WIRE TELEMETRY CONTROLS FOR

GATED-PIPE IRRIGATION SYSTEMS

by

WALTER JAY BRADBURY

B. S., Kansas State University, 1983

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Agricultural Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1984

Approved by:

Harry L Manges

Major Professor

LD
2668
.T4
1984
B72
c. 2

A11202 992408

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Sincere thanks go to Herschel and Tim Webber, without whose cooperation this research project would not have been possible. Also thanks are extended to Mike Schwarz, Stan Black, Jim Frauen, and Mike Olberding for their assistance on the project. Thanks are also given to Dr. J. Garth Thompson and Dr. Mark Schrock for their service on my graduate committee. The support provided by the Kansas Committee on the Relationship of Electricity to Agriculture, the Halliburton Foundation, and the Kansas Agricultural Experiment Station is also greatly appreciated.

Special thanks go to Dr. Harry Manges, whose guidance and assistance helped make my graduate studies an enjoyable experience.

LIST OF FIGURES

INTRODUCTION

Water use in the United States has been under more scrutiny than ever before, particularly in areas relying heavily upon groundwater, such as the Ogallala Aquifer region of the Central Plains. Decreasing water tables and dwindling streams and rivers have caused concern over the future of our water supplies. Heavy water users have been criticized for a lack of efficiency, and since agricultural irrigation is one of the least efficient users, it has been the cause of much of the concern.

Surface irrigation is particularly low in efficiency when compared to sprinkler irrigation or other water uses. Irrigators are reluctant to invest more time and money in labor simply to save water because of the low costs of pumping water. Instead, irrigation patterns are typically adjusted to meet the farmer's schedule. The resulting over-irrigation decreases irrigation efficiencies and increases the cost of irrigating. Drainage and salinity problems can also develop as a result of over-irrigation.

Considerable research has been done in the past few years in trying to increase the efficiency of surface irrigation. One potential means of increasing efficiency is through improved irrigation methods, most notably surge and cutback irrigation. Cutback irrigation is the application of water to an irrigation set in the normal manner until runoff occurs, at which time the flow rate into each furrow or border is reduced; this is often referred to as "cutback head". Surge irrigation is the practice of alternating the flow of water down two irrigation sets on relatively short time intervals, usually from less than 20 minutes to over 3 hours on a set. Although both of these methods suggest increased efficiencies, implementation of them increases the labor required for irrigation. Since few farmers are willing to increase labor to increase efficiency, some means of automatically performing the changes needs to be developed.

Although efforts to automate surface irrigation systems have been relatively successful, they have still lagged behind the automation of sprinkler irrigation. Labor requirements for surface irrigation can push the cost of operation beyond that for sprinkler irrigation, especially if one considers convenience in economic evaluations. The recent trend has been to convert land that is suitable for surface irrigation to sprinkler irrigation. Sprinklers operate at higher working pressures, and as a result have higher energy requirements than surface systems. It is hoped that by reducing labor requirements and increasing efficiencies through automation of surface systems, this recent trend can be reversed.

The Agricultural Engineering Department began a project in 1979 which investigated the use of radio controls in the automation of gated-pipe irrigation systems. While some degree of success was achieved, problems occurred which prompted further research. Research has also been conducted elsewhere in the use of radio controls for surface irrigation, but little work has been completed using wire telemetry (transmission of the signal over wires). This project investigated the potential advantages and disadvantages of applying wire telemetry controls to the automation of gated-pipe irrigation systems.

LITERATURE REVIEW

Efforts to automate surface irrigation began in the late 1950's, when systems using mechanical time clocks to activate control structures were first developed. Pair (1961) saw a need for automation in surface irrigation to reduce labor requirements, and emphasized the need for systems to control the flow of water in open canals and pipelines.

Bowman (1969) described a system which used radio controls to successfully activate control gates in open ditches. The transmitters were located at the lower end of graded borders, and were triggered by the presence of the wetting front. When the receiver detected the signal, it would open a control gate, allowing the flow of water to another border. Since the transmitters used water presence to trigger the signal, no timing device was needed; the only system variable adjustable by the operator was transmitter location. Although no mention was made of interference problems, the use of citizen's band frequencies would today be prone to interference given the current popularity of citizen's band radios.

