Text
    UserFileWrite tmp                 ' 파일에 사용자 한 레코드 작성
    tName.Text = ""                   ' 비우기
    tFile.Text = ""
    tPhone.Text = ""
    tAddr.Text = ""
```

```
Else
    MsgBox "성명이나 파일명을 입력하지 않으셨습니다.", 32, "경고 메시지"
End If
Unload Me
End Sub
Private Sub Form_Load()
'~~~~~~~~~~~~~~~~
' 사용자 설정
'~~~~~~~~~~~~~~~~
Me.Show
tName.Text = ""
tFile.Text = ""
tPhone.Text = ""
tAddr.Text = ""
End Sub
Private Sub Form_Unload(Cancel As Integer)
'우주여행끄기
FormMain.StarSimulOn
End Sub
----------------form local-----------------
Private Sub CData_Click()
'타이머를 셋하고 'R"이라는 제어 명령을
' 송신한다.    타이머 인터벌 간격은 5초로 한다.
' 즉, 5초 마다 하나의 모델에 대한 데이타 11바이트가 온다./
If Flag = LocalForm Then
    If FormMain.MSComm1.PortOpen = False Then
        FormMain.MSComm1.PortOpen = True
    End If
    FCenter.TRev.Enabled = True        ' 타이머 시작
Else
    If TRev.Enabled = True Then Exit Sub

    If FormMain.MSComm1.PortOpen = False Then
        FormMain.MSComm1.PortOpen = True
    End If
    TRev.Enabled = True          ' 타이머 시작
End If
If Err Then
    MsgBox Error, 16, "통신에러메시지"
    Exit Sub
End If
End Sub
Private Sub Exit_Click()
gEnd = True
```

```
Unload Me
End Sub
Private Sub Form_Load()
Call CenterForm(Me)
Me.Show
'단계설정온도의 설정값을 미리 출력한다.
'For i = 1 To 10
'    LStep.AddItem Str(i) + ".  "
'Next i
gPreT = Timer '초시간 초기화
gEnd = False
If Flag = LocalForm Then
    If FormMain.MSComm1.PortOpen = False Then
        FormMain.MSComm1.PortOpen = True
    End If
    FCenter.TRev.Enabled = True        ' 타이머 시작
    CData.Visible = False
End If
End Sub
Private Sub Form_Unload(Cancel As Integer)
TRev.Enabled = False: Timer1.Enabled = False
If FromIs = FromMenu Then '메뉴로 부터의 콜
    FormMain.StarSimulCn        '우주여행on
False ' 중앙감시모드에서 콜 했었을 경우
    '중앙감시모드로 간다.
    FCenter.Visible = True
    If Flag = LocalForm Then Flag = Center
    FCenter.TRev.Enabled = False
End If
End Sub
Private Sub Timer1_Timer()
 Me.Caption = Clock("지역감시체제")
End Sub
 Private Sub TRev_Timer()
' 5초 단위로 데이타를 수신하여 화면과 파일에 나타내거나 저장한다.
Dim InPutStr As String
Static tempstr As String * 11
Dim i As Byte, j As Byte, a%
Dim Count As Integer
If FormMain.MSComm1.PortOpen = False Then
    TRev.Enabled = False
    Exit Sub
End If
' 3초간격의 데이타 수신을 위한 시간 계산
```

```
gNowT = Timer '오늘의 초시간
If Abs(gNowT - gPreT) < 30 Then
    Exit Sub
Else
    gPreT = Timer
End If
InPutStr = ""
FormMain.MSComm1.InputLen = 0
FormMain.MSComm1.Cutput = "R" '데이타를 요구한다.
'11바이트를 체크하는 부분
Do
    a = DoEvents()
    If gEnd = True Then Exit Sub

Loop Until FormMain.MSComm1.InBufferCount >= 11
tempstr = FormMain.MSComm1.Input
Count = Len(tempstr) ' count는 11이어야 겠죠...
j = 1
For i = 1 To Count
    If IsNull(Mid(tempstr, i, 1)) Then
        Item.Item(j) = "00"
    ElseIf Mid(tempstr, i, 1) = "" Then
        Item.Item(j) = "00"
    Else
        InPutStr = PackToHex(Mid(tempstr, i, 1))
        Item.Item(j) = Left$(InPutStr, 2)
        If Len(InPutStr) > 2 Then
            j = j + 1: Item.Item(j) = Format(Right(InPutStr, 2), "00")
        End If
    End If
    j = j + 1
Next i
' 중앙감시체제에서 로드되었을 경우
' 해당 채널만이 유효한 값이다.
If FromIs = FromForm Then
    If Item.Item(1) <> ChannelHexa Then
        Exit Sub
    End If
End If
ComToVar InPutStr '변수에 값을 저장
'화일로 저장한다....화일명은 사용자가 지정한 파일명
VarToFile
If Flag = LocalForm Then
    VarToCeLocal
```

```
Else
    VarToScreen ' 화면으로 데이타 출력
End If
End Sub

----------------form user-----------------
Private Sub cBtn_Click()
Select Case UserMenu
Case USERSEL ' 사용자 선택
    ' 선택된 사용자 지정
    If lst.ListIndex = -1 Then
        MsgBox "사용자가 선택되지 않았습니다.", 32, "메시지"
        Exit Sub
    End If
    UserNumber = lst.ListIndex + 1
Case USERDEL '사용자 삭제
    '선택된 사용자를 삭제한다.
    If lst.ListIndex = -1 Then
        MsgBox "삭제될 사용자가 지정되지 않았습니다", 32, "메시지"
        Exit Sub
    End If
    UserDelete lst.ListIndex + 1   ' 사용자를 삭제한다. 나머지는 당긴다.
Case USERVIEW '사용자 보기
End Select
Unload Me
End Sub
Private Sub cCancel_Click()
Unload Me
End Sub
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자 폼 시작
Private Sub Form_Load()
Dim i%, b As Boolean
Dim Txt$
Static tabs(5) As Long
Call CenterForm(Me)
Me.Show
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
' 사용자가 선택한 메뉴의 기능을 조사하여
' 함수를 수행한다.
Select Case UserMenu
Case USERSEL ' 사용자 선택
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    ' 사용자를 사용자 파일에서 읽어서
```

```
    ' 사용자를 선택한다.선택후에는 UserNumber 변수의
    ' 값이 선택한 번호로 변경된다.
    '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    cBtn.Caption = "선 택"
    b = UserFileRead
Case USERDEL '사용자 삭제
    '~~~~~~~~~~~~~~~~~~~~~~~~
    ' 사용자를 사용자 파일에서 읽어서
    ' 원하는 사용자를 삭제한다.
    ' 삭제선택후, UserNumber변수가
    ' 변화되어 있다. 단, 현재 사용자는 삭제할 수 없다.
    cBtn.Caption = "삭 제"
    b = UserFileRead
Case USERVIEW '사용자 보기

    '~~~~~~~~~~~~~~~~~~~~~~~~~
    ' 사용자를 사용자 파일에서 읽어서
    ' 단지 보여주기 기능을 한다.
    cBtn.Caption = "메인으로"
    cCancel.Visible = False
    b = UserFileRead
End Select
If b = False Then
    MsgBox "등록된 사용자가 없군요", 16, "사용자메시지"
    Exit Sub
End If
For i = 1 To UserMax
    Txt = User(i).Name + Chr$(9) + User(i).FileName + Chr$(9) +
User(i).Phone + Chr$(9) + User(i).Address
    lst.AddItem Txt
Next i
tabs(1) = 0                    ' 탭 위치 지정
tabs(2) = 30
tabs(3) = 45
tabs(4) = 80
tabs(5) = 130
' 탭에 대한 메시지 전송
i = SendMessage(lst.hwnd, LB_SETTABSTOPS, 5, tabs(1))
End Sub
Sub SaveUser()
' 사용자 변수를 파일로 저장한다.
Dim i%
On Error Resume Next
FileName = App.Path + "\" + "User.dat"    '사용자 정의 데이타 파일
```

```
RecordLen = Len(User(1))       '레코드 길이 계산
FilePoint = FreeFile             '사용가능한 파일번호 획득
'랜덤파일 오픈이나 생성
Open FileName For Random As FilePoint Len = RecordLen
'파일의 마지막 레코드 번호 파악
i = 1
While Trim(User(i).Name) <> ""
    Put #FilePoint, i, User(i)
    UserMax = i: i = i + 1
Wend
Close FilePoint     ' 파일 닫기
End Sub
Sub UserDelete(ByVal Number As Integer)
Dim i%, DelFile$
On Error Resume Next '에러 처리
DelFile = Trim(User(Number).FileName)
' 해당 번호의 사용자를 삭제한다.
If Number = UserNumber Then
    MsgBox "현재 사용자를 삭제할 수 없습니다.", 32, "경고메시지"
    Exit Sub
' 마지막 사용자번호와 같을 경우
ElseIf Number = UserMax Then
    User(Number).Name = ""
    User(Number).FileName = ""
    User(Number).Phone = ""
    User(Number).Address = ""
' 현재 사용자 번호보다 작을 경우
ElseIf Number < UserNumber Then
    For i = Number To UserMax - 1
        User(i).Name = User(i + 1).Name
        User(i).FileName = User(i + 1).FileName
        User(i).Phone = User(i + 1).Phone
        User(i).Address = User(i + 1).Address
    Next i
    UserNumber = UserNumber - 1
    UserMax = UserMax - 1
'현재 사용자 번호보다클 경우
ElseIf Number > UserNumber Then
    For i = Number To UserMax - 1
        User(i).Name = User(i + 1).Name
        User(i).FileName = User(i + 1).FileName
        User(i).Phone = User(i + 1).Phone
        User(i).Address = User(i + 1).Address
    Next i
```

```
    UserMax = UserMax - 1
End If
lst.RemoveItem lst.ListIndex   '리스트 박스에서 재거한다.
Kill App.Path + "\" + "User.dat"   '갱신된 사용자 변수를 저장
If Err Then
    MsgBox Error$, 48, "Error Occure"
    Exit Sub
End If
Call SaveUser
Kill App.Path + "\" + DelFile  '사용자 링크파일 삭제
If Err Then
    MsgBox Error$, 48, "Error Occure"
    Exit Sub
End If
End Sub
Private Sub Form_Unload(Cancel As Integer)
FormMain.StarSimulOn   '우주여행끄기
End Sub
Private Sub lst_DblClick()
Call cBtn_Click ' 버튼을 클릭한 것과 같다.
End Sub

----Module1 vbterm--------
DefInt A-Z
#If Win32 Then
    Declare Sub SetWindowPos Lib "user32" (ByVal hwnd As Long, ByVal
hWndInsertAfter As Long, ByVal X As Long, ByVal Y As Long, ByVal Cx As
Long, ByVal Cy As Long, ByVal wFlags As Long)
#Else
    Declare Sub SetWindowPos Lib "User" (ByVal hwnd%, ByVal
hWndInsertAfter%, ByVal X%, ByVal Y%, ByVal Cx%, ByVal Cy%, ByVal wFlags%)
#End If
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd
As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Long) As Long
```

■첨부2) 건조기 전용 모뎀소프트 웨어.

```
;PROGRAM MODEMG.ASM 96.9.30
$INCLUDE (8096.INC)
        ; DEFINE SYMBOL
        ; OUTPUT RELAY SYMBOL
          RSEG    AT      1AH

        DAX:            DSL     1
        AX              EQU     DAX:WORD
        AL              EQU     AX:BYTE
        AH              EQU     AX+1:BYTE
        DX              EQU     DAX+2:WORD
        DL              EQU     DX:BYTE
        DH              EQU     DX+1:BYTE
        BX:             DSW     1
        BL              EQU     BX:BYTE
        BHH             EQU     BX+1:BYTE
        CX:             DSW     1
        CL              EQU     CX:BYTE
        CH              EQU     CX+1:BYTE

        IM_SPCON:       DSB     1

        NO_CONT_TIME1:  DSW     1
        NO_CONT_TIME2:  DSW     1

        RD_POINTER:     DSW     1
        WR_POINTER:     DSW     1

        CHECK_SUM:      DSB     1

        DO_MODEL_NO:    DSB     30

        ;------------------------------
        DUMMY1:         DSW     1

        BUF_START:      DSB     20
        BUF_END:        DSB     1
        DUMMY2:         DSW     20
        SP_POINTER:     DSW     1

        ;------------------------------
```

```
        P8255CE         EQU     0C010H
        G244            EQU     0C014H          ;& !RD

        PIC1            EQU     0C000H          ;& !WR
        PIC2            EQU     0C004H          ;& !WR

        SW_LOW          EQU     0C014H
        SW_HIGH         EQU     ICPCRT0


;***************************************
CSEG            AT      2000H
VECT:           DCW     2080H
                DCW     2080H
                DCW     2080H
                DCW     2080H
                DCW     2080H
                DCW     2080H
                DCW     INT_RXDATA
                DCW     2080H
CSEG            AT      2018H
CCR:            DCB     0FDH

CSEG            AT      2080H           ;RESET
;***********************************

START:

        LD      SP,#SP_POINTER

        CALL    SIO_INIT
        LDB     INT_MASK,#01000000B
        CLRB    INT_PENDING

        EI

        CALL    RESET_MODEM
        CALL    MODEM_INIT
        CALL    SDELAY

        CALL    SEND_AT
        CALL    SDELAY

        CALL    SEND_ATV0E0
```

```
        CALL    DELAY

        LD      BX,#ATX1
        CALL    SEND_COMMAND_MODEM
        JE      START

        CALL    DELAY
        CALL    DELAY

        LD      BX,#ATZ1
        CALL    SEND_COMMAND_MODEM
        JE      START

        CALL    DELAY
        CALL    DELAY

        LD      BX,#ATW1
        CALL    SEND_COMMAND_MODEM
        JE      START

        CALL    DELAY

;-----------------------------------
;   RING FROM MODEM ?

