PAP-697

Small Diameter Meters for Unfiltered Water Systems

This research was supported by the United States
Department of the Interior
Bureau of Reclamation

under Cooperative Agreement No. 1425-3-FC-81-19330 entitled
Water Conservation Innovative Technology Irrigation Water
Water Technology and Environmental Research (WATER)
Project No. WS032

Supplemental funding was also provided by the
Utah Divison of Water Resources

Submitted by S. N. Kartik and Trevor C. Hughes
Utah Water Research Laboratory
Utah State University
Logan, Utah

March 1995

ACKNOWLEDGMENTS

The authors thank Danny L. King and Philip H. Burgi, Bureau of Reclamations's technical representatives, and also Clifford Pugh and David Ehler of the Bureau of Reclamation and Leaunda Hemphill of the Utah Water Research Laboratory for their reviews and suggestions for improvement of this report. Professors Paul A. Wheeler and Blair Stringham provided valuable guidance for the electronic design, as did Professor Gilberto Urroz for the hydraulic design.
The research was made possible by financial support from the Bureau of Reclamation under Cooperative Agreement No. 1425-3-FC-81-19330, and also from Reclamation's Provo Office, and the Utah Division of Water Resources.

ABSTRACT

The growing popularity of residential water systems which deliver unfiltered water for outside use results in large amounts of over irrigation because of the difficulty in providing reasonably priced meters for individual service connections on such systems. There are several commercially available technologies for such meters, but they all appear to cost in excess of $2,000 per meter. The technologies investigated during this research considered both orifice/pressure transducers and strain gauges as possible approaches to providing low cost solutions to this problem. The pressure transducer prototype meter was found to be too sensitive to the very large line pressure variations encountered in such systems. The strain gauge approach was determined to have the best potential during laboratory tests. Ten of these meters were produced and tested during one summer irrigation season in several different dual systems in Utah. During field tests difficulties with both temperature variation and lack of water tight enclosures were encountered, however it is believed that both of these problems could be solved by installation in deeper meter boxes and provision of a water proof container. The cost of the strain gauge meter appears to be an order of magnitude smaller than the existing alternative technologies.

Key Words
Water Meter, Dual System, Strain Gauge, Unfiltered Water

EXECUTIVE SUMMARY

Introduction

Dual water systems - pressurized distribution systems which deliver unfiltered water to residential customers for outdoor irrigation, are widely used in Utah, and are being recommended for additional regions of the western U. S. as a way to conserve high quality local groundwater for indoor uses. While these systems make major reductions in the water required for the treated potable systems, they also cause a large increase in total water required per family. The reason is that none of the usually 1" diameter service connections are metered because of the difficulty in providing reasonably priced meters for unfiltered water. There are several commercially available technologies for such applications, but they all appear to cost in excess of $2,000 per meter.

Objective

The purpose of this research was to design and test a low cost small diameter meter which is suitable for use in unfiltered water systems.

Scope

The technologies tested during the study included laboratory testing of both orifice plates with pressure transducers and strain gauges as possible metering devices. The strain gauge meter appeared to be most promising and was therefore field tested (10 meters in 7 different dual systems) during the summer of 1994.

The strain gauge was connected to a battery powered micro processor and circuitry which was programmed to record data once each 30 seconds and totalize the resulting water volume. Data was recovered by connecting to a Lap top computer.

Results and Conclusions

1. The orifice/pressure transducer prototype meter calibration was found to be very sensitive to the very large line pressure variations encountered in such systems, and therefore was not field tested.
2. The strain gauge approach was determined to have the best potential during laboratory tests. Ten of these meters were produced and tested during one summer irrigation season in several different dual systems in Utah. The calibration of these meters were tested both immediately after installation and also after two to three months of service. The field calibration tests showed an error of 15 percent or less for all meters which were functional. These errors were less than 8 percent for 3/4 of the tests and less than 3 percent for 1/2 of the tests. A large fraction of the larger errors (those over 8 percent) may have been due to

field test conditions other than those related to the strain guage meter device.

3. The strain gauge meter calibration was found to be stable during field tests, except for sensitivity to diurnal temperature variations in meters which were installed very near the ground surface. Various methods were used to reduce the effect of temperature during field tests, including coating with silicone and addition of a thermistor to the circuitry.

4. The temperature sensitivity problem could be completely eliminated if meters were installed approximately three feet deep rather than very near the surface (where most service lines are located). This deep installation, however, would require a water tight enclosure around the electronic package.

5. The cost of the strain gauge meter appears to be an order of magnitude smaller than the commercially available alternatives. Materials for the prototype design cost approximately $100. Purchases in large quantities should be substantially less than this (perhaps 1/3 less), but assembly line type labor cost and profit would add an unknown additional quantity (as much as three times the materials cost). A rough estimate of a mass produced device would therefore be $200.

LIST OF FIGURES

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Problem description

The need for a meter to measure the flow of low quality unfiltered water in small diameter pipes arises from use of dual water systems. Dual water systems are used in several counties in Utah, and are gaining wider acceptance in other western states. Dual systems supply untreated and unfiltered water to domestic consumers to be used for irrigating lawns and vegetable gardens. The motivation for such systems is to reduce demand on the high quality municipal treated water system.

A study of dual systems (Vaughn Hansen Associates [1]) in 1985 found that the residential demand for water is divided roughly equally between indoor consumption and outdoor irrigation. The indoor component is spread almost equally over 12 months while the outdoor component occurs within about six summer months. This implies that by using low quality, untreated water to serve the residential irrigation demand, a city could serve twice the present population with potable water. In addition, the dual water system avoids treatment costs and extends the period over which local natural high quality sources can meet a city's demand.

The experience in Utah has been that communities with dual systems use about double the quantity of outdoor water relative to cities without dual systems [1]. This is because the dual system is not metered, and therefore there is no reason to conserve water. The reason for not metering dual systems is that conventional technology for metering small diameter residential connections requires the use of a propeller type meter. These meters quickly experience maintenance problems with unfiltered water. In large diameter main lines in dual systems, propeller meters are used with significant, but manageable, maintenance costs. However using conventional meters in residential lines, usually 1" diameter pipes, is simply not feasible.

Technology for metering unfiltered water does exist. Some devices, such as magnetic meters, can measure flow through a pipe without any contact with the water, but such meters are very expensive. A conventional municipal water meter costs about $65, while a magnetic flow meter costs about $2500 [2] and therefore is certainly not feasible for installation at every residential outlet. A survey of water meters currently available has determined that no device is available which is both technically and economically feasible for use in small diameter residential lines served by unfiltered water.

Research objectives

The purpose of this project was to design, develop, and test a meter that can be used to meter unfiltered water supplied to residential customers through one inch diameter service connections. Specific sub-objectives included:

1. Conduct a thorough survey of possible technologies which may be used to measure the rate of flow of unfiltered water in small diameter (1") pipes.
2. Develop innovations in both electronics and hydraulic technologies which, when combined, will provide a suitable meter design for the proposed application.
3. Design and construct prototypes of both the strain gauge and the orifice type meters. Any other type of meter which seems promising may also be developed.
4. Test the prototype meters under laboratory conditions at the Utah Water Research Laboratory. Based on these tests, identify which type(s) of meter(s) are likely to perform satisfactorily.
5. Build a number of such meters and install them at selected locations in Utah, and monitor their performance during one full irrigation season.
6. Evaluate the performance of the meters and make recommendations regarding future development and use of the product.

CHAPTER 2

LITERATURE SURVEY

Possible Technologies

There are many types of measurements that can be used to measure flow rate of unfiltered water. However the use of many of these methods is precluded by two major constraints -- cost and energy requirement. To be economically feasible, the cost of the meter must be comparable to the cost of a conventional municipal meter (around $100). Also, the power consumption must be very low (of the order of 1 milliwatt). The latter constraint derives from the need to operate from a battery (which lasts at least one irrigation system) rather than providing a connection to an external power source.

Technologies which use propellers or positive displacement devices such as conventional municipal meters are not discussed here because of their moving parts cause them to clog easily in unfiltered water. Technologies which are technically (but not necessarily economically) feasible are discussed below.

Magnetic Meters: Water is a conductor of electricity. Principles that measure speed of motion of a conductor can be used to measure flow rate. Many meters use the Faraday effect to measure flow rate, which is: a conductor moving across a magnetic field develops an electric potential across its ends. In a magnetic meter, a uniform AC magnetic field is provided along one diameter of the pipe, and a pair of electrodes mounted on the walls of the pipe perpendicular to the magnetic field picks up the induced electric potential. The flow rate is calculated from the induced voltage across the electrodes. Omega, a company that manufactures magnetic meters, produces many types of magnetic meter models, with the prices in the range of $2000 to $4644 [2].

Probe Type Meters: Some meters use small probes that do intrude in the path of flow but do not use moving parts that can get clogged by sediment. These are essentially magnetic meters that have a small electric coil built into the probe to produce a magnetic field. An example of such a meter is the Flo-Tote 286 - a product of Marsh-McBriney Inc. which is marketed at $6200 [3].

Ultrasound: In this method, a transmitter and a receiver are installed in the pipe. The transmitter emits ultrasound waves of a certain frequency. These waves are reflected by suspended particles or gas bubbles (discontinuities) in the liquid. Due to the Doppler effect, the receiver observes these waves at a frequency that is different from the transmitted frequency if these discontinuities are in motion. The difference between

the frequency of transmission and reception is related to the velocity of the liquid and is used to calculate flow rate [2]. An example of this type is a 1" diameter meter marketed by Nu-Sonic Inc. at $3,500.

Vortex Shedding Meters:
When a liquid flows around a vortex-shedding body, vortices are produced downstream of the body. The vortices are shed alternately from one side of the body, and then the other, in a regular pattern. The velocity of the fluid is related to the vortex shedding frequency and the width of the shedder. A sensor mounted on the body (e.g. a piezoelectric sensor) is used to measure the frequency at which vortices are shed, and using this measurement, the fluid velocity can be calculated.

This method does not use any moving parts, but the cost of a vortex flowmeter is very high - Omega Engineering markets its flowmeter at $1752 [5].

Hot wire anemometer
This method involves maintaining a wire at an elevated temperature inside the flow of a fluid. The energy dissipated by the wire is related to the flow rate - higher the flow rate, the greater the energy that is dissipated. A measurement of the electrical power required to maintain the temperature of the wire is used to determine the flow rate of the fluid. Keeping the wire at an elevated temperature consumes a lot of power, and therefore this method could not be used. The hot wire element is also very fragile for use in the field.

Strain gauge
A solid body is introduced into the path of flow of a fluid, so that a force acts upon it. This causes some deformation of the solid or its mounting. The magnitude of this deformation, measured by a strain gauge, can be used to determine the flow rate of the fluid. This arrangement is also called a drag-body flow meter. The use of a strain gauge in a drag-body flow meter was suggested as early as 1962 by Mead Stapler [5]. Blair Stringham investigated the use of a strain gauge meter to measure irrigation water discharge in 1987, and obtained encouraging results [6].

The cost of a strain gauge ranges between $3 and $13, depending on the size and the attachments provided with it [7], and the power consumption of a strain gauge is a few milliwatts at most. This method was therefore considered to be a suitable candidate, and was tested in this project.

Orifice meter
A plate with an orifice cut into it is placed in the path of the flowing liquid. The passage of water through the orifice causes a pressure difference between the two sides of the

orifice. This pressure difference is proportional to the square of the velocity of the liquid. The pressure difference, measured by a differential pressure transducer, is used to calculate the flow rate of the fluid.

Many inexpensive pressure transducers are available in the market, though finding one that met the requirements of this application proved to be quite difficult (as explained in Chapter 3). The power consumption of a pressure transducer may be as low as a few milliwatts, depending on the type of transducer. This method also appeared to be suitable for testing during this project.

CHAPTER 3

DESIGN OF THE METER

The measuring device consists of two main parts - the measuring unit, which is a small length of pipe containing either the pressure transducer or strain gauge, and the meter unit, which contains all the electronics required to convert the pressure or voltage signal into a flow rate and/or volume of water.

Both the strain gauge and the pressure transducer produce a similar kind of output - a DC voltage, of the order of a millivolt, whose magnitude is related to the flow rate. Therefore the meter unit will be substantially similar for both kinds of meters. The only differences will be in the power supply to the transducer, the level of amplification required on the transducer output, and the way the flow rate is calculated from the transducer signal.

To design the measuring unit, an upper bound had to be set on the flow rate that the meter can measure. It was judged that for a 1" diameter pipe connected to a set of sprinklers or hoses, with water pressure at about 60 psi, a maximum flow rate of 22 gallons per minute would be a reasonable estimate. Most of the meters were designed for this range (0-22 gpm), but some meters were designed to be able to measure higher flows also.

The measuring unit

The measuring unit is a small length of pipe containing the assembly required to measure flow rate. It is installed in the pipeline carrying water from the main line to the house. This length of pipe should be long enough that the flow inside most of the pipe is not affected by flow variations created by the joints at either end of the pipe, and is representative of the flow in the line.

Orifice meter

The measuring unit consists of two lengths of pipe, each about a foot long, fitted with a flange at one end. A segmented orifice plate is placed between the two flanges which are then bolted together. There is a pressure tap on each side of the orifice, from which tubes lead to a differential pressure transducer mounted on the meter unit. The flow rate in the pipe is calculated from the pressure difference across the orifice.

The selected orifice was segmented, i.e., the opening is a segment of a circle, and the circular part is flush with the wall of the pipe. The orifice is installed with its circular edge flush with the bottom of the pipe so that solid particles in the water can pass through without getting collected in front of the orifice plate.

Dimensions of the orifice plate

The accuracy pressure transducer is increased by increasing the magnitude of the head loss across the orifice. However, a large pressure drop across the plate may lead to unacceptably low pressure at the outlets. A maximum pressure drop of about two psi was judged to be acceptable, and sufficient for reasonable accuracy. A segmented orifice with the dimensions as shown in Figure 1 was found to suitable, as it produced a pressure drop of about 1.2 psi at 60 psi line pressure for a 18 gallons per minute flow.

Pressure transducers have diaphragms made of either stainless steel or silicon. Steel diaphragms are suited for high pressures and offer good corrosion resistance, but are more expensive, while silicon diaphragms are generally used for lower pressures, offer higher accuracy, and are less expensive. However, finding a pressure transducer suitable for measuring the differential pressure across the orifice was quite difficult, because the transducer would have to measure differential pressures less than 1 psi, while the line pressure could be as much as 150 psi. Only one pressure transducer came close to meeting this specification - the PX160-005DV from Omega Engineering. The differential range of this silicon-diaphragm transducer was 0-5 psi, and it was rated at 60 psi common mode (line) pressure.

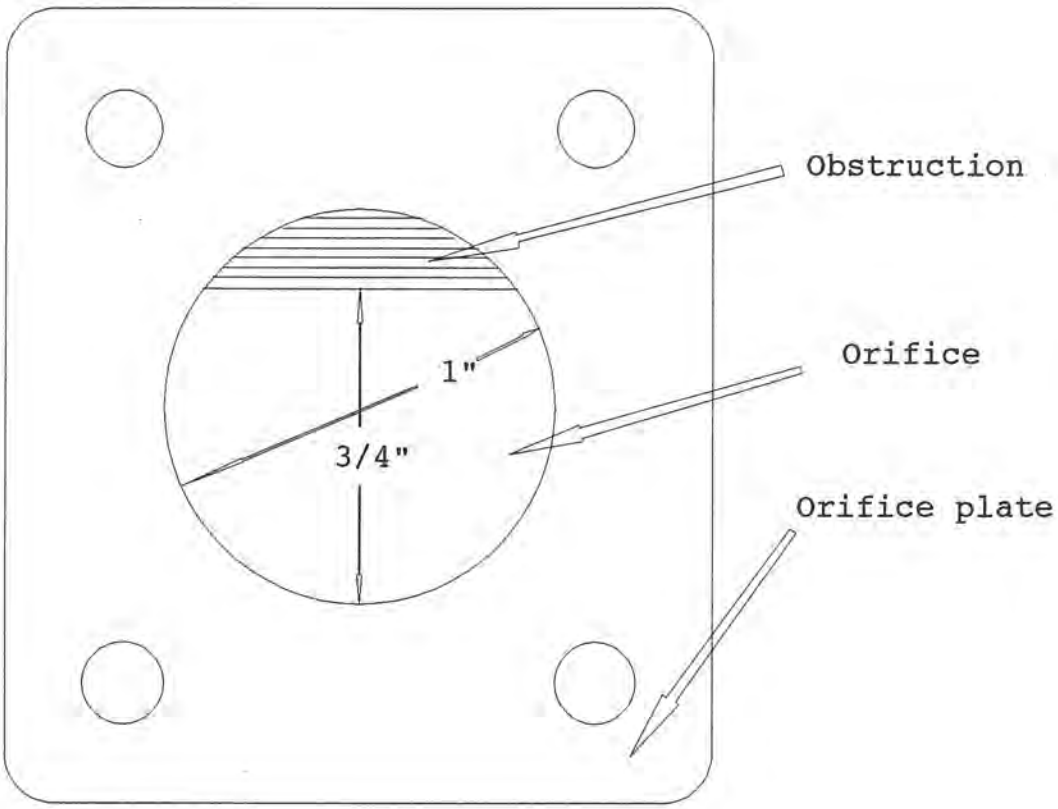

Figure 1. An orifice plate

Strain gauge meter

The measuring unit is a length of pipe that contains some object that gets deformed by the flow of water in the pipe. The object that was chosen to be deformed was a small strip of stainless steel arranged like a cantilever, with one end fixed at the wall of the pipe, and the free end in the path of flow of water. Flow of water inside the pipe produces a force on the strip and causes the metal to bend slightly in the direction of flow.

A strain gauge fixed to the strip was used to obtain a measure of the deformation, which in turn could be used to obtain a measure of the flow rate in the pipe. The resistance of a strain gauge changes whenever the object on which it is fixed, is strained. The magnitude of change in its resistance depends

upon the magnitude of the strain. A strain gauge does not measure bending strain directly - it measures longitudinal strain. The gauge could be fixed either to the surface that suffered compression, or the surface that suffered tension. The compression surface was chosen because it was felt that the strain gauge would be away from the side that the water rushed at, and would thus be a little less vulnerable to a breach in water-proofing. In any case, there was no apparent disadvantage in placing the strain gauge on this side.

Many inexpensive strain gauges were available in the market. The KFG-02-120-C1-11L1M2R from Omega Engineering, which has dimensions of 3.3mm(l)x2.4mm(w), was selected as a suitable strain gauge.

The dimensions of the strip, i.e., its length, width and thickness, determine the magnitude of bending when exposed to a certain flow rate. The width of the strip must be at least as large as that of the strain gauge. It was desirable that the length of the strip be at least half the pipe diameter, so that the strip was exposed to all the layers of flow in the pipe - from zero velocity at the wall, to maximum velocity near the middle of the pipe. Given these minimum dimensions for width and length, the thickness of the strip had to be quite small so that even small flows in the pipe would cause enough bending in the strip to be measured accurately by the strain gauge.

These dimensions can be calculated theoretically from the modula of elasticity of the material, range of flow rates that will be encountered, desired strain etc. Stringham [6] developed a computer program that does all the required calculations, but the thickness of the strip calculated by the program was found to be too small to be practicable. The practical dimensions may be quite different from the calculated ones due to the fact that the layers of water proofing material on top of the strip cause a significant change in the properties of the strip. Hence the dimensions of the strip had to be determined by trial and error.

The probe

The small strip of stainless steel needed to be mounted such that one end - the one near the pipe wall - was held rigid, and the other end - the one that protruded into the path of flow - was free to move. The wires from the strain gauge had to be passed through the walls of the pipe to the meter unit by some leak-proof arrangement. The probe is an assembly that was designed to perform both these functions.

