GOVP1200211386

최 종
연구보고서

지하수 자동 수위관측기 개발 연구

A Research for the Development of Automatic Groundwater Level Meter

연 구 기 관
농 업 기 반 공 사

농 림 부

제 출 문

농림부 장관 귀하

본 보고서를 "지하수 자동 수위관측기 개발연구" 과제의 최종보고서로 제출합니다.

2001년 12 월 일

주관연구기관명 : 농업기반공사 농어촌연구원

총괄연구책임자 : 농어촌연구원 김 경 만

세부연구책임자 : 농어촌연구원 정 형 재

연 구 원 : 농어촌연구원 송 성 호

연 구 원 : 농어촌연구원 배 광 옥

연 구 원 : 농어촌연구원 김 대 년

연 구 원 : 경원엔지니어링 원 용 복

연 구 원 : 경원엔지니어링 강 경 원

요 약 문

Ⅰ. 제 목 : 지하수 자동 수위관측기 개발 연구

Ⅱ. 연구개발의 목적 및 중요성

인구 증가, 산업의 발전 등과 맞물려 대기와 함께 수자원도 급격히 오염이 진행되면서, 마시는 물을 중심으로 청정 수자원인 지하수의 수요가 급증하고 있으며, 이와 관련되어 지하수 보존량의 감소와 수질의 악화가 새로운 문제로 대두되고 있다.

지표수는 수량과 수질의 변화가 비교적 쉽게 확인되지만, 지하수는 육안 관찰이 불가능하고 지하수를 포함하고 있는 대수층이 파괴되거나 수질이 악화되었을 경우 이의 원상 회복에는 많은 비용과 긴 시간을 필요로 한다. 따라서 지하수의 안정적인 확보와 관리를 위해서 지하수의 수량과 수질을 장기적·주기적으로 측정하여 이 자료를 바탕으로 관리가 이루어져야 한다. 이를 위항 정부에서는 전국에 걸쳐 지하수 자동관측기를 설치/운영하고 있다.

지하수 관측망으로 국내에 설치되어 있거나 사용중인 지하수 관측기는 제작사별 고유 모델로 개발되어 있어 다른 제품과는 운영체계가 상이하여 호환이 거의 불가능하고, 한가지 장비에 익숙하더라도 다른 장비의 조작이나 점검·정비가 매우 어렵게 제작되어 있다. 따라서 관측망 운영자의 점검·보정으로 해결할 수 없는 고장이나 부품의 교체 등이 필요한 경우 제작사의 A/S가 필수적인데, 이에 소요되는 시간과 비용은 효율적인 관측망 운영을 어렵게 하고 있다. 이런 이유 때문에 지하수 관측자료집에는 누락된 자료가 상당히 있다. 한번 고장으로 1~2개월, 경우에 따라서는 6개월 이상이 소요되기도 하며, 소요경비 또한 구입가격의 1/3 이상~2/3까지 되는 경우도 있다. 수리기간동안 자료 취득이 불가능하고 소요경비 또한 만만치 않다. 또한 장비가 고가이기 때문에 예비 장비를 확보하는 것도 쉽지 않다.

주요 관정에 지하수 자동관측기를 설치하려면 보호시설을 제외하고 설치비가 500만원 이상 소요된다. 최근 생활용수, 농업용수용 지하수 관정에 관측시설을 설치하는 것을 의무화되는 추세이다. 이를 국산화할 경우 상당한 예산절감효과와 수입대체효과를 가져올 수 있다. 또한 실시간으로 사후관리가 가능하여 자료 누락이나 유지관리비용을 절감할 수 있다.

본 연구는 이러한 차원에서 국내 기술로 지하수 자동관측기를 제작하여 이를 지하수 개발사업, 지하수 장기관측 사업, 지하수 보존관리 등 적극 활용하고자 한다.

III. 연구개발의 목표 및 내용

본 연구는 지하수 전문가들의 현장 사용시 문제점, 개선사항 등을 반영하여 현장 사용자 중심으로 지하수 관측기의 하드웨어와 프로그램을 개발하며 기존 지하수 관측기와 비교하여 기능과 성능면에서 대등할 정도로 하며, 가격 경쟁력을 가진 지하수 자동관측기를 개발하는 것을 목표로 한다.

또한 회로내에서 필터회로 등을 추가하여 전기적 영향을 받지 않도록 하며, 관측기의 특성상 장거리 전송임을 감안하여 센서의 최종출력을 4~20㎃ 전류신호가 되도록 한다. 측정항목은 지하수위, 전기전도도, 온도로 하였다. 당초 연구계획서에서는 지하수위만 측정할 수 있는 시스템 개발을 최종 목적으로 하였지만, 지하수 수위만 측정하는 관측망은 없기 때문에 수위센서만을 개발하는 것이 다소 무의미할 수 있다. 이러한 점을 감안하여 본 연구에서는 수위센서뿐만 아니라 전기전도도센서, 온도센서를 동시에 개발하며, 자료저장, 전송, 표시 등 지하수 자동관측기를 제어하는 자료처리장치 또한 개발하고자 한다.

Ⅳ. 연구개발 결과 및 활용에 대한 건의

1. 연구결과

기존 지하수 자동관측기의 장단점 분석과 현장 사용에서 개선사항 등을 반영하여 지하수 관측기 시작품을 제작하였다. 시작품은 3개의 센서(수위, 전기전도도, 온도), 케이블, 자료처리장치 그리고 자료송수신용 프로그램이다. 시작품은 기존 지하수 관측기와 비교하여 기능과 성능면에서 대등하며, 기존 제품의 2/3가격으로 제작 가능하여 가격 경쟁력을 가질 수 있었다.

가. 수위센서, 전기전도도센서, 온도센서

수위센서는 관측공 최소구경이 54㎜(N규격)에 사용할 수 있도록 직경 25㎜, 커넥트 부분을 포함하여 길이는 145㎜로 소형으로 제작하였다. 측정범위는 0~100m으로 하였으며, 정밀도는 ±0.1%FS이다. 모터에 의한 노이즈 영향을 제거하기 위하여 노이즈 제거회로를 추가하였으며 센서 앞부분에 보호막을 설계, 설치하여 압력센서를 보호하였다. 그리고 수압 10기압에도 견딜 수 있도록 고무링과 실리콘 수지로 완벽한 방수처리를 하였다.

전기전도도 센서는 온도에 영향을 받기 때문에 전기전도도센서에 온도보상회로를 첨가하면서 두 항목을 함께 측정할 수 있도록 하였다. 전기전도도의 정밀도를 높게하기 위하여 4전극방식으로 저농도센서와 고농도센서 두가지 형태로 제작하였다. 저농도 센서의 측정범위는 0~5,000㎲/cm이며 정밀도는 ±0.25%FS이며, 고농도 센서의 측정범위는 0~50,000㎲/cm이며 정밀도는 ±0.5%FS이다. 전기전도도센서의 직경은 수위센서와 동일하게 25㎜로 하였으며, 길이는 전기전도도의 회로기판의 크기에 맞게 400㎜로 제작하였다. 그리고 전기전도도 전극이 있는 부분에는 실리콘 수지로 완전 방수 처리하였다. 온도센서는 Pt100으로 제작하였으며 측정범위는 -20~50℃로 하였다. 정밀도는 ±0.1%이다.

지하수 자동관측기의 센서와 자료처리장치는 100m이상의 장거리이므로 센서의 출력신호를 4~20mA로 하였다.

나. 센서케이블

수위센서의 대기압 보정을 위하여 튜브가 있는 지하수 자동관측기 전용 케이블을 제작하였다. 제작단가는 수입단가보다 약 1/3수준으로 가격 경쟁력을 가질 수 있게 하였다. 중량면에서도 110m에 링을 포함하여 7.5kg밖에 되지 않아 케이블이 늘어나는 경우는 없었다. 장력을 측정한 결과 최대 100kgf까지 가능하였다.

다. 자료처리장치

자료처리장치는 자료측정, 저장, 전송, 측정시간설정, 센서교정 등을 할 수 있게 DS5002FP CPU를 사용하여 제작하였다. 4개 센서입력 채널을 확보하였으며, 노트북과 모뎀 통신을 위하여 통신포트를 설치하였다. 특히 장기 관측망에서 가장 문제시 되는 전원을 해결하기 위해서 전원절약의 한 방법인 Sleep mode를 적용하였다. 측정시에는 자료처리장치에 자동적으로 전원을 투입되어 측정하고 대기시간은 Sleep mode로 자동전환되게 하였다. 이와 같은 방식을 적용한 결과 측정시에는 125mA가 소모되었지만, sleep mode에는 0.05mA만 소모되어 밧데리 교체없이 장기간 측정이 가능케 하였다.

라. 지하수 자동관측기 유형별 제작

지하수 자동관측기 센서의 유형조사결과 일체형과 분리형의 장단점을 가지고 있기 때문에 사용목적, 현장여건 및 수질 등을 고려하여 선택할 수 있게 센서를 제작하였다. 하나의 센서케이블을 이용하여 두가지 이상의 센서를 이용할 수 있도록 하였다. 즉 수위센서, 전기전도도센서, 온도센서를 동시에 사용하거나, 개별적으로 사용할 수 있게 하였다.

또한 지하수 관측기의 저렴화 방안으로 별도로 되어 있는 센서를 하나의 프로브에

장착한 40㎜×250㎜ 크기의 일체형을 제작하였다. 비록 외형이 다소 크더라도 센서를 동시 또는 개별적으로 사용이 가능하며, 한 프로브를 사용하기 때문에 제작단가를 낮출 수 있었다.

바. 운영프로그램

자료처리장치의 운영프로그램은 ANSI C로 작성하였으며, 컴퓨터용 운영프로그램은 JAVA 언어로 작성하였다. 컴퓨터에서도 측정시간설정, 측정자료보기, 자료다운받기 등을 할 수 있게 하였다.

2. 활용방안

지하수 자동관측기를 산업체와 협력하여 제품화시키며, 지하수 및 지표수 수위 및 수질측정분야에 보급하고자 한다.

특허출원 (2건)

- 지하수 수위·수질 자동관측기 (출원일:2001. 1. 3)
- 압력식 수위감지센서 (출원일:2001. 12. 13)

SUMMARY

As the dependance on ground water as water resource increases by rapid increase of living and industrial water from the change of industrial structure and development of economy and by the distrust of water system, ground water becomes to be recognized widely as national water resource. The problem on contamination of ground water has become greater. Surface water is confirmed the change of water quantity and quality. But it is hard to observe contamination of ground water and its recovery. For general conservation and management of ground water resource, measurement of ground water should be carried out. The network of ground water measuring system structured within the country was installed with foreign instruments. it need a lot of cost. Some foreign instruments favor to forbid user's calibration by tapping the sensor permanently. There are also problems comes from various software and complicated manuals. It is difficult to operate, repair and correct these instruments. For these reasons, there are many omission of measuring data, and it is hard for operator to work the network of ground water measuring system for maintenance.

The purpose of research is to solve existing problems from imported measuring systems by developing new own ground water monitoring system which is easy to use and maintenance, and cost-effective system.

The developed ground water measuring system was consisted of sensor(level, EC, temperature), cable, data logger and operating program. Level sensor was made of pressure sensor. The measurable range of the sensor was 0.0~100.0m. The accuracy of sensor was ±0.1%FS. EC sensor was applied four electrodes. The

measurable range of EC sensor was 0.0~50,000 μS/cm. The accuracy of sensor was ±0.5%. Temperature sensor was made of Pt100Ω. The measurable range of the sensor was -20.0~50.℃. The accuracy of sensor was ±0.1%. Output signal of these sensor was 4~20mA for long distance transmission. Filter circuit etcs were installed in sensor circuit board for noise elimination which was generated by water motor. The sensor cable was manufactured with six lines, one steel wire and one tube to recompense a pressure change. This cable had over 100kgf tension.

Data logger to measure and control this system was designed and fabricated DS5002FP. For easy operation and maintenance, hardware was made of replaceable three PCB(display, keypad, main-board) and assembled with cable connectors. LCD(20×2) was implemented to read the measured variables including level, EC, temperature and time with set-values of control. The main board consisted of micro-controller, 4 analog channel, timer and communication module. Specially this data logger consumed power 0.05mA during sleep mode, 50mA during active mode. Because of low consumption power, the ground water measuring system was operated long-term with 12V 1.0Ah battery

Operation program was developed to perform stand alone by C language. Monitoring program was developed by JAVA language. The program had constructed on memory chip in data logger to be branched into several subprogram which could apply to data view, data download and set variables.

Newly the developed measuring system should be applied to network of ground water measuring system and related part.

CONTENTS

목 차

표 목 차

그 림 목 차

제 1 장 서 론

제 1 절 연구개발의 필요성

지구상에 존재하는 지하수의 양(10,000천㎢)은 지표수(1,000천㎢)보다 10배정도 많으나, 실제 활용되는 양은 지표수가 지하수보다 5~10배정도 많으며, 국내의 경우 지표수 이용량은 275억㎥/년, 지하수 이용량은 37억㎥/년('98년 기준)이다. 부존량과 이용량이 서로 역의 관계를 보이는 것은 지하수보다 지표수의 확보와 이용이 상대적으로 용이하기 때문이다.

그러나 인구 증가, 산업의 발전 등과 맞물려 대기와 함께 수자원도 급격히 오염이 진행되면서, 마시는 물을 중심으로 청정 수자원인 지하수의 수요가 급증하고 있으며, 이와 관련되어 지하수 부존량의 감소와 수질의 악화가 새로운 문제로 대두되고 있다.

지표수는 수량과 수질의 변화가 비교적 쉽게 확인되지만, 지하수는 육안 관찰이 불가능하고 지하수를 포함하고 있는 대수층이 파괴되거나 수질이 악화되었을 경우 이의 원상 회복에는 많은 비용과 긴 시간을 필요로 한다. 따라서 지하수의 안정적인 확보와 관리를 위해서 지하수의 수량과 수질을 장기적·주기적으로 측정하여 이 자료를 바탕으로 관리가 이루어져야 한다. 이를 위한 방법으로 전국에 걸쳐 지하수 자동관측기가 설치/운영되고 있다.

지하수 관측망을 설치하는 공통적인 목적은 지하수 부존량의 변화와 수질오염 진행상태 등을 파악함으로써 지하수자원의 효율적인 이용, 관리와 지하수 개발 및 보전관리계획 수립에 활용 가능토록 하는 것이다. 지하수 관측망으로부터 측정된 자료를 바탕으로 지하수위, 배경수질, 오염원의 분류 기준자료, 지역을 대표하는 지하수 원수의 수질기준 적부여부 판단을 위한 자료, 오염진행 및 처리효과 등 여러 가지 자료를 얻을 수 있다.

지하수 관측망으로 국내에 설치되어 있거나 사용중인 지하수 관측기는 제작사별 고

유 모델로 개발되어 있어 다른 제품과는 운영체계가 상이하여 호환이 거의 불가능하고, 한가지 장비에 익숙하더라도 다른 장비의 조작이나 점검·정비가 매우 어렵게 제작되어 있다. 따라서 관측망 운영자의 점검·보정으로 해결할 수 없는 고장이나 부품의 교체 등이 필요한 경우 제작사의 A/S가 필수적인데, 이에 소요되는 시간과 비용은 효율적인 관측망 운영을 어렵게 하고 있다. 정부에서 발간되는 지하수 관측자료집에는 누락된 자료가 상당히 있다. 한번 고장으로 1~2개월, 경우에 따라서는 6개월 이상이 소요되기도 하며, 소요경비 또한 구입가격의 1/3 이상~2/3까지 되는 경우도 있다. 수리기간동안 자료 취득이 불가능하고 소요경비 또한 만만치 않지만 장비가 고가이기 때문에 예비 장비를 확보하는 것도 쉽지 않다.

지하수 관정은 댐시설과 달리 전국적으로 많이 산재되어 있어 지하수 보존을 위하여 이들 시설에 대한 감시와 관리가 필요하다. 이들을 감시 및 관리하기 위하여 현장조사 또는 지하수 관측기가 설치되어야 하지만 국내에 보급되어 있는 지하수 관측기는 대부분 외국산이므로 가격이 비싸며, 국내시설에 잘 적합하지 않은 경우가 많다.

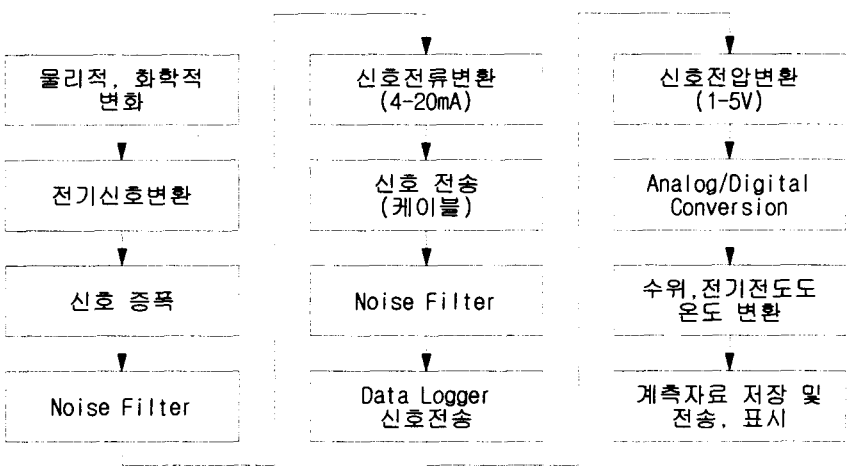
이러한 점을 감안할 때 실시간으로 사후관리가 가능할 뿐만 아니라 지하수 관측망으로 사용이 가능한 관측기의 개발이 필요하다. 지하수 관측기를 개발할 경우 수입대체효과를 얻을 수 있으며 국내 지하수 관측기의 기술향상을 가져올 것으로 판단된다.

제 2 절 연구개발의 목표 및 내용

본 연구에서는 지하수 관측망으로 국내 보급되어 사용 중이거나 지표수 수위 측정용으로 사용중인 센서들의 측정원리와 장단점을 분석하고 그 결과를 토대로 가장 적합한 센서를 선택한다. 수위측정방법 중 플로트식은 비록 가격이 저렴하지만 바람이나 외부의 환경에 민감하므로 별도의 설치공간이 필요하며 기포식은 고압질소 가스통을 설치해야 할 뿐만 아니라 일정한 압력으로 가스가 나올 수 있도록 regulator를 부착되어야 하는 불편이 있다. 압력센서를 응용한 수위센서의 경우 지하수 관정에 센서만을 설치하면 측정을 위한 모든 작업이 완료되기 때문에 현장에서 사용하기에 가장

편리하다. 그러나 기존의 압력식 수위센서의 단점은 센서에서 자료처리장치까지 연결하는 케이블 가격이 다른 부품보다 비싸다는 점이다. 이는 케이블 내부에는 센서의 내부압력과 외부 대기압변화에 대한 보정을 하기 위하여 튜브가 있어 케이블 일부가 손상되었을 경우 전체를 교체해야 하기 때문이다. 지하수 관정 케이싱 상단부는 용접으로 절단되었기 때문에 케이블 손상의 원인이 될 수 있어 손상이 되었을 경우 비싼 가격으로 교체해야 하는 단점을 가지고 있다.

본 연구에서는 압력식 수위센서의 장점을 최대한 이용하면서 케이블에 대한 단점을 보완한 저가 보급이 가능하도록 수위센서를 제작하는 것을 목표로 한다. 기본원리가 <그림 1-1>과 같은 측정방식에서 압력식 수위센서는 기존 개발 제품중 현장에서 사용하기에 편리한 것으로서 국내에서 조달이 가능한 부품을 사용하며, 모터 등의 전기적인 영향에 의한 노이즈 제거방법 등을 회로내에 부가적으로 추가한다.



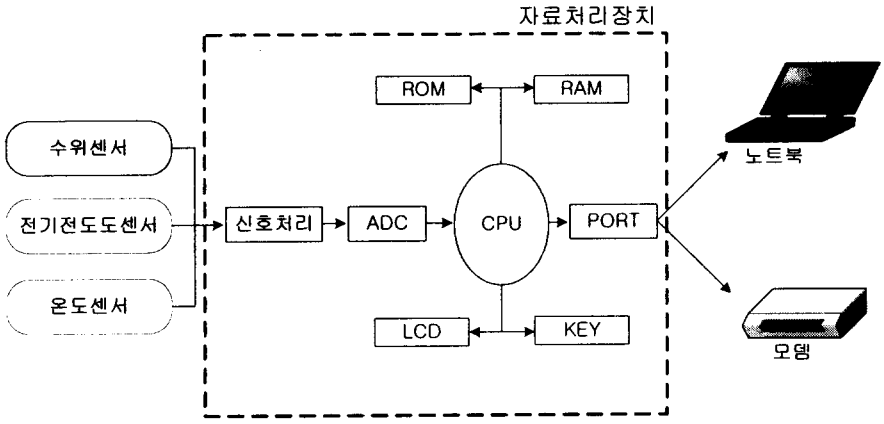
<그림 1-1> 본 연구에서 개발할 센서의 신호처리단계

지하수 관측항목에는 수위뿐만 아니라 수질판단을 위한 전기전도도와 온도를 포함하고 있다. 전기전도도는 지하수 중의 이온량에 의해 결정되는 것으로 이는 용해되어

있는 이온의 종류와 양에 따라 다르게 나타난다. 일반적으로 전기전도도가 증가하는 것은 오염물질의 유입으로 인하여 지하수 내 용존 이온량이 증가하거나 수온이 증가하는 등 여러 가지 원인이 복합되어 있다. 수온은 지하수 내 용존 성분의 농도에 결정적인 영향을 미치며 대체로 대수층내 오염물질이 유입되는 경우 수온은 증가한다.

따라서 지하수 관측망의 측정항목은 지하수위, 전기전도도, 온도이다. 본 연구계획서에서는 지하수위를 측정할 수 있는 시스템 개발을 최종 목적으로 하였지만 지하수 수위만 측정하는 관측망은 없기 때문에 수위센서만을 개발하는 것이 다소 무의미할 수 있다. 이러한 점을 감안하여 본 연구에서는 수위센서뿐만 아니라 전기전도도센서, 온도센서를 동시에 개발하였다.

본 연구에서 개발할 지하수 자동관측기의 기본 구조는 <그림 1-2>와 같이 수위센서, 전기전도도센서, 온도센서, 센서와 자료처리장치를 연결하는 케이블 그리고 관측기를 제어하는 자료처리장치로 구성되어 있다. 자료처리장치는 신호처리, ADC, LCD 그리고 노트북에 연결할 수 있는 통신포트로 되어 있다. 센서 외관은 1인치 관측공에 사용할 수 있도록 크기를 정하며 현장에서 직접 측정항목을 확인하거나 자료처리장치에 저장할 수 있도록 한다.



<그림 1-2> 지하수 자동관측기 기본구조

본 연구는 지하수 전문가들이 제시한 현장 사용시 문제점, 개선사항 등을 반영하여

현장 사용자 중심으로 지하수 관측기의 하드웨어와 프로그램을 개발하며 기존 지하수 관측기와 비교하여 기능과 성능면에서 대등할 정도로 개선하여 가격 경쟁력을 가진 지하수 자동관측기를 개발하는 것을 목표로 한다.

연구목표	연구내용 및 범위
지하수 자동관측기의 장단점 분석과 기준 확립	· 국내의 지하수 관측망 설치현황조사 · 사용중인 지하수 관측기 운영조사 (CR-10,Tuber, CP-300, W/S 3100) · 조사항목 : 본체, 센서의 성능, 통신방법 및 모뎀, 운영프로그램, A/S 체제, 전원장치
지하수 자동관측기의 수위, 전기전도도, 온도센서 제작	· 일체형과 분리형 지하수 자동관측기 제작 · 수위센서 : 0~100m · 저농도 전기전도도센서 : 0~5,000 μS/cm · 고농도 전기전도도센서 : 0~50,000 μS/cm · 온도센서 : -20~50℃ · 모터에 의한 노이즈 제거회로 추가 · 4~20mA 전류형 센서신호 출력 · 최소구경에 사용가능토록 회로설계 분리형 : Φ20mm, 일체형 : Φ40mm
지하수 자동관측기 자료처리장치 제작	· 현장설치가 가능한 소형 자료처리장치 제작 · 12V, 1.9Ah 소형밧데리 200일동안 충전없이 사용할 수 있도록 초절전형 제작 · 5,000회 자료 저장
시작품 현장시험	· 지하수 관측시설에 설치 실증시험 · 모터에 대한 노이즈 영향 · 외부충전장치 없이 12V, 1.9Ah밧데리로 장기간사용

제 2 장 국내 지하수 관측망 설치 및 운영현황

제 1 절 국내 지하수 관측망 설치 현황

국내에는 여러 종류의 지하수 관측망이 각각 다른 목적에 따라 전국적으로 설치 운영되고 있다. 현재 기존에 설치되어있는 관측망의 종류로는 건설교통부에서 주관하는 "광역 지하수관측망", 서울시 지하수 관측망(서울시 지하수 보전관리 목적), 제주도(해수침투 방지 목적) 등 행정구역 단위의 지방자치단체에서 관리하는 "보조 지하수 관측망", 먹는샘물의 보전관리나 해수침투를 방지할 목적으로 설치된 지하수관측망 등이 있다.

지하수 개발과 보존이 중요한 과제인 만큼 지하수에 대한 감시와 관리가 필수적이다. 따라서 앞으로 지하수 관측기의 수요는 증가할 것으로 본다.

(표 2-1) 서울시 지하수관측망 설치내역 (1999말 현재, 서울시)

구청	'97	'98	'99	구청	'97	'98	'99	구청	'97	'98	'99
종로	2	2	-	도봉	2	1	2	영등포	1	3	-
중	1	1	-	노원	1	4	-	동작	2	6	-
용산	1	3	-	서대문	1	2	1	관악	2	2	-
성동	1	1	1	은평	2	3	-	서초	3	2	-
광진	1	6	-	마포	3	4	-	강남	2	-	2
동대문	2	2	-	양천	3	4	-	송파	-	5	-
중랑	1	-	-	강서	2	6	-	강동	1	4	-
성북	-	1	1	구로	2	5	-	계	40	70	8
강북	2	2	-	금천	2	1	1				

(표 2-2) 광역 지하수관측망 설치현황('98년 말 현재, 건설교통부)

시도별	설치시군	관측정 설치(개소)		유역별	관측정 설치(개소)	
		계 획	실 적		계 획	실 적
계	시 60개, 군 83개	310	97	계	310	97
서울시	-	1	-	한 강	62	17
부산시	-	1	1			
대구시	-	4	2	낙동강	63	22
인천시	-	1	-			
광주시	-	2	1	금 강	40	14
대전시	-	2	1			
울산시	-	4	3	섬진강	20	6
강원도	7시 11군	41	8			
경기도	16시 9군	43	12	영산강	13	7
경 남	7시	36	11			
경 북	8시 11군	46	16	서해안	63	16
전 남	4시 15군	36	11			
전 북	6시 8군	34	9	남해안	21	5
충 남	6시 9군	35	12			
충 북	3시 8군	24	10	동해안	28	10
제주도	-	-	-			

(표 2-3) 제주도 해수 침투감시용 관측망 설치 (1999년 말 현재, 제주도)

구 분	계	제주시	서귀포시	북제주군	남제주군
계	60	8	11	23	18
자동관측	36	5	5	13	13
수동관측	24	3	6	10	5

(표 2-4)해수침투 관측망 설치내역 (1999말 현재, 농업기반공사)

구 분	지구명	위치		설치 년도	관정심도 (m)	기기설치심도 (m)
		시군	읍면			
경 기	숭뇌1	강화	송해	'98	80	65
	숭뇌2			'98	94	15
	숭뇌3			'99	137	30
전 남	감정1	신안	지도	'98	127	75
	효지1			'98	43	39
	효지2			'99	67	50
	나리1	진도	군내	'98	150	80
	신기1			'99	80	76
	신기2			'99	101	51
	화흥1	완도	완도	'98	35	30
	화흥2			'98	64	55
	정도1			'99	80	18
경 남	갈화1	남해	고현	'98	80	15
	갈화2			'99	61	15
	덕호1	거제	사동	'98	80	15
	시방1	거제	장목	'98	80	30
	시방2			'99	200	30

제 3 장 지하수 자동관측기의 개요

제 1 절 지하수 자동관측기 비교분석

지하수 관측망은 지하수자원의 수량과 수질을 장기간에 걸쳐 주기적으로 관측할 수 있도록 관정 내부에 수위와 수질을 측정할 수 있는 센서와 이를 제어할 수 있는 자료처리장치로 구성된 계측시스템이다. 지하수 관정 사용목적(농업용수, 산업용수, 생활용수 및 비상용수)과 지역적 특성(육지부, 해수부, 내륙지역, 섬지역)에 따라 여러 종류의 지하수 관측시설이 전국에 걸쳐 설치, 운영되고 있다.

이와 같이 여러 종류의 지하수 관측기가 설치되어 있지만 공통적인 목적은, 지하수 보존량의 변화와 수질오염 진행상태 등을 파악함으로써 지하수자원의 효율적인 이용과 관리를 위한 지하수개발 및 보전관리계획 수립에 활용 가능토록 하는 것이다. 지하수 관측기로부터 지하수위와 배경수질, 오염원의 분류 기준자료, 현지를 대표하는 지하수 원수의 수질기준 적부여부 판단을 위한 자료, 오염진행 및 처리효과 등의 평가자료를 얻을 수 있다.

보다 정확한 관측자료를 수집하기 위하여 관정 내에 정확도가 높은 관측기기를 설치하는 것은 매우 중요하다. 그러나, 수질과 관련된 관측기기는 수질성분 분석을 위한 것이 아니라 수질변화 등을 측정하여 전체적인 추세를 파악하려는 것이 목적이므로 분석장비에서 요구되는 수준의 높은 정확도를 필요로 하지는 않는다.

1. 지하수 자동관측기의 측정항목

국내에 설치되어있는 자동관측기는 통상 4개 항목을 측정할 수 있도록 설계되어 있으며, 그 항목은 수위, 수온, EC(전기전도도), pH(수소이온농도)이다. 그러나 pH는 안정된 자료취득이 어렵고 보정을 실시해도 그 값이 오차의 범위를 벗어나는 경우가 많아 근래에는 측정 항목에서 제외시키는 경우가 많아졌다. 특히 건설교통부에서 운용 중인 광역지하수관측망은 설치단계부터 pH 항목을 제외하였으나, 먹는 샘물의 관

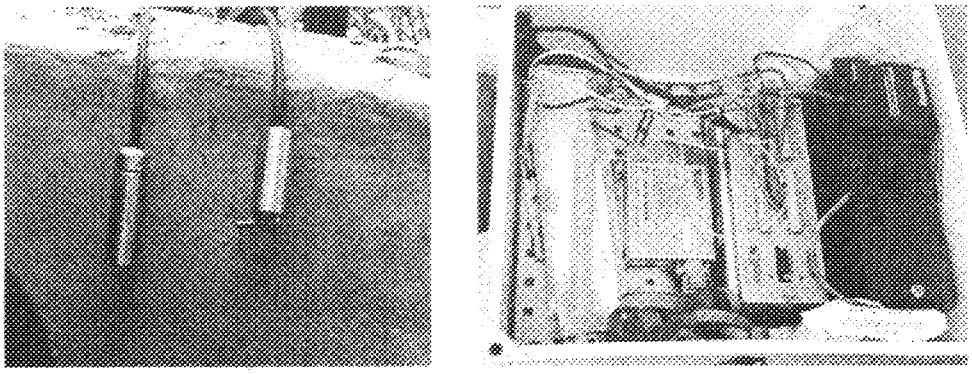
측과 감시를 위한 관측망에서는 pH를 포함토록 먹는물관리법에 규정되어 있어 의무적으로 설치하고 있다.

본 연구에서 개발하고자 하는 지하수 자동관측기에서는 pH를 제외한 수위, 전기전도도 그리고 온도 3항목으로 하였다. 전기전도도는 온도에 따라 민감한 영향을 받기 때문에 온도보상이 필요하므로 전기전도도 센서 내부에 온도보상회로를 첨가하였다. 따라서 온도는 전기전도도에 포함시켜 개발하였다.

2. 지하수 자동관측기의 기종 비교

가. CR-10

수위센서와 전기전도도 센서가 별도로 설치되어 있는 분리형으로 수리나 교정할 경우 간단하게 처리할 수 있도록 구성되어 있다. 특히 자료처리장치는 현장에 적합하게 설계되어 있어 급격한 온도 또는 습도변화에 대하여 방수처리가 충분히 되어 있어 잔고장(누전, 운영중지 등)이 적다. 따라서 다른 관측기에 비하여 자료처리장치의 신뢰도가 높았다. 그러나 수위센서의 수명이 비교적 짧아 자주 교체해야 하며, 수리 및 교환에 대비하여 별도의 수위센서를 확보해야하므로 유지관리비가 많이 드는 편이다.



(센서부) (자료처리장치)

<그림 3-1> CR-10 지하수 관측기

대한민국국회도서관장서

자료처리장치에는 통신코드와 일치하는 전용 모뎀이 장착되어 있어 낙뢰와 같은 외부의 전기적인 충격에 의하여 손상이 되었을 경우 고가의 모뎀을 구입해야 하는 단점이 있다. 시중에 시판되는 일반 외장형 모뎀은 십만원 내외로 구입이 가능하지만 CR-10 관측장비는 백만원 정도인 고가의 전용모뎀을 설치해야 한다. 그러나 전용모뎀은 자료처리장치에서 전원제어를 함으로써 대기상태에서는 전원을 사용하지 않고 있다. 따라서 별도의 솔라셀이나 고용량의 밧데리를 요구하지 않는다. 자료처리장치와 모뎀의 호환성이 높은 관계로 사무실에서 자료전송방법이 간단하다는 장점이 있다. 현장에서 본체와 노트북 연결시 RS232C 통신포트를 이용하지만 전용케이블을 사용해야 하는 단점이 있다.

모뎀과 자료처리장치의 소비전력이 적기 때문에 외부 충전장치 또는 솔라셀과 같은 전원장치가 필요 없다. 현재 설치된 전원은 DC 12V, 1.9Ah 내부 밧데리를 사용하고 있다. 충전없이 3개월이상 사용이 가능하여 2회/년 밧데리 교환으로 충분히 운영될 수 있다.

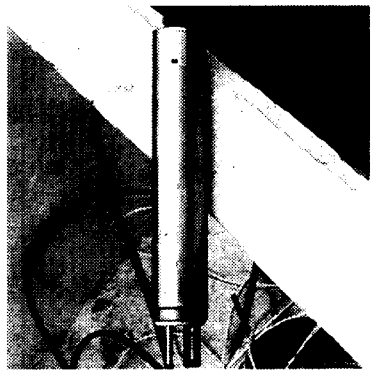
노트북 자료송수신용 전용프로그램은 윈도우에서는 실행되지 않으며, DOS모드에서만 실행이 된다. 바이너리방식으로 자료가 저장되기 때문에 엑셀과 같은 자료 통합처리프로그램에 직접 적용하기 위해서는 자료송신 후 아스키 방식으로 변환해야 하므로 자료처리 및 분석이 불편하다. 현재 간이용으로 통합관리용 프로그램이 개발되어 있지만 아직 일반화되어 있지 않다.

나. Tuber

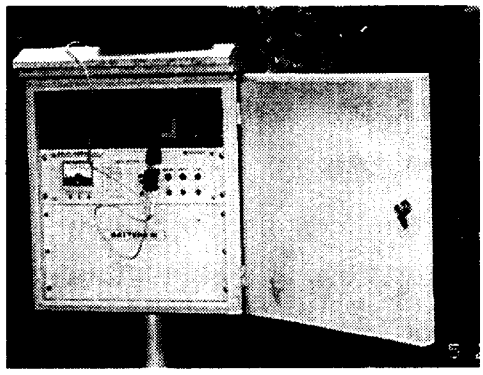
수위센서와 전기전도도센서와 자료처리장치가 함께 구성되어 있는 일체형으로 현장에서 간단하게 설치가 가능하고 교정시에도 다른 장비에 비하여 간단하다. 그러나 일체형의 가장 큰 단점으로는 센서교체가 어렵고 특히 자료처리장치가 고장일 경우 전체를 교체해야 한다는 것이다.

또한 자료처리장치와 노트북의 통신 방식이 일반적인 RS232C 방식이 아니라 장거리용인 RS485 방식으로 노트북과 연결시 Data Transfer라는 별도의 장치가 있어야 한다. 출력방식이 RS485방식이므로 직접 모뎀을 연결할 수 없으므로 현재 센서 공급업체에서 별도의 자료처리장치를 부착하여 공급하고 있다. 따라서 이중으로 가격부담이 되고 있는 실정이다.

AC전원을 이용하여 밧데리를 충전할 경우 센서 출력값이 불안전한 값을 출력하므로 관정 주위에 AC전원이 있더라도 솔라셀을 사용해야 한다. Tuber 관측장비의 가장 큰 특징은 현장에서 쉽게 교정이 가능하며 자료취득시간 설정과 자료처리장치의 제어변수를 노트북을 이용하여 쉽게 접근할 수 있다는 것이다.



(센서부)



(자료처리장치)

<그림 3-2> Tuber 지하수 관측기

자료처리장치에 통신코드가 공개되어 있어 CR-10과는 달리 일반 모뎀 장착이 가능하다. 그러나 일반모뎀에 외부전원을 별도로 공급하여야 한다. 모뎀의 소비전력(LifeStyle 28.8 : 220V에 4.5W)이 자료처리장치에 비하여 크기 때문에 모뎀 동작을 위해서는 반드시 솔라셀이나 고용량의 밧데리가 필요하다.

자료처리장치와 센서의 구동전류는 5mA이하로 12V, 1.2Ah 내부밧데리로 1년 이상 사용이 가능하다. 내부 밧데리는 용량이 작기 때문에 모뎀을 구동하기 위해서는 1개

월마다 대용량 밧데리(12V, 40Ah)를 교체해야 한다. 현재 설치되어 있는 장비는 솔라셀 또는 AC전원이 설치되어 있다. 장기관측망으로 사용하더라도 관측망 운영관리자가 4회/년 횟수로 현장을 방문하여 노트북으로 자료를 송수신할 경우 솔라셀과 같은 충전장치는 필요하지 않을 수 있다.

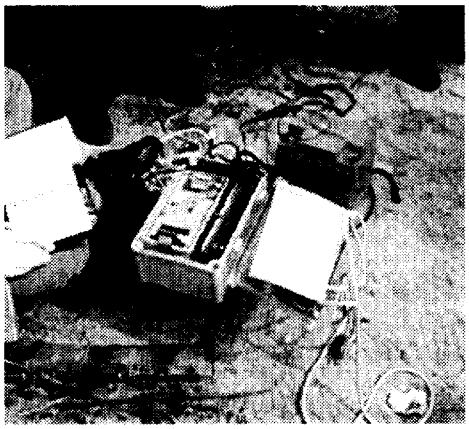
노트북 자료송수신용 전용프로그램은 윈도우환경에서 실행되지 않으며, 영문DOS모드에서만 실행이 된다. 한글모드에서는 메뉴를 읽을 수 없어 일반인(관리자)이 사용하기에 다소 불편하다. 그리고 자료 수신시 아스키 방식으로 자료가 저장되기 때문에 엑셀과 같은 자료 통합처리프로그램에 직접 읽을 수 있어 자료처리 및 분석이 비교적 간단하다. 현재 자료처리 및 전송용으로 별도의 자료처리장치를 설치할 경우 윈도우 환경내에서 작업이 가능토록 프로그램이 개발되어 있다.

다. CP-300

수위센서와 전기전도도 센서와 자료처리장치가 함께 구성되어 있는 일체형으로 현장에서 간단하게 설치가 가능하고 교정방법도 간단하다. 그러나 일체형의 가장 큰 단점으로 센서교체가 어렵고 특히 자료처리장치가 고장일 경우 전체를 교체해야 한다.

또한 자료처리장치와 노트북의 통신 방식이 일반적인 RS232C 방식으로 별도의 장치가 필요하지는 않다. RS232C 방식에서 TXD, RXD, GND 3라인만 사용하고, DCD와 같은 control 라인을 사용하지 않아 본체와 노트북 연결시 접속이 불안전하다.

(센서부)



(자료처리장치)

〈그림 3-3〉 Green Span 지하수 관측기

센서 출력값은 다른 관측장비에 비하여 안정적이나 밧데리 용량 부족시 자료측정이 불가능하며, 특히 자료처리장치에 저장된 자료가 손실가능성이 있어 밧데리 출력전압이 11.0V 이하가 되지 않도록 해야 하는 단점이 있다.

자료처리장치에 통신코드가 공개되어 있어 Tuber 관측장비와 같이 일반 모뎀 장착이 가능하다. 그러나 일반모뎀용 외부 전원을 별도로 공급하여야 한다. 모뎀의 소비전력(LifeStyle 28.8 : 220V에 4.5W)이 자료처리장치에 비하여 크기 때문에 모뎀의 동작을 위해서는 반드시 솔라셀이나 고용량의 밧데리가 필요하다. 또한 사무실내 컴퓨터에서 자료취득을 위해서는 현장에 설치된 모뎀과 동일한 모뎀을 사용해야 한다.

자료처리장치와 센서의 구동전류가 적기 때문에 내부밧데리로 1년 이상 사용이 가능하다. Tuber 장비와 마찬가지로 내부 밧데리는 용량이 작기 때문에 모뎀을 구동하기 위해서는 1개월마다 밧데리(12V, 40Ah)를 교체해야 한다. 현재 설치되어 있는 장비에는 모뎀이 없어 솔라셀 또는 AC전원이 설치되어 있지 않다. 장기관측망으로 사용할 경우 1개월마다 밧데리를 교체하거나 솔라셀과 같은 충전장치를 설치해야 한다.

노트북 자료송수신용 전용프로그램은 윈도우환경에서 실행되지 않으며, 영문DOS모

드에서만 실행이 된다. 한글모드에서는 메뉴를 읽을 수 없어 일반인(관리자)이 사용하기에 다소 불편하다. 현재 자료처리 및 전송용으로 별도로 Data Logger를 설치할 경우 윈도우 환경내에서 작업이 가능토록 프로그램이 개발되어 있다.

라. W/S 3100

수위센서와 전기전도도 센서가 별도로 설치되어 있는 분리형으로 수리나 교정할 경우 간단하게 처리할 수 있도록 구성되어 있다. 그러나 자료처리장치는 CR-10보다 다소 불안전하여 실행중단이 종종 발생하였다. 본체 생산이 단종되어 부품교환비가 장비구입가격의 80%정도가 되어 부품을 교체할 수 없어 새로운 장비를 구입해야 하는 실정이다.

자료처리장치에 통신코드가 공개되지 않아 전용모뎀을 사용해야 한다. 그러나 기존 장비에 모뎀이 장착되어 있지 않아 원거리 통신이 불가능하다.

자료처리장치의 소비전력이 아주 적기 때문에 DC 12V, 1.9Ah 내부 밧데리로 5년동안 사용이 가능하다. 그리고 다른 장비에 대비하여 정확도와 정밀도가 높으며, 고장율도 적어 안정적으로 동작이 되지만 상위 기종이 생산되면서 하위 기종 생산이 중단되어 교체나 수리가 어렵게 되어 있다.

노트북 자료송수신용 전용프로그램은 윈도우환경에서 실행되어 사용이 편리하며 자료 통합처리 및 그래프 처리가 잘 되어 있다.

3. 지하수 자동관측기의 기기조건

본 연구의 시작품에서는 수위, 전기전도도, 온도 3개 항목을 측정하는 것으로 하였다. 기존 지하수 관측기의 조사결과와 지하수 전문가들의 현장경험을 바탕으로 지하수 자동관측기가 갖추어야 할 사항은 다음과 같다.

가. 센서

1) 작동온도 범위

지하수 관측에 사용되는 센서의 경우 지하수위 하부에 설치하므로 지하수위 항온성(14℃)을 고려할 때 굳이 작동온도 범위가 클 필요는 없다. 그러나 지하수위의 강하로 인하여 센서가 지하수 상부에 드러나는 경우에는 대기 온도에 영향을 받으므로 여름철 기온상승과 겨울철 기온하강에 대한 고려가 되어야 한다.

국내 도입된 센서의 작동범위는 최저 -40℃에서 최고 100℃까지 다양하다. 지하수 측정환경을 고려하여 볼 때 센서의 작동범위는 -10~50℃ 범위가 적당하다.

2) 측정과 오차범위

지하수위, 수온, 전기전도도를 측정하는 센서는 작동조건에 의한 오차범위를 가진다. 정확한 지하수위 관측을 위해서는 센서자체의 정밀도와 정확도가 높아야 한다. 지하수위의 경우 오차범위의 단위는 %FS로 기기별로 ±0.01%FS에서 ±3%까지 다양하다. 예를 들면 지하수위의 측정범위가 100m이고 오차범위가 ±0.02%FS 이라면 ±2cm 만큼의 측정오차를 지니는 것이다. 자연적인 지하수위의 변동은 아주 미미하므로 수위변화 데이터를 취득하기 위해서는 100m 기준으로 할 경우 오차범위가 1cm(±0.5cm) 이내를 만족시키는 것이 좋으나 기존의 수위센서 중 이를 만족시키는 센서는 없다. 그리고 정밀도와 정확도는 가격에 비례하므로 정확한 센서만을 기준으로 할 수 없다. 현장 여건과 지하수 관측망의 운영경험으로 볼 때 측정 범위 100m 기준에서 오차범위를 6cm(±3cm) 미만으로 하였다. 만약 측정 범위를 50m 기준이라면 오차범위는 ±1.5cm 까지 정확하게 할 수 있다. 따라서 지하수 수위변동이 적을 경우 측정범위를 줄여 그 정확도를 높게 할 수 있다.

(표 3-1) 지하수 자동관측기별 비교

장비 및 회사별		CR-10	Tuber
본체	전원장치 밧데리 종류 및 교환시기 기억용량 데이터 저장능력	DC 12V 일반밧데리, 2개월 128KB 62,280	DC 12V 내장밧데리 64~128KB 36,000~60,000
센서	오차범위 수위센서(%) 온도센서(℃) EC센서(%)	 ±0.01 ±0.1 ±5.0	 ±0.1 ±0.2 ±2.0
모뎀	통신속도 모뎀 전원장치	9,600 불필요	9,600 외부전원(DC12V)
프로그램	프로그램 운영체계 자료취득소요시간 자료통합관리여부	영문 DOS 1~2분 관정별 분리관리	한글 윈도우 1~2분 통합관리
교정주기	EC, 온도센서	1개월(기본)	2~3개월
	pH 센서	1개월(수명 1년)	1~2개월 (수명 1년)
센서교환		센서종류별 교환 가능	1개센서 교체시 본체를 제작사에 보내야 함

< 계 속 >

장비 및 회사별		CP-300	W/S 3100
본체	전원장치 밧데리 종류 및 교환시기 기억용량 데이터 저장능력	DC 8~27V 일반밧데리, 1개월 64KB(확장가능) 32,000	DC 9~24V 내장밧데리 1,024KB 520,000
센서	오차범위 수위센서(%) 온도센서(℃) EC센서(%)	 ±0.2 ±0.1 ±0.2	 ±0.3 ±0.05 ±2.0
모뎀	통신속도 모뎀 전원장치	모뎀없음	모뎀없음
프로그램	프로그램 운영체계 자료취득소요시간 자료통합관리여부	한글 윈도우 10~12분 통합관리	영문 윈도우 1~2분 지구별 분리
교정주기	EC, 온도센서	2~3개월(기본)	2~3개월
	pH 센서	1개월(수명 1년)	-
센서교환		1개센서 교체시 본체를 제작사에 보내야 함	센서종류별 교환

지하수온의 경우 측정단위는 ℃이며 오차범위도 대부분의 센서가 ±0.1℃이다. 따라서 온도센서의 오차범위는 ±0.1℃이하가 되어야 한다.

전기전도도의 경우 측정되는 단위는 μS/cm를 이용하여 센서의 오차범위는 지하수위 센서와 동일하게 %FS를 이용한다. 측정범위가 0~10,000μS/cm이고 오차범위가 ±0.2%FS라면 실제 ±20μS/cm의 전기전도도 측정오차를 지니는 것이다. 기존 전기전도도 센서의 오차범위는 작게는 ±0.1%FS에서 ±3%FS까지 그 범위가 매우 크다. 본 연구에서는 지하수뿐만 아니라 일반 지표수 수질을 측정하는 것을 고려하여 저농도는 0~5,000μS/cm, 고농도는 0~50,000μS/cm 측정범위로 하는 것이 적당하다.

센서 사양의 결정시 중요시 되는 부분이 오차범위, 측정범위, 정밀도와 정확도이다. 물론 측정범위가 적을수록 정확한 측정이 이루어 질 수 있으나 우선적으로 지하수 사용용도(양수시험용, 장기관측망용, 시험용, 해수침투용)를 결정한 후 그 용도에 적합한 측정범위와 오차범위를 결정해야 한다.

3) 센서교정주기

모든 측정센서는 일정 기간이 경과하면 센서에 대한 보정을 필요로 한다. 현장에 설치된 수위, 수온, 전기전도도의 교정주기를 센서 공급자들은 대체로 1회/3개월 정도 실시를 요구하지만, 지하수 관측망을 운영한 결과 2회/년 가 적당하며, 현장에서 바로 실시할 수 있도록 해야 한다.

지하수 관측기는 정밀기기에 해당되므로 주위 환경에 영향을 받는다. 따라서 가능한 한 자주 센서를 교정 및 점검해 주는 것이 좋다. 그러나 현장이 분산되어 설치되어 있거나, 원거리에 설치되어 있을 경우를 대비하여 보통 지하수 조건에서 2회/년 정도 교정 및 점검하는 것이 적당하다. 또한 1차적으로 소프트웨어적으로 검교정이 가능케하고 소프트웨어적으로 교정이 되지 않을 경우 하드웨어적으로 교정할 수 있어

야 한다.

4) 센서 크기

지하수 관측정 내에 설치되는 센서의 크기는 관측정의 크기, 관측기기의 설치방법과 설치위치 등에 의해 영향을 받는다.

관측기기의 설치방법에 있어서 유지관리의 효율성을 고려하여 수중모터와 연결된 압상파이프와 보호파이프를 동시 설치, 그 내부에 관측기기를 설치하는 경우가 있고 단지 관측망 전용으로 사용하기 위하여 관측기만 설치하는 경우가 있다. 나공 상태에서는 센서의 크기가 큰 문제가 되지 않는다. 보호파이프를 설치한 경우 센서의 길이와 구경에 대한 고려가 요구된다. 구경이 큰 센서의 경우 관측정 내부 보호파이프에 설치가 어렵고, 센서의 교정이나 유지관리를 위한 센서 인양시 기기의 손상이 발생할 우려가 많아 유지관리에 어려움이 있기 때문이다.

관측정의 구경은 법적으로 명시되어 있지는 않지만 관측기기의 설치와 물 시료의 채취가 가능한 규모로 한다. 단 대수층의 수리특성 파악을 위한 대수성 시험이 필요한 경우에는 관측정의 구경은 최소한 150㎜이상이 되어야 하며, 단순히 지하수만을 관측하기 위한 것이라면 54㎜(N규격)이상이 무방하다. 따라서 관측용 센서의 크기는 관정 구경 54㎜에 사용 가능하도록 제작되어야 한다.

나. 자료처리장치(Data Logger)

자료처리장치는 센서로부터 계측된 자료를 저장, 표시하는 소기능 컴퓨터 부분으로서 센서장치에 내장되어 있는 내장형과 센서 외부에 별도로 내장되어 있는 외부형이 있다.

▷ 내장형 저장장치 : 관측장비 가격을 크게 좌우하는 케이블이 필요없어 단지 일반선에 의하여 매달아 두면 되므로 가격이 절약되나, 측정도중에 자료의 정확도를 점검할 수 없다는 단점과, 센서의 부피를 크게하는 단점이 있어(3~4 inch) 호수, 조석

과 같은 지표수 장기관측에 많이 활용되고 있다.

▷ 외장형 저장장치 : 지하수 장기관측에는 센서의 크기를 최소한으로 줄이기 위하여 측정자료 저장장치를 공밖으로 설치하는 외장형이 많이 활용되고 있으며 자료의 측정도중에 점검과 확인이 가능하여 많이 활용되고 있다.

자료처리장치는 기억장치의 크기와 자료저장 프로그램의 간편성이 중요하다. 내부에서는 모든 동작이나 연산을 디지털 값으로 처리하는데 반하여 센서로부터 측정되는 변수는 아날로그 신호인 경우가 대부분이다. 즉 이들 외부기기의 신호는 온도, 수위 등과 같은 어떤 실제의 양에 비례하면서 연속적으로 변화하는 전압 또는 전류신호로 나타난다. 따라서 단계가 분명하게 구분되어 있는 디지털 신호와 연속적인 값을 갖는 아날로그 신호간의 상호변환을 ADC(analog digital converter)라 하며 data logger과 센서간의 인터페이스에 있어서 매우 중요한 비중을 차지하므로 15bit 이상 분해능을 가지는 ADC를 사용하여 아날로그 신호를 0에서 32,000까지 디지털 값으로 분해해야 한다.

측정하는 값들은 자체의 메모리에 기억되거나, 혹은 같이 설치된 컴퓨터에 수록이 가능하며, 또한 원거리에서 자료를 습득하고자 할 경우 모뎀을 통하여 전국 어느 곳이든지 자료를 받아볼 수 있어야 한다. 그러나 컴퓨터 또는 모뎀을 통하여 자료 전송을 하더라도 그동안 측정된 자료는 자료처리장치내 메모리에 저장되어 있어야 한다. 따라서 자료처리장치는 저장기능을 갖추어야 한다. 물론 비휘발성 메모리(volatile memory)와 리튬전지 같은 백업 밧데리가 내장되어 있어 data logger를 이동하거나 전원이 단절되더라도 자료는 보존할 수 있어야 한다. 이때 백업용 밧데리의 용량은 최소 3년이상 사용이 가능해야 하며 쉽게 교체가 가능할 수 있어야 한다.

그리고 자료처리장치에 포함된 기억용량은 자료를 기억할 수 있는 양으로 4회/일 단위로 측정할 경우 6개월 동안 보존할 수 있는 기억용량은 720개(4회×30일×6개월)이면 충분하다. 따라서 여유 기억용량을 두더라도 1024개(1K)까지 기억할 수 있으면 된다.

장기관측망 저장방법에 있어 FIFO(first input first output)방식이 채택되어야 한다. 기존 출시된 자료처리장치의 일부는 기억용량을 초과될 경우 메모리에 저장되어 있는 자료전체가 없어지거나 더 이상 저장되지 않는다. 장기관측망에서 수시로 자료를 전송하므로 이전 자료는 의미가 없다. 따라서 FIFO방식을 채택하여 최근 자료는 저장되고 가장 이전에 저장된 자료는 삭제되도록 해야 한다.

더욱이 현장에서 자료의 입출력, 전송, LCD 표시, 전원공급 등을 위한 장치가 관정 외부에 장착되어 자료처리장치가 관정 밖에 설치되고 센서만 관정에 설치되는 것이 바람직하다. 그리고 측정자료를 자료처리장치에 저장하고 이로부터의 자료전송은 표준통신 규격인 RS-232C방식으로 하여 RS-232C 포트를 가진 어느 컴퓨터 및 기기와도 통신이 가능하도록 설계되어야 한다.

국내에 도입되어 있는 자료저장장치는 매우 다양한 회사의 제품들이 있으며 지하수 장기관측자료 전용으로 제작된 제품과 기상, 수문자료 등 제반 장기관측자료에 다양하게 적용할 수 있는 다용도용이 있으며 가격에 차이가 있다. 지하수 자동관측기 전용으로 자료처리장치를 설계하여 제작단가를 최대한 낮게 할 수 있다.

다. 모뎀 및 통신방법

관측자료 자료수집 방법에는 수동측정기를 가지고 주기적으로 현장을 방문하여 수집하는 방법이 있고, 자동관측기기가 설치된 경우에는 직접 노트북을 가지고 현장을 방문 축적된 자료를 취득하거나 전화선을 설치하여 모뎀, 무선통신이나 전용회선을 이용하여 수신하는 방법이 있다.

직접 현장을 방문하여 노트북으로 자료를 수집하는 방법은 현장에서 수위 확인 및 각종센서의 보정실시가 가능하여 월 1회 정기점검시 마다 실시하면 되는 장점이 있으나, 중간에 훼손 되거나 잘못될 경우 확인이 불가능하며 갑자기 자료 필요시 수집이 불가능한 단점이 있다.

모뎀 이용시는 중앙 집중식이 가능하며 수시로 자료의 확인이 가능한 반면 모뎀을

통한 장거리 전송불량이 문제가 되는 경우가 많다. 자료처리장치의 프로그램과 PC의 프로그램은 제조회사에서 컴파일된 상태에서 납품되기 때문에 사용자가 직접수정 또는 확인이 불가능하다. 따라서 모뎀을 통한 장거리 전송이 원활치 못할 경우 고장의 원인을 찾아내기가 힘들다. 그리고 관측장비 납품시 자료 송수신 시스템까지 일체형으로 외국 제품이 일괄 납품되어 데이터 송수신 모뎀이 동 관측기에서만 사용 가능한 전용모뎀으로 일반 컴퓨터 모뎀과는 송수신 체계가 달라 국산 컴퓨터와 호환이 원활치 않았다. 이러한 낭비 요소를 사전에 제거하기 위하여 통신방법에 있어서 일반 모뎀과 호환성이 있어야 한다.

따라서 모뎀은 국내 구입이 가능해야할 것이며 모뎀 사용전력도 측정장비와 밧데리 공유형태가 아닌 전화선으로부터 공급받는 형태가 바람직하다. 또한 자료처리장치는 기본적으로 두 개의 RS232C 포트를 내장하고 있어야 한다. 한 포트는 노트북과 접속하여 현장에서 직접 자료를 전송하거나 자료처리장치내 변수를 설정, 삭제할 수 있도록 한다. 다른 한 포트는 모뎀을 장착하여 장거리 전송을 통한 자료의 취득이 하는데 사용한다. 이 경우 network을 구성하여 중앙집중식 관리 시스템 구성이 가능하다.

라. 전원장치

장기관측망용 수위, 수질측정기에서 센서의 성능 못지 않게 중요한 것이 전원이다. 관측기 설치장소가 전원이 없는 현장이 대부분으로 자체 밧데리, 외부 밧데리 또는 Solar Cell을 사용할 수 있도록 하드웨어적으로 설치되어야 한다.

실제 현장 운영시 문제점으로 현재 도입되어 있는 대부분의 센서들이 밧데리의 잔여용량 표기 장치가 없어 잔여 용량으로 얼마만큼의 기간동안 측정 가능한 지를 알 수 없어 교체시기를 정확하게 예측하기가 어렵다는 단점이 있다.

다만 관측장비 설치 시점에서 밧데리가 완전하게 충전된 상태로 가정하고 측정센서별로 사양서에 기록된 1회 측정시 소요 전력과 측정주기를 곱하여 최대 측정 가능기간을 추정하고 기 사용기간에서 감하여 교체시기를 추정할 수 있고(누전이나 방전시

측정이 단절될 우려가 크며 실제 시범지구에서도 이러한 사례가 있었음) 측정도중에는 밧데리 전압을 측정하여 잔여 기간을 추정할 수 있으나 정확하지 못하다는 단점이 있으므로 여유율을 고려하고 밧데리를 교체하는 것이 바람직하다.

언급한대로 자료의 전송모뎀을 설치하는 경우 모뎀 사용 전력 소모량이 관측장비 소모량에 비하여 큰데 이 모뎀 사용전력을 측정장비의 밧데리로부터 공급하는 형과 전화선으로 부터 공급받는 형이 있으므로 구입시 사양검토에 주의하여 전화선으로부터 공급받는 형으로 선정함이 필수적이다.

현재 국내에 보급되는 자료처리장치의 경우 대부분이 내부 밧데리를 내장하고 있으나, 제조회사의 사양서를 보면 밧데리의 수명에 대하여 전혀 언급되어 있지 않다. 이는 센서의 소비전력과 수량, 측정간격 및 자료처리장치의 소비전력 등 여러 가지 변수에 좌우하기 때문에 단순히 가능한 사용기간으로 표현이 불가능하다. 계측기가 작동되지 않을 경우 일차적으로 점검해야 할 사항은 전원이다. 그러나 전원공급원인 밧데리가 자료처리장치 내에 내장되어 있기 때문에 일반 사용자가 전원을 전혀 확인할 수 없다. 전원을 교체하거나 재충전할 경우 제조회사에 A/S를 의뢰해야 한다.

따라서 자료처리장치의 전원부분에서 갖추어야 할 요건은 밧데리의 현재 용량을 표시하는 장치가 필요하다. 그렇게 될 경우 사용자가 필요할 때마다 확인이 가능하고 다음 조치사항를 행할 수 있다. 두 번째는 밧데리의 교환이 쉽게 되어야 한다. 가전제품의 경우 내부 건전지를 사용할 수 있고 언제든지 외부전원을 연결하며 자동적으로 내부 건전지는 차단되고 외부 전원으로 연결되는 것처럼 자료처리장치의 외부에 연결 잭이 있어 언제든지 외부 전원이 연결가능할 수 있도록 해야한다. 세 번째는 전원이 자동적으로 전환이 가능해야 한다. AC전원, Solar cell, 외부 밧데리 그리고 내부 밧데리를 사용할 경우 전원순서를 지정할 수 있어야 한다. 예를들면 1차전원으로 AC 전원을 사용할 경우 1차전원이 단절될 때 자동적으로 2차전원이 Solar Cell로 전환되어 자료처리장치의 작동은 정상적으로 되도록 한다. 마지막으로 자료처리장치의 소비전력이 낮아야 한다. 소비전력이 자료처리장치, 센서, 모뎀을 포함하여 최소한

20mA이하가 되어야 한다. 경험적으로 자동차용 밧데리를 사용할 경우 6개월 이상은 작동이 가능하여야 한다.

마. 운영소프트웨어

지하수 관측망의 기본적인 소프트웨어는 크게 2가지로 구성된다. 하나는 자료처리 장치 자체의 기본적인 작동을 제어하는 Controller program으로써 관측망에 설치된 센서값을 측정하거나 모뎀을 통하여 장거리 전송을 하거나 노트북에 자료를 전송하는 역할을 한다. 이 프로그램은 측정값들을 시간별 그리고 일별로 처리하고 저장할 수 있어야 한다. 또한 사용자가 임의로 시간, 저장파일명등 필요한 변수를 입력할 수 있도록 해야한다. 특히 내부 또는 외부 밧데리의 용량, 저장된 자료수, 남은 메모리 용량, 가장 최근 측정된 날짜 및 시간 등이 LCD를 이용하여 디스플레이 될 수 있도록 해야한다.

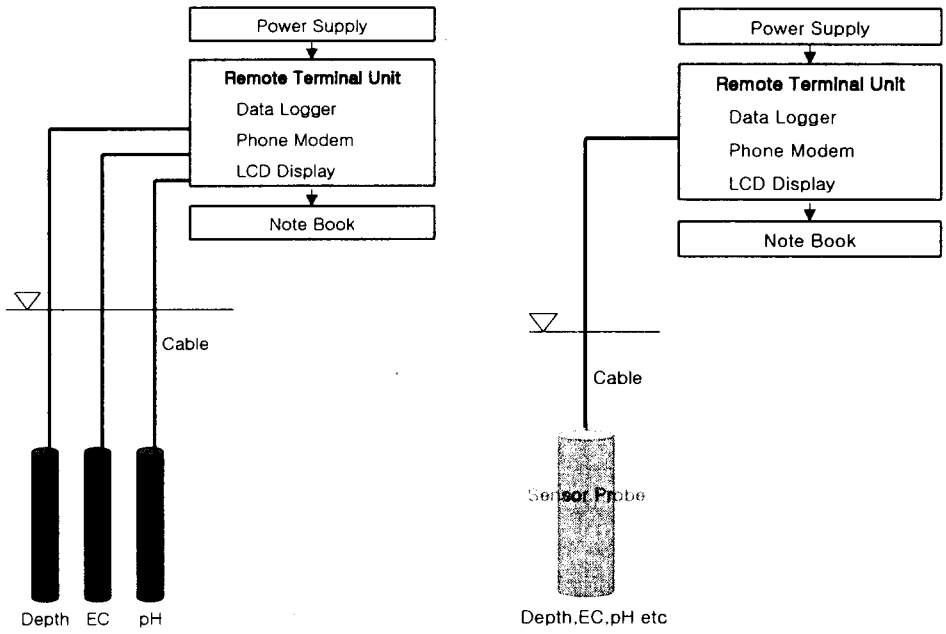
다른 하나의 프로그램은 PC용 프로그램으로 사용자가 쉽게 다룰 수 있도록 대화식으로 구성되어야 하며, 윈도우 화면의 채택으로 사용이 편리해야 한다. Data는 ASCII 표준규격을 준수하며, 언제든지 컴퓨터에서 수치로 볼 수 있고, 수치 처리용 프로그램(예:엑셀)을 사용하여 도표화, 보고서화 할 수 있어야 한다.

4. 지하수 자동관측기의 외형별 비교분석

지하수 자동관측기는 센서의 형태에 따라 <그림 3-4>와 같이 모든 측정센서가 하나의 프로브에 장착되어 있는 일체형과 측정센서들이 각기 하나의 프로브에 장착되는 분리형이 있다.

지하수 관측기기의 제작 초기에는 분리형으로 제작되어 지하수 조사, 또는 장기 관측망에 설치 및 측정하고자 할 때 모든 항목의 센서를 가지고 다녀야 했다. 이러한 불편함을 해소하기 위하여 여러 센서를 하나의 프로브에 장착된 일체형이 제작되었다. 일체형 프로브의 제작 초기에는 아무런 저장장치가 없어 측정자료를 현장에서 기

록해야 했으나 현재에는 자료 저장기능을 가진 프로브가 제작, 보급되어 현장 측정 자료를 별도로 기록하지 않아도 자료를 자동 저장함으로써 실내에 돌아와 이를 컴퓨터로 전송하여 자료분석이 가능하도록 발전되었다. 따라서 기능과 성능면에서는 일체형과 분리형의 차이점은 없다. (표 3-2)는 일체형과 분리형의 장단점 비교이다.



(분리형) (일체형)

<그림 3-4> 지하수 자동관측기 센서 형태

위의 센서 유형을 비교할 때 일체형은 분리형에 비해 최근에 개발된 것으로 보다 진보된 형태이지만 고장이나 센서교체 등으로 센서 인양시 센서전체를 인양하여야 한다. 정상적으로 작동되는 센서도 함께 인양하므로 그 동안의 측정이 불가능하며 자료를 취득할 수 없고 자료의 누락이 발생할 수 있는 단점이 있다. 만약 장기간 수리하는 경우라면 고장난 센서뿐만 아니라 정상인 센서도 측정할 수 없게 된다.

분리형은 고장난 센서만을 인양 또는 수리하므로 나머지 정상적으로 작동되는 센서는 계속 사용할 수 있다. 그러나 분리형은 여러 가닥의 케이블이 설치되어야 하므로 고심도 설치시 케이블의 꼬임이 있을 수 있다. 이를 방지하기 위하여 케이블을 합쳐 중간을 묶어 내려야 한다. 따라서 분리형일지라도 센서의 보수나 유지관리를 위한 전 센서를 모두 인양해야 하는 것은 마찬가지다. 그러나 장기간 측정하는 관측망의 경우 다른 항목은 계속 측정할 수 있는 장점이 있다.

관정 내 여러 개 센서가 설치되고 여러 가닥의 케이블이 설치되는 분리형에 비해 일체형이 관리면에서 효율적이고, 센서 종류별로 케이블을 설치하는 번거로움을 방지할 수 있어 유지관리 면에서도 편리하여 일체형 센서가 최근 많이 보급되고 있다.