CHK_MASTER:

        JBS     IOPORT1,7,BB_MODE       ;CHECK RING
;        JBS     IOPORT1,7,CHK_MASTER     ;CHECK RING
                                         ;IF RING,IOPORT,7=LOW

;        CALL    MODEM_RX_STATUS
;        JE      CHK_MASTER              ;BB_MODE

;        CALL    MODEM_RX_DATA
;        CMPB    AL,#'2'                 ;2= RING,RING
;        JNE     CHK_MASTER              ;BB_MODE

;-----------------------------------
;   CENTRAL COMPUTER MODE

ON_LINE_MODE:

        CALL    SEND_ATA                ;MENUAL CONNECT
```

```
        CALL    XDELAY
        CALL    XDELAY
        CALL    XDELAY

READ_FROM_COMPUTER:

        CALL    MODEM_RX_STATUS
        JE      READ_FROM_DRY

        CALL    MODEM_RX_DATA

;       CMPB    AL,#'R'
;       JE      WRITE_TO_DRY
;       CMPB    AL,#'T'
;       JE      WRITE_TO_DRY
;       CMPB    AL,#'U'
;       JE      WRITE_TO_DRY
;
        CMPB    AL,#'3'                 ;'3' = NO CARRIER
        JE      START
        CMPB    AL,#'4'                 ;'4' = ERROR
        JE      START
;
;       ;CHECK NO CONNECTING TIME
;       CALL    XDELAY
;       INC     NO_CONT_TIME1
;       ADDC    NO_CONT_TIME2,#0000H
;       BC      START
;       BR      READ_FROM_DRY

WRITE_TO_DRY:
        CALL    TXDATA
;       LD      NO_CONT_TIME1,#0000H
;       LD      NO_CONT_TIME2,#0000H

READ_FROM_DRY:
        CALL    RX_STATUS
        JE      READ_FROM_COMPUTER

        CALL    RX_DATA

WRITE_TO_COMPUTER:
```

```
        CALL    MODEM_TXDATA

        BR      READ_FROM_COMPUTER

;-----------------------------------
; BB MODE - RADIO PAGER MODE

BB_MODE:

        LD      BX,#ATCR
        CALL    SEND_COMMAND_MODEM
        BNE     BB_MODE1

        JBC     IOPORT1,7,ON_LINE_MODE  ;CHECK RING
                                        ;IF RING,IOPORT,7=LOW
        BR      START
BB_MODE1:

        CALL    DELAY

        LDB     AL,IOPORT0              ;SWITCH PORT

;       JBS     AL,7,NO_WAIT_R
; FOR TEST
        JBC     AL,7,CHK_MASTER

        ;#### CENTRAL BOARD ####
        ; WAIT 'R',THEN SEND 'R'

WAIT_R:
        CALL    RXDATA
        CMPB    AL,#'R'
        BNE     WAIT_R

        ;#### -- DRYER --  ####
NO_WAIT_R:

        CALL    GET_CH_STATUS           ;SEND 'R'
        JC      SEND_BB

        CALL    CHECK_WARNING
        JC      SEND_BB

        BR      CHK_MASTER
```

```
;-------------------------------------

SEND_EB:                                ;SEND WARNING SIGNAL TO TELEPHONE

        LD      BX,#ATCS
        CALL    MC_STXDATA

        CALL    DELAY
        CALL    DELAY

        LD      BX,#ATEXIT              ;" +,,+,,+"
        CALL    MC_STXDATA
        CALL    DELAY

        LD      BX,#ATH         ;"ATH"
        CALL    MC_STXDATA
        CALL    DELAY

        BR      CHK_MASTER

;--------------------------------------
; WAIT OK FROM MODEM
;--------------------------------------

WAIT_OK:
        CALL    RXDATA
        CMPB    AL,#'R'
        BNE     WAIT_R

;--------------------------------------
; GET CHANNEL STATUS
; INPUT :  BX=  POINTER OF CHANNEL
; OUTPUT:  C FLAG=1 IF ERROR
;--------------------------------------
GET_CH_STATUS:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        ;SET DATA POINTER IN BX
        LD      BX,#DO_MODEL_NO                 ;BASE ADDRESS OF CH DATA
```

```
        ;EMPTIFY RX BUFFER
EMP_RX0:
        CALL    RX_STATUS
        JE      EMP_RX1

        CALL    RX_DATA
        BR      EMP_RX0

EMP_RX1:
        ;SEND 'R'
;       LDB     AL,#' '
;       CALL    TXDATA
        LDB     AL,#'R'
        CALL    TXDATA

        ;RECEIVE DATA
        ;WAIT  FIRST DATA CH NO.
REC_DAT0:
        CALL    RXDATA_NO_WAIT
        BE      REC_DAT2
        CMP     AL,#0AH                 ;CH NO= 0,1,2,3,...9
        BH      REC_DAT0

        STB     AL,[EX]+

        LDB     CL,#10
REC_DAT1:
        CALL    RXDATA_NO_WAIT
        BE      REC_DAT2

        STB     AL,[EX]+

        DJNZ    CL,REC_DAT1

        CLRC                    ;CLEAR ERRCR FLAG

        BR      REC_EXIT

REC_DAT2:
        SETC                     ;SET ERRCR FLAG

REC_EXIT:
        POP     DX
        POP     CX
```

```
        POP     BX
        POP     AX
        RET


;-------------------------------------
; CHECK LED FOR CHECKING WARNING
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------

CHECK_WARNING:

        PUSH    AX
        PUSH    BX

        LD      BX,#DO_MODEL_NO                 ;BASE ADDRESS OF CH DATA

        LDB     AL,6[BX]                        ;LED1

        CMPB    AL,#0FEH
        JE      NC_ERROR_LED

ERROR_LED:
        SETC                                    ;C=1
        BR      CHK_EXIT

NC_ERROR_LED:
        CLRC                                    ;C=0

CHK_EXIT:

        POP     BX
        POP     AX
        RET

M1:       DCB     ' #### M1 SEND AT ####! ',CR,LF
M2:       DCB     ' #### M2 SEND ATV0E0 ###! ',CR,LF
M3:       DCB     ' #### M3 ATX1 ###! ',CR,LF
M4:       DCB     ' #### M3 ATZ1 ###! ',CR,LF
M5:       DCB     ' #### M3 ATW1 ###! ',CR,LF

;************************************

ATCR:     DCB     'AT',CR,LF
```

```
ATX1:       DCB     'ATX1',CR,LF
ATV0E0:     DCB     'ATV0E0',CR,LF
ATA:        DCB     'ATA',CR,LF
ATZ1:       DCB     'AT&Z1=T9,0127345424,1,,,,,,,505424*',CR,LF
ATW1:       DCB     'AT&W1',CR,LF
ATDS:       DCB     'ATDS=1',CR,LF
ATEXIT:     DCB     '+,,+,,+',CR,LF
ATH:        DCB     'ATH',CR,LF

;************************************
; INITIALIZE MODEM
; INPUT  BX = ADDRESS OF STRING
; IF Z=1(E) THEN ERROR
; IF C=1    THEN

SEND_COMMAND_MODEM:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

;       LD      BX,#####
        CALL    MC_STXDATA

        LD      DX,#0000H
SIDT0:
        LD      CX,#0000H
SIDT1:
        CALL    MODEM_RX_STATUS
        JE      SIDT2

        CALL    MODEM_RX_DATA

        CMPB    AL,#'0'                 ;'0'= OK
        JNE     SIDT2

;       CALL    TXDATA
        ORB     AL,AL                   ;IF 'OK' THEN Z=0 ,C=0
        BR      SIDT_EXIT

SIDT2:
        INC     CX
        CMP     CX,#0FFFFH
```

```
        JNE     SIDT1

        INC     DX
        CMP     DX,#20
        JNE     SIDT0

        ANDB    AL,#00H                 ;IF 'NCT CK' THEN Z=1  ,C=0

SIDT_EXIT:
        CLRC

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;************************************
SEND_AT:
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        LD      BX,#ATCR
        CALL    MC_STXDATA

        LD      DX,#0000H
AT_DT0:
        LD      CX,#0000H
AT_DT1:
        CALL    MCDEM_RX_STATUS
        JE      AT_DT2

        CALL    MCDEM_RX_DATA

AT_DT2:
        INC     CX
        CMP     CX,#0FFFFH
        JNE     AT_DT1

        INC     DX
        CMP     DX,#3
        JNE     AT_DT0
```

```
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;-----------------------------------
SEND_ATV0E0:
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        LD      BX,#ATV0E0
        CALL    MC_STXDATA

        LD      DX,#0000H
ETDT0:
        LD      CX,#0000H
ETDT1:
        CALL    MCDEM_RX_STATUS
        JE      ETDT2

        CALL    MCDEM_RX_DATA

ETDT2:
        INC     CX
        CMP     CX,#0FFFFH
        JNE     ETDT1

        INC     DX
        CMP     DX,#3
        JNE     ETDT0

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET
;-----------------------------------
SEND_ATA:
        PUSH    AX
        PUSH    BX
```

```
        PUSH    CX
        PUSH    DX

        LD      BX,#ATA
        CALL    MC_STXDATA

        LD      DX,#0000H
ATDT0:
        LD      CX,#0000H
ATDT1:
        CALL    MCDEM_RX_STATUS
        JE      ATDT2

        CALL    MCDEM_RX_DATA

ATDT2:
        INC     CX
        CMP     CX,#0FFFFH
        JNE     ATDT1

        INC     DX
        CMP     DX,#3
        JNE     ATDT0

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;**************************************
; DISPLAY ERROR ON SIO

ERROR_ON_MODEM:

        PUSH    AX
        PUSH    BX

        LD      BX,#ERROR_MD
        CALL    STXDATA

        POP     BX
        POP     AX
        RET
```

```
ERROR_MD:       DCB     ' ####  ERROR ON MODEM BOARD ####! ',CR,LF

;*************************************
; SEND STRING DATA TO SIO
; UNTILE #LF

STXDATA:

        PUSH    AX
        PUSH    BX
        PUSH    CX

        MOV     CL,#0FFH
STO:
        MOV     AL,[BX]+
;       CALL    TXDATA

        CMPB    AL,#LF
        JE      ST_EXIT

        DJNZ    CL,STO

ST_EXIT:
        POP     CX
        POP     BX
        POP     AX
        RET

;*************************************
; SEND STRING DATA TO MODEM
; UNTILE #LF

MD_STXDATA:

        PUSH    AX
        PUSH    BX
        PUSH    CX

        MOV     CL,#0FFH
MD_STO:
        MOV     AL,[BX]+
        CALL    MODEM_TXDATA
```

```
        CMPB    AL,#LF
        JE      MC_ST_EXIT

        DJNZ    CL,MC_ST0

 MC_ST_EXIT:

        POP     CX
        POP     BX
        POP     AX
        RET

;**********************************
; READ MODEM DATA WITHOUT WAIT PROCEDURE
; INPUT: MODEM
; OUTPUT:AL,Z FLAG ( IF RXDATA THEN Z =0 ELSE Z=1)
; MODIFY:
;**********************************
MRX_NOWAIT:

        PUSH    CX

        LD      CX,#0FF00H
MC_RX1:
        DEC     CX
        CMP     CX,#0000H
        BE      MC_RX_EXIT                      ; Z FLAG =1

        CALL    MODEM_RX_STATUS
        JE      MC_RX1

        CALL    MODEM_RX_DATA

        ADD     CX,#1                           ; Z FLAG =0

MC_RX_EXIT:
        POP     CX

        RET

;**********************************
; SERIAL INITIAL PROCEDURE
; 8_BIT(.NO_PARITY)+1_STOP_BIT
; baud_val = (XTAL/64)/BAUD_RATE
```

```
; INPUT :NONE
; OUTPUT:BAUD_REG,SPCON,IM_SPCON,ICC1,RECEIVE INETERRUPT
; MODIFY:
;***********************************

        baud_val        EQU     72                      ;(2400baud)
        BAUD_HIGH       EQU     ((baud_val-1)/256) OR 80H
        BAUD_LOW        EQU     (baud_val-1) MOD 256

SIO_INIT:

        LD      WR_POINTER,#BUF_START
        LD      RD_POINTER,#BUF_START

        LDB     ICC1, #00100000B

        LDB     BAUD_REG, #BAUD_LOW
        LDB     BAUD_REG, #BAUD_HIGH

        LDB     SPCON, #01011001B
        LDB     IM_SPCON, #00100000B
        ORB     IM_SPCON,SPCON

        LDB     AL,SBUF

        RET

;***********************************
; TX_STATUS
; INPUT: SBUF
; FLAG
; MODIFY:
;***********************************
TX_STATUS:
;       ORB     IM_SPCON,SPCON
        LDB     AH,#00100000B
        ANDB    AH,IM_SPCON
        RET

;***********************************
; TX_DATA PROCEDURE
; INPUT :AL
; OUTPUT:SBUF
; MODIFY:
```

```
;**********************************
TX_DATA:
        ANDB    IM_SPCCN,#11011111B
        LDB     SBUF,AL
        RET


;**********************************
; TXDATA PRCCEDURE
; INPUT :AL
; CUTPUT:SBUF
; MODIFY:
;**********************************
TXDATA:
;       CRB     IM_SPCCN,SPCCN
        BBC     IM_SPCCN, 5,TXDATA
        ANDB    IM_SPCCN,#11011111B
        LDB     SBUF,AL
        RET