The casing of the probe consisted of a brass nipple (a hollow brass cylinder, with an external diameter of 1/4" and threaded

on the outside at both ends). The strain gauge was first fixed to a steel strip and then the strip was introduced into the cylinder, with the strain gauge and a predetermined length of the strip sticking out of one end. The cylinder was then filled with water proof epoxy resin. The resin hardened on standing and held one end of the strip rigidly in place inside the cylinder. This assembly could then be screwed into the pipe through a hole drilled in the pipe wall. Thus the probe provided a rigid mounting for the strip, and a leak proof arrangement for the strain gauge wires to be led out. A diagram of the probe is shown in Figure 2.

The free end of the strip and the strain gauge are placed inside the pipe. The strain gauge had to be kept dry to function properly, therefore some sort of waterproof covering had to be applied on the strain gauge. A few different flexible water-proof coverings were tried. A single layer of any one covering was found to be insufficient. After trying multiple layers of the same covering, and then layers of different coverings, it was found that multiple layers of Dow Corning RTV-3140 silicon rubber produced the best water-proof covering that still retained the elasticity of the strip to a large extent.

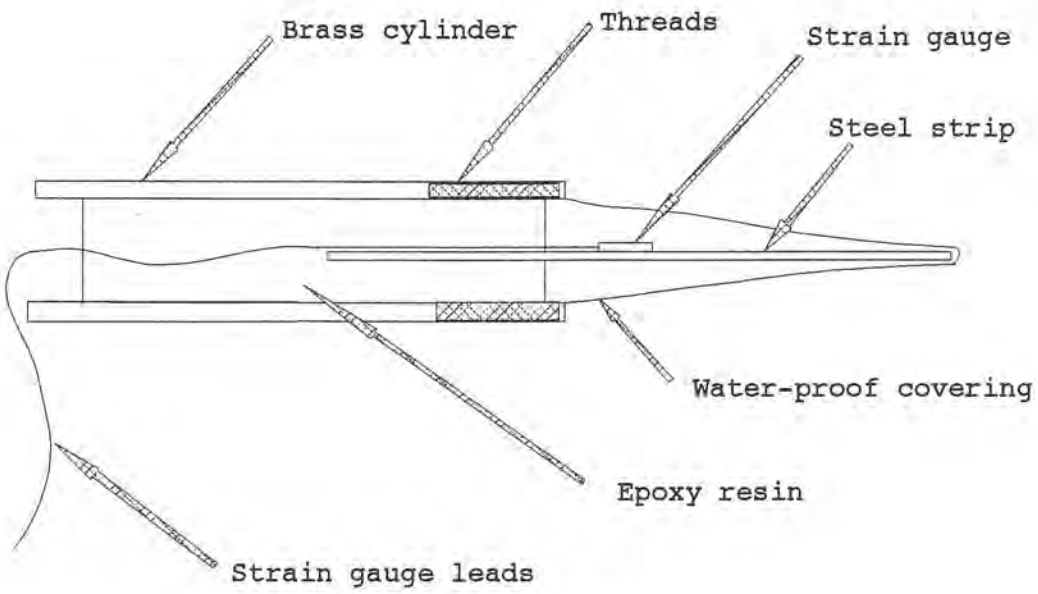

Figure 2. The strain gauge probe

The unfiltered water flowing in the pipe may contain algae, weeds and grass, which could get caught on the probe and cause erroneous readings, resulting in an erroneously large signal. Several features were designed to prevent this. First, the water-proof covering on the strip was made smooth, so that it was difficult for anything to get caught on it. Second, the water-proof covering was tapered towards the free end of the strip so that anything caught on it would tend to slip off. For further protection, the probe was not installed perpendicular to the flow, but at an angle of 45 degrees in the direction of flow, so that the likelihood of anything getting caught on the probe was very small.

A hole was drilled into the pipe at the required 45 degrees angle and threaded so that the probe could be screwed in. Thick walled schedule 80 pvc pipe was used to ensure that there would be sufficient number of threads in the hole to hold the probe securely. Figure 3 shows a probe screwed into a pipe. The probe had been designed so that it could be screwed on and off the pipe, and therefore easily replaced if

the strain gauge or the strip got damaged. However the hole always started leaking sooner or later, therefore epoxy resin was applied all around the hole to make the joint leak-proof. This made the connection between the pipe and the probe a permanent one - to replace any part of the probe, the whole measuring unit (pipe plus probe) had to be replaced.

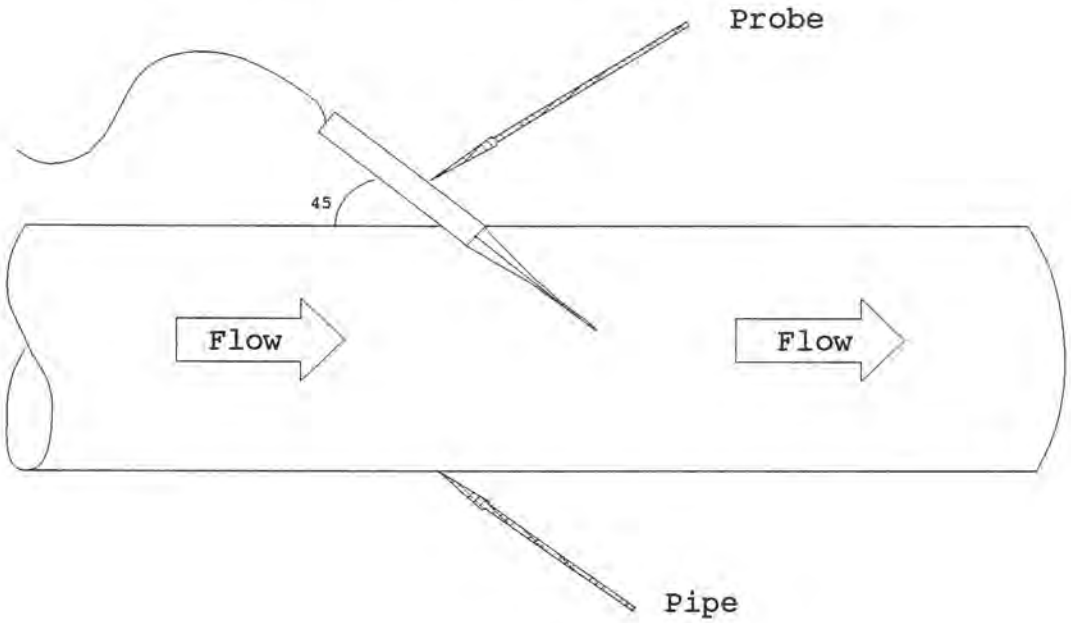

Figure 3. The probe installed in a pipe

The following dimensions were found to work very well :

Thickness of stainless steel strip	: 0.005 inches
Width of strip	: 7/32 inches
Length of strip (free tip to strain gauge)	: 14/32 inches

Using a shorter length of strip caused the sensitivity of the probe to decrease and the maximum measurable flow to increase. Longer strips had the opposite effect. The edges of some of the strips were rounded off for further reducing the chance of debris getting caught on the probe.

The meter unit

The meter unit contains all the electronics necessary for proper operation of the meter. The meter unit carries out the functions of processing the signal from the measuring unit, converting this signal into a measurement of flow rate, accumulating flow rate over time to get the total quantity of water consumed, and providing some means of reading the meter.

The meter unit for a strain gauge meter is identical to that for an orifice meter in most respects. At first, only a "basic" design for the meters was drawn up and realized. The "basic" design simply fulfilled the functional requirements of the meter - there was no attempt to optimize the design in any way. The meters were then tested in the laboratory (described in Chapter 4). After the laboratory testing, various modifications were made to optimize the design for power consumption, circuit board space, etc. The description of the meter unit given below describes the optimized design of the strain gauge meter. The orifice meter unit differed only in the signal conditioning circuitry and the power supply, and these differences are described later.

The meter unit could be divided into the following five sections based on the functions of each:

1) power supply section,
2) signal conditioning section,
3) measurement and calculation section,
4) communication section, and
5) timing and control section.

A schematic of the meter unit is shown in Figure 4. A description of each section is given below.

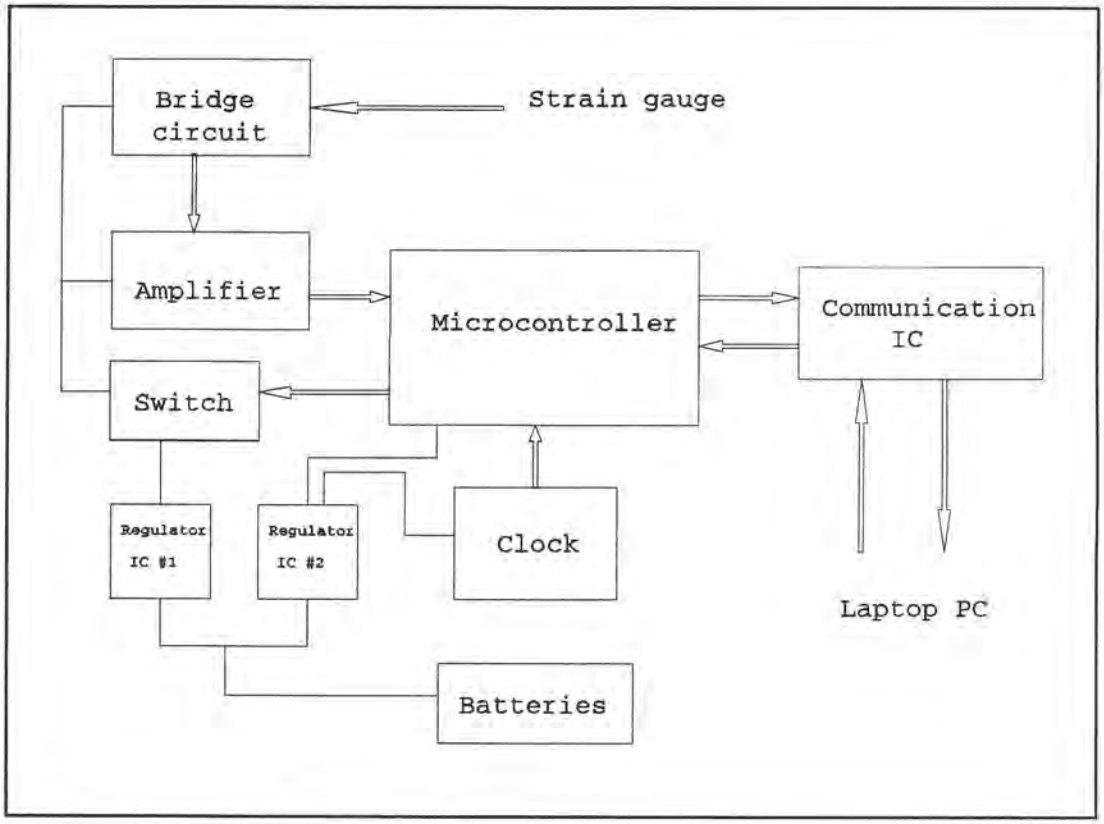

Figure 4. Schematic of the meter

Power supply

The power supply for the meter was a very important component in the design of the meter. The meter had to contain its own power source, which meant some sort of battery. Solar collectors could have provided adequate energy, but would be infeasible both for economic and vandalism reasons. The qualities that the battery needed are high energy content, low price, and small size, in descending order of importance. It was desirable that the battery should last through one full irrigation season, not increase the price of the meter by much, and not cause a significant increase in the size of the meter box being envisaged. The strain gauge meter needed a battery voltage of at least 4.25V. A survey of all the types of batteries available in the market resulted in a

decision to use four Duracell "D" type alkaline cells. Each cell is rated at 1.5V, so four cells in series would give an output voltage of 6V. Each cell had a total energy content of 14,250 mA-hours [8]. The cells would be connected in series, therefore the combined energy content of the battery pack would be 14,250 mA-hours only. The cost of the cells was quite low, and the meter box could hold four or six cells without having to have its size increased.

As an alkaline cell is drained, its voltage, though decreasing, remains at usable levels till a certain voltage is reached, called the cutoff voltage, after which the voltage falls off rather quickly. For a Duracell, the cutoff voltage is 0.8V. However at this stage the power supply to the meter would be at 0.8 x 4 = 3.2V, which was too low for the meter. The voltage regulators needed at least 4.25V to produce a regulated output of 3.25V needed by the measurement circuits [9]. Therefore a battery could not be allowed to drain past 1.0625V. This meant that the meter could use only about 62.5% of the energy content of a cell. Thus the total usable energy content of 4 cells was 62.5% of 14,250 = 8,907 mA-hrs. If the batteries were to last through one irrigation season (6 months = 4320 hours), then the maximum constant current drain from the batteries could not exceed 8,907/4,320 = 2.06 mA. Equivalently, the total power consumption of the meter could not exceed 2.06mA x 5.125V (average useful voltage) = 10.6 mW. This became a critical factor in the selection of many of the components and in the design of the operation of the meter.

The electronic components of the meter involved in the signal generation, signal processing and measurement process needed a constant supply voltage, but the output voltage of the battery is not sufficiently constant over the entire useful life of the battery. Hence a voltage regulator was needed.

The power consumption of ICs generally drops as the supply voltage is reduced, but care must be taken to ensure that this reduction does not degrade the performance of the IC. It was found that the highest voltage required on the circuit was 3.25V, by the amplifier IC. It was also found that reducing the supply voltage of the microprocessor from 3.25V to 3.0V produced very significant power savings. Therefore it was judged worthwhile to have two different power supply voltages in the meter instead of just one, and hence two voltage regulator ICs were used to provide 3.0V and 3.25V supplies.

Constant voltage regulator ICs are quite inexpensive. Using precision resistors to program the IC, any desired constant voltage within a certain range can be obtained. Two MAX663 ICs were used for power supplies, with different resistors connected to their programming pins to produce the desired voltages. A 1M resistor and a 1.348M resistor were connected

to one MAX663 to produce a regulated 3.05V output (which was the closest easily obtainable voltage to 3.0V), and a 1M resistor and a 1.5M resistor were connected to the other MAX663 to produce a 3.25V output.

Signal conditioning

The purpose of the signal conditioning circuits was to convert the transducer output into a form that can be understood by the processor. The processor used in the meter was a Motorola MC68HC811E2, which had a built-in analog to digital convertor (A/DC). The power supply to the microprocessor was at 3V, so the signal conditioning circuits had to convert the sensor output into a voltage signal between 0 and 3V.

The magnitude of change in resistance of a strain gauge indicates the magnitude of strain it is under. A Wheatstone bridge was used to convert this change in resistance into a voltage signal. The bridge was supplied with the 3.25V supply, and the output of the bridge was adjusted to 1 mV when there was no flow, and about 4 mV under maximum flow. This output was fed to an amplifier whose task was to amplify this voltage into a voltage signal between 0 and 3V. The reason for not adjusting the output to zero volts for zero flow was that the amplifier was supplied with only a single polarity supply i.e. 0V and 3.25V, and this resulted in a "dead" zone near 0V, where the amplifier could not distinguish between a 0V input and some input very close to, but greater than 0V.

The amplifier stage consisted of a 358 dual OP-AMP IC. The two OP-AMPS were connected together to form a differential amplifier with a gain of 330. The output of this stage would thus be greater than 0V, and less than 2V. The A/DC of the microcontroller had an input range of 0 to 3V, so the upper bound on the amplified output voltage could have been increased to 3V for better resolution and accuracy, but the behavior of the amplifier started becoming non-linear for voltages higher than 2V (due to the low supply voltage of 3.25V). Therefore the gain of the amplifier was not made greater than 330.

The Wheatstone bridge and the amplifier IC were the biggest power consumers in the meter unit. To reduce power consumption, they were supplied power only every 30 seconds. Typical irrigation practice would be to leave sprinklers on for 1/2 to 1 hour before shutting them off, or starting to irrigate another area. Therefore the flow rate would be relatively constant most of the time, and it was judged that an accurate reading of flow rate twice a minute would be quite representative of the average flow rate throughout the several minute period on use. This allowed the measuring unit and signal conditioning section (the bridge and amplifier) to be

kept idle and without power for most part of a minute, and be activated and supplied with power for only a small duration twice a minute. However, each time these circuits were switched on, they needed a little time to stabilize. Hence, after switching on these circuits, a small interval of two seconds was allowed before any measurements were taken.

Measurement and calculation

The output of the amplifier stage was fed into the microprocessor, whose task was to measure this voltage and calculate the flow rate from it. The flow rate was then accumulated over time to give the volume of water that flowed through the meter.

Because each probe was assembled by hand, there were small differences in the length of the strip, the position of the strain gauge on the strip, the thickness of water proofing layer, and the angle of orientation of the probe, from one measuring unit to another. These differences caused a different output by each measuring unit for the same flow rate. Therefore it was not possible to have a single calibration curve for all measuring units - each unit needed its own calibration curve. This made the problem of finding an analytical equation relating amplified voltage output to the flow rate tedious, because a new equation had to be found for each unit. Hence, instead of an equation, a look-up table was used to find the flow rate corresponding to a certain voltage. A big advantage of using a look-up table is accuracy - fitting a polynomial (perhaps up to the fourth degree) to the calibration curve would lead to some error at many points along the curve, but a look-up table (having perhaps 12 points) used with interpolation was more accurate.

The built-in AD/C in the microprocessor quantified the output voltage of the amplifier into an integer between 0 and 255. The calibration curve was thus a relation between the quantified voltage and the flow rate through the measuring unit. Determining the calibration curve and programming the look-up table are described in Chapter 4.

To get a good measurement of flow rate, the output of the amplifier was measured 256 times in succession, and the average of these readings was taken as the true measurement. This took less than two seconds. The flow rate corresponding to the voltage reading was looked up, and thus determined. One measurement was taken every 30 seconds, thus dividing the flow rate by two yielded the quantity of water that flowed through the meter in 30 seconds. This quantity was added to the total quantity of water consumed up to the previous 30 second interval. Thus a cumulative tally was maintained of the total quantity of water consumed.

Communication

Some method was required to communicate the numbers calculated by the meter (i.e., flow rate, quantity of water metered, present time, and status) to the user. This could take the form of a visual readout or an electronic communication to a recording device. LED displays are inexpensive but consume a lot of power. LCD displays consume much less power but are quite expensive. The added computational complexity of displaying many large numbers on a display with limited number of digits was another factor discouraging the use of a visual display.

Using a personal computer (PC) as a recording device held many attractions. Laptop and notebook PCs are readily available and can be easily carried around by any person wishing to read these meters. Any amount of data could be transmitted to the PC quite easily using a single transceiver IC and a serial cable connecting to the PC's serial communication port. A simple program was written for the PC which would read the numbers, made any necessary calculations, and presented the date to the user in any desired format, and/or recorded the data on disk for future analysis.

The MC68HC811E2 microcontroller was capable of serial data communication at TTL level voltages (0V and +5V), but these had to be converted to the PC's RS-232C format, which used different voltage levels (-12V and +12V). There are many ICs that do this job, but the MAX232 IC was selected as it required only a single polarity supply from which it generated all the voltage levels required for conversion from RS-232C levels to TTL levels and vice-versa. The communication chip did not draw power from the meter unit at all - one of the PC's unused signal wires was switched to the +12V state, and used as the power supply to the chip. Four wires were used for data communication. Each side transmitted on two wires - data on one, and a control signal on the other - and received corresponding signals on the other two.

The important numbers to be communicated to the PC were flow rate, time since installation, and total quantity metered to date. Some other numbers that contained information about the present status and working of the meter were also communicated.