Haise, Kruse, and Dimick (1965) developed a pneumatic valve to be used in automated systems. The valve resembled a large inflatable O-ring mounted between an alfalfa valve seat and lid. Air pressure was supplied to the valve through plastic tubing and was controlled with a 3-way solenoid valve. A central controller utilized an electro-mechanical timing device and relay circuits to provide power to the appropriate solenoid valve. When power was provided to a solenoid, air pressure was directed to the O-ring, which then inflated to form a seal between the alfalfa valve seat and lid, effectively shutting off the flow of water from the riser.

Haise, Kruse, Payne, and Duke (1980) used this valve in testing two automated systems. Tone-encoding devices generated signals which in the first system were carried by radio waves, and in the second system by wires. The transmitters were capable of producing 4 channels or tones. An industrial timer controlled a set of relays to determine which signal channel was transmitted.

In the radio-controlled system, receivers located at each valve decoded the signals and activated latching solenoid valves to operate the diaphragm valves. Batteries located at each receiver provided the power for both the receiver and the solenoid valve.

In the wire telemetry system, 3-strand 16 ga. cable was fed through the air supply line. Two wires supplied the power to the solenoids and the receivers, while the third was used to carry the signal. The system was capable of sending reliable signals up to 2.4 km (1.5 miles) away, but voltage drops occuring in the power lines at this distance required increasing the source voltage to prevent erroneous changes.

The majority of the problems associated with these systems were attributed to the reliability of the electronic components and the construction of the controller. A need was expressed for adequate and reliable safety devices in the pipeline, since occasional problems were noted with inoperative receivers. These researchers contended that multi-frequency signals transmitted by a pair of wires appeared to hold the most promise, since radio controls required an FCC license and could potentially cause interference with neighboring systems.

Fischbach and Goodding (1971) developed a valve similar to the Haise et al. (1965) pneumatic valve. This valve sat directly on a riser, replacing the alfalfa valve and bonnet normally used. Fischbach, Thompson, and Stetson (1970) used these valves in the automation of a surface irrigation system. The controls consisted of an electical clock, relays, time-delay relays, and stepping relays to control small 3-way solenoid valves. These valves were attached to small air lines which delivered air pressure to the appropriate diaphragm valves; one air line was required for each diaphragm valve to be controlled. Tensiometers sensed soil moisture to determine irrigation starting times.

Haise and Payne (1969) eliminated the need for an air pressure source by developing a valve which used water pressure to operate. This created an advantage in remote locations where providing the air pressure source was not practical. There remained, however, the need for a means of directing the flow of water into or out of the diaphragm.

Humpherys and Stacey (1975) modified and refined this idea in developing a valve which also operates using water pressure. A diaphragm was mounted in a housing that was placed directly in the pipeline. Water flowed into the valve, around the diaphragm, and out the other end of the valve (see Figure 1a). A pitot tube mounted on the upstream side of the diaphragm brought water through a 3-way valve and into the diaphragm to inflate it. The diaphragm would inflate until it pressed firmly against a lip on the inside of the valve, shutting off flow through the valve (see Figure 1b). To allow the flow of water through the valve again, the 3-way valve was turned to allow the water in the diaphragm to exhaust to the atmosphere, releasing the seal against the lip inside the valve.

Haise, Kruse, and Erie (1969) relied upon pressurized water to operate water cylinders in automating a canal system. Three-way valves controlled water flow into the cylinders. The most severe problem occurring in this research was the damage done by rodents. Damage was done to the plastic tubing carrying the pressurized water and to the wiring for control of the 3-way valves. It was proposed that water wheels or turbines in the canal or pipeline could be used to develop the required pressure.

The flexibility and reliability of the timing device used in an automated system will have a great influence on its acceptance. Recent advances in the microcomputer and electronics industries have led to more reliable and less expensive components. The accuracy obtained by the crystal-controlled clock

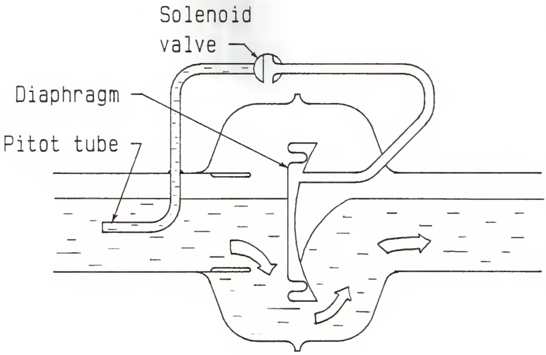


Figure 1a. Flow Control Valve Open.