;**********************************
; RXDATA PRCCEDURE
; INPUT: SBUF
; CUTPUT:AL
; MODIFY:
;**********************************
RXDATA:
        CMP     WR_POINTER,RD_POINTER
        JE      RXDATA

        LDB     AL,[RD_POINTER]
        INC     RD_POINTER

        CMP     RD_POINTER,#BUF_END
        JNE     RXD1

        LD      RD_POINTER,#BUF_START
RXD1:
        RET

;
;       CRB     IM_SPCCN,SPCCN
;       BBC     IM_SPCCN, 6,RXDATA
;       ANDB    IM_SPCCN,#10111111B
```

```
;       LDB     AL,SBUF
;       RET
;
;**********************************
; RX_STATUS
; INPUT: SBUF
; FLAG
; MODIFY:
;**********************************
RX_STATUS:
        CMP     WR_POINTER,RD_POINTER
        RET

;       ORB     IM_SPCON,SPCON
;       LDB     AH,#01000000B
;       ANDB    AH,IM_SPCON
;
;       RET

;**********************************
; RX_DATA PROCEDURE
; INPUT: SBUF
; OUTPUT:AL
; MODIFY:
;**********************************
RX_DATA:

        LDB     AL,[RD_POINTER]
        INC     RD_POINTER

        CMP     RD_POINTER,#BUF_END
        JNE     RX_D1

        LD      RD_POINTER,#BUF_START
RX_D1:
        RET
;
;       ANDB    IM_SPCON,#10111111B
;       LDB     AL,SBUF
;       RET

;**********************************
; RXDATA_NC_WAIT PROCEDURE
; INPUT: SBUF
```

```
; CUTPUT:AL,Z FLAG ( IF RXDATA THEN Z =0 ELSE Z=1)
; MODIFY:
;**********************************
RXDATA_NC_WAIT:
        PUSH    CX

        LD      CX,#0F000H
RX_NW1:
        DEC     CX
        CMP     CX,#0000H
        BE      RX_NW_EXIT              ; Z FLAG =1

        CALL    RX_STATUS               ;CMP     WR_POINTER,RD_POINTER
        JE      RX_NW1

        CALL    RX_DATA

;        ORB     IM_SPCCN,SPCCN
;        BBC     IM_SPCCN,6,RX_NW1
;
;        ANDB    IM_SPCCN,#10111111B
;        LDB     AL,SBUF

        CR      CX,#1                    ; Z FLAG =0

RX_NW_EXIT:
        POP     CX

        RET


CR      EQU     0DH
LF      EQU     0AH
ESC     EQU     0H

;**********************************
; TRANSMIT CR,LF,(BELL)
; INPUT: NONE
; CUTPUT:SBUF
; MODIFY:NONE
;**********************************

CRLF:   PUSH    AX
        LDB     AL,#CR
```

```
        CALL    TXDATA
        LDB     AL,#LF
        CALL    TXDATA
        LDB     AL,#' '
        CALL    TXDATA
        LDB     AL,#' '
        CALL    TXDATA
        POP     AX
        RET

CR_:    PUSH    AX
        LDB     AL,#CR
        CALL    TXDATA
        POP     AX
        RET

;************************************
; 1 HEX TO 2 ASCII PROCEDURE
; INPUT  : AL
; OUTPUT : SBUF
; MODIFY : NONE
;************************************
HTOA2:
        PUSH    AX
        PUSH    BX

        LDB     AH,AL
        SHRB    AL,#04
        CALL    HTOA
        CALL    TXDATA
        LDB     AL,AH
        CALL    HTOA
        CALL    TXDATA

        POP     BX
        POP     AX
        RET

HTOA1:  CALL    HTOA
        CALL    TXDATA
        RET

HTOA:   ANDB    AL,#0FH
        LDB     BHH,#00H
```

```
        LDB     BL,AL
        LDB     AL,T_HTCA[BX]
        RET

T_HTCA: DCB     '0123456789ABCDEF'

;************************************
;2 ASCII TO 1 HEX CONVERT
; INPUT  : [BX]:msb,[BX+1]:lsb
; OUTPUT : AL, C(SET IF ERROR)
; MODIFY : AX
;************************************
ATCH2:
        PUSH    BX

        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        SHLB    AL,#04
        LDB     AH,AL
        INC     BX
        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        ORB     AL,AH

AT1:    POP     BX
        RET

ATOH:   CMPB    AL,#2FH
        BNH     ATO8
        CMPB    AL,#39H
        BH      ATO2
        BR      ATO6

ATO2:   CMPB    AL,#40H
        BNH     ATO8
        CMPB    AL,#46H
        BH      ATO8
        ADDB    AL,#09H

ATO6:   ANDB    AL,#0FH
        CLRC
        BR      ATO9
```

```
ATC8:
        SETC                            ;ERROR NOT HEX

ATC9:   RET

;-------------------------------------

BINTCDEC:

        PUSH    BX
        PUSH    DX

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        STB     DL,BL

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        SHLB    DL,#4                   ;STORE LSB
        ORB     BL,DL

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        STB     DL,BHH

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        SHLB    DL,#4
        ORB     BHH,DL                  ;STORE MSB

        ST      BX,AX

        POP     DX
        POP     BX

        RET

;-------------------------------------

XDELAY:
        PUSHF
        PUSH    DX
        PUSH    CX
```

```
        LD      DX,#10
XDEL0:
        LD      CX,#0FFFFH
XDEL1:
        NOP
        NOP

        DEC     CX
        CMP     CX,#0
        JNE     XDEL1

        DEC     DX
        CMP     DX,#0
        JNE     XDEL0

        POP     CX
        POP     DX
        POPF
        RET

DELAY:
        PUSHF
        PUSH    DX
        PUSH    CX

        LD      DX,#10
DEL0:
        LD      CX,#0FFFFH
DEL1:
        CALL    MCDEM_RX_STATUS
        JE      DEL2

        CALL    MCDEM_RX_DATA
DEL2:

        DEC     CX
        CMP     CX,#0
        JNE     DEL1

        DEC     DX
        CMP     DX,#0
        JNE     DEL0

        POP     CX
```

```
        POP     DX
        POPF
        RET
;-------------------------------------

SDELAY:
        PUSHF
        PUSH    DX
        PUSH    CX

        LD      DX,#10
SDEL0:
        LD      CX,#01FH
SDEL1:
        CALL    MCDEM_RX_STATUS
        JE      SDEL2

        CALL    MCDEM_RX_DATA
        CMPB    AL,#'2'
        JE      SDEL_RING
SDEL2:

        DEC     CX
        CMP     CX,#0
        JNE     SDEL1

        DEC     DX
        CMP     DX,#0
        JNE     SDEL0

        POP     CX
        POP     DX
        POPF

        CLRC
        RET

SDEL_RING:

        POP     CX
        POP     DX
        POPF

        SETC
```

```
        RET

;*************************************
;LRC224ATF

        RBR             EQU     8000H
        THR             EQU     8000H
        IER             EQU     8001H
        IIR             EQU     8002H
        LCR             EQU     8003H
        MCR             EQU     8004H
        LSR             EQU     8005H
        DLL             EQU     8000H
        DLM             EQU     8001H

;*****************************************************

MODEM_INIT:


        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX


;DLAB =1
        LD      DX,#LCR
        LDB     AL,#80H         ;BAUD RATE ENABLE
        STB     AL,[DX]         ;STB     DX,AL
        NOP
        NOP

        LD      DX,#DLL
        LDB     AL,#30H         ;LOW OF 2400H BAUD
        STB     AL,[DX]         ;STB     DX,AL
        NOP
        NOP

        LD      DX,#DLM
        LDB     AL,#00H         ;HIGH OF 2400H BAUD
        STB     AL,[DX]         ;STB     DX,AL
        NOP
        NOP
```

```
; DLAB =0

        LD      DX,#LCR
        LDB     AL,#03H         ;1 STOP BIT + 8BIT DATA+NO PARIRY
        STB     AL,[DX]         ;STB     DX,AL
        NOP
        NOP

        LD      DX,#IER
        LDB     AL,#00H         ;DISABLE INTTERRUPT
        STB     AL,[DX]         ;STB     DX,AL
        NOP
        NOP

        LD      DX,#MCR
        LDB     AL,#03H         ;LOCAL LOOP,RTS,DTR
        STB     AL,[DX]         ;STB     DX,AL

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;*************************************
;RESET MODEM
RESET_MODEM:

        LDB     ICPCRT1,#0FFH
        CALL    XDELAY
        LDB     ICPCRT1,#0F0H
        CALL    XDELAY
        RET

;*************************************
; TEST LINE_STATUS BIT OF MODEM
; TX/RX DATA FROM MODEM

        DR      EQU     01H
        THRE    EQU     20H

MODEM_RX_STATUS:
```

```
        LDB     AH,LSR[0]
        ANDB    AH,#CR
        RET

MODEM_RX_DATA:

        LDB     AL,RBR[0]
        RET

MODEM_TX_STATUS:

        LDB     AH,LSR[0]
        ANDB    AH,#THRE

        RET

MODEM_TX_DATA:

        STB     AL,THR[0]
        RET

MODEM_TXDATA:

        LDB     AH,LSR[0]
        ANDB    AH,#THRE
        JE      MODEM_TXDATA
        STB     AL,THR[0]
        RET

;**********************************
; INTERRUPT RXDATA PROCEDURE
; INPUT: SBUF
; OUTPUT:AL
; MODIFY:
;**********************************
INT_RXDATA:
        PUSHF

        CRB     IM_SPCON,SPCON
        BBC     IM_SPCON, 6,INT_RX1
        ANDB    IM_SPCON,#10111111B

        STB     SBUF,[WR_POINTER]
        INC     WR_POINTER
```

```
        CMP     WR_PCINTER,#BUF_END
        JNE     INT_RX1

        LD      WR_PCINTER,#BUF_START
INT_RX1:

        POPF
        EI
        RET

END
```

■첨부3) 건조기 전용프리터의 프로그램.

```
;PROGRAM PRTB.ASM,96.9.30
$INCLUDE (8096.INC)
        ; DEFINE SYMBOL
        ; OUTPUT RELAY SYMBOL

        ;COMMAND SYMBOL

        RSEG    AT      1AH
        IM_SPCCN:       DSB     1

        DAX:            DSL     1
        AX              EQU     DAX:WORD
        AL              EQU     AX:BYTE
        AH              EQU     AX+1:BYTE
        DX              EQU     DAX+2:WORD
        DL              EQU     DX:BYTE
        DH              EQU     DX+1:BYTE
        BX:             DSW     1
        BL              EQU     BX:BYTE
        BHH             EQU     BX+1:BYTE
        CX:             DSW     1
        CL              EQU     CX:BYTE
        CH              EQU     CX+1:BYTE

        IMAGE_LED:      DSB     1
        IMAGE_MODEL:    DSB     1

        ASC_H:          DSB     1
        ASC_L:          DSB     1

        COUNT:          DSW     1
        COUNT_L         EQU     COUNT:BYTE
        COUNT_H         EQU     COUNT+1:BYTE

        TIME:           DSW     1
        WAIT_TIME:      DSW     1

        DATA_BUFFER:    DSB     40

;*************************************
CSEG            AT      2000H
```

```
VECT:           DCW     TIMER_INT       ;9000H
                DCW     2080H           ;9200H
                DCW     2080H           ;9400H
                DCW     2080H           ;9600H
                DCW     2080H           ;9800H
                DCW     2080H           ;9A00H
                DCW     2080H           ;9C00H
                DCW     2080H           ;9E00H

CSEG            AT      2018H
CCR:            DCB     0FDH

;-------------------------------------------------
CSEG    AT      2080H           ;RESET

   HCUR_SET     EQU     1592            ;1592 (1 HCUR)
   MIN_SET      EQU     1                   ;2.3SEC...          ;26    (1
MINUTE)

    ESC         EQU     0FFH
MODE_00         EQU     00H             ;REQUEST,HEX DUMP
MODE_10         EQU     04H             ;REQUEST,DECODING
MODE_20         EQU     08H             ;
MODE_30         EQU     0CH             ;

START:

        LD      SP,#0E0H

        CALL    INIT_SYS
        LD      COUNT,#0000H

        LD      DX,#SIGN_ON1
        CALL    SOUTPUT
        CALL    CR_LF
        LD      DX,#SIGN_ON2
        CALL    SOUTPUT
        CALL    CR_LF
        LD      DX,#SIGN_ON3
        CALL    SOUTPUT
        CALL    CR_LF
        LD      DX,#SIGN_ON4
        CALL    SOUTPUT
        CALL    CR_LF
```

```
        CALL    MCTOR_CFF

        BR      CHECK_SWITCH

SIGN_CN1:       DCB     '  ANDONG UNIVERSITY',ESC
SIGN_CN2:       DCB     '   YOUNG DONG.CC',ESC
SIGN_CN3:       DCB     '   COMPUTER DRYER',ESC
SIGN_CN4:       DCB     '   0571-821-0911',ESC

CHECK_SWITCH:

        LDB     AL,ICPORT0              ;SWITCH PORT
        ANDB    AL,#0CH

CHK_M00:
        CMPB    AL,#MODE_00
        BNE     CHK_M10

        BR      PROCESS_MODE00

CHK_M10:
        CMPB    AL,#MODE_10
        BNE     CHK_M20

        LD      WAIT_TIME,#10*MIN_SET
        BR      PROCESS_MODE

CHK_M20:
        CMPB    AL,#MODE_20
        BNE     CHK_M30

        LD      WAIT_TIME,#30*MIN_SET
        BR      PROCESS_MODE

CHK_M30:
        CMPB    AL,#MODE_30
        BNE     CHK_M40

        LD      WAIT_TIME,#60*MIN_SET
        BR      PROCESS_MODE

CHK_M40:
        BR      CHECK_SWITCH
```

```
;----------------------------------------
PROCESS_MODE00:
;----------------------------------------
;  CONNECT DRY OR CENTRAL BOARD ?