The meter was read by connecting a serial cable from a PC to the meter, running a meter-reading program on the PC, and then pressing a push-button switch on the meter that signalled the microcontroller to start transmitting data.

Timing and control

All the functions of the meter were controlled by the microcontroller. The functions were supplying power, signal conditioning, reading in measurements and calculating, and transmitting data. None of these functions took much time. Every time a flow rate measurement was taken, some calculation was required to calculate quantity of flow, etc., but measurements were taken only once every 30 seconds, so calculation was also required only once every 30 seconds. Therefore the microcontroller itself was idle most of time, and did useful work only once each 30 seconds. A lot of energy could be conserved if the microcontroller were also switched off in the idle time, but this would lead to problems like loss of data, rebooting the microcontroller, etc. The best solution was to put the microcontroller in a "sleep" or STOP state, where it consumed very little power, retained all memory, and did not have to be rebooted. Unfortunately the microcontroller could not "wake" itself up, so an external electronic clock was required to "wake" up the microcontroller once every 30 seconds.

The external clock is active and consuming power all the time, therefore to be useful, the clock obviously had to consume much less power than the microcontroller. The MM58274 real time clock IC was selected as the electronic clock. It was programmed to generate a signal every 30 seconds that would "wake" up the microcontroller.

The way in which one would expect the microcontroller to work once it was "awakened" is to switch on the measurement circuits, idle for a couple of seconds, take measurements, shut off the measurement circuits, carry out all the calculations, and then go back to "sleep". However, with the intention of reducing power consumption, this sequence was modified slightly. Once the measurements were taken and the measurement circuits switched off, instead of the controller carrying out the required calculations, it put itself to "sleep" immediately. After 30 seconds, when it was "wakened", it switched on the measurement circuits, and while waiting for them to stabilize, it carried out all the calculations for the previous measurement. Thus some part of this idle time was fruitfully used, and the time for which the microcontroller would be "awake" was reduced. This meant that the meter readings would always be 30 seconds behind the actual physical quantities.

A transistor, connected as a switch, was used to switch on and off the bridge and amplifier circuits. The transistor was connected to one of the output pins of the microcontroller, and could be turned on and off under software control.

The microcontroller was normally "wakened" every 30 seconds by the real-time clock, but the push-button which signalled a

request for data transmission could "wake" it up too. Every time the microcontroller was "wakened", it checked to see which device caused it to "wake up", and it took necessary action. When one device requested service while the controller was already awake and doing some work, the controller answered the request after the work at hand was completed. Each request simply turned on a "flag". The microcontroller checked each flag in turn - if a flag was on, the controller would carry out that particular task, and the flag would be turned off. Once both flags were found off, the controller goes to "sleep".

The orifice meter unit

The orifice meter unit differed only in the measurement circuitry and the power supply. The pressure transducer required a power supply of 9V, which required more batteries and a different voltage regulator. The output of the transducer was a voltage signal between 0 and 10 mV (not a change in resistance), so a Wheatstone bridge was not required. The transducer was designed to measure differential pressure in the range of 0 to 5 psi, but the maximum pressure difference across the orifice was about 2 psi. Therefore the output of the pressure transducer was never higher then 5mV. This was fed to the amplifier, which amplified the signal by a factor of 400, so the amplified signal was between 0 and 2V. The calculation of the flow rate was carried out in the same way as for the strain-gauge meter - a look-up table was used to determine the flow rate from a measurement of the amplified transducer output.

Software

The MC68HC811E2 microcontroller has a dynamic Random-Access Memory (RAM) of 256 bytes, and an Electrically Erasable Read-Only Memory (EEPROM) of 2K. The 256 byte RAM was used to store the numbers that the controller kept updating, like the time, flow rate, etc. The 2K EEPROM was used to hold the main program that directed all the operations of the microcontroller. The contents of the RAM are lost whenever power to the controller is switched off, but those of the EEPROM remain intact until they are written over.

The microcontroller was programmed through a personal computer (PC). The serial communication capabilities of the microcontroller were used for this purpose. The serial port of the microcontroller was connected to the serial port of the PC using a serial cable, with a line driver/receiver in the middle for proper signal translation. An assembler program was written on the PC to convert the MC68HC11 series assembly language into executable machine code. Another program was written to transfer executable programs and data to the

microcontroller through the serial port, and then start execution if desired. This program also monitored the serial connection, and displayed the output of the microcontroller on the screen. In this way programs were loaded into the microcontroller and run, and the progress monitored. In short, programming the microcontroller involved writing an assembly language program, assembling it into machine code, loading it into the microcontroller memory, and then executing the program.

Calibration

Before the meter could be used for measurement, it had to be calibrated, i.e., its look-up table had to be created. Once this was done, the meter could start measuring quantities of water. Separate programs had to be written for the calibration and metering modes of the meter. The 2K EEPROM was large enough to hold both these programs at the same time. Every time the controller was powered up or reset, it began execution from the address specified at a certain memory location. When the controller was first programmed, this memory location was made to point to the calibration program. Once calibration was completed, the address at this location was changed to that of the metering program.

The measurement of the output voltage of the amplifier and its conversion from analog to digital, was done under software control. The A/DC was quite fast, and could sample and quantify voltages hundreds of times a second. The A/DC was made to sample the measurement voltage 256 times in succession, and the average of these readings was used by the program as the correct voltage reading. It was hoped that sudden spikes and errors could be avoided this way. As soon as all the conversions were done (which took less than 2 seconds), the A/DC circuits of the controller were shut down to conserve power. These circuits did take a small time to stabilize when powered up, but this was much smaller than that taken by the measurement circuits, and was therefore ignored.

The calibration program repeatedly took these measurements, and transmitted the correct reading to the PC. The monitoring program on the PC displayed these numbers on the screen as they were received. These were used by the program MAKTAB.BAS to create the look-up table. The look-up table was arranged in increasing order of measured voltage and hence flow rate also. MAKTAB.BAS and the calibration procedure are described in Chapter 5.

Once the calibration table was made by MAKTAB, the calibration program was stopped, and the look-up table was loaded into its memory. Along with the loading of the look-up table, the address of the metering program was written in the location

which determines from which address the controller commences execution. The next time the controller was powered up or reset, the metering program would be run.

Metering

Once a meter was calibrated, the metering program would then be used all the time. The metering program is simply a program that kept the microprocessor "asleep" most of the time, and carried out whatever task had to be done when the processor was "awake".

The processor was put to sleep by the "STOP" command. Once the controller was in STOP mode, it could only be "awakened" by an interrupt. There were two types of interrupts - an ordinary interrupt (IRQ), and a non-maskable interrupt (NMI). These interrupts were generated by the action of briefly grounding the IRQ or NMI pins respectively. The NMI pin was connected to the RTC, while the IRQ pin was connected to the push-button which signalled a request for data transfer. Thus the controller was awakened every 30 seconds by the clock, and at anytime by the push-button.

Two memory locations in RAM were used as flags to indicate the occurrence of interrupts. These locations usually held a value of zero. A non-zero value for a flag indicated that particular interrupt had occurred. When an interrupt occurred, the processor, whether already running or in the STOP mode, would jump to a certain address depending on the type of interrupt. The routines at these locations are called interrupt routines. Each interrupt routine simply incremented the value of the associated flag to indicate the type of interrupt that had occurred.

Once the controller was "awake", the main metering program simply checked these two flags. If neither of the flags held a non-zero value, then the program went on to a STOP command, and the processor was put to "sleep". On being "awakened", the cycle was repeated.

The program first checked the flag associated with the NMI. If this was found to be non-zero, it realized that an NMI had occurred, and it executed the corresponding subroutine. When it returned from that subroutine, it reset the NMI flag to zero. It then checked the IRQ flag - a corresponding subroutine was run if the flag was non-zero, and the flag would be reset on completion. The program then checked both flags again to ensure that they were zero. This was to ensure that an interrupt was not missed even if it occurred while the subroutine associated with the other interrupt was being run.

When both flags were found to be zero, the program went on to a STOP command, and the processor was put to "sleep". On being "awakened", the cycle was repeated.

The NMI was generated by the clock to indicate that it was time to take measurements. When the program noticed the NMI flag set, it called the measurement subroutine. The IRQ was generated by the push-button requesting a data transfer, and when the IRQ flag was set, the communication subroutine was called.

The measurement subroutine

The first thing the measurement subroutine did was to switch on the measurement circuits. Then it incremented the time count by one, where each unit represented 30 seconds. The time counter is stored in three bytes, so the meter can maintain correct time for more than 15 years before the counter turned over to zero. The subroutine then calculates the flow rate from the reading taken during the previous cycle.

The previous reading (one byte) was retrieved from where it was stored. Then the look-up table - which was arranged in ascending order of measured voltages - was scanned to find the first entry greater than this value. This value thus lay between this entry and the previous one. The subroutine then calculated the ratio of the distance of the reading from the two entries. The two corresponding flow rates were retrieved, and the flow rate between these that corresponded to the same ratio was found. This flow rate was calculated with a precision of two hexadecimal digits after the hexadecimal point (to the nearest 1/256th). This flow rate was stored in memory as two bytes - one representing the integer part and the other the fractional part.

The flow rate dimensions are gallons per minute, so the quantity was divided by two to get the volume for 30 seconds. This was added to the counter that kept track of the total quantity of water. This counter occupied four bytes, of which one byte stored the fractional part, so the counter could record a total quantity of 2^{24} gallons - a number the meter was never likely to reach.

The time taken to calculate the flow rate and update the total quantity counter was still less than that required for the measuring circuits to stabilize. A small delay loop was necessary after this to consume that much time.

The subroutine then started on the voltage measurement task. The amplifier output voltage was read 256 times in succession and the average taken. This value was stored at a certain

memory location to be processed in the next cycle. The subroutine then ended, returning control to the main program.

The communication subroutine

To read the meter, a user connects a serial cable between a PC computer and the meter, starts up the meter-reading program on the PC, and then presses the push-button on the meter. This signals an IRQ to the controller, and in two seconds, the communication subroutine is called.

All the counters, flags, and status indicators were grouped together in a small part of RAM. The communication subroutine simply transmitted all the contents of this block of memory to the PC. The meter-reading program, receiving the bytes in a certain order, would calculate all the required numbers like flow rate, quantity metered, and display them in an appropriate format. The other data bytes, which represent flags, status indicators, etc. were also displayed by the DOWNLOAD program. The program also adds the current reading to a file containing previous readings of that particular meter, if asked to do so.

The communication between the meter and the PC was carried out quite simply. The meter transmitted the bytes one by one, and the PC acknowledged each byte by echoing it back to the meter as soon as it received it. When the desired bytes were transmitted, the subroutine returned control to the main program.

It is possible for reasons like improper connection, error in transmission, etc. that proper data transfer may not take place. Special precautions were taken to ensure that this did not cause the controller to hang endlessly in the communication subroutine. If the NMI interrupt, which occurred every 30 seconds, took place twice when the program was in the communication subroutine, the program automatically resets itself and the metering program starts again. However no data is lost - only the flags are reset. In this way the communication subroutine was given a minimum of 30 seconds, and a maximum of 60 seconds to complete the data transfer. If it was not completed within that time, the microcontroller was reset.

Whenever the count of the number of NMIs that were pending reached two, the program recognized this as an unusual and error situation, and promptly reset the meter. This was to ensure that the microcontroller never, under any circumstances, got stuck in any part of the program.

The casing

The meter circuit board measured 10" x 5.5". The casings of this size of the type that are usually used for electronic equipment are very expensive - costing over $25 each. The casing was required only to provide a good covering to the circuit board and proper mounting for it and its switches. A search of ordinary containers resulted in many inexpensive containers that could be used to hold the circuit board. Out of these, a fishing tackle box of dimensions 13" x 6" x 5.25" with a handle for easy transportation was found to be very suitable. The switches were mounted on the circuit board itself. The batteries were simply placed at the bottom of the box, and the circuit board was placed on top, wedged between the two sides so that it was firmly mounted; however, the board can easily be lifted out of the box for examination at any time.

Chapter 4

LABORATORY TESTING OF THE METER

The prototype meters were first tested in the laboratory. Tests were conducted to verify the following:

Proper working of the microcontroller
Correct timing by the clock
Proper working by the communication system
Accuracy of the meter
Capability of the components to function properly for a long time (water proofing, etc.).

Calibration

A setup of the calibration apparatus is shown in Figure 5. Unfiltered water was not available in the laboratory, so

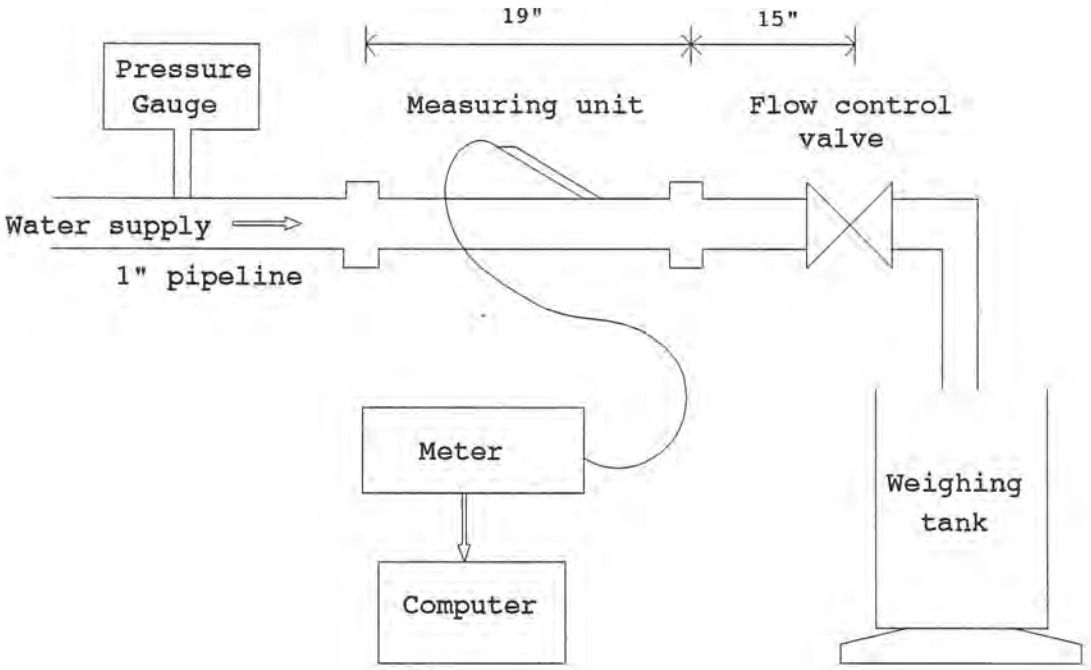

Figure 5. The laboratory calibration setup

ordinary municipal water supply was used. A 1" diameter flexible pipe supplied water to the meter. The meter (measuring unit) was connected to a straight length of 1" pipe on both ends, so that the flow inside the measuring unit would be steady and not affected by turbulence and transients caused at bends, joints, valves, etc. A downstream flow control valve was used to get different rates of flow through the meter. The valve discharged water into a weighing tank.

Calibration involved finding points on the calibration curve, i.e., determining amplified voltages and their corresponding flow rates. The calibration program of the microcontroller continuously read in the amplified voltage and displayed it on the computer screen. A weighing tank and a stopwatch were used to determine the actual flow rate through the meter. This was done by noting the time taken for the weight of water in the weighing tank to increase by a certain amount, e.i., 30 lbs. Since one gallon of water weighs 8.342 lbs, the flow rate was given by:

Flow rate = water weight(lbs)x7.19033(gal./lb)x time(seconds)
in gallons/minute)

First the zero flow reading was noted. Then the flow was increased by a small amount. When the number displayed by the computer reached a constant value, the weights on the scale of the weighing tank were set to a value slightly higher than the present weight. As the weighing tank filled, the water weight would equal the weight on the scale, and a balance would be reached. At this point the stopwatch was started. As the weight of water in the tank continued to increase due to the inflow, the scales would tilt, indicating the tank had become heavier. The weight on the scales was increased by a certain amount, say 30 lbs. When the scales became level again, the stopwatch was stopped, and the time shown indicated the time taken for 30 lbs of water to flow through the meter. The flow rate could then be calculated.

The constant number on the computer screen was the voltage read by the microcontroller. These two quantities gave one point on the calibration curve, and constituted a single reading. The flow rate was then increased using the flow control valve, and this procedure was repeated to get eight to 12 points on the curve. A calibration curve that was obtained for the strain gauge meter is shown in Figure 6.

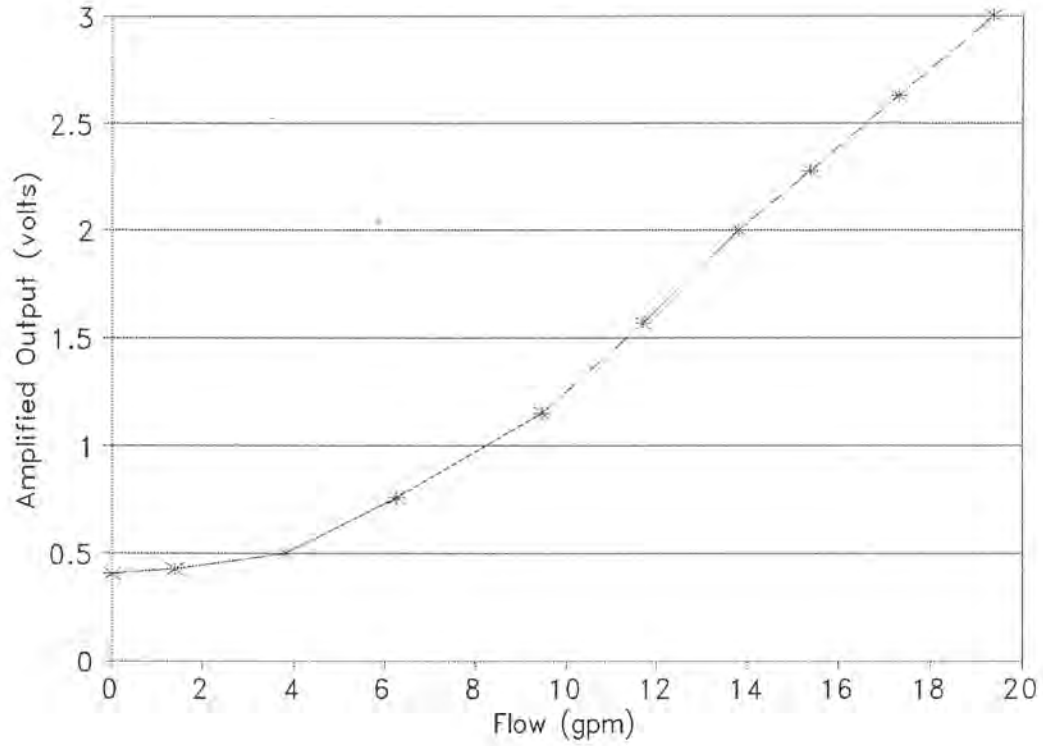

Figure 6. A calibration curve for the strain gauge meter

Creating the look-up table

The microcontroller used the binary number system. For convenience, the hexadecimal system was used for programming and data representation. Each entry in the look-up table contained an integer between 0 and 255 to represent the digital voltage, and a real number to represent the flow rate which would be between 0 and 30. It would ordinarily take more than one byte to represent a real number, but by representing the integer part of flow rate*8 instead of the flow rate, it can be represented in one byte, with a resolution of 1/8 or 0.0125. Thus each entry in the look-up table contained one byte with the digital voltage, and one byte containing flow rate*8.