제 2 절 지하수 자동관측기의 유지관리 문제점

1. 기기의 호환성

지하수 관련 자료를 정기적·주기적으로 취득하기 위하여 필요한 장비로는 자료 송수신용 컴퓨터 및 관정 별 자료저장장치, 모뎀, 송수신 회선, 각 항목에 맞는 관측장비(센서 포함) 등이 있다. 지하수 자동관측기에 사용되는 부품의 대부분은 시중에서 구입 활용이 가능하나, 지하수 관측기의 핵심부인 센서는 대부분 외국제품이 도입되어 있다. 센서로부터 나온 신호를 가공 처리한 후 자료를 저장하거나 사무실에 설치되어 있는 컴퓨터에 전송하는 자료처리장치는 국산화가 상당히 진행되어 있다.

따라서 지하수 자동관측기의 구성면에서 센서는 외국제품이며, 자료처리장치는 국산제품으로 되어 있다. 초기 설치시 충분히 호환성과 관정 특성을 고려하여 하드웨어 및 소프트웨어를 제작하였지만 운영과정에서 고장이 발생한 경우 설치업체에서는 센서부가 고장임을 강조하여 보수비용을 청구하거나 장비교체를 요구한다. 그러나 센서대 자료처리장치의 고장비율이 50:50 으로써 비슷하므로 지하수 자동관측기의 잦은 고장원인은 호환성의 결여로 판단된다.

(표 3-2) 센서의 일체형과 분리형 비교

구분	장점	단점
일체형	· 유지관리가 경제적 · 센서고장시 개별 분리교체 · 센서와 자료처리장치까지 하나의 케이블만 필요하므로 설치가 간단하고 사용이 편리함 · 대부분 RS232C 출력방식이므로 자료처리장치까지 노이즈없이 정확한 자료 취득이 가능함 · 가격이 비교적 저렴함	· 교체시 전체 센서 인양 · 측정자료를 이용하지 못함 · 전체를 교체하는 경우가 많아 유지, 보수비용이 많이 듬
분리형	· 보다 진보된 형태 · 단일 케이블 사용, 관리편리 · 설치작업이 간단 · 센서만 교체 가능함 · 비용이 적음	· 센서 종류별 케이블 별도필요 · 관정에서 자료처리장치까지 센서수 만큼 케이블로 연결되어 있으므로 설치가 복잡하고 사용이 불편함(다중케이블) · 가격이 고가임

2. 기기의 A/S 및 보수

관측자료를 이용하여 수량이나 수질변화를 추정하기 위해서는 취득자료를 수시로 점검하여 비정상적인 자료가 입력되었을 때는 즉각 자료 입력체계를 확인하여야 한다. 특히 관측자료는 연속적으로 취득되어야 수량과 수질의 변화추이 분석이 가능한데, 이를 위하여 3개월에 1회 이상 관측기를 점검하여 오차 한계를 벗어난 것이 확인되면 즉시 보정을 실시하도록 되어 있다.

지하수 관측자료로부터 정확한 자료 취득과 분석을 위해서는 지하수 보전관리를 위해 필요한 자료를 취할 수 있어야 하고, 관측기의 점검, 보정 등 유지관리를 위한 기술도 보유하여야 한다. 그러나 관측기가 수입제품으로서 고장시 지하수 담당자 및 국

내기술로 수리할 수 없는 경우가 종종 발생하여 연속적인 자료취득이 어려운 실정이다.

관측장비는 제작사별 고유 모델로 개발되어 있어 다른 제품과는 운영체계가 상이하여 호환이 거의 불가능하고, 한가지 장비에 익숙하더라도 다른 장비의 조작이나 점검·정비가 매우 어렵게 제작되어 있다.

따라서 관측망 운용자의 점검·보정으로 해결할 수 없는 고장이나 부품의 교체 등이 필요한 경우 제작사의 사후봉사가 필수적인데, 이에 소요되는 시간과 비용은 관측망 운용을 어렵게 한다. 한번 고장으로 1~2개월, 경우에 따라서는 6개월 이상이 소요되기도 하며, 소요경비 또한 구입가격의 1/3 이상~2/3까지 필요할 경우가 있다. 고장이 발생하면 6개월 이상 자료 취득이 불가능하고 소요경비 또한 만만치 않지만 장비가 고가이기 때문에 예비 장비를 확보하는 것도 쉽지 않다.

이러한 점을 감안하여 경험과 기술 미흡으로 정확도에서는 다소 뒤지더라도 실시간으로 사후관리를 행할 수 있도록 가능한 국내에서 구입할 수 있는 부품으로 지하수 관측장비를 제작하였다.

3. 취득자료의 관리

국내에 설치되어있는 여러 종류의 관측망으로부터 많은 자료가 취득되고 있는데, 이들 자료는 관측망별 분석/평가에는 이용되지만 다른 관측망과 연계하여 활용되지는 못하고 있어 자원이 낭비되고 있다.

각 관측망으로부터 취득한 자료를 한군데로 모으기 위해서는 여러 종류의 계측기로부터 취득한 자료를 무리 없이 종합할 수 있는 프로그램의 개발이 필요하고 이 프로그램에 의해 전체를 하나로 묶어 총괄적인 분석/평가가 시행되어야 한다. 취득한 자료는 통합관리가 이루어져야 한다.

그러나 구입단계에서 공공기관은 입찰방식으로 지하수 관측기를 구매한다. 현재 사용중인 관측기를 고려하여 사용목적과 방법에 있어서 비슷한 장비를 구매하는 경우가

비록 있지만 우선적으로 고려되는 항목은 입찰가격이다. 따라서 운영프로그램 뿐만 아니라 취득된 자료도 상이하여 호환성이 결여되는 경우가 많다.

제 4 장 지하수 자동관측기 제작

제 1 절 수위센서

1. 수위센서의 분류

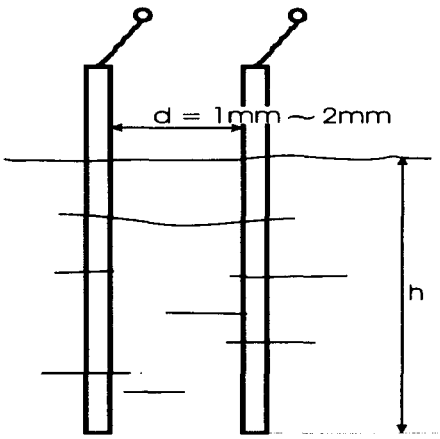
지하수 관측기 3개 항목 중 가장 중요한 측정항목은 지하수위이며, 이는 지하수의 분석 및 활용면에서 다양한 정보를 제공해 준다. 수위센서는 전기용량식, 기포식, 압력식 등 다양하다. 국내에서 사용되고 있는 수위센서는 설치 및 사용이 편리한 압력식이 대부분이다.

가. 전기용량식 수위센서

물은 전기 유도도와 전기 전도도가 있는 물체이기 때문에 만일 수심을 따라 두 개의 절연된 도선을 좁은 간격을 유지하면서 배치하면 이 두 도선이 콘덴서가 된다. 전기용량식 수위계는 <그림 4-1>과 같이 수심에 따라 일부 쌍도선은 공기 중에 있고, 일부는 수중에 배치되어 있다. 물의 전기유도도가 공기보다 훨씬 크기 때문에 콘덴서의 용량은 거의 물에 잠긴 도선쌍의 길이(h)에 비례한다. 따라서 전기용량과 수심(h) 간의 상관관계성을 이용하여 전기용량에 따른 수위를 구할 수 있다. 그러나 물의 전기유도는 물의 특성, 즉 물의 물리성과 화학성에 따라 변한다. 물의 성분과 온도는 수시로 변하기 때문에 전기저항을 정밀하게 측정하여도 오차는 발생한다. 그리고 전기저항의 변화가 매우 작기 때문에 직류성분의 전기신호를 증폭, 가공하는 과정에서 노이즈가 들어가기 쉽고, 전자회로가 복잡하여 실제 노천에서 사용하기는 어렵다.

그러므로 이러한 수위계는 현재 산업용 액체저장탱크의 수위측정에 일부 사용되고 있고, 주로 레벨 스위치로 사용되고 있을 뿐이다. 도선쌍에 직류전원을 사용하지 않고 교류를 사용하여 두 도선간의 복합저항을 측정할 수 있는데, 이 경우에는 교류를 사용하기 때문에 신호처리가 쉽고 잡음을 제거하는데 문제가 없다. 이러한 경우에도

액체의 성분과 온도변화의 영향이 커서 수리, 수문분야의 수위측정에는 사용되고 있지 않다. 현장설치시 시간이 경과되면 도선에 유기미생물 층이나 무기물 층이 형성되어 초기에 교정했던 특성이 전혀 유지되지 않는다. 이와 같이 전기 용량식 수위센서는 지하수, 하천, 저수지 및 관개수로에서의 수위측정으로 적합하지 않다.



<그림 4-1> 전기용량식 수위센서의 구조

나. 기포식 수위센서

기포식 수위센서는 <그림 4-2>와 같이 수심에 따라 수주압 측정관을 h깊이까지 설치하면 파이프관에도 물이 채워져 수면의 위치는 모세관현상을 무시하면 외부수면 위치 H와 동일해진다. 이 때 파이프관 하단에서의 압력은 다음과 같이 된다.

$$P = \rho H \qquad (1)$$

P : 수압, ρ : 물의 밀도

H : 관의 말단에서 수면까지의 높이

만일 수압측정관 상단에 압력계를 설치하고 수압보다 약간 큰 압력으로 압축공기

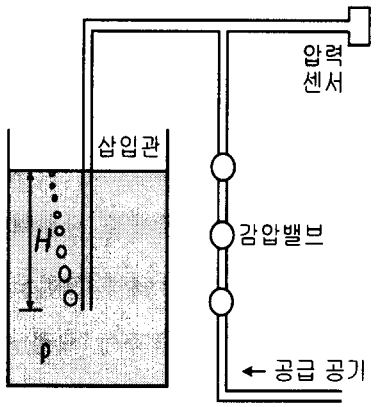
를 공급하면 수압측정관에 채워졌던 물이 관의 하단을 통해 밀려나가고 기포가 나오기 시작한다. 이 때의 수압측정관의 압력은 수압과 동일해졌다고 볼 수 있으므로 압력계로 측정한 압력(P)으로 수위(H)를 계산할 수 있다.

기포식 수위센서의 장점은 수압측정관을 경사지거나 임의로 휘어지게 설치해도 수압측정에 하등의 영향을 미치지 않는다. 측정관으로 폴리우레탄 튜브 등 어떤 종류의 관도 사용할 수 있어 측정관의 설치비가 대폭 절감되며, 측정관의 보수도 필요하지 않다. 다만 압축공기를 이용하기 때문에 이를 만들 수 있는 콤프레서를 사용해야 한다. 수위측정범위가 클수록 용량이 큰 콤프레서를 사용해야 하며 그만큼 대용량 전원이 필요하다. 기포식은 튜브내의 공기를 불어내야 하므로 소형공압기 혹은 질소와 같은 압축공기 탱크가 필요하다. 소형 공기압축기는 측정 수심범위가 작을 경우 (10 - 20m) 밧데리로도 가동이 가능하나 100m 정도 수위측정은 AC전원 또는 대용량 밧데리를 필요로 하는 단점이 있어 20m이하 수위측정용으로만 사용하고 있다.

기포식 수위센서가 압력식 수위센서보다도 먼저 개발되어 활용되었으나 Strain gauge가 온도변화에 따라 심하게 특성이 변하는 단점이 있어 지상에 설치되는 기포식 수위계는 겨울철과 여름철의 기온변화에 크게 오차를 발생하게 된다. 또한 물의 비중은 온도변화, 성분, 특히 부유물의 농도에 따라 변한다. 지하수, 지표수의 비중은 일반적으로 ρ=1.0g/cm^{3}이 아니다. 온도4℃의 증류수 밀도가 1.0g/cm^{3}이나, 지표수나 지하수의 물에는 각종 부유입자들이 포함되어 있으며, 무기물질도 용해되어 있고, 온도도 수시로 변하기 때문에 밀도가 수시로 변화게 된다. 예를 들면 장마기에는 부유입자의 농도가 5g/ℓ정도이고, 물의 온도가 20℃이며, 다른 물질이 용해되어 있지 않다면 0.9997단위 정도가 되어 부유물질에 의한 밀도변화율은 0.5%된다. 수위(h)=1.0m이면 측정오차는 5mm, h=10.0m이면 오차가 5cm 나 된다. 저수지나 지하수의 수위가 10.0m를 초과하는 경우가 많다.

수압을 아무리 정밀하게 측정한다 하여도 수시로 변하는 하천과 저수지 물의 비중을 정확히 모르면 수위측정 오차가 커진다. 결국은 지하수의 수온이 거의 일정하다는

가정하에 Strain gauge가 수중에 설치되어 있는 압력식 수위계가 개발 발전하게 되었다.



<그림 4-2> 기포식 수위계의 구조

최근 다시 기포식 수위센서가 개발되고 있는데, 이는 Strain gauge의 제조기술이 최근 급격하게 발전되어 온도변화에도 저항력이 큰 Strain gauge가 개발되었으며, 온도보상회로가 첨가되어 과거보다는 정확한 측정이 가능하기 때문이다. 그러나 압축공기를 만들기 위하여 외부에 압축공기탱크나 콤프레셔를 설치해야 한다는 점 때문에 Field(노천과 같은 실외)에의 사용실적이 저조하다.

다. 압력식 수위센서

압력식 수위센서는 정밀도가 양호하고 직선성 출력을 가진 압력센서를 사용하므로 설정의 용이성이나 설정오차가 작다는 장점이 있어 많은 제품들이 개발, 보급되어 있다.

압력식 수위센서는 <그림 4-3>과 같이 수압측정원리를 이용하여 압력센서를 수중에 고정설치하고 센서에 전원을 공급하며 센서출력신호를 전달하는 케이블선으로 계측기

가 연결되어 있다. 압력센서에 가해지는 압력은 수압 ρH와 물의 표면에 작용하고 있는 대기압 Pa의 합이다.

$$P = \rho H + Pa \quad (3)$$

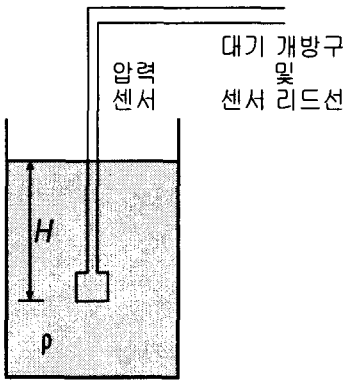
$$H = (P - Pa)/\rho \quad (4)$$

P : 수압

Pa : 대기압

식(3)에서 압력센서에 미치는 압력은 수위와 대기압의 합이다. 그러나 대기압은 높이의 차에만 관련된 것이 아니고 공기중의 습도, 풍속 및 온도차에 의해서 수시로 변한다. 예를 들면 지하수에서 센서가 50m 위치에 설치된 경우 대기압의 변화가 $6 \times 10^{-3}kg/cm^3$이라면 수위측정 오차는 보충적으로 6cm의 오차가 발생하며, 높이차이가 25m이면 3cm, 10이면 1.2cm 의 오차가 발생된다. 이러한 오차를 줄이기 위해서 센서 케이블선안에 내경이 작은 튜브가 들어있고, 한 끝은 센서와 연결되어 있으며, 다른 끝은 압력센서 부근에 있다.

압력식 수위센서는 다른 수위센서보다 설치가 용이하고 측정범위가 넓은 장점을 가지고 있다. 그러나 일년에 최소한 한 번은 특성검사를 해야 하는데 수중에 설치된 압력센서를 대기압에서 측정범위까지 검사한 다음 다시 원위치에 설치해야 하므로 유지관리가 필요하다. 또한 대기압을 보상하기 위하여 케이블내에 튜브를 삽입한 특수한 케이블을 사용해야 하는데 이 케이블 값이 비싸므로(20,000원/m) 초기 설치비용이 과다하다. 그리고 설치 중 또는 외부의 충격에 의하여 손상을 받았을 경우 절단하지 못하고 케이블 전체를 교체해야 하는 단점을 가지고 있다.



<그림 4-3> 압력식 수위센서

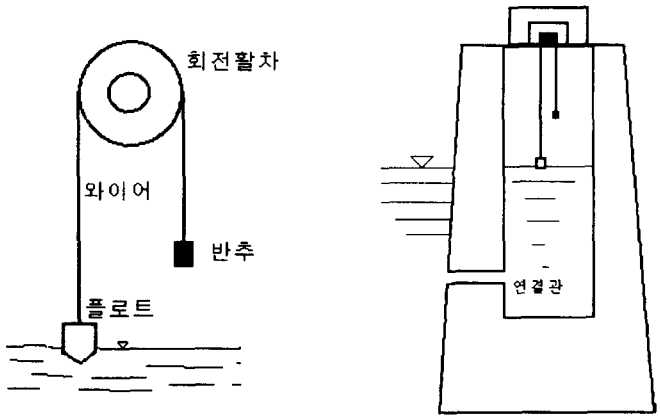
라. 플로트식 수위센서

플로트식은 수면의 변화에 따라 동작하는 플로트의 움직임을 포착하는 방식이며, 플로트의 움직임을 수위로 환산하는 방식이다. <그림 4-4>와 같이 플로트, 회전차, 반추 그리고 줄자로 이루어져 있다. 수위가 상승할 때 플로트의 부력에 의하여 상승하며, 수위가 낮아질 때는 부력이 아니라 물과 플로트의 부착력, 반추의 중력에 의하여 작용한다. 즉 수위상승시와 저하시에 회전활차를 회전시키는 힘이 다르다.

반추의 무게 선택은 매우 복잡한데 우선 플로트가 내려갈 때 플로트가 물에서 이탈되면 안되며, 동시에 회전활차가 플로트이동에 정비례하여 회전할 수 있도록 와이어 줄과의 접촉마찰력을 보장해야 한다. 수위측정 범위가 작을 경우에는 이와 같은 요구조건을 어느 정도 만족시키지만 측정범위가 10m를 초과하면 만족시키기 어렵다.

또한 수위가 빠른 속도로 변할 때 플로트의 위치도 빠르게 변할 수는 있지만 와이어 줄이 회전활차를 회전시키지 못하여 미끄러짐 현상이 발생한다. 자연하천에서는 비교적 천천히 변화하지만 관개용수 등 수위가 급속히 변화하는 지역에서는 사용이 어렵다. 가장 큰 단점은 유수의 표면에 플로트를 띄울 수 없기 때문에 하천, 저수지 및 수로와 연결되는 정수정이 반드시 필요하다. 수위센서보다는 정수설치비용이 많기 때문에 국내 설치가 많이 되어 있지 않지만 정기적으로 보수운영을 하면 수명이 매우

길고 소모품의 교체가 간단하며 전문가가 아닌 일반인도 보수운영을 할 수 있을 뿐만 아니라 동작원리가 쉽게 눈에 보여 신뢰도가 높아 국내에서 다른 수위센서에 비하여 많이 사용되고 있다.



<그림 4-4> 플로트식 수위센서

마. 기타

위에서 언급한 수위센서 외에 액면에 초음파를 방사하여 송신으로부터 액면에서 반사한 에코가 수신되기까지의 시간 간격을 수심으로 변화하는 초음파수위센서, 동위원소, 모터, 레이저 등 여러 가지 방식 등이 있다.

지표수, 지하수 등에 사용되는 수위센서는 현장(노천 등 주로 실외에 설치)에서 동작되어야 하므로 설치조건, 가격, 운영보수의 편의성, 기술 및 경제적 타당성 등을 종합적으로 검토하여 여러 가지 수위센서 중 요구조건들을 만족시키는 수위센서를 선택해야 할 것이다.

(표 4-1) 수위측정방식별 비교

센서 형태	용도별	측정범위 (m)	측정원리	가격 (백만원)	측정 정확도
압력식	장기관측	0~100	수면과 기준면의 수압을 압력계로 측정	3~10	모델별 차이발생 ±0.5%
	양수시험	0~100		7~10	정확함 ±0.1%이하
기포식	장기관측	0~50	압축기체를 불어 넣어 압측정으로 수위환산	7~10	비교적 정확함 ±0.1%이하
장력식	장기관측	0~20	부력에 의한 질량변화를 수위로 환산	2~4	비교적 정확함 ±0.1%이하
음파식	장기관측	0~20	음파로 기준점부터 수면까지 거리측정	2~4	비교적 정확함 ±0.01%이하
플로트식	장기관측	0~10	수면에 플로트를 띄워서 수면의 변화측정	0.8	정확함 ±1%

센서 형태	용도별	단점	구입시 착안점	기타
압력식	장기관측	가격이 비싸고, 가격에 따라 정확도가 다양함	기압변화에 대한 보정기능	소형밧데리 작동가능
	양수시험	가격은 비싸지만 정확함		
기포식	장기관측	급격한 수위변화측정불가능	설치장소에 따른 기포발생 방법	기포발생위한 가스통이 필요함
장력식	장기관측	측정범위가 적음		소형밧데리로 장기간 사용
음파식	장기관측	측정범위가 적음		측정범위가 적어 수문측정용으로 적합
플로트식	장기관측	측정심도가 작음		1.5V 건전지로 1달 사용가능

2. 수위센서 제작

지하수 관측망용 수위센서는 수명이 반영구적인 플로트식, 기포식, 장력식 센서들이 경제적이나 설치시 제약성이 많으므로 설치할 관정 상황 (자연수위, 수위변화폭, 수중모터사용여부 등)에 적합한 방법으로 변형되어 설치하여야 한다. 위 수위센서들은 수위방법이 비교적 단순하여 정확도가 높으며 가격도 저렴하다는 장점이 있다. 이중 플로트식과 장력식으로서 상품화된 센서는 측정범위가 10~20m 정도의 수위변화로 제한되어 있으며, 기포식의 경우 측정범위는 비교적 넓으나 급격한 수위변화 측정이 어렵고 기체공급을 위하여 콤프레사나 고압가스통이 필요하다.

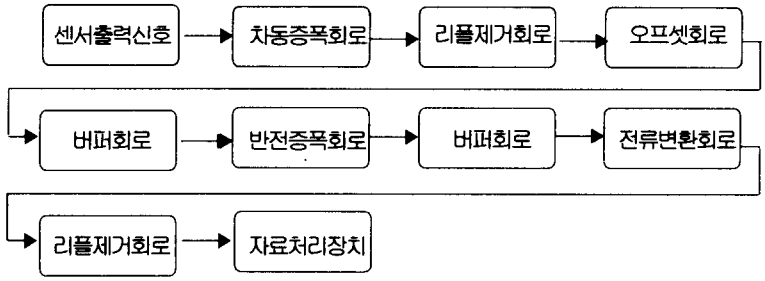
가장 널리 보급되고 다양하게 개발되어 있는 형태는 압력식이며, 실제 장기관측망으로 사용할 때 6개월에 1회정도 교정을 하면서 사용할 경우 상당히 정확한 측정값을 얻을 수 있다. 장기관측망 또는 도서지방 해수침투용에서는 수위변화가 10m 이하로 측정범위가 넓은 센서를 설치할 필요가 없다. 그러나 양수시험 또는 관정 개발시 순간 수위변화가 100m까지 변화가 있을 수가 있으며, 심지어 1분이내에 변화가 일어나기도 한다. 따라서 측정범위가 넓고 반응속도가 빠른 압력식으로 하는 것이 가장 적당하여 본 연구에서는 압력식 수위센서를 선택하였다. 압력식 수위센서는 편리성, 용이성을 고려할 때 다른 측정방식보다 가장 적당하지만 가격이 고가임이 단점이다. 본 연구의 수위센서는 압력식으로 하였으며 최대한 국내 조달이 가능한 부품을 사용하여 저가형으로 개발하였다.

가. 수위센서 회로설계

수위센서의 크기를 최소화하기 위하여 SMD형 전자부품을 사용하였다. 또한 지하수 관측기의 특성상 측정하고 있지 않은 상태(대기모드)에서는 센서전원이 차단된 상태이며, 측정 시에는 약 5초 동안 전원이 센서에 공급된다. 전원이 공급되는 동안 측정, 저장 등 모든 일련의 작업을 처리해야 하므로 안정화 시간이 필요하지 않은 IC를 사용하였다.

신호처리과정에서 노이즈가 발생될 수 있는 부분에 콘덴서와 저항을 사용한 노이즈 제거회로, 버퍼회로를 구성하여 센서회로 내에서 발생한 노이즈를 자료처리장치에 전달되기 전에 제거토록 하였다. Voltage Regulator를 사용하여 +5V, -5V를 만들어 양전원을 사용하여 증폭율을 넓게 하여 신호처리를 보다 정밀하게 처리할 수 있도록 하였다. 또한 수위센서 케이스를 접지로 활용함으로써 신호의 안정성을 높였다.

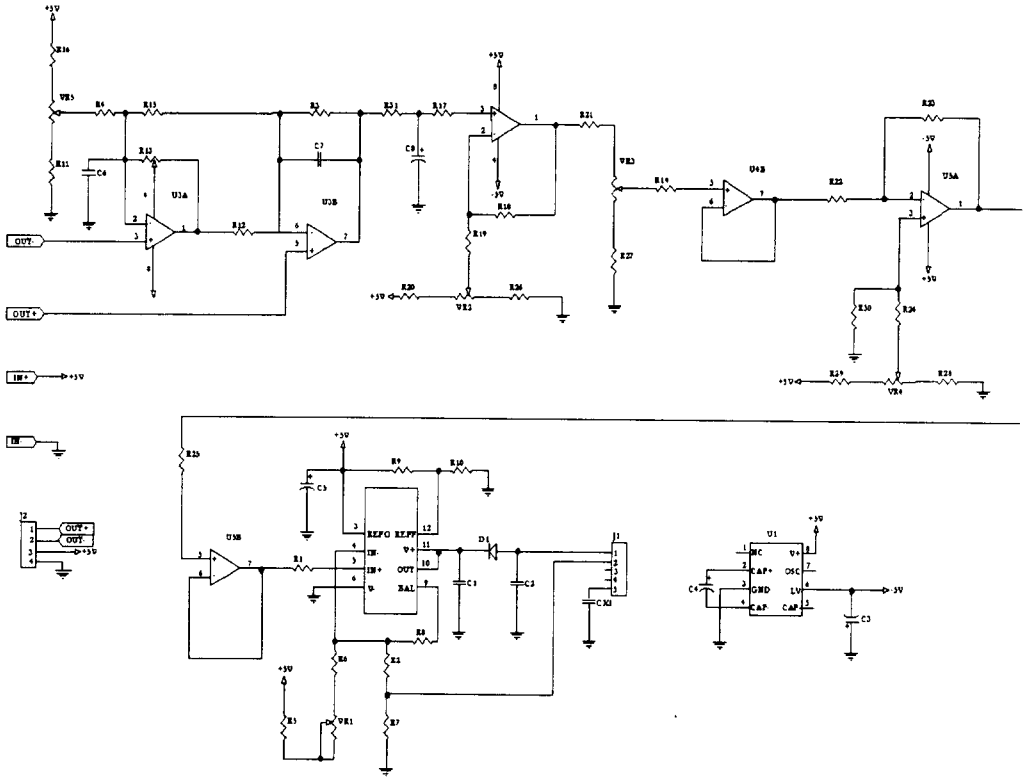
압력센서로부터 나오는 신호를 가공, 처리하여 최종적으로 DC 12V, 4-20mA로 출력할 수 있도록 하였다. 이러한 신호처리과정은 <그림 4-5>와 같이 설계하였으며 회로도는 <그림 4-6>과 같다.



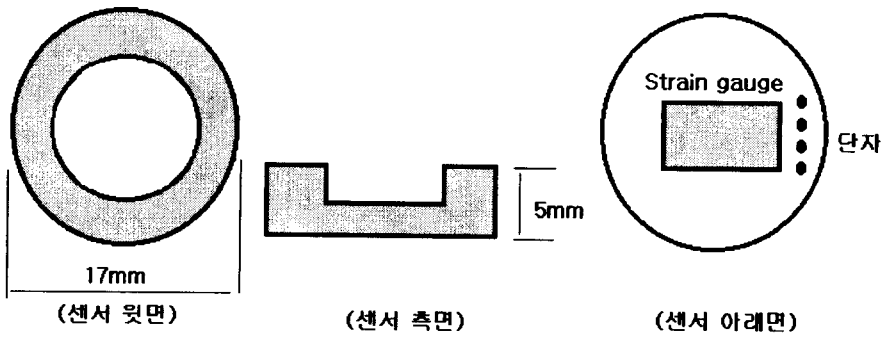
<그림 4-5> 수위센서의 신호처리과정

수위 센서에서 사용된 압력센서(L-100, Germany)내에 <그림 4-7>과 같이 수압변화에 대한 전기적 신호를 1차적으로 증폭하는 브리치회로가 내장되어있다. 입력단자(IN+, IN-)는 전원단자이며, 출력단자(OUT+, OUT-)는 센서 출력신호이다. 전원 0, +5V일 때 출력신호는 <그림 4-8>과 같이 대기압(1기압)에서 약 2.0V, 11기압에서 3.5V이다.

이 출력신호를 가공하여 최종출력신호로 사용할 수 있으나 기준이 되는 대기압조건에서(1기압)에서 0.0V가 되어야 한다. 기준값을 0.0V로 할 경우 증폭회로, 전류변환회로 구성이 비교적 간단하며, 또한 가변저항을 사용하여 쉽게 Zero와 Span조정이 가능하다.

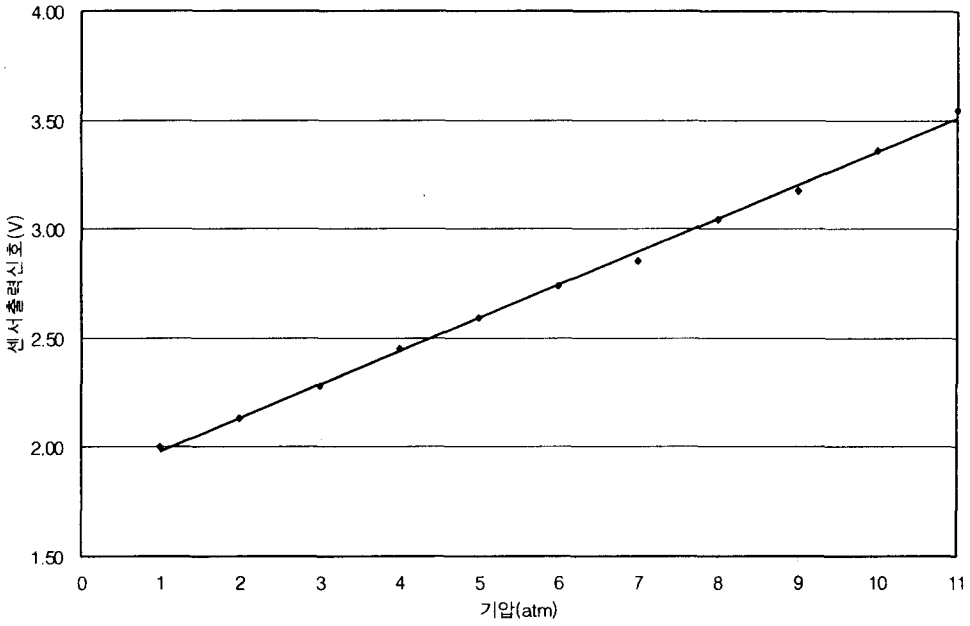


〈그림 4-6〉 수위센서 회로도

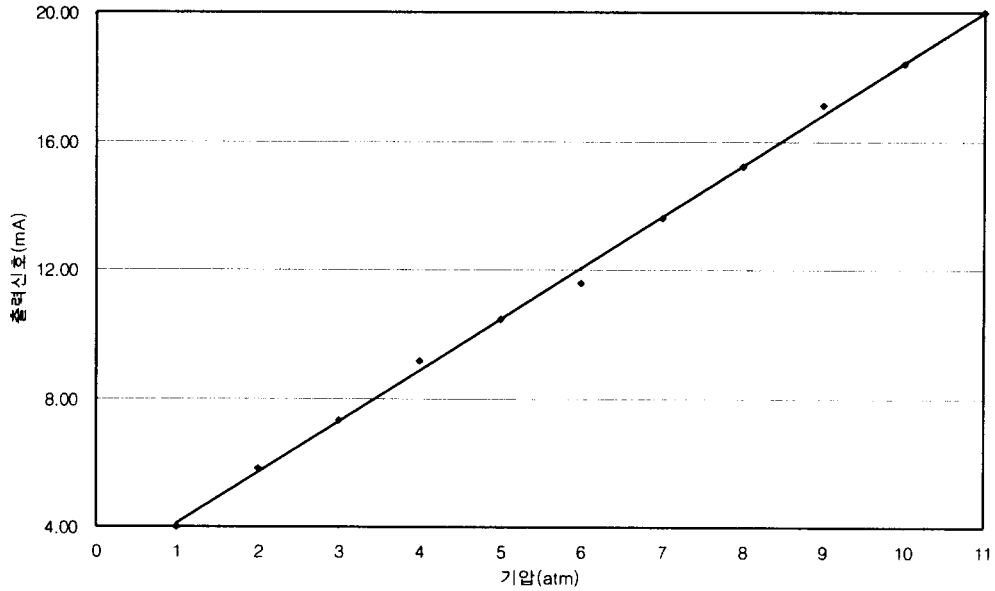


〈그림 4-7〉 수위센서에 사용된 압력센서의 구조

우선 센서로부터 나오는 출력신호(-,+단자)를 개별적으로 처리하는 차동증폭회로에 연결하여 기준점이 되는 대기압에서 가변저항(VR5)을 이용하여 1.0V로 출력되도록 조절하였다. 오프셋회로에서는 기준값 즉 대기압조건 0.0V가 출력되도록 가변저항(VR2)을 이용하여 조절하였다. 최종출력신호 4~20mA로 변환하기 전 비반전증폭회로를 응용하여 입력신호에 대하여 출력신호를 -1.0~1.0V가 되도록 하였다. 최종적으로 LM10NM chip을 사용하여 대기압(0기압)에서는 4.0mA, 10기압(수압)에서는 20mA 출력되도록 하였다.



<그림 4-8> 압력센서로부터 나오는 출력신호



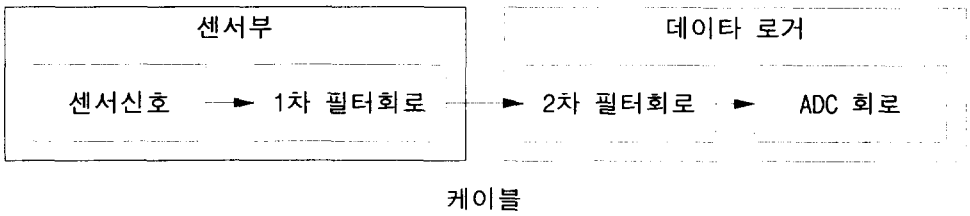
<그림 4-9> 기압에 대한 수위센서 최종출력신호

나. 노이즈제거

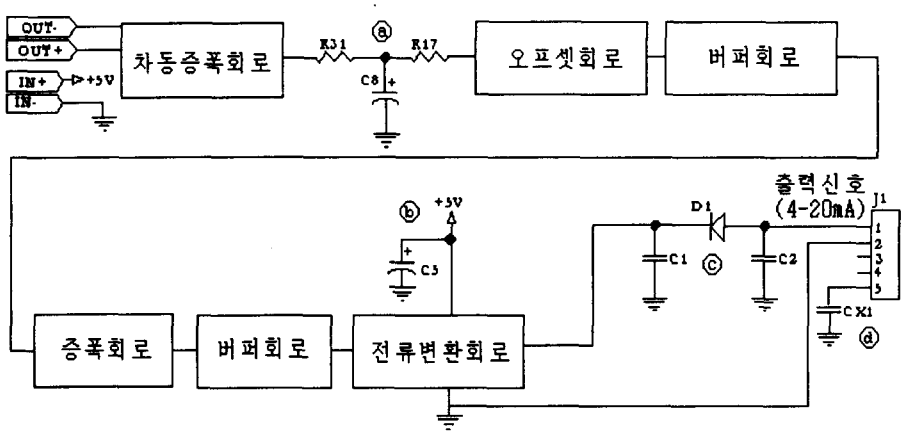
<그림 4-10>과 같이 수위센서의 노이즈제거는 센서와 자료처리장치에 각각 설치하였다. 센서에서 발생되는 노이즈는 센서회로내에서 제거하였으며, 케이블을 통하여 전송되는 과정에서 발생되는 노이즈는 자료처리장치내 필터회로에서 제거토록 하였다. 이와 같이 노이즈 제거회로를 추가하여 모터 등에 의한 전기적 신호에 대하여 안정적으로 출력신호를 얻을 수 있었다.

OP회로 내부에서, 또는 외부에 접속한 저항에 의해 노이즈가 발생된다. 증폭도가 변하면 노이즈 전압도 증폭되기 때문에 <그림 4-11>과 같이 신호처리과정에 노이즈가 발생될 소지가 있거나, 원인이 될 수 있는 지점에 노이즈 제거회로를 추가하였다. ⓐ지점은 압력센서에서 출력된 (+)와 (-)신호를 1차 처리한 지점으로 다음 오프셋나 증폭회로로 이전에 노이즈를 제거하였다. ⓑ지점은 전압을 전류로 변환시키는 전류변환부에 가해지는 전원에 대한 노이즈를 제거하는 부분이다. ⓒ지점은 4~20㎃로 출력

되는 최종출력신호의 노이즈를 하는 부분이다. ⓓ는 수위센서 케이스의 노이즈를 제거하는 지점이다. 이는 케이스를 통하여 들어오는 노이즈를 사전에 방지할 수 있다.



〈그림 4-10〉 수위센서의 필터회로

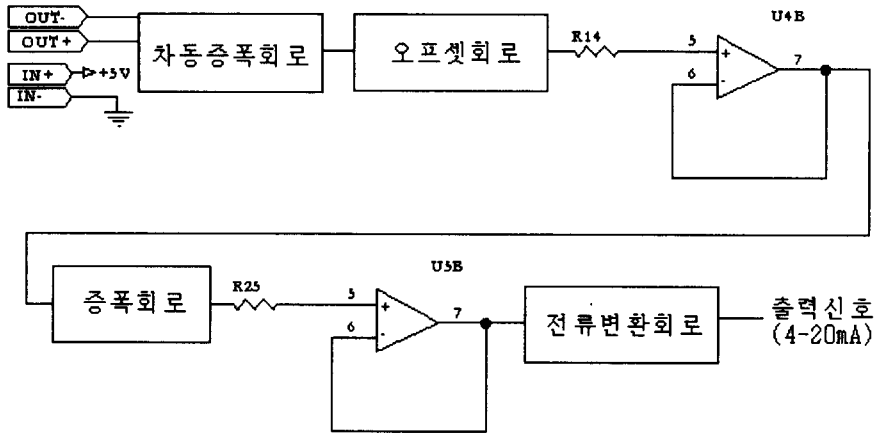


〈그림 4-11〉 수위센서 회로내 노이즈 제거회로

다. 버퍼회로

증폭회로에서 회로내의 저항특성에 의하여 입력전압을 증폭 또는 감쇄한다. 증폭회로의 입력신호는 오프셋회로의 출력신호에 해당된다. 즉 다음 회로의 부하변동에 따

라 전 단계의 부하변동이 일어날 수 있다. 수위센서회로에서 각 단계별 특성을 가지고 있으므로 전후 신호처리단계에서 영향을 미칠 경우 신호교정을 어렵게 한다. 따라서 <그림 4-12>와 상호 신호간섭이 있을 수 있는 오프셋회로와 증폭회로, 증폭회로와 전류변환회로 사이에서 전기적으로 Isolation역할을 할 수 있는 일종의 버퍼회로를 추가 설치하였다.



<그림 4-12> 수위센서 버퍼회로

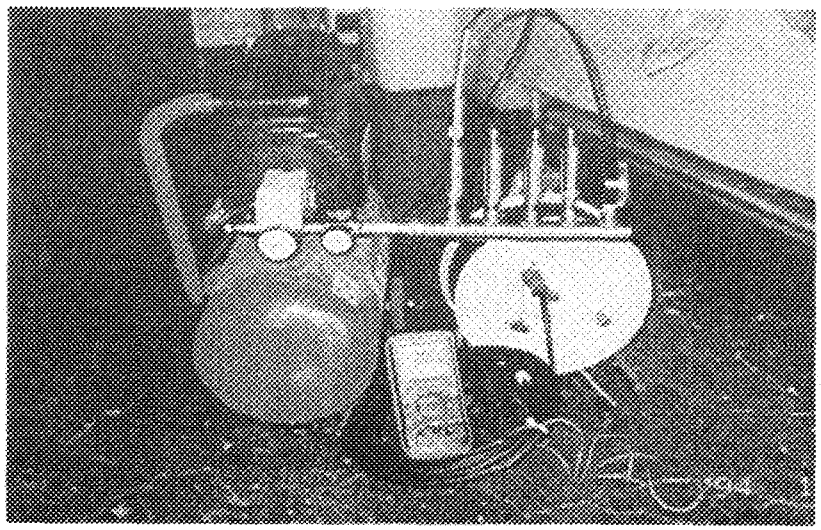
라. 교정방법

수위센서 교정은 소프트웨어와 하드웨어적으로 할 수 있다. 소프트웨어적으로 하는 방법은 부록 1에 실린 지하수 자동관측기 설명서에 언급하였다. 하드웨어적으로 하는 방법은 실내에서 실시하는 교정방법이다. 이는 센서 제작시 또는 수리하는 경우에만 사용이 가능하다. 이 교정방법은 <그림 4-13>과 같이 콤프레사에 수위센서를 연결한 후 압력게이지를 확인하면서 기압증가에 따라 (표 4-2)에 제시한 출력신호가 나오도록 가변저항으로 조절한다. 대기압 조건에서 4.0mA, 10기압에서는 20.0mA가 나오도록 조절한다.

이와 같이 하드웨어적으로 출력신호를 조절한 상태에서 실제 관정에 넣어 수위와 그에 해당되는 디지털을 구하여 각 수위센서에 대한 표정곡선을 작성하였다. 표정곡선 작성방법을 제5장에 설명하였다.

(표 4-2) 교정을 위한 기압과 수위센서출력신호

기압(atm)	출력신호(mA)	기압(atm)	출력신호(mA)
1.0	4.00	6.5	12.80
1.5	4.80	7.0	13.60
2.0	5.60	7.5	14.40
2.5	6.40	8.0	15.20
3.0	7.20	8.5	16.00
3.5	8.00	9.0	16.80
4.0	8.80	9.5	17.60
4.5	9.60	10.0	18.40
5.0	10.40	10.5	19.20
5.5	11.20	11.0	20.00
6.0	12.00		



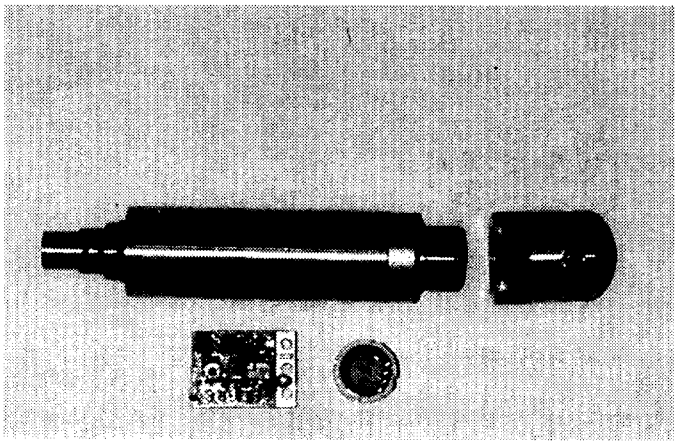
〈그림 4-13〉 수위센서의 하드웨어적 검교정방법

마. 수위센서 외형

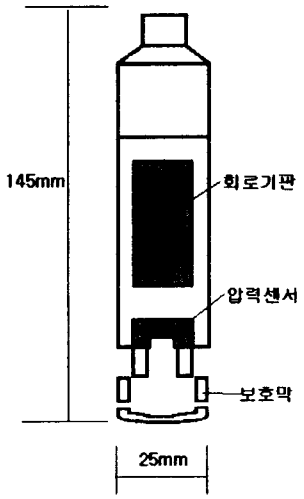
<그림 4-14>와 같이 54㎜ 관측용 관정에 사용할 수 있도록 센서 외경을 25㎜로 하였으며, 커넥트를 포함하여 길이를 145㎜로 하였다. 센서의 외관은 스테인레스(316L)로 하였다.

수위센서의 측정범위가 0~100m 이다. 센서가 100m에 설치되었을 경우 상당한 압력을 받게 되며 결합부위에 물이 스며들 수 있다. 따라서 모든 결합부위에는 이중 고무링과 실리콘 수지를 사용하여 완벽하게 방수처리를 하였다.

수위센서 설치시 수면에 바로 접하게 될 때 수면의 이물질 또는 설치도중에 압력센서가 손상되는 것을 방지하기 위하여 압력센서 부분에 보호막을 별도로 설치하였다. 이 보호막을 센서와 분리할 수 있도록 센서내 이물질 세척을 용이하도록 하였다.



<그림 4-14> 수위센서 외형과 회로기판



<그림 4-15> 수위센서 외형

제 2 절 전기전도도센서

1. 측정원리

물 속에서 한쪽 전극에서 전원을 공급할 때 전해질 물질의 양에 따라 다른 쪽 전극에 도달하는 신호의 세기가 달라지게 된다. 전기전도도센서는 이를 측정하는 것으로 물 속의 전해질들에 의한 저항의 크기로 변환한 것이 전기전도도이다. 전기전도도는 온도에 의한 영향이 크므로 반드시 센서 내부에는 온도센서가 있다. 온도센서와는 달리 전기전도도 센서로부터 일차적으로 변화된 전기적 신호는 아주 미약하여 측정환경 또는 수명에 영향을 받으므로 수시로 교정이 필요하다.

측정원리를 살펴보면 다음과 같다. 용액에 담겨져 있는 2개의 전극에 일정한 전압을 가해 주면 가한 전압이 전류를 흐르게 하며, 이때 흐르는 전류의 크기는 이온세기에 비례한다. 즉 전기전도도와 상관관계가 성립하며 이때 저항을 측정하여 전기전도

도로 환산한다. 전기저항(R)은 다음 식과 같다

$$R(\Omega)=\rho \cdot L/A \qquad (5)$$

위 식에서 ρ는 전기비저항($\Omega \cdot$ cm)이고, L은 두 전극간의 거리(cm), A는 단면적이다. 전기전도도(L)는 다음 식과 같다,

$$L=1/R=A/1 \cdot K \qquad (6)$$

전기전도도는 전기저항의 역수로써 $\mho$ 또는 mho로 나타내나 현재는 국제단위계인 S(simens) 단위를 사용하고 있다. 여기서 K(=1/ρ)는 비전도도(mho/cm)이며, 동일 측정계를 사용할 경우 셀의 규격은 일정하므로 두 전극간의 거리와 단면적은 무시할 수 있다. 따라서 측정결과는 시료의 전기전도도값(mho)에 셀정수(cm^{-1})를 곱하여 시료의 전기전도도값(μmhos/cm)으로 표시한다. 주로 국제단위계인 mS/m, dS/m 또는 μS/cm 단위로 측정결과를 표기하고 있다. 여기서 1S=1mhos와 같고 1μS=1μmhos와 같다.

전도율은 수온에 의해 변화하고 수온이 높아질수록 전기가 통하기 쉬워져 값이 커진다. 따라서 어떤 일정한 기준온도를 정해 놓지 않으면 측정치를 상호 비교할 수 없게 된다. 기준온도로는 25℃가 사용되며, 평균적으로 1℃ 증가하면 전기전도도는 약 2% 증가한다. 예를 들어 현장의 수온(T)과 25℃와의 차이를 $\triangle$t(T-25)℃라고 하면 보정식은 다음과 같이 나타낼 수 있다.

$$K_{25}=K_t(1 + 0.02*\triangle t) \qquad (7)$$

전기전도도 측정계는 25℃에서의 자체온도 보상회로가 장치되어 있어 별도로 온도

보정을 할 필요가 없다.

물의 종류에 따른 전기전도도의 범위는 증류수는 0.5-5 μS/cm, 강수는 5.0-30 μS/cm, 담수·지하수는 3-2,000μS/cm, 해수는 45,000-55,000 μS/cm 그리고 염수는 100,000 μS/cm이상이다.

전기전도도는 실내에서 분석해야 하는 TDS 성분들을 대체하여 물의 특성을 파악하는 데 많이 이용할 수 있다. 전기전도도와 TDS는 ±100mg/l 정도의 오차는 있으나 상관관계가 성립한다. TDS(Total Dissolved Solid)의 계산식은 다음과 같다.

$$TDS(ppm) \fallingdotseq 0.59 \times EC\ (\mu S/cm) \qquad (8)$$

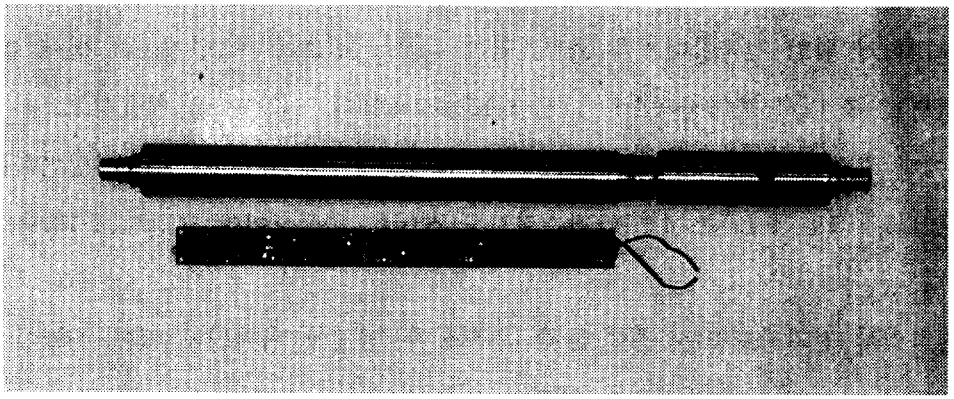
그러나 고용물질의 함량이 1000ppm 이하인 경우 오차가 크지만 고농도의 경우에는 비교적 일치한다.

2. 2전극방식 전기전도도 센서

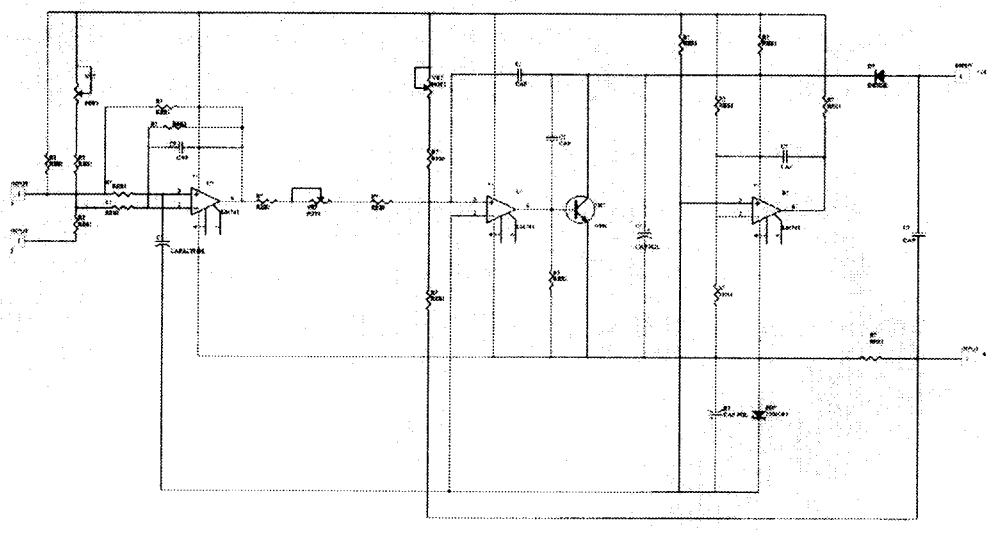
2전극방식 전기전도도센서는 <그림 4-16>과 같이 수위 센서와 동일한 직경 25mm로 하였으며, 커넥터를 포함하여 400mm로 제작하였다. 수위센서와 함께 사용하는 경우와 전기전도도 단독으로 사용하는 경우를 고려하여 전기전도도 센서의 양단자에 커넥터를 설치하였다. 즉 하나는 센서케이블과 연결하는 부분이며, 다른 커넥트는 수위센서와 연결하는 부분이다. 또한 전기전도도 회로기판은 센서 내부에 설치가 가능하도록 240mm×15mm크기로 제작하였다.

전기전도도 센서의 수리와 교환이 용이하도록 하기 위하여 두 부분으로 나누었다. 위 부분은 전극만을 장착한 것으로 실리콘 수지로 완전 방수 처리하였으며 별도의 전기회로를 설치하지 않았다. 아래 부분에는 전기전도도 회로기판을 장착하였다. 외부의 충격이나 전기적인 손상은 주로 회로 부분에서 일어나므로 고장이 있을 경우 회로기판을 쉽게 교환할 수 있도록 설계하였다.

〈그림 4-17〉은 2전극방식 전기전도도 센서의 회로도로서 출력단자에서 1.2kHz의 사인파를 출력하면 입력단자에서 이 신호를 받아 직류전압으로 변환한 후 다시 전류신호 4~20㎃ 정도 변환한 후 센서케이블을 통하여 자료처리장치에 전송하게 된다.



〈그림 4-16〉 2전극 전기전도도센서 외형과 회로기판



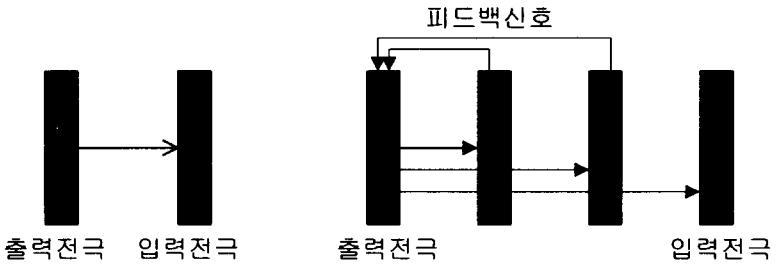
〈그림 4-17〉 2전극 전기전도도 센서 회로도

3. 4전극방식 전기전도도

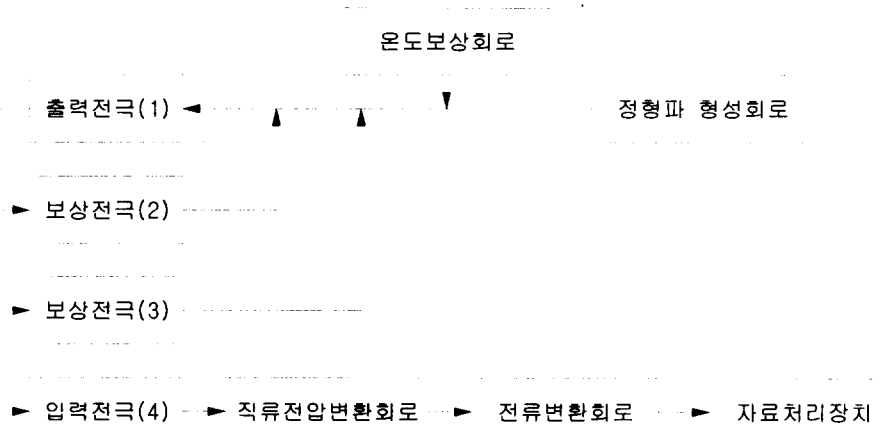
2전극방식은 케이블 길이가 50m이하 일때 문제되지 않지만 50m이상일 때 케이블의 저항 또는 회로부품내의 저항 등에 의하여 오차가 발생될 수 있다. 이를 보완하고자 <그림 4-18>과 같이 4전극방식 전기전도도 센서를 제작하였다. 첫 번째 전극은 출력단자이며, 넷 번째 전극은 입력단자이다. 두 번째와 세 번째 전극은 센서내의 저항을 보상하기 위한 전극이다. 즉 출력전극에 나오는 신호를 두 번째, 세 번째 전극에서 입력한 후 다시 출력전극으로 다시 Feedback하였다. 이 방식을 적용함으로써 회로내부에서 흐르는 전류 및 각종 부품(저항, IC, 코일, 콘덴서)에 의한 전압변화를 보상하였다. <그림 4-19>는 4전극 전기전도도 센서의 구성도이며, <그림 4-20>은 회로도이다.

2전극방식 전기전도도는 센서 몸체 내부에 두 전극이 위치되어 있어 수압이 있을 경우 전극 사이 기포가 발생하지 않지만 낮은 수압(2.0m이하)에서는 전극사이 기포가 형성되어 정상적으로 측정되지 않은 경우가 종종 발생하였다. 4전극방식을 사용한 전기전도도 센서의 전극은 <그림 4-20>과 같이 전극을 외부로 노출시켰다. 또한 전극보호막에 물의 흐름이 원활하도록 가공처리 하였다.

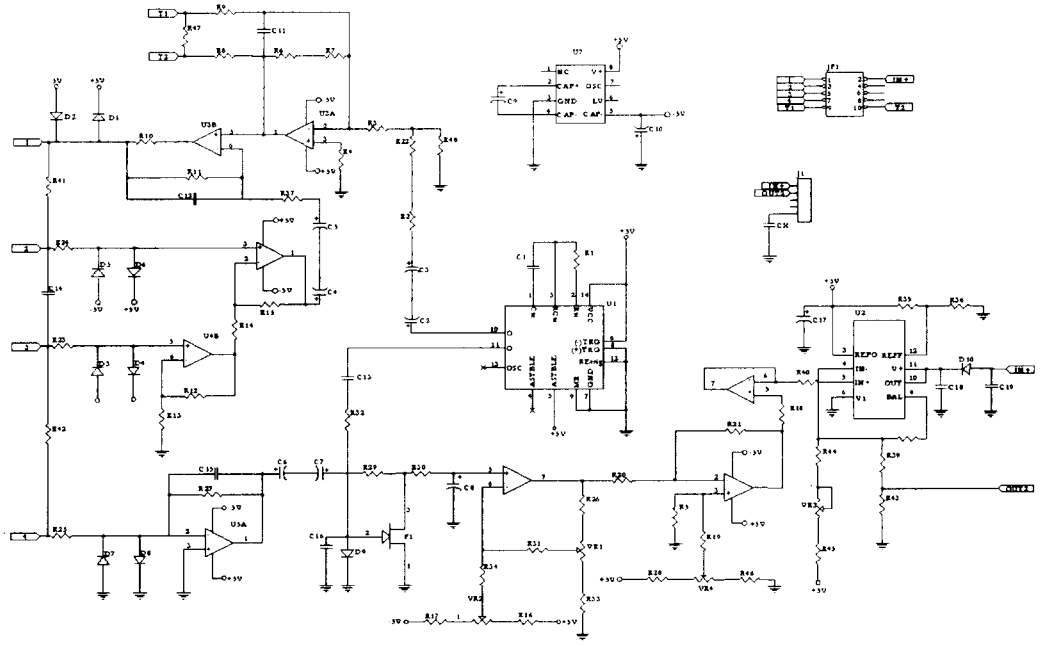
전극 보호막을 금속 재질로 제작하였을 경우 <그림 4-22>와 같이 센서출력값이 직선형이 아니라 곡선형이 되었다. 자료처리장치에서 센서표정곡선 작성시 2점 방식을 채택하므로 측정범위의 중간부분에서는 실측값과 측정값과의 오차가 크게 날 수 있다. 비록 금속 보호막은 외부 충격 등에 안전하지만 이와 같은 문제점이 있어 보호막을 PVC 재질로 대체하였다. 그 결과 <그림 4-23>과 센서출력값이 직선형이 되어 자료처리장치에서 2점식 표정곡선을 작성하더라도 오차범위를 줄일 수 있었다



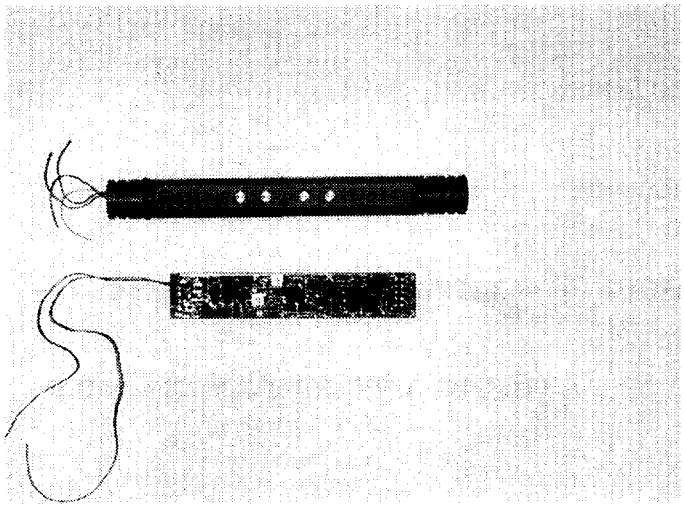
<그림 4-18> 전기전도도센서의 신호전달체계



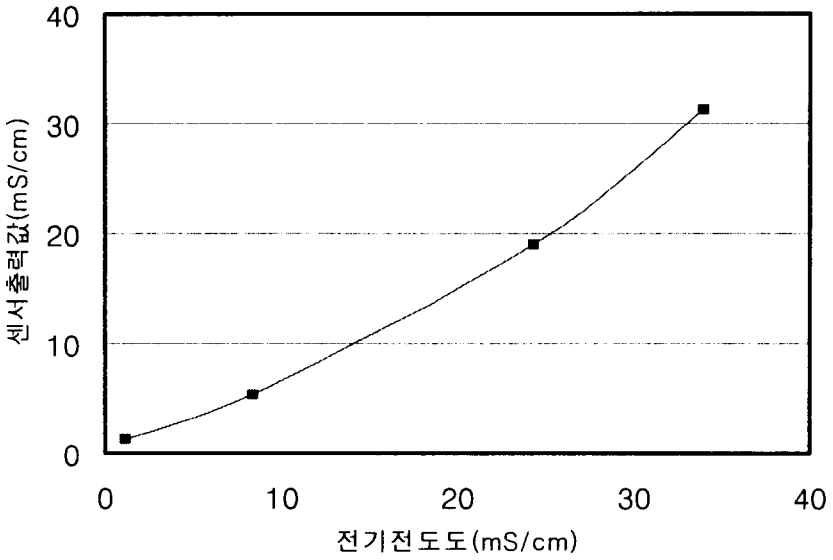
<그림 4-19> 4전극 전기전도도 센서 구성도



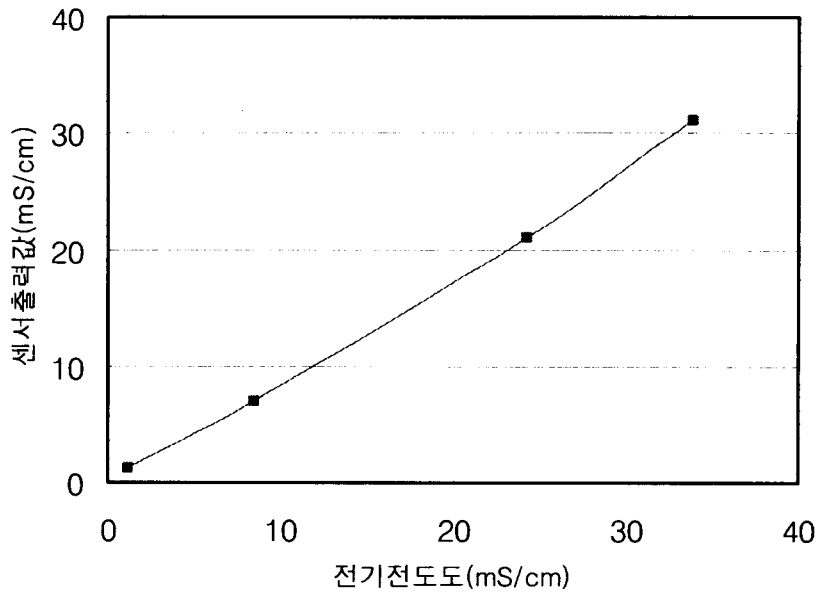
〈그림 4-20〉 4전극 전기전도도 센서 회로도



〈그림 4-21〉 4전극 전기전도도 전극과 회로기판



<그림 4-22> 금속보호막을 사용한 전기전도도 센서출력값



<그림 4-23> PVC보호막을 사용한 전기전도도 센서출력값

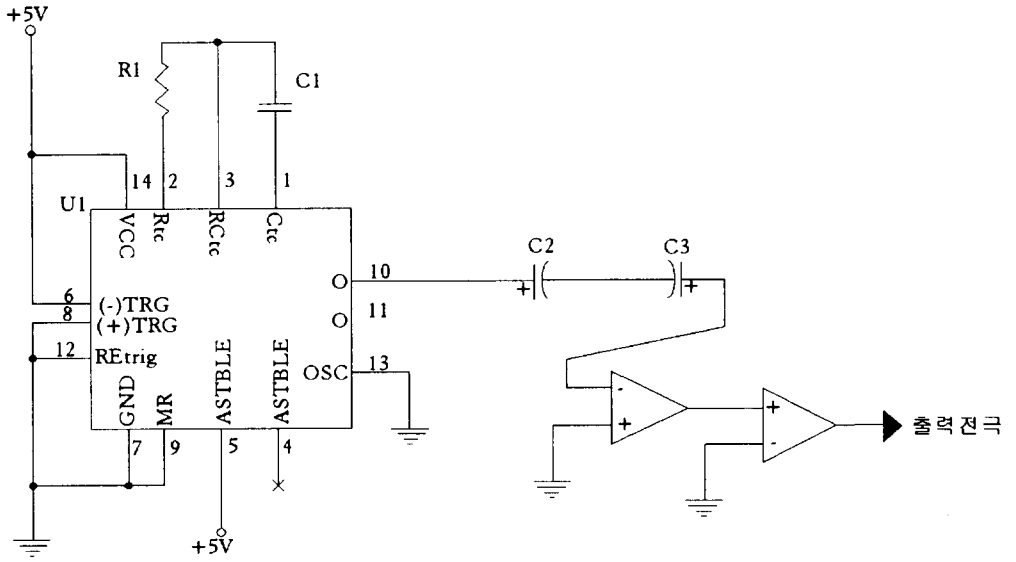
가. 정형파 신호

물속의 이온농도를 측정하는 전기전도도는 한 전극에서 1.2kHz 파형을 가진 정전압 출력한 출력전압과 다른 전극에서 해당되는 파형을 입력한 입력전압을 비교하여 측정하게 된다. 2전극 전기전도도 센서의 출력파형은 사인파이지만, 4전극 전기전도도센서는 정확도를 높히기 위해 <그림 4-24>와 같이 정형파를 만들어 출력신호로 사용하였다.

정형파 발생회로를 <그림 4-25>와 같이 구성하였다. 우선 저소비전력형 CD4047BM를 사용하여 0~5V 출력범위를 가진 1.2kHz 주파수를 발생한 후 극성 콘덴서 C2, C3를 사용하여 ±1.0V 출력범위를 가지는 정형파를 만들어 출력전극에 연결하였다. 이로써 노이즈 및 물 속에 포함되어 있는 이물질에 대하여 영향을 감쇄할 수 있었다.