        LDB     AL,ICPCRT0              ;SWITCH PORT
        JBS     AL,7,PRC_MOD1

        ;CENTRAL BOARD
        ; WAIT 'R',THEN SEND 'R'

        CALL    RXDATA
        CMPB    AL,#'R'
        BNE     CHECK_SWITCH

;       LDB     AL,#'R'
;       CALL    TXDATA

        ;DRY,CENTRAL BOARD
PRC_MOD1:

        INC     COUNT

        CALL    REQUEST

        CALL    GET_DATA

        CALL    PRINT_HEX_BUF

        CALL    MOTOR_OFF

        BR      CHECK_SWITCH

;----------------------------------------
PROCESS_MODE:
;----------------------------------------
;  CONNECT DRY OR CENTRAL BOARD ?

        LDB     AL,ICPCRT0              ;SWITCH PORT
        JBS     AL,7,PRC_M10

        ;CENTRAL BOARD
        ; WAIT 'R',THEN SEND 'R'
```

```
        CALL    RXDATA
        CMPB    AL,#'R'
        BNE     CHECK_SWITCH

;       LDB     AL,#'R'
;       CALL    TXDATA

        ;DRY,CENTRAL BCAED
PRC_M10:
        CALL    REQUEST

        CALL    GET_DATA

        LDB     AL,ICPCRT0              ;SWITCH PORT
        JBC     AL,7,PRC_M11

        CALL    CHECK_NC_ERRCR
        BNE     PRC_M1B

PRC_M11:

        CMP     TIME,WAIT_TIME
        BE      PRC_M1A
        BH      PRC_M1A
        BR      CHECK_SWITCH

PRC_M1A:
        LD      TIME,#00                ;RESET TIMER

PRC_M1B:

        INC     COUNT

        CALL    CUT_DECODE

        CALL    MOTOR_OFF

        BR      CHECK_SWITCH

;*****************************************
; GET_BUFFER
; INPUT  :  AL - CALL RXDATA UTIL AL=CR
; OUTPUT :  DATA_BUFFER
;-----------------------------------------
```

```
GET_BUFFER:
        PUSH    AX
        PUSH    BX
        PUSH    CX

        LD      BX,#DATA_BUFFER
        LDB     CL,#20

GET_BUF1:
        CALL    RXDATA_NO_WAIT
        BE      GET_NO_BUF

        STB     AL,[BX]+

        CMPB    AL,#CR
        JE      GET_BUF_EXIT

        DJNZ    CL,GET_BUF1

        BR      GET_BUF_EXIT

GET_NO_BUF:
        LD      BX,#DATA_BUFFER

        LDB     AL,#ESC
        STB     AL,[BX]

GET_BUF_EXIT:

        POP     CX
        POP     BX
        POP     AX
        RET

;-------------------------------------
; REQUEST 'R'
; INPUT  :  NONE
; OUTPUT :  TXDATA
;-------------------------------------
REQUEST:

        PUSH    AX

        CALL    DELAY
```

```
        CALL    DELAY

        ;EMPTIFY RX BUFFER
        CALL    RX_STATUS
        BBC     IM_SPCCN,6,EMP_RX1
        CALL    RX_DATA

EMP_RX1:
        ;SEND 'R'
;       LDB     AL,#' '
;       CALL    TXDATA
        LDB     AL,#'R'
        CALL    TXDATA

        POP     AX

        RET

;--------------------------------------
; GET DATA
; INPUT  :  AL - CALL RXDATA UTIL HEX LENGTH = 10
; OUTPUT :  DATA_BUFFER
;--------------------------------------
GET_DATA:

        PUSH    AX
        PUSH    BX
        PUSH    CX

       ;RECEIVE DATA

        LD      BX,#DATA_BUFFER

        CALL    RXDATA_NO_LWAIT
        BE      REC_NO_DATA

        STB     AL,[BX]+

        LDB     CL,#10
REC_DAT1:

        CALL    RXDATA_NO_WAIT
        BE      REC_NO_DATA
```

```
        STB     AL,[EX]+

        DJNZ    CL,REC_DAT1

        BR      REC_DAT_EXIT

REC_NO_DATA:                                    ;NO DATA = 00
        LD      EX,#DATA_BUFFER
        LDB     CL,#11
        LDB     AL,#00
REC_NO_1:
        STB     AL,[EX]+
        DJNZ    CL,REC_NO_1

REC_DAT_EXIT:

        POP     DX
        POP     EX
        POP     AX
        RET

;*****************************************
; PRINT_STR   : ASCII MODE
; INPUT :  DATA_BUFFER
; OUTPUT:  PRINTER
;-----------------------------------------
PRINT_STR:
        PUSH    AX
        PUSH    EX
        PUSH    CX

        LDB     CL,#20
        LD      EX,#DATA_BUFFER

        LDB     AL,[EX]                 ;CHECK NO DATA
        CMPB    AL,#ESC
        BE      PRINT_ST2

        CALL    LINE_FEED
        CALL    MOTOR_ON
        CALL    WAIT_HOME_OFF

        CALL    WAIT_TG_ON
        CALL    WAIT_TG_OFF
```

```
        CALL    WAIT_TG_ON
        CALL    WAIT_TG_OFF

PRINT_ST1:
        LDB     AL,[EX]+
        CMPB    AL,#CR
        JE      PRINT_ST2

        CALL    PRINT_CHAR

        DJNZ    CL,PRINT_ST1
PRINT_ST2:

        POP     CX
        POP     BX
        POP     AX
        RET

;*****************************************
; PRINT CR_LF
; INPUT :  NONE
; OUTPUT:  PRINTER
;-----------------------------------------
CR_LF:

        CALL    LINE_FEED
        CALL    MOTOR_ON
        CALL    WAIT_HOME_OFF

        CALL    WAIT_TG_ON
        CALL    WAIT_TG_OFF

        CALL    WAIT_TG_ON
        CALL    WAIT_TG_OFF

        RET

;*****************************************
; PRINT_HEX_BUF     : HEX MODE
; INPUT :  DATA_BUFFER
; OUTPUT:  PRINTER
;-----------------------------------------
PRINT_HEX_BUF:
```

```
        PUSH    AX
        PUSH    BX
        PUSH    CX

        LDB     CL,#10
        LD      BX,#DATA_BUFFER

;       LDB     AL,[BX]                 ;CHECK NO DATA
;       CMPB    AL,#ESC
;       BE      PRINT_ST22

        CALL    CR_LF

PRINT_ST11:
        LDB     AL,[BX]+

        CALL    HTOA2                   ;SAVE IN ASC_H,ASC_L

        LDB     AL,ASC_H
        CALL    PRINT_CHAR

        LDB     AL,ASC_L
        CALL    PRINT_CHAR

        DJNZ    CL,PRINT_ST11
PRINT_ST22:

        POP     CX
        POP     BX
        POP     AX
        RET

;*****************************************
; PRINT_AX
; INPUT :  AX = HEX VALUE
; OUTPUT:  PRINTER = DECIMAL VALUE
;-----------------------------------------
PRINT_AX:
        PUSH    AX
        PUSH    BX

        CALL    BINTODEC
        LD      BX,AX
```

```
        LDB     AL,BHH
        CALL    HTOA2                   ;SAVE IN ASC_H,ASC_L

        LDB     AL,ASC_H
        CALL    PRINT_CHAR

        LDB     AL,ASC_L
        CALL    PRINT_CHAR

        LDB     AL,BL
        CALL    HTOA2                   ;SAVE IN ASC_H,ASC_L

        LDB     AL,ASC_H
        CALL    PRINT_CHAR

        LDB     AL,ASC_L
        CALL    PRINT_CHAR

        POP     BX
        POP     AX
        RET


;*****************************************
; PRINT_AL
; INPUT :  AL = HEX VALUE
; OUTPUT:  PRINTER = DECIMAL VALUE
;-----------------------------------------
PRINT_AL:
        PUSH    AX
        PUSH    BX

        LDB     AH,#00H

        CALL    BINTCDEC
        LD      BX,AX

        LDB     AL,BL
        CALL    HTOA2                   ;SAVE IN ASC_H,ASC_L

        LDB     AL,ASC_H
        CALL    PRINT_CHAR

        LDB     AL,ASC_L
```

```
        CALL    PRINT_CHAR

        POP     BX
        POP     AX
        RET

;*****************************************
; CHECK_NO_ERROR
; INPUT : 6[BX], EX=DATA_BUFFER
; OUTPUT: FLAG
;-----------------------------------------

CHECK_NO_ERROR:
        PUSH    AX
        PUSH    BX

        LD      BX,#DATA_BUFFER

        LDB     AL,1[BX]                ;AL=MODEL
        CMPB    AL,IMAGE_MODEL
        BNE     EXIT_CH_NO

        LDB     AL,6[BX]                ;AL=LED
        CMPB    AL,IMAGE_LED

EXIT_CH_NO:
        STB     AL,IMAGE_LED

        LDB     AL,1[BX]                ;AL=MODEL
        STB     AL,IMAGE_MODEL

        POP     BX
        POP     AX
        RET
;*****************************************
; OUTPUT DECODED VALUE
; INPUT :  EX=DATA_BUFFER
; OUTPUT:  PRINTER
;-----------------------------------------
OUT_DECODE:
        PUSH    AX
        PUSH    BX
        PUSH    CX
```

```
LD      BX,#DATA_BUFFER

LDB     AL,[BX]                 ;CHECK NO DATA
CMPB    AL,#ESC
BE      EXIT_OUT_DEC

CALL    CR_LF

;PRINT COUNT
LD      DX,#STR_NO
CALL    SOUTPUT
LD      AX,COUNT
CALL    PRINT_AX

CALL    CR_LF

LDB     AL,0[BX]
LD      DX,#STR_CH
CALL    DISPLAY_DECODING

CALL    CR_LF

LDB     AL,1[BX]
LD      DX,#STR_MODEL
CALL    DISPLAY_DECODING

CALL    CR_LF

LDB     AL,2[BX]
LD      DX,#STR_DRY_STEP
CALL    DISPLAY_DECODING


CALL    CR_LF

LDB     AL,3[BX]
LD      DX,#STR_STEP_TIME
CALL    DISPLAY_DECODING

CALL    CR_LF

LDB     AL,4[BX]
LD      DX,#STR_TEMP_DRY
CALL    DISPLAY_DECODING
```

```
        CALL    CR_LF

        LDB     AL,5[BX]
        LD      DX,#STR_TEMP_WET
        CALL    DISPLAY_DECCDING

        CALL    CR_LF

        LD      DX,#STR_TCTAL
        CALL    SCUTPUT
        LDB     AL,9[BX]
        LDB     AH,#00H
        CALL    PRINT_AX

        CALL    CR_LF

        LD      DX,#STR_ERRCR
        CALL    SCUTPUT

        CALL    CR_LF

        LDB     AL,6[BX]
        CMPB    AL,#0FEH
        BNE     CUT_C1
        LD      DX,#STR_GD
        CALL    SCUTPUT

        CALL    CR_LF

        BR      CUT_CA
OUT_C1:

        BBS     AL,1,CUT_C2
        LD      DX,#STR_ST
        CALL    SCUTPUT

        CALL    CR_LF

OUT_C2:
        BBS     AL,2,CUT_C3
        LD      DX,#STR_FN
        CALL    SCUTPUT
```

```
        CALL    CR_LF

OUT_D3:
        BBS     AL,3,OUT_D4
        LD      EX,#STR_DP
        CALL    SCUTPUT
        CALL    CR_LF
OUT_D4:
        BBS     AL,4,OUT_D5
        LD      EX,#STR_BN
        CALL    SCUTPUT

        CALL    CR_LF

OUT_D5:
        BBS     AL,5,OUT_D6
        LD      EX,#STR_DH
        CALL    SCUTPUT

        CALL    CR_LF

OUT_D6:
        BBS     AL,6,OUT_D7
        LD      EX,#STR_WH
        CALL    SCUTPUT

        CALL    CR_LF

OUT_D7:
        BBS     AL,7,OUT_DA
        LD      EX,#STR_SEN
        CALL    SCUTPUT

        CALL    CR_LF

OUT_DA:
        CALL    CR_LF

EXIT_CUT_DEC:

        CALL    CR_LF

        POP     CX
        POP     BX
```

```
        POP     AX
        RET

;*****************************************
; DISPLAY DECODING VALUE
; INPUT :  AL  - HEX
;          BX - ADDRESS OF STRING
;-----------------------------------------
DISPLAY_DECODING:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        CALL    SOUTPUT

        CALL    HTOA2                   ;SAVE IN ASC_H,ASC_L

        LDB     AL,ASC_H
        CALL    PRINT_CHAR

        LDB     AL,ASC_L
        CALL    PRINT_CHAR

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;*****************************************
; SOUTPUT  -- STRING OUTPUT MODE (ASCII)
; INPUT :  BX= ADDRESS OF STRING
; OUTPUT:  PRINTER
;-----------------------------------------
SOUTPUT:
        PUSH    AX
        PUSH    CX

        LDB     CL,#20
SOUT1:
        LDB     AL,[DX]+                ;CHECK NO DATA
        CMPB    AL,#ESC
```

```
        BE      SCUT2

        CALL    PRINT_CHAR

        DJNZ    CL,SCUT1

SCUT2:

        POP     CX
        POP     AX
        RET

STR_NC:          DCB     'NC: ',ESC
STR_CH:          DCB     'CH         :',ESC
STR_MODEL:       DCB     'MODEL      :',ESC
STR_DRY_STEP:    DCB     'DRY STEP   :',ESC
STR_STEP_TIME:   DCB     'STEP TIME :',ESC
STR_TEMP_DRY:    DCB     'DRY        :',ESC
STR_TEMP_WET:    DCB     'WET        :',ESC
STR_ERROR:       DCB     'ERROR: ',ESC
STR_TOTAL:       DCB     'TOTAL TIME:',ESC

STR_GO:          DCB     '  NO Err,',ESC
STR_ST:          DCB     '  Sel Test !',ESC
STR_FN:          DCB     '  Fan !',ESC
STR_DP:          DCB     '  Damp !',ESC
STR_BN:          DCB     '  Burn !',ESC
STR_DH:          DCB     '  Dry Hi !',ESC
STR_WH:          DCB     '  Wet Hi !',ESC
STR_SEN:         DCB     '  Sensor !',ESC

;*****************************************
; PRINT_CHAR
; INPUT :  AL
;
;-----------------------------------------
PRINT_CHAR:
        PUSH    AX
        PUSH    BX
        PUSH    CX

        LD      BX,#FONT
        LDB     AH,#00H
        SHL     AX,#3
```

```
        ADD     BX,AX

        LDB     CL,#7

PRINT_C1:
        LDB     AL,[BX]+

        CALL    WAIT_TG_ON

        CALL    DCT_ON

        CALL    WAIT_TG_OFF

        CALL    DCT_OFF

        DJNZ    CL,PRINT_C1

        POP     CX
        POP     BX
        POP     AX
        RET


font:
    DCB     000h,000h,000h,000h,000h,000h,000h,000h
    DCB     03eh,05bh,04fh,05bh,03eh,000h,000h,000h
    DCB     03eh,06bh,04fh,06bh,03eh,000h,000h,000h
    DCB     01ch,03eh,07ch,03eh,01ch,000h,000h,000h
    DCB     018h,03ch,07eh,03ch,018h,000h,000h,000h
    DCB     01ch,057h,07dh,057h,01ch,000h,000h,000h
    DCB     01ch,05eh,07fh,05eh,01ch,000h,000h,000h
    DCB     000h,018h,03ch,018h,000h,000h,000h,000h
    DCB     0ffh,0e7h,0c3h,0e7h,0ffh,000h,000h,000h
    DCB     000h,018h,024h,018h,000h,000h,000h,000h
    DCB     0ffh,0e7h,0dbh,0e7h,0ffh,000h,000h,000h
    DCB     030h,048h,03ah,006h,00eh,000h,000h,000h
    DCB     026h,029h,079h,029h,026h,000h,000h,000h
    DCB     040h,07fh,005h,005h,007h,000h,000h,000h
    DCB     040h,07fh,005h,025h,03fh,000h,000h,000h
    DCB     05ah,03ch,0e7h,03ch,05ah,000h,000h,000h
    DCB     07fh,03eh,01ch,01ch,008h,000h,000h,000h
    DCB     008h,01ch,01ch,03eh,07fh,000h,000h,000h
    DCB     014h,022h,07fh,022h,014h,000h,000h,000h
    DCB     05fh,05fh,000h,05fh,05fh,000h,000h,000h
```

```
DCB     006h,009h,07fh,001h,07fh,000h,000h,000h
DCB     000h,066h,089h,095h,06ah,000h,000h,000h
DCB     060h,060h,060h,060h,060h,060h,000h,000h
DCB     094h,0a2h,0ffh,0a2h,094h,000h,000h,000h
DCB     008h,004h,07eh,004h,008h,000h,000h,000h
DCB     010h,020h,07eh,020h,010h,000h,000h,000h
DCB     008h,008h,02ah,01ch,008h,000h,000h,000h
DCB     008h,01ch,02ah,008h,008h,000h,000h,000h
DCB     01eh,010h,010h,010h,010h,000h,000h,000h
DCB     00ch,01eh,00ch,01eh,00ch,000h,000h,000h
DCB     030h,038h,03eh,038h,030h,000h,000h,000h
DCB     006h,00eh,03eh,00eh,006h,000h,000h,000h
DCB     000h,000h,000h,000h,000h,000h,000h,000h
DCB     000h,000h,05fh,000h,000h,000h,000h,000h
DCB     000h,007h,000h,007h,000h,000h,000h,000h
DCB     014h,07fh,014h,07fh,014h,000h,000h,000h
DCB     024h,02ah,07fh,02ah,012h,000h,000h,000h
DCB     023h,013h,008h,064h,062h,000h,000h,000h
DCB     036h,049h,056h,020h,050h,000h,000h,000h
DCB     000h,008h,007h,003h,000h,000h,000h,000h
DCB     000h,01ch,022h,041h,000h,000h,000h,000h
DCB     000h,041h,022h,01ch,000h,000h,000h,000h
DCB     02ah,01ch,07fh,01ch,02ah,000h,000h,000h
DCB     008h,008h,03eh,008h,008h,000h,000h,000h
DCB     000h,080h,070h,030h,000h,000h,000h,000h
DCB     008h,008h,008h,008h,008h,000h,000h,000h
DCB     000h,000h,060h,060h,000h,000h,000h,000h
DCB     020h,010h,008h,004h,002h,000h,000h,000h
DCB     03eh,051h,049h,045h,03eh,000h,000h,000h
DCB     000h,042h,07fh,040h,000h,000h,000h,000h
DCB     072h,049h,049h,049h,046h,000h,000h,000h
DCB     021h,041h,049h,04dh,033h,000h,000h,000h
DCB     018h,014h,012h,07fh,010h,000h,000h,000h
DCB     027h,045h,045h,045h,039h,000h,000h,000h
DCB     03ch,04ah,049h,049h,031h,000h,000h,000h
DCB     041h,021h,011h,009h,007h,000h,000h,000h
DCB     036h,049h,049h,049h,036h,000h,000h,000h
DCB     046h,049h,049h,029h,01eh,000h,000h,000h
DCB     000h,000h,014h,000h,000h,000h,000h,000h
DCB     000h,040h,034h,000h,000h,000h,000h,000h
DCB     000h,008h,014h,022h,041h,000h,000h,000h
DCB     014h,014h,014h,014h,014h,000h,000h,000h
DCB     000h,041h,022h,014h,008h,000h,000h,000h
DCB     002h,001h,059h,009h,006h,000h,000h,000h
```

```
DCB     03eh,041h,05dh,059h,04eh,000h,000h,000h
DCB     07ch,012h,011h,012h,07ch,000h,000h,000h
DCB     07fh,049h,049h,049h,036h,000h,000h,000h
DCB     03eh,041h,041h,041h,022h,000h,000h,000h
DCB     07fh,041h,041h,041h,03eh,000h,000h,000h
DCB     07fh,049h,049h,049h,041h,000h,000h,000h
DCB     07fh,009h,009h,009h,001h,000h,000h,000h
DCB     03eh,041h,041h,051h,073h,000h,000h,000h
DCB     07fh,008h,008h,008h,07fh,000h,000h,000h
DCB     000h,041h,07fh,041h,000h,000h,000h,000h
DCB     020h,040h,041h,03fh,001h,000h,000h,000h
DCB     07fh,008h,014h,022h,041h,000h,000h,000h
DCB     07fh,040h,040h,040h,040h,000h,000h,000h
DCB     07fh,002h,01ch,002h,07fh,000h,000h,000h
DCB     07fh,004h,008h,010h,07fh,000h,000h,000h
DCB     03eh,041h,041h,041h,03eh,000h,000h,000h
DCB     07fh,009h,009h,009h,006h,000h,000h,000h
DCB     03eh,041h,051h,021h,05eh,000h,000h,000h
DCB     07fh,009h,019h,029h,046h,000h,000h,000h
DCB     026h,049h,049h,049h,032h,000h,000h,000h
DCB     003h,001h,07fh,001h,003h,000h,000h,000h
DCB     03fh,040h,040h,040h,03fh,000h,000h,000h
DCB     01fh,020h,040h,020h,01fh,000h,000h,000h
DCB     03fh,040h,038h,040h,03fh,000h,000h,000h
DCB     063h,014h,008h,014h,063h,000h,000h,000h
DCB     003h,004h,078h,004h,003h,000h,000h,000h
DCB     061h,059h,049h,04dh,043h,000h,000h,000h
DCB     000h,07fh,041h,041h,041h,000h,000h,000h
DCB     002h,004h,008h,010h,020h,000h,000h,000h
DCB     000h,041h,041h,041h,07fh,000h,000h,000h
DCB     004h,002h,001h,002h,004h,000h,000h,000h
DCB     040h,040h,040h,040h,040h,000h,000h,000h
DCB     000h,003h,007h,008h,000h,000h,000h,000h
DCB     020h,054h,054h,078h,040h,000h,000h,000h
DCB     07fh,028h,044h,044h,038h,000h,000h,000h
DCB     038h,044h,044h,044h,028h,000h,000h,000h
DCB     038h,044h,044h,028h,07fh,000h,000h,000h
DCB     038h,054h,054h,054h,018h,000h,000h,000h
DCB     000h,008h,07eh,009h,002h,000h,000h,000h
DCB     018h,0a4h,0a4h,09ch,078h,000h,000h,000h
DCB     07fh,008h,004h,004h,078h,000h,000h,000h
DCB     000h,044h,07dh,040h,000h,000h,000h,000h
DCB     020h,040h,040h,03dh,000h,000h,000h,000h
DCB     07fh,010h,028h,044h,000h,000h,000h,000h
```

```
DCB     000h,041h,07fh,040h,000h,000h,000h,000h
DCB     07ch,004h,078h,004h,078h,000h,000h,000h
DCB     07ch,008h,004h,004h,078h,000h,000h,000h
DCB     038h,044h,044h,044h,038h,000h,000h,000h
DCB     0fch,018h,024h,024h,018h,000h,000h,000h
DCB     018h,024h,024h,018h,0fch,000h,000h,000h
DCB     07ch,008h,004h,004h,008h,000h,000h,000h
DCB     048h,054h,054h,054h,024h,000h,000h,000h
DCB     004h,004h,03fh,044h,024h,000h,000h,000h
DCB     03ch,040h,040h,020h,07ch,000h,000h,000h
DCB     01ch,020h,040h,020h,01ch,000h,000h,000h
DCB     03ch,040h,030h,040h,03ch,000h,000h,000h
DCB     044h,028h,010h,028h,044h,000h,000h,000h
DCB     04ch,090h,090h,090h,07ch,000h,000h,000h
DCB     044h,064h,054h,04ch,044h,000h,000h,000h
DCB     000h,008h,036h,041h,000h,000h,000h,000h
DCB     000h,000h,077h,000h,000h,000h,000h,000h
DCB     000h,041h,036h,008h,000h,000h,000h,000h
DCB     002h,001h,002h,004h,002h,000h,000h,000h
DCB     03ch,026h,023h,026h,03ch,000h,000h,000h
DCB     01eh,0a1h,0a1h,061h,012h,000h,000h,000h
DCB     03ah,040h,040h,020h,07ah,000h,000h,000h
DCB     038h,054h,054h,055h,059h,000h,000h,000h
DCB     021h,055h,055h,079h,041h,000h,000h,000h
DCB     021h,054h,054h,078h,041h,000h,000h,000h
DCB     021h,055h,054h,078h,040h,000h,000h,000h
DCB     020h,054h,055h,079h,040h,000h,000h,000h
DCB     00ch,01eh,052h,072h,012h,000h,000h,000h
DCB     039h,055h,055h,055h,059h,000h,000h,000h
DCB     039h,054h,054h,054h,059h,000h,000h,000h
DCB     039h,055h,054h,054h,058h,000h,000h,000h
DCB     000h,000h,045h,07ch,041h,000h,000h,000h
DCB     000h,002h,045h,07dh,042h,000h,000h,000h
DCB     000h,001h,045h,07ch,040h,000h,000h,000h
DCB     0f0h,029h,024h,029h,0f0h,000h,000h,000h
DCB     0f0h,028h,025h,028h,0f0h,000h,000h,000h
DCB     07ch,054h,055h,045h,000h,000h,000h,000h
DCB     020h,054h,054h,07ch,054h,044h,000h,000h
DCB     07ch,00ah,009h,07fh,049h,041h,000h,000h
DCB     032h,049h,049h,049h,032h,000h,000h,000h
DCB     032h,048h,048h,048h,032h,000h,000h,000h
DCB     032h,04ah,048h,048h,030h,000h,000h,000h
DCB     03ah,041h,041h,021h,07ah,000h,000h,000h
DCB     03ah,042h,040h,020h,078h,000h,000h,000h
```