A program MAKTAB.BAS (given in Appendix B) does all the work necessary to create a look-up table. Given the weight of water

that went into the tank, the time required, and the voltage measured by the microcontroller, MAKTAB did some interpolation to get the flow rates to the nearest 1/8th, and created the look-up table in a file suitable to be loaded directly into the microcontroller memory. By default, MAKTAB also included the address of the metering program in the file, so that when this file was loaded into the controller's memory, the controller was automatically switched into metering mode the next time the unit was powered up.

Testing

Initially, the meter unit was tested without connecting the measuring unit. By creating a small imbalance in the bridge, a small voltage signal was fed to the amplifier. The output voltage of the amplifier was measured to check if the amplifier was performing properly. A dummy calibration table was already loaded into the controller's memory. The device was connected to the PC, so that the meter could be read any time. Given the voltage input from the amplifier, and the contents of the look-up table, the meter could be checked by taking a reading at regular intervals to determine that correct flow rate and total volume were being calculated. The proper working of the communication system on the meter was also checked this way.

The calibration of the meter also needed to be checked, to see if the measuring circuits had good precision, i.e., if the meter, once calibrated, held the same calibration from then on. While testing the orifice meter, it was discovered that the calibration was not holding. On checking the output from the pressure transducer, it was found that for the same flow rate, the output differed from a previous reading.

Orifice Meter Pressure Transducer

The differential pressure transducer was tested separately, with an independent, regulated power supply. The transducer was connected across the orifice and the consistency of the transducer itself was tested, by determining its calibration curve at different line pressures. It was found that the calibration curves were different for different pressures. The flow rate vs. output graph for 60 psi is shown in Figure 7, while that for 90 psi is shown in Figure 8.

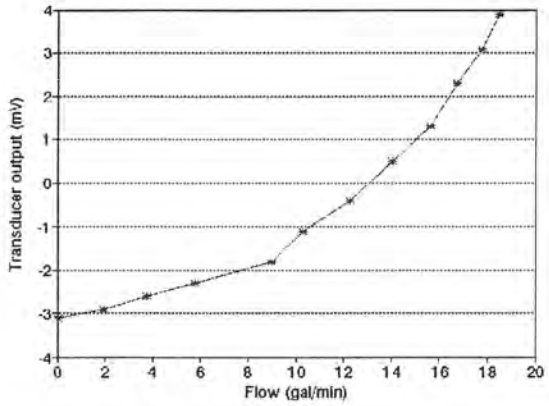

Figure 7. Transducer characteristics at 60 psi

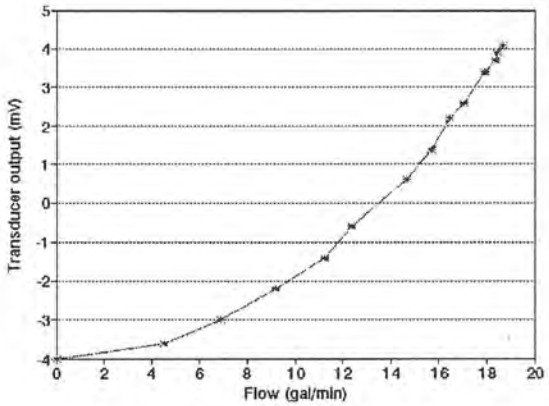

Figure 8. Transducer characteristics at 90 psi

This difference in calibration made the transducer unusable for the meter. No other low-cost pressure transducer could be found that could work under line pressures like 60 psi, and still measure pressure differences less that one psi accurately (some transducers were found that met these specifications, but they were designed to work with gasses only). Therefore research with the orifice meter was stopped, and the project proceeded using only the strain gauge technology.

Strain Gauge Calibration:
The strain gauge was found to be quite consistent - the calibration curve did not change. However it was noticed that the probe showed some signs of inertia. Often, it took more than a minute for the steady output of the gauge to be reached. From a condition of "no flow", when a certain flow rate was initiated, the output of the gauge was not exactly its steady output - some value very close, but less than that value. The output then increased very slowly until the steady value was reached in a minute or two. After that, the output was constant if the flow rate was not changed. A similar inertial effect was noted when flow rate was decreased. The output at first was greater than its steady state value, but the output then decreased to that value in a minute or two. However, there was no signs of hysteresis - for a particular flow rate, the steady output was the same, whether that flow rate was reached by increasing or decreasing the flow rate.

A possible explanation for the inertia of the probe was that the layers of silicon rubber water proofing around the steel strip were somewhat plastic in their characteristics, and took a little time to deform into a new position. This was not a problem because aside from the time delay, the final output of the probe was always consistent with previous steady outputs for the same flow. The inertia would lead to a slightly inaccurate reading for one or two minutes, after which the readings would be accurate.

The strain gauge meter was installed in a pipeline and water was run through the pipe for extended durations for several days to check if the calibration and the water-proofing deteriorated. It was found that the water proofing was not damaged in any way, and the calibration remained unchanged for the entire duration of the test, even under different pressure conditions. There was every indication that the probe and the meter would function properly for several months without problems.

Summary: The testing of the meters in the laboratory showed that the orifice meter could not be used as a meter because of a fatal defect in the pressure transducer. The strain gauge meter was found to retain its calibration for a long time

without problems. It was noticed that the probe possessed some inertia, but this was not serious and did not affect the ability of the strain gauge meter to function properly in its intended role.

Chapter 5

INSTALLATION, MONITORING, AND MODIFICATIONS IN THE FIELD

For field testing of the strain gauge meter, several meters were built and installed in various locations in Cache, Weber and Utah counties. The intention was to install the meters and monitor them for the duration of one irrigation season, from the beginning of May to the end of September. A total of nine meters were installed - two in Cache Valley, four in North Ogden, and three in Payson City. These locations represented a wide variety of water qualities (all unfiltered). Each meter was given a number between zero and nine. The meters and the location of their installation are listed in Table 1.

Table 1: Meter locations

Meter no.	Location	Date installed	Comment
0	3636W 2600N (Ogden)	6/10/94	In lawn
1	Smithfield (Cache Valley)	5/17/94	On slope
2	1325W Pleasant View (Ogden)	6/10/94	Valve box exposed
3	901E 2600N (Ogden)	6/10/94	Exposed, land dry
4	1616 Farr West (Ogden)	6/10/94	In lawn
5	North Logan (Cache Valley)	5/27/94	Exposed, pipe on surface
6	358S 600W (Payson)	6/14/94	Dry, not much water used
8	Next to above (Payson)	6/14/94	Dry land
9	245S 500E (Payson)	6/14/94	In wet lawn Lot of water use

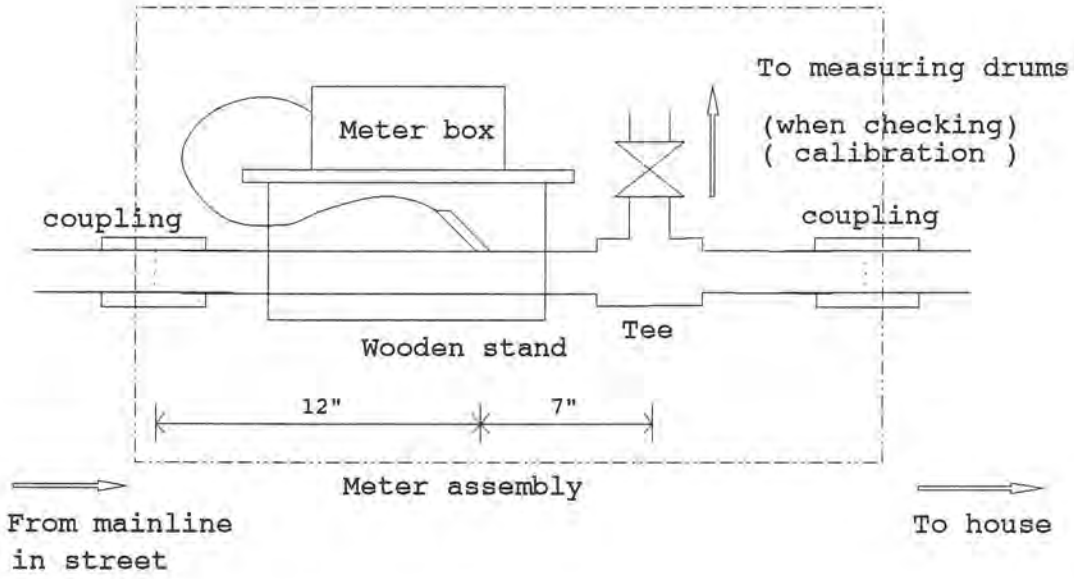

Figure 9. Meter field installation.

A diagram of the installation setup is shown in Figure 9. The supply line leading to a house had to be cut at some place to insert the measuring unit in the line. To facilitate field testing of the meter calibration, a Tee was provided immediately downstream from the meter. A valve was placed on the Tee outlet so that by connecting a hose to the valve, water could be drawn from the pipeline into measuring containers. The measuring containers used were two 55-gallon drums connected with a three inch diameter pipe to form a 100-gallon measuring unit. To check the calibration and accuracy of the meter, all the water outlets on that pipeline were shut down, and the reading on the meter was noted. A hose was then connected to the valve and run to the measuring drums. Then the valve was opened and water was run to the drums until the 100 gallon levels, marked on the drums, was reached. The meter was then read again, and by comparing the difference between the readings to the 100 gallons of water through the hose into the measuring drums, the calibration and accuracy of the meter could be checked. The measuring unit and the tee and valve were put together in the laboratory, so that installation simply meant cutting off a length of pipe in the line and inserting this assembly. The service lines were

Figure 10. Meter pipe insert and strain gauge field installation.

Figure 11. Electronic package and base.

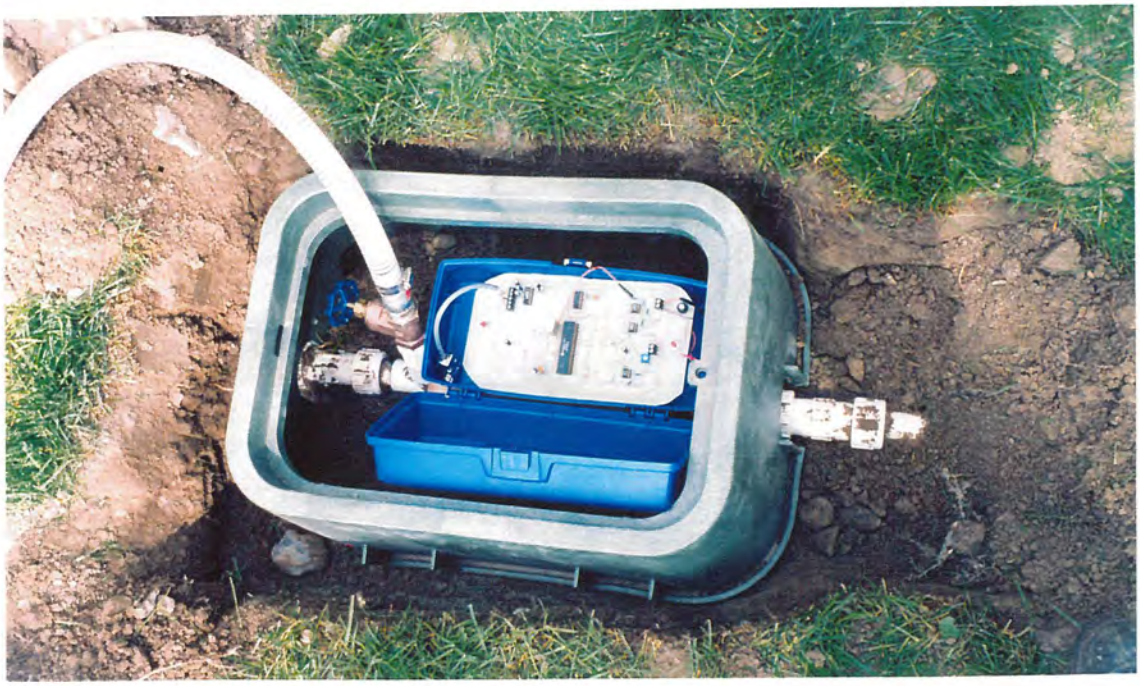

Figure 12. Completed installation showing hose for field calibration test.

Figure 13. Barrels used for field calibration check.

usually buried about one foot underground, so some amount of soil had to be excavated. Once a large enough hole was dug and the measuring assembly installed, a meter box was installed around the unit, and buried in the ground to such an extent that the top of the meter box was almost flush with the ground level. A wooden stand was made to support the electronic package without touching the gauge assembly.

Meter data was obtained by opening the large meter box, opening the small electronic package box, and connecting a lap top computer to a serial port cable.

Field Installation Description:
Initially, two meters were built, and installed in Cache Valley -- one in Smithfield and one in North Logan. The one in Smithfield was installed much deeper than the others, and over irrigation caused the valve box to be totally submerged in water. The meter box was not water-proof, so water got inside the box and corroded the electronic components. That meter was no longer usable.

Four meters were installed in the Weber County (all in systems of the Pine View Water Users); and three were installed in Payson City dual systems.

Temperature Sensitivity:
When the meter was being designed and tested in the laboratory, the assumption was that the meter box and the depth beneath the surface would insulate the meter to a great extent from the range of variations of air temperature. However, most of the service lines were not as deep as expected, and some were on the ground surface. In all except the Smithfield meter, the temperature insulation was not adequate, and the meter itself was subjected to much significant diurnal variations in temperature.

The meter in North Logan was read frequently and it was determined that the output of the amplifier varied significantly with temperature. The amplified output decreased with increase in temperature, and increased with decrease in temperature. The meter was removed and taken back to the laboratory, where the behavior of the amplifier circuits when exposed to different temperatures was studied. This was done by using a hand-held dryer to heat certain parts selectively, and using a larger heater to heat the entire meter box. It was determined that the output of the bridge itself was varying with temperature. The bridge was made of three resistors and one strain gauge. The strain gauge had different temperature characteristics compared to the resistors; so to balance the thermal characteristics of the

bridge, one of the resistor adjacent to the strain gauge was replaced by another strain gauge with the same thermal characteristics as the original. This gauge was attached to the body of the probe. For ideal temperature compensation, the two probes should be at exactly the same temperature, but it was found that the resistance of the strain gauges did not change much with temperature, and the temperature compensation was quite good despite the two gauges being at slightly different temperatures.

This improved the temperature stability of the measuring circuit significantly, but some variation of the amplified output with temperature continued. Subjecting the probe to different temperatures made little effect on the bridge output, so it was concluded that the probe was stable with respect to temperature. Suspecting the resistors used in the amplifier circuit, these were replaced with higher precision resistors (1% accuracy). This did not lead to any significant improvement. It appeared that the amplifier IC itself was being affected by temperature, or the effect of temperature on each component in the measurement circuit had a net result which was a decrease in output with increase in temperature.

The microcontroller had four analog-to-digital convertors. It appeared logical that if the microcontroller could itself measure the temperature, then it should be able to compensate for temperature effects on the measuring circuit and calculate what the correct amplified output would be. A thermistor was therefore added and used to measure temperature. Keeping the flow-rate constant, the output of the thermistor and that of the amplifier were measured, in an attempt to find a relation between the two. Unfortunately no reliable relation could be found, i.e. on different occasions that the same temperature was reached (with a constant bridge output), the amplified outputs were different. Thus there did not seem to be any way to use a measurement of temperature in the meter box to compensate for the effect of temperature on the measurement circuits. There was nothing wrong with the thermistor - it measured temperature with sufficient accuracy. The reason for this inconsistent temperature effect is unknown. Perhaps it is because of different parts of the board, and different components being heated up to different temperatures.

When it was determined that it would not be possible to compensate properly for temperature, insulating the meter from the varying temperature outside seemed to be the best option. To insulate the circuit board and components, the circuit board was covered by a layer of silicon rubber about 0.5 cm thick. In addition, the valve box was painted white so that much of the solar radiation would be reflected.

There was one accident associated with the silicon rubber covering that was given to the meters. It was first applied to the meter in North Logan and the meter was monitored closely to study the results of the added insulation. Favorable results were being obtained, when it was noticed that the circuit etched on the board and several components started showing signs of corrosion. It was determined that this particular type of silicon rubber used an acidic base that could corrode some metals. In a few days time, the working of the meter began to get affected - the meter would occasionally stop working and would have to be reset. The situation grew worse and the meter had to be abandoned.

A different type of silicon rubber was found, which used an ammonia base and was not corrosive to metals. This material was used on the other meters. There was no reaction between this silicon rubber and the circuit components.

These steps helped, but the meter could not be fully insulated from the large difference in temperature between daytime and nighttime, and the meter could not remain accurate over the full 24 hours. In addition, the calibration was carried out in the laboratory which could have been at a temperature significantly different from that in the field, and in such cases the calibration curve (relation between voltage and flow rate) was no longer the same as in the lab.

In the laboratory, it was determined that the effect of temperature was to shift the entire calibration curve by a certain amount (a translation rather than a distortion). This meant that the amplified output voltage corresponding to zero flow changed according to temperature, but the voltage increment corresponding to a certain flow rate remained the same. Thus if the 'zero flow' voltage could be continuously tracked by the meter, then the difference between the current voltage reading and the 'zero flow' voltage, could be used with the calibration curve to calculate the true flow rate.

The 'zero flow' voltage decreased with increase in temperature. It was maximum at night, and gradually decreased in the morning, until it reached a minimum sometime in the afternoon. The voltage then started increasing, as the temperature dropped during the evening reaching a maximum around early morning.

The 'zero voltage' could not be directly calculated from a measurement of temperature. Hence the meter (the microprocessor, in fact) had to continuously keep track of the value of this voltage 24 hours a day. This tracking was possible when there was no flow in the pipe because the change in voltage due to change in temperature was very small and slow, while the change due to commencement of flow was quite

large and took place abruptly. Therefore, when the flow rate was zero and a very small change in voltage took place, after which the voltage remained constant, the meter recognized this as a shift in the 'zero flow' voltage and updated its memory accordingly. However when there was some flow through the pipe, there was always some fluctuation in the output voltage due to the flow characteristics, and a small change in voltage due to temperature could not be differentiated from the fluctuations.

This implies that the 'zero flow' voltage could be tracked only when there was no flow in the pipe. This was not considered to be a serious problem because residential plots are irrigated for just an hour or two at a time. The 'zero flow' voltage did not change significantly during this interval. Even if there was a small change, once the flow stopped, the voltage returned to a value very close to the 'zero flow' value at commencement of flow, and the meter was programmed to recognize this to be termination of flow, and assign this voltage as the new 'zero flow' voltage. Thus the meter should be able to keep an updated record of the 'zero flow' voltage at all times.

This method of 'zero flow' tracking was programmed into the microprocessor of each meter, and along with the silicon rubber coating and the white paint on the exterior, it was expected that the meter would be able to measure flow rate accurately even under conditions of large temperature changes during a single day, as well as over several days as the season changed.

This method however was not fool-proof. If the ambient temperature decreased by a large amount while water was flowing through the meter, then when the flow is stopped, the actual 'zero flow' voltage would have increased by a significant amount, and there could be a difference between this and the 'zero flow' voltage at the commencement of flow. In that case the meter might not recognize this as termination of flow, but as a reduction in flow rate. This situation may occur if a consumer started irrigating in the afternoon when the temperature was quite high, and continued irrigating until night. In that case, the meter would record some flow rate throughout the night and the next morning, and finally come to zero in the afternoon. This is a worst case example. Anytime water flows through the meter for a long time, i.e., six hours, there will be inaccuracies in the measurement because the 'zero flow' voltage would not remain constant for such a long time. However if the consumer irrigates for only one or two hours a day, as a large number of them do, or if the consumer irrigates for up to two hours at a time, stops the water, and then switches it on again for irrigating some other

area, then the meter should be able to achieve reasonable accuracy.