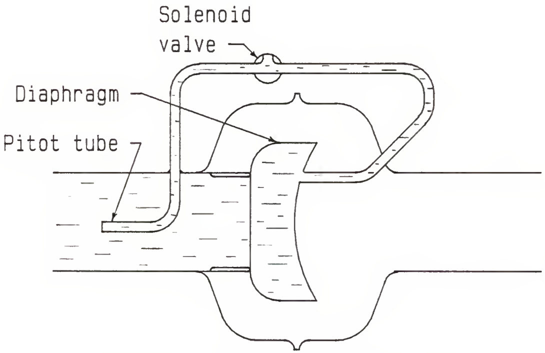


Figure 1b. Flow Control Valve Closed.

and the flexibility gained has led to more widespread use of microprocessors and microcomputers in this type of control. Several researchers (Edling et al., 1978; and Fisher et al., 1978) have used microprocessors and electronic timers to control surface irrigation systems. Lillevik (1982) describes a controller based upon the popular Z-80 microprocessor. Conventional electronics (rather than CMOS) were used and therefore required an automotive battery for one season's operation. Lamb et al. (1982) developed a system which not only controls a solid-set system but also schedules the irrigations.

Fischbach and Somerhalder (1971) report water-distribution efficiencies of 92% and water-application efficiencies of 92% when using automated gated-pipe systems with reuse systems. This is a considerable improvement over typical surface irrigation efficiencies and is comparable to those of sprinkler efficiencies. Automation seems to be a feasible way of increasing surface irrigation efficiencies without increasing labor inputs, and wire telemetry controls have received little attention as the means of accomplishing automation.

INVESTIGATIONS

Objectives

The objectives of this research were:

1. To automate a gated-pipe irrigation system using wire telemetry controls.

2. To provide flexibility in the controls to permit improved water management practices.

3. To evaluate the performance of wire telemetry controls in gated-pipe irrigation systems.

4. To evaluate the economic and labor-reducing potential for wire telemetry controls in gated-pipe irrigation systems.

Previous Work

During the summers of 1977 and 1978, a small project was carried out on the Herschel Webber farm in Haskell County, Kansas. The project received funding and support from the Southwest Kansas Groundwater Management District No. 3, the Kansas Agricultural Experiment Station, and the Hastings Irrigation Pipe Co., Inc. of Hastings, Nebraska.

One of the objectives of the project was to irrigate a field using flow control valves of the type developed by Humpherys and Stacey (1975). A 3-way brass pilot valve mounted on the side of each flow control valve directed the flow of water into or out of the diaphragm. The flow control valves were used with some success, and helped spur interest in developing controls for gated-pipe systems.

A second objective of the project was to test different irrigation practices. Although the test results were not conclusive, the investigations prompted further research.

A larger project was begun on the same farm in the spring of 1979 which used radio controls to operate the flow control valves (Blume, 1978 and Manges et al., 1980). Funding was provided by a grant from the United States Department of Energy and the Kansas Agricultural Experiment Station.

In this project, a central control station, consisting of radio transmitters, irrigation timers, and a plywood box, was mounted on two posts near an equipment shed at the north side of the field. Power was delivered to the box through a 110 VAC power cord plugged into an outlet inside the equipment shed. A four-receptacle electrical outlet was provided inside the box.

Each flow control valve, similar to those used in the previous project, controlled a complete and independent irrigation set. Three-way pilot valves were used to direct the flow of water into or out of the diaphragm of the flow control valves. Radio receivers and servo motors were located at each of 18 flow control valves. When the receiver detected the appropriate signal, it would position the servo motor to open the pilot valve, causing the diaphragm in the flow control valve to deflate.