```
DCB     000h,09dh,0a0h,0a0h,07dh,000h,000h,000h
DCB     039h,044h,044h,044h,039h,000h,000h,000h
DCB     03dh,040h,040h,040h,03dh,000h,000h,000h
DCB     03ch,024h,0ffh,024h,024h,000h,000h,000h
DCB     048h,07eh,049h,043h,066h,000h,000h,000h
DCB     02bh,02fh,0fch,02fh,02bh,000h,000h,000h
DCB     0ffh,009h,029h,0f6h,020h,000h,000h,000h
DCB     0c0h,088h,07eh,009h,003h,000h,000h,000h
DCB     020h,054h,054h,079h,041h,000h,000h,000h
DCB     000h,000h,044h,07dh,041h,000h,000h,000h
DCB     030h,048h,048h,04ah,032h,000h,000h,000h
DCB     038h,040h,040h,022h,07ah,000h,000h,000h
DCB     000h,07ah,00ah,00ah,072h,000h,000h,000h
DCB     07dh,00dh,019h,031h,07dh,000h,000h,000h
DCB     026h,029h,029h,02fh,028h,000h,000h,000h
DCB     026h,029h,029h,029h,026h,000h,000h,000h
DCB     030h,048h,04dh,040h,020h,000h,000h,000h
DCB     038h,008h,008h,008h,008h,000h,000h,000h
DCB     008h,008h,008h,008h,038h,000h,000h,000h
DCB     02fh,010h,0c8h,0ach,0bah,000h,000h,000h
DCB     02fh,010h,028h,034h,0fah,000h,000h,000h
DCB     000h,000h,07bh,000h,000h,000h,000h,000h
DCB     008h,014h,02ah,014h,022h,000h,000h,000h
DCB     022h,014h,02ah,014h,008h,000h,000h,000h
DCB     0aah,000h,055h,000h,0aah,000h,000h,000h
DCB     0aah,055h,0aah,055h,0aah,055h,000h,000h
DCB     055h,0aah,055h,0aah,055h,0aah,000h,000h
DCB     000h,000h,000h,0ffh,000h,000h,000h,000h
DCB     010h,010h,010h,0ffh,000h,000h,000h,000h
DCB     014h,014h,014h,0ffh,000h,000h,000h,000h
DCB     010h,010h,0ffh,000h,0ffh,000h,000h,000h
DCB     010h,010h,0f0h,010h,0f0h,000h,000h,000h
DCB     014h,014h,014h,0fch,000h,000h,000h,000h
DCB     014h,014h,0f7h,000h,0ffh,000h,000h,000h
DCB     000h,000h,0ffh,000h,0ffh,000h,000h,000h
DCB     014h,014h,0f4h,004h,0fch,000h,000h,000h
DCB     014h,014h,017h,010h,01fh,000h,000h,000h
DCB     010h,010h,01fh,010h,01fh,000h,000h,000h
DCB     014h,014h,014h,01fh,000h,000h,000h,000h
DCB     010h,010h,010h,0f0h,000h,000h,000h,000h
DCB     000h,000h,000h,01fh,010h,010h,000h,000h
DCB     010h,010h,010h,01fh,010h,010h,000h,000h
DCB     010h,010h,010h,0f0h,010h,010h,000h,000h
DCB     000h,000h,000h,0ffh,010h,010h,000h,000h
```

```
DCB     010h,010h,010h,010h,010h,010h,000h,000h
DCB     010h,010h,010h,0ffh,010h,010h,000h,000h
DCB     000h,000h,000h,0ffh,014h,014h,000h,000h
DCB     000h,000h,0ffh,000h,0ffh,010h,000h,000h
DCB     000h,000h,01fh,010h,017h,014h,000h,000h
DCB     000h,000h,0fch,004h,0f4h,014h,000h,000h
DCB     014h,014h,017h,010h,017h,014h,000h,000h
DCB     014h,014h,0f4h,004h,0f4h,014h,000h,000h
DCB     000h,000h,0ffh,000h,0f7h,014h,000h,000h
DCB     014h,014h,014h,014h,014h,014h,000h,000h
DCB     014h,014h,0f7h,000h,0f7h,014h,000h,000h
DCB     014h,014h,014h,017h,014h,014h,000h,000h
DCB     010h,010h,01fh,010h,01fh,010h,000h,000h
DCB     014h,014h,014h,0f4h,014h,014h,000h,000h
DCB     010h,010h,0f0h,010h,0f0h,010h,000h,000h
DCB     000h,000h,01fh,010h,01fh,010h,000h,000h
DCB     000h,000h,000h,01fh,014h,014h,000h,000h
DCB     000h,000h,000h,0fch,014h,014h,000h,000h
DCB     000h,000h,0f0h,010h,0f0h,010h,000h,000h
DCB     010h,010h,0ffh,010h,0ffh,010h,000h,000h
DCB     014h,014h,014h,0ffh,014h,014h,000h,000h
DCB     010h,010h,010h,01fh,000h,000h,000h,000h
DCB     000h,000h,000h,0f0h,010h,010h,000h,000h
DCB     0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,000h,000h
DCB     0f0h,0f0h,0f0h,0f0h,0f0h,0f0h,000h,000h
DCB     0ffh,0ffh,0ffh,000h,000h,000h,000h,000h
DCB     000h,000h,000h,0ffh,0ffh,0ffh,000h,000h
DCB     00fh,00fh,00fh,00fh,00fh,00fh,000h,000h
DCB     038h,044h,044h,038h,044h,000h,000h,000h
DCB     07ch,02ah,02ah,03eh,014h,000h,000h,000h
DCB     07eh,002h,002h,006h,006h,000h,000h,000h
DCB     002h,07eh,002h,07eh,002h,000h,000h,000h
DCB     063h,055h,049h,041h,063h,000h,000h,000h
DCB     038h,044h,044h,03ch,004h,000h,000h,000h
DCB     040h,07eh,020h,01eh,020h,000h,000h,000h
DCB     006h,002h,07eh,002h,002h,000h,000h,000h
DCB     099h,0a5h,0e7h,0a5h,099h,000h,000h,000h
DCB     01ch,02ah,049h,02ah,01ch,000h,000h,000h
DCB     04ch,072h,001h,072h,04ch,000h,000h,000h
DCB     030h,04ah,04dh,04dh,030h,000h,000h,000h
DCB     030h,048h,078h,048h,030h,000h,000h,000h
DCB     0bch,062h,05ah,046h,03dh,000h,000h,000h
DCB     03eh,049h,049h,049h,000h,000h,000h,000h
DCB     07eh,001h,001h,001h,07eh,000h,000h,000h
```

```
    DCB     02ah,02ah,02ah,02ah,02ah,000h,000h,000h
    DCB     044h,044h,05fh,044h,044h,000h,000h,000h
    DCB     040h,051h,04ah,044h,040h,000h,000h,000h
    DCB     040h,044h,04ah,051h,040h,000h,000h,000h
    DCB     000h,000h,0ffh,001h,003h,000h,000h,000h
    DCB     0e0h,080h,0ffh,000h,000h,000h,000h,000h
    DCB     008h,008h,06bh,06bh,008h,008h,000h,000h
    DCB     036h,012h,036h,024h,036h,000h,000h,000h
    DCB     006h,00fh,009h,00fh,006h,000h,000h,000h
    DCB     000h,000h,018h,018h,000h,000h,000h,000h
    DCB     000h,000h,010h,010h,000h,000h,000h,000h
    DCB     030h,040h,0ffh,001h,001h,000h,000h,000h
    DCB     000h,01fh,001h,001h,01eh,000h,000h,000h
    DCB     000h,019h,01dh,017h,012h,000h,000h,000h
    DCB     000h,03ch,03ch,03ch,03ch,000h,000h,000h
    DCB     000h,000h,000h,000h,000h,000h,000h,000h
    DCB     0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh,0ffh


;*****************************************
; LDELAY --LONG DELAY
; MODIFY : NONE
;-----------------------------------------

LDELAY:
        PUSH    AX

        LD      AX,#10
LDEL1:
        CALL    DELAY
        DEC     AX
        JNE     LDEL1

        POP     AX
        RET

;*****************************************
; DELAY --LONG DELAY
; MODIFY : NONE
;-----------------------------------------

DELAY:
        PUSH    CX
```

```
        LD      CX,#5
DEL1:
        NOP
        DEC     CX
        JNE     DEL1

        POP     CX
        RET

;*****************************************
; LINE FEED
;-----------------------------------------
LINE_FEED:

        CALL    MCTCR_CN

        CALL    WAIT_HOME_OFF
        CALL    WAIT_HOME_ON

        CALL    MCTCR_OFF

        RET

;*****************************************
; WAIT_TG_CN
; input:  WAIT UNTIL  TG IS "CN "
;-----------------------------------------
WAIT_TG_CN:
        PUSH    AX
        PUSH    DX

WAIT_TG_C1:
        LDB     AL,ICPCRT0
        JBC     AL,1,WAIT_TG_C1
        CALL    DELAY
;
        LDB     AL,ICPCRT0
        JBC     AL,1,WAIT_TG_C1

        POP     DX
        POP     AX
        RET
```

```
;****************************************
; WAIT_TG_OFF
; input:  WAIT UNTIL  TG IS "OFF "
;----------------------------------------
WAIT_TG_OFF:
        PUSH    AX
        PUSH    CX

WAIT_TG_F1:

        LDB     AL,ICPCRT0
        JBS     AL,1,WAIT_TG_F1
        CALL    DELAY
;
        LDB     AL,ICPCRT0
        JBS     AL,1,WAIT_TG_F1

        POP     CX
        POP     AX
        RET

;****************************************
; WAIT_HOME_ON
; input:  WAIT UNTIL  HOME SWITCH IS "ON "
;----------------------------------------
WAIT_HOME_ON:
        PUSH    AX
        PUSH    CX

WAIT_H1_ON:

        LDB     AL,ICPCRT0
        JBC     AL,0,WAIT_H1_ON
        CALL    DELAY
;
        LDB     AL,ICPCRT0
        JBC     AL,0,WAIT_H1_ON

        POP     CX
        POP     AX
        RET

;****************************************
; WAIT_HOME_OFF
```

```
; input:  WAIT UNTIL  HOME SWITCH IS "OFF "
;----------------------------------------
WAIT_HOME_OFF:
        PUSH    AX
        PUSH    DX

WAIT_H1_OFF:

        LDB     AL,ICPCRT0
        JBS     AL,0,WAIT_H1_OFF
        CALL    DELAY

        LDB     AL,ICPCRT0
        JBS     AL,0,WAIT_H1_OFF

        POP     DX
        POP     AX
        RET

;*****************************************
; MOTOR_ON   - MOTOR ON ,NO PRINTING
; input:       motor on always" high"
;----------------------------------------
MOTOR_ON:
        PUSH    AX
        PUSH    DX

        LDB     AL,#80H
        STB     AL,ICPCRT1

        POP     DX
        POP     AX
        RET

;*****************************************
; MOTOR_OFF
; input:   NONE
;          MOTOR OFF, NO PRINTING
;----------------------------------------
MOTOR_OFF:
        PUSH    AX
        PUSH    DX

        LDB     AL,#00H
```

```
        STB     AL,ICPCRT1

        POP     CX
        POP     AX
        RET

;*****************************************
; DOT_ON   - PRINT DOT
; input:   al: dot 1..dot7
;          al: msb -- motor on always" high"
;-----------------------------------------
DOT_ON:
        PUSH    AX
        PUSH    CX

        CRB     AL,#80H
        STB     AL,ICPCRT1

        POP     CX
        POP     AX
        RET

;*****************************************
; DOT_OFF
; input:   NONE
;          motor on always" high"
;-----------------------------------------
DOT_OFF:
        PUSH    AX
        PUSH    CX

        LDB     AL,#80H
        STB     AL,ICPCRT1

        POP     CX
        POP     AX
        RET

;************************************
; SEIAL INITIAL PROCEDURE
; 8_BIT(.NO_PARITY)+1_STOP_BIT
; baud_val = (XTAL/64)/BAUD_RATE
; INPUT :NONE
; OUTPUT:BAUD_REG,SPCON,IM_SPCON,ICC1
```

```
; MODIFY:
;************************************

        baud_val        EQU     72                      ;(2400baud)
        BAUD_HIGH       EQU     ((baud_val-1)/256) OR 80H
        BAUD_LOW        EQU     (baud_val-1) MOD 256

SIO_INIT:

        LDB     BAUD_REG, #BAUD_LOW
        LDB     BAUD_REG, #BAUD_HIGH

        LDB     SPCON, #01011001B
        LDB     IM_SPCON, #01111001B
        RET
;************************************
; TXDATA PROCEDURE
; INPUT :AL
; OUTPUT:SBUF
; MODIFY:
;************************************
TXDATA:
        CRB     IM_SPCON,SPCON
        BBC     IM_SPCON, 5,TXDATA
        ANDB    IM_SPCON,#11011111B
        LDB     SBUF,AL
        RET

;************************************
; RXDATA_NO_WAIT PROCEDURE
; INPUT: SBUF
; OUTPUT:AL,Z FLAG ( IF RXDATA THEN Z =0 ELSE Z=1)
; MODIFY:
;************************************
RXDATA_NO_WAIT:
        PUSH    CX

        LD      CX,#0FFFFH
RX_NW1:
        DEC     CX
        CMP     CX,#0000H
        BE      RX_NW_EXIT              ; Z FLAG =1

        CRB     IM_SPCON,SPCON
```

```
        BBC     IM_SPCCN,6,RX_LNW1

        ANDB    IM_SPCCN,#10111111B
        LDB     AL,SBUF

        ADD     CX,#1                   ;Z FLAG =0

RX_LNW_EXIT:
        POP     CX

        RET
;-----------------------------------------

RXDATA_NC_WAIT:
        PUSH    CX

        LD      CX,#02000H
RX_NW1:
        DEC     CX
        CMP     CX,#0000H
        BE      RX_NW_EXIT              ; Z FLAG =1

        CRB     IM_SPCCN,SPCCN
        BBC     IM_SPCCN,6,RX_NW1

        ANDB    IM_SPCCN,#10111111B
        LDB     AL,SBUF

        ADD     CX,#1                   ;Z FLAG =0

RX_NW_EXIT:
        POP     CX

        RET

;***********************************
; RXDATA PRCCEDURE
; INPUT: SBUF
; CUTPUT:AL
; MODIFY:
;***********************************
RXDATA:
        CRB     IM_SPCCN,SPCCN
        BBC     IM_SPCCN, 6,RXDATA
```