Each meter was calibrated in the laboratory before being installed in the field. Once installed, the meter was checked at least once for accuracy and to detect any change in calibration.

Chapter 6

RESULTS

A total of nine meters were installed for field testing - two in Cache Valley, four in North Ogden, and three in Payson. One in Cache Valley was destroyed by flooding very soon after installation. The other meter in the valley was inadvertently destroyed sometime in the middle of the season by corrosive silicone, but the meter was working until that time. One in North Ogden did not work due to a fault in the electronic circuit or connections.

All the remaining meters were working until the end of the season. The calibration of all the meters were tested around the middle of the irrigation season. The results of the calibration tests on all the meters are given in Table 2. The results were quite good - all the meters except one were measuring with good accuracy, and their calibration had not changed. The faulty meter (number 0) was measuring much less than the actual flow rate. The measuring unit of this meter was found to be leaking at the place where the probe was installed in the pipe. It was not known whether this affected the low readings.

Table 2: Field Calibration test results

Meter no.	Test date	Metered quantity	Actual quantity
0	6/10/94	99.65	100
0	8/29/94	58.8	100
1	5/17/94	88.57	100
2	6/10/94	100.64	100
2	8/29/94	98.36	100
3	8/29/94	113.93	100
5	5/17/94	99.63	100
6	6/14/94	98.62	100
6	9/1/94	97.39	100
8	6/14/94	98.62	100
8	9/1/94	107.31	100
9	6/14/94	115.10	100
9	9/1/94	103.09	100

The percent error between the volume added to the barrells (labelled actual quantity in the table) and the metered quantities, plus the probability of these quantities being exceeded are displayed in Table 3. Meter number 0 was deleted from Table 3 since it was obviously not functioning properly. When considering the error fractions one should note that all of the differences are not due to meter calibration. Other error sources include the difference between the 100 gallons assumed to be in the barrels and the actual volume discharged between beginning and ending meter readings. A two gallon error could have been common -- resulting in a two percent difference in readings. Another source of error which should not be attributed to the meter calibration is the possible error in timing between 30 second data recording times. A test could have been ended at either the beginning or the end of a 30 second interval. If the flow rate were 20 gpm, the error from this source would vary from zero to as much as ten percent, because 100 gallons would have required a test period of 5 minutes (.5/5 = 10%). A uniform distribution of this random error would suggest an average difference of five percent. Combined with a 2 percent error from barrel filling and levelling, a total 7 % error in addition to any calibration error would result.

Table 3. Field Calibration Error Analysis

Meter	Error Rank	Percent Difference	Excedance Probability
9	1	15.1	.91
3	2	13.9	.82
1	3	11.4	.75
8	4	7.3	.67
9	5	3.1	.58
6	6	2.6	.50
2	7	1.6	.42
6	8	1.4	.33
8	9	1.4	.25
2	10	0.6	.17
5	11	0.4	.08

As shown in Table 3, the total error from all sources was less than 2.6 percent during 50 percent of the tests, and less than

8 percent during 67 percent of the tests. These results indicate that the calibration of the meters was holding, and that the accuracy of the meters during field calibration checks was quite good.

The accuracy of the meters before the silicon rubber covering was put on was badly affected by temperature. Those meters that had some shade over them were less affected than the others, but they did not escape completely the temperature effect.

Once the silicon rubber covering was provided, and the 'zero flow' tracking program was installed, the accuracy of the meter depended only on the meter being able to track the 'zero flow' voltage properly. The results of the calibration test show that the meters held their calibration and were quite accurate as long as the 'zero flow' voltage was being tracked properly. However, even when the calibration was holding, if the 'zero flow' voltage was not tracked properly, the meter readings would be inaccurate.

At the end of the season, the meters were removed from the field and examined. It was found that the probes did not show any signs of damage. The water-proofing on the probes was still in good condition. The battery voltage was satisfactory, and probably good enough to last the full 6 month season.

Summary of Flow Data
The readings are divided into those taken before the temperature compensating program was loaded, and those taken after.

Table 4 : Summary of Meter Readings

Meter no.	Total days	Total qnty metered	Before compensation			After compensation		
			No.of days	Qnty metered	Average (glns/day)	No.of days	Qnty metered	Average (glns/day)
0	115	1679514	0	0	0	115	1679514	14573
2	115	1256641	45	690994	15384	70	565646	8052
3	87	809906	17	56096	3319	70	753809	10750
5	87	734662	48	576676	12025	39	157985	4029
6	130	991658	78	982895	12568	52	8762	169
8	130	1211599	78	253370	3243	52	958228	18472
9	130	397118	78	368903	4724	52	28214	544

The meters were found to be inaccurate before the changes for temperature compensation were made. Those that were installed deeper underground and under some shade fared better than the others. The worst affected meters were those where the pipeline was at ground level and where the sun shone directly on the meter box.

After the compensation program was installed in all the meters, the temperature related error was certainly reduced, but not eliminated, as explained in the previous chapter. The average water consumption in gallons per day calculated from the meter readings was often smaller for the compensated meter than for the uncompensated one. The smaller quantities are more likely to be correct.

Cost

The total of costs of all the meter components was $99.57, as itemized in Table 4.

Table 5 : Itemized costs

Item	Cost ($)
Microprocessor MC68HC811E2	13.51
Voltage regulators(2) MAX663	7.44
Communication IC MAX232	2.55
Real Time Clock MM58274	7.49
Amplifier IC LM358	0.71
Transistors and diodes	1.56
Resistors	4.73
Microprocessor and clock crystals	5.74
Switches and terminals	4.85
IC sockets	8.56
Serial connector and cable	1.28
Alkaline cells and battery holder	5.08
Tackle box	5.96
Pipe	0.48
Brass cylinder	1.78
Circuit board	10.00
Strain gauge	9.32
Dummy gauge	3.00
Caulk and spray paint	5.53
Total :	99.57

All these parts were bought at single-piece prices, i.e. without any bulk discount. If the meter were to be mass produced, the parts' price would certainly be less; however the added costs of fabrication and assembly labor for a mass produced unit are unknown. Assuming a 30 % discount for buying components in large quantities, and using a multiplier of 3.0 times the materials cost for labor and profit, a manufacturer could sell this meter for about $200 per unit.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

The orifice type meter was abandoned after lab testing determined that large pressure variations would have caused serious errors in the meter calibration.

The strain gauge concept for measuring unfiltered water in small diameter pipelines was determined to be a useful approach which can be used to meter dual system service connections.

The principal difficulty encountered during the field tests of the strain gauge device was the drifting of the meter calibration due to diurnal temperature changes in the electronic components (they are, however, unaffected by variations in line pressure).

The temperature problem, was caused by the very shallow trenches (less than one foot) in which most water users install their sprinkling systems, and in which the meters were inserted. That problem could be avoided by installing the meters at the same depth as the main lines (usually three feet of cover) ahead of the private line extensions. If the meters were installed in a new system during construction this would present no difficulties, and the ambient temperature variations should have no measurable effect. Therefore, while the temperature variations caused a great deal of difficulty during this testing program, the problem was due to the need to cut into existing shallow services -- a problem that would not exist for installation in future systems.

The most likely market for this meter is in new systems or in extensions of existing systems. Use in existing customer service connections is likely to be actively resisted because of the history of use of water with zero marginal cost.

Burying the meters three feet will require an additional change in the prototype meter casing. A locally available plastic box which was not watertight was used in the testing program. This was adequate for all of the shallow service line installations; however, the one meter which was buried more than one foot deep resulted in a totally submerged meter and resulting damage to the electronic circuitry. Clearly, for a commercial model of the meter, the electronic components would have to be inside a watertight box.

The cost of the prototype meter components was approximately $100. Purchases in large quantities should be substantially less than this (perhaps 1/3 less), but assembly line type labor cost and profit would add an unknown additional quantity (perhaps as much as three times the materials cost). A rough estimate of a mass produced device would therefore be $200. This is more than the cost of conventional propeller type meters (about $65 for 3/4" meters), but is an order of magnitude less than the cost of currently available meters which are suitable for operation with unfiltered water.

This project demonstrated that the strain gauge type device is a viable approach for metering small diameter service lines with unfiltered water. This technology also appears have both much lower cost than other alternatives, and a sufficiently low energy requirement. Before the device developed here could be successfully mass produced for long term service, some additional development will be required, as discussed in the Recommendations section.

Assuming that the strain gauge devices were installed in a water proof meter box, about three feet below ground, the average error should not exceed 3 percent. Some of the field calibration tests resulted in errors higher than this, however, much of that can be attributed to the short duration of the field test design, and should not occur during longer durations of sprinkler use.

Recommendations for Future Research

1. Some meters should be installed at a depth of three feet and tested to verify that this would eliminate the temperature variation difficulties.

2. If it is considered desirable to also produce a meter suitable for installation at shallower depths, some additional experiments with adding thermistors in different portions of the circuitry could be useful. Also, temperature compensated models are available for some components; however, their use would add significantly to the cost.

3. A NEMA type water and humidity tight box should be used to contain the electronic package. To minimize the size of the box, an effort should be made to further miniaturize the electronic circuitry. The board is currently ten inches long; but that dimension could probably be cut in half by a specialist in this type of design. Other necessary design details which need improvement include:

   a. A method of permanently attaching the NEMA box to the meter pipe.

b. Water proofing the opening in the NEMA box for both the wires from the strain gauge to the box and from the box to the laptop computer connector.

c. A very desirable enhancement would be the addition of a transmitter which would allow remote access to the data - without removing the large meter box lid. This would of course, add to the hardware cost, but would also allow a large decrease the labor cost of meter reading - which represents a large fraction of municipal utility overhead costs. Since such technology is now available for municipal meter reading, the related costs and benefits should be investigated for this application.

References

1. "Identification and assessment of certain water management options for the Wasatch front", Vaughn Hansen Associates, Salt Lake City, UT, June 1985.

2. "The flow and level handbook : vol.28", Omega Engineering Inc., Stamford, Connecticut, 1992.

3. Product literature, Marsh-McBriney Inc., Fredrick, Maryland.

4. Product literature, Global Water, Fair Oaks, California.

5. Stapler M., "Drag-body flowmeter", Instruments & Control Systems, November 1962.

6. Stringham B.L., "Design and calibration of a drag body flow meter for irrigation", Department of Agricultural Engineering, University of Alberta, Canada, December 1987.

7. "The pressure, strain and force handbook : vol.28", Omega Engineering Inc., Stamford, Connecticut, 1992.

8. Alkaline primary cells and batteries data sheet, Duracell U.S.A., Bethel, Connecticut.

9. MAXIM IC data book, MAXIM Integrated Products, 1989.

APPENDIX A

METER.MSM
This is the program that was loaded into the microprocessor, and which controls all the operations of the microprocessor and the meter. This program is listed below.

```
;The complete program has the following structure :
;
;F800   :  CLOCKINT
;F880   :  CALIBRATION
;FA00   :  MAIN PROGRAM incl BOOT SECTION
;FC80   :  LOOKUP TABLE      -      loaded from separate file
;FD00   :  IRQ VECTOR ROUTINE
;FD80   :  XIRQ VECTOR ROUTINE
;
;FFF2   :  IRQ VECTOR
;FFF4   :  XIRQ VECTOR
;FFFE   :  RESET VECTOR

PORTA   EQU 00
PORTB   EQU 04
PORTC   EQU 03
DDRC    EQU 07
PACTL   EQU 26
SPCR    EQU 28
BAUD    EQU 2B
SCCR1   EQU 2C
SCCR2   EQU 2D
SCSR    EQU 2E
SCDR    EQU 2F
SCDAT   EQU 2F
ADCTL   EQU 30
ADR1    EQU 31
ADR2    EQU 32
ADR3    EQU 33
ADR4    EQU 34
BPROT   EQU 35
OPTION  EQU 39
PPROG   EQU 3B
INIT    EQU 3D
CONFIG  EQU 3F

TABLEADD    EQU IFC7E    ; LTABLE = lookup table addr - 2


; ----------  CLOCKINT     ----------

;  This places a routine at F800 which initializes the clock and programs a
;  repeating interrupt of time period decided by the byte at F82C.
;  FC = 5 sec
```

```
; FD = 10 sec
; FE = 30 sec
; FF = 60 sec

; CLOCKINT  =>  F800
; READC     =>  F83C
; WRITEC    =>  F848
; DELAYDO   =>  F854

OFFSET F800

CLOCKINT:

CLOCKSET:
BSET    PACTL,X 80  ; Port A : MSB o/p
BSET    PACTL,X 08
BSET    PORTA,X 80  ; Turn WR off (high)
BSET    PORTA,X 08  ; Turn RD off

LDAA    IFF         ; Port C : all o/p
STAA DDRC,X

LDAA I05
JSR WRITEC

LDAA I21
LDAB I0D
STAA PORTC,X
PUTALL:
JSR WRITEC
ADDA I10
DECB
BNE PUTALL

GODNOSWY:
LDAA I01
JSR     WRITEC          ; start CLOCK

LDAA I03
JSR WRITEC
LDAA    IFE             ; repeating interrupt using 0E byte (30 sec)
JSR WRITEC
LDAA I00
JSR WRITEC

GOREAD:
LDAA IF0
STAA DDRC,X

RTS

READC:
```

```
BCLR    PORTA,X 08
JSR DELAYDO
LDAA    PORTC,X          ; i/p byte in A
BSET    PORTA,X 08
RTS

WRITEC:
STAA PORTC,X
BCLR    PORTA,X 80
JSR DELAYDO
BSET    PORTA,X 80
RTS


DELAYDO:
PSHA
LDAA IFF
LOOPY:
DECA
BNE GODUSNTNOWY
GODUSNTNOWY:
PULA
RTS


; ----------    CALIBRATION    ----------

; This is a calibration program for the strain gauge and it runs the EEPROM
; voltage measurement and interpolation routines giving the same time
; intervals as the normal EEPROM program. It sends the average measured
; voltage to the PC to be used in contructing a lookup table. (FC80).
;
; When the chip is enabled in EEPROM bootup, this is the program it runs
; initially. Once the calibration is obtained, the uC is separated (if nec)
; and the EEPROM LTABLE programmed. The IRQ and RESET vectors are then
; directed to FD00  and  FA00 for normal meter operation.

; It places routine SENDWRTS =>  F8E0
; It has its own  CALIBIRQ  at   F8F0

ORG     0080    ; F880 - CALIBRATION

LDX     I1000
BSET    SPCR,X  20
LDAA    I22
STAA    BAUD,X          ;  976 baud
LDAA    I40
STAA    SCCR1,X
LDAA    I0C
STAA    SCCR2,X
LDY     I00D0
```

```
TYS
BSET   39,X  IA0  ; turn on A/DC charge pump ; IRQ to edge level

;OPACK:  BRCLR   PORTA,X  I01  OPACK   ; wait until C0 = 1 to start

NOP
NOP
NOP

LDAA   I01
STAA   ED          ; at ED  0 => delayed calib, nonzero => fast initial
LDAA   I40         ; IRQ and STOP bits cleared (enabled). XIRQ suntil OFF
TAP              ; A -> P  (sets flags)


SORTDEL:           ; abt 8 sec delay
NOP
NOP
NOP
LDAA   ED          ; get MODE
BNE    GOFORVOLT
BCLR   PORTB,X 03  ; All OFF
LDAA   I0A
SUntilDEL:
PSHA
JSR    SOMDELAY
PULA
DECA
BNE    SUntilDEL

NOP
NOP
NOP

GOFORVOLT:
BSET   PORTB,X 02  ; Bridge ON
JSR    XIRQSTUFF       ; returns avg voltage in F1
BSET   PORTB,X 03  ; COMM IC ON
NOP
NOP
NOP
NOP
NOP
NOP

NOP
NOP
NOP    ;JSR    SOMDELAY
NOP
NOP
NOP    ;JSR    SOMDELAY
NOP
```

```
NOP
NOP    ;JSR    SOMDELAY
NOP
NOP
NOP
LDAA   F1
JSR    SENDWRTS     ; send avg ADC to PC
; BCLR   PORTB,X 03  ; All OFF
JMP    SORTDEL


ORG    00E0         ; SENDWRTS routine at F8E0

SENDWRTS:
BRCLR  PORTA,X 02  SENDWRTS
LDAB   SCSR,X       ; need to read this reg before o/p
STAA   SCDR,X       ; write to SCI
WAITACK:
BRCLR  SCSR,X  20  WAITACK
LDAA   SCDR,X
RTS


ORG    00F0  ; CALIBIRQ  : F8F0 abs
CALIBIRQ:
LDAB   ED          ; get MODE
LDAA   I01
SBA                ; A now NEWMODE = 1 - OLDMODE
STAA   ED          ; store NEWMODE
RTI



;      ---------------  MAIN METER PROGRAM  -----------------

; Uses READC at F83A (put by CLOCKINT) to clear CLOCK int.
; CLOCKINT sets repeated interrupts : addr = F800

; CLOCKINT   EQU F800
; READC      EQU F83C
; DELAYDO    EQU F854


;The complete program has the following structure :
;
;F800   :  CLOCKINT
;F880   :  CALIBRATION
;FA00   :  MAIN PROGRAM
;FC80   :  LOOKUP TABLE     -     loaded from separate file
;FD00   :  IRQ VECTOR ROUTINE
;FD80   :  XIRQ VECTOR ROUTINE
```

```
;
;FFF2   :  IRQ VECTOR
;FFF4   :  XIRQ VECTOR
;FFFE   :  RESET VECTOR - initially F880, then FA00


; TABLEADD    EQU IFC7E    ; LTABLE = lookup table addr - 2

ORG    0200    ; OFFSET FA00


JMP ARAMBH

ARAMBH:
LDX I1000
BSET   SPCR,X  20
LDAA I22
STAA   BAUD,X        ;  976 baud
LDAA I40
STAA SCCR1,X
LDAA I0C
STAA SCCR2,X
LDY I00D0
TYS

NOP
NOP
NOP
NOP
NOP

BSET   39,X   IA0  ; turn on A/DC charge pump ; IRQ to edge level

BCLR   PORTB,X I03 ;  turn OFF S.G.REG and COMM IC

JSR    SEEIFFF    ; see if RESET or POWERUP and do stuff

JSR CLOCKINT

LDAA   IFF        ; pattern to get reqd byte address on o/p
STAA FF
DECA
STAA FE
DECA
STAA FD
DECA
STAA FC

NOP
NOP
NOP
NOP
```

```
HEREGOES:
LDAA   FB        ;  XIRQ
BEQ NOHAPP
BSET   PORTB,X 02  ;  turn ON s.g.reg
NOP
NOP
NOP
LDAA I00
JSR READC
NOP
NOP
NOP
JSR XIRQSTUFF
BCLR   PORTB,X 02  ;  turn OFF s.g.reg
NOP
NOP
NOP
LDAA I00
STAA FB

NOP
NOP
NOP
NOP

NOHAPP:
LDAA   FA        ;      IRQ
BEQ NOATALL

BSET   PORTB,X 03  ;  ON both s.g.reg & 2N2222

NOP
NOP
NOP

JSR PUTOUT
LDAA I00
STAA FA
BCLR   PORTB,X 03  ;  OFF

NOP
NOP
NOP

NOATALL:
LDAA FA
ADDA FB
BNE HEREGOES

LDAA I00
JSR READC
LDAA I00
```

```
STAA    DDRC,X      ; C = i/p only
BCLR    PACTL,X 08  ; set A pins i/p only
BCLR    PACTL,X 80
NOP     ;TPA
NOP     ;ANDA I2F
TAP
STOP
BSET    PACTL,X 08  ; set A pins o/p only
BSET    PACTL,X 80
LDAA IF0
STAA    DDRC,X      ; i/p & o/p

JMP HEREGOES

NOP
NOP
NOP
NOP
NOP

SOMDELAY:
LDY I1500
DELAYPOO:
JSR DELAYDO
DEY
BNE DELAYPOO
RTS

NOP
NOP
NOP
NOP
NOP

PUTOUT:             ;  send out 00FF to 00EB
PSHY
JSR SOMDELAY
JSR SOMDELAY
JSR    SOMDELAY     ;  COMM IC needs short time to be OK
LDAA SCDAT,X
LDAA SCSR,X
LDY I0015
KEEPON:
LDAA EA,Y
STAA SCDAT,X
WAITIP:
TPA
ANDA    IBF         ;  put XIRQ ON
TAP
BRCLR   SCSR,X 20   WAITIP
LDAA SCDAT,X
DEY
```

```
BNE KEEPON
PULY
RTS

NOP
NOP
NOP
NOP
NOP
NOP

XIRQSTUFF:
INCTIME:
INC    00F9
BNE TIMEDONE
INC    00F8
BNE TIMEDONE
INC    00F7
NOP
NOP
NOP
NOP
NOP
                    ;BNE    TIMEDONE
                    ;INC    00F6

TIMEDONE:
NOP
NOP
NOP
NOP
NOP

PREVCALC:       ;   calc of qnty for previous ADC rding to save time
                        ;   and take up time reqd for bridge to stabilize

LDAA    F1      ; where prev. ADC reading is stored
LDX     TABLEADD    ; address of lookup table - 2

TESTZERO:
CMPA    03,X
BHS     SEARCH      ; FLOW > 0
LDAA    03,X        ; else FLOW = 0 eqlnt

SEARCH:
INX
INX
CMPA    03,X
BHS     SEARCH      ; search until ADC < table value

NOP
NOP
```

```
NOP
NOP
NOP

INTERPOLATE:
SUBA   01,X       ; A = diff from largest entry < ADC
LDAB   03,X
SUBB   01,X       ; B = diff between the 2 entries enclosing ADC
PSHX
PSHB
LDAB   I00        ; now D = A*100H
PSHB              ; setting up X (divisor)
PULX
IDIV
XGDX              ; A=0 B=fraction of ADC in enclosure (*100)

PULX              ; back to table entry
LDAA   02,X       ; gal/min value
SUBA   00,X       ; step size in gal/min
MUL               ; gets interpolated addition in gal/min
ADDA   00,X       ; add base gal/min

; LDX   I1000     ; X back in action -> done in sub

JSR   ADJUSTD     ; see if gauge inertia, etc & div by 8

STD    EE      ; measure of flow in GAL/MIN in EE,EF

LSRD              ; D = D/2 = gallons in 30 seconds =  reqd qnty
                  ; D actually has integer + 2 HEX decimal (!!!) digits
                  ; A=INT part, B=FRAC part. L.C.=1/256=0.00391 gallons
                  ; GALLONS = A + B / 256
ADDD   F4         ; accumulate in memory
STD    F4
BCC    DONEIPOL   ; here doing 4 byte addition - ADDD to F2-F5
INC    00F3
BNE DONEIPOL
INC    00F2

DONEIPOL:

;NOP
;NOP
;NOP
;NOP

LDAA   F1
STAA   D3         ; save prev rding

NOP
NOP
NOP
```

```
NOP
NOP
NOP

LDAA    I03     ; Delay to give s.g. & AMP abt 2 secs to stabilize
PROPDEL:
PSHA
JSR SOMDELAY
PULA
DECA
BNE PROPDEL

NOP
NOP
NOP

MEASURE:            ; get 256 samples and take average
LDY I0000
LDAA I00

STARTOFF: BCLR  30,X    I3F
WAITING:    BRCLR   30,X I80 WAITING

LDAB 31,X
ABY

DECA
BNE STARTOFF

XGDY                ; D = sum of 256 samples => A = sum/256 = mean

STAA    F1          ; put ADC in F1 to be processes next session

NOP
NOP
NOP

RTS

NOP
NOP
NOP
NOP
NOP
NOP
NOP

SEEIFFF:
LDAA IFE
CMPA FE
BNE RAMNOTOK
DECA
```

```
CMPA FD
BNE RAMNOTOK
DECA
CMPA FC
BNE    RAMNOTOK        ;  00FE-00FC shd be FE,FD,FC if RESET
LDAA FB
NOP
NOP
CMPA    I04        ;  FB shd be < 4
BCC RAMNOTOK
LDAA    ED         ;  MODE byte - shd be 0
BNE RAMNOTOK
NOP
NOP
NOP
NOP
NOP
RAMFINE:
STAA    FA         ;  0 -> FA
STAA    FB         ;  0 -> FB - cancelling any pending interrupts ?
RTS
NOP
NOP
NOP
NOP

RAMNOTOK:
LDAA I00
LDY I002E
MAKEZERO:          ;  fill FF - D2 with 0
STAA D1,Y
DEY
BNE MAKEZERO
RTS

NOP
NOP
NOP
NOP
NOP


ADJUSTD:
LDX    I1000       ; X back in action
PSHB
PSHA               ; D -> stack
LDAA    F1          ; ADC
SUBA    I06         ; nearness to "zero"
CMPA    FC81        ; cmp with actual zero ADC
BHI    ACTUALRD     ; much > 0

ISNEARNIL:
```

```
ADDA    I0A         ; min range to drop = 0A-04
SUBA    D3          ; sub prev ADC
BCC     WASNEARNIL  ; prev rding also near 0

JUSTCHNGD:
LDAA    I09         ; no of cycles small ADCs will be ignored
STAA    D4
BRA     CUTOOZ

WASNEARNIL:
LDAA    D4          ; D4=0 if no ooze detected earlier
BEQ     ACTUALRD

JUSTOOZ:
DEC     00D4        ; one more cyle gone thru

CUTOOZ:
PULB
PULB                ; just to set stack OK
LDAA    I00
TAB                 ; now D = 0
RTS

ACTUALRD:
LDAA    I00
STAA    D4          ; just clearing up D4
PULA
PULB                ; D = D as measurement OK
LSRD
LSRD
LSRD                ; divide by 8 (because calib table has gal/min * 8)
RTS


ORG     0480    ; abs FC80 - dummy lookup table until calibration
DW I000A        ; The table can be any length, but must have FF as last ADC
DW I0214
DW I0520
DW I0A30
DW I0F40
DW I1450
DW I1960
DW I1AFF


ORG     0500    ; IRQ  : FD00 abs
INC     00FA
RTI

ORG     0580        ; XIRQ  : FD80 abs
PULA
ORAA I40
```

```
PSHA
LDAA FB
BEQ    LIFEOK     ; no pending XIRQ => RS232 not stuck
LDX    IFDF0      ; otherwise need to reroute : X = addr of dummy RTS
TSY
STX    07,Y       ; make return addr = FDF0
LIFEOK:
INC    00FB
LDX    I1000      ; could be changed for DIV, etc.
LDAA IF0
STAA   DDRC,X     ; i/p & o/p
LDAA I00
JSR READC
LDAA I00
JSR READC
                          ; JSR    SOMDELAY
NOP
NOP
NOP
LDAA I00
STAA   DDRC,X     ; C = i/p only
RTI

ORG    05F0       ; abs FDE0, comes here if XIRQs pending too long
INC    00EC       ; inc count of RESETs
JMP    FA00       ; eqlnt to RESET


;ORG  abs FFF2     ;  IRQ vector
ORG    07F2
DW     IF8F0      ; initially F8F0 for calib, normal = FD00

;ORG abs  FFF4     ;  XIRQ vector
ORG    07F4
DW     IFD80

;ORG abs  FFFE     ;  RESET vector
ORG    07FE
DW     IF880      ; initially F880 for calibration, then FA00 for normal op


; For CALIB,     IRQ -> F8F0  and  RESET -> F880

; For NORMAL op,  IRQ -> FD00  and  RESET -> FA00
```