(2전극 전기전도도센서) (4전극 전기전도도센서)

<그림 4-24> 전기전도도센서에 사용된 신호파형



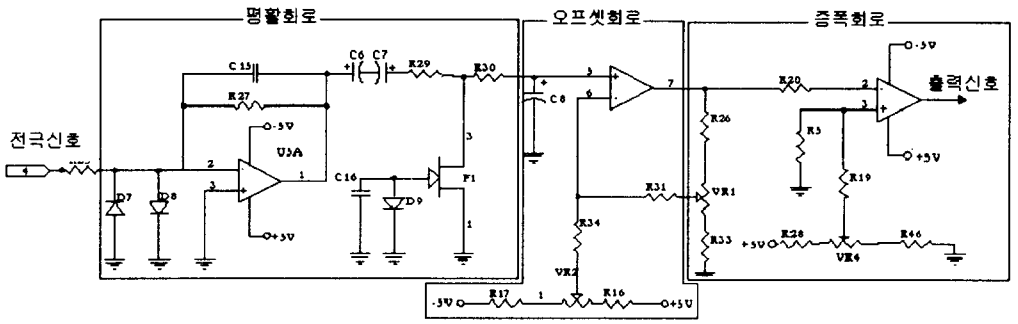
<그림 4-25> 정형파 발생회로도

나. 출력신호처리

전기전도도센서의 1전극에서 출력되는 신호는 ±1.0V수준에서 물의 이온농도에 따라 4전극에 입력되는 신호는 아주 미약한 100mV이하의 정형파 신호가 입력된다. 이 신호를 ADC에 접속하기 위해서 정형파를 직류성분으로 변환하였다. <그림 4-26>과 같이 1차적으로 100mV이하의 미약한 주파수 신호를 적분회로에 적용하여 콘덴서(C15)에 충방전으로 우선 (-)성분을 (+)성분에 반영하여 1차적으로 신호를 증폭하였다. 트랜지스터(F1)을 사용하여 (-)성분을 완전히 소멸시킨 후 콘덴서(C8)를 지나면서 주파수 신호를 평활시킨 후 오프셋회로와 증폭회로에 연결하였다.

오프셋회로에서는 기준값 즉 전기전도도 센서가 공기중에 노출되어 있을 때 0.0V가 출력되도록 가변저항(VR2)을 이용하여 조절하였다. 신호증폭은 비반전증폭회로로 구성하였다. 출력신호는 가변저항(VR1)을 사용하여 증폭율을 조정하였으며, 가변저항(VR4)을 사용하여 증폭후 오프셋을 조정하였다. 증폭신호의 출력신호를 -1.0V ~1.0V

1.0V로 하였다. 즉 -1.0V는 VR2로, +1.0V는 VR1으로 교정하였다. 다음 단계인 전류변환회로는 수위센서회로와 동일하게 구성하였다.

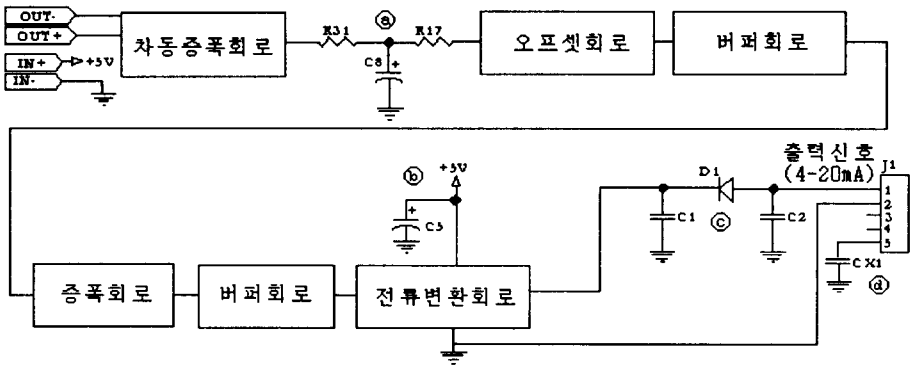


<그림 4-26> 전기전도도 센서 출력신호처리 회로도

다. 센서 노이즈 제거회로

수위센서와 마찬가지로 전기전도도 센서 또한 외부 전기적신호에 영향을 받는다. 외부 전기적신호에 대한 노이즈 대책으로 <그림 4-27>과 같이 신호처리과정에서 노이즈가 발생할 수 있는 지점이나 원인이 될 수 있는 지점에 노이즈 제거회로를 설치하였다.

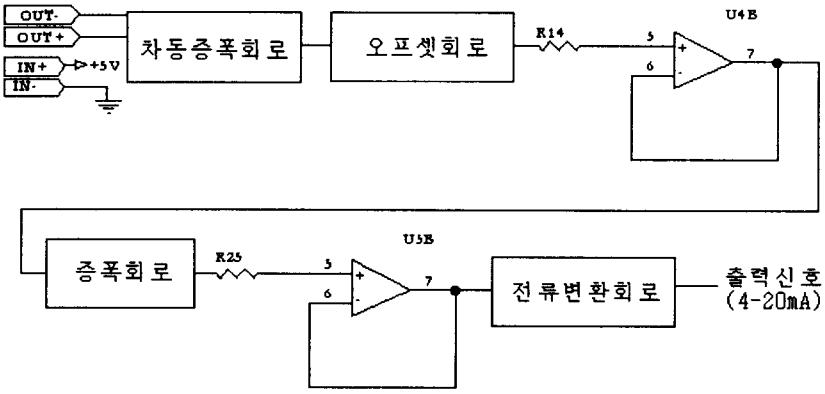
OP회로 내부에서, 또는 외부에 접속한 저항에 의해 노이즈가 발생된다. 증폭도가 변하면 노이즈 전압도 증폭되기 때문에 신호처리과정에 노이즈가 발생될 소지가 있거나, 원인이 될 수 있는 지점에 노이즈 제거를 하였다. ⓐ지점은 전압을 전류로 변환시키는 전류변환부에 가해지는 전원에 대한 노이즈를 제거하는 부분이다. ⓑ지점은 4~20mA로 출력되는 최종출력신호의 노이즈를 제거하는 부분이다. ⓒ지점은 전기전도도센서 케이스의 노이즈를 제거하는 지점이다. 이는 케이스를 통하여 들어오는 노이즈를 사전에 방지할 수 있다.



<그림 4-27> 전기전도도센서 회로내 노이즈 제거부분

라. 버퍼회로

전기전도도 센서 회로에서 상호 신호간섭이 있을 수 있는 증폭회로와 전류변환회로 사이에서 전기적으로 Isolation역할을 할 수 있는 일종의 버퍼회로를 <그림 4-28>과 같이 추가 설치하였다.



<그림 4-28> 전기전도도센서 회로내 버퍼회로

제 3 절 온도센서

1. 온도센서의 종류

온도센서에는 크게 측정 대상물과 열적으로 접촉시켜 양자의 온도를 같게 하여 대상물의 온도를 측정하는 접촉식과, 센서와 대상물을 공간적으로 근접시키지 않고 열방사를 이용하여 온도를 측정하는 비접촉식이 있다. 대부분 온도는 접촉식 센서를 사용하고 있다. 온도센서의 바람직한 재료로는 압력, 전기장, 자기장 등 다른 상태량에는 독립적이며, 온도에만 의존성이 있어야 하기 때문에 그 종류는 제한되어 있다. 그 밖에 감도 또는 정밀도, 안정성, 선형특성, 응답속도, 재현성, 열용량, 크기 등이 센서로서의 적합성을 결정하는 요소가 된다.

일반적인 온도센서는 열전대(thermocouple), 측온 저항체(RTD), 써미스터(thermistor)가 있다. 온도센서로 가장 많이 응용되고 있는 것은 측온저항체로서 이는 저항온도계수가 크고 온도-저항 특성에 직선적이며, 열적, 화학적, 기계적으로 안정하므로 교정 및 수명에 대하여 매우 안전하다. 따라서 별도의 교정을 요구하지 않는다. 그리고 온도센서가 별도로 구성되어 있는 것보다 전기전도도와 pH가 온도에 대하여 민감한 변화를 가지고 오기 때문에 전기전도도센서 또는 pH센서 내에 보상회로로 첨가되어 있다. 그러나 센서 주위에 이물질, 퇴적층 및 유기물질로 말미암아 측정오차가 발생될 소지가 있으므로 일정한 기간 간격으로 청결을 유지할 필요가 있다.

2. 온도센서 제작

온도센서는 다른 센서와 달리 온도변화에 대한 민감한 물리적 성질을 나타내는 측온저항체를 사용하였다. 측온저항체는 저항 온도계수가 크며, 온도-저항 특성이 직선적이며, 열적, 화학적, 기계적으로 안정하고, 경시변화가 적다.

지하수 관측기의 온도센서의 역할은 지하수 자체의 온도를 측정할 뿐만 아니라 전기전도도의 온도에 따른 값의 보상회로에 적용된다. 전기전도도는 약 1℃ 온도차이가 있을 때 전기전도도의 값은 2%정도 변화가 있다. 따라서 전기전도도 값의 정확도를

확보하기 위해서는 우선 온도를 정확하게 측정해야 한다.

전기적으로 전기전도도와 온도센서를 연결할 경우 전기전도도값의 온도보상 방법은 전기적으로 보상할 수 있게 된다. 이 방법은 소프트웨어적으로는 간단할 수 있지만, 전기전도도값은 항상 온도값에 종속하게 되므로, 유지관리에서 온도값의 보상이 항상 필요하게 되는 단점이 있다.

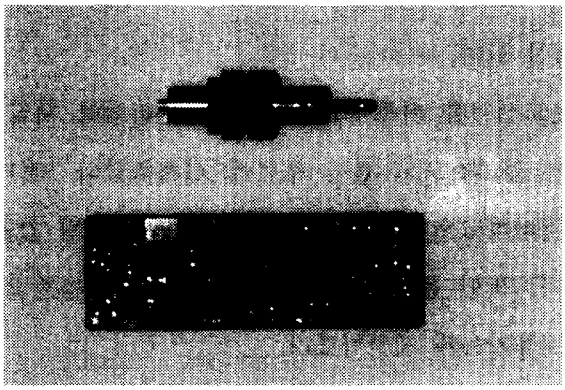
본 연구에서는 온도센서를 회로적으로 완전히 분리하여 제작하였다. 온도센서는 <그림 4-29>와 같이 전기전도도 전극 부근에 설치하였다. 전기전도도의 온도보상은 소프트웨어적으로 처리하였으며, 전기전도도는 온도 보상된 값과 보상되지 않은 값을 구할 수 있게 하였다. 또한 온도센서가 별도의 기판을 사용함으로써 수위센서와 연결하거나, 단독으로 사용가능케 제작하였다.

<그림 4-30>은 Pt100을 사용한 온도신호처리 회로도를 나타낸 것이다. Pt100의 저항치가 온도에 대하여 비직선식의 관계가 있기 때문에 선형화할 수 있는 계측회로를 구성하였다. 선형화 방법으로 계측온도 범위(-20~50℃)내에서 최저값, 최대값 및 중앙값을 기준으로 브리지 회로를 구성하였다. 온도회로에서 VR1은 최저값 조절용으로 사용하였으며, VR2는 최대값 조절용으로 사용하였다.

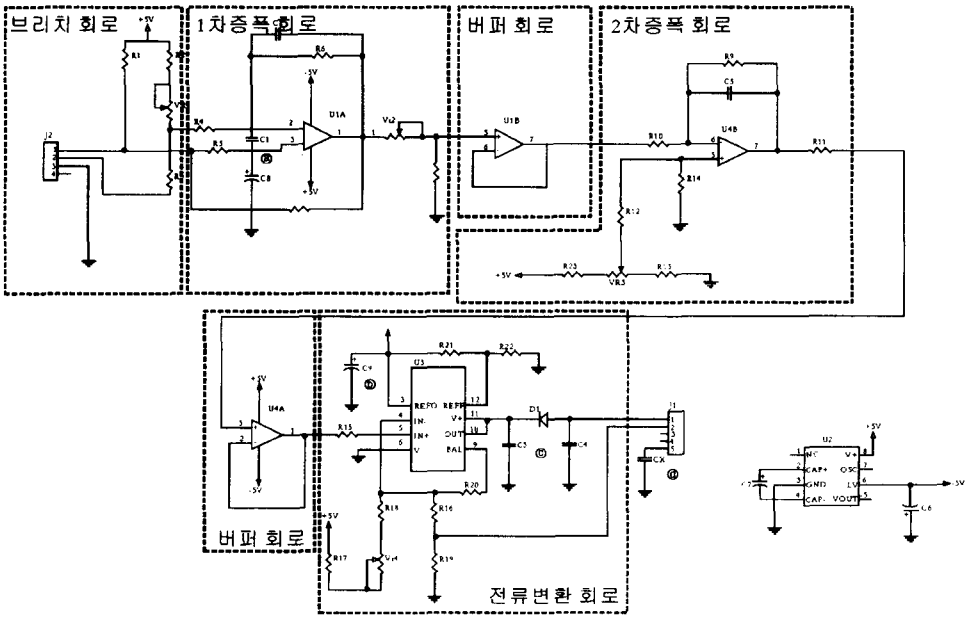
온도신호처리회로 내부, 또는 외부에 접속한 저항에 의해 노이즈가 발생된다. 증폭도가 변하면 노이즈 전압도 증폭되기 때문에 수위센서, 전기전도도회로와 마찬가지로 신호처리과정에 노이즈가 발생될 소지가 있거나, 원인이 될 수 있는 지점에 노이즈를 제거 하였다. ⓐ지점은 pt100에서 출력된 신호의 노이즈가 다음 회로인 증폭회로, 전류변환회로 등에 영향을 미치지 않도록 사전에 제거하는 부분이다. ⓑ지점은 수위, 전기전도도 센서와 마찬가지로 전압을 전류로 변환시키는 전류변환부에 가해지는 전원에 대한 노이즈를 제거하는 부분이다. ⓒ지점은 4~20mA로 출력되는 최종출력 신호의 노이즈를 제거하는 부분이다. ⓓ지점은 케이스의 노이즈를 제거하는 지점이다. 이는 케이스를 통하여 들어오는 노이즈를 사전에 방지할 수 있다.

온도센서회로는 각 단계별 특성을 가지고 있으므로 전후 신호처리단계에서 영향을

미칠 수 있는 지점에 대하여 1차 증폭회로와 2차 증폭회로, 2차 증폭회로와 전류변환 회로 사이에 버퍼회로를 설치하였다.



〈그림 4-29〉 온도센서와 회로기판



〈그림 4-30〉 온도센서 회로도

제 4 절 자료처리장치

자료처리장치는 지하수 관정의 외부에 설치되어 있으며, 지하에 설치되어 있는 수위, 전기전도도, 온도를 측정하거나 통신을 위한 모뎀제어, 자료저장 및 표시 등 지하수 자동관측기에서 가장 중요한 부분이다. 따라서 기온이나 습도 변화에 안정적으로 운영되어야 할 뿐만 아니라 사용자 측면에서 쉽게 다룰 수 있도록 제작되어야 한다.

자료처리장치의 기본 기능은 측정자료의 입력, 저장, 통신, 관측시스템의 전반적인 제어, 센서보정 및 이상여부 판단 등이 있다. 특히 지하수 관측은 수일 내지 수개월 이상 설치되어 있는 경우가 많다. 측정된 자료는 원격전송방식으로 호스트 컴퓨터에 전송되므로 자료처리장치내에 저장할 필요는 없지만 통신장애 등으로 인하여 전송할 수 있는 경우를 대비하여 자료처리장치내 메모리에 저장되어 있어야 한다. 따라서 별도의 충분한 메모리가 확보되어야 한다.

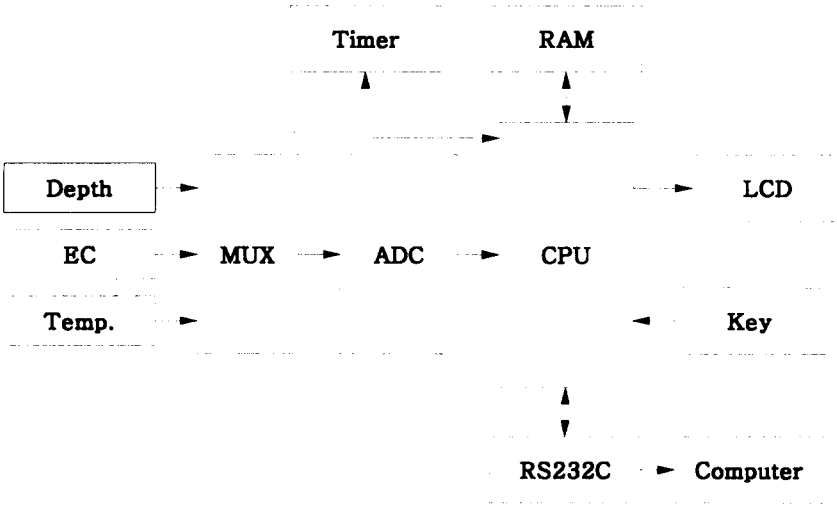
또한 지하수 관측망에 설치되는 관측기는 현장에 설치되어 있는 관계로 소비전력이 아주 중요한 요소가 된다. 소비전력이 클 경우 고용량의 밧데리를 설치해야 할 뿐만 아니라 자주 교체되어야 하는 불편한 점이 있다. 통신용으로 사용되는 모뎀의 소비전력이 크기 때문에 AC전원, Solar Cell 또는 외부 밧데리를 사용해야 한다. 만약 현장 여건상 AC전원이 없을 경우 Solar Cell을 설치해야 하는데 가격이 대당 300~400만원이므로 설치가 어려운 지역은 도리어 외부 밧데리를 사용해야 한다. 외부 밧데리를 사용할 경우 사용기간이 1개월이내이므로 자주 교체해야 할 경우가 발생한다.

이러한 점을 감안하여 본 연구에서 개발한 자료처리장치는 기본적인 기능 외에 저소비전력과 메모리 확보 중심으로 하였다.

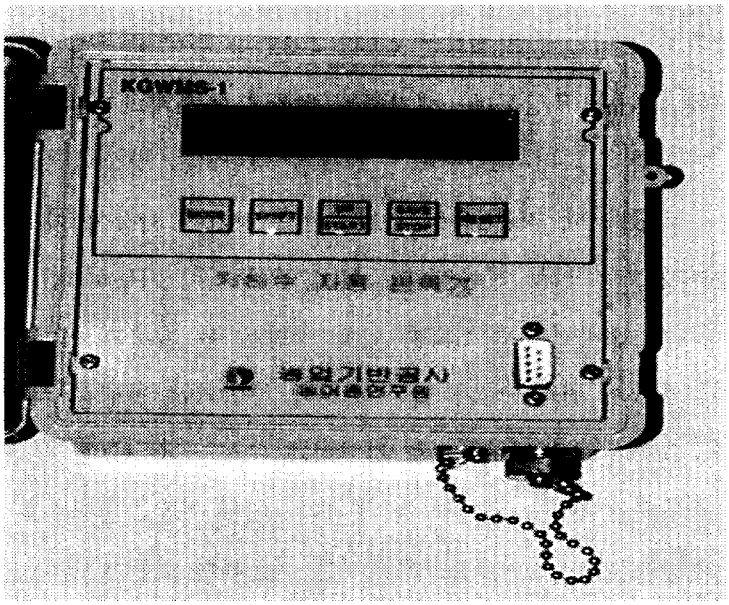
자료처리장치는 <그림 4-31>과 같이 지하수 자동관측기를 완전히 자동 제어할 수 있는 기능을 갖도록 하기 위하여 ROM, RAM, 마이크로프로세서, I/O, ADC, Timer 등 IC chip 들을 사용하였다. 자료처리장치의 내부 모습은 <그림 4-32>와 같으며, 외부는 <그림 4-33>과 같다. 자료처리장치의 크기는 120(W)×130(D)×90(H)mm이며 내부

에 12V, 1.9Ah 밧데리를 내장하였다. 상세한 회로도는 부록4에 실었다. 자료처리장치의 기본 기능은 다음과 같다.

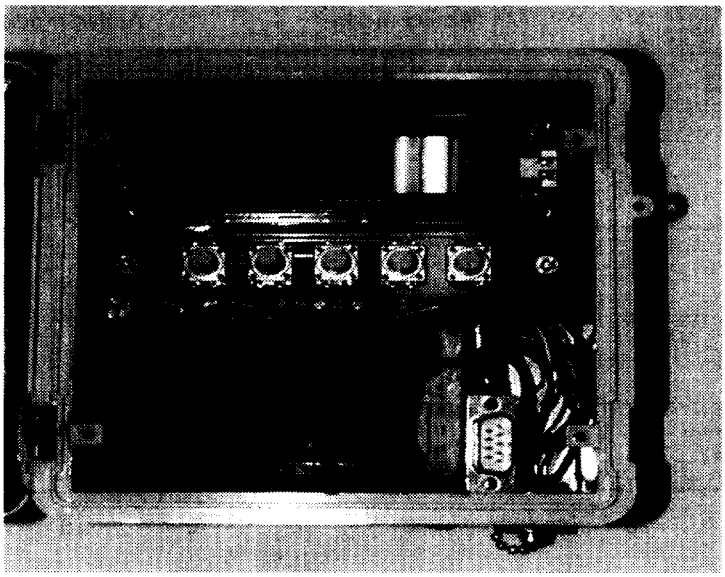
- 소비전력 : 동작시 50mA, Sleep mode시 0.050mA 이하
- 사용자 프로그램을 작성하여 기록할 수 있는 64K 메모리 확보
- 연속적으로 측정한 계측자료를 저장 및 분석할 수 있는 64K 메모리 확보
- 15 bit ADC 4 채널 확보
- 통신포트(시리얼 포트) 채널 확보
- RTC (real time controller) 내장
- 2 × 20 LCD 보드 제어 기능
- 버튼 스위치로 제어변수 설정기능
- 전원 : DC 9~12V



<그림 4-31> 자료처리장치 구성도



〈그림 4-32〉 자료처리장치 외부

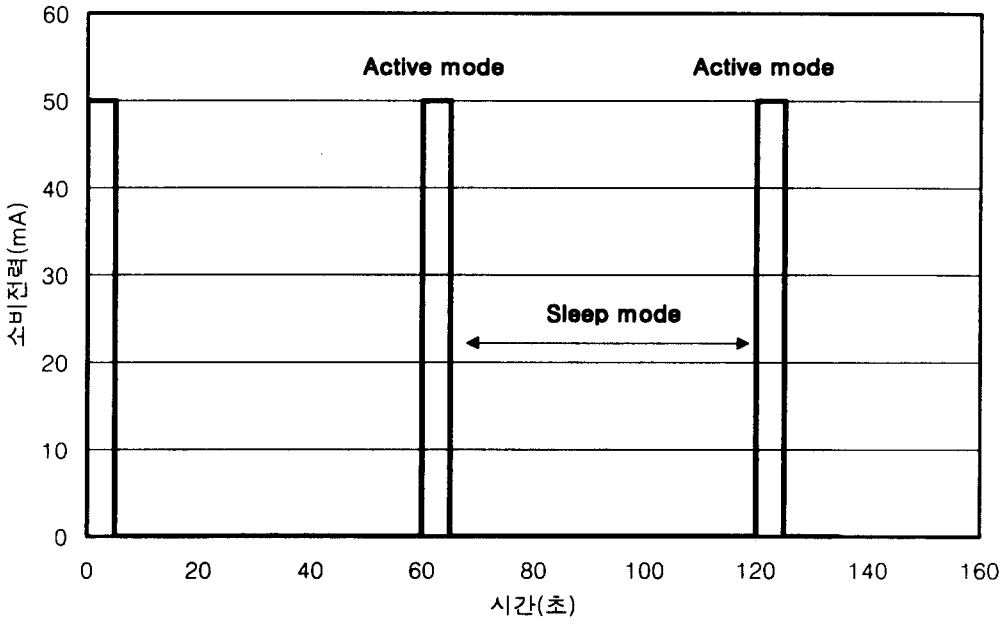


〈그림 4-33〉 자료처리장치 내부

1. 저소비기능

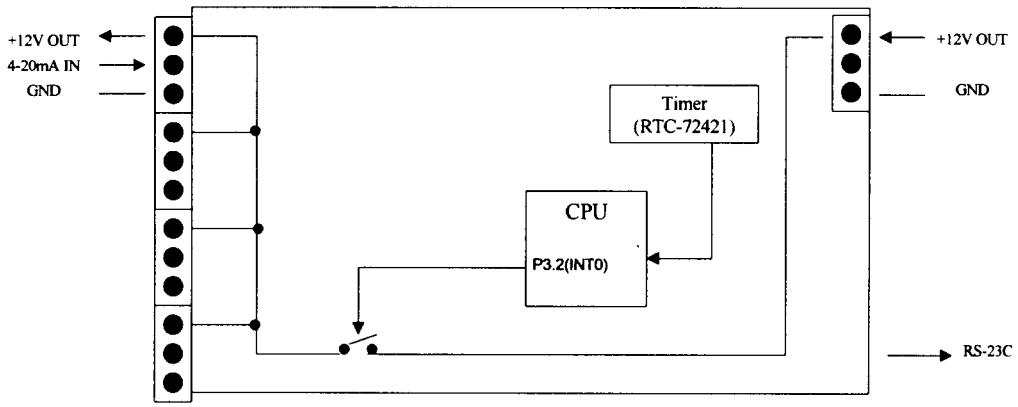
자료처리장치의 소비전력 최소화는 <그림 4-33>과 같이 측정할 동안 센서뿐만 아니라 모든 장치에 전원을 투입하며, 대기하는 동안은 자료처리장치 동작에 필요한 부분에만 전원을 투입하고 나머지 부분에는 전원을 차단하는 방식이 있다. 이와 같은 방법으로 동작이 되었을 때 적은 용량의 밧데리로 장기간 사용이 가능하며 최소한 6개월 동안은 솔라셀 또는 외부 전원장치 등이 없이 동작이 가능하다.

자료처리장치의 소비전력 최소화를 위한 방법은 두 가지가 있다. 첫 번째는 지하수 자동관측기 중에서 전력소비가 많은 센서에 투입되는 전원을 차단하는 방법이다. 이 방법은 프로그램과 하드웨어적으로 간단하게 구성할 수 있다. 두 번째는 타이머외 모든 부품에 투입되는 전원을 차단하는 것이다. 이는 프로그램적으로 복잡할 뿐만 아니라 이러한 기능을 가진 마이크로프로세서를 사용해야 하는 단점이 있다.



<그림 4-34> 자료처리장치의 소비전력 변화

<그림 4-35>는 첫 번째 방법으로 소비전력을 최소화시키는 방법이다. 외부타이머를 사용하여 측정 시 마이크로프로세서를 Wake Up 시키고, 마이크로프로세서는 센서에 전원을 공급하기 위하여 Relay를 동작시킨다. 측정 완료 후 마이크로프로세서는 센서 전원을 차단하고 스스로 대기상태로 전환된다. 비록 마이크로프로세서는 대기상태로 되지만 자료처리장치내 부품들의 전원은 연결된 상태로 유지된다. 따라서 대기상태가 되더라도 소비전력은 약 5mA 수준이 된다.



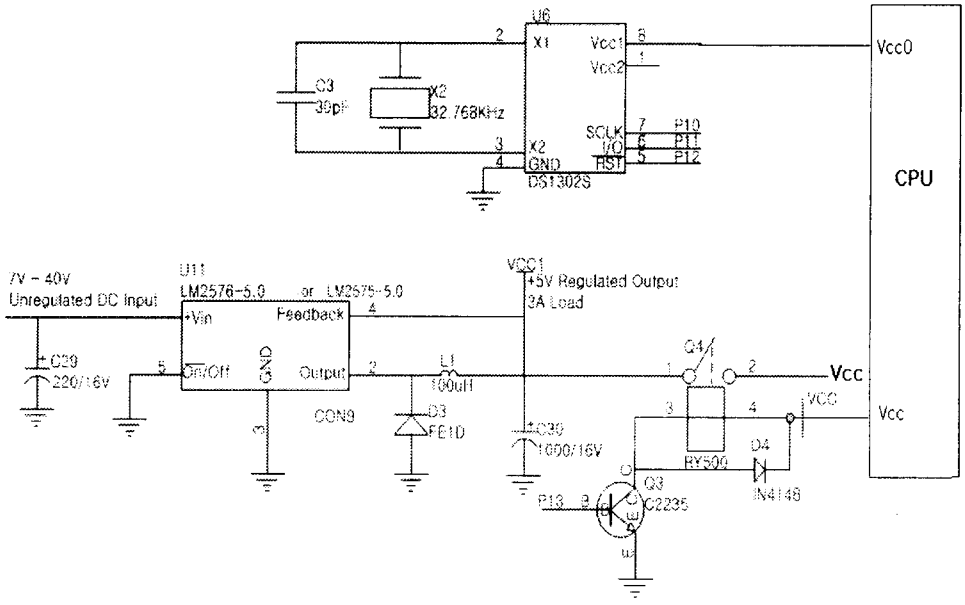
<그림 4-35> 자료처리장치 전원제어

5mA 전력소비는 12V, 1.9Ah 밧데리의 기준으로 봤을 때 상당한 소비전력에 해당된다. 즉 센서를 연결하지 않더라도 5mA 소비전력은 2개월 정도 사용할 수 있는 전력에 해당된다.

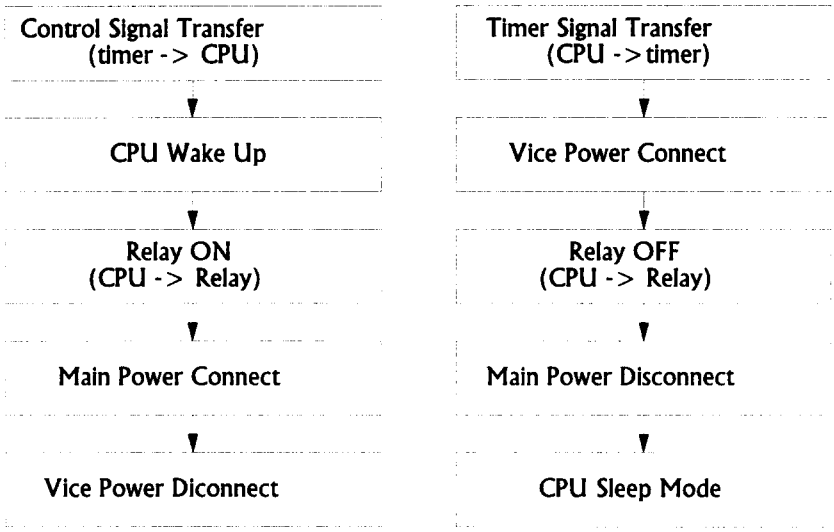
대기상태에서 이 소비전력도 절약하기 위하여 본 연구에서는 두 번째 방식을 사용하였다. 즉 자료처리장치의 저소비전력 기능은 외부 타이머를 사용하여 마이크로프로세서를 제어하는 방식으로 <그림 4-36>과 같이 구성하였다. 대기상태시 자료처리장치에 공급되는 전원은 차단되고 타이머만 동작하게 된다. 측정시에는 타이머에 의하여

마이크로프로세서가 Wake Up되어 측정이 이루어진다. 측정 완료 후 다시 전원을 차단하고 대기상태로 전환된다.

이와 같은 저소비전력을 위한 제어순서는 <그림 4-37>과 같다. 측정이 완료된 후 대기상태로 전환시점에 도달하였을 때 마이크로프로세서는 Vcc(13핀) 단자에 릴레이를 OFF시키는 신호를 발생하여 자료처리장치의 주전원을 차단한다. 마이크로프로세서는 자동적으로 주전원 대신에 3.0V 수은밧데리로 전환되어 동작하게 된다. 또한 마이크로프로세서는 3.0V 수은밧데리로 공급된 전원을 타이머(DS13029)에 공급하여 타이머를 작동하게 된다. 동시에 마이크로프로세서는 대기상태로 전환하게 된다.



<그림 4-36> 저소비전력을 위한 자료처리장치내 회로도



<그림 4-37> 저소비전력 순서도

센서가 연결되어 있지 않을 경우 자료처리장치 자체의 소비전력은 65mA이며 동작시 50μA 정도 전력이 소비된다. 센서가 연결되어 있을 경우 센서 자체소비전력이 포함되므로 최대소비전력 60mA(3×20mA)에 자체 소비전력 65mA를 포함하여 약 125mA가 소비된다. 예를 들면 측정간격이 1분이고 측정시간은 5초일 때 5초 동안은 125mA가 소모되고 55초 동안 0.05mA만 소모하게 된다. 이와 같은 방식으로 관측기가 작동할 경우 (표 4-3)에서 보듯이 12V, 1.9Ah 소형 밧데리로 6개월 장기간 사용이 가능하다.

(표 4-3) 자료처리장치의 밧데리 사용가능기간

측정간격	평균 소비전력	사용가능기간
1분	4.17mA	20일
1시간	0.05mA	1,140일
사용밧데리 : 12V , 1.9Ah		

2. 데이터 저장방식

본 연구에서는 메모리 절약을 위하여 측정 자료를 숫자보다 16진수 방식으로 저장하도록 하였다. (표 4-4)는 숫자로 저장하는 경우와 16진수로 저장할 경우 필요한 메모리 용량 비교이다. 날짜와 시간은 정수로 저장할 경우 한 숫자에 대하여 2바이트가 필요하므로 총 12바이트가 필요하며. 측정 자료를 실수로 저장할 경우 한 값에 대하여 4바이트가 필요하다. 따라서 1회분 측정 자료를 저장하는데 필요한 바이트수는 26이다. 그러나 16진수로 저장할 경우 날짜정보, 시간정보 및 측정 자료를 합하여 13바이트만 필요하다. 따라서 적은 메모리 용량으로 충분한 측정 자료를 저장할 수 있다.

자료처리장치의 전체 메모리 용량은 128K이며, 이중 64K는 프로그램용으로 사용하였으며, 나머지 64K는 자료저장용으로 사용하였다. 자료 저장용 64K 메모리에 이와 같은 방식을 적용함으로써 5,000회 분량의 자료저장이 가능하였다. 메모리에 저장되는 포맷은 <그림 4-38>과 같다.

(표 4-4) 측정 자료를 저장하는데 필요한 바이트수

구분	날짜정보		시간정보			수위	전기전도도	온도	예비	총바이트수
	월	일	시	분	초					
숫자저장	2	2	2	2	2	4	4	4	4	26
16진수저장	1	1	1	1	1	2	2	2	2	13

STX (0x02)	P	E	S	D	A	T	A	예약어	
02	50	45	53	44	41	54	41	FF	FF

날짜 및 시간 정보					Depth		EC		온도	
6월	28일	23시	52분	12초	8.23m		1291uS/cm		110.2℃	
06	1C	17	34	0C	03	37	05	0B	04	4E

예약어		Check SUM				ETX (0x03)
FF	FE	00	00	00	00	03

㉠ 월 : char형 (예 : 0x06 --> 6월)
㉡ 일 : char형 (예 : 0x1C --> 28일)
㉢ 시 : char형 (예 : 0x17 --> 23시)
㉣ 분 : char형 (예 : 0x34 --> 52분)
㉤ 초 : char형 (예 : 0x0C --> 12초)
㉥ Depth : int형 (예 : 0x0337 --> 8.23m)
㉦ EC : int형 (예 : 0x050B --> 1291uS/cm)
㉧ Temp(온도) : int형 (예 : 0x044E --> 110.2℃)

<그림 4-38> 측정 자료가 메모리에 저장되는 방식

3. 센서신호처리

정확도(Accuracy)는 계측기로부터 측정값이 얼마나 참값에 가까운가를 나타내는데 사용되는 용어이다. 계측기의 정확도는 다음 식으로 정의되는 오차의 백분율로서 측정값의 %오차이다.

$$\text{정확도} = (\text{측정값} - \text{참값}) / \text{참값} \times 100 \quad (9)$$

예로써 100m까지 측정 가능한 수위센서의 정확도가 ±1%FS(full scale)라면 이 수위센서로 측정한 수위와 실제 수위와의 차이는 항상 1m이내임을 의미한다.

정밀도(Precision)는 동일한 조건에서 반복하여 얻은 측정값이 평균값에 얼마나 가까운 것인가를 뜻하는데 사용되는 용어로써 다음과 같이 정의된다.

$$\text{정밀도} = (\text{최대측정값} - \text{평균측정값}) / \text{평균측정값} \times 100 \qquad (10)$$

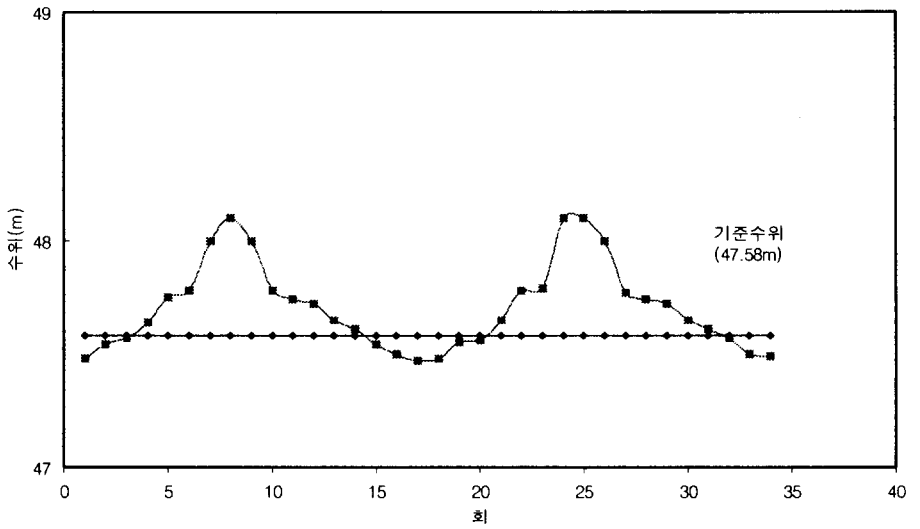
예로서 100m 수위를 수위계로 5회 측정한 값이 104, 103, 105, 103, 105m라면 평균값은 104m, 최대편차는 ±1m이다. 따라서 이 계측기의 정밀도는 약 ±1%로 산출된다. 센서의 정확도는 교정에 의하여 향상될 수 있으나 그것의 정밀도를 초과할 수는 없다.

이와 같이 정밀도는 센서의 중요한 사양이다. 특히 지하수 관측망으로 사용되고 있는 관정은 지하수 관측 전용 관정이라기 보다는 수중펌프가 설치되어 있는 사용관정에 사용하는 경우가 대부분이다. 수위센서 내부에는 미약한 전기적인 신호로 동작되고 있기 때문에 모터에 의한 노이즈에 민감한 영향을 받는다.

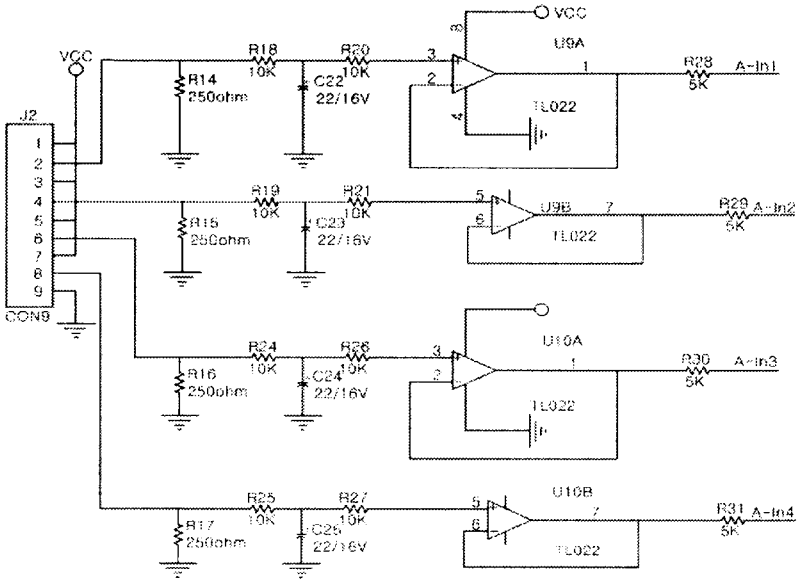
본 연구에서 제작된 수위센서를 관정에 설치하여 자료처리장치내 노이즈 제거회로를 거치지 않고 직접 ADC에 연결하였을 때 모터 가동시 측정자료는 <그림 4-40>과 같이 노이즈 영향을 받고 있음을 확인할 수 있다. 수위 47.58m에서 측정값의 변동은 47.48~47.81m 로 정밀도는 0.34%가 된다. 이 노이즈는 센서신호가 케이블로 전달되는 과정에서 전기적인 영향을 받은 것이라 생각된다. 따라서 자료처리장치내에서 노이즈 제거회로가 필요함을 알 수 있다.

센서와 자료처리장치의 연결은 <그림 4-40>과 같이 구성하였다. 센서로부터 출력되는 신호는 4~20mA 전류신호를 저항 250Ω을 사용하여 1.0~5.0V 신호로 변환하였다. 변환되는 과정에서 TL022를 사용하여 버퍼회로를 추가하였으며, 콘덴서를 센서에 투입되는 전원선과 신호출력선에 연결하였으며, 전압으로 변환된 후에도 콘덴서(C28,

C29, C30, C31)를 설치하여 추가적으로 노이즈를 제거하였다.



<그림 4-39> 모터에 의한 수위센서의 노이즈



<그림 4-40> 자료처리장치내 전류-전압신호 변환회로도

4. 아날로그 분해능력

센서로부터 전달된 신호는 1.0~5.0V로 변환한 후 저항(R32, R37)을 사용하여 전압강하를 시켰다. 또한 4채널 멀티플렉스(4052)로 동시에 4개 채널을 선택적으로 측정하도록 하였다. ADC(TC835)에 입력되는 최종 전압범위는 0.0~1.0V이다.

센서의 전압범위는 0.0~1.0V로 수위측정범위(0~10,000)에 비하여 다소 미약한 신호에 해당되므로 1mV이하의 변화에 대해서도 계측이 가능해야 한다. 아무리 센서 측정이 정확하더라도 디지털로 변환되는 과정에서 정확하게 분해하지 못할 경우 센서의 정밀도 또한 떨어지게 된다.

만약 12비트 ADC를 사용할 경우 (표 4-5)와 같이 각 측정항목에 대하여 분해능과 정확도가 결정된다. 수위에서는 디지털값 '1'변화는 수위변화 3cm에 해당된다. 센서회로에서 아날로그 출력신호가 아무리 정확하더라도 디지털값으로 변환시 '1'정도는 항상 일어날 수 있다.

(표 4-5) 12비트 ADC의 분해능

측정항목	측정범위(V)	센서신호범위		측정값/디지탈값 (D)
		아날로그 신호(A)	디지탈신호(D)	
수위	0~10,000cm	4~20mA	820~4096	3.05cm
전기전도도	0~3,000 μS/cm	4~20mA	820~4096	0.91 μS/cm
온도	-10~50℃	4~20mA	820~4096	0.018℃

12비트 ADC는 수위측정에 문제시될 수 있으므로 그 이상의 분해능을 가진 ADC를 사

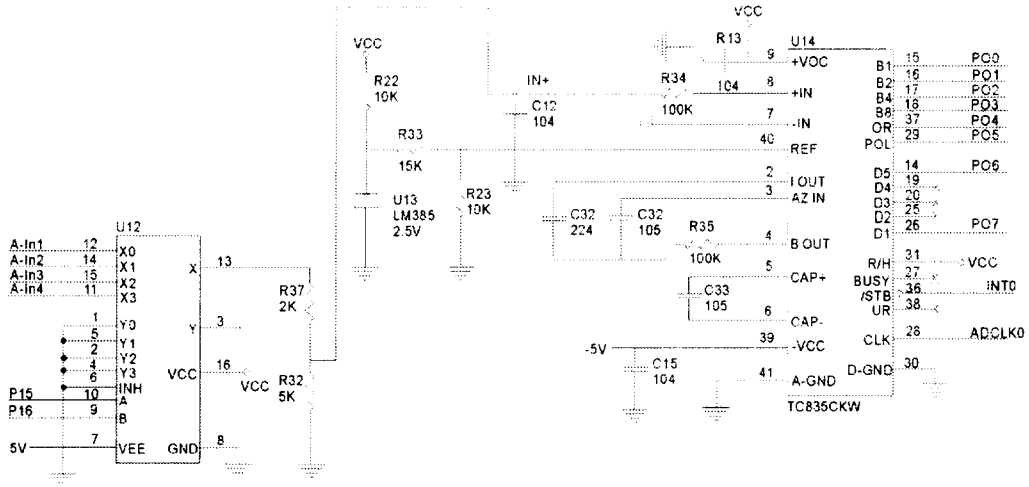
용하여 아날로그 신호 분해능력을 높일 필요가 있다.

본 자료처리장치에서는 15bitADC인 TC835를 사용 <그림 4-41>과 같이 구성하였다. TC825의 입력전원 범위는 -2.0 ~ +2.0V가 된다. 센서의 출력신호는 0.0~1.0V로 하였으므로 기준전압 V-는 0V로 V+는 1.0V로 하였다. TC825 특성상 입력전압이 0V일 때 디지털값 '0'이 되며, 1.0V일 때 +19,999가 된다. 따라서 센서범위의 최소값은 0이며, 최대값은 +19,999가 된다.

(표 4-6)는 15bit ADC(TC835)를 사용하였을 때 각 측정 항목에 대한 센서출력값과 디지털값의 관계이다. 15Bbit ADC를 사용한 결과 수위의 경우 디지털값 '1'변화는 수위변화 0.50cm에 해당된다. 저농도 전기전도도는 디지털 '1'변화는 0.25㎲/cm, 고농도 전기전도도에서는 2.50㎲/cm에 해당된다. ADC분해능에서 발생될 수 있는 오차범위는 ±0.005%로 센서자체에서 발생될 수 있는 오차범위보다 상당히 적은 값이므로 디지털 변환시 발생될 수 있는 오차는 무시할 수 있다.

(표 4-6) 15비트 ADC의 분해능

측정항목	측정범위(V)	센서신호범위		측정값/디지탈값 (D)
		아날로그신호(A)	디지탈신호(D)	
수위	0~100,00cm	4~20mA	0~19,999	0.50cm
전기전도도	0~5,000㎲/cm	4~20mA	0~19,999	0.25㎲/cm
	0~50,000㎲/cm	4~20mA	0~19,999	2.50㎲/cm
온도	-20~50℃	4~20mA	0~19,999	0.003℃



<그림 4-41> 자료처리장치내 15bit ADC회로도

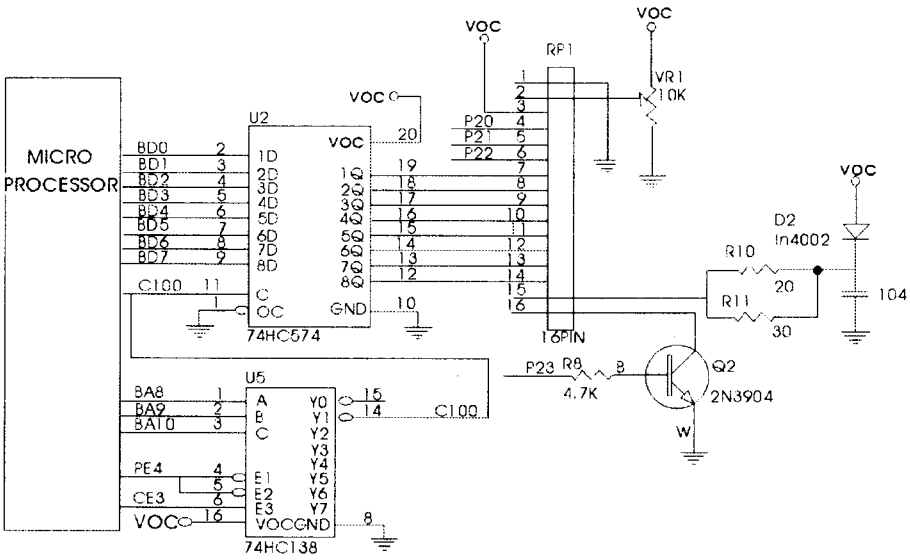
5. 표시부와 입력부

표시부는 자료처리장치의 동작상태, 측정 자료값 등을 표시하며, 입력부는 동작설정 및 수정 등 필요한 조치를 담당한 조작 키에 관련된 회로로서 키 조작 시 작동하는 부저회로를 추가하였다.

<그림 4-42>는 자료처리장치의 표시부와 관련된 회로도이다. 표시장은 20문자×2라인 Text mode LCD를 사용하였다. LCD의 밝기는 사용자 선택에 의하여 사용할 수 있도록 하였다. +5V를 저항 (R10)을 통해서 LCD의 전원(Vss)에 연결하여 LCD 라이트 제어용 키를 입력하게 되면 마이크로프로세서에서 키입력을 확인하여 출력포트(P2)에 High 신호를 보내게 된다. 이 신호는 다시 TR(2N3994)의 베이스에 전류를 흐르게 하면서 LCD 라이트가 작동하게 된다.

입력부는 push-button switch로 <그림 4-44>와 같이 구성하였으며, 마이크로프로세서 입력포트에 직접 연결하였다. 따라서 마이크로프로세서는 항상 키 입력 대기상태로 된다. 동작 또는 대기시라도 키 입력이 발생하였을 때 언제든지 필요한 기능을 수행할 수 있도록 프로그램을 작성하였다. 키 입력시 발생하는 chattering에 대하여

Schimit trigger 회로와 같은 별도의 회로를 설치하지 않고 소프트웨어에서 처리 하였다.

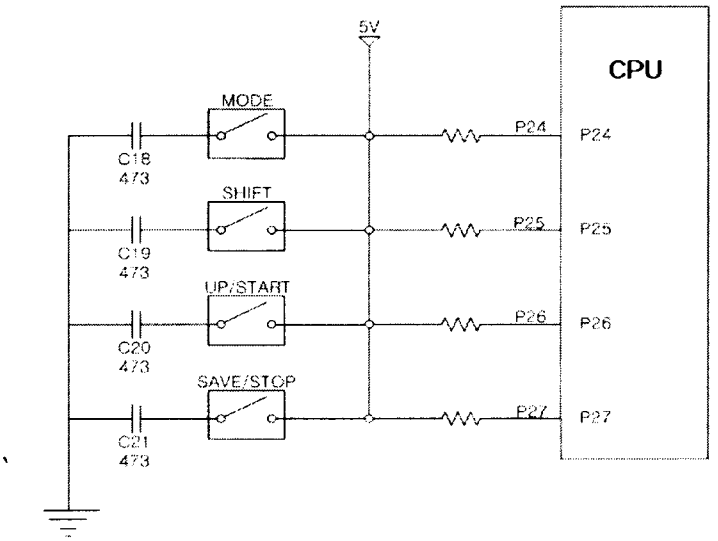


<그림 4-42> 자료처리장치내 LCD 회로도

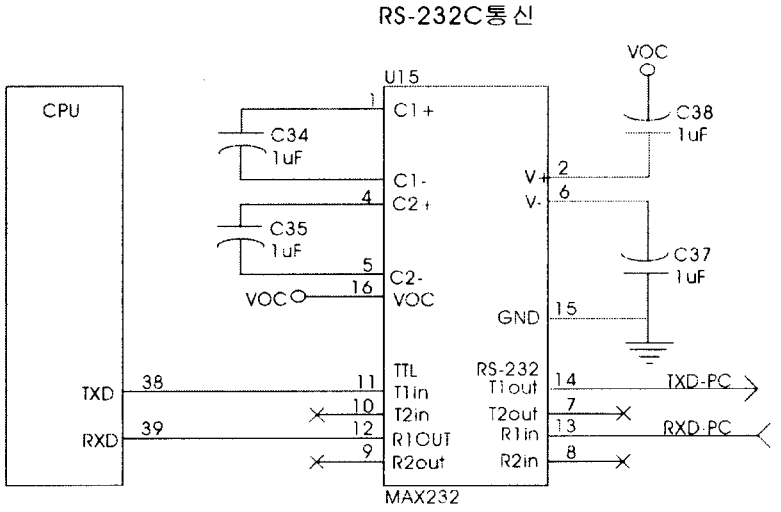
6. 통신회로

통신회로는 각 센서로부터 입력된 데이터를 호스트 컴퓨터에 일정한 시간 간격으로 연속적으로 보냄으로써 자료의 온라인화, 저장, 분석 등 중요한 수단이 될 수 있고, 운영프로그램을 자료처리장치 메모리에 전송할 수 있는 잇점을 제공하여 준다.

본 연구에 사용한 DS5002FP는 시리얼 통신포트를 내장하고 있지만 RS23C 통신규약에 적합한 전기적인 신호는 아니다. 원거리 전송 및 모뎀을 제어하기 위해서는 ± 12.0V 신호의 승압이 필요하다. 본 자료처리장치에서는 데이터 승압 전용칩으로는 Maxim사의 MAX232 IC를 사용하여 <그림 4-44>와 같이 통신회로도를 구성하였다. 데이터 전송속도를 9,600bps로 하였으며, 데이터는 8bit, stop bit 는 1 bit, parity는 none로 설정하였다.



<그림 4-43> 자료처리장치내부 키입력 회로도



<그림 4-44> 자료처리장치내부 시리얼통신 회로도

제 5 절 센서케이블

센서케이블은 센서가 설치되어 있는 프로브와 자료처리장치를 연결하는 선으로서 프로브의 설치 심도에 따라 필요한 케이블의 길이가 달라진다. 또한 관정 심도, 수위 뿐만 아니라 자료처리장치의 설치장소도 고려되어야 하므로 현장마다 케이블 길이는 다를 수 있다. 따라서 기본적인 측정범위(0~100m)에 여분의 길이를 감안하여 110m로 하였다,

케이블은 프로브의 최대 설치 심도인 100m를 기준으로 하여 이에 걸리는 장력을 견딜 수 있어야 하며, 교정주기(6개월) 이내에 케이블 길이의 변화가 지하수위 오차범위인 ±2cm를 만족시켜야 한다. 또한 충격에 강하고 온도에 의한 수축이나 팽창이 적으며 완전한 방수가 가능해야 한다. 특히 지하수 내에 용해되어 있는 화학물질과의 반응성이 전혀 없는 테프론 또는 폴리우레탄 수지로 피복되어 있어야 한다.

본 연구에서는 우선 시중에 쉽게 구할 수 있는 센서 전용 케이블을 사용하였다. 장시간 사용하지 않아 지하수와의 반응성은 확인할 수 없었지만 케이블 110m에 해당하는 중량은 21kg정도로 운반하기가 어렵고 내부에 철심이 없어 장기관측망으로 사용시 케이블이 늘어날 우려가 있었다.

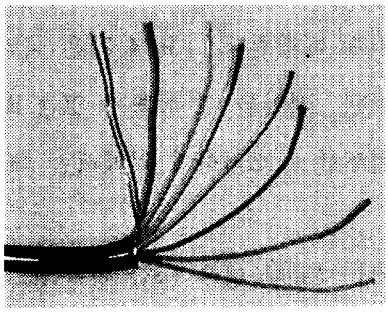
또한 지하수 관측기에 사용되는 센서케이블은 다른 케이블과는 달리 내부에 대기압 보정을 위한 튜브가 내장되어 있어야 한다. 사용 도중 케이블이 손상되었을 경우는 이 튜브 때문에 전체를 교환해야 한다. 따라서 케이블 유지보수 비용이 상당히 높으며, 외국 관측기 가격의 약 40%는 케이블 가격에 해당된다. (표 4-7)은 국내 도입 사용되고 있는 지하수 관측기의 케이블의 내역이다. 케이블 내 튜브가 없는 H사의 경우는 대기압 보정이 되지 않는 경우이다. 오차범위를 최소화하기 위해서는 대기압 보정을 할 수 있는 케이블을 선택해야 한다.

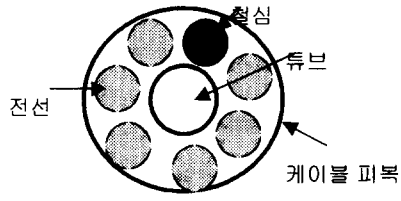
센서 전용 케이블을 외국으로부터 수입하여 사용할 경우 1m당 평균 20,000원으로 케이블 구입단가가 너무 높기 때문에 당초 연구계획에서 제시한 저가보급형 관측기 개발이 불가능하게 된다. 따라서 국내 전선 제작회사에 의뢰하여 지하수 관측기 전용

케이블을 제작하였다. <그림 4-45>는 제작한 센서케이블로서 내부에 6개 라인과 철선을 넣어 강력한 장력을 가지도록 하였다. <그림 4-46>과 같이 센서케이블을 장력시험을 한 결과 100kgf 까지 가능하였다. 센서 중량과 110m에 해당하는 케이블 중량이 7.5kg임을 감안할 때 장기간 사용하더라도 케이블이 늘어나거나 끊어지는 경우는 없을 것이라 판단된다. 그리고 센서피복에는 지하수내 용해되어 있는 화학물질과의 반응성이 없는 폴리우레탄으로 처리하였다. <그림 4-47>은 케이블 링에 감겨진 상태이며 링의 반경은 30cm로 운반하기에 용이하였다. 또한 센서와 케이블을 연결하는 커넥터를 보호하기 위하여 링 주위에 받침대를 설치하였으며, 커넥터의 핀을 보호하기 위하여 보호막을 제작, 설치하였다.

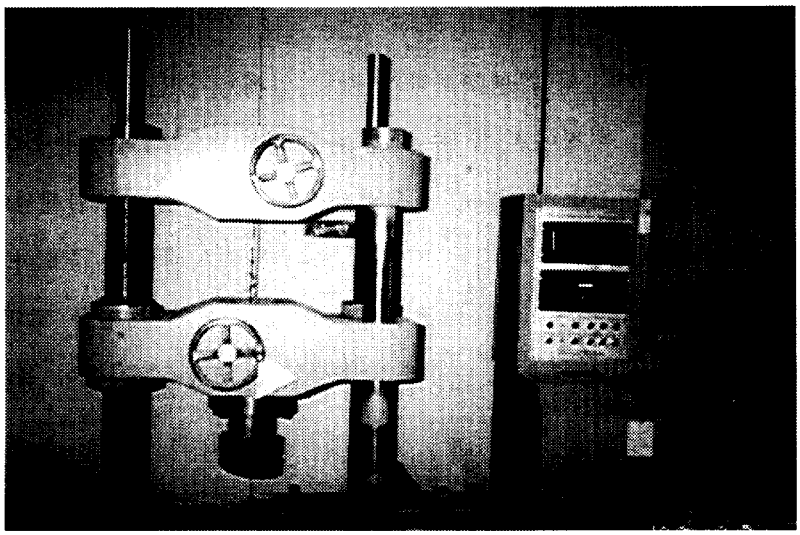
(표 4-7) 국내 사용중인 지하수 관측기의 센서케이블

제조회사	원산지	외경(mm)	전선수	튜브	가격 (원/m)	형태
H사	미국	10	8	없음	22,000	일체형
G사	호주	10	8	있음	18,000	일체형
I사	미국	6	2	있음	26,000	분리형

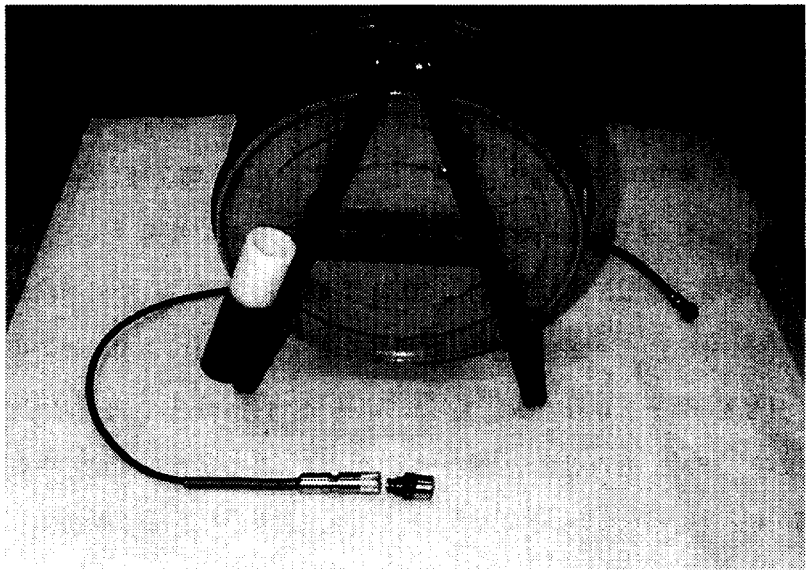




<그림 4-45> 센서케이블



〈그림 4-46〉 센서케이블 장력시험



〈그림 4-47〉 센서케이블링과 커넥트 받침대와 보호막

제 6 절 지하수 자동관측기 형태별 제작

지하수 자동관측기 센서의 유형조사결과 각각 일체형과 분리형의 장단점을 가지고 있기 때문에 사용목적, 현장여건 및 수질 등을 고려하여 선택되어야 한다. 예를 들면 장기관측망으로 사용하고자 할 경우 사후관리측면을 고려할 때 일체형이 유리하다. 양수 시험 및 지하수 개발과정에서는 지하수 수위가 일차적으로 중요하기 때문에 수위만을 정확하게 측정해야 한다. 이 경우 현장에서 사용자 측면을 고려한다면 모든 센서가 부착되어 있는 일체형보다 수위만을 측정할 수 있는 분리형이 유리할 것이다.

1. 분리형 지하수 자동관측기

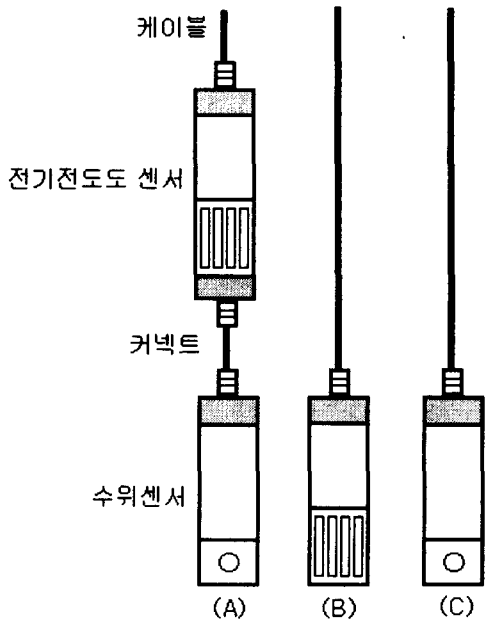
가격측면에서는 일체형이 유리하지만 양수시험과 같은 지하수 개발현장에서는 필요한 항목만을 측정할 수 있는 분리형이 유리하다. 그러나 장기관측망, 서울시 지하수 관측망, 해수침투조사용등 지하수 관측기는 일체형으로 되어 있다. 기존 관측망에 설치되어 있는 관측기의 수리 및 교체 횟수가 29회(50대, 3년동안)이상이 되어 수리기간동안 다른 측정항목도 측정할 수 없는 경우가 많다. 따라서 A/S 횟수와 자료의 연속성을 고려할 때 오히려 분리형보다 유리하다고는 볼 수 없다.

이와 같이 일체형과 분리형이 장단점을 가지고 있기 때문에 본 연구에서는 두 형태의 장점만을 살린 분리형을 개발하였다. 즉 하나의 케이블을 이용하여 두가지 이상의 센서를 이용할 수 있도록 하였다. <그림 4-48>에서 (A)는 수위센서와 전기전도도 센서를 동시에 사용한 것으로 일종의 일체형으로 볼 수 있다. 장기관측망이나 해수침투용으로 사용할 경우 (A)와 같은 방식으로 두 센서를 하나의 케이블에 연결하여 사용할 수 있다. (B)는 전기전도도 센서만을 연결한 경우이며, (C)는 수위센서만을 연결한 형태이다. (B)와 (C)는 분리형이라 할 수 있다. 양수시험용 또는 지하수 관정개발시 (C)와 같이 수위만을 연결하여 사용가능하며, 전기전도도와 온도만을 측정할 경우 (B)와 같이 연결하면 된다. <그림 4-49>는 수위센서만 사용하는 경우이고, <그

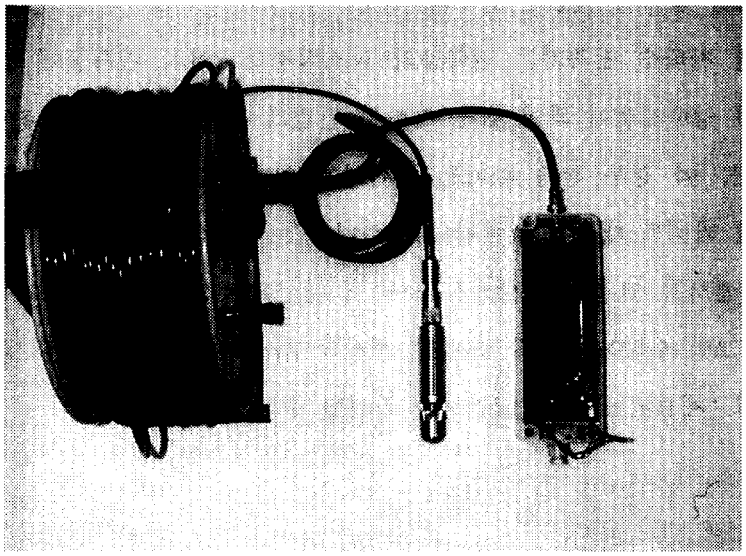
림 4-50>는 전기전도도 센서만을 사용한 경우이다. 그리고 <그림 4-51>는 전기전도도와 수위센서를 동시에 사용한 경우이다.

이 혼합형 지하수 관측기는 관측공의 최소직경인 2인치 관측공에 설치사용 가능할 뿐만 아니라 양수시설이 되어 있는 지하수 관정의 경우 별도의 PVC 관측 파이프가 설치되어 있다. 이 경우 관측 파이프가 휘어져 있더라도 수위센서와 전기전도도 센서 사이의 연결 커넥트 때문에 설치가 가능하다.

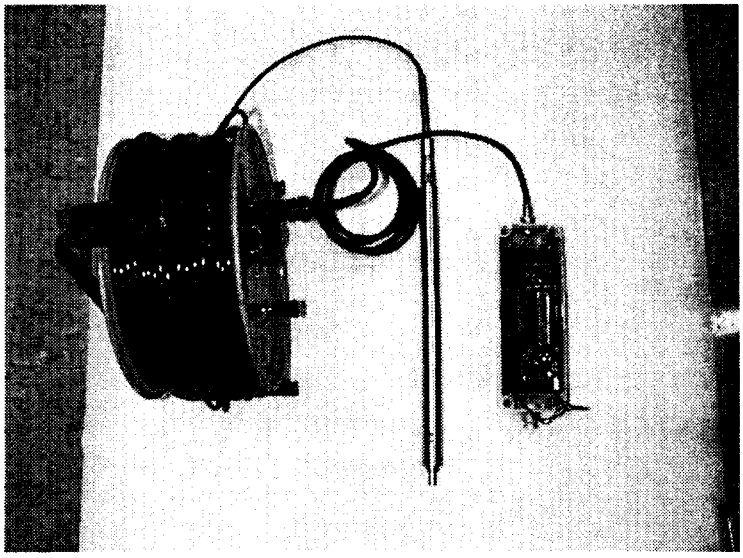
우선 두 센서를 연결하는 방법은 커넥트 대신에 <그림 4-52>와 같이 직접 연결할 수 있다. <그림 4-53>은 직접 연결한 센서의 사진이다. 이와 같은 방법으로 제작할 경우 가공비, 재료비에서 약 12만원을 절감할 수 있다.