```
        ANDB    IM_SPCCN,#10111111B
        LDB     AL,SBUF
        RET

;***********************************
; RX_STATUS
; INPUT: SBUF
; FLAG
; MODIFY:
;***********************************
RX_STATUS:
        CRB     IM_SPCCN,SPCCN
        RET

;***********************************
; RX_DATA PROCEDURE
; INPUT: SBUF
; OUTPUT:AL
; MODIFY:
;***********************************
RX_DATA:

        ANDB    IM_SPCCN,#10111111B
        LDB     AL,SBUF
        RET

CR      EQU     0DH
LF      EQU     0AH

;*********************************
; TRANSMIT CR,LF,(BELL)
; INPUT: NONE
; OUTPUT:SBUF
; MODIFY:NONE
;*********************************

CRLF:   PUSH    AX
        LDB     AL,#CR
        CALL    TXDATA
        LDB     AL,#LF
        CALL    TXDATA
        LDB     AL,#' '
        CALL    TXDATA
        LDB     AL,#' '
```

```
        CALL    TXDATA
        POP     AX
        RET

CR_:    PUSH    AX
        LDB     AL,#CR
        CALL    TXDATA
        POP     AX
        RET

;***********************************
; 1 HEX TO 2 ASCII PROCEDURE
; INPUT  : AL
; OUTPUT : ASC_H,ASC_L
; MODIFY : NONE
;***********************************
HTOA2:
        PUSH    AX
        PUSH    BX

        LDB     AH,AL
        SHRB    AL,#04

        CALL    HTOA
        LDB     ASC_H,AL

        LDB     AL,AH
        CALL    HTOA
        LDB     ASC_L,AL


        POP     BX
        POP     AX
        RET

HTOA1:  CALL    HTOA
        CALL    TXDATA
        RET

HTOA:   ANDB    AL,#0FH
        LDB     BHH,#00H
        LDB     BL,AL
        LDB     AL,T_HTOA[BX]
        RET
```

```
T_HTCA: DCB     '0123456789ABCDEF'

;***********************************
;2 ASCII TO 1 HEX CONVERT
; INPUT  : [BX]:msb,[BX+1]:lsb
; OUTPUT : AL, C(SET IF ERROR)
; MODIFY : AX
;***********************************
ATCH2:
        PUSH    BX

        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        SHLB    AL,#04
        LDB     AH,AL
        INC     BX
        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        CRB     AL,AH

AT1:    POP     BX
        RET

ATOH:   CMPB    AL,#2FH
        BNH     ATO8
        CMPB    AL,#39H
        BH      ATO2
        BR      ATO6

ATO2:   CMPB    AL,#40H
        BNH     ATO8
        CMPB    AL,#46H
        BH      ATO8
        ADDB    AL,#09H

ATO6:   ANDB    AL,#0FH
        CLRC
        BR      ATO9
ATO8:
        SETC                            ;ERROR NOT HEX
```

```
ATC9:   RET

;-------------------------------------

BINTCDEC:

        PUSH    BX
        PUSH    DX

        LD      DX,#0000H                       ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                         ;DAX DIVIDE BY 10
        STB     DL,BL

        LD      DX,#0000H                       ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                         ;DAX DIVIDE BY 10
        SHLB    DL,#4                           ;STORE LSB
        ORB     BL,DL

        LD      DX,#0000H                       ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                         ;DAX DIVIDE BY 10
        STB     DL,BHH

        LD      DX,#0000H                       ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                         ;DAX DIVIDE BY 10
        SHLB    DL,#4
        ORB     BHH,DL                          ;STORE MSB

        ST      BX,AX

        POP     DX
        POP     BX

        RET

;*****************************************************
; TIMER OVERFLOW INTERRUPT SERVICE ROUTINE AT #0900OH
;       USING PWM;
;         (LOB   PWM_CONTROL,#80H)
;         OUTPUT: COUNT +1
;-----------------------------------------------------

TIMER_INT:

        PUSHF
```

```
        INC     TIME

        POPF

        EI
        RET

;*************************************
; INTERRUPT INITIALIZE
;     :ENABLE,SOFT TIMER
;     :ENABLE ADC0,ADC2
;-------------------------------------
INIT_SYS:
        PUSH    AX

        LDB     AL,ICPCRT2
        ANDB    AL,#7FH
        STB     AL,ICPCRT2

        CALL    SIO_INIT

        CALL    DELAY

        CALL    LINE_FEED

        LD      TIME,#0000H

        ;INTERRUPT MASK REGISTIER SET
        LDB     INT_MASK,#00000001B             ;ENABLE TIMER_OVERFLOW

        LDB     ICC0,#00000000B                 ;TIMER2CLK
        LDB       ICC1,#00101001B                       ;PWM,TIMER2 OVERFLOW
ENABLE, HOLDING REG LOAD
        LDB     PWM_CONTRCL,#80H                ;CUT PWM

        CLRB    INT_PENDING

        EI

        POP     AX

        RET
        END
```

■첨부4) 중앙 감시 보드 소프트 웨어.

```
;PROGRAM CDP8T.ASM, 96.9.30
$INCLUDE (8096.INC)
        ; DEFINE SYMBOL
        ; OUTPUT RELAY SYMBOL
          RSEG    AT      1AH

        DAX:            DSL     1
        AX              EQU     DAX:WORD
        AL              EQU     AX:BYTE
        AH              EQU     AX+1:BYTE
        DX              EQU     DAX+2:WORD
        DL              EQU     DX:BYTE
        DH              EQU     DX+1:BYTE
        BX:             DSW     1
        BL              EQU     BX:BYTE
        BHH             EQU     BX+1:BYTE
        CX:             DSW     1
        CL              EQU     CX:BYTE
        CH              EQU     CX+1:BYTE
        SIX:            DSW     1
        DIX:            DSW     1
        IM_SPCON:       DSB     1

        CHECK_SUM:      DSB     1

        P8255CE         EQU     0C010H
        G244            EQU     0C014H          ;& !RD

        PIC1            EQU     0C000H          ;& !WR
        PIC2            EQU     0C004H          ;& !WR

        SW_LOW          EQU     0C014H
        SW_HIGH         EQU     ICPCRT0

;***************************************
CSEG            AT      2000H

CSEG            AT      2018H
CCR:            DCB     0FDH

CSEG            AT      2080H           ;RESET
```

```
;***********************************

START:
        ;CHANNEL SWITCH

        LD      SP,#0B0H
        LDB     ICPCRT1,#0FH

        CALL    SIO_INIT
        CALL    P8255_INIT

        CALL    RX_STATUS
        CALL    RX_DATA

        ;WAIT
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY

START0:

        LD      BX,#0
START1:
        LD      CX,#SW_LOW
        LDB     AL,[CX]
        LDB     AH,SW_HIGH

        LDB     CL,BL
        INCB    CL

        SHR     AX,CL
        JC      CHANNEL_ENABLE

CHANNEL_DISABLE:

        CALL    DISPLAY_OFF

        BR      CHECK_NEXT

CHANNEL_ENABLE:
```

```
        CALL    GET_CH_STATUS

        JNC     DISPLAY_LED

        CALL    SET_ERRCR_CH

DISPLAY_LED:

        CALL    DISPLAY_OUT

        CALL    CHECK_WARNING

DISPLAY_PC:

        CALL    SEND_STATUS_PC

        CALL    SEND_STATUS_MODEM

        CALL    SEND_STATUS_PRT

CHECK_NEXT:

        INC     BX
        CMP     BX,#9
        JNH     NEXT_CH

        BR      START0

NEXT_CH:

        BR      START1
;--------------------------------------
; GET CHANNEL STATUS
; INPUT :  BX=  POINTER OF CHANNEL
; OUTPUT:  C FLAG=1 IF ERROR
;--------------------------------------
GET_CH_STATUS:

        PUSH    AX
        PUSH    BX
        PUSH    DX

        ;CHANNEL SWITCH
        LDB     ICPORT1,BL
```

```
        ;WAIT
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY

        ;SET DATA POINTER IN EX
        LD      CX,#D0_MODEL_NO                 ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                           ;POINTER*16
        ADD     BX,CX

        ;EMPTIFY RX BUFFER
        CALL    RX_STATUS
        BBC     IM_SPCCN,6,EMP_RX1
        CALL    RX_DATA

EMP_RX1:
        ;SEND 'R'
        LDB     AL,#' '
        CALL    TXDATA
        LDB     AL,#'R'
        CALL    TXDATA

        ;RECEIVE DATA

        SETC                    ;SET ERRCR FLAG

        LDB     CL,#11
REC_DAT1:
        CALL    RXDATA_NO_WAIT
        BE      REC_EXIT

        STB     AL,[EX]+

        DJNZ    CL,REC_DAT1

        CLRC                    ;CLEAR ERRCR FLAG
REC_EXIT:
        POP     CX
        POP     BX
        POP     AX
        RET
```

```
;-------------------------------------
; SEND  DATA OF CHANNEL STATUS TO PC
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------
 SEND_STATUS_PC:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        ;CHANNEL SWITCH
        LDB     ICPORT1,#0FH

        ;WAIT
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY

        LDB     AL,#'R'
        CALL    TXDATA

        CALL    RXDATA_NO_WAIT
        BE      MODEM_EXIT
        CMPB    AL,#'R'
        BNE      MODEM_EXIT


        CALL    SEND_STATUS

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;-------------------------------------
; SEND  DATA OF CHANNEL STATUS TO MODEM
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------
 SEND_STATUS_MODEM:
```

```
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        ;CHANNEL SWITCH
        LDB     ICPCRT1,#0AH

        ;WAIT
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY

        LDB     AL,#'R'
        CALL    TXDATA

        CALL    RXDATA_NO_WAIT
        BE      MCDEM_EXIT
        CMPB    AL,#'R'
        BNE      MODEM_EXIT


        CALL    SEND_STATUS

MCDEM_EXIT:

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;--------------------------------------
; SEND  DATA OF CHANNEL STATUS TO PRT
; INPUT :  EX=  POINTER OF CHANNEL
;--------------------------------------
 SEND_STATUS_PRT:

        PUSH    AX
        PUSH    BX
```

```
        PUSH    CX
        PUSH    DX

        ;CHANNEL SWITCH
        LDB     ICPCRT1,#0BH

        ;WAIT
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY

        LDB     AL,#'R'
        CALL    TXDATA

        CALL    RXDATA_NO_WAIT
        BE      PRT_EXIT
        CMPB    AL,#'R'
        BNE      PRT_EXIT

        CALL    SEND_STATUS

PRT_EXIT:

;       CALL    SEND_STATUS

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;-------------------------------------
; SEND  DATA OF CHANNEL STATUS TO
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------
 SEND_STATUS:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        LDB     AL,BL
        LDB     CHECK_SUM,BL
        CALL    TXDATA

        ;SET DATA POINTER IN EX
        LD      DX,#DO_DRY_STEP             ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                       ;POINTER*16
        ADD     BX,DX

        ;SEND DATA TO PC

        LDB     CL,#9

 SEND_ST1:

        LDB     AL,[BX]+
        ADDB    CHECK_SUM,AL
        CALL    TXDATA                      ;HTCA2

        DJNZ    CL,SEND_ST1

        LDB     AL,CHECK_SUM
        CALL    TXDATA
                                            ;CALL    CR_
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
        CALL    DELAY
                                            ;CALL    CR_
        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;--------------------------------------
; DISPLAY OUT TO DISPLAY BOARD
; INPUT :  EX=  POINTER OF CHANNEL
;--------------------------------------
DISPLAY_OUT:

        PUSH    AX
        PUSH    BX
        PUSH    CX
```

```
        PUSH    DX

        LD      DX,#DO_DRY_STEP             ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                       ;POINTER*16

        LDB     AH,BL                       ;BOARD SELECTOR ON AH
        ADD     BX,DX

        LDB     DL,#01H
        CRB     DL,AH
        LDB     AL,5[BX]                     ;LED1
        CALL    DATA_OUT

        LDB     DL,#02H
        CRB     DL,AH
        LDB     AL,4[BX]                     ;TEMP_WET
        CALL    DATA_OUT

        LDB     DL,#03H
        CRB     DL,AH
        LDB     AL,3[BX]                     ;TEMP_DRY
        CALL    DATA_OUT

        LDB     DL,#04H
        CRB     DL,AH
        LDB     AL,1[BX]                       ;DRY_STEP
        CALL    DATA_OUT

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;-------------------------------------
; DISPLAY OFF
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------
DISPLAY_OFF:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX
```

```
        LD      DX,#CO_DRY_STEP             ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                       ;POINTER*16

        LDB     AH,BL                       ;BOARD SELECTOR ON AH
        ADD     BX,DX

        LDB     DL,#01H
        CRB     DL,AH
        LDB     AL,#0FFH                     ;LOAD OFF DATA
        STB     AL,5[BX]                     ;LED1
        CALL    DATA_OUT

        LDB     DL,#02H
        CRB     DL,AH
        LDB     AL,#0FFH                     ;LOAD OFF DATA
        STB     AL,4[BX]                     ;TEMP_WET
        CALL    DATA_OUT

        LDB     DL,#03H
        CRB     DL,AH
        LDB     AL,#0FFH                     ;LOAD OFF DATA
        STB     AL,3[BX]                     ;TEMP_DRY
        CALL    DATA_OUT

        LDB     DL,#04H
        CRB     DL,AH
        LDB     AL,#0FFH                     ;LOAD OFF DATA
        STB     AL,1[BX]                     ;DRY_STEP
        CALL    DATA_OUT