APPENDIX B

Program listing of the PC programs
A PC was used to program and monitor the meter during testing, calibrate the meter prior to installation, and read the meter when it was installed in the field. The programs that were written for the PC to carry out all these functions are listed in this appendix.

DOWNLOAD.BAS : Program to read the meter, display important information on the screen, and store the data on disk.

```
DECLARE FUNCTION ROUND! (P AS DOUBLE)
DECLARE SUB SEND (CHAR%)
DECLARE SUB SENDACK (CHAR%)
DEFINT A-M, X-Z
DEFDBL P, R
DIM CODE(256), INFO(17), PINFO(17)
COMMON SHARED X, Z, CHAR, DPORT, LSTAT, N
H$ = "&H"


DPORT = &H3F8
LCON = &H3FB
MCON = &H3FC
LSTAT = &H3FD
MSTAT = &H3FE

'BAUD = 2400
BAUD = 976
DVR = 115200 / BAUD

X = &H80
OUT LCON, X          ' access baud divider reg
OUT DPORT, DVR         ' low byte
OUT DPORT + 1, 0       ' high byte

X = 3                ' 00000011 = no bk, no prty, 1 stop, 8 bit data
OUT LCON, X
OUT MCON, X            ' 3 = turn DTR and RTS on (+ve => 0 for MPU)


PRINT "Connect the PC serial port to the meter and                [ Press  Q ]"
PRINT "press the meter's COMM button (next to the meter cable)       [ to quit  ]"
PRINT
DO WHILE (((INP(LSTAT) AND 1) = 0) AND INKEY$ <> "Q" AND INKEY$ <> "q")
LOOP

DO
   Z = INP(LSTAT)
```

```
   IF (INT(Z / 2) * 2 <> Z) THEN X = INP(DPORT)
   Z = INP(LSTAT)
LOOP UNTIL (Z = 96)

N = 0
M = 0
ADDR = VAL("&H00FC")   '  252


FOR I = 0 TO 21
   CHAR = I
   SENDACK (CHAR)
   IF N <> 0 THEN I = 300                    ' ABORT !!!!
   ADDR = ADDR - M
   PINFO(ADDR - 235) = X
   IF M = 1 THEN PRINT HEX$(ADDR); "  "; RIGHT$("00" + HEX$(X), 2); " "; X
   IF X = 252 THEN M = 1
   IF ADDR = VAL("&H00EC") THEN M = 0
NEXT I

IF I < 100 THEN                ' not aborted

PGALQ = PINFO(7) * 65536 + PINFO(8) * 256 + PINFO(9) + PINFO(10) / 256
PGALMIN = PINFO(3) + PINFO(4) / 256
PTIME = (PINFO(12) * 65536 + PINFO(13) * 256 + PINFO(14)) * 30
M = INT(PTIME / 86400)
N = PTIME - M * 86400
PRINT
PRINT
PRINT

LOCATE 10, 20: PRINT "Qnty metered (gal)  : "; ROUND(PGALQ)
LOCATE 11, 20: PRINT "Flow rate (gal/min) : "; ROUND(PGALMIN)
LOCATE 12, 20: PRINT "Time measured (sec) : "; PTIME; " ("; M; " days,"; N; " secs )"


IF INSTR(COMMAND$, "/N") = 0 AND INSTR(COMMAND$, "/n") = 0 THEN
LOCATE 22: PRINT "Save data in a file ? (Enter meter no. if yes, -1 if no) ";
INPUT N
ELSE
N = -1
LOCATE 23
END IF

IF N >= 0 THEN
   OPEN "M" + LTRIM$(STR$(N)) + ".DAT" FOR APPEND AS #3
   PRINT #3, VAL(DATE$), MID$(DATE$, 4, 2), LEFT$(TIME$, 5),
   PRINT #3, ROUND(PGALQ), ROUND(PGALMIN), PTIME
   CLOSE #3
END IF

END IF         ' if not aborted
```

```
DEFSNG P, R
FUNCTION ROUND (P AS DOUBLE)
P = (INT(P * 100) / 100)
ROUND = P
END FUNCTION

SUB SEND (CHAR)

OUT DPORT, CHAR

END SUB

SUB SENDACK (CHAR)

Z = INP(LSTAT)
OUT DPORT, CHAR
X = 0
IF CHAR < 20 THEN
   DO
      Z = INP(LSTAT)
      A$ = INKEY$
      X = X + 1
      LOOP WHILE ((INT(Z / 2) * 2 = Z) AND A$ <> "Q" AND A$ <> "q" AND (CHAR < 19
OR X < 200))
   IF A$ = "Q" OR A$ = "q" THEN N = 1          ' ABORT !!!
ELSE
   FOR T = 1 TO 20
      OUT DPORT, CHAR
   NEXT T
   X = INP(DPORT)
END IF

X = INP(DPORT)
'PRINT X, CHAR

END SUB
```