The radio transmitters were Heathkit 8-Channel Digital Proportional Radio Control Transmitters. They transmitted signals which were pulse-modulated RF (Radio Frequency) carriers that permitted the control of up to 8 devices on the same frequency. The signal consisted of a frame of 9 pulses repeated every 25 milliseconds. The time interval between pulses was variable; it was this time interval that was used to position the servos. One of these variable segments was used to control each servo. Three Rainbird AG-7 Automatic Irrigation Controllers inside the box controlled the valve sequencing.

Originally, a special 3-way sliding valve consisting of a teflon spool inside an aluminum body was used to switch water flow to and from the diaphragm. Nicks in the teflon which developed during manufacture resulted in problems with leaking. Also, high temperatures resulted in the valve becoming difficult to move while colder temperatures caused leaking due to the different coefficients of thermal expansion for the two materials. These valves were later replaced with 3-way brass valves, which operated with few problems.

A high capacity (14 Amp-hour) 6-volt battery supplied the power for both the receiver and the servo motor on each valve. Each battery was charged with a separate silicon solar panel, which provided enough electrical power to maintain battery charge during continuous use and recharge what was used during the night.

The radio controls were operated with limited success, as numerous problems were encountered initially. The servos continually chattered at the far end of the field due to signal degradation. Also, lightning damaged some of the servo motors during field tests. The final design completed about 80% of the valve changes that it was scheduled to make.

Again, most of the problems encountered in this research were related to commercial components not being designed to operate in the adverse environment. More reliable means of not only sending and receiving the signals is needed, but also of performing the change once the signal is received. Wire telemetry seemed to hold promise in reliability of signals, and commercial solenoid valves were hoped to be a more reliable means of performing the change.

Initial System Design

Permission was obtained from Mr. Herschel Webber of Haskell County, Kansas to irrigate a 65-ha (160-acre) field using wire telemetry controls. Funding was provided by the Kansas Committee on the Relationship of Electricity to Agriculture, the Halliburton Foundation, and the Kansas Agricultural Experiment Station.

Field Layout

Water was supplied from the existing pump and well at the southwest corner of the field, and was pumped through an underground pipeline along the west end (see Figure 2). Risers were located approximately every 92 m (300 feet). Tees were connected

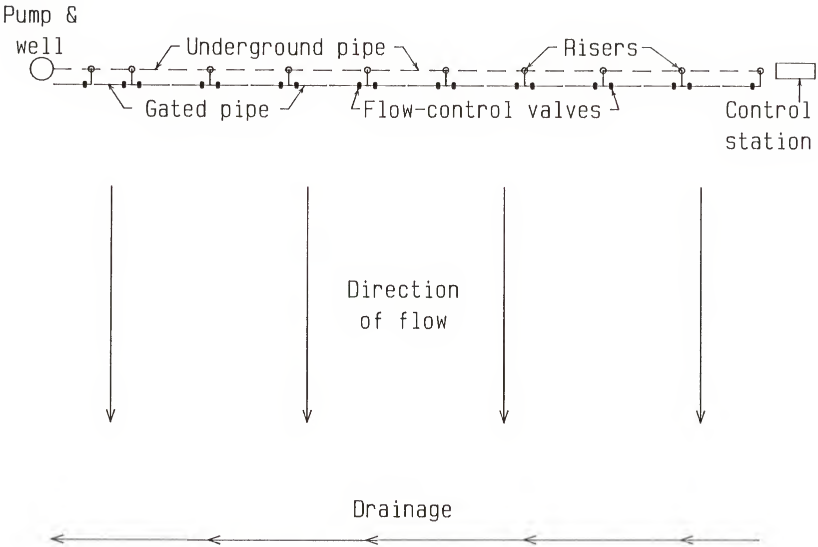


Figure 2. Field Layout.

to bonnets on the risers, and a flow control valve was mounted on each side of the tee, as shown in Figure 3. About 46 m (150 feet) of gated pipe (enough to adequately handle the flow from the well) was attached to each flow control valve, with a plug installed in the outer end. All the gates were opened to a predetermined position before the irrigation began. In this way, each flow control valve had independent control of a single and complete irrigation set.

Control Valves

The flow control valves were the same as those used by Blume (1980), and were similar to those developed by Humpherys and Stacey (1975). Clear vinyl hose was used to connect the pitot tube to the diaphragm of each valve. A small reservoir attached to the body of each valve provided the water necessary to fill the diaphragms at the start of each irrigation. Figure 4 shows the solenoid valves that were installed in the diaphragm lines to switch the flow of water to and from the diaphragm.