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET


;-------------------------------------
; SET ERROR ON CHENNEL
; INPUT :  EX=  POINTER OF CHANNEL
;-------------------------------------
SET_ERROR_CH:
```

```
        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        LD      DX,#DO_DRY_STEP             ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                       ;POINTER*16

        LDB     AH,BL                       ;BOARD SELECTOR ON AH
        ADD     BX,DX

        LDB     DL,#01H
        CRB     DL,AH
        LDB     AL,5[BX]                     ;LED1

        ANDB     AL,#7FH                          ;SET ERROR ON CHANNEL
LINE       CRB      AL
        STB     AL,5[BX]                     ;STORE ERROR

        CALL    DATA_OUT

        POP     DX
        POP     CX
        POP     BX
        POP     AX
        RET

;--------------------------------------
; CHECK LED FOR CHECKING WARNING
; INPUT :  BX=  POINTER OF CHANNEL
;--------------------------------------

CHECK_WARNING:

        PUSH    AX
        PUSH    BX
        PUSH    CX
        PUSH    DX

        LD      DX,#DO_DRY_STEP             ;BASE ADDRESS OF CH DATA
        SHL     BX,#4                       ;POINTER*16

        LDB     AH,BL                       ;BOARD SELECTOR ON AH
        ADD     BX,DX
```

```
        LDB     CL,#01H
        CRB     CL,AH
        LDB     AL,5[BX]                        ;LED1

        CMPB    AL,#0FEH
        JE      NC_ERRCR_LED

ERROR_LED:

        LD      CX,#0C012H
        LDB     AL,[CX]
        CRB     AL,#01H
        STB     AL,[CX]

        ER      CHK_EXIT
NC_ERROR_LED:

        LD      CX,#0C012H
        LDB     AL,[CX]
        ANDB     AL,#0FEH
        STB     AL,[CX]

CHK_EXIT:
        POP     CX
        POP     CX
        POP     BX
        POP     AX
        RET

;-------------------------------------
; DATA OUT TP DISPLAY BCARD
;INPUT :  AL= DATA
;         CL= ADDRESS OF CHANNEL   LOW4=LED SWLECTOR,HIGH4=BOARD SELECTCR
;-------------------------------------
DATA_OUT:
        PUSH    BX
        PUSH    CX

        LD      BX,#0C004H
        STB     AL,[BX]

        LD      BX,#0C000H
        CRB     CL,#08H
```

```
        STB     DL,[BX]

        ANDB    DL,#0F7H
        STB     DL,[BX]
        NOP
        NOP
        ORB     DL,#08H
        STB     DL,[BX]

        POP     DX
        POP     BX
        RET

;************************************
; SEIAL INITIAL PROCEDURE
; 8_BIT(.NO_PARITY)+1_STOP_BIT
; baud_val = (XTAL/64)/BAUD_RATE
; INPUT :NONE
; OUTPUT:BAUD_REG,SPCON,IM_SPCON,ICC1
; MODIFY:
;************************************

        baud_val        EQU     72                      ;(2400baud)
        BAUD_HIGH       EQU     ((baud_val-1)/256) OR 80H
        BAUD_LOW        EQU     (baud_val-1) MOD 256

SIO_INIT:
        LDB     ICC1, #00100000B
        LDB     BAUD_REG, #BAUD_LOW
        LDB     BAUD_REG, #BAUD_HIGH

        LDB     SPCON, #01011001B
        LDB     IM_SPCON, #01111001B
        RET

;************************************
; INTEL 8255 INITIAL PROCCEDURE
;************************************
P8255_INIT:
        PUSH    DX
        PUSH    AX
                                ;  P8255CE  EQU     0C010H
        LD      DX,#0C013H
        LDB     AL,#90H
```

```
        STB     AL,[DX]

        LDB     AL,#0FFH
        LDB     DX,#SW_HIGH
        STB     AL,[DX]

        POP     AX
        POP     DX

        RET
;***********************************
; TXDATA PROCEDURE
; INPUT :AL
; OUTPUT:SBUF
; MODIFY:
;***********************************
TXDATA:
        CRB     IM_SPCON,SPCON
        BBC     IM_SPCON, 5,TXDATA
        ANDB    IM_SPCON,#11011111B
        LDB     SBUF,AL
        RET

;***********************************
; RXDATA PROCEDURE
; INPUT: SBUF
; OUTPUT:AL
; MODIFY:
;***********************************
RXDATA:
        CRB     IM_SPCON,SPCON
        BBC     IM_SPCON, 6,RXDATA
        ANDB    IM_SPCON,#10111111B
        LDB     AL,SBUF
        RET

;***********************************
; RX_STATUS
; INPUT: SBUF
; FLAG
; MODIFY:
;***********************************
RX_STATUS:
        CRB     IM_SPCON,SPCON
```

```
        RET

;***********************************
; RX_DATA PROCEDURE
; INPUT: SBUF
; OUTPUT:AL
; MODIFY:
;***********************************
RX_DATA:

        ANDB    IM_SPCON,#10111111B
        LDB     AL,SBUF
        RET

;***********************************
; RXDATA_NC_WAIT PROCEDURE
; INPUT: SBUF
; OUTPUT:AL,Z FLAG ( IF RXDATA THEN Z =0 ELSE Z=1)
; MODIFY:
;***********************************
RXDATA_NC_WAIT:
        PUSH    CX

        LD      CX,#0F000H
RX_NW1:
        DEC     CX
        CMP     CX,#0000H
        BE      RX_NW_EXIT          ; Z FLAG =1

        ORB     IM_SPCON,SPCON
        BBC     IM_SPCON,6,RX_NW1

        ANDB    IM_SPCON,#10111111B
        LDB     AL,SBUF

        ADD     CX,#1               ; Z FLAG =0

RX_NW_EXIT:
        POP     CX

        RET

CR      EQU     0DH
LF      EQU     0AH
```

```
ESC     EQU     0H

;*********************************
; TRANSMIT CR,LF,(BELL)
; INPUT: NONE
; OUTPUT:SBUF
; MODIFY:NONE
;*********************************

CRLF:   PUSH    AX
        LDB     AL,#CR
        CALL    TXDATA
        LDB     AL,#LF
        CALL    TXDATA
        LDB     AL,#' '
        CALL    TXDATA
        LDB     AL,#' '
        CALL    TXDATA
        POP     AX
        RET

CR_:    PUSH    AX
        LDB     AL,#CR
        CALL    TXDATA
        POP     AX
        RET

;************************************
; 1 HEX TO 2 ASCII PROCEDURE
; INPUT  : AL
; OUTPUT : SBUF
; MODIFY : NONE
;************************************
HTOA2:
        PUSH    AX
        PUSH    BX

        LDB     AH,AL
        SHRB    AL,#04
        CALL    HTOA
        CALL    TXDATA
        LDB     AL,AH
        CALL    HTOA
        CALL    TXDATA
```

```
        POP     BX
        POP     AX
        RET

HTCA1:  CALL    HTOA
        CALL    TXDATA
        RET

HTOA:   ANDB    AL,#0FH
        LDB     BHH,#00H
        LDB     BL,AL
        LDB     AL,T_HTCA[BX]
        RET

T_HTCA: DCB     '0123456789ABCDEF'

;************************************
;2 ASCII TO 1 HEX CONVERT
; INPUT  : [BX]:msb,[BX+1]:lsb
; OUTPUT : AL, C(SET IF ERROR)
; MODIFY : AX
;************************************
ATOH2:
        PUSH    BX

        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        SHLB    AL,#04
        LDB     AH,AL
        INC     BX
        LDB     AL,[BX]
        CALL    ATOH
        BC      AT1
        ORB     AL,AH

AT1:    POP     BX
        RET

ATOH:   CMPB    AL,#2FH
        BNH     ATO8
        CMPB    AL,#39H
        BH      ATO2
```

```
        BR      ATO6

ATO2:   CMPB    AL,#40H
        BNH     ATO8
        CMPB    AL,#46H
        BH      ATO8
        ADDB    AL,#09H

ATO6:   ANDB    AL,#0FH
        CLRC
        BR      ATO9
ATO8:
        SETC                            ;ERROR NOT HEX

ATO9:   RET

;---------------------------------------

BINTODEC:

        PUSH    BX
        PUSH    DX

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        STB     DL,BL

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        SHLB    DL,#4                   ;STORE LSB
        ORB     BL,DL

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        STB     DL,BHH

        LD      DX,#0000H               ;DX=0000,AX=NUMBER
        DIVU    DAX,#10                 ;DAX DIVIDE BY 10
        SHLB    DL,#4
        ORB     BHH,DL                  ;STORE MSB

        ST      BX,AX

        POP     DX
```

```
        POP     BX

        RET

;-------------------------------------

DELAY:
        PUSH    CX

        LD      CX,#0FFFFH
DEL1:   NOP
        DEC     CX
        JNE     DEL1

        POP     CX
        RET

;-------------------------------------

DSEG      AT     9000H

DO_MODEL_NO:    DSB     1
DO_DRY_STEP:    DSB     1
DO_TIME_STEP:   DSB     1

DO_TEMP_DRY:    DSB     1
DO_TEMP_WET:    DSB     1
DO_LED1:        DSB     1

DO_RELAY_VALUE: DSB     1
DO_CT_VALUE:    DSB     1
DO_TIME_TOTAL:  DSB     1
DO_SUM:         DSB     1

DO_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D1_MODEL_NO:    DSB     1
D1_DRY_STEP:    DSB     1
D1_TIME_STEP:   DSB     1
```

```
D1_TEMP_DRY:     DSB      1
D1_TEMP_WET:     DSB      1
D1_LED1:         DSB      1

D1_RELAY_VALUE:  DSB      1
D1_CT_VALUE:     DSB      1
D1_TIME_TOTAL:   DSB      1
D1_SUM:          DSB      1

D1_ERROR_WORK:   DSB      1
                 DSB      1
                 DSB      1
                 DSB      1
                 DSB      1
                 DSB      1

D2_MODEL_NO:     DSB      1
D2_DRY_STEP:     DSB      1
D2_TIME_STEP:    DSB      1

D2_TEMP_DRY:     DSB      1
D2_TEMP_WET:     DSB      1
D2_LED1:         DSB      1

D2_RELAY_VALUE:  DSB      1
D2_CT_VALUE:     DSB      1
D2_TIME_TOTAL:   DSB      1
D2_SUM:          DSB      1

D2_ERROR_WORK:   DSB      1
                 DSB      1
                 DSB      1
                 DSB      1
                 DSB      1
                 DSB      1

D3_MODEL_NO:     DSB      1
D3_DRY_STEP:     DSB      1
D3_TIME_STEP:    DSB      1

D3_TEMP_DRY:     DSB      1
D3_TEMP_WET:     DSB      1
D3_LED1:         DSB      1
```

```
D3_RELAY_VALUE: DSB     1
D3_CT_VALUE:    DSB     1
D3_TIME_TOTAL:  DSB     1
D3_SUM:         DSB     1

D3_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D4_MODEL_NO:    DSB     1
D4_DRY_STEP:    DSB     1
D4_TIME_STEP:   DSB     1

D4_TEMP_DRY:    DSB     1
D4_TEMP_WET:    DSB     1
D4_LED1:        DSB     1

D4_RELAY_VALUE: DSB     1
D4_CT_VALUE:    DSB     1
D4_TIME_TOTAL:  DSB     1
D4_SUM:         DSB     1

D4_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D5_MODEL_NO:    DSB     1
D5_DRY_STEP:    DSB     1
D5_TIME_STEP:   DSB     1

D5_TEMP_DRY:    DSB     1
D5_TEMP_WET:    DSB     1
D5_LED1:        DSB     1

D5_RELAY_VALUE: DSB     1
D5_CT_VALUE:    DSB     1
D5_TIME_TOTAL:  DSB     1
D5_SUM:         DSB     1
```

```
D5_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D6_MODEL_NO:    DSB     1
D6_DRY_STEP:    DSB     1
D6_TIME_STEP:   DSB     1

D6_TEMP_DRY:    DSB     1
D6_TEMP_WET:    DSB     1
D6_LED1:        DSB     1

D6_RELAY_VALUE: DSB     1
D6_CT_VALUE:    DSB     1
D6_TIME_TOTAL:  DSB     1
D6_SUM:         DSB     1

D6_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D7_MODEL_NO:    DSB     1
D7_DRY_STEP:    DSB     1
D7_TIME_STEP:   DSB     1

D7_TEMP_DRY:    DSB     1
D7_TEMP_WET:    DSB     1
D7_LED1:        DSB     1

D7_RELAY_VALUE: DSB     1
D7_CT_VALUE:    DSB     1
D7_TIME_TOTAL:  DSB     1
D7_SUM:         DSB     1

D7_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
```

```
                DSB     1
                DSB     1
                DSB     1

D8_MODEL_NO:    DSB     1
D8_DRY_STEP:    DSB     1
D8_TIME_STEP:   DSB     1

D8_TEMP_DRY:    DSB     1
D8_TEMP_WET:    DSB     1
D8_LED1:        DSB     1

D8_RELAY_VALUE: DSB     1
D8_CT_VALUE:    DSB     1
D8_TIME_TOTAL:  DSB     1
D8_SUM:         DSB     1

D8_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1

D9_MODEL_NO:    DSB     1
D9_DRY_STEP:    DSB     1
D9_TIME_STEP:   DSB     1

D9_TEMP_DRY:    DSB     1
D9_TEMP_WET:    DSB     1
D9_LED1:        DSB     1

D9_RELAY_VALUE: DSB     1
D9_CT_VALUE:    DSB     1
D9_TIME_TOTAL:  DSB     1
D9_SUM:         DSB     1

D9_ERROR_WORK:  DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
                DSB     1
END
```