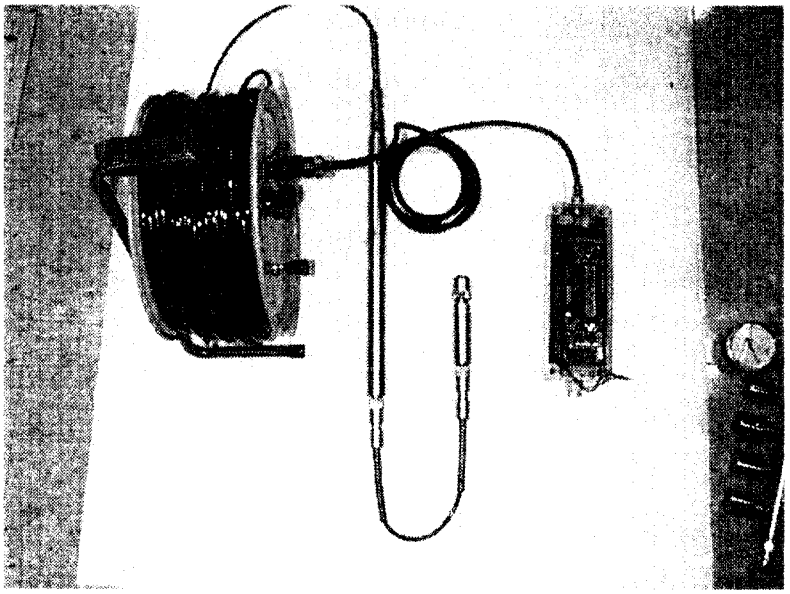
<그림 4-48> 분리형 지하수 자동관측기



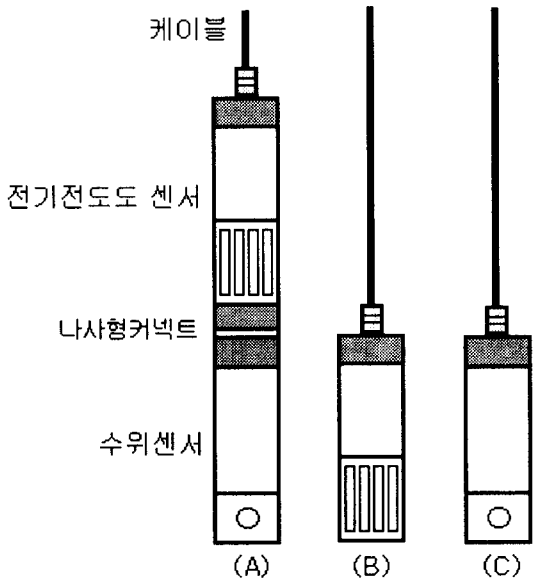
〈그림 4-49〉 수위 측정시 연결방법



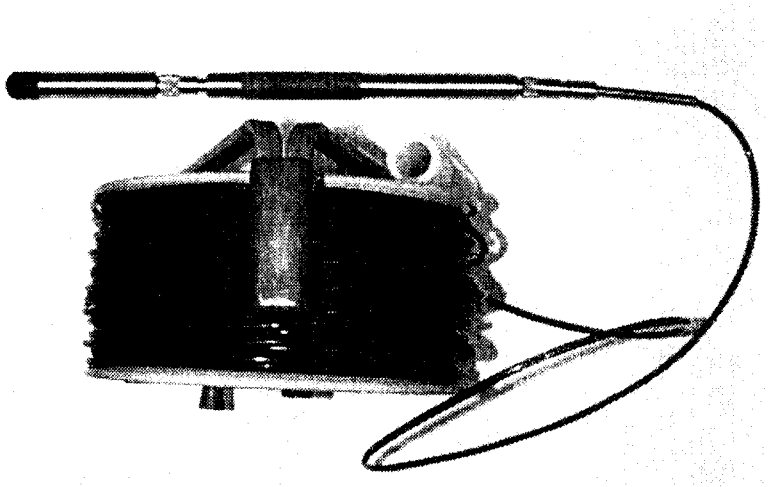
〈그림 4-50〉 전기전도도 측정시 연결방법



〈그림 4-51〉 수위와 전기전도도 동시 측정시 연결방법



〈그림 4-52〉 커넥트케이블을 제거한 분리형 지하수 자동관측기



<그림 4-53> 커넥트케이블을 제거한 분리형 지하수 자동관측기 사진

2. 일체형 지하수 자동관측기

최근 분리형보다 일체형이 많이 보급되는 이유는 일체형이 분리형보다 가격면에서 저렴하다는 것이다. 가격차이가 발생하는 이유는 케이블 때문이다. 센서의 정밀도와 정확도, 자료처리장치의 능력에 따라 다소 차이는 있지만, 관측기 케이블 길이가 150m일 때 케이블 가격이 20,000원/m이므로 센서종류별 케이블을 별도로 구입할 경우 300만~500만원 정도 차이가 발생한다.

지하수를 효율적으로 감시 및 관리하기 위해서는 장기적으로 계측할 수 있는 수량과 수질을 동시에 측정할 수 있는 관측기가 설치되어 있어야 함에도 불구하고 대부분의 지하수 관정은 방치된 상태이다. 최근 정부에서는 사용하지 않는 관정에 대해서 폐공을 실시하고 있다. 주변 지역의 지하수변화를 관측하기 위한 관측공으로 사용하는 것이 바람직하지만 지자체의 인식부족과 함께 대부분 수입되는 장비가 비싸기 때문에 제대로 설치하지 못하는 실정이다. 이를 해결하기 위해서는 우선 관측기의 가격이 저렴해야 한다. 관측기의 가격이 (표 4-8)에서 보듯이 400~500만원 정도이며 자

료처리장치를 포함할 경우 600만원 이상이기 때문에 예산상 설치가 불가능하다.

(표 4-8) 외국 제품의 수입가격

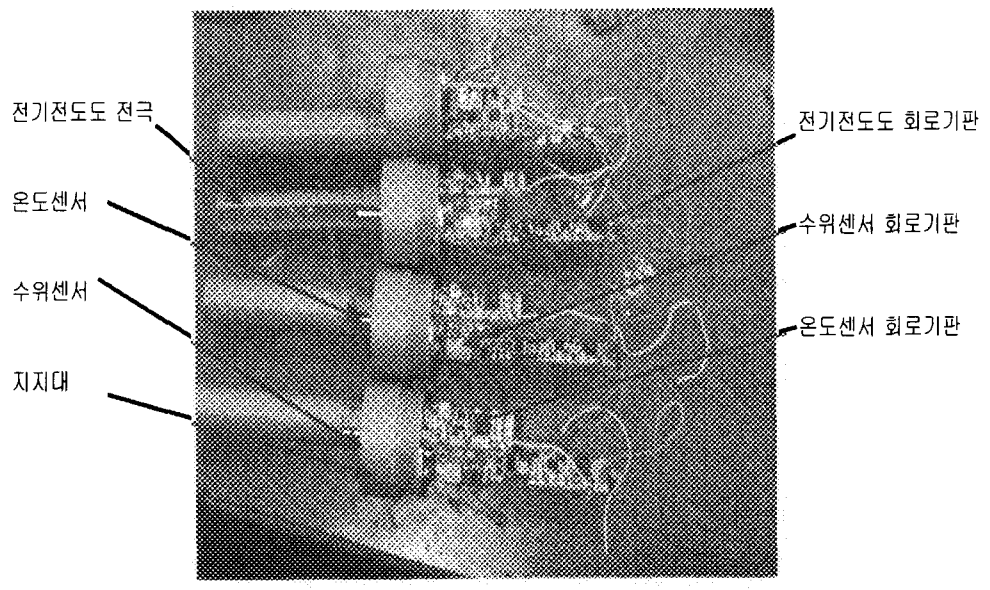
제조회사	제품명	규격	단가	비고
I사(미국)	Data Logger	8채널 통신포트 분석프로그램	$3,744	1999.12
	수위센서	0~200m	$3,438	
	부속장치	케이블 링 밧데리 통신케이블	$100	
M사(영국)	일체형	data logger 수위센서 전기전도도 온도센서	₩4,725,000	1996. 12
H사(미국)	일체형	수위센서 전기전도도 온도센서	₩5,280,000	1999. 10
		Data logger	₩933,000	

본 연구의 목적은 지하수 관측기의 국산화 뿐만 아니라 가격면에서도 충분한 경쟁력을 갖춘 일체형을 제작하는데 있다. 일반적으로 분리형으로 제작한 지하수 관측기의 가격이 기존 관측기에 대비하여 하나의 케이블만을 사용하기 때문에 가격경쟁력을 갖추고 있지만 외국산 일체형과 비교시 가격면에서 큰 차이는 나지 않았다. 이는 센서마다 외형을 별도로 제작해야 하기 때문인데 이를 해결하고자 분리형과 동일한 수위센서, 전기전도도센서 및 온도센서회로를 사용하여 일체형을 제작하여 가격을 저렴화하고자 하였다.

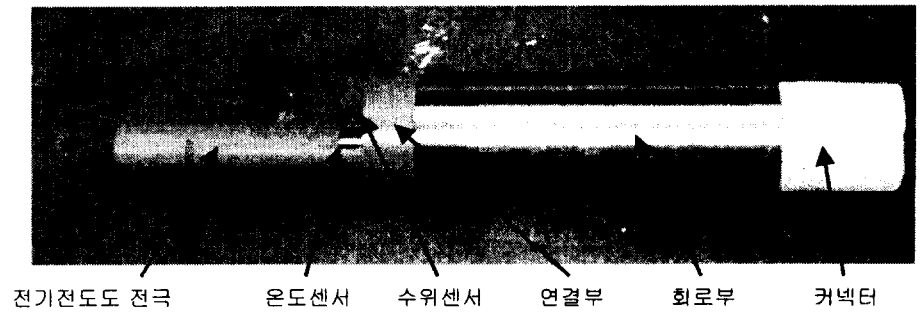
<그림 4-54>와 같이 관측기 센서지지대의 외측에는 전기전도도 전극과 수위센서에 사용되는 압력센서와 온도센서를 취부하고 내측에는 각 센서의 회로기판을 장착한 형태의 일체형 지하수 관측기를 제작하였다. 일체형 관측기의 외형은 <그림 4-55>와 같

이 40㎜ × 250㎜로 하였다. 만약 수위센서만을 사용할 경우 <그림 4-56>의 (a)와 같이 센서지지대에 수위센서만 장착하면 된다. 전기전도도와 온도센서만을 사용할 경우 (c)와 같이 센서지지대에 전기전도도와 온도센서만 장착하면 된다. 3가지 항목을 별도로 측정하고자 할 때는 (b),(d)와 같이 구성하면 된다. 이와 같이 구성한 일체형은 기존 관측기 가격의 절반수준이 되어 충분한 가격 경쟁력을 가질 수 있었다.

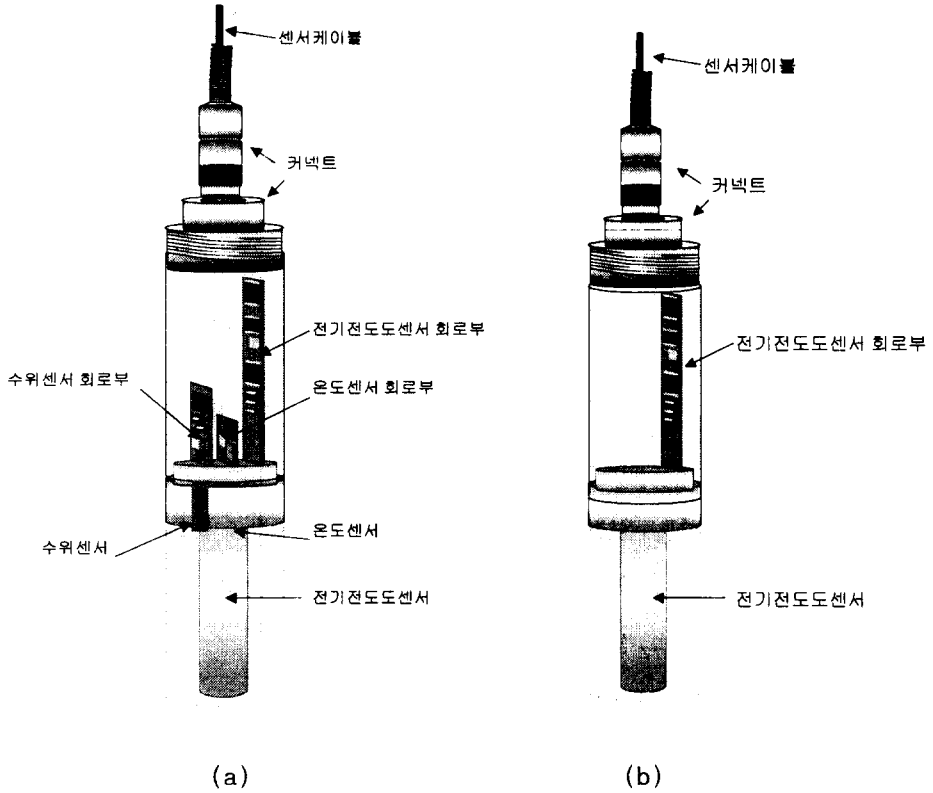
본 연구에서 개발한 수위센서, 전기전도도센서 및 온도센서를 사용하여 여러 가지 설치목적에 따라 적합한 형태로 제작이 가능하였다.

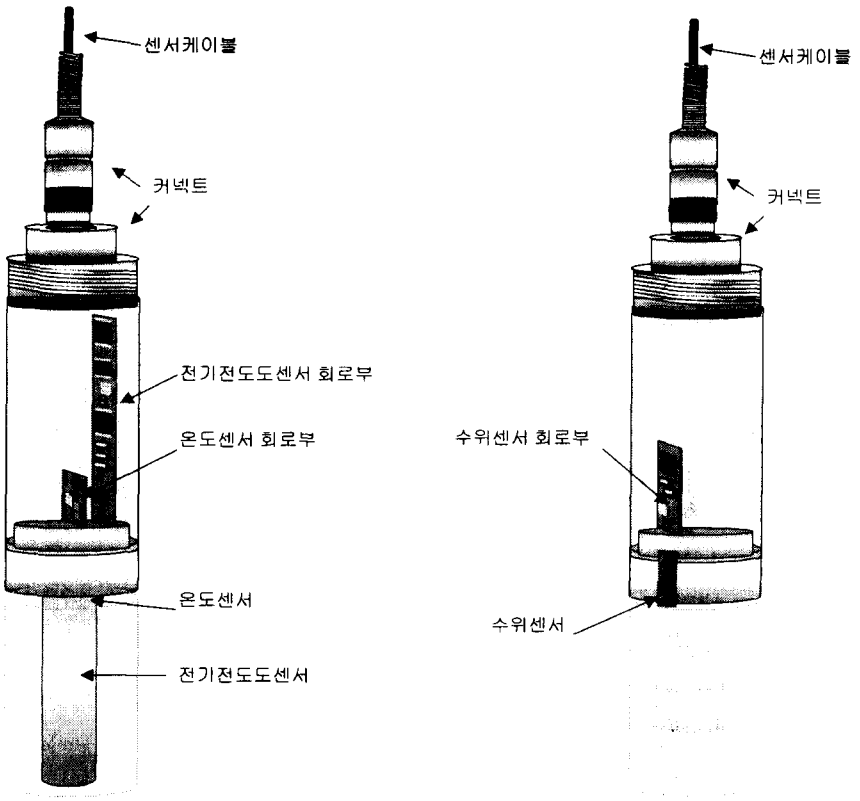


<그림 4-54> 일체형 지하수 관측기 내부사진



<그림 4-55> 일체형 지하수 관측기 외형





〈그림 4-56〉 일체형 지하수 자동관측기의 센서연결방법

(a)수위, 전기전도도, 온도센서연결

(b)전기전도도센서연결

(c)전기전도도, 온도센서연결

(d)수위센서연결

제 5 장 지하수 자동관측기 운영프로그램 및 사용방법

제 1 절 자료처리장치 운영프로그램

지하수 관측기의 운영프로그램은 사용자가 전자 및 전산 분야의 비전문가라는 전제하에 간단한 조작으로 전 기능을 발휘할 수 있도록 하였다. 또한 현재 계측상태를 잘 나타내도록 하고 필요에 따라서는 시간과 같은 제어변수를 쉽게 변경하거나 센서를 교정할 수 있도록 하였다. 또한 향후 관측기의 개선에 대비하여 프로그램을 변경할 필요가 생기는 것을 감안해서 프로그램을 모듈화 하였다. 그리고 복잡한 루틴들은 다시 서브루틴으로 작성함으로써 프로그램 수정을 쉽게 하였다. 특히 프로그램이 복잡한 LCD 제어부분은 서브루틴을 미리 개발함으로써 해당 서브루틴을 수행시킴으로써 그 기능을 수행할 수 있도록 하였다. (표 5-1)은 운영프로그램의 서브루틴들과 그것의 기능이다. 이 프로그램은 계측, 센서교정, 측정시간제어, 절전기능과 통신기능을 가지며 초기화 프로그램과 주 프로그램 루틴, 인터럽트 서비스루틴, 서브루틴들로 구성되어 있다. 지하수 자동관측기의 전체 프로그램은 부록 5에 실었다.

1. 운영프로그램 작성방법

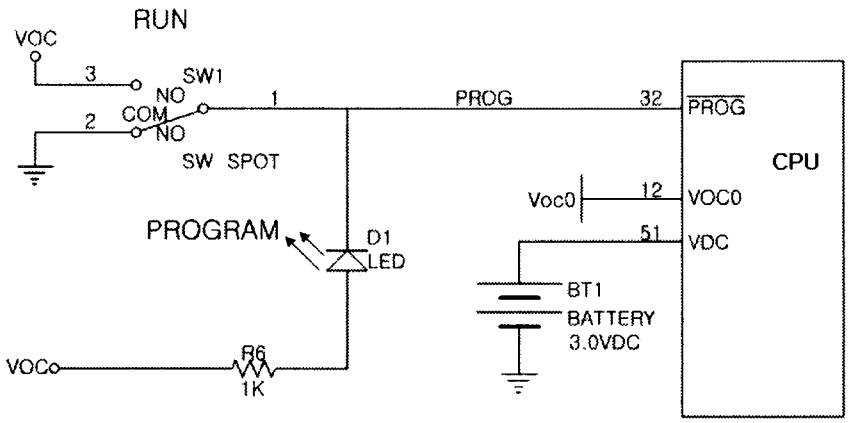
지하수 자동관측기 운영프로그램은 우선 ANSI C 언어로 작성하였다. 이 소스파일을 KEIL 컴파일러를 사용하여 인텔방식의 HEX파일로 만든 후 DOS 환경하에서 프로그램을 CPU에 loading하였다. Loading은 DOS환경에서 KIT.EXE를 사용하였다. KIT.EXE를 실행하면 'kit>' 프롬프트가 나오는데 우선 PC COM port를 설정하고 다음은 통신속도를 '9600'으로 설정하면, CPU에 Status 명령어를 사용하여 정상적으로 설정되었나를 확인 한 후 HEX파일을 loading한다.

자료처리장치의 시리얼통신포트를 운영프로그램 loading과 함께 PC로의 자료송수신용으로도 사용할 수 있게 설계하였다. 즉 운영프로그램을 loading 할경우 <그림 5-1>

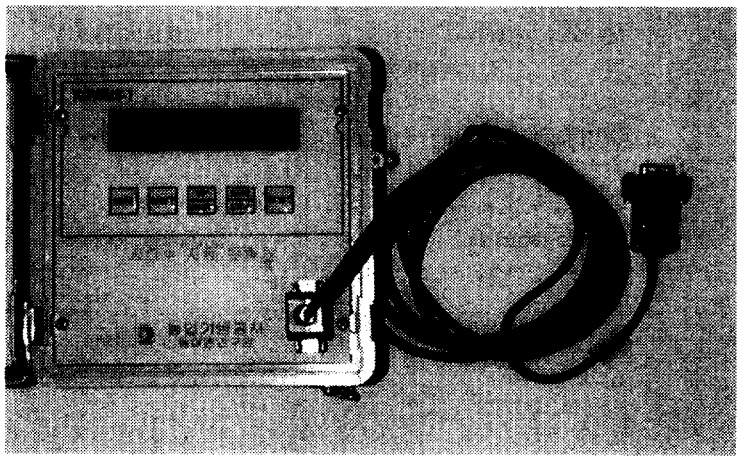
(표 5-1) 자료처리장치의 서브루틴과 기능

서 브 루 틴	기 능
initial_int	MPU를 위한 초기화
delay	Time Delay Routine, SoftwareTime Delay
Wdt_out	WatchDog 신호
init_port	MCS51의 Port 초기화
Buz_sound	Buzzer Sound Output
E_Pulse	LCD Enable 신호
Func_set	LCD 컨트롤러 HD44780의 기능설정
Init_Lcd	LCD를 초기화 함수
write_char	LCD에 한 문자 표시
lcd_printf	LCD의 주소를 찾아 문자열 표시
Lcd_OneCharWrite	LCD에 한 주소 문자를 표시
Init_DS1305	DS1305 초기화 동작
Write_DS1305	DS1305로 데이타를 쓰기
char Read_DS1305	DS1305에 데이타를 읽기
set_time	Time 변수들에 초기값을 입력
At_PowerON	전원을 ON시 실행하는 초기화 루틴
CHKPLS	Key 눌림상태 정의
getkey	Key switch sensing
tx232_1	관측기의 Data 전부 송신
tx232_2	현재 관측기의 Data 송신
tx232_3	관측기의 Data 7회분 송신
time_wr	time 설정
time_read	DS1305에서 Time Data 읽기
DS1305_SetValue_Extract	DS1305에 저장된 각종 Setting값 추출
int getadc	Routine for data fetched of A/D(TC835)
SMD_init	PC에서 들어온 명령어 초기화
pc_cmd_div	PC에서 들어온 명령어 분리 작업
DS1305_SettingUp	PC에서 System의 시간 설정
Ram_Clear_All	저장된 5000회 분 Data를 Clear
calibration_mode	교정하여 교정값을 SAVE 시키는 Mode
int Inteval_Set_time	RS232 통신을 통한 저장 시간 간격 조정
Massange_Disp	측정하고 있는 값 표시
MassDis012	고정된 Display Title
MassTime	현재 날자 및 시간 Display
Current_Data	RS232를 통하여 보내질 현재 Data
AD_theorem	A/D Converter값 정리

과 같이 마이크로프로세서 PROG(32핀)에 Low신호를 입력하고, PC와 자료송수신할 경우 High신호를 입력하게 한다. 이러한 기능 스위치를 <그림 5-2>와 같이 자료처리장치와 PC와 통신하는 시리얼케이블에 설치하였다.



<그림 5-1> 통신포트의 운영프로그램 Loading과 자료전송기능 전환회로도

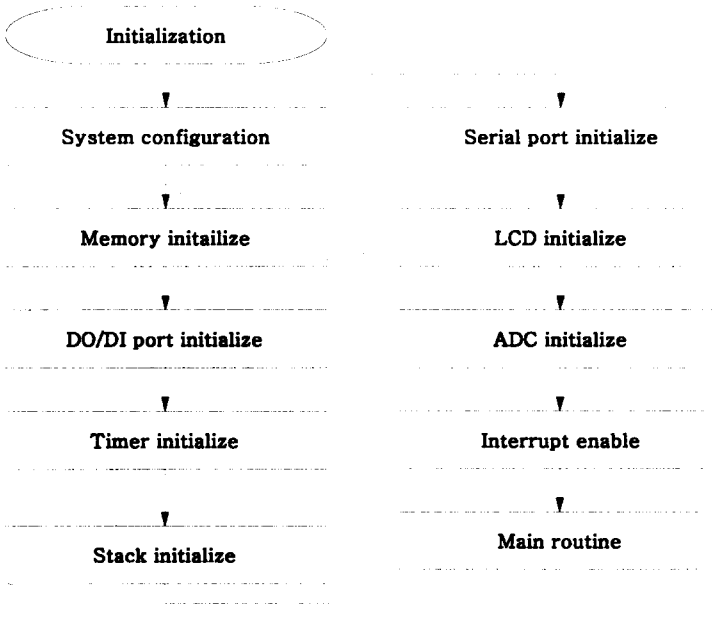


<그림 5-2> 자료처리장치의 시리얼통신 케이블

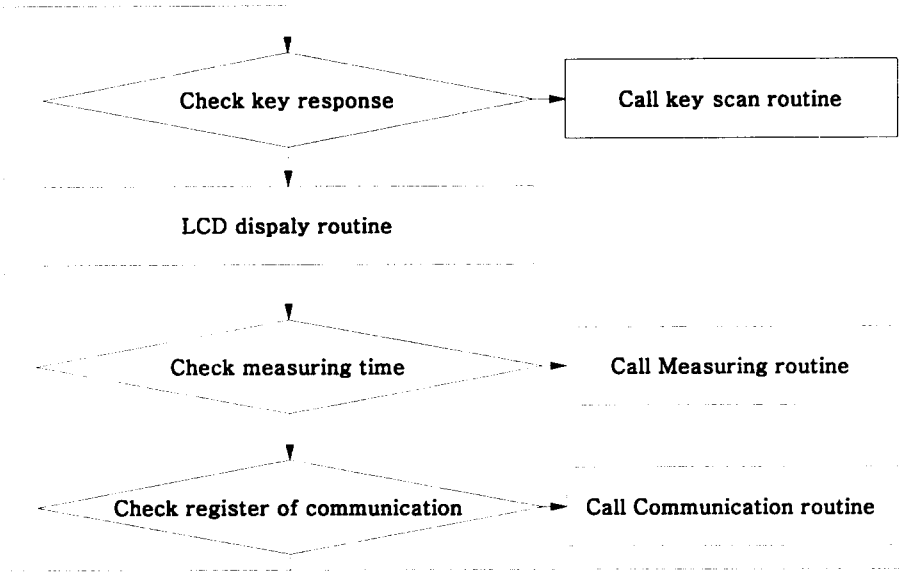
2. 초기화 프로그램과 주 프로그램

지하수 자동관측기의 운영프로그램 구성을 <그림 5-3>과 같이 설계하였다. 초기화 루틴에서 먼저 마이크로프로세서의 입출력 포트 설정, 인터럽트 우선순위를 결정하고, 각 모듈을 초기화 및 설정하고, LCD를 초기화하였다. 마지막을 주 프로그램을 수행하기 전에 인터럽트를 가능하게 하고 그 우선 순위를 결정하였다.

주 프로그램은 무한 루프 형식을 취하였는데 프로그램은 다음과 같은 과정을 밟도록 하였다. 즉 전원이 공급되면 system configuration register를 확인하고, 메모리에 저장되어 있는 제어변수(측정간격, 측정자료를 저장할 메모리 위치, 저장된 자료수 등), 키 상태, 정전여부를 확인하고, 측정시간에 도달하였을 때 측정하는 일을 반복한다. 이런 작업을 하는 도중 인터럽트가 걸리면 인터럽트 서비스 루틴을 수행하고 주 프로그램을 되돌아오게 설계하였다. 메인루틴 프로그램 순서도는 <그림 5-4>와 같다.



<그림 5-3> 자료처리장치 운영프로그램의 초기화 과정

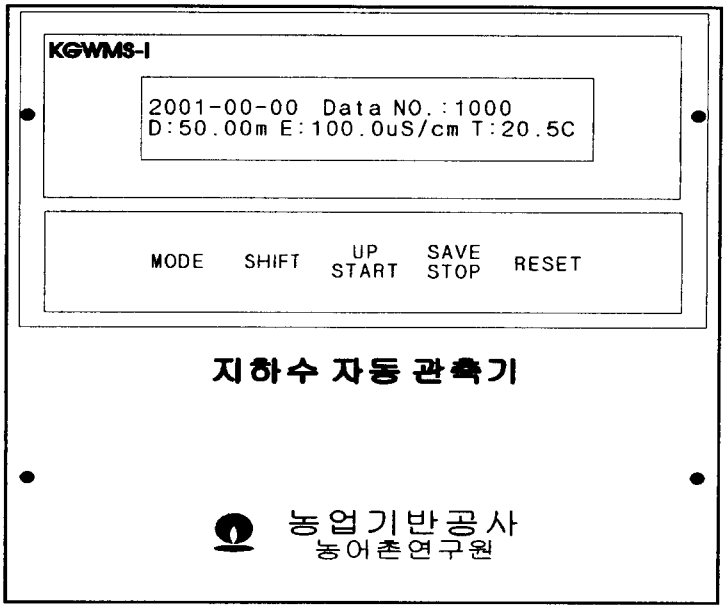


<그림 5-4> 자료처리장치 운영프로그램의 순서도

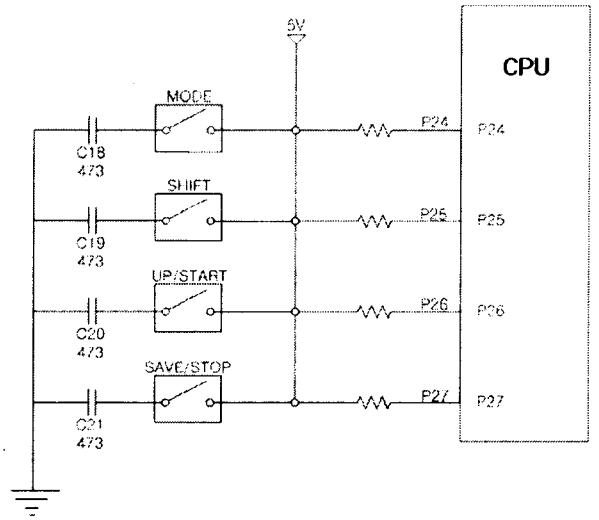
3. 지하수 자동관측기 사용방법

지하수 자동관측기 사용방법은 입력키의 기능에 의존한다. 자료처리장치의 입력키는 배치는 <그림 5-5>와 같이 전면에 MODE, SHIFT, UP/START, SAVE/STOP, RESET 5개 키를 배치하였으며, 그 회로는 <그림 5-6>와 같이 키 하나에 입력신호 한 라인을 할당하여 마이크로프로세서의 입출력포트에 직접 연결하였다. 키 입력신호는 항상 High 상태로 유지하다가 키가 눌러 졌을 때 Low 상태로 된다. 운영프로그램에서 어떤 키가 눌러 졌는지 확인하며, 이때 입력상태에 따라 해당되는 서브루틴을 실행하게 된다.

키 사용방법은 (표 5-2)와 같다. 지하수 관측기가 측정대기상태에서 측정자료 확인 및 제어변수 재설정을 위해서는 우선 자료처리장치를 Wake Up해야 한다. SHIFT키와 RESET 키를 동시에 누를 때 이 기능이 실행된다. 다시 대기상태로 하고자 할 때는



〈그림 5-5〉 지하수 자동관측기 입력키 배치도



〈그림 5-6〉 자료처리장치내 키입력 회로도

UP키를 누르면 된다. 그 외 센서 교정을 위해서는 MODE키를 4초 이상 누르면 교정모드로 전환된다. 센서 교정방법은 5-3장에 별도로 설명하였다.

(표 5-2) 지하수 자동관측기 키 사용방법

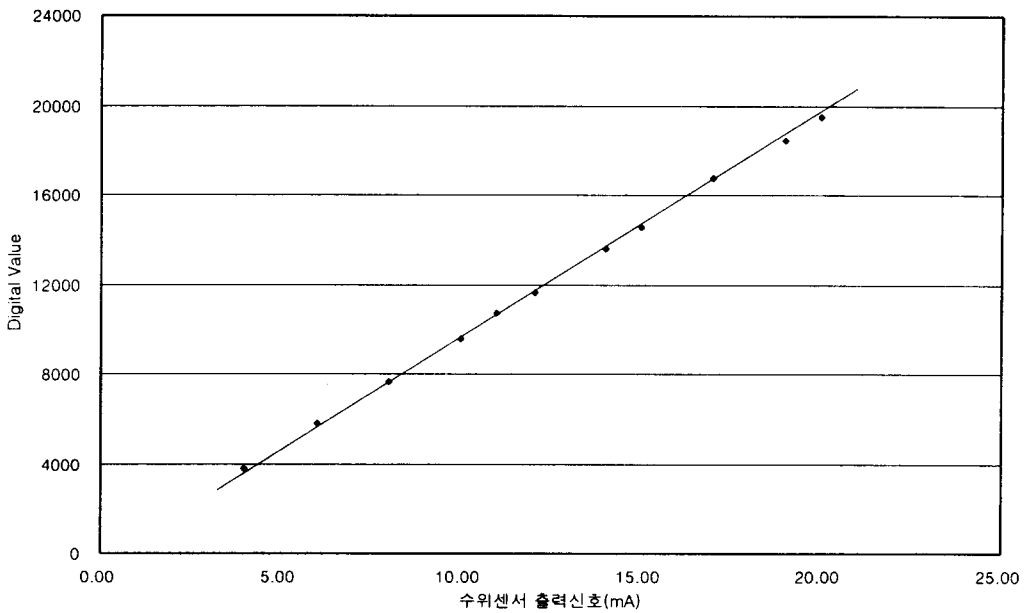
기 능	키 입력	실 행 설 명
Wake Up	SHIFT + RESET	자료처리장치에 주 전원 공급되며, LCD에 날짜, 시간이 표시됨
LCD 표시	MODE	수위→전기전도도→온도→날짜,시간 순으로 LCD에 표시됨
LCD Light	SHIFT	LCD light를 ON 시킴
대기상태	UP	자료처리장치의 주전원을 차단하고 다시 대기상태로 전환
디지털 값	SAVE	선택된 센서의 디지털 값을 LCD에 표시
교정 모드	MODE 4초동안 누름	센서의 교정모드로 전환

4. 계측방법

지하수 자동관측기에 사용된 TC835 ADC의 분해능력은 15bit로 아날로그신호(0～+V)를 최소 0에서 최대 19,999까지 디지털 신호로 변환이 가능하다. 아날로그 입력채널에 강제적으로 0mA에서 20mA 까지 증가시키면서 관측기로부터 출력되는 디지털값으로 변환할 때 이론적으로 0mA를 연결했을 때 디지털값은 0이며, 20mA일 때 19,999가 되어야 한다. 그러나 <그림 5-7>을 보면 12.00mA 아날로그신호에 11,999 디지털값이 출력되어야 하는데 실제 디지털 값은 11,938이 출력이 된다. 이 값은 약 30cm 정도의 오차에 해당된다. 따라서 센서신호가 정확하게 직선적으로 출력되더라도 저항과 같은 부품자체의 오차에 의하여 출력신호는 차이가 날 수 있다. 이 오차를 센서교정 모드에서 보정할 수 있도록 하였다. 즉 교정모드에서 센서의 측정범위내에서 실제값을 입

력하게 되며 자료처리장치에서 디지털값을 읽어 자동적으로 표정곡선을 구하며 이 식은 메모리에 저장된다. 따라서 한 센서를 계속 같은 아날로그 입력채널에 연결할 경우 교정할 필요가 없다. 그러나 다른 센서를 연결할 경우에는 교정을 해야 한다.

센서신호를 디지털 값으로 변환하는 순서는 다음과 같다. 우선 지하수 관측기의 ADC에 연결되는 아날로그 4채널 중 1번 채널은 수위, 2번 채널은 전기전도도, 3번 채널은 온도센서를 연결한다. 초기화 루틴에서 변수들을 초기화하고, 멀티플렉스(4052) AN0에 연결되어 있는 수위신호를 선택한 후 10회 평균값을 저장하였다. 그 다음에는 AN1, AN2, AN3을 차례대로 각각 10번씩 읽어들어 그 평균값을 저장하도록 하였다.



<그림 5-7> 수위센서출력신호와 디지털 값

5. 자료 송수신

컴퓨터와 지하수 관측기의 자료처리장치의 송수신은 인터럽트방식으로 처리할 수 있도록 하였다. 컴퓨터의 시리얼 포트와 자료처리장치의 포트를 연결한 후 신호를 보내게 되면 외부에 어떤 동작도 필요없이 두 시스템은 인터럽트가 발생하여 그 해당되는 루틴으로 처리하도록 하였다. 두 시스템의 통신속도는 9,600bps, 데이터는 8비트, 1 stop bit, none parity로 설정하였다.

컴퓨터에서 자료처리장치에 전달되는 명령어는 (표 5-3)과 같다. 자료처리장치는 컴퓨터로부터 수신된 자료를 받아 어떤 명령어인지 식별한 후 버퍼에 저장하고 지정된 명령어와 일치하면 해당 서브루틴을 실행한다. 우선 컴퓨터와 자료처리장치가 연결되었을 때 자료처리장치는 16진수0X02(STX)와 0X03(ETX)을 검색한다. 이 두 자리수가 검색되었을 때 중간에 있는 데이터를 자료처리장치의 명령어와 제어변수를 구분하여 명령어에 따라 메모리에 설정된 제어변수를 변경하거나, 그에 해당되는 자료를 컴퓨터에 전송한다. 컴퓨터로부터 수신되는 명령어포맷은 <그림 5-8>과 같다.

컴퓨터에서 전체자료요청(PRESENT) 및 현재자료요청(REQFULL) 명령어가 있을 때 해당되는 자료를 컴퓨터에 전송해야 한다. 현재자료요청시 <그림 5-9>와 같이 아스키코드로 자료를 전송한다. 컴퓨터에서는 전송된 자료를 해석하여 화면에 표시한다. 그러나 전체자료요청시 제3장에서 언급한 바와 같이 메모리 절약을 위하여 측정자료를 16진수방식으로 저장하기 때문에 컴퓨터에 송신되는 자료는 <그림 5-10>과 같이 16진수로 전송한다. 컴퓨터에서 16진수 자료를 받아 날짜, 시간은 정수로, 계측자료는 실수로 변환하고 변환된 자료를 화면에 표시하거나 그래프로 도식화했으며, 하드디스크에 저장할 수 있게 하였다.

(표 5-3) 컴퓨터로 수신되는 명령어

명령어	기 능
PRESENT	현재 측정된 자료를 PC에 전송
REQFULL	메모리에 저장된 전체 자료를 PC에 전송
MODIFYt	자료처리장치 타이머 시간 수정
DSITIME	측정간격 수정(최소 단위:1분)
CLSDATA	메모리에 저장된 모든 자료 삭제

시작	명령어							Check Sum				종료
STX	P	R	E	S	E	N	T	0	0	0	0	ETX

(현재자료요청)

시작	명령어							Check Sum				종료
STX	R	E	Q	F	U	L	L	0	0	0	0	ETX

(전체자료요청)

시작	명령어							년		월		일		시		분		Check Sum				종료
STX	M	O	D	I	F	Y	t	0	1	0	7	1	2	2	3	4	5	0	0	0	0	ETX

(자료처리장치 시계수정)

시작	명령어							시		분									Check Sum				종료
STX	D	S	I	T	I	M	E	0	0	0	2	0	0	0	0	0	0	0	0	0	0	ETX

(측정간격조정)

시작	명령어							Check Sum				종료
STX	C	L	S	D	A	T	A	0	0	0	0	ETX

(저장된 자료삭제)

〈그림 5-8〉 컴퓨터로부터 수신되는 명령어 포맷

시작	예약어				Depth(m)							EC(uS/cm)					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
STX(0x02)	D	C	T	P	+	1	0	0	.	0	0	+	1	0	0	0	0

Temp(온도 ℃)						Channel 4의 A/D값						Check Sum				종료
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
+	0	5	1	.	6	+	1	2	3	4	5	0	0	0	0	ETX(0x03)

<그림 5-9> 현재자료요청시 컴퓨터에 전송되는 자료포맷

```
02 50 45 53 44 41 54 41 FF FF      : Start 예약어
06 1C 17 34 0C 03 37 05 0B 04 4E  : 1회분 Data (HEX값으로 표시)
06 1C 17 36 0C 03 37 04 ED 04 4E  : 2회분 Data
06 1C 17 38 0C 03 37 05 0B 04 4E  : 3회분 Data
06 1C 17 3A 0C 03 37 05 0B 04 4E  : 4회분 Data
                  .
                  .
                  .
06 1D 00 00 0C 03 3A 05 0E 04 51  : 5000회분 Data
FF FE 00 00 00 00 03              : END 예약어
```

<그림 5-10> 전체자료요청시 컴퓨터에 전송되는 자료포맷

제 2 절 컴퓨터용 운영프로그램

컴퓨터용 운영프로그램은 지하수 자동관측기의 자료처리장치와의 자료 송수신을 위한 프로그램으로서 JAVA언어를 사용하여 시스템의 초기화, 자료 입출력, 입력 자료 그래프 및 저장 등의 기능을 보유하도록 하였다. 이 프로그램은 윈도우(win95, win98, winNT, win2000, winXP 포함) 환경하에서 실행되며, 컴퓨터에 설치하기 위해서는 하드디스크에 40MB 이상의 여유공간이 필요하다. 또한 실행파일(install.exe)로 프로그램을 설치가 가능하며, 제어판의 프로그램 추가/삭제를 통해서 프로그램의 제거가 또한 가능하다.

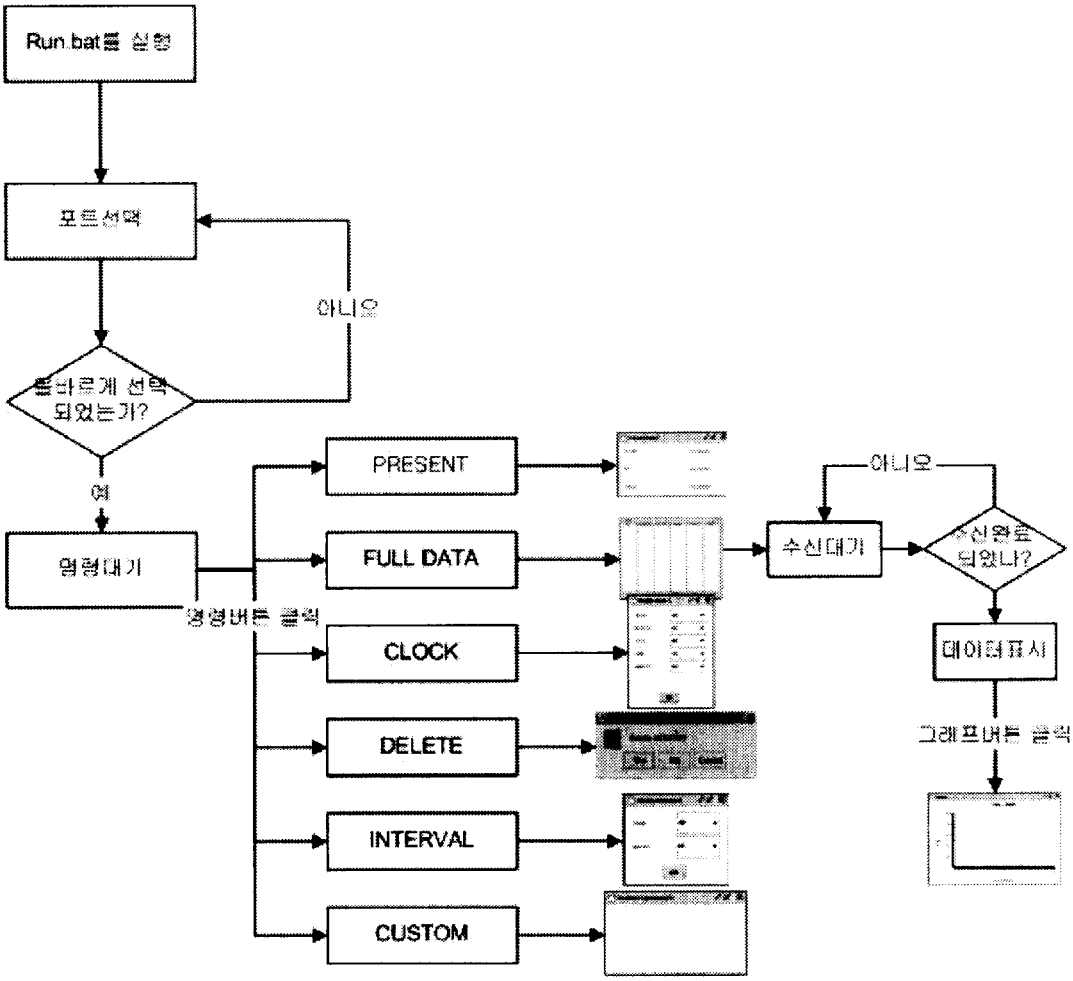
<그림 5-11>은 개발된 컴퓨터용 운영프로그램의 흐름도를 나타낸 것이다.

1. 프로그램 시작

설치된 폴더의 run.bat를 더블클릭 하면 본 프로그램이 실행된다. 프로그램 시작 후 <Port Select> 리스트를 클릭하며 자료처리장치가 연결되어 있는 포트를 설정해 준다. 포트가 정상적으로 설정되었을 경우 ToolTip 항목에 <ComX running>이라는 메시지가 표시된다. 포트설정을 완료된 후에 명령실행 버튼을 클릭하며 해당 명령을 수행할 수 있다.

2. 현재 자료 호출

PRESENT 버튼을 클릭하면 지하수 자동관측기에서 측정한 현재 자료를 컴퓨터에 전송하며, 그 자료는 <그림 5-12>와 같이 표시된다. 이 때 자료처리장치와의 연결상태가 불안전하거나 여러 번 실행하여도 정상적으로 자료가 출력되지 않을 경우 포트설정을 변경하거나 자료처리장치의 연결 및 전원상태를 확인한다.



<그림 5-11> 컴퓨터용 운영프로그램 흐름도

3. 저장된 전체 자료 호출

FULLDATA 버튼을 클릭하면 지하수 자동관측기에 저장된 전체자료를 컴퓨터에 전송하며, 그 자료는 <그림 5-13>와 같이 표시된다. 현재 자료 호출 기능과 마찬가지로 연결상태가 좋지 못한 경우가 포트설정, 연결상태, 전원상태를 확인하고 재실행한다.

전체 자료가 송수신이 완료될 때까지는 다른 명령을 수행할 수 없다. 자료수신이

완료된 후에는 자료를 이용하여 <그림 5-14>와 같이 측정항목별 그래프를 그리거나 컴퓨터에 저장이 가능하다. 우선 원하는 범위를 선택하여 그래프를 그리거나 저장할 수 있으며 범위를 선택하지 않은 경우에는 전체 자료에 대해서 그래프를 그리거나 저장하게 된다. 그래프의 x축은 시간이며 y축 값은 수위, 전기전도도, 온도 중 한 항목을 선택할 수 있다. 또한 항목비교를 위하여 다중선택도 가능하다. 그래프의 크기가 작을 경우 창의 크기를 늘려 그래프 크기를 늘릴 수 있으며(창의 크기에 따라 그래프 비율이 자동으로 맞추어짐), 일부분만 확대해서 보고싶은 경우 좌표상에서 해당영역을 마우스로 드래그할 수 있다. 나머지 영역은 스크롤바를 이용하여 이동하면서 확대된 그래프를 볼 수 있다. 자료 저장시 csv 포맷으로 저장하므로 엑셀을 이용하여 이 자료를 불러들일 수 있다.

4. 시간변경

CLOCK 버튼을 클릭하면 <그림 5-15>와 같이 표시되어 마우스 및 키를 이용하여 시간을 변경할 수 있다. 연도설정은 2001년부터 2030년까지 가능하며, 설정이 완료되었을 경우 하단의 OK 버튼을 클릭하여 설정된 날짜, 시간자료를 자료처리장치에 전송한다.

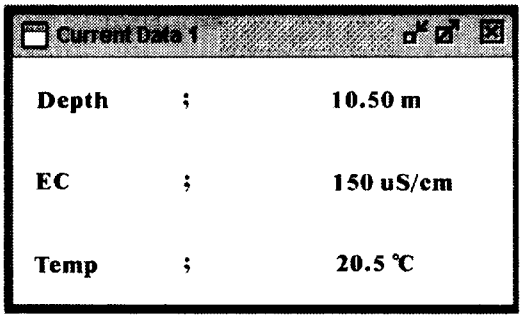
5. 자료삭제

DELETE 버튼을 클릭하면 지하수 자동관측기에 저장된 전체자료를 삭제할 수 있다. 삭제하기 이전 우선 <그림 5-16>과 같이 사용자에게 확인을 묻는 화면이 표시된다. 삭제를 원할 경우 <예>를 취소할 경우는 <아니오>나 <취소>를 선택하면 된다.

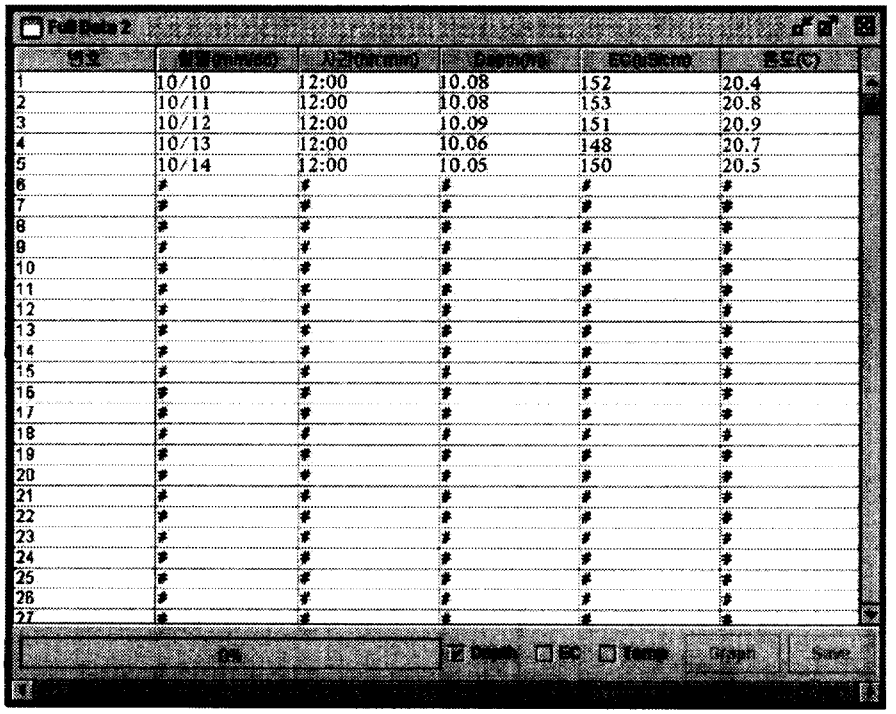
6. 저장간격 조절

INTERVAL 버튼을 클릭하면 <그림 5-17>과 같이 자동지하수 관측기의 저장간격을 조절할 수 있다. 저장간격을 설정 후 하단의 OK버튼을 클릭하면 저장간격을 자료처리장

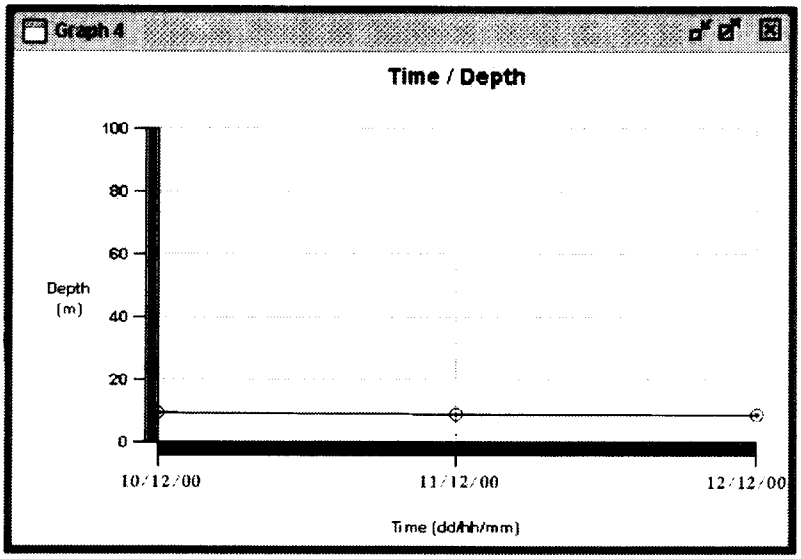
치에 전송한다.



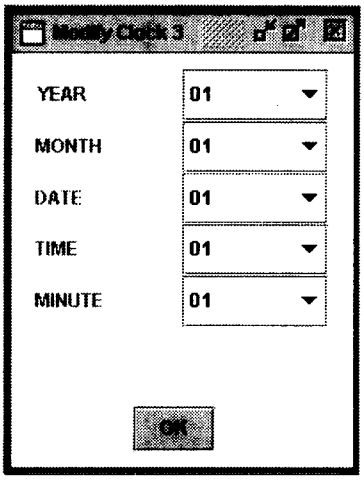
<그림 5-12> 현재자료요청시 화면상태



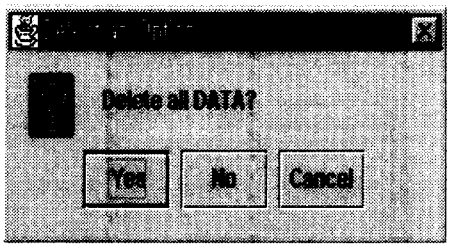
<그림 5-13> 전체자료요청시 화면상태



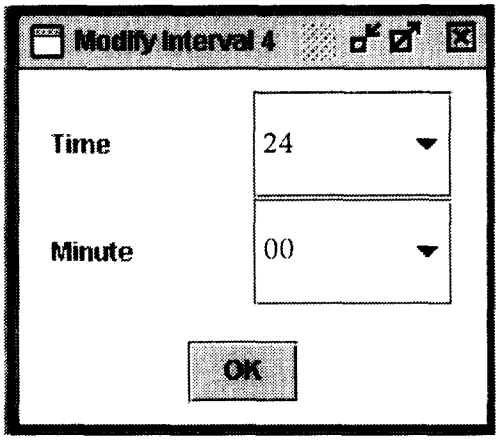
〈그림 5-14〉 전송된 자료 그래프 화면상태



〈그림 5-15〉 시간변경시 화면상태



〈그림 5-16〉 자료삭제시 화면상태

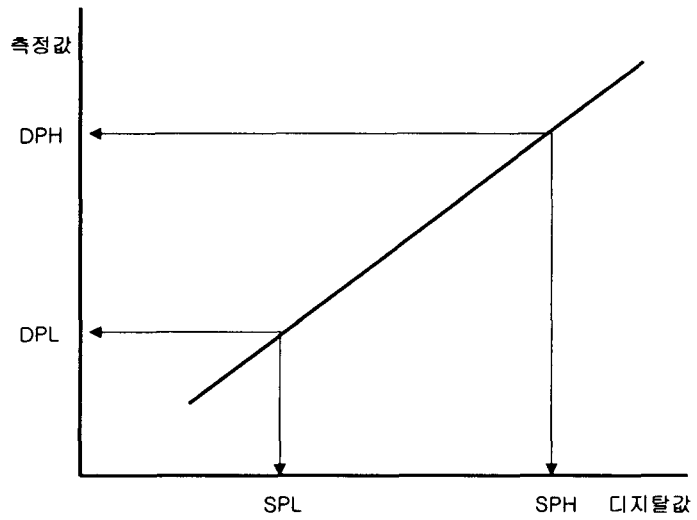


<그림 5-17> 측정간격 변경시 화면상태

제 3 절 센서교정방법

지하수 자동관측기의 정확도를 보장하기 위해서는 센서를 주기적으로 교정하는 것이 바람직하다. 주변 기압이나, 온도 등의 요인은 수위센서 신호에 영향을 미칠 수 있으며 전기전도도는 전극표면상태에 따라 측정값이 다를 수 있다. 따라서 측정시마다 센서를 교정하는 것이 바람직하지만 제작 시 초기값을 가지고 있으므로 정확도에 이상이 없으면 계속 사용이 가능하지만 6개월 간격으로 교정하는 것이 바람직하다.

센서의 교정방법은 두 지점을 이용한다. 즉 측정범위의 최소지점과 최대지점의 실측값과 디지털값을 구한 후 센서의 표정곡선을 작성한다. 표정식은 식(11)과 같다. SPH와 SPL는 직접 입력해야 하며, DPH와 DPL는 자동적으로 입력된다.



$$NDPV = \frac{(DPH - DPL)}{(SPH - SPL)} \times (NMV - SPL) + DPL \qquad (11)$$

NDPV : 시료의 측정값

NMV : 시료의 디지탈값

SPH : 측정범위의 최대일 때 수위, 전기전도도, 온도

SPL : 측정범위의 최소일 때 수위, 전기전도도, 온도

DPH : 측정범위의 최대일 때 디지탈값

DPL : 측정범위의 최소일 때 디지탈값

1. 수위센서 교정

자료처리장치의 전원이 ON상태 MODE 키를 4초이상 누르면 교정모드가 된다. 교정모드상태에서 수위센서의 교정절차는 다음과 같다.

① 측정범위를 결정한 후 교정범위 최대지점과 최소지점을 정한다. 수위센서의 기준값은 대기압이므로 최소지점을 대기압으로 하고, 최대지점은 측정범위보다 약간 높게 한다.

② 먼저 최소지점부터 교정을 실시한다. 자료처리장치의 LCD에는 다음과 같이 표

시된다.

Cal. Depth : +03403
Low Value : 009.30 m

LCD 표시창에 Cal. Depth : +03403은 수위센서로부터 출력되는 아날로그 신호가 디지털 값으로 변환된 값이다. Low Value : 009.30m는 센서로부터 실제 측정한 값이다. 이것을 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 입력이 완료되면 SAVE 키를 누르면 현재 LCD 표시창에서 3403을 수위센서 표정식의 SPL로, 9.30를 DPL값으로 자료처리장치내 메모리에 저장한다.

③ 다음은 최대지점에 대한 교정을 실시한다. 수위센서를 관정에 넣거나, 콤프레사를 이용하여 압력센서에 압력을 가하여 측정범위의 최대지점에 도달하였을 때 교정을 실시한다.

Cal. Depth : +16450
High Value : 094.70m

Cal. Depth : +03403은 아날로그 신호가 디지털값으로 변환된 값이며 High Value : 094.70m는 센서로부터 실제 측정한 값이다. 최저값 입력과 마찬가지로 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 입력이 완료되면 SAVE 키를 누르면 현재 LCD 표시창에서 16450을 표정식의 SPH로, 94.70을 DPH값으로 자료처리장치내 메모리에 저장한다.

2. 전기전도도센서 교정

수위센서 교정이 완료되면 자동적으로 전기전도도 센서 교정모드로 진행된다. 전기전도도 교정절차는 수위센서와 동일하다.

① 측정범위를 결정한 후 교정범위 최대지점과 최소지점을 정한다. 대기압 조건에서 전기전도도 값은 0이므로 최소지점을 대기압으로 하고, 최대지점은 측정범위보다 약간 높게 한다.

② 먼저 최소지점부터 교정을 실시한다. 자료처리장치의 LCD에는 다음과 같이 표시된다.

Cal. Ec : +02348
Low Value : 00034 uS/cm

LCD 표시창에 Cal. EC : +02348은 전기전도도로부터 출력되는 아날로그 신호(아날로그 2번 채널값에 해당함)가 디지털 값으로 변환된 값이다. Low Value : 00034 uS/cm는 센서로부터 실제 측정한 값이다. 이것을 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 대기압일 경우 '0'를 입력한다. 입력이 완료되면 SAVE 키를 누르면 2348을 전기전도도 표정곡선 식의 SPL로, 34를 DPL값으로 자료처리장치내 메모리에 저장한다.

③ 다음은 최대지점에 대한 교정을 실시한다. 측정하고자 하는 범위 이상의 표준용액에 전기전도도 센서를 넣어 교정을 실시한다.

Cal. Ec : +16730
High Value : 09078 uS/cm

Cal. Ec : +016730은 아날로그 신호가 디지털값으로 변환된 값이며 High Value :

09078 uS/cm는 센서로부터 실제 측정한 값이다. 최저값 입력과 마찬가지로 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 입력이 완료되면 SAVE 키를 누르면 16730를 SPH로, 9078를 DPH값으로 자료처리장치내 메모리에 저장한다.

3. 온도센서 교정

전기전도도센서 교정이 완료되면 자동적으로 전기전도도 센서 교정모드로 진행된다. 온도 교정절차는 수위센서, 전기전도도센서와 동일하다.

① 측정범위를 결정한 후 교정범위 최대지점과 최소지점을 정한다.

② 먼저 최소지점부터 교정을 실시한다. 자료처리장치의 LCD에는 다음과 같이 표시된다.

Cal. Temp : +03460
Low Value : -19.3 ℃

LCD 표시창에 Cal. Temp : +03460은 온도센서로부터 출력되는 아날로그 신호(아날로그 3번 채널값에 해당함)가 디지털 값으로 변환된 값이다. Low Value : -19.3℃는 측정한 값이다. 이것을 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 입력이 완료되면 SAVE 키를 누르면 3460을 온도 표정곡선 식의 SPL로, 34를 DPL값으로 자료처리장치내 메모리에 저장된다.

③ 다음은 최대지점에 대한 교정을 실시한다.

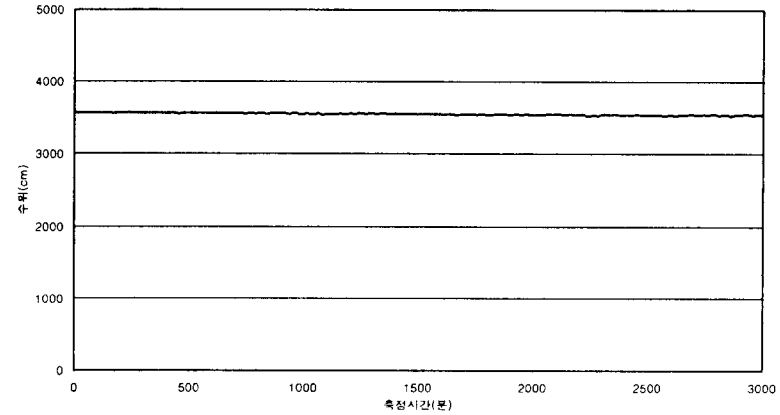
Cal. Temp : +17921
High Value : +48.5 ℃

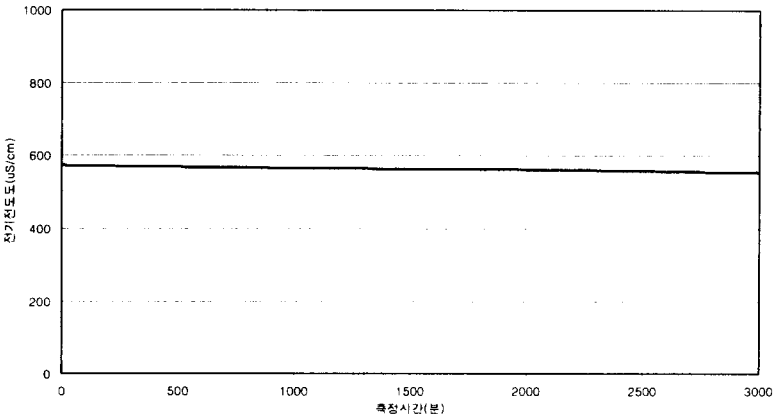
Cal. Temp : +017921는 아날로그 신호가 디지털값으로 변환된 값이며 High Value : +48.5 ℃는 센서로부터 측정한 값이다. 최저값 입력과 마찬가지로 SHIFT 키와 UP 키를 사용하여 실제 정확한 값을 입력한다. 입력이 완료되면 SAVE 키를 눌러 16730를 SPH로, 9078를 DPH값으로 자료처리장치내 메모리에 저장한다.

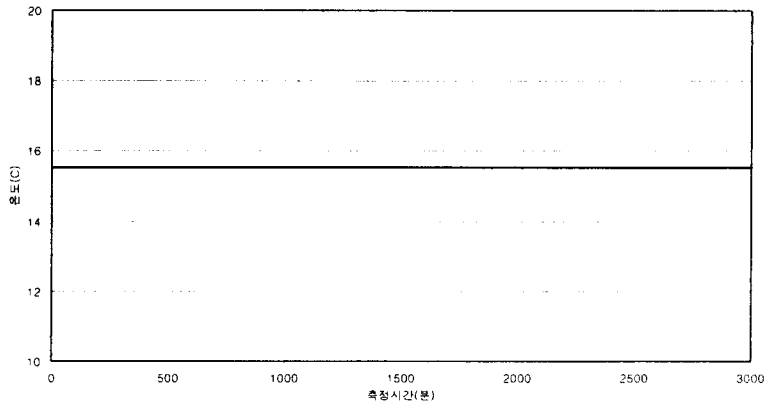
제 6 장 지하수 자동관측기 현장실험

본 연구에서 제작한 지하수 자동관측기를 지하수 관정에 설치하여 수위, 전기전도도, 온도를 계측하였다. 센서의 안정성을 검토하기 위하여 지하수 미사용 관정에 시작품을 설치하였다. 1분 간격으로 3000회 정도 측정한 결과는 <그림 6-1>과 같다. 초기 수동식 수위계로 측정한 수위값은 35.75m 로 실측값과의 차이는 10cm으로 오차범위 이내에 측정됨을 확인하였다. 또한 2일 동안 측정한 결과에서도 오차범위내에서 수렴하였다. 전기전도도와 온도에서도 휴대용 계측기를 사용하여 지하수 시료를 측정한 값은 574 μS/cm, 15.53℃로서 관측기 값과 비교한 결과 수위센서와 마찬가지로 오차범위 이내에서 측정되고 있음을 확인할 수 있었다.

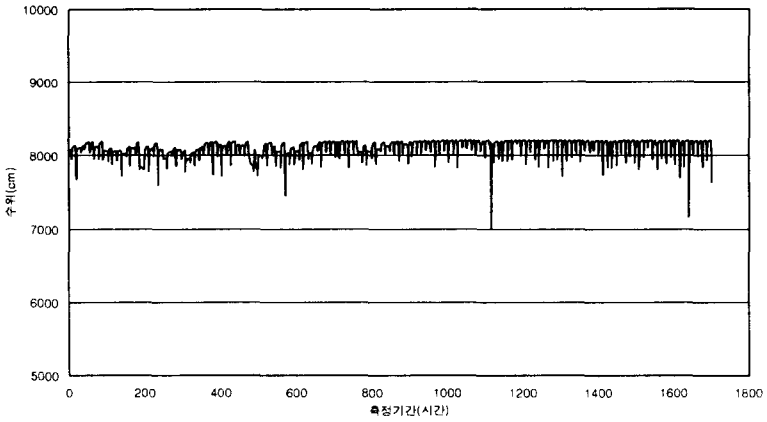
센서의 정확도를 검증하기 위하여 지하수 사용 관정에 시작품을 설치하였다. 수중모터 가동시 센서의 안정성도 검증하고자 센서위치를 수중모터로부터 5.0m 근처에 설치하였다. 1시간 간격으로 70일동안 측정한 결과는 <그림 6-2>와 같다. 이 관정은 아침시간과 저녁시간에만 개방하여 지역주민들이 사용하고 있는 관정으로 2회/일정도 수위변화가 있는 관정이다. 수동식 수위계로 시작시점과 70일후 자연수위를 측정한 센서의 정확도를 비교한 결과 18cm 정도 오차가 있었다. 이는 설치 당시 케이블 길이의 오차가 있으므로 센서 자체의 오차는 비교적으로 적을 것으로 판단된다. 그리고 휴대용 계측장비로 전기전도도와 온도를 측정한 결과에서 오차범위내에서 관측기의 전기전도도와 온도센서도 비교적 정확하게 측정되었다. 수위변화는 모터에 의한 노이즈 변화라기보다는 실제 수위변화로 볼 수 있으며, 온도와 전기전도도의 변화는 사용에 따른 지하수 자체의 변화라 판단된다. 따라서 수위, 전기전도도와 온도 모터에 의한 노이즈 영향을 크게 받지 않음을 확인할 수 있었다.