ASM.BAS : Program to assemble MC68HC11 series microprocessor assembly langauge into machine code. The output of this program is in the format that is used by MPU.BAS to load the program into the microprocessor.

```
' Use OFFSET to place address anywhere in memory, e.g. in EEPROM.
' Only one OFFSET can be used in a program - at the beginning.
' If no OFFSET is used, OFFSET = 0 is assumed.
' All ORGs are taken relative to this OFFSET.

' enter <fname> /E for EEPROM assembly, /S to leave out BASE.MSM
' /N does not make .MPN file


DEFINT A-Z
DIM EQUNAM$(100), EQUVAL$(100), ADLOC(100), ADNAM$(100)
DIM FADLOC(100), FADNAM$(100)
DIM CODE(2048), BINST$(145), BBYTE(145), MBYTE(145), PBYTE(145), ABYTE(145)
DIM SBYTE(145), MPN$(1000), MS1$(1000)  ' 1000 lines of code, 2000 opcodes
H$ = "&H"

FILE$ = COMMAND$


WMPN = 1: BSE = 1: EEPROG = 0: OFFSET = 0

IF FILE$ <> "" THEN
  P1 = INSTR(FILE$, "/N")       ' dont make .MPN
  IF P1 <> 0 THEN WMPN = 0: FILE$ = LEFT$(FILE$, P1 - 1) + MID$(FILE$, P1 + 2)
  P1 = INSTR(FILE$, "/S")       ' dont insert in BASE.MSM
  IF P1 <> 0 THEN BSE = 0: FILE$ = LEFT$(FILE$, P1 - 1)
  P1 = INSTR(FILE$, "/E")      ' EEPROM FILE
  IF P1 <> 0 THEN BSE = 0:  EEPROG = 1:  FILE$ = LEFT$(FILE$, P1 - 1)
ELSE
  INPUT "GIVE FILENAME (.MSM assumed) ", FILE$
END IF

FILE$ = RTRIM$(FILE$)
IF EEPROG = 1 THEN TBYTES = 0 ELSE TBYTES = 256

P1 = INSTR(FILE$, ".")
IF P1 = 0 THEN FILE$ = FILE$ + "." ELSE FILE$ = LEFT$(FILE$, P1)

FOR N = 1 TO 145
   BBYTE(N) = 0
   PBYTE(N) = 0
   ABYTE(N) = 0
   SBYTE(N) = 0
NEXT N

' BINST$, BBYTE, PBYTE, ABYTE
```

```
FOR N = 1 TO 145
   READ BINST$(N), B$, C$, D$
   BBYTE(N) = VAL(H$ + B$)
   PBYTE(N) = VAL(H$ + C$)
   ABYTE(N) = VAL(H$ + D$)
   IF ABYTE(N) = -1 THEN SBYTE(N) = 1: ABYTE(N) = 1  ' for sp pbytes nml
   IF ABYTE(N) = -2 THEN SBYTE(N) = 2: ABYTE(N) = 1  ' for sp pbytes spl 1
NEXT N

OPEN FILE$ + "MSM" FOR INPUT AS #1
IF BSE = 1 THEN OPEN "BASE.MSM" FOR INPUT AS #2

NEQU = 0: MS = 0: TL = 0

IF BSE = 1 THEN                    ' get initial part of BASE.MPN
   DO
      TL = TL + 1
      LINE INPUT #2, MS1$(TL)
   LOOP UNTIL (INSTR(MS1$(TL), "--------") <> 0)  ' place to insert
END IF

DO                                 ' read in source file
   TL = TL + 1
   LINE INPUT #1, MS1$(TL)
LOOP UNTIL EOF(1)

IF BSE = 1 THEN                    ' get end part of BASE.MPN
   DO
      TL = TL + 1
      LINE INPUT #2, MS1$(TL)
   LOOP UNTIL EOF(2)
END IF


FOR LINES = 1 TO TL
   A$ = MS1$(LINES)
   IF LEN(A$) > 1 THEN
      P = INSTR(A$, ";"): IF P > 0 THEN A$ = LEFT$(A$, P - 1)
      A$ = LTRIM$(RTRIM$(A$))          ' throw away comments

      P = INSTR(A$, CHR$(9))           ' make TAB into SPACE
      DO WHILE (P <> 0)
         MID$(A$, P, 1) = " "
         P = INSTR(A$, CHR$(9))
      LOOP

      P = INSTR(A$, "  ")              ' delete > 1 spaces
      DO WHILE (P <> 0)
         A$ = LEFT$(A$, P - 1) + MID$(A$, P + 1)
         P = INSTR(A$, "  ")
      LOOP
```

```
      IF INSTR(A$, "EQU") <> 0 THEN
         NEQU = NEQU + 1
         P1 = INSTR(A$, " ")
         EQUNAM$(NEQU) = LEFT$(A$, P1 - 1)
         EQUVAL$(NEQU) = MID$(A$, P1 + 5)
      ELSE
         FOR N = 1 TO NEQU
            P1 = INSTR(A$, EQUNAM$(N))
            IF P1 <> 0 THEN
                        A$ = LEFT$(A$, P1 - 1) + EQUVAL$(N) + MID$(A$, P1 + LEN(EQUNAM$(N)))
               N = NEQU + 1
            END IF
         NEXT N
         A$ = LTRIM$(RTRIM$(A$))
         IF LEN(A$) > 1 THEN
            MS = MS + 1
            MS1$(MS) = A$
         END IF
      END IF
   END IF

NEXT LINES

MSLINE = MS

CLOSE #1: CLOSE #2                    ' PASS 1 over.

OPEN FILE$ + "MCM" FOR OUTPUT AS #2
IF WMPN = 1 THEN OPEN FILE$ + "MPN" FOR OUTPUT AS #3  ' if no /N

PNO = 0: PADDR = 0: FADNO = 0: MS = 0

FOR MS = 1 TO MSLINE
 A$ = MS1$(MS)
 OA$ = A$
 P1 = INSTR(A$, " "): IF P1 = 0 THEN P1 = LEN(A$) + 1
 C1$ = LEFT$(A$, P1 - 1)
 IF RIGHT$(C1$, 1) = ":" THEN
   FADNO = FADNO + 1
   FADNAM$(FADNO) = LEFT$(C1$, P1 - 2)
   FADLOC(FADNO) = PADDR
   A$ = MID$(A$, P1 + 1)
   IF P1 < 7 THEN PRINT "ERROR : Label name < 5 chars"
 END IF

 IF LEN(A$) > 1 THEN
  P1 = INSTR(A$, " "): IF P1 = 0 THEN P1 = LEN(A$) + 1
  C1$ = LEFT$(A$, P1 - 1)
  A$ = MID$(A$, P1 + 1)
  NOBYTE = 0

  IF C1$ = "DB" THEN
```

```
  BBYTE = VAL(H$ + MID$(A$, 2))
  MBYTE = BBYTE
  ABYTE = 0: PBYTE = 0
  A$ = "": C1$ = ""
END IF

IF C1$ = "DW" THEN
  BBYTE = VAL(H$ + MID$(A$, 2, 2))
  MBYTE = BBYTE
  ABYTE = 1: XBYTE(1) = VAL(H$ + MID$(A$, 4))
  A$ = "": C1$ = ""
END IF

IF C1$ = "ORG" THEN
  PADDR = VAL(H$ + A$)
  ABYTE = 0: BBYTE = 0: MBYTE = 0: PBYTE = 0
  NOBYTE = 1
  C1$ = ""
END IF

IF C1$ = "OFFSET" THEN
  OFFSET = VAL(H$ + A$)
  ABYTE = 0: BBYTE = 0: MBYTE = 0: PBYTE = 0
  NOBYTE = 1
  C1$ = ""
END IF


IF LEN(C1$) > 1 THEN
  INST = 0: M = 0
  DO
     INST = INST + 1
     IF BINST$(INST) = C1$ THEN M = -1
  LOOP UNTIL ((BINST$(INST) > C1$) OR (M = -1))

  IF M THEN            'found
    PBYTE = 0: XBYTE(1) = 0: XBYTE(2) = 0: XBYTE(3) = 0: ABYTE = 0
    BBYTE = BBYTE(INST):    MBYTE = BBYTE: SBYTE = SBYTE(INST)
    IF ABYTE(INST) <> 0 THEN                ' addnl bytes
      IF ASC(A$) = 73 OR ASC(A$) = 105 THEN        'imm (leading I or i)
        A$ = MID$(A$, 2)
        ABYTE = LEN(A$) / 2
        XBYTE(1) = VAL(H$ + LEFT$(A$, 2))
        IF ABYTE = 2 THEN XBYTE(2) = VAL(H$ + MID$(A$, 3))

        IF SBYTE <> 0 THEN
          IF SBYTE = 1 THEN PBYTE = &H18
          IF SBYTE = 2 THEN PBYTE = &H1A
        END IF

      ELSE                                  ' not imm
        IF INSTR(A$, ",X") <> 0 THEN                ' ,X
```

```
  IF ABYTE(INST) <= 1 THEN
     MBYTE = &H20: ABYTE = 1
     XBYTE(1) = VAL(H$ + LEFT$(A$, 2))
     IF SBYTE <> 0 THEN PBYTE = &H1A
  ELSE                                  ' sp BR etc cases
     ABYTE = ABYTE(INST)
     PBYTE = 0: MBYTE = &H1C
     A$ = LEFT$(A$, 2) + MID$(A$, 5)    ' remove ,X
     FOR N = 1 TO 2          '--- last one maybe label, READ 2 /-
       IF ASC(A$) = 73 OR ASC(A$) = 105 THEN A$ = MID$(A$, 2) ' i
       XBYTE(N) = VAL(H$ + LEFT$(A$, 2))
       A$ = MID$(A$, 4)
     NEXT N
  END IF
ELSE
  IF INSTR(A$, ",Y") <> 0 THEN              ' ,Y
    IF ABYTE(INST) <= 1 THEN
     PBYTE = PBYTE(INST)
     MBYTE = &H20: ABYTE = 1
     XBYTE(1) = VAL(H$ + LEFT$(A$, 2))
     IF SBYTE <> 0 THEN
       IF SBYTE = 1 THEN PBYTE = &H18
       IF SBYTE = 2 THEN PBYTE = &HCD
     END IF
    ELSE                                'sp case BR etc
     ABYTE = ABYTE(INST)
     PBYTE = &H18: MBYTE = &H1C
     A$ = LEFT$(A$, 2) + MID$(A$, 5)
     FOR N = 1 TO 2          ' ------ maybe label
       IF ASC(A$) = 73 OR ASC(A$) = 105 THEN A$ = MID$(A$, 2) 'i
       XBYTE(N) = VAL(H$ + LEFT$(A$, 2))
       A$ = MID$(A$, 4)
     NEXT N
    END IF
  ELSE
    IF LEN(A$) = 2 OR ABYTE(INST) > 1 THEN      ' dir (incl BR)

     IF ABYTE(INST) <= 1 THEN
       MBYTE = &H10: ABYTE = 1
       XBYTE(1) = VAL(H$ + A$)
       IF SBYTE <> 0 THEN
         IF SBYTE = 1 THEN PBYTE = &H18
         IF SBYTE = 2 THEN PBYTE = &H1A
       END IF
     ELSE                                   ' sp cases
       ABYTE = ABYTE(INST)
       MBYTE = &H10
       FOR N = 1 TO 2          ' ------- maybe label
         IF ASC(A$) = 73 OR ASC(A$) = 105 THEN A$ = MID$(A$, 2)
         XBYTE(N) = VAL(H$ + LEFT$(A$, 2))
         A$ = MID$(A$, 4)
       NEXT N
```

```
          END IF
         ELSE
          IF LEN(A$) = 4 THEN                    ' ext
           MBYTE = &H30: ABYTE = 2
           XBYTE(1) = VAL(H$ + LEFT$(A$, 2))
           XBYTE(2) = VAL(H$ + MID$(A$, 3))
           IF SBYTE <> 0 THEN
             IF SBYTE = 1 THEN PBYTE = &H18
             IF SBYTE = 2 THEN PBYTE = &H1A
           END IF
          ELSE
           IF LEN(A$) > 4 THEN                   ' label => ext
            ADNO = ADNO + 1
            IF ABYTE(INST) <> -3 THEN
               ABYTE = 2: MBYTE = &H30
               ADLOC(ADNO) = PADDR + 1
               ADNAM$(ADNO) = A$
               IF SBYTE <> 0 THEN
                  IF SBYTE = 1 THEN PBYTE = &H18
                  IF SBYTE = 2 THEN PBYTE = &H1A
               END IF
            ELSE                                 ' rel
               ABYTE = 1
               ADLOC(ADNO) = -(PADDR + 1)
               MBYTE = BBYTE
               ADNAM$(ADNO) = A$
            END IF                ' rel / abs
           END IF                 ' LABEL
          END IF                  ' EXT
         END IF                   ' DIR
        END IF                    ' ,Y
       END IF                     ' ,X
      END IF                      ' i
     ELSE                         ' abyte
      MBYTE = BBYTE
      PBYTE = PBYTE(INST)
     END IF                       ' ABYTE
    ELSE
     PRINT "unrecognized mnemonic in "; C1$; " "; A$
    END IF                        '   M

    IF ABYTE = 3 THEN                        ' BR etc cases with R jump
       IF LEN(A$) = 2 THEN
          XBYTE(3) = VAL(H$ + A$)     ' directly offset given
       ELSE
          ADNO = ADNO + 1
          ADNAM$(ADNO) = A$
          ADLOC(ADNO) = -(PADDR + SGN(PBYTE) + 3)    ' ABYTE=3
       END IF
    END IF

  END IF                          ' A$ > 1
```

```
    PN$ = RIGHT$("0000" + HEX$(PADDR + OFFSET), 4) + "   "

    IF PBYTE <> 0 AND NOBYTE = 0 THEN
       CODE(PADDR) = PBYTE: PADDR = PADDR + 1
       PN$ = PN$ + RIGHT$("00" + HEX$(PBYTE), 2) + " "
    END IF

    CODE(PADDR) = ((BBYTE AND &HCF) OR MBYTE)
    PN$ = PN$ + RIGHT$("00" + HEX$(CODE(PADDR)), 2) + " "
    IF NOBYTE = 0 THEN PADDR = PADDR + 1

    FOR N = 1 TO ABYTE
       CODE(PADDR) = XBYTE(N)
       PN$ = PN$ + RIGHT$("00" + HEX$(CODE(PADDR)), 2) + " "
       PADDR = PADDR + 1
    NEXT N

       IF NOBYTE = 0 AND EEPROG = 1 THEN TBYTES = TBYTES + PBYTE + ABYTE + 1

    PN$ = LEFT$(PN$ + SPACE$(30), 25) + OA$

    PNO = PNO + 1: MPN$(PNO) = PN$

'  END IF                ' A$ > 1
 END IF                 ' another A$ > 1

NEXT MS


FOR N = 1 TO ADNO
    C1$ = ADNAM$(N)
    T = 0: M = 0

    DO
      T = T + 1
      IF FADNAM$(T) = C1$ THEN
          M = -1
          P = FADLOC(T) + OFFSET
          IF ADLOC(N) >= 0 THEN          ' abs / rel
             P1 = INT(P / 256)
             P2 = P - P1 * 256
             CODE(ADLOC(N)) = P1
             CODE(ADLOC(N) + 1) = P2
          ELSE                           ' rel
             P1$ = RIGHT$(HEX$(FADLOC(T) + ADLOC(N) - 1), 2)
             P1 = VAL(H$ + P1$)
             CODE(-ADLOC(N)) = P1
          END IF
      END IF
    LOOP UNTIL (M = -1 OR T >= FADNO)
```

```
   IF M = 0 THEN PRINT "label "; C1$; " not found"
NEXT N

IF EEPROG = 0 THEN
 FOR N = 0 TO 255                ' 256 bytes exc for normal RAM
     C1$ = RIGHT$("00" + HEX$(CODE(N)), 2)
     PRINT #2, C1$               ' get it back with INPUT #
 NEXT N
END IF


M = 0
FOR N = 1 TO PNO                 ' filling in for MPN jmp addrs
   A$ = MPN$(N)
   IF INSTR(A$, "ORG") <> 0 OR INSTR(A$, "OFFSET") <> 0 THEN
      M = VAL(H$ + LEFT$(A$, 4)) - OFFSET
   ELSE
      P2 = INSTR(8, A$, "  ")
      P = 8
      DO
       P1 = VAL(H$ + MID$(A$, P))
       IF P1 <> CODE(M) THEN
        IF MID$(A$, P, 2) <> "00" THEN
           PRINT "MPN - MCM mismatch at .MPN line "; N
        END IF
        MID$(A$, P) = RIGHT$("00" + HEX$(CODE(M)), 2)
       END IF
       M = M + 1
       P = P + 3
      LOOP UNTIL P >= P2
   END IF
   MPN$(N) = A$
   IF WMPN = 1 THEN PRINT #3, A$
NEXT N

IF EEPROG = 1 THEN
  FOR N = 1 TO PNO
   A$ = MPN$(N)
   IF INSTR(A$, "ORG") = 0 AND INSTR(A$, "OFFSET") = 0 THEN
     P1 = 8: M = 0
     PADDR = VAL(H$ + LEFT$(A$, 4)) - OFFSET
     DO
      C1$ = RIGHT$("0000" + HEX$(PADDR + OFFSET), 4)
      PN$ = LEFT$(C1$, 2) + " " + RIGHT$(C1$, 2) + " " + MID$(A$, P1, 2)
      P1 = P1 + 3
      PRINT #2, PN$
      PADDR = PADDR + 1
     LOOP UNTIL (MID$(A$, P1, 1) = " ")
   END IF
  NEXT N
END IF
```

```
PRINT TBYTES; " total program bytes"

CLOSE #2: CLOSE #3              ' PASS 2 completed. ALL DONE !!

'  INST, BYTE, PBY, ABY

DATA ABA,1B,0,0
DATA ABX,3A,0,0
DATA ABY,3A,18,0
DATA ADCA,89,18,1
DATA ADCB,C9,18,1
DATA ADDA,8B,18,1
DATA ADDB,CB,18,1
DATA ADDD,C3,18,1
DATA ANDA,84,18,1
DATA ANDB,C4,18,1
DATA ASL,78,18,1
DATA ASLA,48,0,0
DATA ASLB,58,0,0
DATA ASLD,05,0,0
DATA ASR,77,18,1
DATA ASRA,47,0,0
DATA ASRB,57,0,0
DATA BCC,24,0,-3
DATA BCLR,15,18,2          ' ---------
DATA BCS,25,0,-3
DATA BEQ,27,0,-3
DATA BGE,2C,0,-3
DATA BGT,2E,0,-3
DATA BHI,22,0,-3
DATA BHS,24,0,-3
DATA BITA,85,18,1
DATA BITB,C5,18,1
DATA BLE,2F,0,-3
DATA BLO,25,0,-3
DATA BLS,23,0,-3
DATA BLT,2D,0,-3
DATA BMI,2B,0,-3
DATA BNE,26,0,-3
DATA BPL,2A,0,-3
DATA BRA,20,0,-3
DATA BRCLR,13,18,3     '----------------
DATA BRN,21,0,-3
DATA BRSET,12,18,3     '----------
DATA BSET,14,18,2      ' -----------
DATA BSR,8D,0,-3
DATA BVC,28,0,-3
DATA BVS,29,0,-3
DATA CBA,11,0,0
DATA CLC,0C,0,0
```

```
DATA CLI,0E,0,0
DATA CLR,7F,18,1
DATA CLRA,4F,0,0
DATA CLRB,5F,0,0
DATA CLV,0A,0,0
DATA CMPA,81,18,1
DATA CMPB,C1,18,1
DATA COM,73,18,1
DATA COMA,43,0,0
DATA COMB,53,0,0
DATA CPD,83,CD,-2     '--------
DATA CPX,8C,CD,1
DATA CPY,8C,18,-1     '--------
DATA DAA,19,0,0
DATA DEC,7A,18,1
DATA DECA,4A,0,0
DATA DECB,5A,0,0
DATA DES,34,0,0
DATA DEX,09,0,0
DATA DEY,09,18,0      '---------
DATA EORA,88,18,1
DATA EORB,C8,18,1
DATA FDIV,03,0,0
DATA IDIV,02,0,0
DATA INC,7C,18,1
DATA INCA,4C,0,0
DATA INCB,5C,0,0
DATA INS,31,0,0
DATA INX,08,0,0
DATA INY,08,18,0      '-----------
DATA JMP,7E,18,1
DATA JSR,9D,18,1
DATA LDAA,86,18,1
DATA LDAB,C6,18,1
DATA LDD,CC,18,1
DATA LDS,8E,18,1
DATA LDX,CE,CD,1
DATA LDY,CE,18,-1     '----------
DATA LSL,78,18,1
DATA LSA,48,0,0
DATA LSB,58,0,0
DATA LSLD,05,0,0
DATA LSR,74,18,1
DATA LSRA,44,0,0
DATA LSRB,54,0,0
DATA LSRD,04,0,0
DATA MUL,3D,0,0
DATA NEG,70,18,1
DATA NEGA,40,0,0
DATA NEGB,50,0,0
DATA NOP,01,0,0
DATA ORAA,8A,18,1
```

```
DATA ORAB,CA,18,1
DATA PSHA,36,0,0
DATA PSHB,37,0,0
DATA PSHX,3C,0,0
DATA PSHY,3C,18,0     '---------
DATA PULA,32,0,0
DATA PULB,33,0,0
DATA PULX,38,0,0
DATA PULY,38,18,0     ' ----------
DATA ROL,79,18,1
DATA ROLA,49,0,0
DATA ROLB,59,0,0
DATA ROR,76,18,1
DATA RORA,46,0,0
DATA RORB,56,0,0
DATA RTI,3B,0,0
DATA RTS,39,0,0
DATA SBA,10,0,0
DATA SBCA,82,18,1
DATA SBCB,C2,18,1
DATA SEC,0D,0,0
DATA SEI,0F,0,0
DATA SEV,0B,0,0
DATA STAA,97,18,1
DATA STAB,D7,18,1
DATA STD,DD,18,1
DATA STOP,CF,0,0
DATA STS,9F,18,1
DATA STX,DF,CD,1
DATA STY,DF,18,-1      '---------------
DATA SUBA,80,18,1
DATA SUBB,C0,18,1
DATA SUBD,83,18,1
DATA SWI,3F,0,0
DATA TAB,16,0,0
DATA TAP,06,0,0
DATA TBA,17,0,0
DATA TEST,00,0,0
DATA TPA,07,0,0
DATA TST,7D,18,1
DATA TSTA,4D,0,0
DATA TSTB,5D,0,0
DATA TSX,30,0,0
DATA TSY,30,18,0       '----------
DATA TXS,35,0,0
DATA TYS,35,18,0       '-----------
DATA WAI,3E,0,0
DATA XGDX,8F,0,0
DATA XGDY,8F,18,0      '------------
```

CALIB.BAS : Program used when calibrating the meter. It simply displays the quantized voltages being measured by the microprocessor, and this is one of the measurements used to create the look-up table.

```
DEFINT A-M, X-Z
COMMON SHARED X, Z, CHAR, DPORT, LSTAT, N
H$ = "&H"


DPORT = &H3F8
LCON = &H3FB
MCON = &H3FC
LSTAT = &H3FD
MSTAT = &H3FE

'BAUD = 2400
BAUD = 976
DVR = 115200 / BAUD

X = &H80
OUT LCON, X          ' access baud divider reg
OUT DPORT, DVR         ' low byte
OUT DPORT + 1, 0       ' high byte

X = 3                ' 00000011 = no bk, no prty, 1 stop, 8 bit data
OUT LCON, X
OUT MCON, X            ' 3 = turn DTR and RTS on (+ve => 0 for MPU)

DO
   Z = INP(LSTAT)
   IF (INT(Z / 2) * 2 <> Z) THEN X = INP(DPORT)
   Z = INP(LSTAT)
LOOP UNTIL (Z = 96)

OUT MCON, 1                  ' turn XXXDTR, RTS off => 1 for MPU
A$ = ""
DO
    DO
    A$ = INKEY$: IF A$ = "q" THEN A$ = "Q"
    LOOP UNTIL ((INP(LSTAT) AND 1) = 1 OR A$ = "Q")   ' let data reach
    X = INP(DPORT)
    PRINT TAB(10); RIGHT$("00" + HEX$(N), 2);
    PRINT "  "; RIGHT$("00" + HEX$(X), 2); " "; X
    N = N + 1
    OUT DPORT, X
LOOP UNTIL (A$ = "Q")

OUT MCON, 3                  ' turn DTR, RTS ON => 0 for MPU
END
```