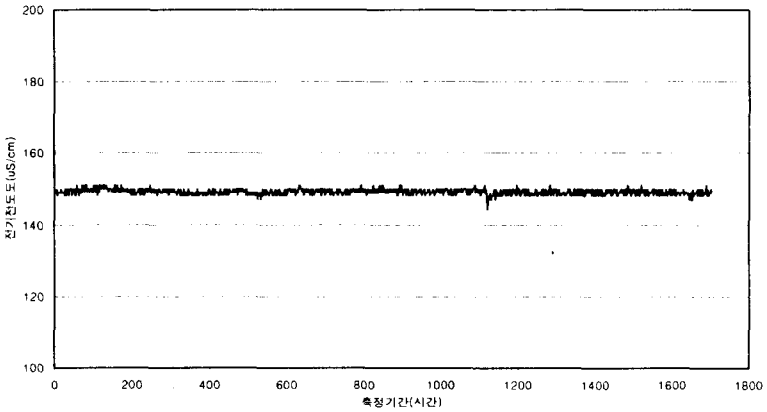


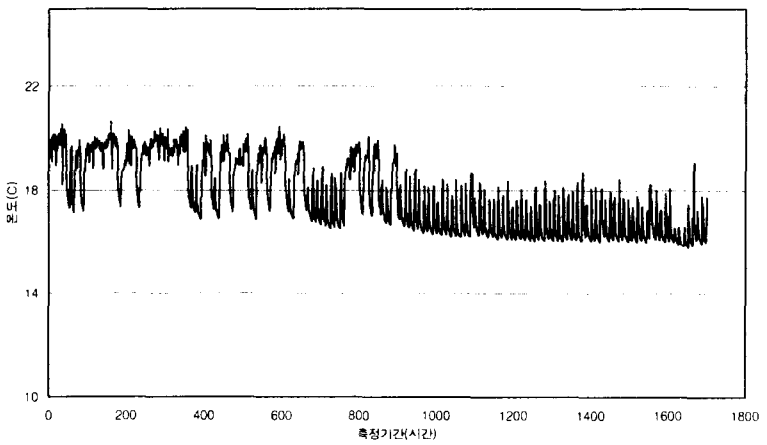




〈그림 6-1〉 미사용 관정에 설치한 시작품의 수위, 전기전도도, 온도변화







〈그림 6-2〉 사용 관정에 설치한 시작품의 수위, 전기전도도, 온도변화

제 7 장 결 론

본 연구로 지하수 관측망 및 지하수 개발에서 사용할 수 있는 수위, 전기전도도, 온도를 측정할 수 있는 지하수 자동관측기를 개발하였다. 이 관측기는 사용자 중심으로 지하수 관측기의 하드웨어와 프로그램을 개발하였으며 기존 지하수 자동관측기와 비교하여 기능과 성능면에서 대등할 정도이며, 가격에서도 2/3수준으로 제작 및 보급이 가능하였다.

또한 센서회로내에서 신호증폭회로뿐만 아니라 필터회로, 버퍼회로 등을 추가하여 전기적 영향을 받지 않도록 하였으며, 지하수 자동관측기의 특성상 장거리 전송임을 감안하여 센서의 최종출력을 4~20mA 전류신호가 되도록 하였다. 또한 자료저장, 전송, 표시 등 지하수 관측기를 제어하는 자료처리장치를 개발하였다.

1. 센서

수위센서는 관측공 최소구경이 54mm(N규격)에 사용할 수 있도록 직경 25mm, 커넥트 부분을 포함하여 길이는 145mm로 소형으로 제작하였다. 측정범위는 0~100m으로 하였으며, 정밀도는 0.1%FS이다. 모터에 의한 노이즈 영향을 받지 않도록 하기 위하여 노이즈 제거회로를 추가하였으며 센서 앞부분에 보호막을 설계, 설치하여 압력센서를 보호하였다. 그리고 수압 10기압에도 견딜 수 있도록 고무링과 실리콘 수지로 완벽한 방수처리를 하였다.

전기전도도 센서는 온도에 영향을 받기 때문에 전기전도도센서에 온도보상회로를 첨가하면서 두 항목을 함께 측정할 수 있도록 하였다. 전기전도도 센서의 측정범위는 정밀도를 높게 하고자 4전극방식을 적용하였으며, 저농도센서와 고농도센서 두가지 유형을 개발하였다. 저농도 센서의 측정범위는 0~5,000μS/cm이며 정밀도는 ±0.25%FS이며, 고농도 센서의 측정범위는 0~50,000 μS/cm이며 정밀도는 ±0.5%이다. 전기전도도 센서의 직경은 수위센서와 동일하게 25mm로 하였으며, 길이는 전기전도도의 회로기판의 크기에 맞게 400mm로 제작하였다. 그리고 전기전도도 전극이 있는 부

분에는 실리콘 수지로 완전 방수 처리하였다. 온도센서는 Pt100으로 제작하였으며 측정범위는 -20~50℃로 하였다. 정밀도는 ±0.1%이다.

지하수 자동관측기의 센서와 자료처리장치는 100m이상의 장거리이므로 센서의 출력신호를 4~20mA로 하였다.

2. 센서케이블

지하수 자동관측기에서 유지관리비용이 많이 드는 원인은 센서케이블이다. 이는 지하수 관측기에 사용되는 센서케이블은 내부에 튜브가 있기 때문에 사용 중 손상되었을 경우 케이블 전체를 교체해야 하는 경우가 있으므로 센서케이블 또한 지하수 관측기의 중요한 부품에 해당된다. 따라서 본 연구에서는 수위센서의 대기압 보정을 위하여 튜브가 있는 지하수 관측기 전용 케이블을 제작하였다. 제작단가는 수입단가보다 약 1/3수준으로 가격 경쟁력을 가질 수 있게 하였다. 중량면에서도 110m에 링을 포함하여 7.5kg밖에 되지 않아 케이블이 늘어나는 경우가 없었다. 장력시험을 실시한 최대 100kgf까지 견딜 수 있었다.

3. 자료처리장치

자료처리장치는 자료측정, 저장, 전송, 측정시간설정, 센서교정 등을 할 수 있게 DS5002FP CPU를 사용하여 제작하였다. 4개 센서입력 채널을 확보하였으며, 노트북 통신과 모뎀 통신을 위하여 통신포트를 설치하였다. 특히 장기 관측망에서 가장 문제가 되는 전원을 해결하기 위해서 전원절약의 한 방법인 Sleep mode를 적용하였다. 측정시에는 자료처리장치에 자동적으로 전원을 투입되어 측정하고 대기시간은 Sleep mode로 자동 전환되게 하였다. 이와 같은 방식을 적용한 결과 측정시에는 50mA가 소모되었지만, sleep mode에는 0.005mA만 소모되어 12V, 1.9Ah밧데리로 6개월이상 장기간 측정이 하였다.

4. 지하수 관측기 유형별 제작

지하수 자동관측기 센서의 유형조사결과 일체형과 분리형의 장단점을 가지고 있기 때문에 사용목적, 현장여건 및 수질 등을 고려하여 선택할 수 있게 센서를 제작하였다. 하나의 센서케이블을 이용하여 두 가지 이상의 센서를 이용할 수 있도록 하였다. 즉 수위센서, 전기전도도센서, 온도센서를 동시에 사용하거나, 개별적으로 사용할 수 있게 하였다.

또한 지하수 관측기의 저렴화 방안으로 별도로 되어 있는 센서를 하나의 프로브에 장착한 40㎜×250㎜ 크기의 일체형을 제작하였다. 비록 외형이 다소 크더라도 센서를 동시 또는 개별적으로 사용이 가능하며, 한 프로브를 사용하기 때문에 제작 단가를 낮출 수 있었다.

5. 운영프로그램

자료처리장치의 운영프로그램은 ANSI C로 작성하였으며, 컴퓨터용 운영프로그램은 JAVA 언어로 작성하였다. 컴퓨터에서도 측정시간설정, 측정자료보기, 자료다운받기 등을 할 수 있게 하였다.

본 연구과제와 관련하여 특허출원을 실시하였으며, 산업체와 연계하여 제품화하여 지하수 관측망사업, 보존 관리사업 및 개발용으로 활용토록 하며, 지하수뿐만 아니라 지표수 수질측정분야에서도 활용할 수 있도록 한다.

특허출원 (2건)

- 지하수 수위·수질 자동관측기 (출원일:2001. 1. 3)
- 압력식 수위감지센서 (출원일:2001. 12. 13)

참 고 문 헌

1. Albert P. Malvino, 1992, Electronic Principles 4th ed. McGraw-Hill.

2. American Public Health Association, 1992, Standard Methods for the Examination of Water and Wastewater, 18th ed. Washington, D.C., U.S.A

3. Bouwer, H. and R. C. Rice, 1976, A slug test for determing hydraulic conductivity of unconfined aquifers completely penetrating wells, Water Resources Research, 12, 423-428.

4. Butler, J. J., Jr., G. C. Bohling, J. Hyder and C. D. McElwee. 1994, The use of slug test to describe vertical variations in hydraulic conductivity. J. Hydrol., 156, 137-162.

5. Dawson, K. J., J. D. Istok, Aquifer Testing Design analysis of pumping and slug tests, Lewis Publishers, Michigan.

6. Englewood Cliffs, N. J., 1983, Motorola : M6805 HMOS, M146805 CMOS Family Microcomputer/Microprocessor User's Manual, 2nd ed., Prentice Hall, Inc.

7. Fetter, C., 1998, Aplied Hydrogeology, Merrill Publishing Company, Columbus.

8. Freeze, R. A. and J. A. Cherry, 1979, Groundwater, Englewood Cliffs, NJ. Prentice-Hall.

9. Fred J. Molz, O. Guven, J. G. Melville, R. D. Crocker, K. T. Matteson, 1986, Performance, Analysis and Simulation of a Two-Well Tracer Test at the Mobile Site, Water Resources Research, 22(7), 1031-1037.

10. Guven O., R. W. Falta, F. J. Molz, and J. G. Melville, 1986, A Simplified Analysis of Two-Well Tracer Test in Stratified Aquifers, Ground Water, 24(1), 63-71.

11. James J. Brophy, 1991, Basic Electronics for scientists 5th ed., McGraw-Hill

12. John F. Pickens, Gerald E. Grisak, 1981, Scale-Dependent Dispersion in a Stratified Granular Aquifer, Water Resources Research, 17(4), 1191-1211.

13. Jurado J., Mohseni S., 1992, Design and implementation of a programmable controller based on differential pressure sensors to contol the runoff for a lauter tub. Technical-Quarterly, Master Brewers' Association of the Americas, 29(1), 6-10.

14. Kemblowski, M. W. and C. L. Klein, 1988, An automated numerical evaluation of slug test data, Ground Water, 26, 435-438

15. Kyung Man Kim, Jae Kun Chun, 1993, Development of automatic measurement and control method based on single chip microcomputer for tackjoo

fermentation, Korean-Journal-of-Food-Science-and-Technology, 25(4), 391-394.

16. Mitchell, B. W., 1983, Instrumentation and Measurement for Environmental Sciences, 2nd ed., Americal Society of Agriculture Engineers, St. Joseph, Michigan.

17. O. Guven, R. W. Falta, F. L. Molz, J. G. Melville, 1986, Analysis and Interpretation of Single-Well Tracer Tests in Stratified Aquifers, Water Resources Research, 24(1), 676-684.

18. OrCad/SDT III, Release 3.02, OrCAD System Co.

19. Patrick A. Domenico and Franklin W. Schwartz, 1990, Physical and chemical hydrogeology.

20. Punidadas P., Decloux M. Trystram G., 1991, Computer control of a cross flow microfiltration pilot plant, Food-Control, 2(3), 152-161.

21. Renard, P., D. Dochain, G. Bastin, H. Naveau and E. -J. Nyns, 1988, Adaptive Control of Anaerobic Methane Digestion Process with the aid of a Mathematical Model, Third European Congress on Biotechnology, 3, 137-144.

22. Ward C. H., W. Giger, P. L. McCarty, 1985, GROUND WATER QUALITY.

23. 센서와 주변회로, 도서출판 세원 (1988)

24. 농어촌진흥공사, 1995, 지하수법령해설집, 197p.

25. 일본자동화기술, 센서기술, 도서출판 세화(1990)

26. 정형재, 1996, 순간수위변화시험에 의한 수리전도도 산출, 농공기술, 51, 55-65

27. 최병수, 1996, 자유면 대수층지역에서 지하수위 변동자료 해석에 의한 대수층 특성연구, 농공기술, 51, 3-13

부 록

여 백

부 록 1

지하수 관측기 사용설명서

1. 기기의 사양

1) CPU : Dallas사의 DS5002FP

2) 4 ~ 20mA의 4 Channel Analog input

16 Bits 상당의 A/D Converter(±20000 Counts)

3) 전원 : +12VDC, 2.5Ah 축전지

① 동작할 때 : 49.6 ~ 55 mA

② Sleep Mode 시 : 40 ~ 50 μA

③ Back Light 사용 시 : 67 ~ 68 mA

4) Real Time Clock(DS1305E) 내장

5) Memory 사양 :

① Program Memory : 64K Bytes

② Data Memory : 64K Bytes

6) Analog Input 사양 :

① Depth : 0 ~ 100.00 m

② EC : 저농도(0~5,000uS/cm), 고농도(0~50,000 μS/cm)

③ 온도 : -10.0℃ ~ +50.0℃

7) Data 저장 능력 : 5000번 측정치

저장내용 : 월, 일, 시, 분, 초, Depth, EC, Temp, pH

8) RS-232C 통신 : 9600, N, 8, 1

내용 : ① Data 요청 - 5000회분 전부 호출

- 현재 Data 호출

- 5회분 Data 호출

② 시간 정보 수정

③ Data 저장 시간조정

④ Power시 Data 저장 간격 조정 ⑤ 저장된 Data 전부 Clear

9) 2 Point Data 조정 방식

10) 20 X 2 Line LCD 채택

11) Key 사용 시 Buzzer 작동

12) 간단한 4 Key 조작 방식

2. 기기의 동작

1) 본 기기는 자동으로 On, Off되는 장비로 Battery의 효율을 최대한 활용하였다.

2) 기기 전원을 넣고 On 시키고 Off 시키는 방법

① Battery를 연결한다.

② Reset S/W를 누른다.

③ 측정하고 있는 Message가 나오면 Shift Key를 누르면 전원이 Off되지 않고 사용할 수 있다.

④ Off 시키는 방법은 Up Key를 눌러 Off 시킨다.

3) 자동으로 On, Off시의 동작

① 자동으로 On 되는 시간은 Calibration Mode 중 Save Time of Data Mode 에서 조정할수 있으며 Data를 수집하고 나면 Off 된다.

Water Anaysis
Water Pollution

Data Gathering
Equipment

② 상기 과정은 MPU를 초기화하는 과정이고 초기화가 끝나면 Data를 수집하게 된다.

Data Gathering Equip
Depth : 082.20m

Data Gathering Equip
EC : 09023uS/cm

Data Gathering Equip
Temp : +025.6℃

③ 상기의 순서대로 Display되고 온도까지 측정이 끝나면 Sleep Mode로 들어가서 다음 Data를 수집할 시간까지 기다린다.

4) 수동으로 측정하는 방법

① Shift Key를 누르고 전원을 On 시킨후 사용한다.

② Off 시키는 방법은 Up Key를 눌러 Off 시킨다.

③ 다음 Massage가 나오면 Calibration Mode의 전원 On 상태 저장 시간 간격 조정 (Save Time at Power On)에서 Setting한 시간 간격에 의해 3)의 ②과정을 실행한다.

Water Analysis Dev.
2001/05/18 15:36 23s

④ Mode Key를 눌렀다 때면 Depth 측정 Mode가 된다.

Data Gathering Equip
Depth : 082.20m

⑤ 여기서 Save Key를 누르고 있으면 A/D Converter의 Data가 직접 Display 된다. 수동으로 측정하는 Mode 모두가 이 방식을 채택하고 있으므로 언제 든지 Save Key를 눌러 A/D Converter의 값을 확인할 수 있다.

Data Gathering Equip
ORIGIN A/D : +12340

⑥ Mode Key를 눌렀다 때면 EC 측정 Mode가 된다.

Data Gathering Equip
EC : 09023uS/cm

⑦ Mode Key를 눌렀다 때면 온도 측정 Mode가 된다.

Data Gathering Equip
Temp : +025.6℃

⑧ Mode Key를 눌렀다 때면 pH 측정 Mode가 된다.

이 Mode는 Option으로 현재는 Channel 4의 A/D Converter의 값을 Display 한다.

Data Gathering Equip
pH : +00000

⑨ Mode Key를 눌렀다 때면 ③의 측정 대기 Mode가 된다.

Water Analysis Dev.
2001/05/18 15:36 23s

⑩ Key 사용을 1분 이상 사용하지 않으면 LCD의 Back Light가 Off된다.

다시 Key를 사용하면 LCD의 Back Light는 On된다.

5) Serial(RS232C) 통신

① Baud Rate : 9600, N, 8,1

② 명령어

현재 Data 요청												
시작	명령어							Check Sum				종료
STX	P	R	E	S	E	N	T	0	0	0	0	ETX

전체 Data 요청												
시작	명령어							Check Sum				종료
STX	R	E	Q	F	U	L	L	0	0	0	0	ETX

시계 수정																						
시작	명령어							년		월		일		시		분		Check Sum				종료
STX	M	O	D	I	F	Y	t	0	1	0	7	1	2	2	3	4	5	0	0	0	0	ETX

Data 저장 간격 조정																						
시작	명령어							시		분								Check Sum				종료
STX	D	S	I	T	I	M	E	0	0	0	2	0	0	0	0	0	0	0	0	0	0	ETX

전체 Data Clear												
시작	명령어							Check Sum				종료
STX	C	L	S	D	A	T	A	0	0	0	0	ETX

5회분 Data 요청												
시작	명령어							Check Sum				종료
STX	F	I	V	E	D	A	T	0	0	0	0	ETX

여기서 STX는 16진수 0x02이고 ETX는 16진수 0x03이다.

또한 청색은 명령어로 영문자이고 적색은 숫자이다(ASCII문자로 표현).

③ 전체 Data 요청(5000회분 Data) 명령에 대한 응답

Data는 RS232 Port로 내 보내면서 LCD창에는

Data transmit by
Data Request of PC

의 Message가 뜬다.

5000회분의 Data는 다음과 같다.

02 50 45 53 44 41 54 41 FF FF : Start 예약어

06 1C 17 34 0C 03 37 05 0B 04 4E : 1회분 Data (HEX값으로 표시)

06 1C 17 36 0C 03 37 04 ED 04 4E : 2회분 Data

06 1C 17 38 0C 03 37 05 0B 04 4E : 3회분 Data

06 1C 17 3A 0C 03 37 05 0B 04 4E : 4회분 Data

.

.

.

06 1D 00 00 0C 03 3A 05 0E 04 51 : 5000회분 Data

FF FE 00 00 00 00 03 : END 예약어

STX (0x02)	P	E	S	D	A	T	A	예약어	
02	50	45	53	44	41	54	41	FF	FF

날자 및 시간 정보					Depth		EC		온도	
6월	28일	23시	52분	12초	8.23m		1291uS/cm		110.2℃	
06	1C	17	34	0C	03	37	05	0B	04	4E

예약어		Check SUM				ETX (0x03)
FF	FE	00	00	00	00	03

적색은 모두 16진수(HEX값)임

총 Bytes수 :--

11 X 5000 = 55,000 Bytes

60,000 + 10(Start 예약어) Bytes + 7(END 예약어) Bytes

= 55,017 Bytes

㉠ 월 : char형 (예 : 0x06 --> 6월)

㉡ 일 : char형 (예 : 0x1C --> 28일)

㉢ 시 : char형 (예 : 0x17 --> 23시)

㉣ 분 : char형 (예 : 0x34 --> 52분)

㉤ 초 : char형 (예 : 0x0C --> 12초)

㉥ Depth : int형

(예 : 0x0337 --> 823 소수점은 끝에서 두 번째로 약속함, 즉 8.23m)

㉦ EC : int형

(예 : 0x050B --> 1291uS/cm)

㉧ Temp(온도) : int형

(예 : 0x044E --> 1102 소수점은 끝에서 첫자리로 약속함, 즉 110.2℃)

④ 현재 Data 요청 명령에 대한 응답

시작	예약어				Depth(m)							EC(uS/cm)					
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
STX(0x02)	D	C	T	P	+	1	0	0	.	0	0	+	1	0	0	0	0

Temp(온도 ℃)						Channel 4의 A/D값						Check Sum				종료
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
+	0	5	1	.	6	+	1	2	3	4	5	0	0	0	0	ETX(0x03)

ASCII Code값

⑤ 시계 수정 요청 명령에 대한 응답

상기 명령에서 빠진 초(Second)는 00으로 Setting된다.

LCD창에는 다음과 같이 뜬다.

Water Analysis Dev.
Setting Time & Date

⑥ Data 저장시간 간격 명령에 대한 응답

최소 분 단위까지 저장가능하며 LCD창에는 다음과 같이 뜬다.

Water Analysis Dev.
Inteval of Save Data

⑦ Ram Clear(Data 삭제) 명령에 대한 응답

"Ram Clear"를 하면 저장된 모든 Data가 지워진다.

새로운 Data 저장은 처음부터 다시 이루어지고 역시 5000회 분량의 Data가 모두 저장되고 나면 처음부터 쓰여진다.

또한 LCD창에는 다음과 같이 뜬다.

Old Data Clear
Initialling

⑧ 5회분 Data 요청 명령에 대한 응답

5000회분의 Data중 단 5회분만 송신하며 LCD창에는 다음과 같이 뜬다.

Data transmit by
Request of PC 5times

6) Calibration Mode

(1) Calibration Mode에는 Depth, EC, Temp, 저장 시간 간격 조정, 전원 On 상태의 저장 시간 간격 조정, 날짜 및 시간 조정이 있다.

㉠ Depth, EC, Temp는 2 Point 조정 방식에 의해 결정한다.

Low값과 High값을 결정하여 그때의 A/D Converter값과 Setting값을 기억하여 환산한 값을 Display한다.

㉡ 저장 시간 간격 조정은 자동으로 전원을 On, Off 하기 위해서 시계 IC(DS1305)에 Setting하는 값이다.

전부 00으로 Setting하면 초당 On, Off 하고 초 단위만 Setting하면 초(Second)가 일치했을 때 On, off 한다. 즉 매분마다 동일한 초에 On,Off하는 것이다.

이런 방식으로 분 단위 이하를 Setting하면 시간 단위 On, Off 이며 또한 시간단위 이하를 Setting하면 매일 같은 시간 같은 분 같은 초에 On, Off 한다.

㉢ 전원 On 상태의 저장 시간은 System이 항시 On 상태에서 Data를 저장하는 시간간격이다. 최소 분단위로 할 수 있고 ㉡과 다른 점은 Data가 5

분이라면 5분 간격으로 Data를 저장한다는 것이다.

㉣ 날짜 및 시간 조정은 년, 월, 일, 시간, 분, 초를 수정 가능하다.

㉤ Calibration Mode는 측정 Mode에서 Mode Key를 4초 동안 누르고 있으면 들어올수 있다.

㉥ 측정 Mode로 나갈 때는 다음 Message에서 Shift Key와 Save Key를 동시에 누르면 Calibration Mode를 나가 측정 Mode로 간다.

Calibration Mode
Depth, Ec, Temp, pH

㉦ Calibration Mode에서는 Shift Key는 우측으로 이동하는 자리 이동용 key이다.

가장 우측에 있을 때는 좌측 첫자리로 이동한다.

현재 자리 구분은 현 위치 숫자가 깜박이므로 위치 파악이 쉽다.

㉧ Up Key는 현 위치의 숫자를 증가시키는 역할을 한다.

가장 상위 숫자면 0(Zero)부터 다시 증가한다.

Calibration Mode
Ec : 00000uS/cm

㉨ Save Key는 현재 교정된 값을 저장하는 기능이다.

Calibration Mode
--- Save ---

⑵ Depth Low Value Calibration

㉠ 먼저 High와 Low 지점을 정하여 조정한다.

㉡ Mode Key에 의해 이 Mode로 들어올 수 있다.

㉢ 먼저 Low Value를 교정한다.

Cal. Depth : +03403
Low Value : 009.30 m

㉣ 상기 Cal. Depth : +03403은 Analog Channel 1의 A/D Converter 값이다. Low Value : 009.30m는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

Cal. Depth : +03403 과 Low Value : 009.3m 모두 저장됨.

(3) Depth High Value Calibration

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Cal. Depth : +16450
High Value : 094.70m

㉡ 상기 Cal. Depth : +03403은 Analog Channel 1의 A/D Converter 값이다. High Value : 094.70m는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 High Value 교정은 완료된다.

(4) EC Low Value Calibration

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Cal. Ec : +02348
Low : 00032 uS/cm

㉡ 상기 Cal. Ec : +02348은 Analog Channel 2의 A/D Converter 값이다.

Low Value : 00032uS/cm는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(5) EC High Value Calibration

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Cal. Ec : +16730
High : 09078 uS/cm

㉡ 상기 Cal. Ec : +16730은 Analog Channel 2의 A/D Converter 값이다.

Low Value : 09078uS/cm는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(6) Temp(온도) Low Value Calibration

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Cal. Temp : +03460
Low Value : -12.5℃

㉡ 상기 Cal. Temp : +03460은 Analog Channel 3의 A/D Converter 값이다.

Low Value : -12.5℃는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(7) Temp(온도) High Value Calibration

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Cal. Temp : +17921
High Value : +48.5℃

㉡ 상기 Cal. Temp : +17921은 Analog Channel 3의 A/D Converter 값이다.

High Value : +48.5℃는 센서로 실제 측정한 값이다. 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(8) 저장 시간 간격 조정

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Time of Save Data
00H : 00M 00S

㉡ 저장 시간 간격 조정은 자동으로 전원을 On, Off 하기 위해서 시계 IC(DS1305)에 Setting하는 값이다.

전부 00으로 Setting하면 초당 On, Off 하고 초 단위만 Setting하면 초(Second)가 일치했을 때 On, off 한다. 즉 매분 동일한 초에 On, Off 하는 것이다.

이런 방식으로 분 단위 이하를 Setting하면 시간 단위 On, Off 이며 또한 시간단위 이하를 Setting하면 매일 같은 시간 같은 분 같은 초에 On, Off 한다.

이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(9) 전원 On 상태의 저장 시간 간격 조정

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

SaveTime at Power On
00:00

㉢ 전원 On 상태의 저장 시간은 System이 항시 On 상태에서 Data를 저장하는 시간 간격이다. 최소 분단위로 할 수 있고 (8)과 다른 점은 Data가 5분이라면 5분 간격으로 Data를 저장한다는 것이다.

이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

(10) 날짜 및 시간 조정

㉠ Mode Key에 의해 이 Mode로 들어올 수 있다.

Setting Time & Date
2001/05/18 15:36 23s

㉡ 이것을 Shift Key와 Up Key를 사용하여 입력한다. 현재의 값을 측정 Data의 기초로 사용하려면 Save Key를 사용하여 Save 시키면 Low Value 교정은 완료된다.

㉢ 년도의 2001년 중 앞의 두 자리 20은 수정되지 않음.

㉣ 교정이 완료되면 이 Mode에서 Mode Key를 눌러 다음 Massage가 나오면 Shift Key와 Save Key를 동시에 누르면 Calibration Mode를 나가 측정 Mode로 간다.

Calibration Mode
Depth, Ec, Temp, pH

7) 전체 Ram Clear(Data 삭제) 방법

Ram Clear를 하면 저장된 모든 Data가 지워진다.

새로운 Data 저장은 처음부터 다시 이루어지고 역시 5000회 분량의 Data가 다 저장되고 나면 처음부터 쓰여진다.

또한 LCD창에는 다음과 같이 뜬다.

Old Data Clear
Initialling

System Power를 On 시킨 후 (Reset Key를 누르고(Data가 나타날 때까지 누름) Shift Key를 사용 On 시킴) Ram Clear를 하려면 Shift Key를 5초 동안 누르고 있으면 Ram Clear를 시킬 수 있다.

부 록 2

<전기전도도센서 부품목록>

번호	구분	부품명	도면 표시	규격
1	저항	1.5K	R37	SMR2012
2	〃	1.5K	R36	SMR2012
3	〃	1.6K	R8	SMR2012
4	〃	1.6K	R9	SMR2012
5	〃	1K	R33	SMR2012
6	〃	1K	VR1	SMVR3314G
7	〃	1M	R34	SMR2012
8	〃	1M	R41	SMR2012
9	〃	1M	R42	SMR2012
10	〃	3.3K	R2	SMR2012
11	〃	3.3K	R27	SMR2012
12	〃	4.7K	R4	SMR2012
13	〃	5.1K	R48	SMR2012
14	〃	5.6K	R47	SMR2012
15	가변저항	5K	VR4	SMVR3314G
16	저항	7.5K	R5	SMR2012
17	〃	8.2K	R29	SMR2012
18	콘덴서	10/16V	C6	SMC1AB
19	〃	10/16V	C4	SMC1AB
20	〃	10/16V	C5	SMC1AB
21	〃	10/16V	C9	SMC1AB
22	〃	10/16V	C10	SMC1AB
23	〃	10/16V	C7	SMC1AB
24	〃	10/16V	C8	SMC1AB
25	〃	10/16V	C2	SMC1AB
26	〃	10/16V	C3	SMC1AB
27	저항	10K	R46	SMR2012
28	〃	10K	R12	SMR2012
29	〃	10K	R17	SMR2012
30	〃	10K	R16	SMR2012
31	〃	10K	R19	SMR2012

<전기전도도센서 부품목록>

번호	구분	부품명	도면 표시	규격
32	저항	10K	R18	SMR2012
33	〃	10K	R15	SMR2012
34	〃	10K	R14	SMR2012
35	다이오드	IN4148	D10	SMD4148
36	〃	IN4148	D9	SMD4148
37	〃	IN4148	D5	SMD4148
38	〃	IN4148	D6	SMD4148
39	〃	IN4148	D7	SMD4148
40	〃	K30A	F1	SM2SK94
41	IC	LM10CWM	U2	SOL-14
42	콘덴서	M104	C19	SMR2012
43	〃	M104	C18	SMR2012
44	저항	10K	R13	SMR2012
45	〃	10K	R20	SMR2012
46	〃	10K	R21	SMR2012
47	〃	10K	R22	SMR2012
48	〃	12K	R7	SMR2012
49	가변저항	20K	VR2	SMVR3314G
50	콘덴서	22/16V	C17	SMC1AB
51	저항	22K	R30	SMR2012
52	〃	22K	R31	SMR2012
53	〃	22K	R32	SMR2012
54	〃	33K	R6	SMR2012
55	〃	36K	R35	SMR2012
56	〃	47K	R28	SMR2012
57	〃	47K	R1	SMR2012
58	〃	51K	R3	SMR2012
59	〃	51ohm	R43	SMR2012
60	〃	100K	R39	SMR2012
61	〃	100K	R40	SMR2012
62	가변저항	100K	VR3	SMVR3314G

<전기전도도센서 부품목록>

번호	구분	부품명	도면 표시	규격
63	콘데선	100P	C16	SMR2012
64	저항	100ohm	R24	SMR2012
65	〃	100ohm	R25	SMR2012
66	〃	100ohm	R23	SMR2012
67	〃	100ohm	R26	SMR2012
68	〃	180ohm	R10	SMR2012
69	〃	200K	R38	SMR2012
70	〃	220K	R11	SMR2012
71	〃	330K	R45	SMR2012
72	〃	470K	R44	SMR2012
73	IC	1458	U3	SOJ-8
74	〃	1458	U4	SOJ-8
75	〃	1458	U6	SOJ-8
76	〃	1458	U5	SOJ-8
77	〃	4047	U1	SOJ-14
78	〃	7660	U7	SO-8
79	콘덴서	C102	C14	SMR2012
80	〃	C102	C15	SMR2012
81	〃	C104	CX	SMR2012
82	〃	C332	C12	SMR2012
83	〃	C332	C11	SMR2012
84	〃	C332	C13	SMR2012
85	〃	C472	C1	SMR2012
86	커넥트	CON5	J1	MOLEX5
87	〃	HEADER 5X2	JP1	HIF3F10
88	다이오드	IN4148	D8	SMD4148
89	〃	IN4148	D4	SMD4148
90	〃	IN4148	D3	SMD4148
91	〃	IN4148	D1	SMD4148
92	〃	IN4148	D2	SMD4148

<수위센서 부품목록>

번호	구분	부품명	도면 표시	규격
1	저항	1.5K	R10	SMR2012
2	〃	1.8K	R12	SMR2012
3	〃	1.8K	R13	SMR2012
4	〃	5.1K	R28	SMR2012
5	〃	5.1K	R30	SMR2012
6	가변저항	5K	VR5	SMVR3314G
7	〃	5K	VR2	SMVR3314G
8	〃	5K	VR4	SMVR3314G
9	〃	5K	VR3	SMVR3314G
10	콘덴서	10/16V	C3	SMC1AB
11	〃	10/16V	C8	SMC1AB
12	〃	10/16V	C4	SMC1AB
13	저항	10K	R19	SMR2012
14	〃	10K	R11	SMR2012
15	〃	10K	R18	SMR2012
16	〃	10K	R17	SMR2012
17	〃	10K	R31	SMR2012
18	〃	10K	R25	SMR2012
19	〃	10K	R22	SMR2012
20	〃	10K	R23	SMR2012
21	〃	10K	R24	SMR2012
22	〃	16K	R20	SMR2012
23	〃	18K	R14	SMR2012
24	콘덴서	22/16V	C5	SMC1AB
25	저항	22K	R26	SMR2012
26	〃	24K	R27	SMR2012
27	〃	36K	R9	SMR2012
28	〃	47K	R29	SMR2012
29	〃	47K	R21	SMR2012
30	〃	47K	R15	SMR2012
31	〃	51K	R16	SMR2012

<수위센서 부품목록>

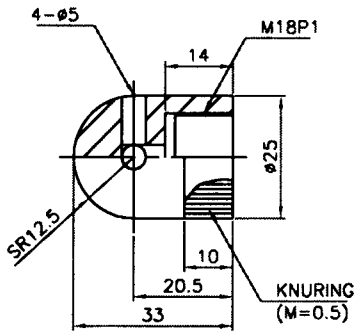
번호	구분	부품명	도면 표시	규격
32	저항	100K	R4	SMR2012
33	〃	100K	R2	SMR2012
34	〃	100K	R3	SMR2012
35	〃	100K	R1	SMR2012
36	가변저항	100K	VR1	SMVR3314G
37	저항	200K	R8	SMR2012
38	〃	330K	R5	SMR2012
39	〃	470K	R6	SMR2012
40	〃	510ohm	R7	SMR2012
41	IC	1458	U5	SOJ-8
42	〃	7660	U1	SO-8
43	스위치	CAP	CX1	SMR2012
44	커넥트	CON4	J2	MOLEX4
45	〃	CON5	J1	MOLEX5
46	다이오드	IN4002	D1	SMD4148
47	IC	LM10CWM	U2	SOL-14
48	콘덴서	M104	C1	SMR2012
49	〃	M104	C2	SMR2012
50	〃	M104	C7	SMR2012
51	〃	M104	C6	SMR2012
52	IC	TL022	U3	SOJ-8
53	〃	TL022	U4	SOJ-8

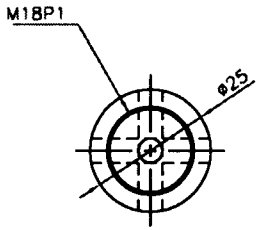
<온도센서 부품목록>

번호	구분	부품명	도면 표시	규격
1	저항	1.5K	R22	SMR2012
2	〃	1K	R4	SMR2012
3	〃	1K	R5	SMR2012
4	〃	5K	R14	SMR2012
5	가변저항	5K	VR2	SMVR3314G
6	저항	10.22K	R2	SMR2012
7	콘덴서	10/16V	C6	SMC1AB
8	〃	10/16V	C7	SMC1AB
9	저항	10K	R10	SMR2012
10	〃	10K	R8	SMR2012
11	〃	10K	R11	SMR2012
12	〃	12K	R1	SMR2012
13	〃	18K	R6	SMR2012
14	콘덴서	22/16V	C8	SMC1AB
15	〃	22/16V	C9	SMC1AB
16	저항	36K	R9	SMR2012
17	〃	36K	R21	SMR2012
18	〃	62K	R7	SMR2012
19	〃	100K	R15	SMR2012
20	〃	100K	R16	SMR2012
21	가변저항	100K	VR4	SMVR3314G
22	저항	100ohm	R3	SMR2012
23	〃	200K	R20	SMR2012
24	〃	330K	R17	SMR2012
25	〃	470K	R18	SMR2012
26	가변저항	500ohm	VR1	SMVR3314G
27	저항	510ohm	R19	SMR2012
28	IC	7660	U2	SO-8
29	스위치	CAP	CX	SMR2012
30	커넥트	CON4	J2	MOLEX4
31	커넥트	CON5	J1	MOLEX5

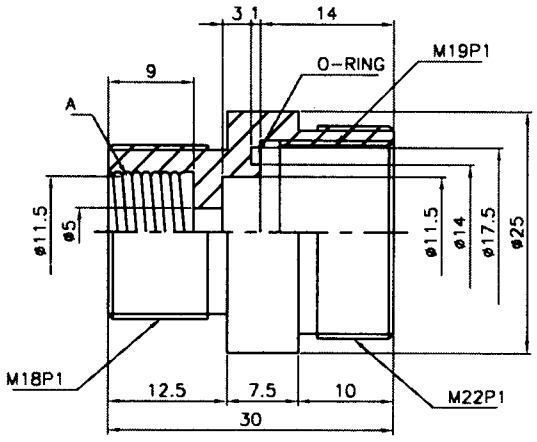
<온도센서 부품목록>

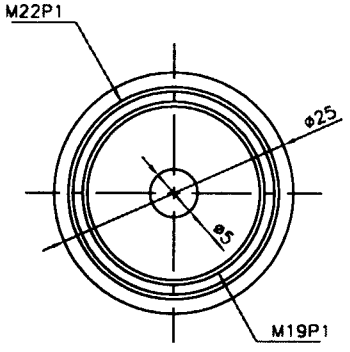
번호	구분	부품명	도면 표시	규격
32	다이오드	IN4002	D1	SMD4148
33	IC	LM10CWM	U3	SOL-14
34	콘덴서	M104	C1	SMR2012
35	〃	M104	C5	SB012
36	〃	M104	C4	SB012
37	〃	M104	C2	SB012
38	〃	M104	C3	SB012
39	IC	TL02	U1	SB
40	〃	TL02	U4	SB





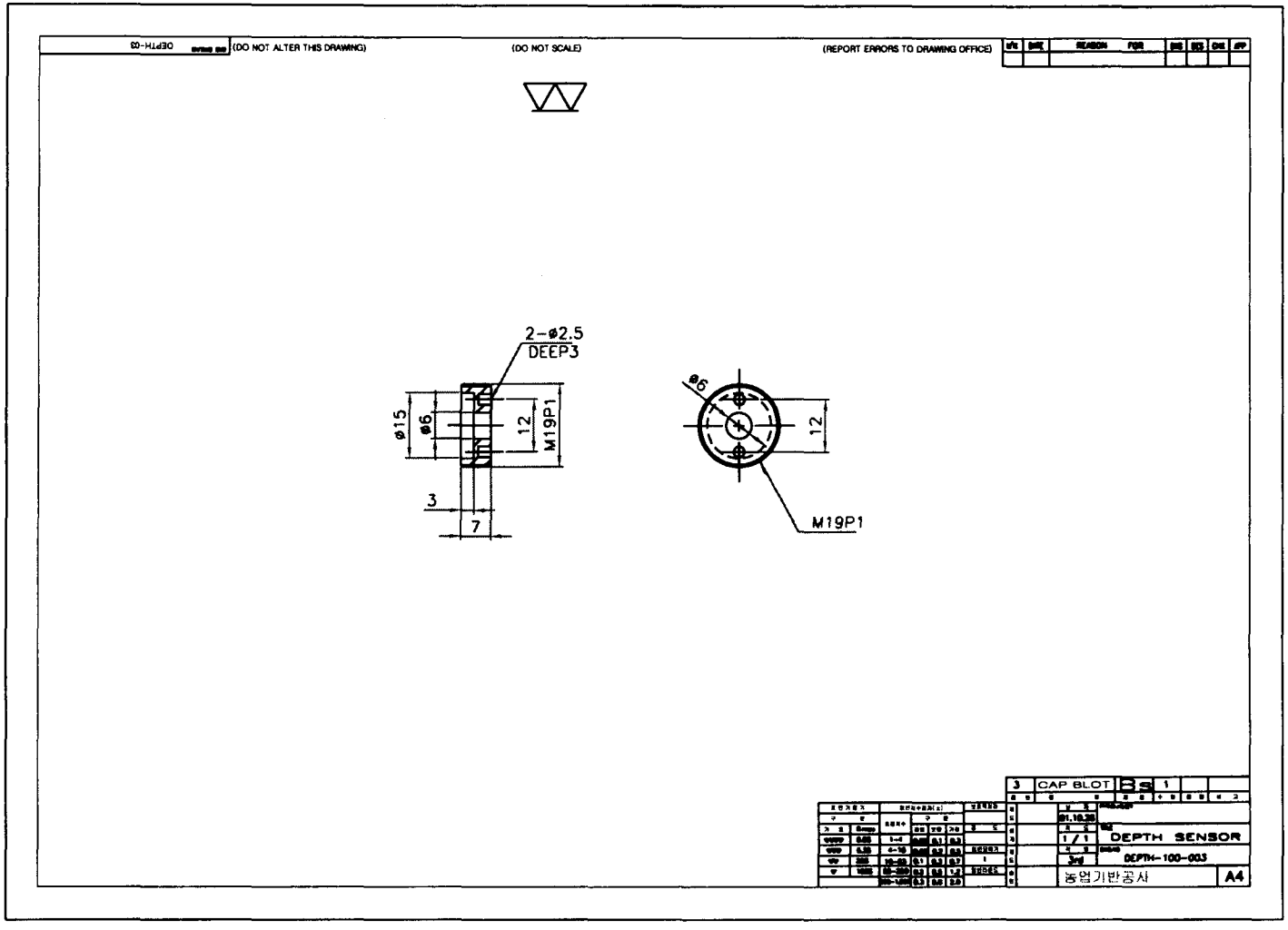
1	HEAD CAP	SUS316	1
DETPH SENSOR			
DETPH-100-001			
농업기반공사			A4

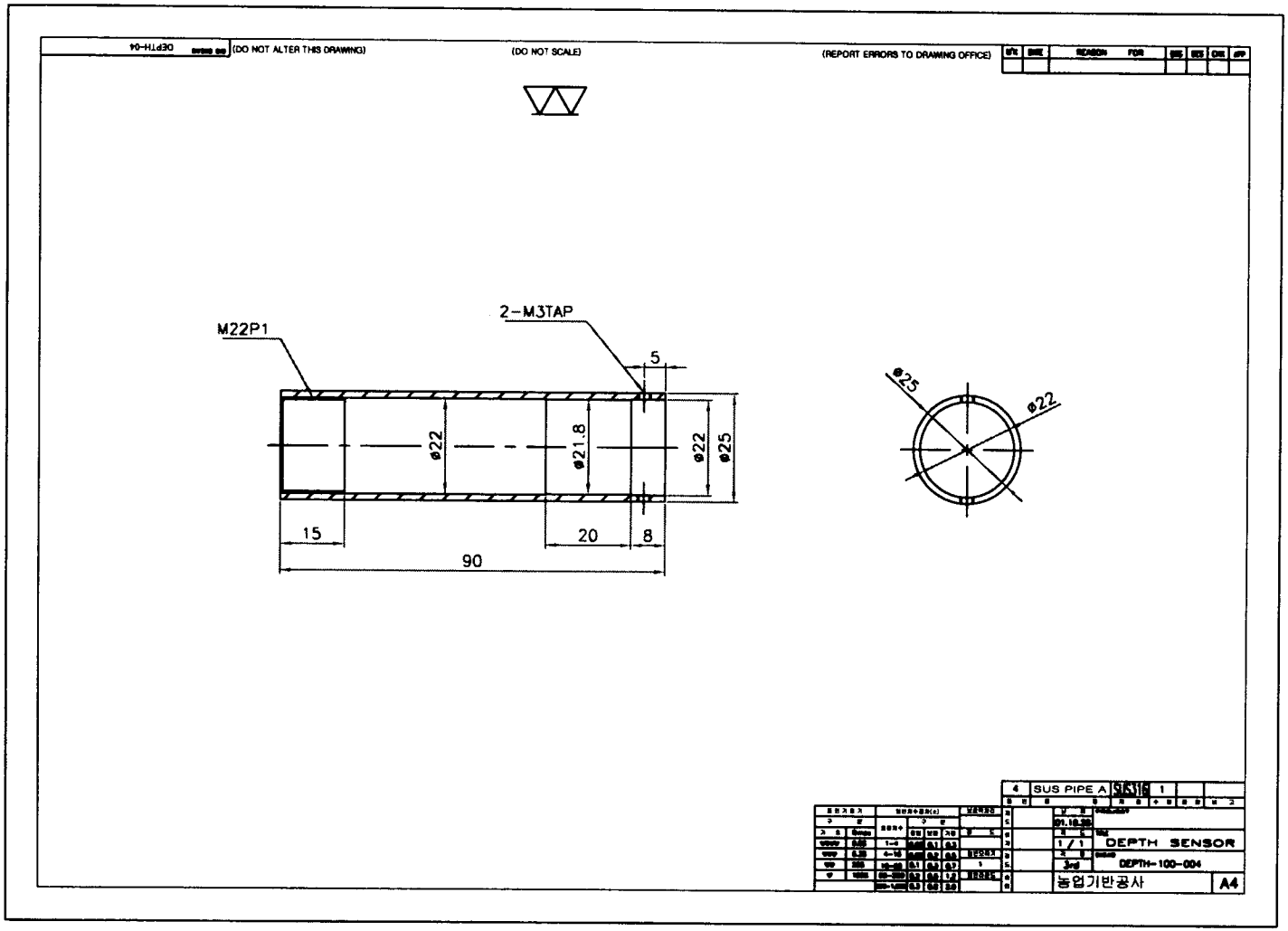


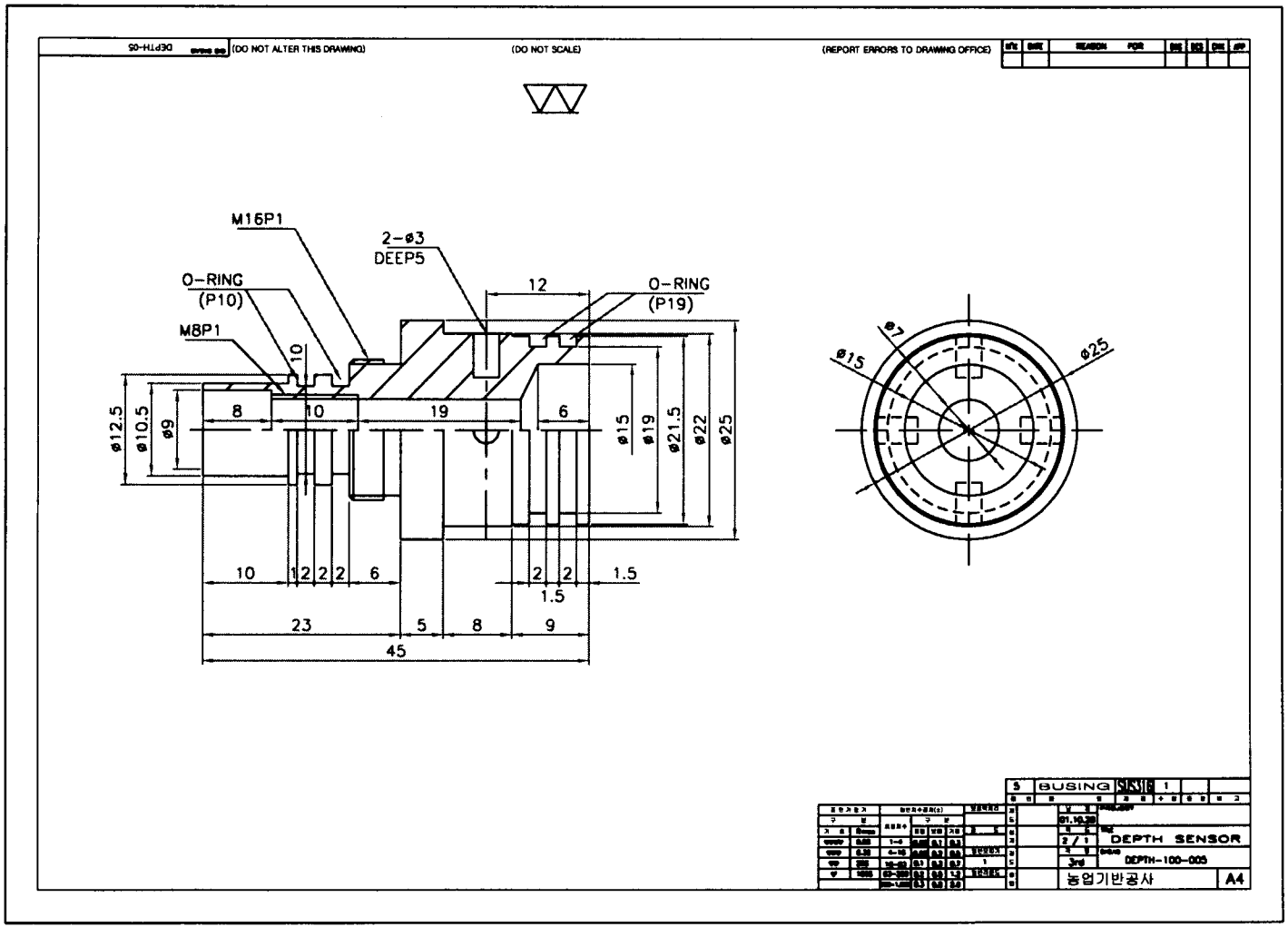


NOTE
1. A 부분은 스프링에 맞 추어서 가공할것.

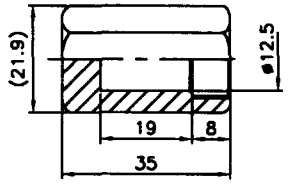
2	BUSING	SUS316	1	
			2 / 1	DEPTH SENSOR
			3rd	DEPTH-100-002
			농업기반공사	A4

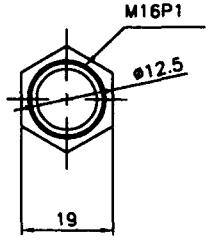

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
2-ø2.5
DEEP3
ø15
ø6
12
M19P1
3
7
ø6
12
M19P1
CAP BLOT
DEPTH SENSOR
DEPTH-100-003
농업기반공사
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M22P1
2-M3TAP
5
ø22
ø21.8
ø22
ø25
15
20
8
90
ø25
ø22
SUS PIPE A
DEPTH SENSOR
DEPTH-100-004
농업기반공사
A4

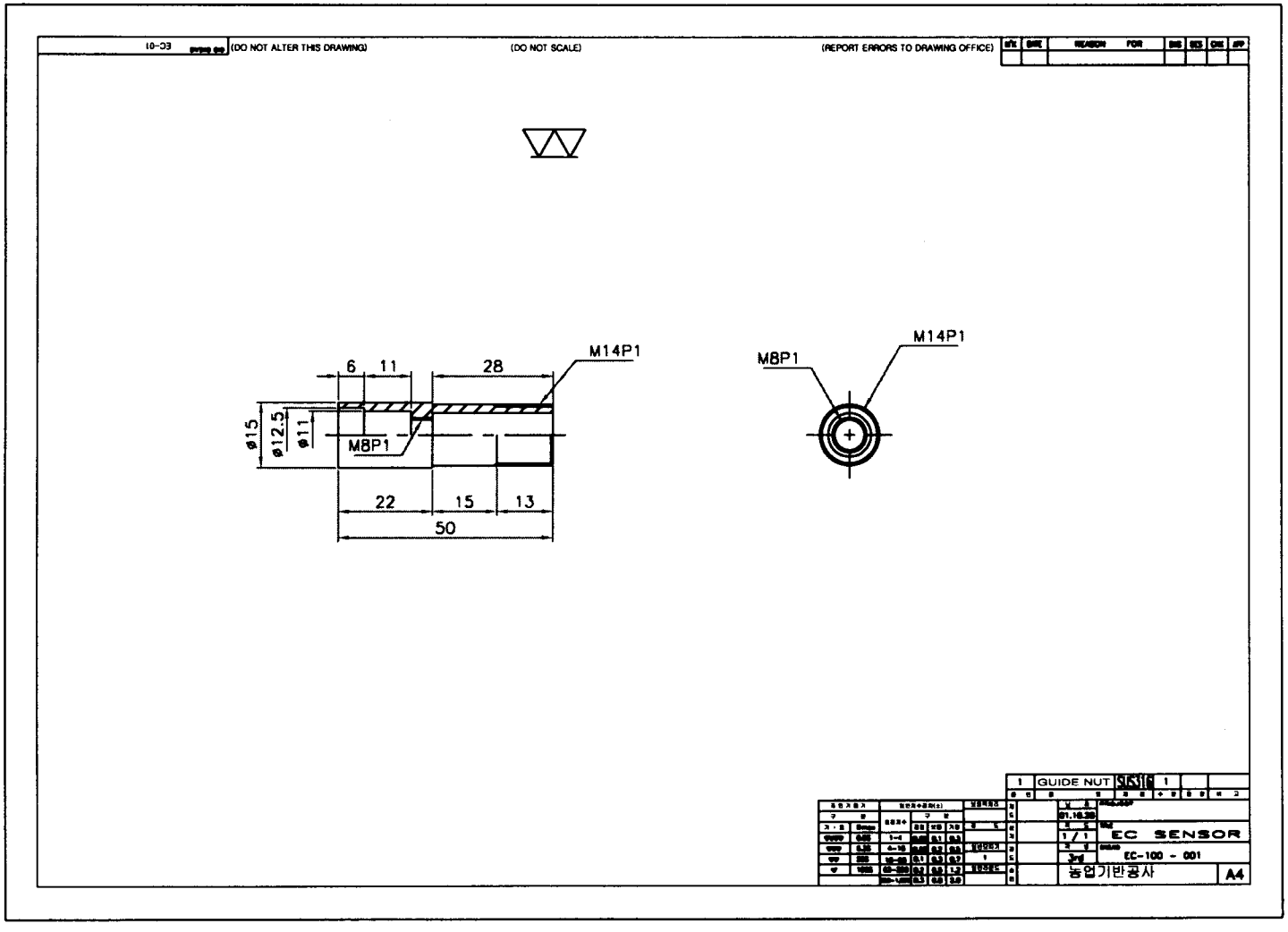

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M16P1
2-ø3
DEEP5
O-RING
(P10)
M8P1
O-RING
(P19)
12
10
ø12.5
ø10.5
ø9
8
10
19
6
ø15
ø19
ø21.5
ø22
ø25
10
12
2
2
6
2
2
1.5
1.5
23
5
8
9
45
ø7
ø15
ø25
BUSING
DEPTH SENSOR
DEPTH-100-005
농업기반공사
A4

(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)

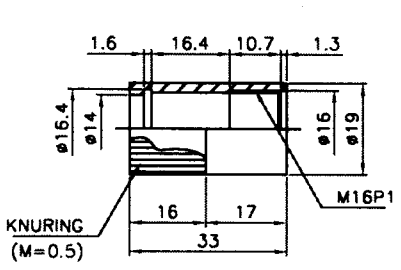


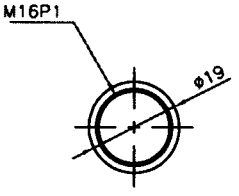


6	CAP BOLT	SCM	1
		DEPTH SENSOR	
		DEPTH-100-006	
	농업기반공사		A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
6
11
28
M14P1
ø15
ø12.5
ø11
M8P1
22
15
13
50
M8P1
M14P1
1
GUIDE NUT
EC SENSOR
EC-100 - 001
농업기반공사
A4

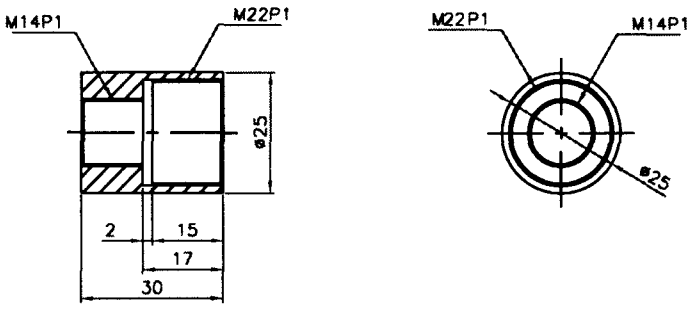
20-03 (DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)



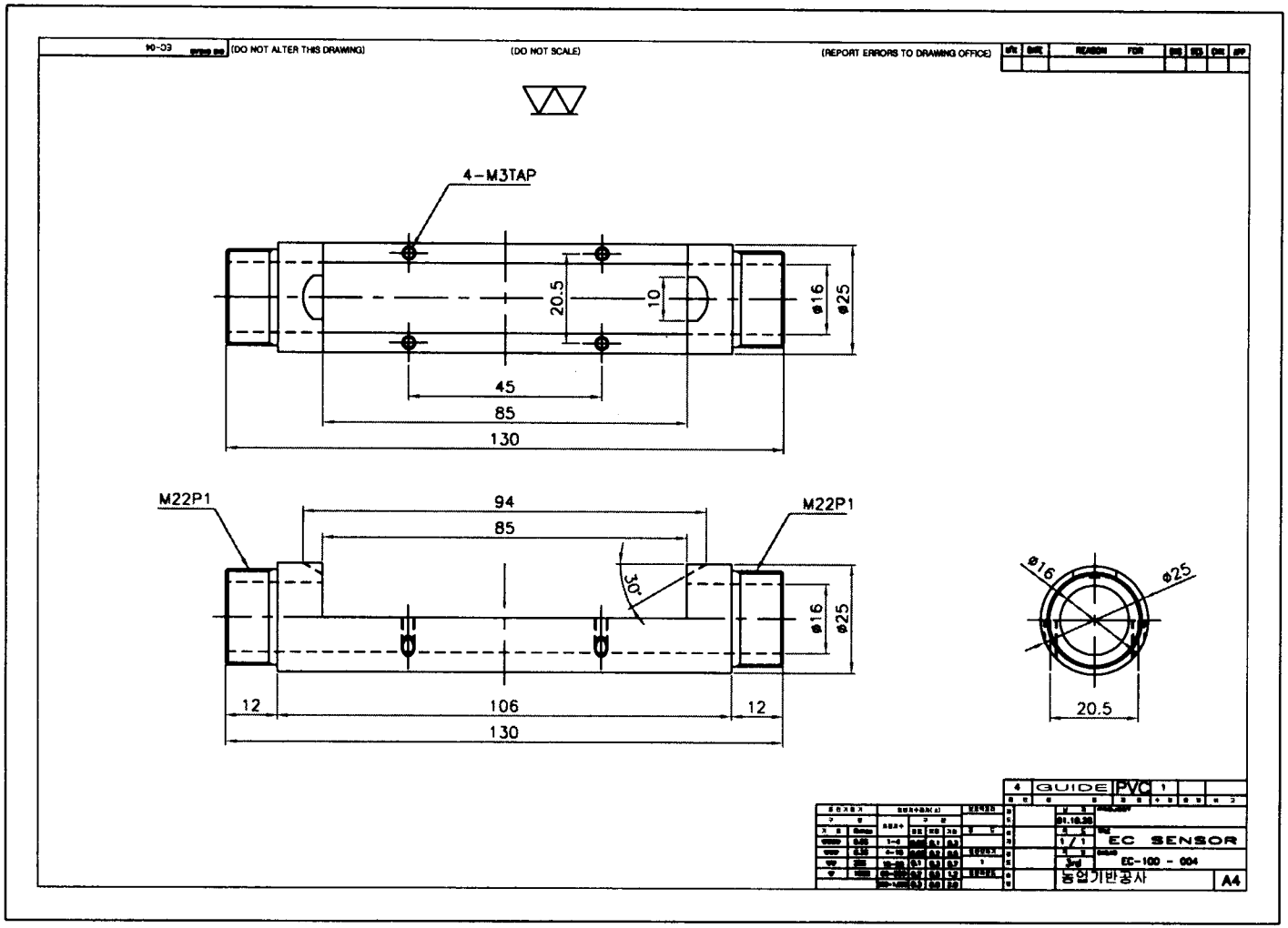


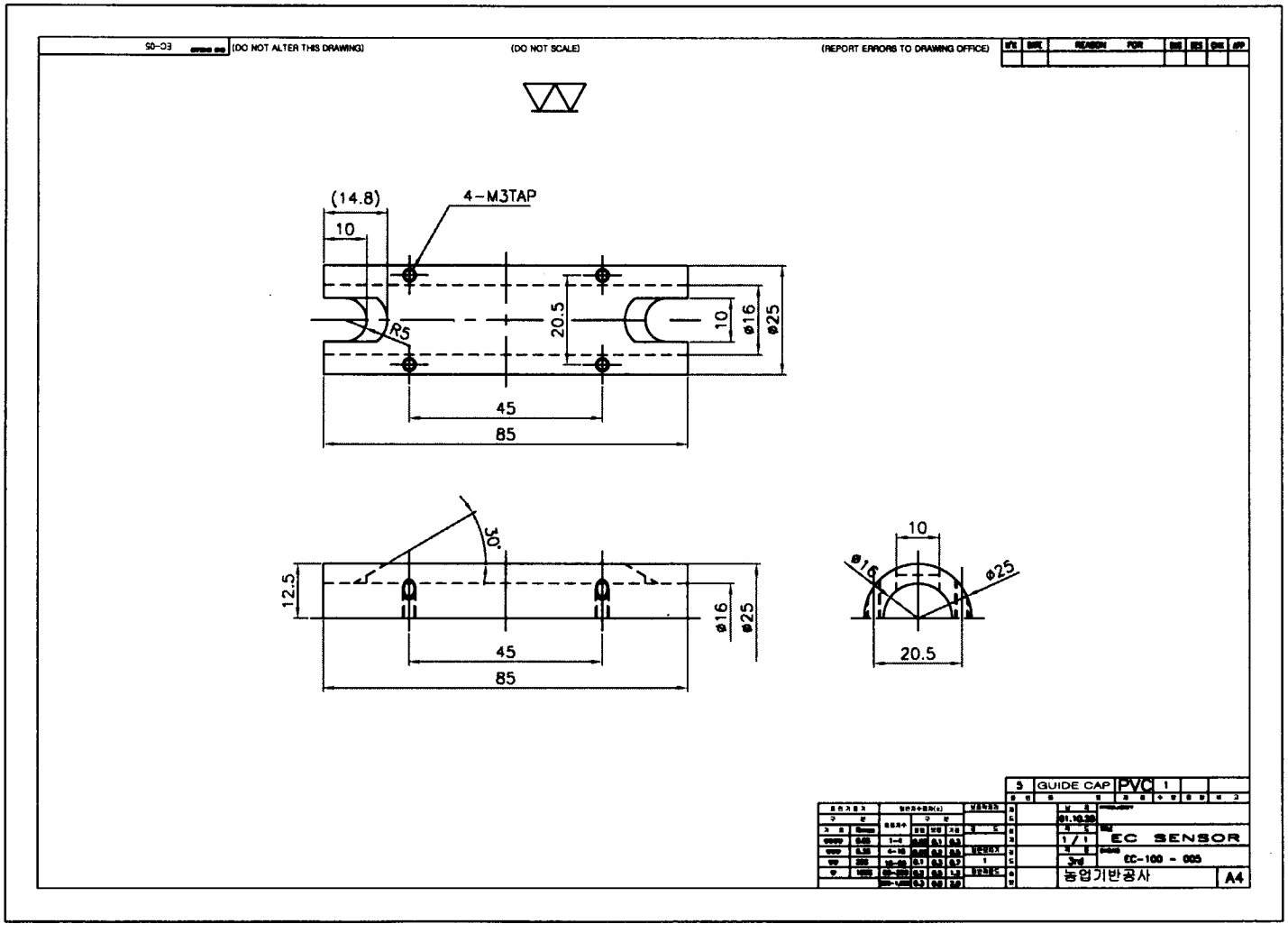
2	RING NUT	SUS316	1	

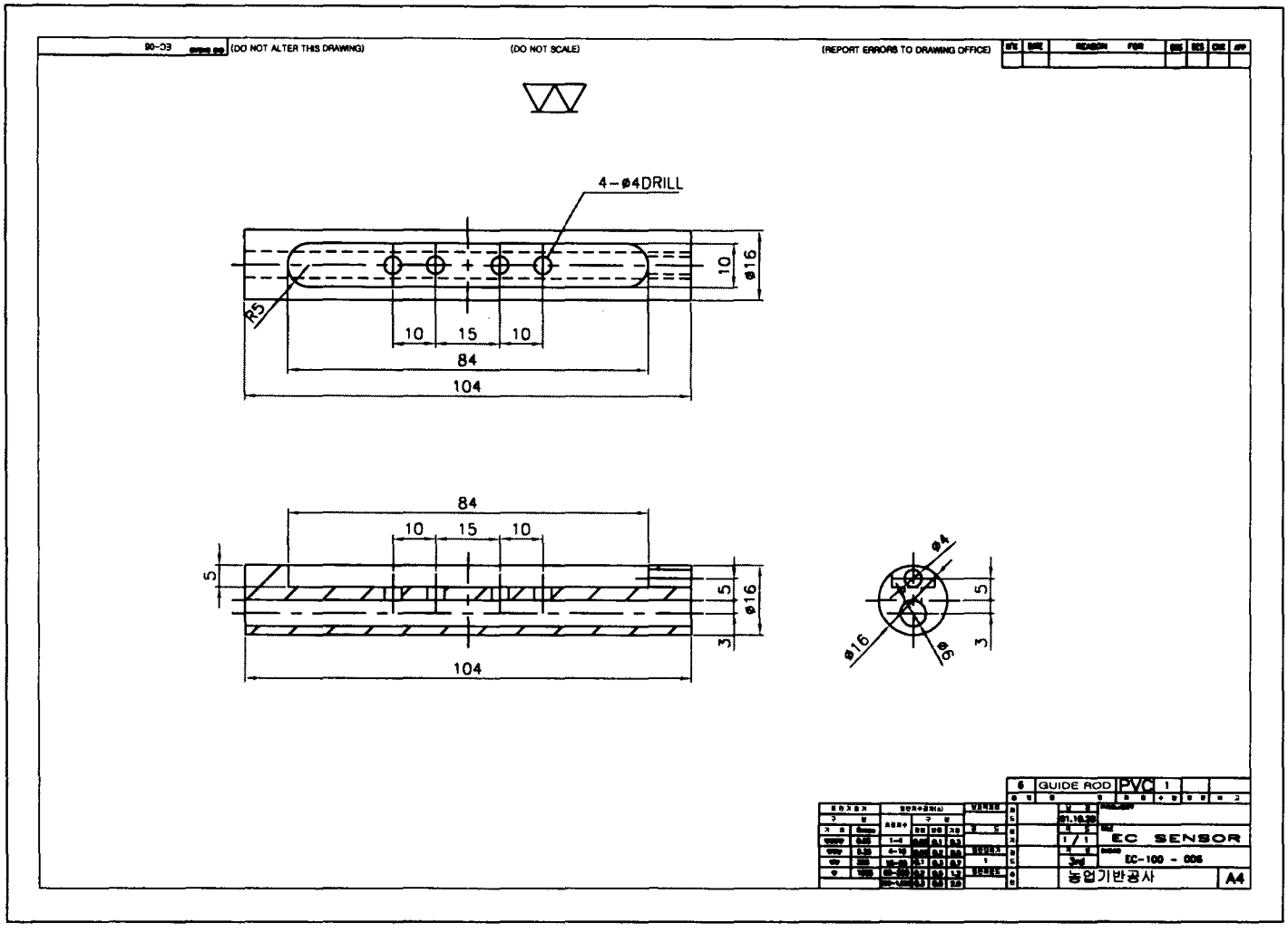
	EC SENSOR
	EC-100 - 002
	농업기반공사 A4



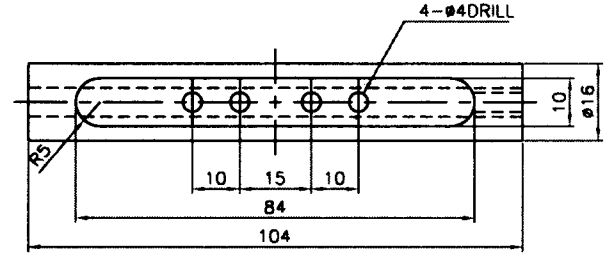
3	CAP BUSHING	SUS316	1		
	EC SENSOR				
	EC-100 - 003				
	농업기반공사			A4	

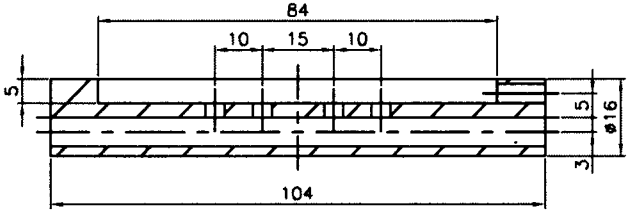

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
4-M3TAP
20.5
10
ø16
ø25
45
85
130
M22P1
M22P1
94
85
30°
ø16
ø25
12
106
12
130
ø16
ø25
20.5
GUIDE
PVC
EC SENSOR
EC-100 - 004
농업기반공사
A4

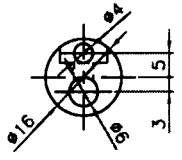

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
(14.8)
4-M3TAP
10
R5
20.5
10
ø16
ø25
45
85
30°
12.5
ø16
ø25
45
85
10
ø16
ø25
20.5
5
GUIDE CAP
PVC
1
EC SENSOR
EC-100 - 005
농업기반공사
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
4-ø4DRILL
R5
10
ø16
10
15
10
84
104
84
10
15
10
5
5
ø16
3
104
ø4
5
ø16
90
3
6
GUIDE ROD
PVC
1
EC SENSOR
EC-100 - 006
농업기반공사
A4

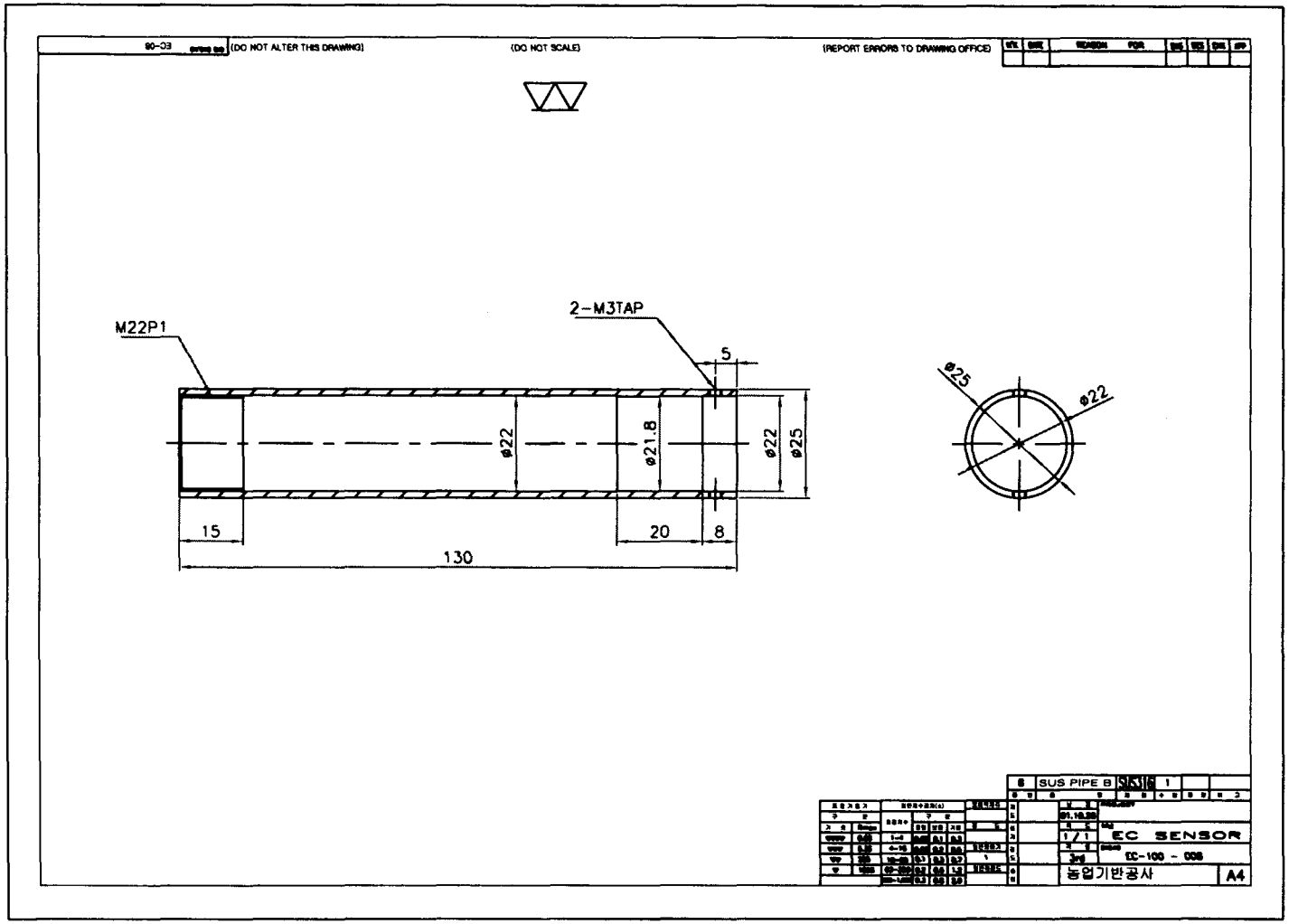
(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)

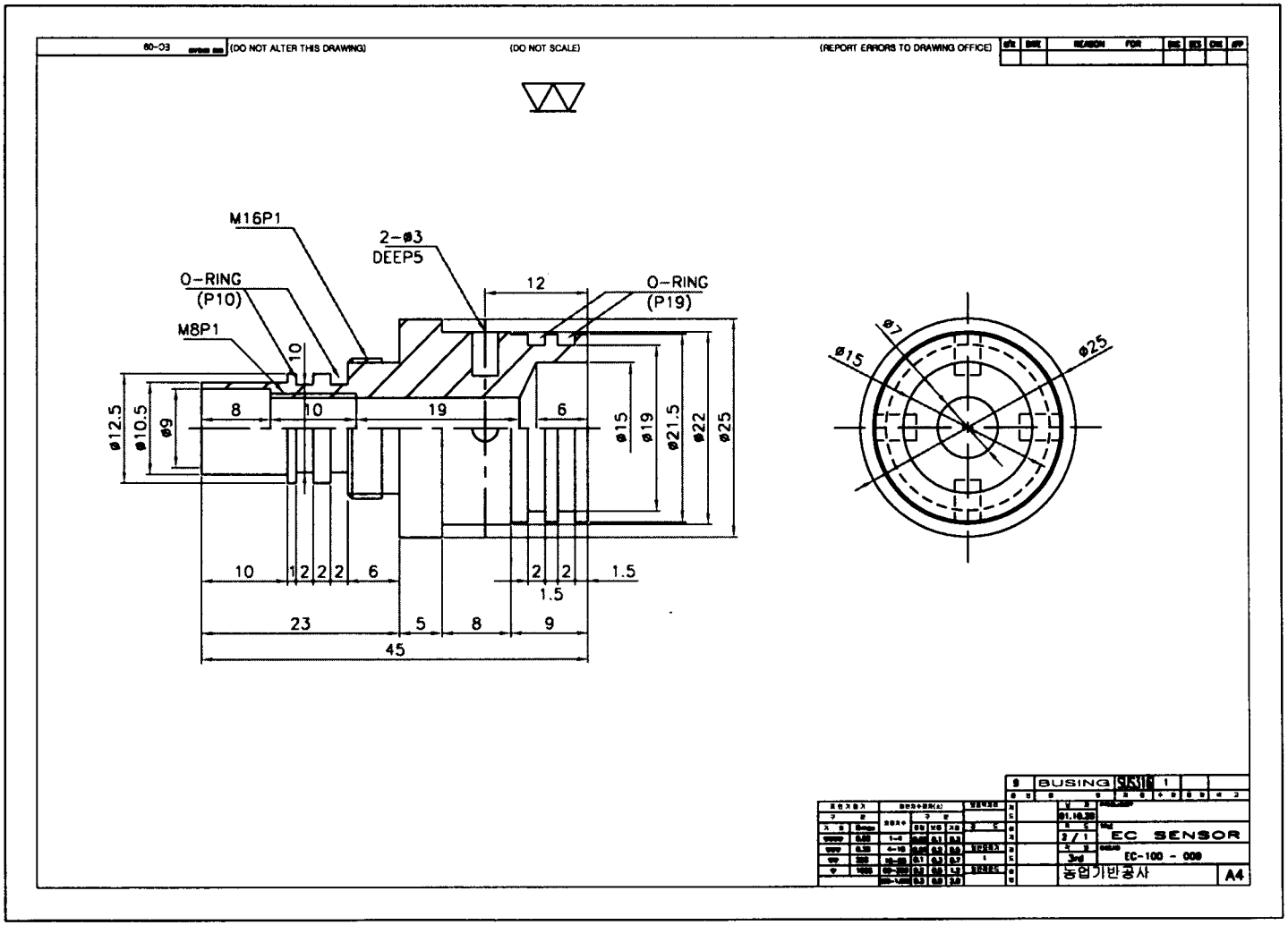


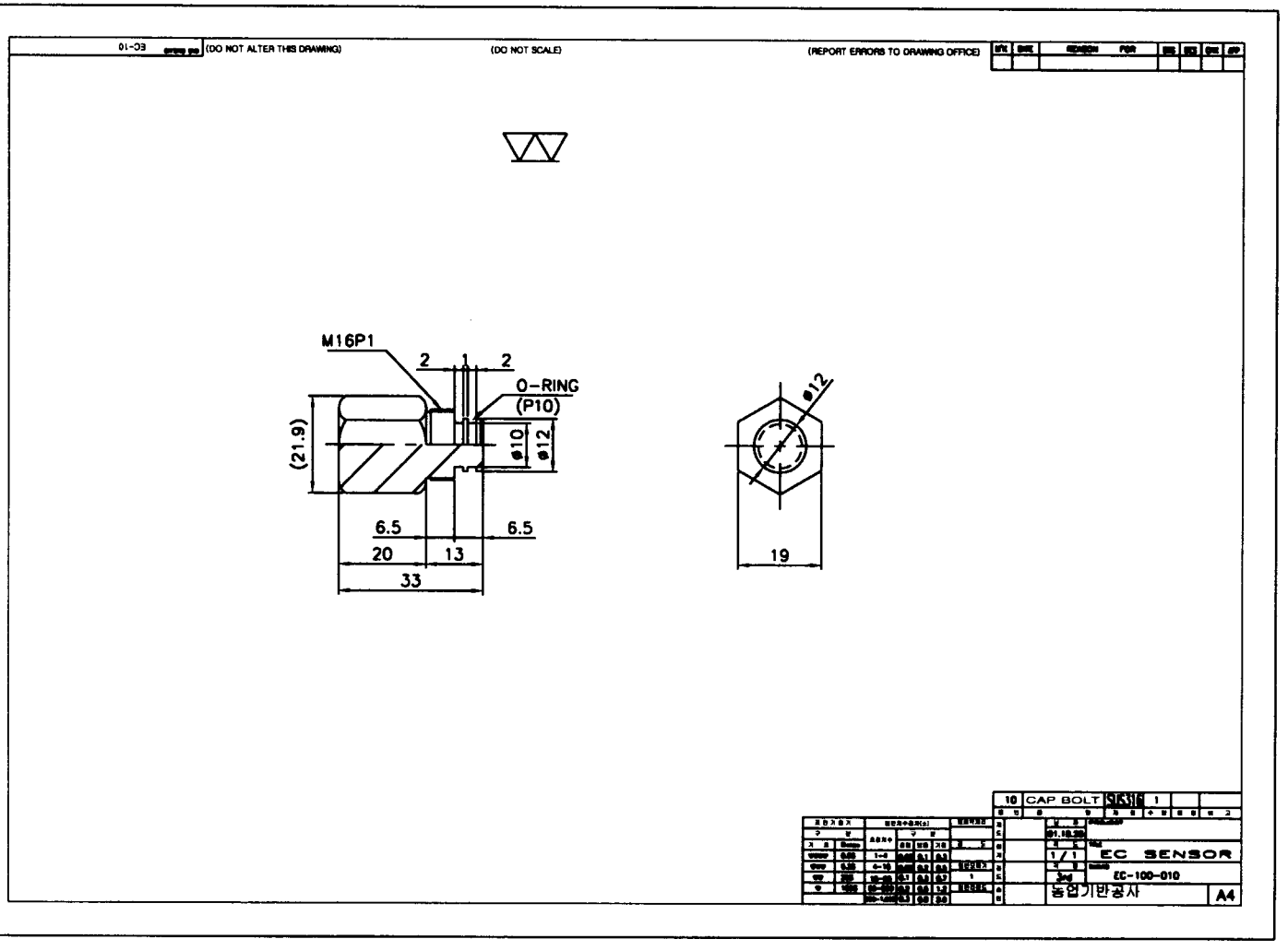


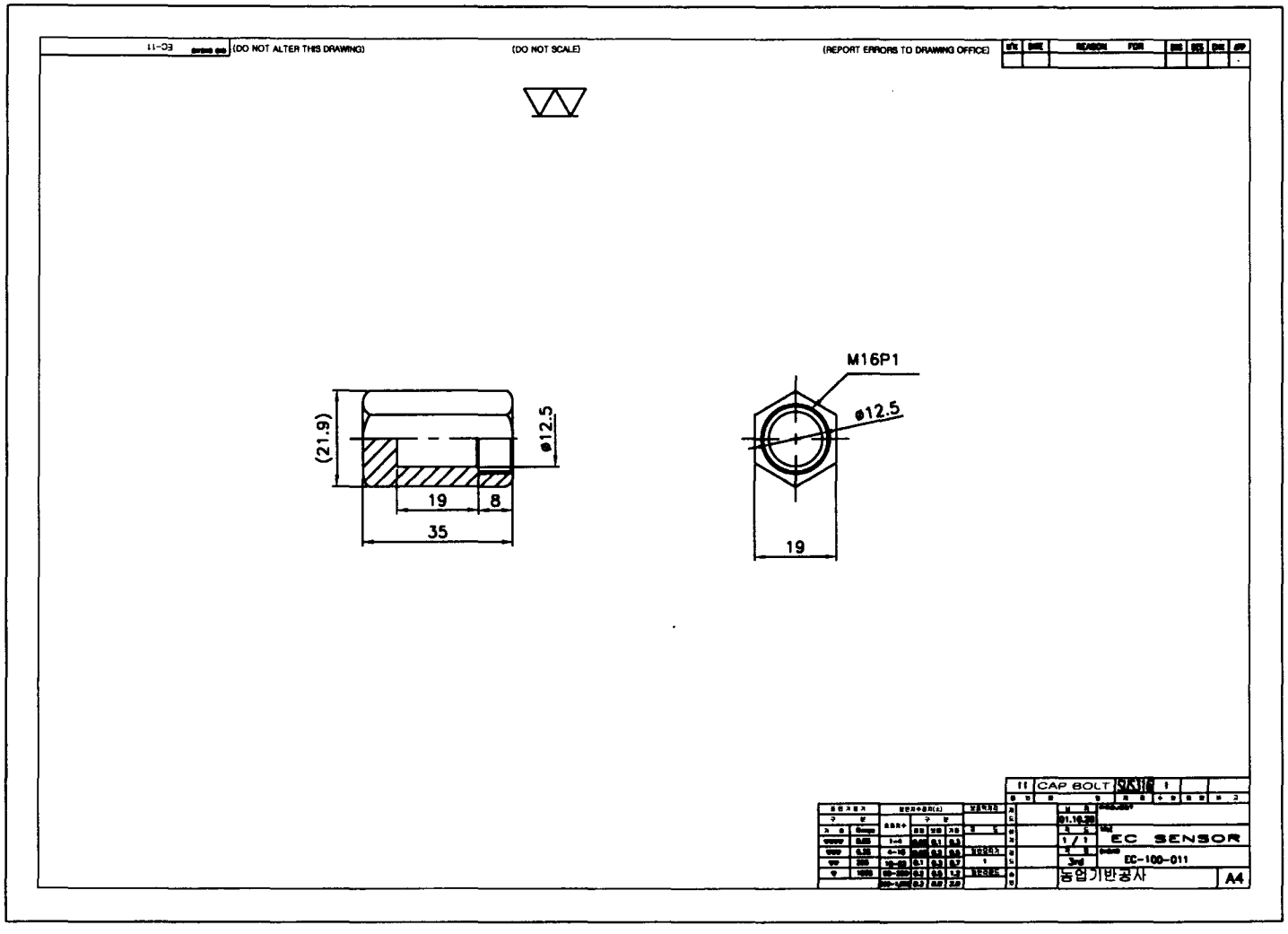


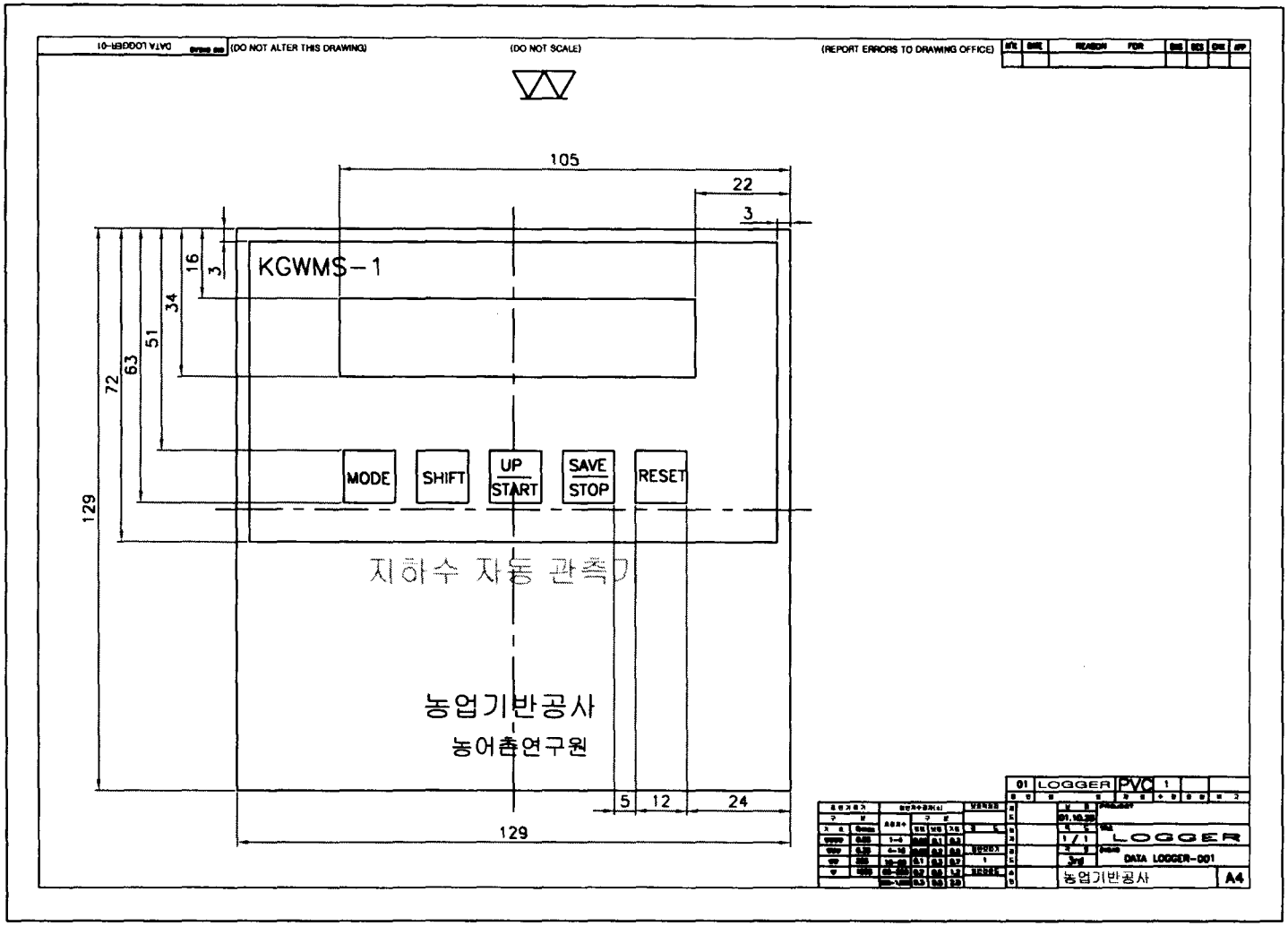
7	GUIDE ROD	PVC	1		
			01.10.25		
			1/1	EC SENSOR	
			3kg	EC-100 - 007	
			농업기반공사		A4

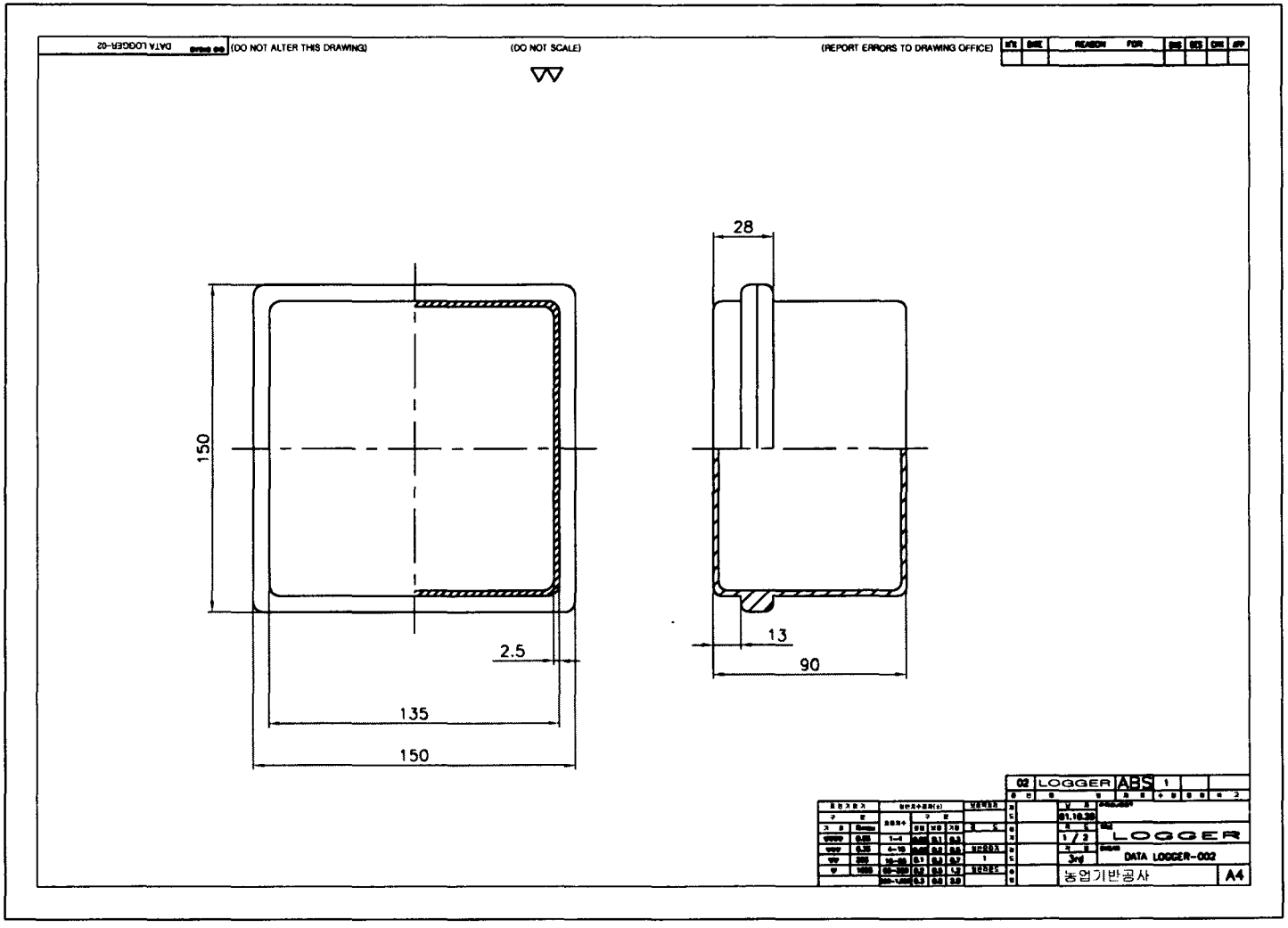

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M22P1
2-M3TAP
5
ø22
ø21.8
ø22
ø25
15
20
8
130
ø25
ø22
SUS PIPE B SUS316
EC SENSOR
EC-100 - 006
농업기반공사
A4

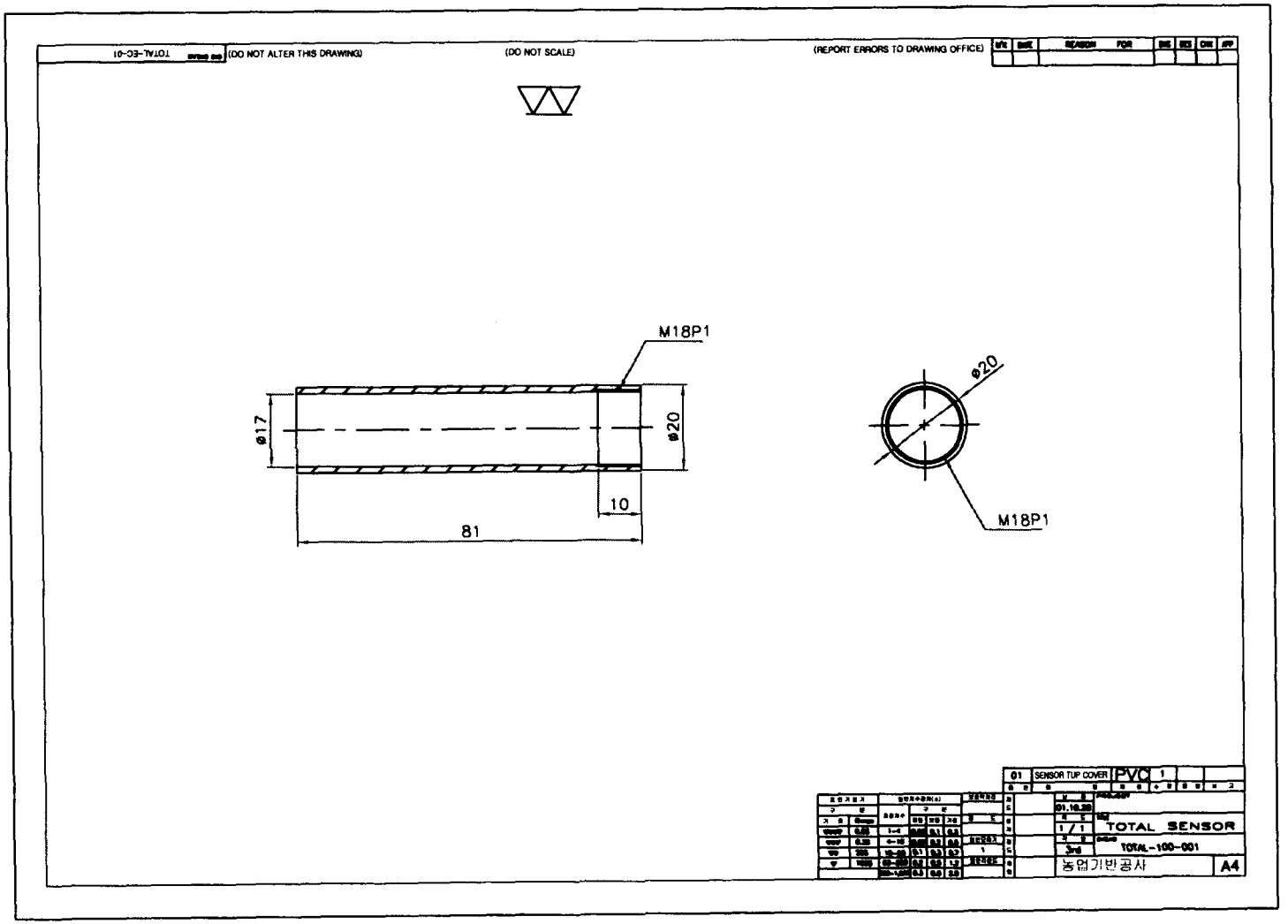

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M16P1
2-ø3
DEEP5
O-RING
(P10)
O-RING
(P19)
M8P1
12
10
ø12.5
ø10.5
ø9
8
10
19
6
ø15
ø19
ø21.5
ø22
ø25
ø7
ø15
ø25
10
1
2
2
2
6
2
2
1.5
1.5
23
5
8
9
45
BUSING
EC SENSOR
EC-100 - 000
농업기반공사
A4

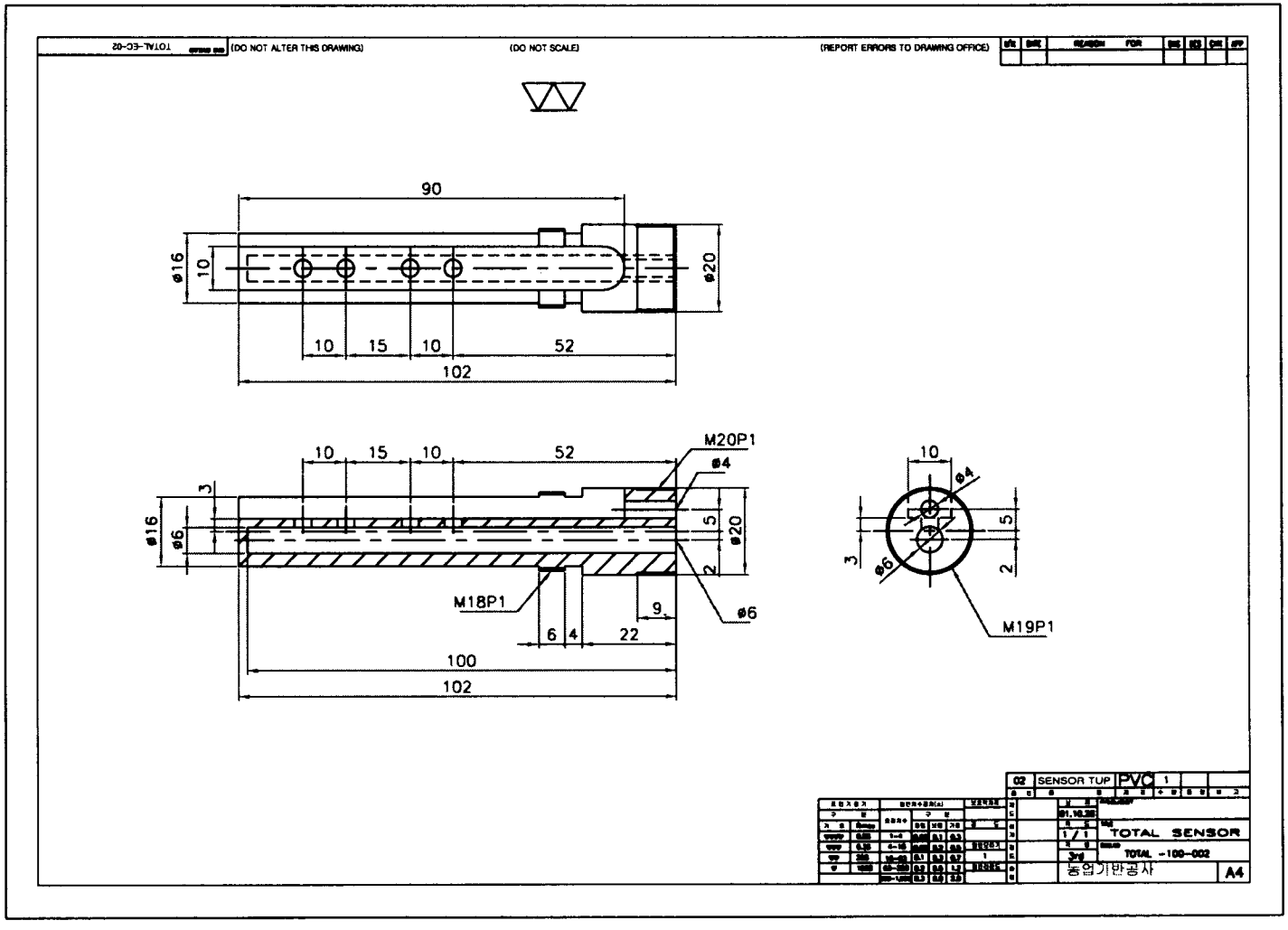

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M16P1
2
1
2
O-RING
(P10)
(21.9)
ø10
ø12
6.5
6.5
20
13
33
ø12
19
10
CAP BOLT
SS316
1
EC SENSOR
EC-100-010
농업기반공사
A4

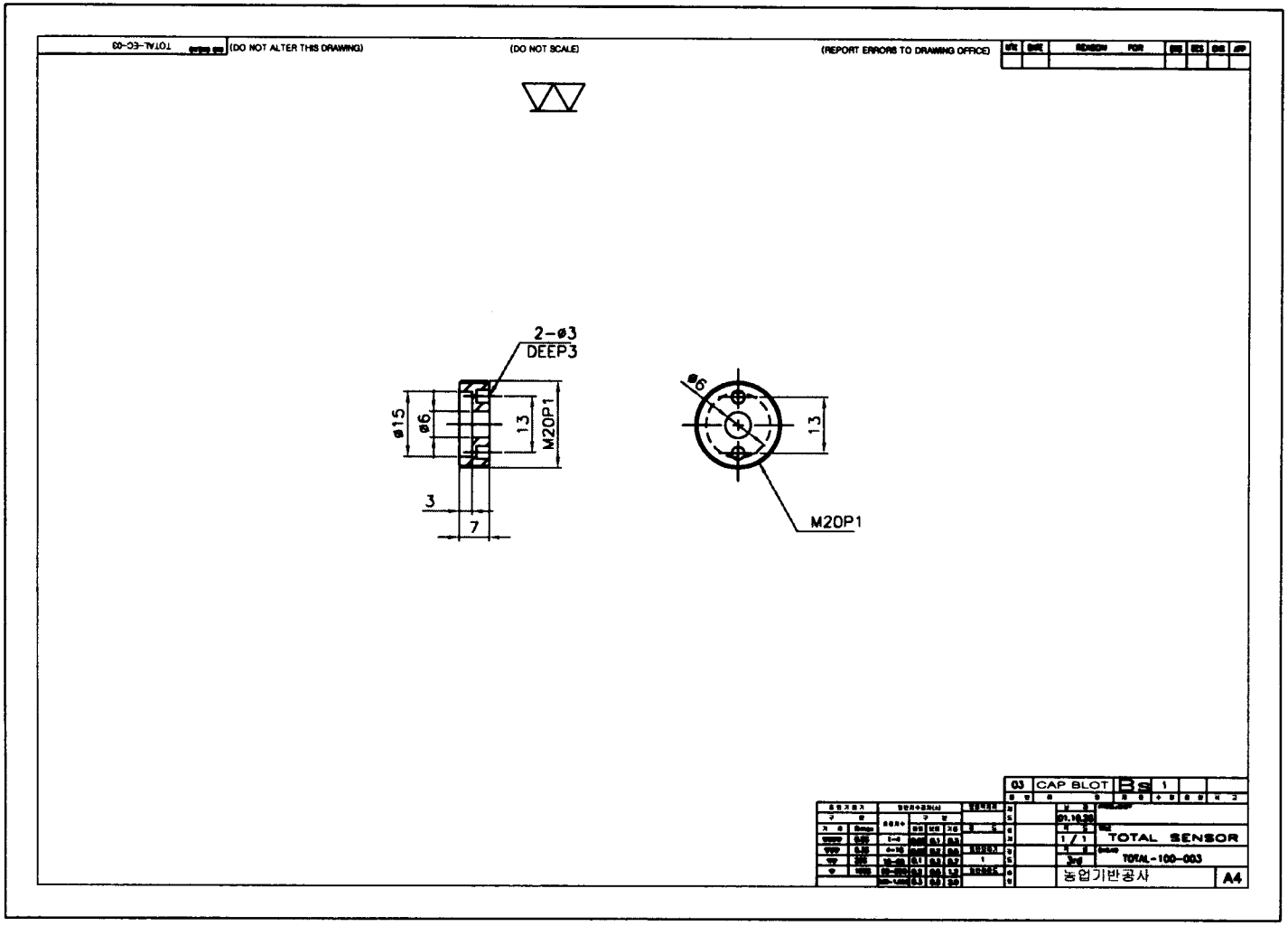

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
(21.9)
ø12.5
19
8
35
M16P1
ø12.5
19
CAP BOLT
EC SENSOR
EC-100-011
농업기반공사
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
105
22
3
16
3
34
51
63
72
129
KGWMS-1
MODE
SHIFT
UP
START
SAVE
STOP
RESET
지하수 자동 관측기
농업기반공사
농어촌연구원
5
12
24
129
01 LOGGER PVC 1
LOGGER
DATA LOGGER-001
농업기반공사
A4

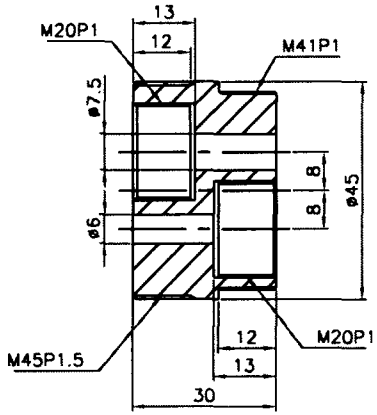

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
28
150
2.5
135
150
13
90
02 LOGGER ABS 1
LOGGER
DATA LOGGER-002
농업기반공사
A4

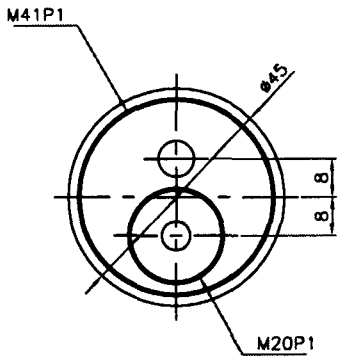

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M18P1
ø17
ø20
10
81
ø20
M18P1
01
SENSOR TIP COVER
PVC
TOTAL SENSOR
TOTAL-100-001
농업기반공사
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
90
ø16
10
ø20
10
15
10
52
102
M20P1
ø4
3
5
2
M18P1
9
ø6
6
4
22
100
102
10
ø4
5
3
ø6
2
M19P1
02
SENSOR TUP
PVC
TOTAL SENSOR
TOTAL -100-002
농업기반공사
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
2-ø3
DEEP3
ø15
ø6
13
M20P1
3
7
ø6
13
M20P1
03
CAP BLOT
TOTAL SENSOR
TOTAL-100-003
농업기반공사
A4

TOTAL-EC-04 (DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)





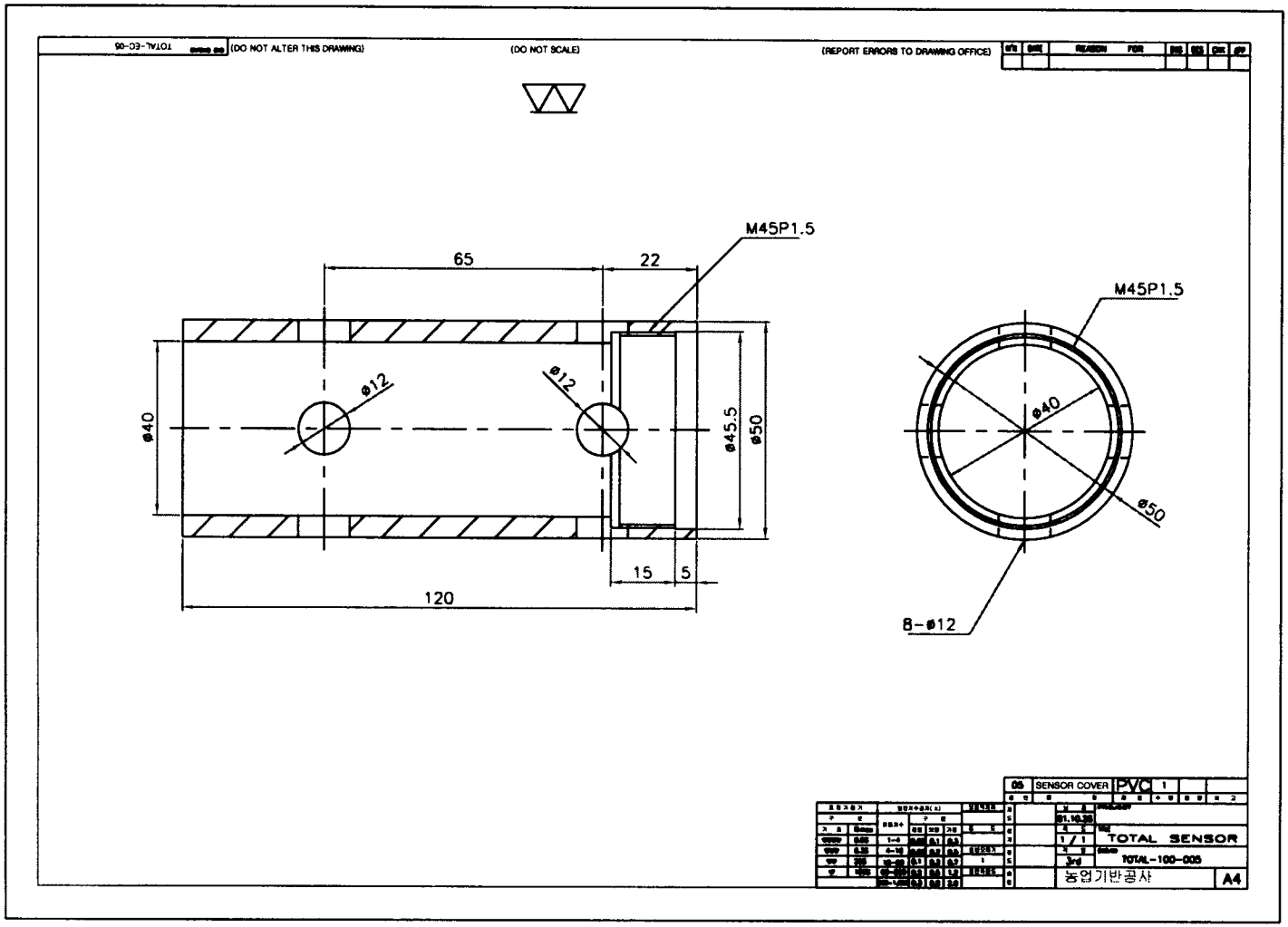
04	CAP BRACKET	PVC	1		

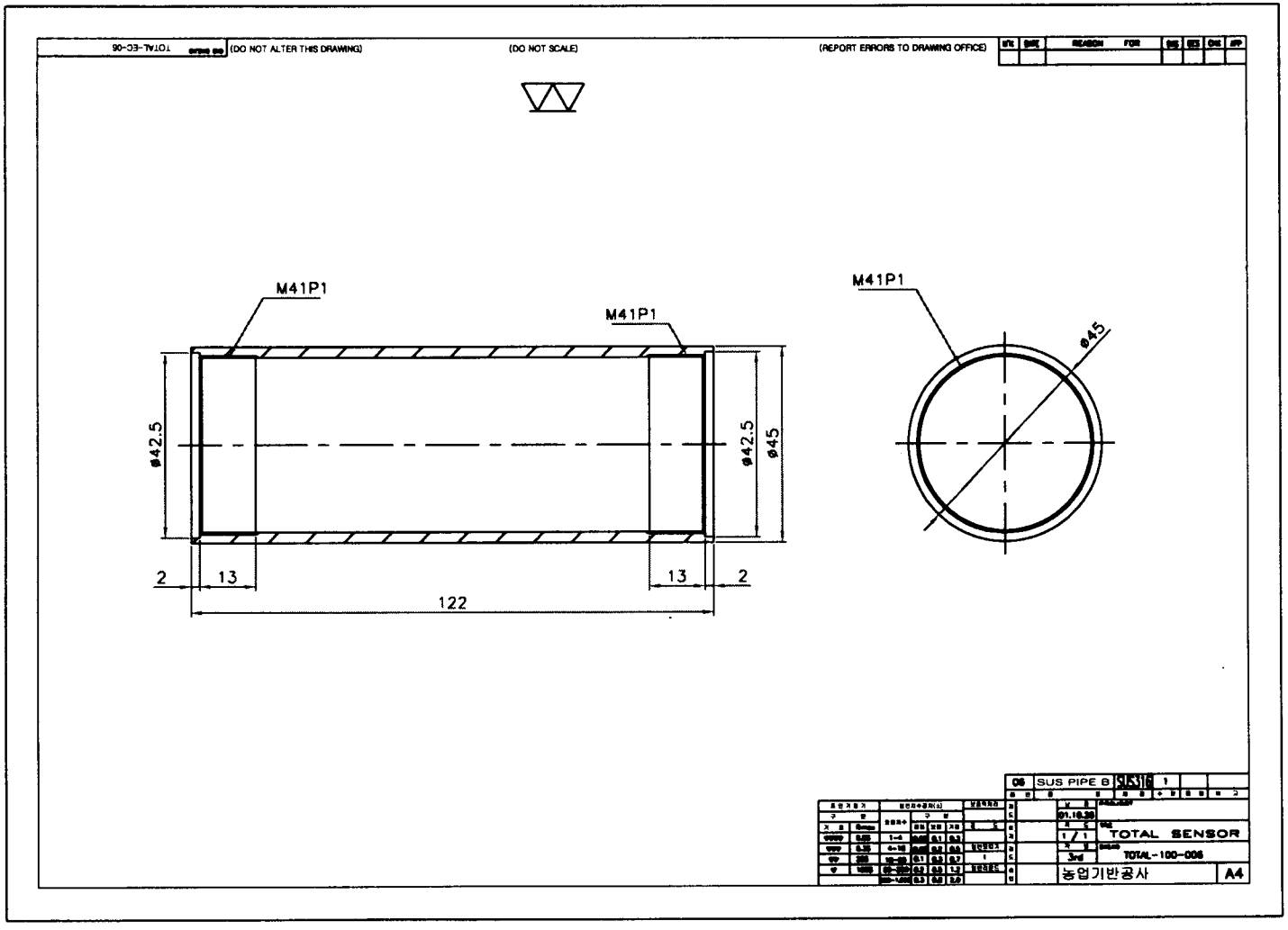
TOTAL SENSOR

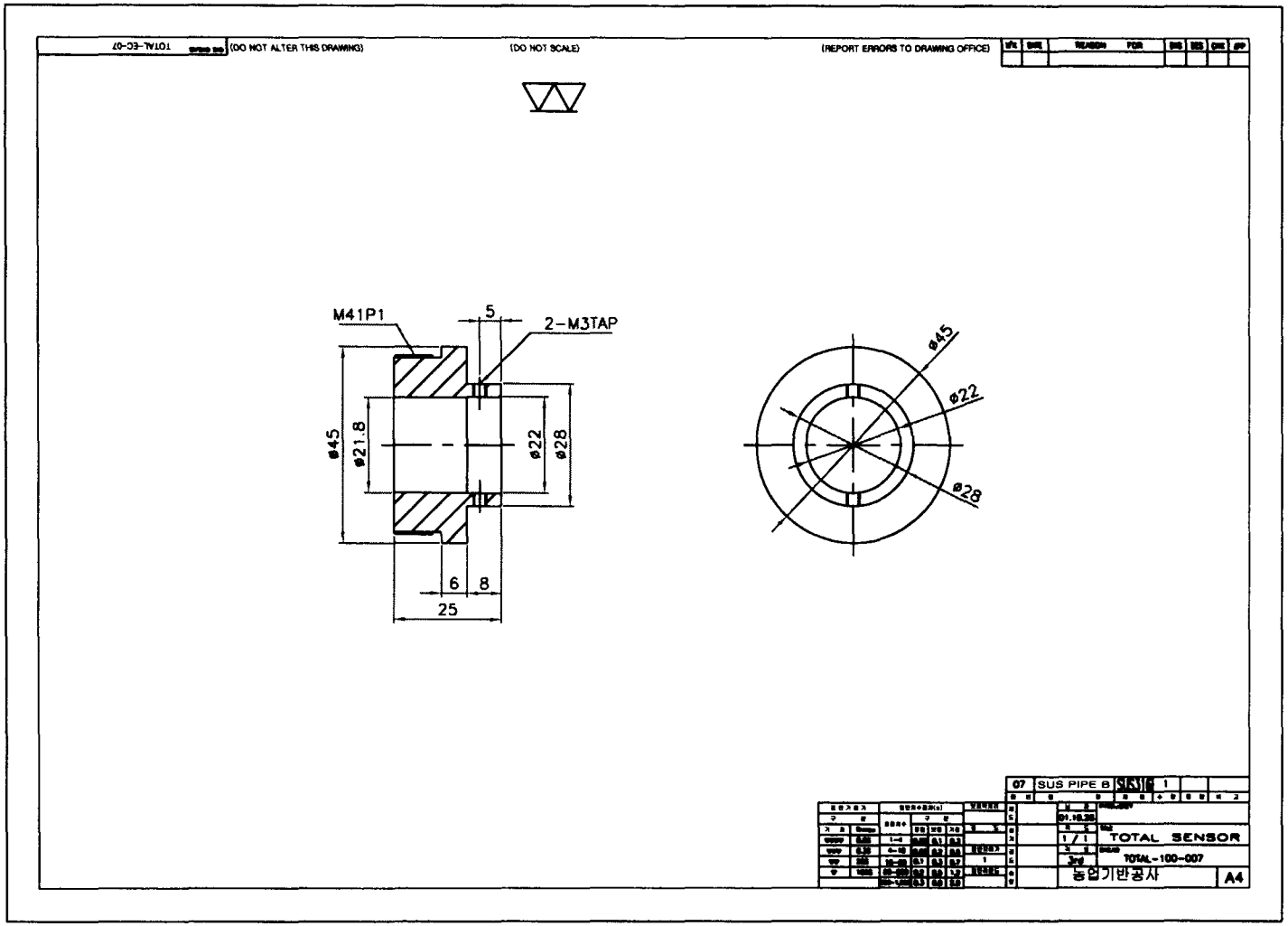
TOTAL-100-004

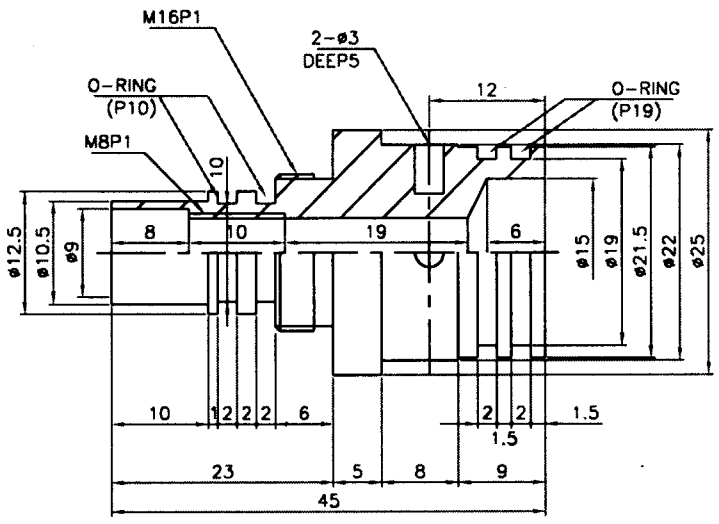
농업기반공사

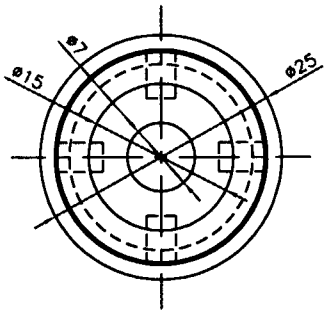
A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M45P1.5
65
22
ø12
ø12
ø40
ø45.5
ø50
15
5
120
M45P1.5
ø40
ø50
8-ø12
05
SENSOR COVER
PVC
1
TOTAL SENSOR
TOTAL-100-005
농업기반공사
A4


TOTAL-EC-06
(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M41P1
M41P1
M41P1
ø42.5
ø42.5
ø45
ø45
2
13
13
2
122
06
SUS PIPE B
SUS316
01.10.30
1/1
TOTAL SENSOR
3rd
TOTAL-100-006
농업기반공사
A4


TOTAL-EC-07
(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
M41P1
5
2-M3TAP
ø45
ø21.8
ø22
ø28
6
8
25
ø45
ø22
ø28
07
SUS PIPE B
1
TOTAL SENSOR
TOTAL-100-007
농업기반공사
A4





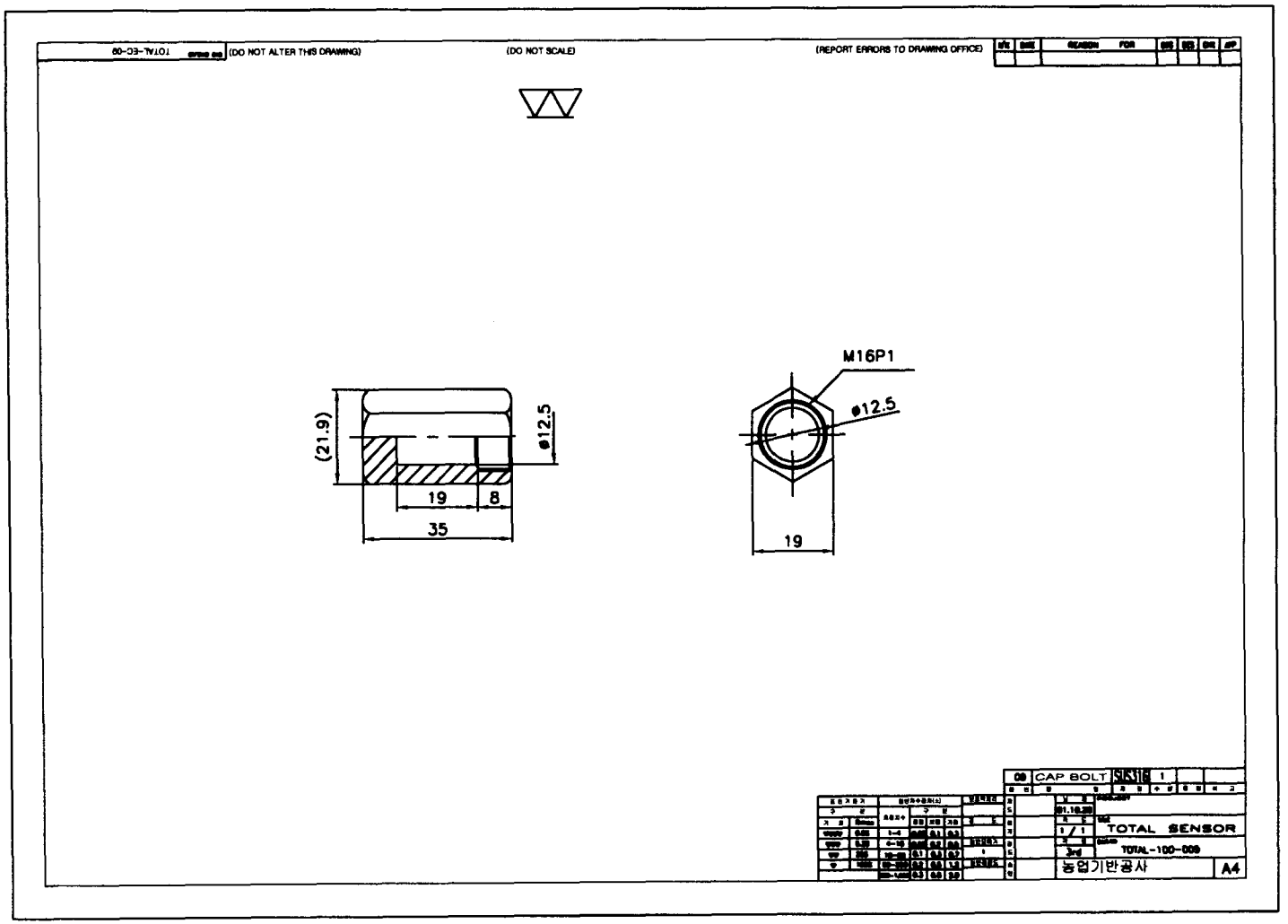
(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)

08 BUSING SUS316 1

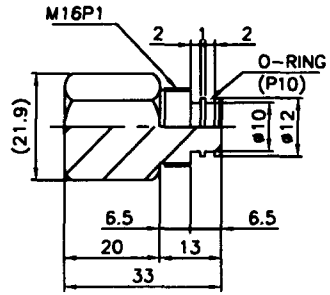
TOTAL SENSOR

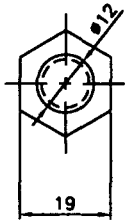
TOTAL-100-008

농업기반공사 A4

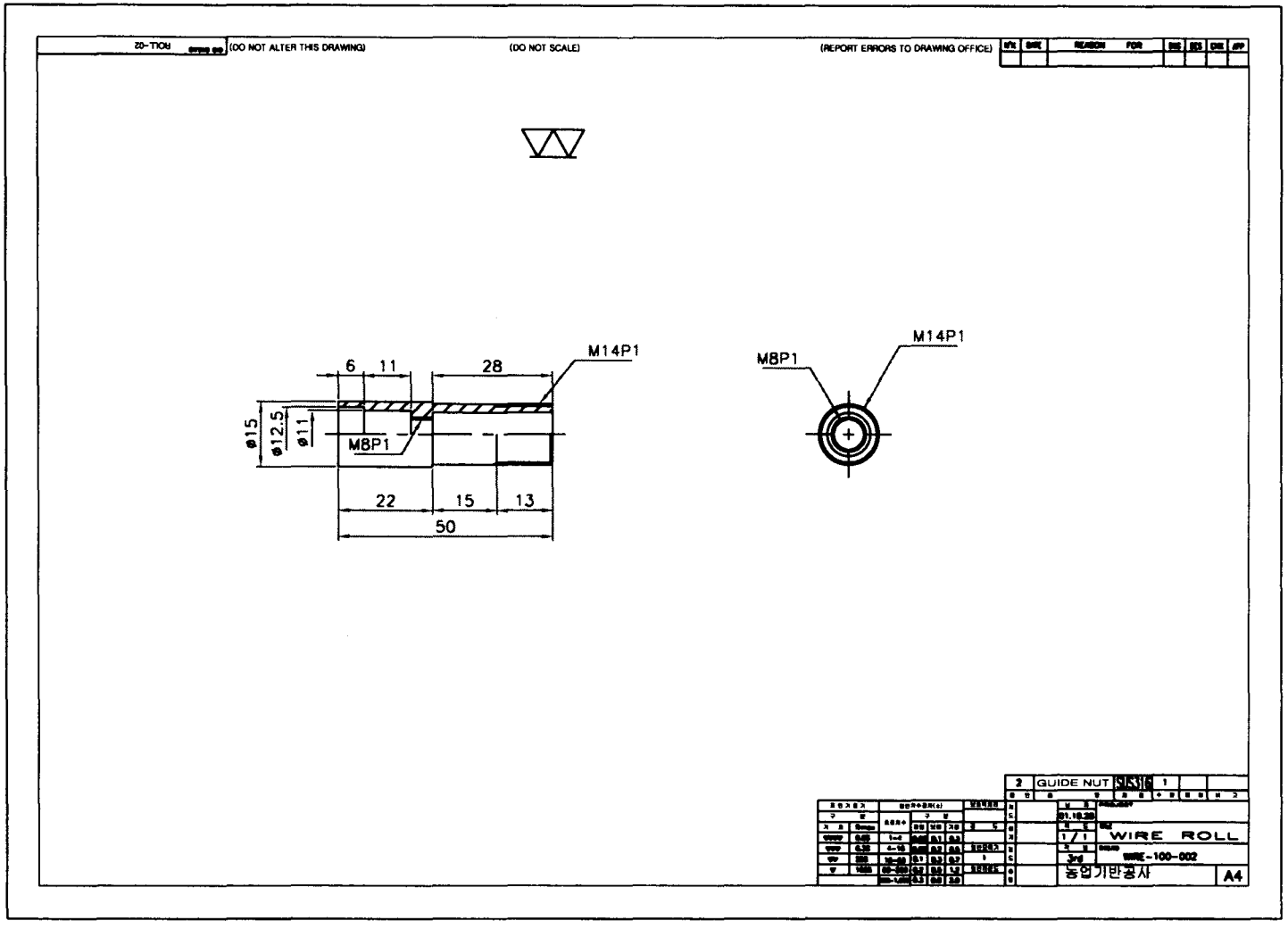

(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
(21.9)
ø12.5
19
8
35
M16P1
ø12.5
19
09
CAP BOLT
SKS1B
1
TOTAL SENSOR
TOTAL-100-009
농업기반공사
A4

(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)

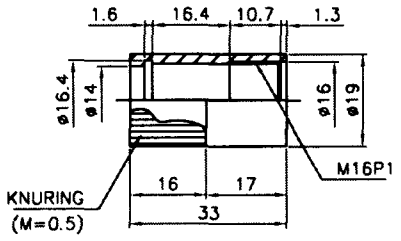


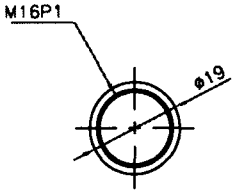


01	CAP BOLT	SUS316	1		
			01.10.30		
			1/1	WIRE ROLL	
			3rd	WRE-100-001	
			중압기반공사		A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
6
11
28
M14P1
ø15
ø12.5
ø11
M8P1
22
15
13
50
M8P1
M14P1
2
GUIDE NUT
SUS316
1
WIRE ROLL
WIRE-100-002
농업기반공사
A4

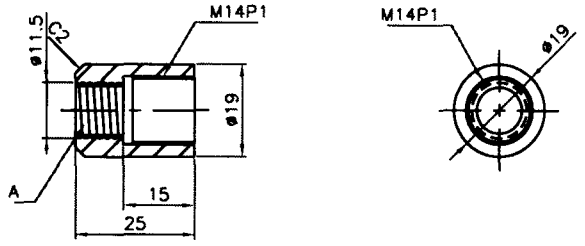
(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)





03	RING NUT	SKS3	1	
	WIRE ROLL		1/1	
	WIRE-100-003		3rd	
	농업기반공사			A4

(DO NOT ALTER THIS DRAWING) (DO NOT SCALE) (REPORT ERRORS TO DRAWING OFFICE)



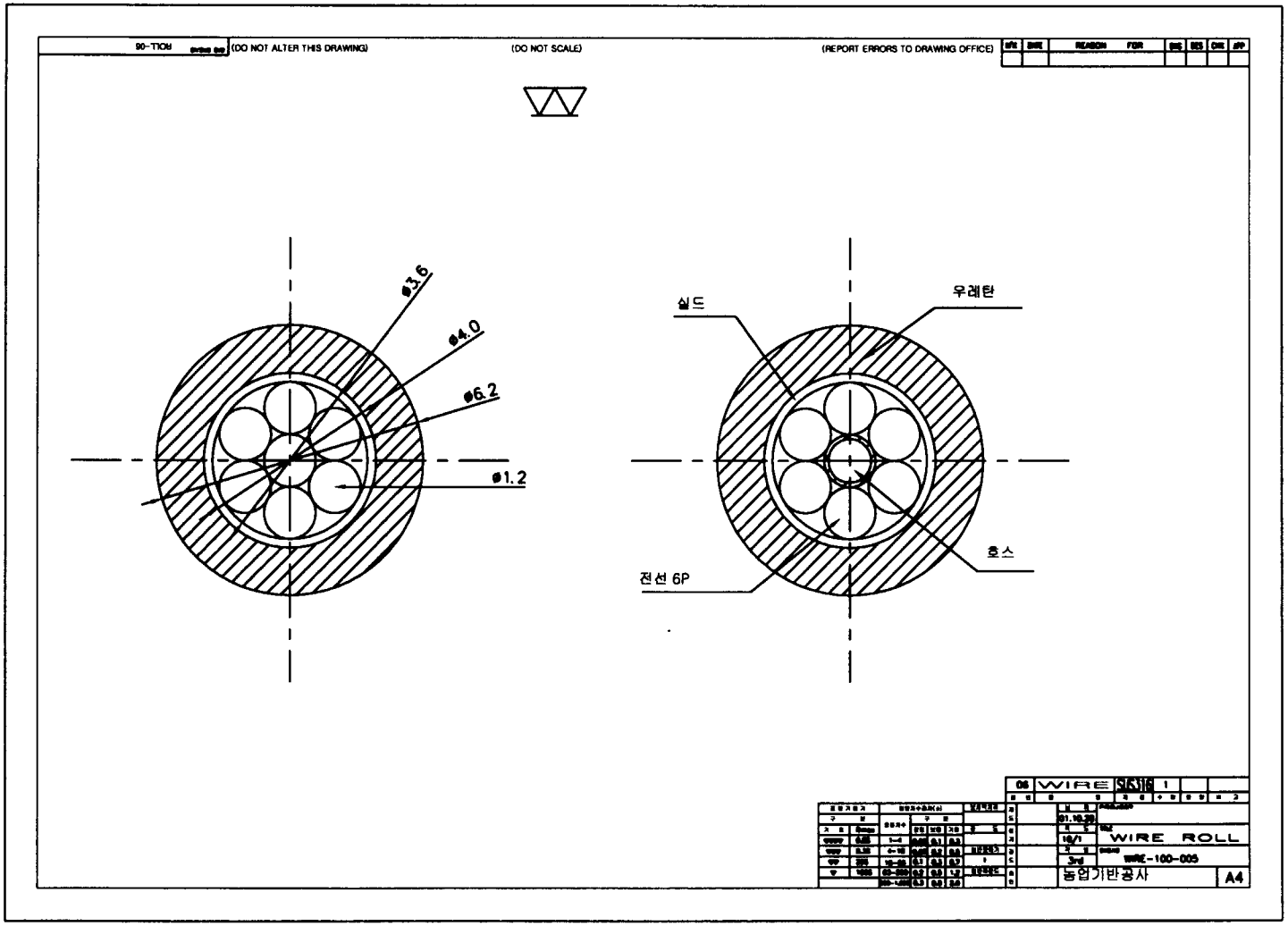
NOTE
1. A 부분은 스프링에 맞추어서 가공할것.