MAKTAB.BAS : Program to create a look-up table from readings of time in seconds, weight of water in lbs, and computer readout of quantized voltage. Two look-up tables are created - one in a format suitable to be loaded into the microprocessor, and the other is created as a text file that shows the entries in the table, and can be used to plot the calibration curve.

```
DEFINT A-D, H-N
DEFSNG E-G, O-Z

DIM TIMESEC(15), LBSWT(15), GALMIN(15), GALRND(15), DECVAL(15)
DIM GALBYTE$(15), ADCBYTE$(15)

IF LEN(COMMAND$) = 0 THEN

PRINT
PRINT "This program creates a lookup table in .MCM format to be loaded into"
PRINT "the uP. It also generates a .OUT file with all the data for comparing"
PRINT "and graphing calibration curves."
PRINT
PRINT "Input is taken from a .IN file in the format :"
PRINT "Wt(lbs)    Time(sec)    Decimal byte output"
PRINT
PRINT "Input file name ? (.IN assumed) ";
INPUT FILE$

ELSE
L = INSTR(COMMAND$, "/")
IF L <> 0 THEN FILE$ = LEFT$(COMMAND$, L - 1) ELSE FILE$ = COMMAND$
IF INSTR(COMMAND$, "/S") <> 0 OR INSTR(COMMAND$, "/s") <> 0 THEN NONORM =
1 ELSE NONORM = 0
END IF

OPEN FILE$ + ".IN" FOR INPUT AS #1
OPEN FILE$ + ".OUT" FOR OUTPUT AS #2
OPEN FILE$ + ".MCM" FOR OUTPUT AS #3

I = 0
DO
I = I + 1
      INPUT #1, LBSWT(I), TIMESEC(I), DECVAL(I)
      GALMIN(I) = LBSWT(I) * 7.19033 / TIMESEC(I)
      ADCBYTE$(I) = RIGHT$("00" + HEX$(DECVAL(I)), 2)
      GALRND(I) = INT(GALMIN(I) * 8 + .5) / 8
      GALBYTE$(I) = RIGHT$("00" + HEX$(GALMIN(I) * 8), 2)
LOOP UNTIL EOF(1)

IF GALMIN(I) < 25 THEN
'       QM1 = (GALMIN(I) - GALMIN(I - 1)) / (DECVAL(I) - DECVAL(I - 1))
'       QM2 = (GALMIN(I - 1) - GALMIN(I - 2)) / (DECVAL(I - 1) - DECVAL(I - 2))
'QDM = QM1
```

```
'       QDM = QM1 - QM2
'       I = I - 1
'       QM3 = (GALMIN(I) - GALMIN(I - 1)) / (DECVAL(I) - DECVAL(I - 1))
'       QM4 = (GALMIN(I - 1) - GALMIN(I - 2)) / (DECVAL(I - 1) - DECVAL(I - 2))
'       QDM2 = QM3 - QM4
'       IF QDM1 > QDM2 THEN QDM = QDM1 ELSE QDM = QDM2
'       I = I + 2
'    I = I + 1
'       GALMIN(I) = 25
'       DECVAL(I) = (25 - GALMIN(I - 1)) / (QM2 + QDM * 1.3) + DECVAL(I - 1)

'DECVAL(I) = (25 - GALMIN(I - 1)) / QDM + DECVAL(I - 1)

      QM1 = (DECVAL(I) - DECVAL(I - 1)) / (GALMIN(I) - GALMIN(I - 1))
      I = I - 1
      QM2 = (DECVAL(I) - DECVAL(I - 1)) / (GALMIN(I) - GALMIN(I - 1))
      I = I + 2
      GALMIN(I) = 25
      QDM = QM1 - QM2  ' change in slope
      DECVAL(I) = (QDM / 2 + QM1) * (25 - GALMIN(I - 1)) + DECVAL(I - 1)

      ADCBYTE$(I) = RIGHT$("00" + HEX$(DECVAL(I)), 2)
      GALRND(I) = 25
      GALBYTE$(I) = HEX$(25 * 8)
END IF

I = I + 1
DECVAL(I) = 255: GALMIN(I) = 30
ADCBYTE$(I) = "FF"
GALRND(I) = 30
GALBYTE$(I) = HEX$(30 * 8)

PRINT
PRINT "Gal/min", "Rounded", "Decbyte", "Galbyte", "Adcbyte"
PRINT

ADDR = VAL("&H" + "FC80")

N = I

FOR I = 1 TO N
PRINT GALMIN(I), GALRND(I), DECVAL(I), GALBYTE$(I), ADCBYTE$(I)
IF I < N THEN PRINT #2, GALMIN(I), GALRND(I), DECVAL(I), GALBYTE$(I),
ADCBYTE$(I)
PRINT #3, LEFT$(HEX$(ADDR), 2) + " " + RIGHT$(HEX$(ADDR), 2) + " " + GALBYTE$(I)
ADDR = ADDR + 1
PRINT #3, LEFT$(HEX$(ADDR), 2) + " " + RIGHT$(HEX$(ADDR), 2) + " " + ADCBYTE$(I)
ADDR = ADDR + 1
NEXT I

CLOSE #1
CLOSE #2
```

```
IF NONORM = 0 THEN
      OPEN "NORMODE.MCM" FOR INPUT AS #1
      DO
         INPUT #1, A$
         PRINT #3, A$
      LOOP UNTIL EOF(1)
END IF

CLOSE #1
CLOSE #3

END
```

MPU.BAS : Program that loads programs (and data) into the microprocessor using the serial port, runs them, and displayes any information that the microprocessor sends back.

```
DECLARE SUB SEND (CHAR%)
DECLARE SUB SENDACK (CHAR%)
DEFINT A-M, X-Z
DIM CODE(256)
COMMON SHARED X, Z, CHAR, DPORT, LSTAT, N
H$ = "&H"

DPORT = &H3F8
LCON = &H3FB
MCON = &H3FC
LSTAT = &H3FD
MSTAT = &H3FE

'BAUD = 2400
BAUD = 976
DVR = 115200 / BAUD

X = &H80
OUT LCON, X             ' access baud divider reg
OUT DPORT, DVR          ' low byte
OUT DPORT + 1, 0        ' high byte

X = 3                   ' 00000011 = no bk, no prty, 1 stop, 8 bit data
OUT LCON, X
OUT MCON, X             ' 3 = turn DTR and RTS on (+ve => 0 for MPU)

FILE$ = COMMAND$
EEPROG = 0
P1 = INSTR(FILE$, "/E")
IF P1 <> 0 THEN
      EEPROG = 1
      FILE1$ = RTRIM$(LEFT$(FILE$, P1 - 1))
      IF INSTR(FILE1$, ".") = 0 THEN FILE1$ = FILE1$ + ".MCM"
      FILE$ = "EEPROG.MCM"          ' std EEPROM proging file
END IF

IF FILE$ = "" THEN INPUT "GIVE FILENAME (.MCM assumed) ", FILE$
IF INSTR(FILE$, ".") = 0 THEN FILE$ = FILE$ + ".MCM"
OPEN FILE$ FOR INPUT AS #1
FOR I = 0 TO 255
   INPUT #1, A$
   CODE(I) = VAL(H$ + A$)
NEXT I

CLOSE #1

PRINT "Reset MPU and press any key to start download"
DO
```

```
LOOP WHILE INKEY$ = ""

DO
   Z = INP(LSTAT)
   IF (INT(Z / 2) * 2 <> Z) THEN X = INP(DPORT)
   Z = INP(LSTAT)
LOOP UNTIL (Z = 96)

SEND (&HFF)

DO
   Z = INP(LSTAT)
   IF (INT(Z / 2) * 2 <> Z) THEN X = INP(DPORT)
   Z = INP(LSTAT)
LOOP UNTIL (Z = 96)

N = 0

FOR I = 0 TO 255
   CHAR = CODE(I)
   SENDACK (CHAR)
   IF N <> 0 THEN I = 300                    ' ABORT !!!!
NEXT I

'PRINT "Press any key to start the MPU"
'DO
'LOOP UNTIL (INKEY$ <> "")

'OUT MCON, 0                  ' turn DTR, RTS off => 1 for MPU
'PRINT

OUT MCON, 1                   ' turn XXXDTR, RTS off => 1 for MPU

IF EEPROG = 0 THEN            ' simply wait for data or PROG end

'     ----------NEW TEST---------

A$ = ""
DO
    DO
    A$ = INKEY$: IF A$ = "q" THEN A$ = "Q"
    LOOP UNTIL ((INP(LSTAT) AND 1) = 1 OR A$ = "Q")   ' let data reach
    X = INP(DPORT)
    PRINT TAB(10); RIGHT$("00" + HEX$(N), 2);
    PRINT "  "; RIGHT$("00" + HEX$(X), 2); " "; X
    N = N + 1
    OUT DPORT, X
LOOP UNTIL (A$ = "Q")

OUT MCON, 2                   ' ? turn DTR, RTS ON => 0 for MPU
END
```

```
'      ---------END TEST----------

IF I < 300 THEN

A$ = ""
N = 0

DO
   DO
      I = INP(MSTAT)
      A$ = INKEY$
      IF A$ = "q" THEN A$ = "Q"
      IF (I AND 32) <> 0 THEN                ' DSR goes off
         A$ = "Q"
         PRINT
         PRINT "Program terminated normally"
      END IF
      I = I AND 16
   LOOP UNTIL (A$ = "Q" OR I = 0)

   IF (I = 0) THEN           ' data sent
    DO
    LOOP UNTIL ((INP(LSTAT) AND 1) = 1)     ' wait for data to reach
    X = INP(DPORT)
    PRINT TAB(10); RIGHT$("00" + HEX$(N), 2);
    PRINT "  "; RIGHT$("00" + HEX$(X), 2); " "; X
    N = N + 1
   END IF

   OUT MCON, 2               ' RTS  ->  0


   IF A$ <> "Q" THEN
         I = INP(MSTAT)
         IF (I AND 32) <> 0 THEN               ' DSR goes off
            A$ = "Q"
            PRINT
            PRINT "Program terminated normally"
         END IF
         OUT MCON, 0               ' RTS  ->  1
   END IF

LOOP UNTIL (A$ = "Q")

END IF                             ' END IF of I abort

ELSE                               ' EEPROG
  I = 0
  N = 0
  OPEN FILE1$ FOR INPUT AS #1
  DO WHILE EOF(1) = 0 AND N = 0
   INPUT #1, A$
```

```
    SENDACK (VAL(H$ + MID$(A$, 7, 2)))
    SENDACK (VAL(H$ + MID$(A$, 4, 2)))
    SENDACK (VAL(H$ + LEFT$(A$, 2)))
    I = I + 1
    LOCATE 1, 1: PRINT I
  LOOP

  IF N = 0 THEN                    ' if Q not pressed
     SENDACK (255)
     SENDACK (1)
     SENDACK (0)                   ' signal EOTransmission

   PRINT "Program complete"
  END IF

END IF                             ' endif of EEPROG

FOR N = 1 TO 30000
NEXT N

OUT MCON, 2                  ' ? DRT, RTS -> 0

SUB REC

END SUB

SUB SEND (CHAR)

OUT DPORT, CHAR

END SUB

SUB SENDACK (CHAR)

Z = INP(LSTAT)
OUT DPORT, CHAR
DO
   Z = INP(LSTAT)
   A$ = INKEY$: IF A$ = "q" THEN A$ = "Q"
LOOP WHILE ((INT(Z / 2) * 2 = Z) AND A$ <> "Q")
IF A$ = "Q" THEN N = 100
X = INP(DPORT)
IF X <> CHAR AND N <> 100 THEN PRINT "MISS >> "; X

END SUB
```

Meter no.	Location	Date	Reading (gallons)
Meter No. 0			
		06/10/94	0
		06/10/94	0
		06/10/94	9.41
		06/10/94	27.42
		07/25/94	1417514
		07/25/94	1417514
		07/25/94	1417514
		07/28/94	1484975
		07/28/94	1484975
		08/05/94	1526450
		08/25/94	1612430
		08/29/94	1620854
		08/29/94	1756734
		08/29/94	1756782
		08/29/94	1756797
		08/29/94	1756841
		09/01/94	1767073
		10/03/94	1815393
Meter No. 1			
		05/17/94	115.01
		05/17/94	203.58
		05/20/94	10535.26
		08/06/94	10535.26
Meter No. 2			
		06/10/94	0
		06/10/94	37.7
		06/10/94	50.51
		06/10/94	58.87
		06/10/94	151.15
		06/14/94	18505.35
		07/25/94	690994.6
		07/25/94	690994.6
		07/25/94	*690994.6*
		07/25/94	*690994.6*
		07/28/94	*724790.9*

08/05/94	*788804.2*
08/05/94	*788804.2*
08/25/94	*1206993*
08/25/94	*1206993*
08/25/94	*1206993*
08/25/94	*1206993*
08/25/94	*1206993*
08/25/94	*1206993*
08/29/94	*1206993*
08/29/94	*1206993*
08/29/94	*1206993*
08/29/94	*1207032*
08/29/94	*1207038*
08/29/94	*1207065*
08/29/94	*1207131*
09/01/94	*1210046*
10/03/94	*1256641*

Meter No. 3

05/24/94	5.17
06/10/94	56096.37
07/25/94	56096.37
07/28/94	66986.89
07/28/94	67038.49
07/28/94	67038.49
08/05/94	169970.9
08/25/94	619307.1
08/25/94	619307.1
08/29/94	652440.9
08/29/94	652440.9
08/29/94	652521.6
08/29/94	652602.2
08/29/94	652680.5
08/29/94	652716.1
09/01/94	694705.4
10/03/94	809986.7

Meter No. 4

06/10/94	123245.7
07/25/94	123245.7
07/28/94	123259.6
07/28/94	123259.6
08/05/94	123259.6

Meter No. 5

05/27/94	0

05/27/94	8.72
05/27/94	39.35
05/28/94	11418.4
05/30/94	32549.13
05/30/94	32552.35
05/30/94	32563.65
05/30/94	32583.1
05/31/94	32839.36
05/31/94	32840.61
05/31/94	32844.36
05/31/94	32880.68
06/01/94	36525.2
06/01/94	36528.34
06/01/94	36529.59
06/02/94	39680.84
06/12/94	78717.24
06/12/94	78717.24
06/12/94	78717.24
06/12/94	78721.74
06/12/94	78725.96
06/12/94	83125.59
06/12/94	83152.75
06/12/94	83152.75
06/12/94	83160.75
06/25/94	233059.3
06/25/94	233066.8
06/29/94	298050.3
07/12/94	567510.4
07/12/94	567510.4
07/12/94	567512.9
07/12/94	567515.7
07/12/94	567523.8
07/12/94	567559
07/12/94	567607.6
07/12/94	567656.1
07/12/94	567704.7
07/14/94	576822.6
07/14/94	576830.6
07/15/94	579714.4
07/15/94	582722.1
07/16/94	582722.1
07/16/94	582722.1
07/16/94	582731.6
07/19/94	615808.8
07/19/94	615808.8
07/21/94	633700.8
07/24/94	643231.3
07/25/94	650507.5
07/30/94	676904.6
07/31/94	694625.3
08/01/94	695738.4
08/01/94	695738.4

08/02/94	706951.5
08/07/94	706960.6
08/07/94	706960.6
08/08/94	710075.8
08/09/94	720240.6
08/09/94	733520.6
08/10/94	748088.1
08/22/94	748088.1

Meter No. 6

06/14/94	0
06/14/94	45.59
06/14/94	45.59
06/14/94	68.21
06/14/94	98.86
06/14/94	144.21
08/31/94	982895.4
09/01/94	982895.4
09/01/94	982903
09/01/94	982946.4
09/01/94	982997.5
09/01/94	983013.8
09/01/94	983094.9
09/01/94	983243.4
09/01/94	983391.8
10/22/94	992005.4

Meter No. 8

06/14/94	0
06/14/94	14.09
06/14/94	49.61
06/14/94	99.32
08/31/94	253370.1
09/01/94	256799.3
09/01/94	256806.6
09/01/94	256838.7
09/01/94	256845.4
09/01/94	256892.4
09/01/94	256932.4
09/01/94	256946
09/01/94	260521.9
10/22/94	1215175

Meter No. 9

06/14/94	0
06/14/94	0

06/14/94	63.63
06/14/94	115.1
06/14/94	230.2
08/31/94	369018.2
08/31/94	369018.2
09/01/94	369018.2
09/01/94	369021.2
09/01/94	369063.3
09/01/94	369082
09/01/94	369088.6
09/01/94	369127.6
09/01/94	369166.3
09/01/94	369314.5
09/01/94	369315.1
10/22/94	397381.1

APPENDIX D

Circuit diagram and parts list
The circuit diagram of the meter is given in Figure 9.
The elctronic components used in the meter are listed below.

Microprocessor
Motorola MC68HC811E2 (1)

ICs
MAX663 (2)
MAX232 (1)
LM358 (1)
MM58274 (1)

IC sockets
48 pin DIP (1)
16 pin DIP (2)
8 pin DIP (3)

Strain gauge
KFG-02-120-C1-11L1M2R (1)
(from Omega Engineering)

Resistors
5% tolerance:
1M (1)
100K (9)
50K (1)
10K (21)
1K (1)
300 (2)
120 (1)
1% tolerance:
1.5M (3)
1M (4)
348K (1)
100K (1)
50K (2)

Potentiometer
20 ohms

Capacitors
10uF 35V (8)
0.01uF (1)
15pF (2)
10pF (2)

Transistor
2N2222 (1)

Diodes
1N914 (2)

Crystals
1 Mhz (1)
3.2768 kHz (1)

Miscellaneous
2-terminal Terminal blocks (5)
5-terminal Terminal blocks (1)
Push button switch - momentarily ON (2)
SPST switch (1)
Serial cable with connector (1)
Duracell "D" cells (4)
4 "D" cells battery-holder (1)

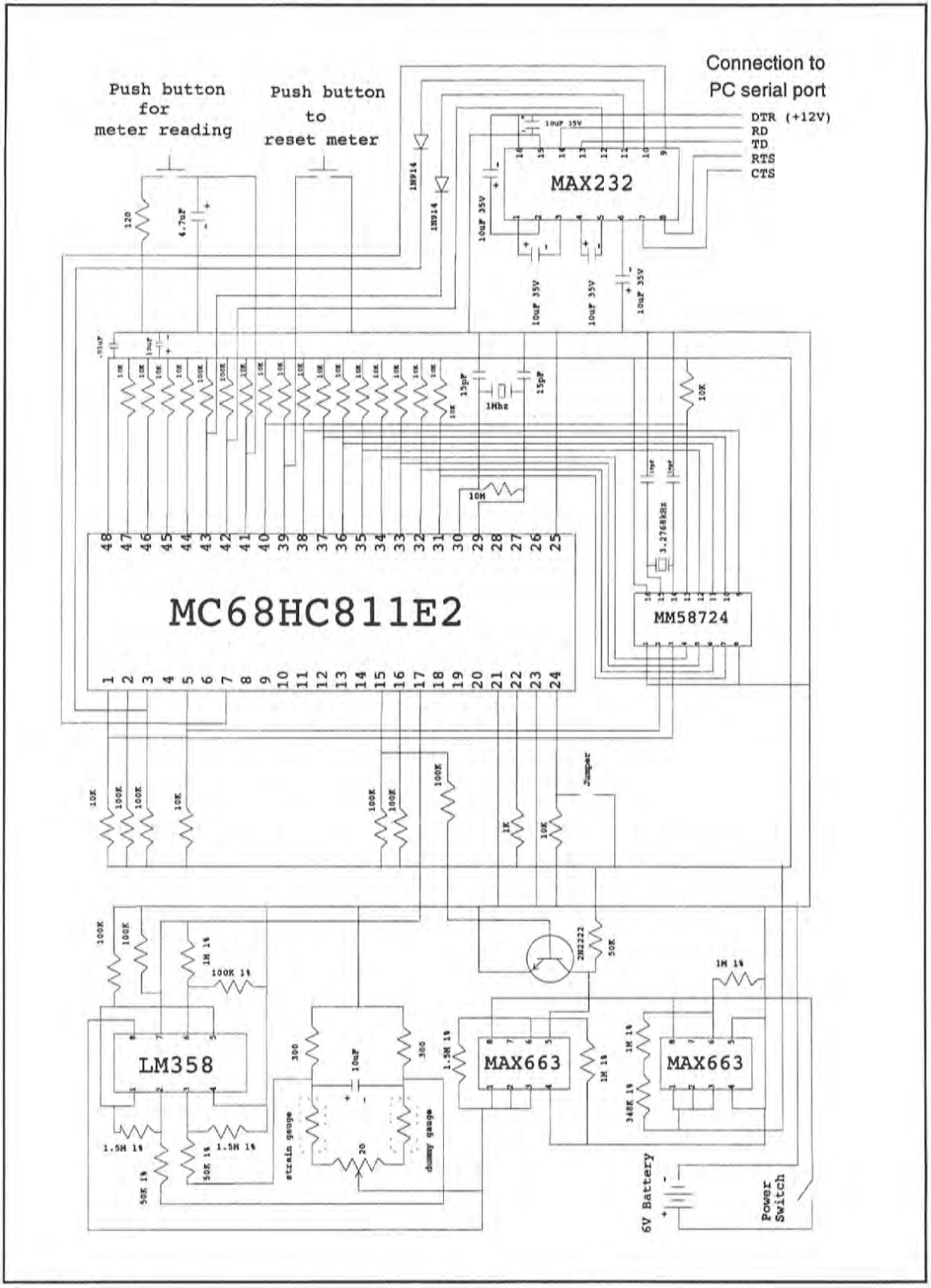

Figure 14. Circuit diagram