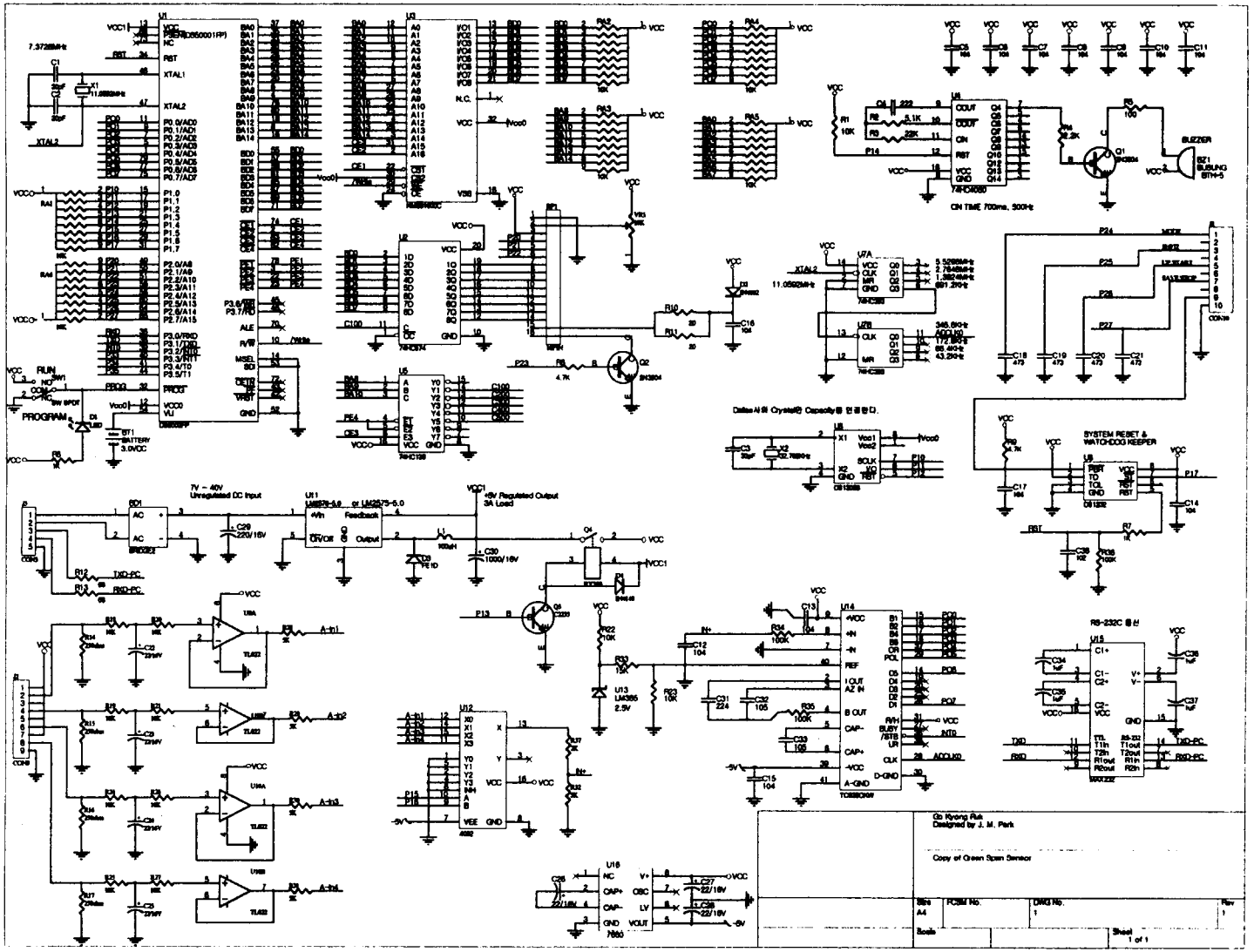
04 SPRING ADAPTER SUS316 1

WIRE ROLL

WIRE-100-004

농업기반공사 A4


(DO NOT ALTER THIS DRAWING)
(DO NOT SCALE)
(REPORT ERRORS TO DRAWING OFFICE)
REASON FOR
ø3.6
ø4.0
ø6.2
ø1.2
실드
우레탄
호스
전선 6P
WIRE
WIRE ROLL
농업기반공사
A4


Qb Kyong Rok
Designed by J. M. Park
Copy of Green Span Sensor
Size A4
PCBM No.
DWG No. 1
Rev 1
Scale
Sheet 1 of 1
PROGRAM
RUN
SYSTEM RESET & WATCHDOG KEEPER
BUZZER
ON TIME 700ms, 500Hz
7V ~ 40V Unregulated DC Input
+5V Regulated Output 3A Load
RS-232C 통신
Dallas사의 Crystal의 Capacity를 만족할것.

부 록 5

```
*/

#include <absacc.h>
#include <stdio.h>
#include <math.h>
#include <stdio.h>     // I/O에 관련된
                          Prototype 선언
#include <string.h>    // String, Buffer에
                          관련된 header file
#include <stdlib.h>    // 형 변환, 메모리 배치에
                          관련된 Function 정의
#include <ctype.h>   // 부록 참조
#include <intrins.h>  // _nop_(); 사용
#include "DS5002FP.H"
#include "PES_105.H"
#include "PES_VAR2.H"
#include "PES_STR5.H"

/*************************
     함수 선언
*************************/
void initial_int(void);  // MPU를 위한 초기화
void delay(int i);     // Time Delay Rountine ,
                    Software적 Time Delay임
void Wdt_out();       // WatchDog 신호
void init_port(void);   // MCS51의 Port 초기화
void Buz_sound();       // Buzzer Sound Output
void E_Pulse();         // LCD Enable 신호
void  Func_set();       // LCD 컨트롤러 HD44780의
                  기능을 설정하는 함수
void Init_Lcd();  // LCD를 초기화 함수
void write_char(unsigned char s);
                // LCD에 문자를 한개 써넣기
void    lcd_printf(unsigned    char    lcd_addr,
unsigned char *c);
  // lcd의 주소를 찾아 문자열을 wirte하는 루틴
void  Lcd_OneCharWrite(unsigned  char  lcd_addr,
unsigned char s);
      //LCD에 한주소 문자를 한개 써넣기
void Init_DS1305();
      // DS1305 처음으로 동작시킬때 사용한다.
void Write_DS1305(unsigned char addr ,unsigned
char value);
          //DS1305로 데이터를 쓴다.
unsigned char Read_DS1305(unsigned char addr);
   // DS1305에 데이타를 읽는다.
void set_time();
          //Time 변수들에 초기값을 입력한다.
void  At_PowerON();     //  본장비의  전원을  On
          시키면 처음으로 실행하는 루틴
char  CHKPLS(struct  KEY_STAT  *cp,char  ky);
          // Key 눌림상태 정의
char getkey();     // Key switch sensing
void tx232_1();    // System의 Data 전부 송신
void tx232_2();    // 현재 Logger의 Data 송신
void tx232_3();    // System의 Data 7회분 송신
void time_wr();    // time 설정
void time_read();
    //  DS1305에서 Time Data 읽기
void DS1305_SetValue_Extract();
    // DS1305에 저장된 각종 Setting값 추출
int  getadc();
    // Routine for data fetched of A/D(TC835)
void SMD_init();
    // PC에서 들어온 명령어 초기화
void pc_cmd_div();
    // PC에서 들어온 명령어 분리 작업
void DS1305_SettingUp();
    // PC에서 System의 시간 설정
void Ram_Clear_All();
    // 저장된 5000회 분의 Data를 Clear 시킨다.
void calibration_mode();
    // 교정하여 교정값을 SAVE 시키는 Mode
int  Inteval_Set_time();
    // RS232 통신을 통한 저장 시간 간격 조정
void Massange_Disp(char valuel, long value2);
  // 측정하고 있는 값을 Display한다.
void MassDis012(char value);
    // 고정된 Display Title
void MassTime(int value);
    // 현재 날자 및 시간 Display
void  Current_Data(unsigned  char  mode21,  long
value);  // RS232를 통하여 보내질 현재 Data
void AD_theorem();  // A/D Converter 값 정리
long Get_ADValue();

char BZ_sound, digit;
char Exchange_time, Wait_set_time, Ram_CLS_time;

char Lcd_backlight_time, Lcd_Off_status;
unsigned char Receive_inform;
int  depth_sph,  depth_spl,  ec_spl,  ec_sp2,
ec_sp3, ec_sp4, temp_sph, temp_spl;
int  spv_depth,  spv_ec1,  spv_ec2,  spv_ec3,
spv_temp, n, holssu;
int  DPV_Depth,  DPV_Ec1,  DPV_Ec2,  DPV_Ec3,
```

```
DPV_Temp;
unsigned int t500ms, t3sec, t4sec, t5sec, t1min;
unsigned int g = 0;
char selv_mode = 0, Set_mode = 0;
int total_min = 0, Conversion_min = 0, Ba = 0;
long Na[20];
int EC_PD1, EC_PD2, EC_PD3, EC_PD4;

int Depth_DPH = 1000, EC_PDH =1000;
int Depth_DPL = 0, EC_PDL = 0;
int Temp_PDH  = 500;
int Temp_PDL  = -200;

unsigned char RS232_rx_Buf;
   /* PC에서 설정을 요구할 때 명령을 위한
Buffer */
unsigned char Set_Cmd[35];
   /* PC에서 보내는 Data를 넣어 놓는 곳 */
unsigned char SIO_CMD[30];

   /*
        공용체 및 구조체 선언
                                    */
   /* Key의 상태 보존을위한 Memory 확보 */
struct KEY_STAT          mode_key;
struct KEY_STAT          shift_key;
struct KEY_STAT          up_key;
struct KEY_STAT          enter_key;

union  KEY_BUFF          keyb;
union  KEY_MAPS          key;
// union  TIME_DATA          time_data[10];
union  RS232_STATUS      RS232;
union  Reg_Def_SEC       Second[3];
union  Reg_Def_MIN       Minute[3];
union  Reg_Def_HR        Hour[3];
union  Reg_Def_DATE      Date[2];
union  Reg_Def_MONTH     Month[2];
union  Reg_Def_YEAR      Year[2];
union  Reg_DS1305_CTRL   Ctrl_RTC;
union  ICM_DATA          icm[10];
union  ASC_HEX1         Save_Value7[19];
union  Save_PES_input    PES_Div_Data;
struct Save_PES_Output  PES_Out_Data[5000];
union  ADC_BUFF adc;
union  ADC_BUFF TC835[10];


// 현재 Logger의 Data
code char NowDT007[35] =
        {"DCTP+100.00+10000+025.6+123450000"};
//        01234567890123456789012345678901234
unsigned char Now_Data[35];
// main function
void main(void) {

  // Variables First stage
  int AD_ORG_value = 0;
  int Con_save_T = 0, mode_time3 = 0;
  char tx_start1 = 0, Ma = 0;
  char sio_snum = 0;
  long AD_value1 = 0, Last_Value2 = 0;
  long SP_Value = 0;
  char imsi_cal_set = 0;
  char imsi_clear1 = 0, clear2 = 0,
       ram_CLS = 0;
  char cal_set22 = 0;
  unsigned char Buz_var = 0;
  char spectrum = 0;
  char hujun = 0, Leesuyeong = 0;
  char channel = 0, first1 = 0;
  char Disp317 = 0, Disp328 = 0;
  char imsi77, Power33 = 0, Power35 = 0;

  At_PowerON();
  imsi77 = Read_DS1305(R_STA_REG);
           // Status register reading
  key.x.word = getkey();
  Buz_sound();
  selv_mode = 0;
  Ba = holssu;

  // Do forever , Program main rountine
  while(1) {

    key.x.word = getkey();
                  // Key switch sensing
    if(key.h.mode_key == M2) {
         selv_mode++;
         Disp328 = 0;
         first1 = 0;
         Leesuyeong = 0;
         if(selv_mode > 4) selv_mode = 0;
    }

    // Data Gathering Equipment of Water
       Pollution & Analysis
    if(selv_mode == 0) {
         Leesuyeong = 0;
         Disp328 = 0;
```

```
      time_read();
      // Power를 On(Shift Key) 시켰을
        때의 동작으로 Data 저장 시간에
    의해서 이루어 진다.
    if(Power33 == 1) {
      Con_save_T = (int)
        (Conversion_min % total_min);
      if(Con_save_T == 0) Ma = 1;
      else {
          Ma = 0;
          hujun = 0;
      }
      if((Ma == 1) && (hujun == 0)) {
          hujun = 1;
          channel = 0;
          spectrum = 0;
          Analog_SEL1 = LO;
          Analog_SEL2 = LO;
          delay(10);
          Exchange_time = 0;
          t3sec = T3SEC;
      }
}

// RTC(시계 IC)에 의해 동작 할때
if((Power33 == 0) && (Power35 == 0)) {
      Ma = 1;
      spectrum = 0;
      Exchange_time = 0;
      t3sec = T3SEC;
      Power35 = 1;
}

if((Ma == 1) && (spectrum == 0)) {

    AD_valuel = Get_ADValue();

//                  01234567890123456789
lcd_printf(0x80, "Data Gathering Equip");

if(Disp317 == 0) {
      if(channel == 0) {
          lcd_printf(0xC0,"              ");
          MassDis012(1);
      }
      if(channel == 1) {
          lcd_printf(0xC0,"              ");
          MassDis012(2);
      }
      if(channel == 2) {
          lcd_printf(0xC0,"              ");
          MassDis012(3);
      }
      Disp317 = 1;
}

if(channel == 0) {
      SP_Value = ((((AD_valuel -
                  (long)depth_spl) *
                  (long)DPV_Depth) /
                  (long)spv_depth) +
                  (long)Depth_DPL);
      Last_Value2 = SP_Value;
      Massange_Disp(1, Last_Value2);
      // 측정하고 있는 값을 Display한다.
      // System의 Analog Channel를 On
         시킨지 3초 후의 동작
      if(Exchange_time == 1) {

      PES_Div_Data.h.DepthOfWater =
                        Last_Value2;
      Current_Data(1, Last_Value2);
      channel = 1;
      Analog_SEL1 = HI;
      Analog_SEL2 = LO;
      Exchange_time = 0;
      t3sec = T3SEC;
      Disp317 = 0;
      delay(10);
   }
}

if(channel == 1) {
      if(AD_valuel < (long)ec_sp2) {
        SP_Value = ((((AD_valuel -
        (long)ec_spl) * (long)DPV_Ec1) /
        (long)spv_ec1) + (long)EC_PD1);
      }
      if(AD_valuel < (long)ec_sp3) {
        SP_Value = ((((AD_valuel -
        (long)ec_sp2) * (long)DPV_Ec2) /
        (long)spv_ec2) + (long)EC_PD2);
      }
      if(AD_valuel >= (long)ec_sp3) {
        SP_Value = ((((AD_valuel -
        (long)ec_sp3) * (long)DPV_Ec3) /
        (long)spv_ec3) + (long)EC_PD3);
      }
      // SP_Value = ((((AD_valuel -
        (long)ec_spl) * (long)DPV_Ec) /
```

```
        (long)spv_ec) + (long)EC_PDL);
      Last_Value2 = SP_Value;
      Massange_Disp(2, Last_Value2);
        // 측정하고 있는 값을 Display한다.
      // System의 Analog Channel를
        변경한지 3초 후의 동작
      if(Exchange_time == 1) {
          PES_Div_Data.h.Conductivity
                  = Last_Value2;
          Current_Data(2, Last_Value2);
          channel = 2;
          Analog_SEL1 = LO;
          Analog_SEL2 = HI;
          Exchange_time = 0;
          t3sec = T3SEC;
          Disp317 = 0;
          delay(10);
      }
}

if(channel == 2) {
    SP_Value = ((((AD_valuel -
                (long)temp_spl) *
                (long)DPV_Temp) /
                (long)spv_temp) +
                (long)Temp_PDL);
    Last_Value2 = SP_Value;
    Massange_Disp(3, Last_Value2);
       // 측정하고 있는 값을 Display한다.
    // System의 Analog Channel를
       변경한지 3초 후의 동작
    if(Exchange_time == 1) {
        PES_Div_Data.h.Temperature
            = Last_Value2;
        Current_Data(3, Last_Value2);
        channel = 0;
        Analog_SEL1 = LO;
        Analog_SEL2 = LO;
        time_read();
        delay(10);
    // Data Conversion하고 Save
    PES_Out_Data[Ba].Bytes1  =
                PES_Div_Data.x.Bytes31;
    PES_Out_Data[Ba].Bytes2  =
                PES_Div_Data.x.Bytes32;
    PES_Out_Data[Ba].Bytes3  =
                PES_Div_Data.x.Bytes33;
    PES_Out_Data[Ba].Bytes4  =
                PES_Div_Data.x.Bytes34;
    PES_Out_Data[Ba].Bytes5  =
                PES_Div_Data.x.Bytes35;
    PES_Out_Data[Ba].Bytes6  =
                PES_Div_Data.x.Bytes36;
    PES_Out_Data[Ba].Bytes7  =
                PES_Div_Data.x.Bytes37;
    PES_Out_Data[Ba].Bytes8  =
                PES_Div_Data.x.Bytes38;
    PES_Out_Data[Ba].Bytes9  =
                PES_Div_Data.x.Bytes39;
    PES_Out_Data[Ba].Bytes10 =
                PES_Div_Data.x.Bytes40;
    PES_Out_Data[Ba].Bytes11 =
                PES_Div_Data.x.Bytes41;
    Ba = Ba + 1;
    if(Ba > 5000) Ba = 0;
    holssu = Ba;
    spectrum = 1;
    Disp317 = 0;
    //  시계 IC에 의한 동작시 Power Off
    if(Power33 == 0) {
      EX0 = 0;
        // IE.0  INT0 Interrupt Enable
      ET0 = 0;
        // IE.1  TIMER0 Interrupt Enable
      ES  = 0;
        // IE.4  시리얼 포트 인터럽트
              Enable
      TR0 = 0;
        // TCON.4  Timer 0 Run
      TR1 = 0;
        // TCON.6  Timer 1 Run
      EA  = 0;  // IE.7  Interrupt enable
      // P0 = 0x00;
      // P1 = 0xFF;
      // P2 = 0xFF;
      System_Power = HI;
      PCON |= 0x03;
            // Clear IDL bit in power
               control register
      // PCON |= 0x02;
            // Clear STOP bit in power
                control register
     }
    }
   }
   Wdt_out(); // WatchDog 신호
} // if((Ma == 1) && (spectrum == 0))
 의 Ends
else {
     MassTime(holssu);
```

```
        delay(500);
    }
} // if(selv_mode == 0)의 Ends
// 각 Channel 별 수동으로 Data 확인
if(selv_mode != 0) {
    time_read();
    if(Lcd_Off_status == 1) {
        if(Leesuyeong == 0) {
            mode_time3 =
                Conversion_min;
            Leesuyeong = 1;
        }
        if(abs(Conversion_min -
            mode_time3) > 3) {
            selv_mode = 0;
            Disp328 = 0;
        }
    }

    if(Disp328 == 0) {
        if(selv_mode == 1) MassDis012(1);
        if(selv_mode == 2) MassDis012(2);
        if(selv_mode == 3) MassDis012(3);
        if(selv_mode == 4) MassDis012(4);
        Disp328 = 1;
    }

    AD_valuel = Get_ADValue();

    //              01234567890123456789
    lcd_printf(0x80, "Data Gathering Equip");

    // Depth Value Display
    if(selv_mode == 1) {
        Analog_SEL1 = LO;
        Analog_SEL2 = LO;
        SP_Value = ((((AD_valuel -
                    (long)depth_spl) *
                    (long)DPV_Depth) /
                    (long)spv_depth) +
                    (long)Depth_DPL);
        Last_Value2 = SP_Value;
        Massange_Disp(1, Last_Value2);
        Current_Data(1, Last_Value2);
    }

    // EC Value Display
    if(selv_mode == 2) {
        Analog_SEL1 = HI;
        Analog_SEL2 = LO;
        if(AD_valuel < (long)ec_sp2) {
            SP_Value = ((((AD_valuel -
                        (long)ec_spl) *
                        (long)DPV_Ec1) /
                        (long)spv_ec1) +
                        (long)EC_PD1);
        }
        if(AD_valuel < (long)ec_sp3) {
            SP_Value = ((((AD_valuel -
                        (long)ec_sp2) *
                        (long)DPV_Ec2) /
                        (long)spv_ec2) +
                        (long)EC_PD2);
        }
        if(AD_valuel >= (long)ec_sp3) {
            SP_Value = ((((AD_valuel -
                        (long)ec_sp3) *
                        (long)DPV_Ec3) /
                        (long)spv_ec3) +
                        (long)EC_PD3);
        }
       // SP_Value = ((((AD_valuel -
                    (long)ec_spl) *
                    (long)DPV_Ec) /
                    (long)spv_ec) +
                    (long)EC_PDL);
        Last_Value2 = SP_Value;
        Massange_Disp(2, Last_Value2);
        Current_Data(2, Last_Value2);
    }

    // Temperature Value Display
    if(selv_mode == 3) {
        Analog_SEL1 = LO;
        Analog_SEL2 = HI;
        SP_Value = ((((AD_valuel -
                    (long)temp_spl) *
                    (long)DPV_Temp) /
                    (long)spv_temp) +
                    (long)Temp_PDL);
        Last_Value2 = SP_Value;
        Massange_Disp(3, Last_Value2);
        Current_Data(3, Last_Value2);
```

```
    }


    // pH Value Display Mode 이지만 현재
  는 A/D Value를 Display한다.
    if(selv_mode == 4) {
        Analog_SEL1 = HI;
        Analog_SEL2 = HI;
        Massange_Disp(4, AD_value1);
        Current_Data(4, AD_value1);
    }
}

// Origin A/D Value
while(key.h.enter_key == MI) {
    key.x.word = getkey();
            // Key switch sensing
    AD_ORG_value = getadc();
    Massange_Disp(5, (long)AD_ORG_value);
    Disp328 = 0;
    Wdt_out(); // WatchDog 신호
    delay(200);
}

// pc에서 명령이 들어오면 Data를 처리하는
  루틴(RS232 Port의 처리루틴)
if(RS232.h.Rx_sta == HI) {
    tx_start1 = 1;
    sio_snum = Receive_inform;
}

if(tx_start1 == 1) {
    EX0 = 0;     // 인터럽트 금지
    // System의 Data 송신
    if(sio_snum == 1) {
        //          01234567890123456789
        lcd_printf
          (0x80, "Data transmit by   ");        ");
        lcd_printf
          (0xC0, "Data Request of PC ");
        tx232_1();
        tx_start1 = 0;
        SMD_init();
        RS232.h.Rx_sta = LO;
        Receive_inform = 0x00;
    }

    // 현재 Logger의 Data 송신
    if(sio_snum == 2) {
        tx232_2();
        tx_start1 = 0;
        SMD_init();
        RS232.h.Rx_sta = LO;
        Receive_inform = 0x00;
    }

    // PC에서 System의 시간 설정
    if(sio_snum == 3) {
        DS1305_SettingUp();
        SMD_init();
        tx_start1 = 0;
        RS232.h.Rx_sta = LO;
        Receive_inform = 0x00;
    }

    // System의 Data를 저장하는
      시간 간격을 설정
    if(sio_snum == 4) {
        //
        total_min = Inteval_Set_time();
        //          01234567890123456789
        lcd_printf(0x80,
                "Inteval of Save Data");
        SMD_init();
        tx_start1 = 0;
        RS232.h.Rx_sta = LO;
        Receive_inform = 0x00;
        delay(20000);
        delay(20000);
    }

    // 저장된 5000회 분의 Data를 Clear
      시킨다.
    if(sio_snum == 5) {
        lcd_printf(0x80,
                "Old Data Clear      ");
        lcd_printf(0xC0,          "Initialling

        Ram_Clear_All();
        SMD_init();
        tx_start1 = 0;
        RS232.h.Rx_sta = LO;
        Receive_inform = 0x00;
        Ba = 0;
        holssu = 0;
    }
```

```
        // System의 Data 5회분 송신
        if(sio_snum == 6) {
            //          01234567890123456789
            lcd_printf(0x80,
                    "Data transmit by    ");
            lcd_printf(0xC0,
                    "Request of PC 5times");
            tx232_3();
            tx_start1 = 0;
            SMD_init();
            RS232.h.Rx_sta = LO;
            Receive_inform = 0x00;
        }

        EX0 = 1;  // 인터럽트 가능
    } // if(tx_start1 == 1)문의 Ends

    // RAM CLEAR
    if((key.h.shift_key == M1) &&
                (imsi_clear1 == 0)) {
            imsi_clear1 = 1;
            Ram_CLS_time = 0;
            t5sec = T5SEC;
        }
        if((key.h.shift_key == M1) &&
                (imsi_clear1 == 1)) clear2 = 1;
        else {
            clear2 = 0;
            imsi_clear1 = 0;
        }
        if((clear2 == 1) && (Ram_CLS_time ==
                        1)) ram_CLS = 1;
        while(ram_CLS) {
            lcd_printf(0x80, "Old Data Clear  ");
            lcd_printf(0xC0,          "Initialling
");
            Ram_Clear_All();
            ram_CLS = 0;
            Ba = 0;
            holssu = 0;
        }

        /* Mode Key를 4초 동안 누르고 있으면 Auto
        Calibration 변수를 입력하는 Mode로 들어간
        다. */
        if((key.h.mode_key == M1) &&
                (imsi_cal_set == 0)) {
            imsi_cal_set = 1;
            Wait_set_time = 0;
            t4sec = T4SEC;
        }

        if((key.h.mode_key == M1) &&
             (imsi_cal_set == 1)) cal_set22 = 1;
        else {
             cal_set22 = 0;
             imsi_cal_set = 0;
        }

        // Auto Calibration
        if((cal_set22 == 1) && (Wait_set_time
                        == 1)) Set_mode = 1;

        // Auto Calibration 변수를 입력하는
          Mode
        while(Set_mode) {
            if(Buz_var == 0) {
               _BUZZER = LO;
               BZ_sound = 0;
               t500ms = T500MS;
               Buz_var = 1;
            }

            if(BZ_sound == 1) {
                _BUZZER = HI;
                BZ_sound = 0;
                t500ms = T500MS;
            }

            key.x.word = getkey();

            Wdt_out(); // WatchDog 신호

            calibration_mode(); // 교정하여
            교정값을 SAVE 시키는 Mode

            key.x.word = getkey();
            // Up Key와 Shift Key를 동시에
            누르면 Calibration Mode에서 나온다.
            if((key.h.up_key == M1) &&
                      (key.h.shift_key == M1)){
                Set_mode = 0;
                Buz_var = 0;
                DS1305_SetValue_Extract();
                cal_set22 = 0;
                Wait_set_time = 0;
```

```
            imsi_cal_set = 0;
        }
        first1 = 0;
} // while(Set_mode) ends

  /* Key 발생이 없을 때 1분 뒤에 Back
  Light Off 하고 2분 뒤에는 LCD off
     Key가 다시 발생 하면 LCD On 시키고
     Back Light도 On 시킨다. */
    if((Lcd_backlight_time == 0) &&
       (Power33 == 1)) BACK_LIGHT = HI;
    else BACK_LIGHT = LO;

    if(key.h.shift_key == M2) {
       Analog_Power = HI;
       System_Power  = LO;
       Init_Lcd();
       Power33 = 1;
       BACK_LIGHT = HI;
    }

    if(key.h.up_key == M2) {
       EX0 = 0;
            // IE.0  INT0 Interrupt Enable
       ET0 = 0;
           // IE.1  TIMER0 Interrupt Enable
       ES  = 0;
       // IE.4  시리얼 포트 인터럽트 Enable
       TR0 = 0;  // TCON.4  Timer 0 Run
       TR1 = 0;  // TCON.6  Timer 1 Run
       EA  = 0;  // IE.7  Interrupt enable
       // P0 = 0xFF;
       // P1 = 0xFF;
       // P2 = 0xFF;
       // PCON |= 0x03;   // Clear IDL bit
              in power control register
       System_Power = HI;
       // PCON |= 0x02;   // Clear STOP bit
              in power control register
       Analog_Power = LO;
       // imsi77 =
              Read_DS1305(R_STA_REG);
                // Status register reading
    }
    Wdt_out(); // WatchDog 신호
  } // while(1) Ends
} // End main()

// MPU를 위한 초기화
void initial_int(void) {
    SCON = 0x52;  // Setup serial port
                        control register
                  // mode 1, 8 - bit UART,
                        Enable Receiver
    PCON &= 0x7f;  // Clear SMOD bit in
                      power control register
    PCON |= 0x08;
    TMOD &= 0xcf;  // Set timer/ counter
                        mode register clear
                   // M1 & M0 for timer 1
    TMOD = 0X21;   // Timer 1, mode 2, 8
                        - bit autoreload
                  // Timer 0, 16-bit Timer
    TH1  = 0xfd;   // Set autoreload value
                     for timer 1, 9600 baud

    TH0 = 0x4c;    // 50ms set
    TL0 = 0x00;
    MCON = 0xCa; // MSEL 단자(Pin 14)
                      = 0, RG1(MCON.3) = 1
    RPCTL = 0x01;  // RG0(RPCTL.0) = 1

    PS  = 1;
    IT0 = 1;
      // TCON.0  INT0 Falling edge 동작
    EX0 = 1;
      // IE.0  INT0 Interrupt Enable
    ET0 = 1;
      // IE.1  TIMER0 Interrupt Enable
    ES  = 1;
      // IE.4  시리얼 포트 인터럽트 Enable
    TR0 = 1;  // TCON.4  Timer 0 Run
    TR1 = 1;  // TCON.6  Timer 1 Run
    EA  = 1;  // IE.7  Interrupt enable
    TI = 0;
    RI = 0;
}

/*
  Time Delay Rountine, Software 적
  Time Delay임
*/
```

```
void delay(int i) {
     int delay_watchdog = 0;
     while(i) {
          i--;
          /* Delay 시간이 너무 길어지면
     Watchdog에 뿌려줄 신호 때문에 */
          delay_watchdog++;
          if(delay_watchdog > 1000) {
               _WATCHDOG = LO;
               _WATCHDOG = HI;
               delay_watchdog = 0;
          }
     }
}

/*
     MCS51 의 Port 초기화
                                        */
void init_port(void) {
     System_Power  = HI;
     SCLKi         = LO;
     _DIO          = LO;
     _RST          = LO;
     Analog_Power  = HI;
     _BUZZER       = HI;
     Analog_SEL1   = LO;
     Analog_SEL2   = LO;
     LCD_RS        = LO;
     LCD_RW        = LO;
     LCD_E         = LO;
     BACK_LIGHT    = LO;
     _WATCHDOG     = LO;
     _WATCHDOG     = HI;
     delay(300);
}

/* WatchDog에 보내는 신호 */
void Wdt_out() {
     _WATCHDOG = LO;  delay(10);
     _WATCHDOG = HI;
}

// Buzzer Sound Output
void Buz_sound() {
     _BUZZER = LO;
     delay(10000);
     _BUZZER = HI;
}

/*
     LCD DIsplay를 위한 함수들
                                        */
void E_Pulse() { /* Enable Pulse */
     char i;
     LCD_E = 1; /* LCD Enable 신호
               'HIGH' */
     for(i=0;i<10;i++); /* LCD Enable 펄스를
어느 시간 이상 지연해 준다. */
     LCD_E = 0; /* LCD Enable 신호
                    'LOW' */
}

/* LCD 컨트롤러 HD44780의 기능을 설정하는
함수로 데이터 라인을 8비트 인터페이스,
   2라인 디스플레이, 5 X 7 글자폰트를 설정
   한다. */
void Func_set() {
     LCD_RW = 0;
     LCD_RS = 0;

     PE_EN();  // PES=1, MCON 1=0x04로
               치환, 스크랩되지 않음
     outportb(OUT_LCD_ADDR,0x38);
      /* LCD DB0 ~ DB7까지의 값이다. */
     XDATA_EN();  // 다시 복귀 MCON &
       0xFB로 치환, 외부 메모리와 스크랩됨

     /* 모든 명령(데이터 입력) 후 반드시
      LCD Enable 펄스를 하나 내주어야
 한다. */
     E_Pulse();
}

 void Init_Lcd() { /* LCD를 초기화 함수이다.
*/
     int i;
     LCD_E = 0;
     for(i=0;i<200;i++); /* 전원이 4.5V 이상
                 으로 올라간 후에
              15ms 이상을 기다려
          야 한다. */
     Func_set();
     for(i=0;i<100;i++); /* 4.1ms 이상 기다려
                 야 한다. */
```

```
    Func_set();
    for(i=0;i<20;i++);   /* 100㎲ 이상 기다려
                야 한다. */

    Func_set();

     PE_EN();  // PES=1, MCON 1=0x04
              로 치환, 스크랩되지 않음
    /* 디스플레이 ON/OFF 제어 : 디스플레
    이와 커서를 ON하고, 커서는 깜빡이게 된다.
    초기화 직후이기 때문에 글자는 하나도 나타
    나지 않는다. */
    outportb(OUT_LCD_ADDR,0x0C);
    XDATA_EN();  // 다시 복귀 MCON &
                0xFB로  치환,  외부
              메모리와 스크랩됨

    E_Pulse();

    PE_EN();  // PES=1, MCON 1=0x04로
              치환, 스크랩되지 않음
    /* 엔트리 모드 세트 : 어드레스는 +1,
      D.D.  /C.G.RAM  쓰기  후에  커서는
  오른쪽으로  시프트하도록  한다.  이때
디스플레이는 시프트하지 않는다. */
    outportb(OUT_LCD_ADDR,0x06);
    XDATA_EN(); // 다시 복귀 MCON &
                0xFB로 치환, 외부
                메모리와 스크랩됨

    E_Pulse();
}

void write_char(unsigned char s) { /* LCD에
                문자를 한개 써넣기 */
    data unsigned char munjja;

    munjja = s;

    LCD_RS = 1;

    PE_EN();  // PES=1, MCON 1=0x04로
              치환, 스크랩되지 않음
    outportb(OUT_LCD_ADDR, munjja);
    XDATA_EN(); // 다시 복귀 MCON &
              0xFB로  치환,  외부  메모리
            와 스크랩됨
    E_Pulse(); /* Enable 신호로 Write를
              허가 */
}

// lcd의 주소를 찾아 문자열을 wirte하는 루틴
void lcd_printf(unsigned char lcd_addr,
            unsigned char *c) {
    data unsigned char addr;

    addr = lcd_addr;
    LCD_RS = 0;
    LCD_RW = 0;
    PE_EN();  // PES=1, MCON 1=0x04로
              치환, 스크랩되지 않음
    outportb(OUT_LCD_ADDR, addr);
    XDATA_EN();  // 다시 복귀 MCON
                & 0xFB로 치환, 외부
              메모리와 스크랩됨

    E_Pulse();     // LCD에 써넣기 허가
    while(*c) {
       write_char(*c++);
   }
}

// LCD에 한주소 문자를 한개 써넣기
void Lcd_OneCharWrite(unsigned char
            lcd_addr,  unsigned char s) {
    data unsigned char addr;

    addr = lcd_addr;
    LCD_RS = 0;
    LCD_RW = 0;
    PE_EN();  // PES=1, MCON
        1=0x04로 치환, 스크랩되지 않음
    outportb(OUT_LCD_ADDR, addr);
    XDATA_EN();      //  다시  복귀  MCON
          &  0xFB로  치환,  외부
       메모리와 스크랩됨
    E_Pulse();     // LCD에 써넣기 허가
    write_char(s);
}

// DS1305 처음으로 동작시킬때 사용한다.
void Init_DS1305() {
    SCLKi = LO;
    _RST = LO;
}

// DS1305로 데이타를 쓴다.
void Write_DS1305(unsigned char addr
            ,unsigned char value) {
    unsigned char i;
```

```
    _RST = LO;
    _nop_();
    SCLKi = LO;
    _nop_();
    _RST = HI;

    for(i=0; i<8;i++) {
        if(addr & 1) _DIO = HI;
        else _DIO = LO;
        _nop_();
        SCLKi = HI;
        _nop_();
        _nop_();
        _nop_();
        SCLKi = LO;
        addr >>= 1;
    }

    for(i=0; i<8; i++) {
        if(value & 1) _DIO = HI;
        else _DIO = LO;
        _nop_();
        SCLKi = HI;
        _nop_();
        _nop_();
        _nop_();
        SCLKi = LO;
        _nop_();
        value >>= 1;
    }
    _RST = LO;
}

unsigned char Read_DS1305(unsigned char addr) {
    unsigned char i,ret,aa;

    _RST = LO;
    _nop_();
    SCLKi = LO;
    _nop_();
    _RST = HI;

    for(i=0; i<8;i++) {
        if(addr & 1) _DIO = HI;
        else _DIO = LO;
        _nop_();
        SCLKi = HI;
        _nop_();
        _nop_();
        SCLKi = LO;
        _nop_();
        addr >>= 1;
    }

    ret=0;
    aa=1;
    _DIO = HI;
        // 입력포트값을 올바르게 읽기위함
    for(i=0; i<8; i++) {
        if(_DIO) ret |= aa;
        _nop_();
        SCLKi = HI;
        _nop_();
        SCLKi = LO;
        _nop_();
        aa = aa << 1;
    }
    _RST = LO;
    return(ret);
}

/* 본장비의 전원을 On 시키면 처음으로 실행
하는 루틴 */
void At_PowerON() {
    char i;

    initial_int();   /*   Interrupt   초기화
*/
    init_port(); /* MCS51 의 Port 1 의
                    초기화        */
    set_time(); /* Time 변수들에 초기값을
                   입력한다. */
    delay(20000); /* TIME DELAY */
    delay(20000);

    Init_DS1305();  /* DS1305 초기화 */
    Init_Lcd();   /* LCD를 초기화하는
                     함수 */
    Wdt_out();

    /* LCD에 초기 Message 표시 */
    //                0123456789012345678 9
    lcd_printf(0x80, "Water Analysis   ");
    lcd_printf(0xC0, "Water Pollution  ");
    delay(20000);   // TIME DELAY
    System_Power  = LO;
    Wdt_out();
```

```
        lcd_printf(0x80, "Data Gathering   ");
        lcd_printf(0xC0, "Equipment       ");
        delay(20000);   // TIME DELAY
        Wdt_out();


        Ctrl_RTC.x.word =
            Read_DS1305(R_CONTROL_ADDR);
                    // 시계 IC(DS1305)의
                      Control Register reading
        if(Ctrl_RTC.h.EOSC == HI) {
           Ctrl_RTC.h.EOSC = LO;
                    // Start the osillator
           Ctrl_RTC.h.WP = LO;
                    // Write Permit
           Write_DS1305(W_CONTROL_ADDR,
           Ctrl_RTC.x.word);
                    // Write Permit
           Ctrl_RTC.h.INTCN = HI;
                    // Activate ALARM
           Ctrl_RTC.h.AIE1 = HI;
                    // Permits the Interrupt 1
           Ctrl_RTC.h.AIE0 = HI;
                    // Permits the Interrupt 0
           Write_DS1305(W_CONTROL_ADDR,
Ctrl_RTC.x.word);
                    // Write Permit
           Ctrl_RTC.h.WP = HI;
           Write_DS1305(W_CONTROL_ADDR,
Ctrl_RTC.x.word);
                    // Write Protect
        }

        if(Ctrl_RTC.h.INTCN == LO) {
           Ctrl_RTC.h.WP = LO;
                    // Write Permit
           Ctrl_RTC.h.INTCN = HI;
                    // Activate ALARM
           Ctrl_RTC.h.AIE1 = HI;
                    // Permits the Interrupt 1
           Ctrl_RTC.h.AIE0 = HI;
                    // Permits the Interrupt 0
           Write_DS1305(W_CONTROL_ADDR,
Ctrl_RTC.x.word);
                    // Write Permit
           Ctrl_RTC.h.WP = HI;
           Write_DS1305(W_CONTROL_ADDR,
Ctrl_RTC.x.word);
                    // Write Protect
        }

        /* DS1305에서 Data 읽기 */
        time_read();

        DS1305_SetValue_Extract();

        SMD_init();
        Wdt_out();

        for(i=0;i<35;i++) {
            Now_Data[i] = NowDT007[i];
        }
    }

    /*
       Key 눌림상태 정의
                                        */
    char CHKPLS(struct KEY_STAT
                                *cp,charky) {
        char result = M0; /* Disuse Condition */
        if(ky == HI) {    /* Key 눌림 감지 */
           if(cp->CP2 == LO) { /* Key 눌림
                                  초기 루틴 */
              if(cp->CP1  == HI) { /* Key 감지
                                  두번째 */
                 cp->CP2 = HI;
                 cp->CP1 = LO;
                 result  = M3;  /* Rising
                                  edge mode
                                  declarration */
              }
              else cp->CP1 = HI;  /* Key
                                  감지 첫번째 */
           }
           else {  /* Key 발생이 계속되는 상태
                      (CP2 == HI) */
              cp->CP1 = HI;
              result = M1; /* Continued KEY
                   occupy mode declaration */
              tlmin = TIMIN;
              Lcd_backlight_time = 0;
           }
        }
        else {  /* KEY 발생이 없는 상태 */
           if(cp->CP2 == HI) {  /* KEY 발생이
                                  없어지는 첫번째 */
              if(cp->CP1 == HI)  {
                 result = M2; /* Falling edge
                              mode declaration */
                 Buz_sound();
```

```
            }
        }
        cp->CP1 = LO;
        cp->CP2 = LO;
    }
    return(result);
}

/* Key Scan routine  */
char getkey() {
    key.h.mode_key
    = CHKPLS(&mode_key, ~(Mode_KEY));
    key.h.shift_key
    = CHKPLS(&shift_key, ~(Shift_KEY));
    key.h.up_key
    = CHKPLS(&up_key, ~(Up_KEY));
    key.h.enter_key
    = CHKPLS(&enter_key, ~(Enter_KEY));
    return(key.x.word);
}

// System의 Data 송신
void tx232_1() {
    int i;

    ES = 0;
    TI = 0;

    SBUF = STX;  while(!TI) ;TI=0;
    SBUF = 'P';  while(!TI) ;TI=0;
    SBUF = 'E';  while(!TI) ;TI=0;
    SBUF = 'S';  while(!TI) ;TI=0;
    SBUF = 'D';  while(!TI) ;TI=0;
    SBUF = 'A';  while(!TI) ;TI=0;
    SBUF = 'T';  while(!TI) ;TI=0;
    SBUF = 'A';  while(!TI) ;TI=0;
    SBUF = 0xFF; while(!TI) ;TI=0;
    SBUF = 0xFF; while(!TI) ;TI=0;

    for( i=0; i < 5000; i++ ) {
        SBUF = PES_Out_Data[i].Bytes1;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes2;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes3;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes4;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes5;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes6;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes7;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes8;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes9;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes10;
                while(!TI) ;TI=0;
        SBUF = PES_Out_Data[i].Bytes11;
                while(!TI) ;TI=0;
        Wdt_out(); /* WatchDog 신호 */
    }

    SBUF = 0xFF; while(!TI) ;TI=0;
    SBUF = 0xFE; while(!TI) ;TI=0;
    // CRC Check, Check Sum
    SBUF = 0x00; while(!TI) ;TI=0;
    SBUF = 0x00; while(!TI) ;TI=0;
    SBUF = 0x00; while(!TI) ;TI=0;
    SBUF = 0x00; while(!TI) ;TI=0;
    SBUF = ETX;  while(!TI) ;TI=0;

    ES = 1;
    Wdt_out(); /* WatchDog 신호 */
}

/* 현재 Logger의 Data 송신 */
void tx232_2() {
    char i;

    for( i=0; i < 35; i++ ) {
        Set_Cmd[i] = Now_Data[i];
    }
    // 현재 Logger의 Data 정리

    ES = 0;
    TI = 0;
    for( i=0; i < 35; i++ ) {
        SBUF = Now_Data[i];
                while(!TI) ;TI=0;
    }
    ES = 1;
    Wdt_out(); /* WatchDog 신호 */
}

// System의 Data 송신
void tx232_3() {
    int i;
```

```
ES = 0;
TI = 0;

SBUF = STX;  while(!TI) ;TI=0;
SBUF = 'P';  while(!TI) ;TI=0;
SBUF = 'E';  while(!TI) ;TI=0;
SBUF = 'S';  while(!TI) ;TI=0;
SBUF = 'D';  while(!TI) ;TI=0;
SBUF = 'A';  while(!TI) ;TI=0;
SBUF = 'T';  while(!TI) ;TI=0;
SBUF = 'A';  while(!TI) ;TI=0;
SBUF = 0xFF; while(!TI) ;TI=0;
SBUF = 0xFF; while(!TI) ;TI=0;

for( i=0; i < 5; i++ ) {
     SBUF = PES_Out_Data[i].Bytes1;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes2;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes3;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes4;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes5;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes6;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes7;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes8;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes9;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes10;
               while(!TI) ;TI=0;
     SBUF = PES_Out_Data[i].Bytes11;
               while(!TI) ;TI=0;
     Wdt_out(); /* WatchDog 신호 */
}

SBUF = 0xFF; while(!TI) ;TI=0;
SBUF = 0xFE; while(!TI) ;TI=0;
// CRC Check, Check Sum
SBUF = 0x00; while(!TI) ;TI=0;
SBUF = 0x00; while(!TI) ;TI=0;
SBUF = 0x00; while(!TI) ;TI=0;
SBUF = 0x00; while(!TI) ;TI=0;
SBUF = ETX;  while(!TI) ;TI=0;
ES = 1;
     Wdt_out(); /* WatchDog 신호 */
}


void time_wr() {     /* time 설정 */
     Ctrl_RTC.h.WP = LO;
     Write_DS1305(W_CONTROL_ADDR,
          Ctrl_RTC.x.word); // Write Permit
     Write_DS1305(W_SEC_ADDR,
          Second[1].x.word);   // sec write
     Write_DS1305(W_MIN_ADDR,
          Minute[1].x.word);   // min write
     Write_DS1305(W_HOUR_ADDR,
          Hour[1].x.word);     // hr write
     Write_DS1305(W_DATE_ADDR,
          Date[1].x.word);     // date write
     Write_DS1305(W_MONTH_ADDR,
          Month[1].x.word);  // month write
     Write_DS1305(W_YEAR_ADDR,
          Year[1].x.word);     // year write
     Ctrl_RTC.h.WP = HI;
     Write_DS1305(W_CONTROL_ADDR,
          Ctrl_RTC.x.word); // Write Protect
}

/*************************
     현재시간을 읽는다.
*************************/
void time_read() {
     int temp = 0;
     unsigned char imsi = 0;
     Second[0].x.word =
          Read_DS1305(R_SEC_ADDR);
                    // 초 reading
     Minute[0].x.word =
          Read_DS1305(R_MIN_ADDR);
                    // 분 reading
     Hour[0].x.word   =
          Read_DS1305(R_HOUR_ADDR);
                    // 시간  reading
     Date[0].x.word   =
          Read_DS1305(R_DATE_ADDR);
                    // date  reading
     Month[0].x.word  =
          Read_DS1305(R_MONTH_ADDR);
                    // month reading
     Year[0].x.word   =
          Read_DS1305(R_YEAR_ADDR);
```

```
                    // year  reading

    temp = temp + ((int) Hour[0].h.HR_10
                    * 600);
    temp = temp + ((int) Hour[0].h.HR_1
                    * 60);
    temp = temp + ((int)Minute[0].h.MIN_10
                    * CDS2);
    temp = temp + ((int) Minute[0].h.MIN_1
                    * CDS1);

    Conversion_min = temp;

    imsi = 0;
    imsi = imsi + (Month[0].h.MONTH_1
                    * CDS1);
    imsi = imsi + (Month[0].h.MONTH_10
                    * CDS2);
    PES_Div_Data.h.Month12 = imsi;

    imsi = 0;
    imsi = imsi + (Date[0].h.DATE_1
                    * CDS1);
    imsi = imsi + (Date[0].h.DATE_10
                    * CDS2);
    PES_Div_Data.h.Day31 = imsi;

    imsi = 0;
    imsi = imsi + (Hour[0].h.HR_1
                    * CDS1);
    imsi = imsi + (Hour[0].h.HR_10
                    * CDS2);
    PES_Div_Data.h.Hour24 = imsi;

    imsi = 0;
    imsi = imsi + (Minute[0].h.MIN_1
                    * CDS1);
    imsi = imsi + (Minute[0].h.MIN_10
                    * CDS2);
    PES_Div_Data.h.Minute60 = imsi;

    imsi = 0;
    imsi = imsi + (Second[0].h.SEC_1
                    * CDS1);
    imsi = imsi + (Second[0].h.SEC_10
                    * CDS2);
    PES_Div_Data.h.Second60 = imsi;
}
```

```
// DS1305에 저장된 각종 Setting값 추출
void DS1305_SetValue_Extract() {
    Save_Value7[0].h.LowByte  =
          Read_DS1305(R_PRES11z_ADDR);
    Save_Value7[0].h.HighByte  =
          Read_DS1305(R_PRES12z_ADDR);
    Save_Value7[1].h.LowByte  =
          Read_DS1305(R_PRES21s_ADDR);
    Save_Value7[1].h.HighByte  =
          Read_DS1305(R_PRES22s_ADDR);
    Save_Value7[2].h.LowByte  =
          Read_DS1305(R_CON11z_ADDR);
    Save_Value7[2].h.HighByte  =
          Read_DS1305(R_CON12z_ADDR);
    Save_Value7[3].h.LowByte  =
          Read_DS1305(R_CON21s_ADDR);
    Save_Value7[3].h.HighByte  =
          Read_DS1305(R_CON22s_ADDR);

    Save_Value7[13].h.LowByte  =
          Read_DS1305(R_CON11z_ADDR);
    Save_Value7[13].h.HighByte  =
          Read_DS1305(R_CON12z_ADDR);
    Save_Value7[14].h.LowByte  =
          Read_DS1305(R_CON21s_ADDR);
    Save_Value7[14].h.HighByte  =
          Read_DS1305(R_CON22s_ADDR);

    Save_Value7[4].h.LowByte  =
          Read_DS1305(R_TEMP11z_ADDR);
    Save_Value7[4].h.HighByte  =
          Read_DS1305(R_TEMP12z_ADDR);
    Save_Value7[5].h.LowByte  =
          Read_DS1305(R_TEMP21s_ADDR);
    Save_Value7[5].h.HighByte  =
          Read_DS1305(R_TEMP22s_ADDR);

    Save_Value7[6].h.LowByte  =
          Read_DS1305(R_Depth11z_ADDR);
    Save_Value7[6].h.HighByte  =
          Read_DS1305(R_Depth12z_ADDR);
    Save_Value7[7].h.LowByte  =
          Read_DS1305(R_Depth21s_ADDR);
    Save_Value7[7].h.HighByte  =
```

```
        Read_DS1305(R_Depth22s_ADDR);
Save_Value7[8].h.LowByte   =
        Read_DS1305(R_EC11z_ADDR);
Save_Value7[8].h.HighByte  =
        Read_DS1305(R_EC12z_ADDR);

Save_Value7[9].h.LowByte   =
        Read_DS1305(R_EC21s_ADDR);
Save_Value7[9].h.HighByte  =
        Read_DS1305(R_EC22s_ADDR);

Save_Value7[15].h.LowByte   =
        Read_DS1305(R_EC31z_ADDR);
Save_Value7[15].h.HighByte  =
        Read_DS1305(R_EC32z_ADDR);
Save_Value7[16].h.LowByte   =
        Read_DS1305(R_EC41s_ADDR);
Save_Value7[16].h.HighByte  =
        Read_DS1305(R_EC42s_ADDR);

Save_Value7[10].h.LowByte  =
        Read_DS1305(R_Temp31z_ADDR);
Save_Value7[10].h.HighByte =
        Read_DS1305(R_Temp32z_ADDR);
Save_Value7[11].h.LowByte  =
        Read_DS1305(R_Temp41s_ADDR);
Save_Value7[11].h.HighByte =
        Read_DS1305(R_Temp42s_ADDR );
Save_Value7[12].h.LowByte  =
        Read_DS1305(R_STHint_ADDR);
Save_Value7[12].h.HighByte =
        Read_DS1305(R_STMint_ADDR);

Depth_DPL = Save_Value7[0].x.word;
Depth_DPH = Save_Value7[1].x.word;

EC_PD1    = Save_Value7[2].x.word;
EC_PD2    = Save_Value7[3].x.word;
EC_PD3    = Save_Value7[13].x.word;
EC_PD4    = Save_Value7[14].x.word;

Temp_PDL  = Save_Value7[4].x.word;
Temp_PDH  = Save_Value7[5].x.word;

DPV_Depth = Depth_DPH - Depth_DPL;
DPV_Ec1   = EC_PD2 - EC_PD1;
DPV_Ec2   = EC_PD3 - EC_PD2;
DPV_Ec3   = EC_PD4 - EC_PD3;
// DPV_Ec4    = EC_PD2 - EC_PD1;
DPV_Temp = Temp_PDH - Temp_PDL;

    depth_spl = Save_Value7[6].x.word;
    depth_sph = Save_Value7[7].x.word;

    ec_sp1    = Save_Value7[8].x.word;
    ec_sp2    = Save_Value7[9].x.word;
    ec_sp3    = Save_Value7[15].x.word;
    ec_sp4    = Save_Value7[16].x.word;

    temp_spl  = Save_Value7[10].x.word;
    temp_sph  = Save_Value7[11].x.word;

    spv_depth = depth_sph - depth_spl;
    spv_ec1   = ec_sp2 - ec_sp1;
    spv_ec2   = ec_sp3 - ec_sp2;
    spv_ec3   = ec_sp4 - ec_sp3;
    spv_temp  = temp_sph - temp_spl;

    total_min  = Save_Value7[12].x.word;
}

/* ************************************
 Part of routine for data fetched of
 A/D(TC835)
 Revision History :
 05/12/2001 - intial release
              data sensing for A/D
              converter & coding
              Used TC835( A/D Converter )
              - about 16 bit
   ************************************/
int  getadc() {
     int result = 0;

     if((TC835[0].h.BCD >= 0) &&
                (TC835[0].h.BCD <= 9)) {
        icm[0].h.bcd = TC835[0].h.BCD;
        TC835[5].h.BCD = TC835[0].h.BCD;
     }
     else icm[0].h.bcd = TC835[5].h.BCD;

     if((TC835[1].h.BCD >= 0) &&
                (TC835[1].h.BCD <= 9)) {
        icm[1].h.bcd = TC835[1].h.BCD;
        TC835[6].h.BCD = TC835[1].h.BCD;
     }
     else icm[1].h.bcd = TC835[6].h.BCD;

     if((TC835[2].h.BCD >= 0) &&
                (TC835[2].h.BCD <= 9)) {
```

```
        icm[2].h.bcd = TC835[2].h.BCD;
        TC835[7].h.BCD = TC835[2].h.BCD;
    }
    else icm[2].h.bcd = TC835[7].h.BCD;


    if((TC835[3].h.BCD >= 0) &&
              (TC835[3].h.BCD <= 9)) {
        icm[3].h.bcd = TC835[3].h.BCD;
        TC835[8].h.BCD = TC835[3].h.BCD;
    }
    else icm[3].h.bcd = TC835[8].h.BCD;

    if((TC835[4].h.BCD >= 0) &&
              (TC835[4].h.BCD <= 9)) {
        icm[4].h.bcd = TC835[4].h.BCD;
        TC835[9].h.BCD = TC835[4].h.BCD;
    }
    else icm[4].h.bcd = TC835[9].h.BCD;

    result = result + ((int) icm[0].h.bcd
                     * 1);
    result = result + ((int) icm[1].h.bcd
                     * 10);
    result = result + ((int) icm[2].h.bcd
                     * 100);
    result = result + ((int) icm[3].h.bcd
                     * 1000);
    result = result + ((int) icm[4].h.bcd
                     * 10000);

    if(TC835[4].h.OVER_RANGE ==
                  HI) result = 20000;
    else {
        if(TC835[4].h.SIGN == LO) {
            result = result * (-1);
        }
    }
    return(result);
}

// A/D Converter값 정리
void AD_theorem() {
    long result = 0;

    if((TC835[0].h.BCD >= 0) &&
              (TC835[0].h.BCD <= 9)) {
        icm[0].h.bcd = TC835[0].h.BCD;
        TC835[5].h.BCD = TC835[0].h.BCD;
    }
    else icm[0].h.bcd = TC835[5].h.BCD;

    if((TC835[1].h.BCD >= 0) &&
              (TC835[1].h.BCD <= 9)) {
        icm[1].h.bcd = TC835[1].h.BCD;
        TC835[6].h.BCD = TC835[1].h.BCD;
    }
    else icm[1].h.bcd = TC835[6].h.BCD;

    if((TC835[2].h.BCD >= 0) &&
              (TC835[2].h.BCD <= 9)) {
        icm[2].h.bcd = TC835[2].h.BCD;
        TC835[7].h.BCD = TC835[2].h.BCD;
    }
    else icm[2].h.bcd = TC835[7].h.BCD;

    if((TC835[3].h.BCD >= 0) &&
              (TC835[3].h.BCD <= 9)) {
        icm[3].h.bcd = TC835[3].h.BCD;
        TC835[8].h.BCD = TC835[3].h.BCD;
    }
    else icm[3].h.bcd = TC835[8].h.BCD;

    if((TC835[4].h.BCD >= 0) &&
              (TC835[4].h.BCD <= 9)) {
        icm[4].h.bcd = TC835[4].h.BCD;
        TC835[9].h.BCD = TC835[4].h.BCD;
    }
    else icm[4].h.bcd = TC835[9].h.BCD;

    result = result + ((long) icm[0].h.bcd
                     * 1);
    result = result + ((long) icm[1].h.bcd
                     * 10);
    result = result + ((long) icm[2].h.bcd
                     * 100);
    result = result + ((long) icm[3].h.bcd
                     * 1000);
    result = result + ((long) icm[4].h.bcd
                     * 10000);

    if(TC835[4].h.OVER_RANGE ==
                  HI) result = 20000;
    else {
        if(TC835[4].h.SIGN == LO) {
            result = result * (-1);
        }
    }
    Na[digit] = result;
}
```

```
long Get_ADValue() {
     long result = 0;
     AD_theorem();
     result = (long) ((Na[0]+Na[1]+Na[2]
                    +Na[3]+Na[4]+Na[5]
                    +Na[6]+Na[7]+Na[8]
                    +Na[9]) / 10);
     return(result);
}

// PC에서 들어온 명령어 분리 작업
void pc_cmd_div() {
     char i;
     char value = 0;

     /* System의 측정 Data 전체 요구 */
     if((SIO_CMD[0]==STX)&&
       (SIO_CMD[1]=='R')&&
       (SIO_CMD[2]=='E')&&
       (SIO_CMD[3]=='Q')&&
       (SIO_CMD[4]=='F')&&
       (SIO_CMD[5]=='U')&&
       (SIO_CMD[6]=='L')&&
       (SIO_CMD[7]=='L')&&
       (SIO_CMD[12]==ETX)) {
       value = 1;
     }

     /* System의 현재 측정 Data 요구 */
     if((SIO_CMD[0]==STX)&&
       (SIO_CMD[1]=='P')&&
       (SIO_CMD[2]=='R')&&
       (SIO_CMD[3]=='E')&&
       (SIO_CMD[4]=='S')&&
       (SIO_CMD[5]=='E')&&
       (SIO_CMD[6]=='N')&&
       (SIO_CMD[7]=='T')&&
       (SIO_CMD[12]==ETX)) {
       value = 2;
     }

     /* System의 시간 설정 */
     if((SIO_CMD[0]==STX)&&
       (SIO_CMD[1]=='M')&&
       (SIO_CMD[2]=='O')&&
       (SIO_CMD[3]=='D')&&
       (SIO_CMD[4]=='I')&&
       (SIO_CMD[5]=='F')&&
       (SIO_CMD[6]=='Y')&&
       (SIO_CMD[7]=='t')&&
       (SIO_CMD[22]==ETX)) {
       value = 3;
       for(i = 0; i < 10; i++) {
            Set_Cmd[i] = SIO_CMD[i+8];
        }
     }

     /* System의 Data를 저장하는 시간
        간격을 설정 */
     if((SIO_CMD[0]==STX)&&
       (SIO_CMD[1]=='D')&&
       (SIO_CMD[2]=='S')&&
       (SIO_CMD[3]=='I')&&
       (SIO_CMD[4]=='T')&&
       (SIO_CMD[5]=='I')&&
       (SIO_CMD[6]=='M')&&
       (SIO_CMD[7]=='E')&&
       (SIO_CMD[22]==ETX)) {
       value = 4;
       for(i = 0; i < 10; i++) {
            Set_Cmd[i] = SIO_CMD[i+8];
        }
     }

     /* System를 Reset 시킨다. */
     if((SIO_CMD[0]==STX)&&
       (SIO_CMD[1]=='C')&&
       (SIO_CMD[2]=='L')&&
       (SIO_CMD[3]=='S')&&
       (SIO_CMD[4]=='D')&&
       (SIO_CMD[5]=='A')&&
       (SIO_CMD[6]=='T')&&
       (SIO_CMD[7]=='A')&&
       (SIO_CMD[12]==ETX)) {
       value = 5;
```

```
    }

    /* System의 측정 Data 5회분 요구 */
    if((SIO_CMD[0]==STX)&&
      (SIO_CMD[1]=='F')&&
      (SIO_CMD[2]=='I')&&
      (SIO_CMD[3]=='V')&&
      (SIO_CMD[4]=='E')&&
      (SIO_CMD[5]=='D')&&
      (SIO_CMD[6]=='A')&&
      (SIO_CMD[7]=='T')&&
      (SIO_CMD[12]==ETX)) {
      value = 6;
    }

    Receive_inform = value;
    Wdt_out(); /* WatchDog 신호 */

}

void DS1305_SettingUp() {
    char i = 0;

    time_read();

    Second[1].x.word = Second[0].x.word;
    Minute[1].x.word = Minute[0].x.word;
    Hour[1].x.word   = Hour[0].x.word;
    Date[1].x.word   = Date[0].x.word;
    Month[1].x.word  = Month[0].x.word;
    Year[1].x.word   = Year[0].x.word;

    for(i=0;i<10;i++) {
        icm[i].x.word = Set_Cmd[i];
    }

    Second[1].h.SEC_1    = 0;
    Second[1].h.SEC_10   = 0;
    Minute[1].h.MIN_1    = icm[9].h.bcd;
    Minute[1].h.MIN_10   = icm[8].h.bcd;
    Hour[1].h.HR_1       = icm[7].h.bcd;
    Hour[1].h.HR_10      = icm[6].h.bcd;
    Hour[1].h.HR12_24    = LO;
    Date[1].h.DATE_1     = icm[5].h.bcd;
    Date[1].h.DATE_10    = icm[4].h.bcd;
    Month[1].h.MONTH_1   = icm[3].h.bcd;
    Month[1].h.MONTH_10  = icm[2].h.bcd;
    Year[1].h.YEAR_1     = icm[1].h.bcd;
    Year[1].h.YEAR_10    = icm[0].h.bcd;
    time_wr();
    lcd_printf(0x80, "Setting Time
            & Date ");
    delay(20000);
    delay(20000);
}

// Time 변수들에 초기값을 입력한다.
void set_time() {
    RS232.h.Rx_sta   = LO;
    RS232.h.Tx_sta   = LO;
    RS232.h.Rx_end   = HI;
    RS232.h.Tx_end   = LO;
    RS232.h.Rx_Error = LO;
    RS232.h.Rx_Error = LO;

    t500ms = T500MS;
    t3sec = T3SEC;
    t4sec = T4SEC;
    tlmin = T1MIN;

    BZ_sound = 0;
    Exchange_time = 0;
    Wait_set_time = 0;
    Lcd_backlight_time = 0;
    Ram_CLS_time = 0;
    Receive_inform = 0x00;
    Lcd_Off_status = 0;
    Conversion_min = 0;
    digit = 0;
}

// 저장된 5000회 분의 Data를 Clear 시킨다.
void Ram_Clear_All() {
    int i = 0;
    for(i=0;i<5000;i++) {
        PES_Out_Data[i].Bytes1  = 0x00;
        PES_Out_Data[i].Bytes2  = 0x00;
        PES_Out_Data[i].Bytes3  = 0x00;
        PES_Out_Data[i].Bytes4  = 0x00;
        PES_Out_Data[i].Bytes5  = 0x00;
        PES_Out_Data[i].Bytes6  = 0x00;
        PES_Out_Data[i].Bytes7  = 0x00;
        PES_Out_Data[i].Bytes8  = 0x00;
        PES_Out_Data[i].Bytes9  = 0x00;
        PES_Out_Data[i].Bytes10 = 0x00;
        PES_Out_Data[i].Bytes11 = 0x00;
```

```
        Wdt_out(); /* WatchDog 신호 */
    }
}


// PC에서 들어온 명령어 초기화
void SMD_init() {
    char i;
    for(i=0;i<30;i++) {
        SIO_CMD[i] = 0;
    }
}

/*
-----------------------------------
교정하여 교정값을 SAVE 시키는 Mode
-----------------------------------
*/
void calibration_mode() {
    unsigned int a = 0;
    int leedong = 0;
    unsigned int sang = 0;
    unsigned int selv_cal = 0;
    unsigned char i = 0;
    int TopSync = 0;
    long imsi = 0;
    int AD_value3 = 0;
    int value0, value1, value2, value3,
       value4;
    int value5, value6, value7, value8,
       value9, temp;
    int AD_Temp = 0;

    union ASC_HEX1 CalMode[17];
    union ICM_DATA icm0[6];
    union ICM_DATA icm1[6];
    union ICM_DATA icm2[5];
    union ICM_DATA icm3[5];
    union ICM_DATA icm4[5];
    union ICM_DATA icm5[5];
    union ICM_DATA icm7[15];
    union ICM_DATA icm8[20];
    union ICM_DATA icm9[5];
    union ICM_DATA icm10[5];
    union ICM_DATA icm11[5];
    union ICM_DATA icm66[6];

    value0 = value1 = value2 = value3 =
            value4 = 0;
    value5 = value6 = value7 = value8 =
            value9 = temp = 0;
    Wdt_out(); /* WatchDog 신호 */

    CalMode[0].h.LowByte   =
        Read_DS1305(R_PRES11z_ADDR);
    CalMode[0].h.HighByte  =
        Read_DS1305(R_PRES12z_ADDR);
    CalMode[1].h.LowByte   =
        Read_DS1305(R_PRES21s_ADDR);
    CalMode[1].h.HighByte  =
        Read_DS1305(R_PRES22s_ADDR);
    CalMode[2].h.LowByte   =
        Read_DS1305(R_CON11z_ADDR);
    CalMode[2].h.HighByte  =
        Read_DS1305(R_CON12z_ADDR);
    CalMode[3].h.LowByte   =
        Read_DS1305(R_CON21s_ADDR);
    CalMode[3].h.HighByte  =
        Read_DS1305(R_CON22s_ADDR);
    CalMode[4].h.LowByte   =
        Read_DS1305(R_TEMP11z_ADDR);
    CalMode[4].h.HighByte  =
        Read_DS1305(R_TEMP12z_ADDR);
    CalMode[5].h.LowByte   =
        Read_DS1305(R_TEMP21s_ADDR);
    CalMode[5].h.HighByte  =
        Read_DS1305(R_TEMP22s_ADDR);
    CalMode[12].h.LowByte  =
        Read_DS1305(R_STHint_ADDR);
    CalMode[12].h.HighByte =
        Read_DS1305(R_STMint_ADDR);
    CalMode[13].h.LowByte  =
        Read_DS1305(R_CON31z_ADDR);
    CalMode[13].h.HighByte =
        Read_DS1305(R_CON32z_ADDR);
    CalMode[14].h.LowByte  =
        Read_DS1305(R_CON41s_ADDR);
    CalMode[14].h.HighByte =
        Read_DS1305(R_CON42s_ADDR);

    value0 = CalMode[0].x.word;
    value1 = CalMode[1].x.word;
    value2 = CalMode[2].x.word;
    value3 = CalMode[3].x.word;
    value4 = CalMode[4].x.word;
    value5 = CalMode[5].x.word;
    value6 = CalMode[12].x.word;
    value8 = CalMode[13].x.word;
    value9 = CalMode[14].x.word;
```

```
time_read();
Second[1].x.word = Second[0].x.word;
Minute[1].x.word = Minute[0].x.word;
Hour[1].x.word   = Hour[0].x.word;
Date[1].x.word   = Date[0].x.word;
Month[1].x.word  = Month[0].x.word;
Year[1].x.word   = Year[0].x.word;
Hour[1].h.HR12_24 = LO;

icm8[0].h.asc = 3;
icm8[1].h.asc = 3;
icm8[2].h.asc = 3;
icm8[3].h.asc = 3;
icm8[4].x.word = '/';
icm8[5].h.asc = 3;
icm8[6].h.asc = 3;
icm8[7].x.word = '/';
icm8[8].h.asc = 3;
icm8[9].h.asc = 3;
icm8[10].x.word = ' ';
icm8[11].h.asc = 3;
icm8[12].h.asc = 3;
icm8[13].x.word = ':';
icm8[14].h.asc = 3;
icm8[15].h.asc = 3;
icm8[16].x.word = ' ';
icm8[17].h.asc = 3;
icm8[18].h.asc = 3;
icm8[19].x.word = 's';

icm8[18].h.bcd = Second[1].h.SEC_1;
icm8[17].h.bcd = Second[1].h.SEC_10;
icm8[15].h.bcd  = Minute[1].h.MIN_1;
icm8[14].h.bcd  = Minute[1].h.MIN_10;
icm8[12].h.bcd  = Hour[1].h.HR_1;
icm8[11].h.bcd  = Hour[1].h.HR_10;
icm8[9].h.bcd  = Date[1].h.DATE_1;
icm8[8].h.bcd  = Date[1].h.DATE_10;
icm8[6].h.bcd  = Month[1].h.MONTH_1;
icm8[5].h.bcd = Month[1].h.MONTH_10;
icm8[3].h.bcd  = Year[1].h.YEAR_1;
icm8[2].h.bcd  = Year[1].h.YEAR_10;
icm8[1].h.bcd = 0;
icm8[0].h.bcd = 2;

// Depth Low Value
icm0[0].h.asc = 3;
icm0[1].h.asc = 3;
icm0[2].h.asc = 3;
icm0[3].x.word = '.';
icm0[4].h.asc = 3;
icm0[5].h.asc = 3;

icm0[5].h.bcd = value0 % 10; value0 =
                (int) (value0 / 10);
icm0[4].h.bcd = value0 % 10; value0 =
                (int) (value0 / 10);
icm0[2].h.bcd = value0 % 10; value0 =
                (int) (value0 / 10);
icm0[1].h.bcd = value0 % 10; value0 =
                (int) (value0 / 10);
icm0[0].h.bcd = value0 % 10;

// Depth High Value
icm1[0].h.asc = 3;
icm1[1].h.asc = 3;
icm1[2].h.asc = 3;
icm1[3].x.word = '.';
icm1[4].h.asc = 3;
icm1[5].h.asc = 3;

icm1[5].h.bcd = value1 % 10; value1 =
                (int) (value1 / 10);
icm1[4].h.bcd = value1 % 10; value1 =
                (int) (value1 / 10);
icm1[2].h.bcd = value1 % 10; value1 =
                (int) (value1 / 10);
icm1[1].h.bcd = value1 % 10; value1 =
                (int) (value1 / 10);
icm1[0].h.bcd = value1 % 10;

// Ec 1st Value
icm2[0].h.asc = 3;
icm2[1].h.asc = 3;
icm2[2].h.asc = 3;
icm2[3].h.asc = 3;
icm2[4].h.asc = 3;

icm2[4].h.bcd = value2 % 10; value2 =
                (int) (value2 / 10);
icm2[3].h.bcd = value2 % 10; value2 =
                (int) (value2 / 10);
icm2[2].h.bcd = value2 % 10; value2 =
                (int) (value2 / 10);
icm2[1].h.bcd = value2 % 10; value2 =
                (int) (value2 / 10);
icm2[0].h.bcd = value2 % 10;

// Ec 2nd Value
```

```
icm3[0].h.asc = 3;
icm3[1].h.asc = 3;
icm3[2].h.asc = 3;
icm3[3].h.asc = 3;
icm3[4].h.asc = 3;

icm3[4].h.bcd = value3 % 10; value3 =
                (int) (value3 / 10);
icm3[3].h.bcd = value3 % 10; value3 =
                (int) (value3 / 10);
icm3[2].h.bcd = value3 % 10; value3 =
                (int) (value3 / 10);
icm3[1].h.bcd = value3 % 10; value3 =
                (int) (value3 / 10);
icm3[0].h.bcd = value3 % 10;

// Ec 3rd Value
icm10[0].h.asc = 3;
icm10[1].h.asc = 3;
icm10[2].h.asc = 3;
icm10[3].h.asc = 3;
icm10[4].h.asc = 3;

icm10[4].h.bcd = value8 % 10; value8 =
                (int) (value8 / 10);
icm10[3].h.bcd = value8 % 10; value8 =
                (int) (value8 / 10);
icm10[2].h.bcd = value8 % 10; value8 =
                (int) (value8 / 10);
icm10[1].h.bcd = value8 % 10; value8 =
                (int) (value8 / 10);
icm10[0].h.bcd = value8 % 10;

// Ec 4th Value
icm11[0].h.asc = 3;
icm11[1].h.asc = 3;
icm11[2].h.asc = 3;
icm11[3].h.asc = 3;
icm11[4].h.asc = 3;

icm11[4].h.bcd = value9 % 10; value9 =
                (int) (value9 / 10);
icm11[3].h.bcd = value9 % 10; value9 =
                (int) (value9 / 10);
icm11[2].h.bcd = value9 % 10; value9 =
                (int) (value9 / 10);
icm11[1].h.bcd = value9 % 10; value9 =
                (int) (value9 / 10);
icm11[0].h.bcd = value9 % 10;
```

```
// Temp Low Value
if(value4 < 0) icm4[0].x.word = '-';
else icm4[0].x.word = '+';
icm4[1].h.asc = 3;
icm4[2].h.asc = 3;
icm4[3].x.word = '.';
icm4[4].h.asc = 3;

icm4[4].h.bcd = (abs(value4)) % 10;
          value4 = (int) (value4 / 10);
icm4[2].h.bcd = (abs(value4)) % 10;
          value4 = (int) (value4 / 10);
icm4[1].h.bcd = (abs(value4)) % 10;

// Temp High Value
if(value5 < 0) icm5[0].x.word = '-';
else icm5[0].x.word = '+';
icm5[1].h.asc = 3;
icm5[2].h.asc = 3;
icm5[3].x.word = '.';
icm5[4].h.asc = 3;

icm5[4].h.bcd = (abs(value5)) % 10;
          value5 = (int) (value5 / 10);
icm5[2].h.bcd = (abs(value5)) % 10;
          value5 = (int) (value5 / 10);
icm5[1].h.bcd = (abs(value5)) % 10;

icm9[0].h.asc = 3;
icm9[1].h.asc = 3;
icm9[2].x.word = ':';
icm9[3].h.asc = 3;
icm9[4].h.asc = 3;

value7 = (int) (value6 / 60);
icm9[1].h.bcd = value7 % 10;
icm9[0].h.bcd = (int) (value7 / 10);
value7 = value6 % 60;
icm9[4].h.bcd = value7 % 10;
icm9[3].h.bcd = (int) (value7 / 10);

// Data 저장 시간 간격
Second[2].x.word =
   Read_DS1305(R_SEC_ALARM); //
        초 Alarm reading
Minute[2].x.word =
   Read_DS1305(R_MIN_ALARM); //
        분 Alarm reading
Hour[2].x.word   =
   Read_DS1305(R_HOUR_ALARM); //
```

시간 Alarm reading

```
icm7[0].h.asc   = 3;
icm7[1].h.asc   = 3;
icm7[2].x.word  = 'H';
icm7[3].x.word  = ' ';
icm7[4].x.word  = ':';
icm7[5].x.word  = ' ';
icm7[6].h.asc   = 3;
icm7[7].h.asc   = 3;
icm7[8].x.word  = 'M';
icm7[9].x.word  = ' ';
icm7[10].x.word = ' ';
icm7[11].h.asc  = 3;
icm7[12].h.asc  = 3;
icm7[13].x.word = 'S';

icm7[12].h.bcd = Second[2].h.SEC_1;
icm7[11].h.bcd = Second[2].h.SEC_10;
icm7[7].h.bcd = Minute[2].h.MIN_1;
icm7[6].h.bcd = Minute[2].h.MIN_10;
icm7[1].h.bcd  = Hour[2].h.HR_1;
icm7[0].h.bcd  = Hour[2].h.HR_10;

lcd_printf(0x80, "Calibration Mode    ");
lcd_printf(0xC0, "Depth, Ec, Temp,
          pH ");

key.x.word = getkey();

if(key.h.mode_key == M2) {
     // Mode Key에 의하여 Depth, EC,
        Temp, pH의 순서로 절환
     selv_cal++;
     if(selv_cal > 11) selv_cal = 0;
     if((selv_cal == 1) ||
       (selv_cal == 2)) {
          Analog_SEL1 = LO;
          Analog_SEL2 = LO;
     }
     if((selv_cal == 3) ||
       (selv_cal == 4) ||
       (selv_cal == 5) ||
       (selv_cal == 6)) {
          Analog_SEL1 = HI;
          Analog_SEL2 = LO;
     }
     if((selv_cal == 7) ||
       (selv_cal == 8)) {
          Analog_SEL1 = LO;
          Analog_SEL2 = HI;
     }
}

if((key.h.enter_key == M1) &&
  (key.h.shift_key == M1)){
     Set_mode = 0;
     DS1305_SetValue_Extract();
}

/* ------------------------
   Depth Value 입력하는 Mode
   ------------------------ */
while(selv_cal == 1) {

   Wdt_out(); /* WatchDog 신호 */

   imsi = Get_ADValue();
   AD_value3 = (int) imsi;

   icm66[1].h.asc = 3;
   icm66[2].h.asc = 3;
   icm66[3].h.asc = 3;
   icm66[4].h.asc = 3;
   icm66[5].h.asc = 3;

   icm66[0].x.word = (imsi < 0) ? '-'
                     : '+';
   imsi = labs(imsi);
   icm66[5].h.bcd = imsi % 10; imsi
              = (long) (imsi / 10);
   icm66[4].h.bcd = imsi % 10; imsi
              = (long) (imsi / 10);
   icm66[3].h.bcd = imsi % 10; imsi
              = (long) (imsi / 10);
   icm66[2].h.bcd = imsi % 10; imsi
              = (long) (imsi / 10);
   icm66[1].h.bcd = imsi % 10;

   // Depth Value를 표시하고 입력
      가능한 상태
   key.x.word = getkey();

   if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의
           순서로 절환
        selv_cal++;
        if(selv_cal > 11) selv_cal
           = 0;
```

```
            if((selv_cal == 1) ||
              (selv_cal == 2)) {
                 Analog_SEL1 = LO;
                 Analog_SEL2 = LO;
            }

        if((selv_cal == 3) ||
          (selv_cal == 4) ||
          (selv_cal == 5) ||
          (selv_cal == 6)) {
             Analog_SEL1 = HI;
             Analog_SEL2 = LO;
          }
          if((selv_cal == 7) ||
            (selv_cal == 8)) {
               Analog_SEL1 = LO;
               Analog_SEL2 = HI;
          }
    }

    //                0123456789012345678 9
// lcd_printf(0x80, "Cal. Depth : +00000 ");
   lcd_printf(0x80, "Cal. Depth : ");
// lcd_printf(0xC0, "Low Value : 000.00m ");
   lcd_printf(0xC0, "Low Value : ");
   Lcd_OneCharWrite(0xD2, 'm');

    for(a = 0; a < 6; a++) {
        Lcd_OneCharWrite((0x8D + a),
        icm66[a].x.word);
    }

    // 선택된 자리 깜박임 표시
    if(g < 30) { // Data ON Display
        for(a = 0; a < 6; a++) {
           Lcd_OneCharWrite((0xCC + a),
                          icm0[a].x.word);
        }
    }

    // Data 표시 OFF
    if(g >= 30) Lcd_OneCharWrite((0xCC +
        leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 60) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 5) leedong = 0;
    }

    if(leedong == 0) {
        sang = icm0[0].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
          if(sang > 1) sang = 0;
        }
        icm0[0].h.bcd = sang;
    }

    if(leedong == 1) {
        sang = icm0[1].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm0[1].h.bcd = sang;
    }

    if(leedong == 2) {
        sang = icm0[2].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm0[2].h.bcd = sang;
    }

    if(leedong == 3) leedong = 4;

    if(leedong == 4) {
        sang = icm0[4].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm0[4].h.bcd = sang;
    }

    if(leedong == 5) {
        sang = icm0[5].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm0[5].h.bcd = sang;
    }

    if(key.h.enter_key == M2) {
```

```
            temp = 0;
            temp = temp + (int) icm0[5].h.bcd
                   * CDS1;
            temp = temp + (int) icm0[4].h.bcd
                   * CDS2;
            temp = temp + (int) icm0[2].h.bcd
                   * CDS3;
            temp = temp + (int) icm0[1].h.bcd
                   * CDS4;
            temp = temp + (int) icm0[0].h.bcd
                   * CDS5;
            CalMode[0].x.word = temp;
            CalMode[6].x.word = AD_value3;

            Ctrl_RTC.h.WP = LO;
            Write_DS1305(W_CONTROL
                         _ADDR,
                         Ctrl_RTC.x.word);
                         // Write Permit
            Write_DS1305(W_PRES11z
                         _ADDR,
                    CalMode[0].h.LowByte);
            Write_DS1305(W_PRES12z
                         _ADDR,
                    CalMode[0].h.HighByte);
            Write_DS1305(W_Depth11z
                         _ADDR,
                    CalMode[6].h.LowByte);
            Write_DS1305(W_Depth12z
                         _ADDR,
                    CalMode[6].h.HighByte);
            Ctrl_RTC.h.WP = HI;
            Write_DS1305(W_CONTROL
                         _ADDR,
                         Ctrl_RTC.x.word);
                         // Write Protect

            //           01234567890123456789
            lcd_printf(0xC0, " --- Save ---");
            delay(30000);
            delay(30000);
            delay(30000);
        }
    } // while(selv_cal == 1) loop ends

    while(selv_cal == 2) {

        Wdt_out(); /* WatchDog 신호 */

        imsi = Get_ADValue();
        AD_value3 = (int) imsi;

        icm66[1].h.asc = 3;
        icm66[2].h.asc = 3;
        icm66[3].h.asc = 3;
        icm66[4].h.asc = 3;
        icm66[5].h.asc = 3;
        icm66[0].x.word = (imsi < 0) ? '-' :
                          '+';
        imsi = labs(imsi);
        icm66[5].h.bcd = imsi % 10; imsi =
                         (long) (imsi / 10);
        icm66[4].h.bcd = imsi % 10; imsi =
                         (long) (imsi / 10);
        icm66[3].h.bcd = imsi % 10; imsi =
                         (long) (imsi / 10);
        icm66[2].h.bcd = imsi % 10; imsi =
                         (long) (imsi / 10);
        icm66[1].h.bcd = imsi % 10;

        // Depth Value를 표시하고 입력 가능한
          상태
        key.x.word = getkey();

        if(key.h.mode_key == M2) {
            // Mode Key에 의하여 Depth
               Value, EC, Temp, pH의
               순서로 절환
            selv_cal++;
            if(selv_cal > 11) selv_cal = 0;
            if((selv_cal == 1) ||
              (selv_cal == 2)) {
                 Analog_SEL1 = LO;
                 Analog_SEL2 = LO;
            }
            if((selv_cal == 3) ||
              (selv_cal == 4) ||
              (selv_cal == 5) ||
              (selv_cal == 6)) {
                 Analog_SEL1 = HI;
                 Analog_SEL2 = LO;
            }
            if((selv_cal == 7) ||
              (selv_cal == 8)) {
                 Analog_SEL1 = LO;
                 Analog_SEL2 = HI;
            }
        }
```

```
    //                0123456789012345678
    lcd_printf(0x80, "Cal. Depth : ");
// lcd_printf(0xC0, "High Value :000.00m");
    lcd_printf(0xC0, "High Value : ");
    Lcd_OneCharWrite(0xD3, 'm');

    for(a = 0; a < 6; a++) {
        Lcd_OneCharWrite((0x8D + a),
                    icm66[a].x.word);
    }

    // 선택된 자리 깜박임 표시
    if(g < 30) { // Data ON Display
        for(a = 0; a < 6; a++) {
            Lcd_OneCharWrite((0xCD +a)
                    , icm1[a].x.word);
        }
    }

    // Data 표시 OFF
    if(g >= 30) Lcd_OneCharWrite((0xCD +
        leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 60) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 5) leedong = 0;
    }

    if(leedong == 0) {
        sang = icm1[0].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 1) sang = 0;
        }
        icm1[0].h.bcd = sang;
    }

    if(leedong == 1) {
        sang = icm1[1].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm1[1].h.bcd = sang;
    }

    if(leedong == 2) {
        sang = icm1[2].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm1[2].h.bcd = sang;
    }

    if(leedong == 3) leedong = 4;

    if(leedong == 4) {
        sang = icm1[4].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm1[4].h.bcd = sang;
    }

    if(leedong == 5) {
        sang = icm1[5].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm1[5].h.bcd = sang;
    }

    if(key.h.enter_key == M2) {

        temp = 0;
        temp = temp + (int) icm1[5].h.bcd
                * CDS1;
        temp = temp + (int) icm1[4].h.bcd
                * CDS2;
        temp = temp + (int) icm1[2].h.bcd
                * CDS3;
        temp = temp + (int) icm1[1].h.bcd
                * CDS4;
        temp = temp + (int) icm1[0].h.bcd
                * CDS5;
        CalMode[1].x.word = temp;

        CalMode[7].x.word = AD_value3;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
                ADDR, Ctrl_RTC.x.word);
                // Write Permit
        Write_DS1305(W_PRES21s_ADDR,
```

```
                    CalMode[1].h.LowByte);
          Write_DS1305(W_PRES22s_
             ADDR, CalMode[1].h.HighByte);
          Write_DS1305(W_Depth21s_
             ADDR, CalMode[7].h.LowByte);

          Write_DS1305(W_Depth22s_
             ADDR, CalMode[7].h.HighByte);
          Ctrl_RTC.h.WP = HI;
          Write_DS1305(W_CONTROL_
             ADDR, Ctrl_RTC.x.word);
             // Write Protect

          //          01234567890123456789
          lcd_printf(0xC0, "--- Save ---");
          delay(30000);
          delay(30000);
          delay(30000);
     }
} // while(selv_cal == 2) loop ends

/* -------------------------
    EC Value 입력하는 Mode
   ------------------------- */
while(selv_cal == 3) {

     Wdt_out(); /* WatchDog 신호 */

     imsi = Get_ADValue();
     AD_value3 = (int) imsi;

     icm66[1].h.asc = 3;
     icm66[2].h.asc = 3;
     icm66[3].h.asc = 3;
     icm66[4].h.asc = 3;
     icm66[5].h.asc = 3;

     icm66[0].x.word = (imsi < 0) ? '-' :
                       '+';
     imsi = labs(imsi);
     icm66[5].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[4].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[3].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[2].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[1].h.bcd = imsi % 10;

     // Ec Value를 표시하고 입력 가능한
       상태
     key.x.word = getkey();

     if(key.h.mode_key == M2) {
          // Mode Key에 의하여 Depth
             Value, EC, Temp, pH의 순서로
             절환
          selv_cal++;
          if(selv_cal > 11) selv_cal = 0;
          if((selv_cal == 1) ||
            (selv_cal == 2)) {
               Analog_SEL1 = LO;
               Analog_SEL2 = LO;
          }
          if((selv_cal == 3) ||
            (selv_cal == 4) ||
            (selv_cal == 5) ||
            (selv_cal == 6)) {
               Analog_SEL1 = HI;
               Analog_SEL2 = LO;
          }
          if((selv_cal == 7) ||
            (selv_cal == 8)) {
               Analog_SEL1 = LO;
               Analog_SEL2 = HI;
          }
     }

          //               01234567890123456789
          lcd_printf(0x80, "Cal. Ec :     ");
//        lcd_printf(0xC0, "Low : 00000 uS/cm ");
          lcd_printf(0xC0, "1st :    ");
//        Lcd_OneCharWrite(0xCA, '.');
          lcd_printf(0xCD, " uS/cm ");

          for(a = 0; a < 6; a++) {
               Lcd_OneCharWrite((0x8D + a),
                           icm66[a].x.word);
          }

          // 선택된 자리 깜박임 표시
          if(g < 30) { // Data ON Display
               for(a = 0; a < 5; a++) {
```

```
        Lcd_OneCharWrite((0xC8 + a),
                    icm2[a].x.word);
    }
}


// Data 표시 OFF
if(g >= 30) Lcd_OneCharWrite((0xC8 +
    leedong), 0x20); // 공백 문자 출력

g = g + 1;
if(g > 60) g = 0;

if(key.h.shift_key == M2) {
    leedong++;
    if(leedong > 4) leedong = 0;
}

if(leedong == 0) {
    sang = icm2[0].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 3) sang = 0;
    }
    icm2[0].h.bcd = sang;
}

if(leedong == 1) {
    sang = icm2[1].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm2[1].h.bcd = sang;
}

if(leedong == 2) {
    sang = icm2[2].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm2[2].h.bcd = sang;
}

if(leedong == 3) {
    sang = icm2[3].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm2[3].h.bcd = sang;
}


if(leedong == 4) {
    sang = icm2[4].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm2[4].h.bcd = sang;
}

if(key.h.enter_key == M2) {
    temp = 0;
    temp = temp + (int) icm2[4].h.bcd
            * CDS1;
    temp = temp + (int) icm2[3].h.bcd
            * CDS2;
    temp = temp + (int) icm2[2].h.bcd
            * CDS3;
    temp = temp + (int) icm2[1].h.bcd
            * CDS4;
    temp = temp + (int) icm2[0].h.bcd
            * CDS5;
    CalMode[2].x.word = temp;

    CalMode[8].x.word = AD_value3;

    Ctrl_RTC.h.WP = LO;
    Write_DS1305(W_CONTROL_
      ADDR, Ctrl_RTC.x.word);
          // Write Permit
    Write_DS1305(W_CON11z_
      ADDR, CalMode[2].h.LowByte);
    Write_DS1305(W_CON12z_
      ADDR,  CalMode[2].h.HighByte);
    Write_DS1305(W_EC11z_
      ADDR, CalMode[8].h.LowByte);
    Write_DS1305(W_EC12z_ADDR,
CalMode[8].h.HighByte);
    Ctrl_RTC.h.WP = HI;
    Write_DS1305(W_CONTROL_
      ADDR, Ctrl_RTC.x.word);
          // Write Protect

    //          0123456789012345678 9
    lcd_printf(0xC0, "--- Save --- ");
```

```
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 3) loop ends

while(selv_cal == 4) {

    Wdt_out(); /* WatchDog 신호 */

    imsi = Get_ADValue();
    AD_value3 = (int) imsi;

    icm66[1].h.asc = 3;
    icm66[2].h.asc = 3;
    icm66[3].h.asc = 3;
    icm66[4].h.asc = 3;
    icm66[5].h.asc = 3;

    icm66[0].x.word = (imsi < 0) ? '-' :
                      '+';
    imsi = labs(imsi);
    icm66[5].h.bcd = imsi % 10; imsi
                   = (long) (imsi / 10);
    icm66[4].h.bcd = imsi % 10; imsi
                   = (long) (imsi / 10);
    icm66[3].h.bcd = imsi % 10; imsi
                   = (long) (imsi / 10);
    icm66[2].h.bcd = imsi % 10; imsi
                   = (long) (imsi / 10);
    icm66[1].h.bcd = imsi % 10;

    // Ec Value를 표시하고 입력 가능한
      상태
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의 순서로
           절환
        selv_cal++;
        if(selv_cal > 11) selv_cal = 0;
        if((selv_cal == 1) ||
          (selv_cal == 2)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 3) ||
          (selv_cal == 4) ||
          (selv_cal == 5) ||
          (selv_cal == 6)) {
            Analog_SEL1 = HI;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 7) ||
          (selv_cal == 8)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = HI;
        }
    }

    //              01234567890123456789
    lcd_printf(0x80, "Cal. Ec :    ");
// lcd_printf(0xC0, "High : 00000 uS/cm");
    lcd_printf(0xC0, "2nd  :  ");
// Lcd_OneCharWrite(0xCA, '.');
    lcd_printf(0xCD, " uS/cm ");

    for(a = 0; a < 6; a++) {
        Lcd_OneCharWrite((0x8D + a),
                  icm66[a].x.word);
    }

    // 선택된 자리 깜박임 표시
    if(g < 30) { // Data ON Display
        for(a = 0; a < 5; a++) {
            Lcd_OneCharWrite((0xC8 + a),
                  icm3[a].x.word);
        }
    }

    // Data 표시 OFF
    if(g >= 30) Lcd_OneCharWrite((0xC8
      + leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 60) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 4) leedong = 0;
    }

    if(leedong == 0) {
        sang = icm3[0].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 3) sang = 0;
        }
        icm3[0].h.bcd = sang;
```

```
}

if(leedong == 1) {
     sang = icm3[1].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm3[1].h.bcd = sang;
}

if(leedong == 2) {
     sang = icm3[2].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm3[2].h.bcd = sang;
}

if(leedong == 3) {
     sang = icm3[3].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm3[3].h.bcd = sang;
}

if(leedong == 4) {
     sang = icm3[4].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm3[4].h.bcd = sang;
}

if(key.h.enter_key == M2) {

     temp = 0;
     temp = temp + (int) icm3[4].h.bcd
            * CDS1;
     temp = temp + (int) icm3[3].h.bcd
            * CDS2;
     temp = temp + (int) icm3[2].h.bcd
            * CDS3;
     temp = temp + (int) icm3[1].h.bcd
            * CDS4;
     temp = temp + (int) icm3[0].h.bcd
            * CDS5;
     CalMode[3].x.word = temp;

     CalMode[9].x.word = AD_value3;

     Ctrl_RTC.h.WP = LO;
     Write_DS1305(W_CONTROL_
         ADDR, Ctrl_RTC.x.word);
         // Write Permit
     Write_DS1305(W_CON21s_ADDR,
         CalMode[3].h.LowByte);
     Write_DS1305(W_CON22s_ADDR,
         CalMode[3].h.HighByte);
     Write_DS1305(W_EC21s_ADDR,
   CalMode[9].h.LowByte);
     Write_DS1305(W_EC22s_ADDR,
         CalMode[9].h.HighByte);
     Ctrl_RTC.h.WP = HI;
     Write_DS1305(W_CONTROL_
          ADDR, Ctrl_RTC.x.word);
          // Write Protect

     //          0123456789012345678 9
     lcd_printf(0xC0, "--- Save ---");
     delay(30000);
     delay(30000);
     delay(30000);
   }
} // while(selv_cal == 4) loop ends
```

```
/* ------------------------
    EC Value 입력하는 Mode
   ------------------------ */
while(selv_cal == 5) {

     Wdt_out(); /* WatchDog 신호 */

     imsi = Get_ADValue();
     AD_value3 = (int) imsi;

     icm66[1].h.asc = 3;
     icm66[2].h.asc = 3;
     icm66[3].h.asc = 3;
     icm66[4].h.asc = 3;
     icm66[5].h.asc = 3;

     icm66[0].x.word = (imsi < 0) ? '-' :
                       '+';
     imsi = labs(imsi);
     icm66[5].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[4].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[3].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[2].h.bcd = imsi % 10; imsi
                    = (long) (imsi / 10);
     icm66[1].h.bcd = imsi % 10;

     // Ec Value를 표시하고 입력 가능한
       상태
     key.x.word = getkey();

     if(key.h.mode_key == M2) {
          // Mode Key에 의하여 Depth
             Value, EC, Temp, pH의 순서로
             절환
          selv_cal++;
          if(selv_cal > 11) selv_cal = 0;
          if((selv_cal == 1) ||
            (selv_cal == 2)) {
               Analog_SEL1 = LO;
               Analog_SEL2 = LO;
          }
          if((selv_cal == 3) ||
            (selv_cal == 4) ||
            (selv_cal == 5) ||
            (selv_cal == 6)) {
               Analog_SEL1 = HI;
               Analog_SEL2 = LO;
          }
          if((selv_cal == 7) ||
            (selv_cal == 8)) {
               Analog_SEL1 = LO;
               Analog_SEL2 = HI;
          }
     }

     //                0123456789012345678 9
     lcd_printf(0x80, "Cal. Ec :    ");
//   lcd_printf(0xC0, "Low : 00000 uS/cm ");
     lcd_printf(0xC0, "3rd :    ");
//   Lcd_OneCharWrite(0xCA, '.');
     lcd_printf(0xCD, " uS/cm ");

     for(a = 0; a < 6; a++) {
          Lcd_OneCharWrite((0x8D + a),
                    icm66[a].x.word);
     }

     // 선택된 자리 깜박임 표시
     if(g < 30) { // Data ON Display
          for(a = 0; a < 5; a++) {
               Lcd_OneCharWrite((0xC8 + a),
                         icm10[a].x.word);
          }
     }

     // Data 표시 OFF
     if(g >= 30) Lcd_OneCharWrite((0xC8 +
          leedong), 0x20); // 공백 문자 출력

     g = g + 1;
     if(g > 60) g = 0;

     if(key.h.shift_key == M2) {
          leedong++;
          if(leedong > 4) leedong = 0;
     }

     if(leedong == 0) {
          sang = icm10[0].h.bcd;
          if(key.h.up_key == M2) {
               sang++;
```

```
            if(sang > 3) sang = 0;
        }
        icm10[0].h.bcd = sang;
    }


    if(leedong == 1) {
        sang = icm10[1].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm10[1].h.bcd = sang;
    }

    if(leedong == 2) {
        sang = icm10[2].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm10[2].h.bcd = sang;
    }

    if(leedong == 3) {
        sang = icm10[3].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm10[3].h.bcd = sang;
    }

    if(leedong == 4) {
        sang = icm10[4].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm10[4].h.bcd = sang;
    }

    if(key.h.enter_key == M2) {
        temp = 0;
        temp = temp + (int) icm10[4].h.bcd
                * CDS1;
        temp = temp + (int) icm10[3].h.bcd
                * CDS2;
        temp = temp + (int) icm10[2].h.bcd
                * CDS3;
        temp = temp + (int) icm10[1].h.bcd
                * CDS4;
        temp = temp + (int) icm10[0].h.bcd
                * CDS5;
        CalMode[13].x.word = temp;

        CalMode[15].x.word = AD_value3;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
              ADDR, Ctrl_RTC.x.word);
              // Write Permit
        Write_DS1305(W_CON31z_ADDR,
              CalMode[13].h.LowByte);
        Write_DS1305(W_CON32z_ADDR,
              CalMode[13].h.HighByte);
        Write_DS1305(W_EC31z_ADDR,
              CalMode[15].h.LowByte);
        Write_DS1305(W_EC32z_ADDR,
              CalMode[15].h.HighByte);
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
              ADDR, Ctrl_RTC.x.word);
              // Write Protect

        //          01234567890123456789
        lcd_printf(0xC0, " --- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 5) loop ends

while(selv_cal == 6) {

    Wdt_out(); /* WatchDog 신호 */

    imsi = Get_ADValue();
    AD_value3 = (int) imsi;

    icm66[1].h.asc = 3;
    icm66[2].h.asc = 3;
    icm66[3].h.asc = 3;
    icm66[4].h.asc = 3;
    icm66[5].h.asc = 3;

    icm66[0].x.word = (imsi < 0) ? '-' :
                      '+';
    imsi = labs(imsi);
    icm66[5].h.bcd = imsi % 10; imsi
```

```
                = (long) (imsi / 10);
    icm66[4].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[3].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);

    icm66[2].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[1].h.bcd = imsi % 10;

    // Ec Value를 표시하고 입력 가능한
      상태
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
         // Mode Key에 의하여 Depth
            Value, EC, Temp, pH의 순서로
            절환
         selv_cal++;
         if(selv_cal > 11) selv_cal = 0;
         if((selv_cal == 1) ||
           (selv_cal == 2)) {
              Analog_SEL1 = LO;
              Analog_SEL2 = LO;
         }
         if((selv_cal == 3) ||
           (selv_cal == 4) ||
           (selv_cal == 5) ||
           (selv_cal == 6)) {
              Analog_SEL1 = HI;
              Analog_SEL2 = LO;
         }
         if((selv_cal == 7) ||
           (selv_cal == 8)) {
              Analog_SEL1 = LO;
              Analog_SEL2 = HI;
         }
    }

    //              01234567890123456789
    lcd_printf(0x80, "Cal. Ec :    ");
//  lcd_printf(0xC0, "High :  00000 uS/cm
            ");
    lcd_printf(0xC0, "4th  :  ");
//  Lcd_OneCharWrite(0xCA, '.');
    lcd_printf(0xCD, " uS/cm ");

    for(a = 0; a < 6; a++) {
         Lcd_OneCharWrite((0x8D + a),
                   icm66[a].x.word);
    }

    // 선택된 자리 깜박임 표시
    if(g < 30) { // Data ON Display
         for(a = 0; a < 5; a++) {
              Lcd_OneCharWrite((0xC8 +a),
                        icm11[a].x.word);
         }
    }

    // Data 표시 OFF
    if(g >= 30) Lcd_OneCharWrite((0xC8
       + leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 60) g = 0;

    if(key.h.shift_key == M2) {
         leedong++;
         if(leedong > 4) leedong = 0;
    }

    if(leedong == 0) {
         sang = icm11[0].h.bcd;
         if(key.h.up_key == M2) {
              sang++;
              if(sang > 3) sang = 0;
         }
         icm11[0].h.bcd = sang;
    }

    if(leedong == 1) {
         sang = icm11[1].h.bcd;
         if(key.h.up_key == M2) {
              sang++;
              if(sang > 9) sang = 0;
         }
         icm11[1].h.bcd = sang;
    }

    if(leedong == 2) {
         sang = icm11[2].h.bcd;
         if(key.h.up_key == M2) {
              sang++;
              if(sang > 9) sang = 0;
         }
```

```
        icm11[2].h.bcd = sang;
}



if(leedong == 3) {
        sang = icm11[3].h.bcd;
        if(key.h.up_key == M2) {
                sang++;
                if(sang > 9) sang = 0;
        }
        icm11[3].h.bcd = sang;
}

if(leedong == 4) {
        sang = icm11[4].h.bcd;
        if(key.h.up_key == M2) {
             sang++;
             if(sang > 9) sang = 0;
        }
        icm11[4].h.bcd = sang;
}

if(key.h.enter_key == M2) {

        temp = 0;
        temp = temp + (int) icm11[4].h.bcd
                * CDS1;
        temp = temp + (int) icm11[3].h.bcd
                * CDS2;
        temp = temp + (int) icm11[2].h.bcd
                * CDS3;
        temp = temp + (int) icm11[1].h.bcd
                * CDS4;
        temp = temp + (int) icm11[0].h.bcd
                * CDS5;
        CalMode[14].x.word = temp;

        CalMode[16].x.word = AD_value3;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
          ADDR, Ctrl_RTC.x.word);
          // Write Permit
        Write_DS1305(W_CON41s_
          ADDR, CalMode[14].h.LowByte);
        Write_DS1305(W_CON42s_
          ADDR, CalMode[14].h.HighByte);
        Write_DS1305(W_EC41s_ADDR,
          CalMode[16].h.LowByte);
        Write_DS1305(W_EC42s_ADDR,
          CalMode[16].h.HighByte);
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
         ADDR, Ctrl_RTC.x.word);
         // Write Protect

        //          0123456789012345678
        lcd_printf(0xC0, " --- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 6) loop ends

while(selv_cal == 7) {

    Wdt_out(); /* WatchDog 신호 */

    imsi = Get_ADValue();
    AD_value3 = (int) imsi;

    icm66[1].h.asc = 3;
    icm66[2].h.asc = 3;
    icm66[3].h.asc = 3;
    icm66[4].h.asc = 3;
    icm66[5].h.asc = 3;

    icm66[0].x.word = (imsi < 0) ? '-' :
                      '+';
    imsi = labs(imsi);
    icm66[5].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[4].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[3].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[2].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[1].h.bcd = imsi % 10;

    // Temp Value를 표시하고 입력 가능한
       상태
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의 순서로
           절환
```

```
    selv_cal++;
    if(selv_cal > 11) selv_cal = 0;
    if((selv_cal == 1) ||
      (selv_cal == 2)) {
        Analog_SEL1 = LO;
        Analog_SEL2 = LO;
    }
    if((selv_cal == 3) ||
      (selv_cal == 4) ||
      (selv_cal == 5) ||
      (selv_cal == 6)) {
        Analog_SEL1 = HI;
        Analog_SEL2 = LO;
    }
    if((selv_cal == 7) ||
      (selv_cal == 8)) {
        Analog_SEL1 = LO;
        Analog_SEL2 = HI;
    }
}

//                0123456789012345678
lcd_printf(0x80, "Cal. Temp :  ");
lcd_printf(0xC0, "Low Value :  ");
// lcd_printf(0xC0, "Low Value :
          -00.0?");
Lcd_OneCharWrite(0xD2, 0xDF);
Lcd_OneCharWrite(0xD3, 'C');

for(a = 0; a < 6; a++) {
    Lcd_OneCharWrite((0x8D + a),
              icm66[a].x.word);
}

// 선택된 자리 깜박임 표시
if(g < 30) { // Data ON Display
    for(a = 0; a < 5; a++) {
        Lcd_OneCharWrite((0xCD +a),
                  icm4[a].x.word);
    }
}

// Data 표시 OFF
if(g >= 30) Lcd_OneCharWrite((0xCD +
    leedong), 0x20); // 공백 문자 출력

g = g + 1;
if(g > 60) g = 0;

if(key.h.shift_key == M2) {
    leedong++;
    if(leedong > 4) leedong = 0;
}

if(leedong == 0) {
    if(key.h.up_key == M2) {
        if(icm4[0].x.word == '-')
            icm4[0].x.word = '+';
        else icm4[0].x.word = '-';
    }
}

if(leedong == 1) {
    sang = icm4[1].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm4[1].h.bcd = sang;
}

if(leedong == 2) {
    sang = icm4[2].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm4[2].h.bcd = sang;
}

if(leedong == 3) leedong = 4;

if(leedong == 4) {
    sang = icm4[4].h.bcd;
    if(key.h.up_key == M2) {
        sang++;
        if(sang > 9) sang = 0;
    }
    icm4[4].h.bcd = sang;
}

if(key.h.enter_key == M2) {

    temp = 0;
    temp = temp + (int) icm4[4].h.bcd
           * CDS1;
    temp = temp + (int) icm4[2].h.bcd
           * CDS2;
    temp = temp + (int) icm4[1].h.bcd
           * CDS3;
```

```
        if(icm4[0].x.word == '-') temp
                        = temp * (-1);

        CalMode[4].x.word = temp;

        CalMode[10].x.word = AD_value3;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
            // Write Permit
        Write_DS1305(W_TEMP11z_
           ADDR, CalMode[4].h.LowByte);
        Write_DS1305(W_TEMP12z_
           ADDR, CalMode[4].h.HighByte);
        Write_DS1305(W_Temp31z_ADDR,
     CalMode[10].h.LowByte);
        Write_DS1305(W_Temp32z_ADDR,
           CalMode[10].h.HighByte);
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
           // Write Protect

        //          01234567890123456789
        lcd_printf(0xC0, "--- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 7) loop ends

while(selv_cal == 8) {

    Wdt_out(); /* WatchDog 신호 */

    imsi = Get_ADValue();
    AD_value3 = (int) imsi;

    icm66[1].h.asc = 3;
    icm66[2].h.asc = 3;
    icm66[3].h.asc = 3;
    icm66[4].h.asc = 3;
    icm66[5].h.asc = 3;

    icm66[0].x.word = (imsi < 0) ? '-' :
                      '+';
    imsi = labs(imsi);
    icm66[5].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[4].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[3].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[2].h.bcd = imsi % 10; imsi
                = (long) (imsi / 10);
    icm66[1].h.bcd = imsi % 10;

    // Temp Value를 표시하고 입력 가능한
       상태
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의 순서로
           절환
        selv_cal++;
        if(selv_cal > 11) selv_cal = 0;
        if((selv_cal == 1) ||
          (selv_cal == 2)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 3) ||
          (selv_cal == 4) ||
          (selv_cal == 5) ||
          (selv_cal == 6)) {
            Analog_SEL1 = HI;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 7) ||
         (selv_cal == 8)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = HI;
        }
    }
     //              01234567890123456789
    lcd_printf(0x80, "Cal. Temp :  ");
    lcd_printf(0xC0, "High Value : ");
    Lcd_OneCharWrite(0xD2, 0xDF);
    Lcd_OneCharWrite(0xD3, 'C');

    for(a = 0; a < 6; a++) {
        Lcd_OneCharWrite((0x8D + a),
                    icm66[a].x.word);
    }

    // 선택된 자리 깜박임 표시
    if(g < 30) { // Data ON Display
        for(a = 0; a < 5; a++) {
```

```
            Lcd_OneCharWrite((0xCD + a),
                        icm5[a].x.word);
        }
    }


    // Data 표시 OFF
    if(g >= 30) Lcd_OneCharWrite((0xCD +
        leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 60) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 4) leedong = 0;
    }

    if(leedong == 0) {
        if(key.h.up_key == M2) {
            if(icm5[0].x.word == '-')
                icm5[0].x.word = '+';
            else icm5[0].x.word = '-';
        }
    }

    if(leedong == 1) {
        sang = icm5[1].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm5[1].h.bcd = sang;
    }

    if(leedong == 2) {
        sang = icm5[2].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm5[2].h.bcd = sang;
    }

    if(leedong == 3) leedong = 4;

    if(leedong == 4) {
        sang = icm5[4].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm5[4].h.bcd = sang;
    }

    if(key.h.enter_key == M2) {

        temp = 0;
        temp = temp + (int) icm5[4].h.bcd
                * CDS1;
        temp = temp + (int) icm5[2].h.bcd
                * CDS2;
        temp = temp + (int) icm5[1].h.bcd
                * CDS3;
        if(icm5[0].x.word == '-') temp
                        = temp * (-1);

        CalMode[5].x.word = temp;

        CalMode[11].x.word = AD_value3;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
            // Write Permit
        Write_DS1305(W_TEMP21s_
           ADDR, CalMode[5].h.LowByte);
        Write_DS1305(W_TEMP22s_
           ADDR, CalMode[5].h.HighByte);
        Write_DS1305(W_Temp41s_ADDR,
     CalMode[11].h.LowByte);
        Write_DS1305(W_Temp42s_ADDR,
           CalMode[11].h.HighByte);
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
           // Write Protect

        //          01234567890123456789
        lcd_printf(0xC0, " --- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 8) loop ends
```

```
// 저장 시간 간격 조정
while(selv_cal == 9) {

    Wdt_out(); /* WatchDog 신호 */
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의 순서로
           절환
        selv_cal++;
        if(selv_cal > 11) selv_cal = 0;
        if((selv_cal == 1) ||
          (selv_cal == 2)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 3) ||
          (selv_cal == 4) ||
          (selv_cal == 5) ||
          (selv_cal == 6)) {
            Analog_SEL1 = HI;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 7) ||
          (selv_cal == 8)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = HI;
        }
    }

    //              01234567890123456789
    lcd_printf(0x80, "Save Time of Data ");
//  lcd_printf(0xC0, "00H : 00M 00S  ");
    lcd_printf(0xCE, "      ");
    // 선택된 자리 깜박임 표시
    if(g < 60) { // Data ON Display
        for(a = 0; a < 14; a++) {
            Lcd_OneCharWrite((0xC0+ a),
                        icm7[a].x.word);
        }
    }

    // Data 표시 OFF
    if(g >= 60) Lcd_OneCharWrite((0xC0 +
        leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 120) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 12) leedong = 0;
    }

    if(leedong == 0) {
        sang = icm7[0].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 2) sang = 0;
        }
        icm7[0].h.bcd = sang;
    }

    if(leedong == 1) {
        sang = icm7[1].h.bcd;
        if(icm7[0].h.bcd == 2) {
            if(key.h.up_key == M2) {
                sang++;
                if(sang > 3) sang = 0;
            }
        }
        else {
            if(key.h.up_key == M2) {
                sang++;
                if(sang > 9) sang = 0;
            }
        }
        icm7[1].h.bcd = sang;
    }

    if((leedong > 1) && (leedong < 6))
                leedong = 6;

    if(leedong == 6) {
        sang = icm7[6].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 5) sang = 0;
        }
        icm7[6].h.bcd = sang;
    }

    if(leedong == 7) {
```

```
        sang = icm7[7].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm7[7].h.bcd = sang;
    }

    if((leedong > 7) && (leedong < 11))
                leedong = 11;

    if(leedong == 11) {
        sang = icm7[11].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 5) sang = 0;
        }
        icm7[11].h.bcd = sang;
    }

    if(leedong == 12) {
        sang = icm7[12].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm7[12].h.bcd = sang;
    }

    if((icm7[0].h.bcd == 0) &&
      (icm7[1].h.bcd == 0)) {
        if((icm7[6].h.bcd == 0) &&
          (icm7[7].h.bcd == 0)) {
            if((icm7[11].h.bcd == 0) &&
              (icm7[12].h.bcd == 0)) {
                Second[2].h.M = HI;
                Minute[2].h.M = HI;
                Hour[2].h.M = HI;
            }
            else {
                Second[2].h.M = LO;
                Minute[2].h.M = HI;
                Hour[2].h.M = HI;
            }
        }
        else {
            Second[2].h.M = LO;
            Minute[2].h.M = LO;
            Hour[2].h.M = HI;
        }
    }
    else {
        Second[2].h.M = LO;
        Minute[2].h.M = LO;
        Hour[2].h.M = LO;
    }

    if(key.h.enter_key == M2) {
        Second[2].h.SEC_1 =icm7[12].h.bcd;
        Second[2].h.SEC_10=icm7[11].h.bcd;
        Minute[2].h.MIN_1 = icm7[7].h.bcd;
        Minute[2].h.MIN_10 =icm7[6].h.bcd;
        Hour[2].h.HR12_24   = LO;
        Hour[2].h.HR_1 = icm7[1].h.bcd;
        Hour[2].h.HR_10  = icm7[0].h.bcd;

        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
            ADDR, Ctrl_RTC.x.word);
            // Write Permit
        Write_DS1305(W_SEC_ALARM,
           Second[2].x.word); // sec write
        Write_DS1305(W_MIN_ALARM,
           Minute[2].x.word); // min write
        Write_DS1305(W_HOUR_ALARM,
           Hour[2].x.word);    // hr write
        Write_DS1305(W_DAY_ALAM,
            0x80);    // date write
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
            ADDR, Ctrl_RTC.x.word);
            // Write Protect

        //          01234567890123456789
        lcd_printf(0xC0, "--- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
    }
} // while(selv_cal == 9) loop ends
```

```
// 전원 On 상태의 저장 시간 간격 조정
while(selv_cal == 10) {

    Wdt_out(); /* WatchDog 신호 */
    key.x.word = getkey();

    if(key.h.mode_key == M2) {
        // Mode Key에 의하여 Depth
           Value, EC, Temp, pH의 순서로
           절환
        selv_cal++;
        if(selv_cal > 11) selv_cal = 0;
        if((selv_cal == 1) ||
          (selv_cal == 2)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 3) ||
          (selv_cal == 4) ||
          (selv_cal == 5) ||
          (selv_cal == 6)) {
            Analog_SEL1 = HI;
            Analog_SEL2 = LO;
        }
        if((selv_cal == 7) ||
          (selv_cal == 8)) {
            Analog_SEL1 = LO;
            Analog_SEL2 = HI;
        }
    }

    //              0123456789012345678
    lcd_printf(0x80, "SaveTime at Power
           On");
// lcd_printf(0xC0, "        00:00       ");
    lcd_printf(0xC0, "            ");
    Lcd_OneCharWrite(0xCA, ':');
    lcd_printf(0xCD, "        ");

    // 선택된 자리 깜박임 표시
    if(g < 60) { // Data ON Display
        for(a = 0; a < 5; a++) {
            Lcd_OneCharWrite((0xC8+ a),
                    icm9[a].x.word);
        }
    }

    // Data 표시 OFF
    if(g >= 60) Lcd_OneCharWrite((0xC8 +
        leedong), 0x20); // 공백 문자 출력

    g = g + 1;
    if(g > 120) g = 0;

    if(key.h.shift_key == M2) {
        leedong++;
        if(leedong > 4) leedong = 0;
    }

    if(leedong == 0) {
        sang = icm9[0].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 2) sang = 0;
        }
        icm9[0].h.bcd = sang;
    }

    if(leedong == 1) {
        sang = icm9[1].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm9[1].h.bcd = sang;
    }

    if(leedong == 2) leedong = 3;

    if(leedong == 3) {
        sang = icm9[3].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 5) sang = 0;
        }
        icm9[3].h.bcd = sang;
    }

    if(leedong == 4) {
        sang = icm9[4].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
```

```
            icm9[4].h.bcd = sang;
        }



        if(key.h.enter_key == M2) {
            temp = 0;
            temp = temp + (int) icm9[4].h.bcd
                   * CDS1;
            temp = temp + (int) icm9[3].h.bcd
                   * CDS2;
            temp = temp + (int) icm9[1].h.bcd
                   * 60;
            temp = temp + (int) icm9[0].h.bcd
                   * 600;
            CalMode[12].x.word = temp;
            total_min = temp;

            Ctrl_RTC.h.WP = LO;
            Write_DS1305(W_CONTROL_
                ADDR, Ctrl_RTC.x.word);
                // Write Permit
            Write_DS1305(W_STHint_ADDR,
                CalMode[12].h.LowByte);
            Write_DS1305(W_STMint_ADDR,
                CalMode[12].h.HighByte);
            Ctrl_RTC.h.WP = LO;
            Write_DS1305(W_CONTROL_
                ADDR, Ctrl_RTC.x.word);
                // Write Protect

            //          0123456789012345678
            lcd_printf(0xC0, "--- Save --- ");
            delay(30000);
            delay(30000);
            delay(30000);
        }
    } // while(selv_cal == 10) loop ends

    // 날짜 및 시간 조정
    while(selv_cal == 11) {

        Wdt_out(); /* WatchDog 신호 */
        key.x.word = getkey();

        if(key.h.mode_key == M2) {
            // Mode Key에 의하여 Depth
               Value, EC, Temp, pH의 순서로
               절환
            selv_cal++;
            if(selv_cal > 11) selv_cal = 0;
            if((selv_cal == 1) ||
              (selv_cal == 2)) {
                 Analog_SEL1 = LO;
                 Analog_SEL2 = LO;
            }
            if((selv_cal == 3) ||
              (selv_cal == 4) ||
              (selv_cal == 5) ||
              (selv_cal == 6)) {
                 Analog_SEL1 = HI;
                 Analog_SEL2 = LO;
            }
            if((selv_cal == 7) ||
              (selv_cal == 8)) {
                 Analog_SEL1 = LO;
                 Analog_SEL2 = HI;
            }
        }

        //                0123456789012345678
        lcd_printf(0x80, "Setting Time & Date
                 ");
        // lcd_printf(0xC0, "2001/05/18 15:36
                    23s");

        // 선택된 자리 깜박임 표시
        if(g < 60) { // Data ON Display
            for(a = 0; a < 20; a++) {
                Lcd_OneCharWrite((0xC0+ a),
                              icm8[a].x.word);
            }
        }

        // Data 표시 OFF
        if(g >= 60) Lcd_OneCharWrite((0xC0 +
            leedong), 0x20); // 공백 문자 출력

        g = g + 1;
        if(g > 120) g = 0;

        if(key.h.shift_key == M2) {
            leedong++;
            if(leedong > 19) leedong = 2;
        }

        if((leedong == 0) ||
          (leedong == 1)) leedong = 2;
```

```
if(leedong == 2) {
     sang = icm8[2].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm8[2].h.bcd = sang;
}

if(leedong == 3) {
     sang = icm8[3].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 9) sang = 0;
     }
     icm8[3].h.bcd = sang;
}

if(leedong == 4) leedong = 5;

if(leedong == 5) {
     sang = icm8[5].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 1) sang = 0;
     }
     icm8[5].h.bcd = sang;
}

if(leedong == 6) {
     sang = icm8[6].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(icm8[6].h.bcd == 1) {
               if(sang > 2) sang = 0;
          }
          else {
               if(sang > 9) sang = 0;
          }
     }
     icm8[6].h.bcd = sang;
}

if(leedong == 7) leedong = 8;

if(leedong == 8) {
     sang = icm8[8].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 3) sang = 0;
     }
     icm8[8].h.bcd = sang;
}

if(leedong == 9) {
     sang = icm8[9].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(icm8[8].h.bcd == 3) {
               if(sang > 1) sang = 0;
          }
          else {
               if(sang > 9) sang = 0;
          }
     }
     icm8[9].h.bcd = sang;
}

if(leedong == 10) leedong = 11;

if(leedong == 11) {
     sang = icm8[11].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 2) sang = 0;
     }
     icm8[11].h.bcd = sang;
}

if(leedong == 12) {
     sang = icm8[12].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(icm8[12].h.bcd == 2) {
               if(sang > 4) sang = 0;
          }
          else {
               if(sang > 9) sang = 0;
          }
     }
     icm8[12].h.bcd = sang;
}

if(leedong == 13) leedong = 14;

if(leedong == 14) {
     sang = icm8[14].h.bcd;
     if(key.h.up_key == M2) {
          sang++;
          if(sang > 5) sang = 0;
```

```
        }
        icm8[14].h.bcd = sang;
    }

    if(leedong == 15) {
        sang = icm8[15].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm8[15].h.bcd = sang;
    }

    if(leedong == 16) leedong = 17;

    if(leedong == 17) {
        sang = icm8[17].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 5) sang = 0;
        }
        icm8[17].h.bcd = sang;
    }

    if(leedong == 18) {
        sang = icm8[18].h.bcd;
        if(key.h.up_key == M2) {
            sang++;
            if(sang > 9) sang = 0;
        }
        icm8[18].h.bcd = sang;
    }

    if(leedong > 18) leedong = 2;

    if(key.h.enter_key == M2) {
        Second[1].h.SEC_1 =icm8[18].h.bcd;
        Second[1].h.SEC_10=icm8[17].h.bcd;
        Minute[1].h.MIN_1 =icm8[15].h.bcd;
        Minute[1].h.MIN_10=icm8[14].h.bcd;
        // Hour[1].h.HR12_24   = LO;
        Hour[1].h.HR_1   =  icm8[12].h.bcd;
        Hour[1].h.HR_10  =  icm8[11].h.bcd;
        Date[1].h.DATE_1 = icm8[9].h.bcd;
        Date[1].h.DATE_10 =icm8[8].h.bcd;
        Month[1].h.MONTH_1 =
                            icm8[6].h.bcd;
        Month[1].h.MONTH_10 =
                            icm8[5].h.bcd;
        Year[1].h.YEAR_1 = icm8[3].h.bcd;
        Year[1].h.YEAR_10 =icm8[2].h.bcd;



        Ctrl_RTC.h.WP = LO;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
           // Write Permit
        Write_DS1305(W_SEC_
           ADDR, Second[1].x.word);
           // sec write
        Write_DS1305(W_MIN_ADDR,
           Minute[1].x.word); // min write
        Write_DS1305(W_HOUR_ADDR,
           Hour[1].x.word);    // hr write
        Write_DS1305(W_DATE_ADDR,
           Date[1].x.word);   // date write
        Write_DS1305(W_MONTH_
           ADDR, Month[1].x.word);
           // month write
        Write_DS1305(W_YEAR_ADDR,
           Year[1].x.word);   // year write
        Ctrl_RTC.h.WP = HI;
        Write_DS1305(W_CONTROL_
           ADDR, Ctrl_RTC.x.word);
           // Write Protect

        //          01234567890123456789
        lcd_printf(0xC0, "--- Save ---");
        delay(30000);
        delay(30000);
        delay(30000);
      }
    } // while(selv_cal == 11) loop ends
  } // calibration_mode() ends

  // RS232 통신을 통한 저장 시간 간격 조정
  int  Inteval_Set_time() {
      char i = 0;
      int temp = 0;
      union ASC_HEX1 IST_data;

      for(i=0;i<10;i++) {
          icm[i].x.word = Set_Cmd[i];
      }

      temp = temp + ((int) icm[0].h.bcd * 600);
      temp = temp + ((int) icm[1].h.bcd * 60);
```

```
    temp = temp + ((int) icm[2].h.bcd * 10);
    temp = temp + ((int) icm[3].h.bcd);

    IST_data.x.word = temp;

    Ctrl_RTC.h.WP = LO;
    Write_DS1305(W_CONTROL_ADDR,
          Ctrl_RTC.x.word); // Write Permit
    Write_DS1305(W_STHint_ADDR,
          IST_data.h.LowByte);
    Write_DS1305(W_STMint_ADDR,
          IST_data.h.HighByte);
    Ctrl_RTC.h.WP = HI;
    Write_DS1305(W_CONTROL_ADDR,
          Ctrl_RTC.x.word); // Write Protect

    return(temp);
}

// 측정하고 있는 값을 Display한다.
void Massange_Disp(char value1, long value2) {
    char imsi = 0, c = 0;
    long temp = 0;
    union ICM_DATA icm38[8];

    /*                  01234567890123456789
    if(imsi == 1) lcd_printf(0xC0, "Depth :
                +000.00m    ");
    if(imsi == 2) lcd_printf(0xC0, "Ec :
                +00000uS/cm    ");
    if(imsi == 3) lcd_printf(0xC0, "Temp :
                +100.0 C    ");
    if(imsi == 4) lcd_printf(0xC0, "pH : + ");
    if(imsi == 5) lcd_printf(0xC0, "ORIGIN
                A/D : +     "); */

    for(c = 0; c < 8; c++) icm38[c].h.asc = 3;
    imsi = value1;

    if(imsi == 1) {
        temp = value2;
        icm38[0].x.word = (temp < 0) ? '-'
                        : '+';
        temp = labs(temp);
        icm38[6].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[1].h.bcd = temp % 10;
        icm38[4].x.word = '.';
        for(c = 0; c < 7; c++) {
            Lcd_OneCharWrite((0xC8 + c),
                    icm38[c].x.word);
        }
    }

    if(imsi == 2) {
        temp = value2;
        icm38[0].x.word = (temp < 0) ? '-'
                        : '+';
        temp = labs(temp);
        icm38[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[4].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[1].h.bcd = temp % 10;
        for(c = 0; c < 6; c++) {
            Lcd_OneCharWrite((0xC5 + c),
                    icm38[c].x.word);
        }
    }

    if(imsi == 3) {
        temp = value2;
        icm38[0].x.word = (temp < 0) ? '-'
                        : '+';
        temp = labs(temp);
        icm38[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        icm38[1].h.bcd = temp % 10;
        icm38[4].x.word = '.';
        icm38[6].x.word = 0xDF;
        icm38[7].x.word = 'C';
        for(c = 0; c < 8; c++) {
            Lcd_OneCharWrite((0xC7 + c),
                    icm38[c].x.word);
        }
    }
```

```
    if(imsi == 4) {
        temp = value2;
        temp = labs(temp);
        icm38[0].x.word = (temp < 0) ? '-'
                        : '+';
        icm38[5].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[4].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[3].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[2].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[1].h.bcd = temp % 10;
        for(c = 0; c < 6; c++) {
            Lcd_OneCharWrite((0xC5 + c),
                        icm38[c].x.word);
        }
    }

    if(imsi == 5) {
        temp = value2;
        temp = labs(temp);
        icm38[0].x.word = (temp < 0) ? '-'
                        : '+';
        icm38[5].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[4].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[3].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[2].h.bcd = temp % 10; temp
                       = (long) (temp / 10);
        icm38[1].h.bcd = temp % 10;
        //              0123456789012345678
        lcd_printf(0xC0, "ORIGIN A/D :  ");
        for(c = 0; c < 6; c++) {
            Lcd_OneCharWrite((0xCD + c),
                        icm38[c].x.word);
        }
    }
}

// 고정된 Display Title
void MassDis012(char value) {
    char imsi = 0;

    imsi = value;
    //                  0123456789012345678
  // if(imsi == 1) lcd_printf(0xC0, "Depth :
                              +000.00m    ");
    if(imsi == 1) lcd_printf(0xC0, "Depth :
                               .   m    ");
  // if(imsi == 2) lcd_printf(0xC0, "Ec :
                             +00000uS/cm    ");
    if(imsi == 2) lcd_printf(0xC0, "Ec :
                             uS/cm     ");
  // if(imsi == 3) lcd_printf(0xC0, "Temp :
                              +100.0 C     ");
    if(imsi == 3) lcd_printf(0xC0, "Temp : +
                              .  C     ");
    if(imsi == 4) lcd_printf(0xC0, "pH : +  ");
    if(imsi == 5) lcd_printf(0xC0, "ORIGIN
                              A/D : +      ");
}

// 현재 날자 및 시간 Display
void MassTime(int value) {
    char a;
    int Save_Count = 0;
    union ICM_DATA icm82[20];
    union ICM_DATA chuga[5];

    Save_Count = value;

    time_read();
    Second[1].x.word = Second[0].x.word;
    Minute[1].x.word = Minute[0].x.word;
    Hour[1].x.word   = Hour[0].x.word;
    Date[1].x.word   = Date[0].x.word;
    Month[1].x.word  = Month[0].x.word;
    Year[1].x.word   = Year[0].x.word;
    Hour[1].h.HR12_24 = LO;

    icm82[0].h.asc = 3;
    icm82[1].h.asc = 3;
    icm82[2].h.asc = 3;
    icm82[3].h.asc = 3;
```

```
    icm82[4].x.word = '/';
    icm82[5].h.asc = 3;
    icm82[6].h.asc = 3;
    icm82[7].x.word = '/';
    icm82[8].h.asc = 3;
    icm82[9].h.asc = 3;
    icm82[10].x.word = ' ';
    icm82[11].h.asc = 3;
    icm82[12].h.asc = 3;
    icm82[13].x.word = ':';
    icm82[14].h.asc = 3;
    icm82[15].h.asc = 3;
    icm82[16].x.word = ' ';
    icm82[17].h.asc = 3;
    icm82[18].h.asc = 3;
    icm82[19].x.word = 's';

    icm82[18].h.bcd = Second[1].h.SEC_1;
    icm82[17].h.bcd = Second[1].h.SEC_10;
    icm82[15].h.bcd  = Minute[1].h.MIN_1;
    icm82[14].h.bcd  = Minute[1].h.MIN_10;
    icm82[12].h.bcd  = Hour[1].h.HR_1;
    icm82[11].h.bcd  = Hour[1].h.HR_10;
    icm82[9].h.bcd  = Date[1].h.DATE_1;
    icm82[8].h.bcd  = Date[1].h.DATE_10;
    icm82[6].h.bcd  = Month[1].h.MONTH_1;
    icm82[5].h.bcd  = Month[1].h.MONTH_10;
    icm82[3].h.bcd  = Year[1].h.YEAR_1;
    icm82[2].h.bcd  = Year[1].h.YEAR_10;
    icm82[1].h.bcd = 0;
    icm82[0].h.bcd = 2;

    //                  01234567890123456789
    // lcd_printf(0x80, "Setting Time & Date
                        ");
// lcd_printf(0x80, "Water Analysis Dev. ");
    //                  01234567890123456789
// lcd_printf(0x80, "Water Analysis  5000");
    lcd_printf(0x80, "Water Analysis  ");

    chuga[0].h.asc = 3;
    chuga[1].h.asc = 3;
    chuga[2].h.asc = 3;
    chuga[3].h.asc = 3;
    Save_Count = abs(Save_Count);
    chuga[3].h.bcd = Save_Count % 10;
       Save_Count = (int) (Save_Count / 10);
    chuga[2].h.bcd = Save_Count % 10;
       Save_Count = (int) (Save_Count / 10);
    chuga[1].h.bcd = Save_Count % 10;
       Save_Count = (int) (Save_Count / 10);
    chuga[0].h.bcd = Save_Count % 10;



    for(a = 0; a < 4; a++) {
        Lcd_OneCharWrite((0x90 + a),
                    chuga[a].x.word);
    }

    //   lcd_printf(0xC0,   "2001/05/18   15:36
23s");

    for(a = 0; a < 20; a++) {
        Lcd_OneCharWrite((0xC0 + a),
                    icm82[a].x.word);
    }
}

// RS232를 통하여 보내질 현재 Data
void Current_Data(unsigned char mode21, long
                value) {
    char c = 0;
    long temp = 0;
    union ICM_DATA cdma200[8];

    for(c = 0; c < 8; c++) cdma200[c].h.asc
                        = 3;

    if(mode21 == 1) {
        temp = value;
        cdma200[0].x.word = (temp < 0) ?
                            '-' : '+';
        temp = abs(temp);
        cdma200[6].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        cdma200[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        cdma200[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        cdma200[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
        cdma200[1].h.bcd = temp % 10;
        cdma200[4].x.word = '.';
        for(c = 0; c < 7; c++) {
            Now_Data[c+5] =
                    cdma200[c].x.word;
        }
    }
```

```
if(mode21 == 2) {
     temp = value;
     cdma200[0].x.word = (temp < 0) ?
                          '-' : '+';
     temp = abs(temp);
     cdma200[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[4].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[1].h.bcd = temp % 10;
     for(c = 0; c < 6; c++) {
          Now_Data[c+12] =
                    cdma200[c].x.word;
     }
}

if(mode21 == 3) {
     temp = value;
     cdma200[0].x.word = (temp < 0) ?
                          '-' : '+';
     temp = abs(temp);
     cdma200[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[1].h.bcd = temp % 10;
     cdma200[4].x.word = '.';
     for(c = 0; c < 6; c++) {
          Now_Data[c+18]  =
                    cdma200[c].x.word;
     }
}

if(mode21 == 4) {
     temp = value;
     cdma200[0].x.word = (temp < 0) ?
                          '-' : '+';
     temp = abs(temp);
     cdma200[5].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[4].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[3].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[2].h.bcd = temp % 10; temp
                    = (long) (temp / 10);
     cdma200[1].h.bcd = temp % 10;
     for(c = 0; c < 6; c++) {
          Now_Data[c+24] =
                    cdma200[c].x.word;
     }
     }
}

/* ---------------------------
   외부 인터럽트 0 서비스 루틴
   ---------------------------  */
void EX0_int() interrupt 0 using 1
     {
     EX0 = 0;      // 인터럽트 금지
     adc.x.word = P0;  // A/D Converter Data
     Fetch

     if(adc.h.OVER_RANGE == HI)
          TC835[4].h.OVER_RANGE = HI;
     else TC835[4].h.OVER_RANGE = LO;

     if(adc.h.SIGN == LO)
          TC835[4].h.SIGN = LO;
     else TC835[4].h.SIGN = HI;

     if(adc.h.D5 == HI)
          TC835[4].h.BCD = adc.h.BCD;
     if(AD_DIGIT4 == HI)
          TC835[3].h.BCD = adc.h.BCD;
     if(AD_DIGIT3 == HI)
          TC835[2].h.BCD = adc.h.BCD;
     if(AD_DIGIT2 == HI)
          TC835[1].h.BCD = adc.h.BCD;
     if(adc.h.D1  == HI) {
          TC835[0].h.BCD = adc.h.BCD;
          // AD_theorem();
          digit = digit + 1;
          if(digit > 9) digit = 0;
     }
     EX0 = 1;  // 인터럽트 가능
}
```

```
/*****************************
    timer0 인터럽트 처리
    5ms time int
*****************************/
void T0_int() interrupt 1  using 2
     {
     TH0 = 0x4c;
     TL0 = 0x00;
     TR0 = 1;

     t500ms--;
     if( !t500ms ) {
          t500ms = T500MS;
          BZ_sound = 1;
     }

     t3sec--;
     if(!t3sec) {
          t3sec = T3SEC;
          Exchange_time = 1;
     }

     t4sec--;
     if(!t4sec) {
          t4sec = T4SEC;
          Wait_set_time = 1;
     }

     t5sec--;
     if(!t5sec) {
          t5sec = T5SEC;
          Ram_CLS_time = 1;
     }

     t1min--;
     if(!t1min) {
          t1min = T1MIN;
          Lcd_backlight_time = 1;
     }
}

/* Computer가 제어하는 명령어를 읽기 위한
   함수 (Serial Interrupt 사용) */
void S_REC1() interrupt 4  using 1 {
     /* ES = 0; Interrupt 금지 */
     EX0 = 0;      // 인터럽트 금지
     /* 수신 인터럽트일 때 */
     if(RI) {
          RS232_rx_Buf = SBUF; /* Serial
                         Data를 읽는다. */
          RI = 0; /* 수신 표시기를 지운다. */

          if((n == 0)&&(RS232_rx_Buf
                    != STX)) n = n-1;

          if(RS232_rx_Buf == STX) {
               n = 0;
               RS232.h.Rx_sta = LO;
          }

          if(n >= 0) {
                SIO_CMD[n] =RS232_rx_Buf;
                n = n + 1;
          }

          if(n > 25) {
               n = 0;
               RS232.h.Rx_sta = HI;
          }

          if(RS232_rx_Buf == ETX) {
               n = 0;
               pc_cmd_div();
               /* 수신이 완료된 상태 표시 */
               RS232.h.Rx_sta = HI;
          }
     }
     EX0 = 1;  // 인터럽트 가능
}

/* ----------  End of Program  ------- */
```