The solenoid valves received the power for their operation from a 24 VAC transmission line running along the west end of the field alongside the risers and gated pipe. This line was laid on top of the ground during testing of the system, but in a permanent installation would likely be buried with the supply pipeline.

Twenty-four VAC solenoid valves were not available with a sufficient orifice size to allow the flow control valves to

Figure 3. Flow Control Valves Mounted on a Riser.

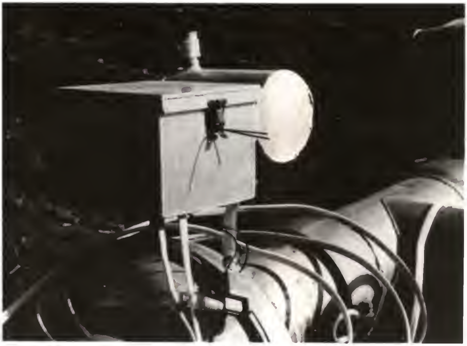


Figure 4. Solenoid Valve Connected in Diaphragm Line.

operate fast enough (a C_v value of 0.5 or greater was desired). The manufacturer explained that the 24 VAC solenoid required more space than the standard enclosure for this orifice size allowed. Adequate 24 VDC valves were available, so the 24 VAC was fed through a full-wave bridge rectifier to provide the 24 VDC for the solenoids. The result was a full-wave rectified sine wave. Normally a large capacitor is used to smooth the wave to obtain a more uniform DC voltage. However, the manufacturer of the transmitters expressed concern that such a circuit would severely distort the signal. Laboratory tests confirmed this, so these capacitors were not used.

The size of the wire for the transmission lines was governed by the voltage drop occurring when two solenoids at the farthest end of the field were on. The solenoids had a minimum working voltage of 20 VAC and required about 0.7 amps of current. Calculations indicated the required copper wire size was #12 AWG, and since the line was to be laid out in direct sunlight, type UF insulation was necessary.

In anticipation of problems with making dependable changes, a safety feature was installed on one of the flow control valves. A tee was placed in the diaphragm hose, and an additional ten-foot section of hose was attached to the tee. The top end of this hose was strapped to a post as shown in Figure 5 at a height determined to be the maximum desired pipe pressure. When pressure in the pipe exceeded the height of the hose end, water expelled from the diaphragm, which allowed water to flow out that

Figure 5. Safety Valve in Field.

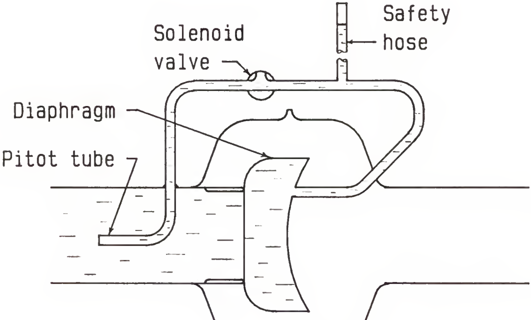


Figure 6. Cross-section of Safety Valve.

irrigation set and relieved the pressure in the pipeline. When the pressure receded back to the height of the hose end, the diaphragm partially closed, maintaining the maximum desired pressure in the pipeline. Figure 6 shows a cross-section of the safety valve and hose.

Wire Telemetry Components

As previously mentioned, the transmission line providing power to the solenoids was operating at 24 VAC. Although 24 VAC would not carry the power as efficiently as 110 VAC, the safety of the lower voltage was a deciding factor, especially since the system was experimental and consisted largely of aluminum components. In addition, the Rainbird Irrigation Controller which was used in the initial phases had 24 VAC on its outputs; to use another voltage would have required the use of an external relay for each channel. Transmitters and receivers were available in either 24 or 110 VAC.

The 24 VAC transmission line not only carried the power for the receivers and solenoids, but also provided the means of carrying the transmitted signal to the receivers. Southwood Electronics Model T8-24 transmitters, mounted in a control panel at the northwest corner of the field, superimposed a high frequency sine wave on the 60 Hz, 24 VAC power line. The signal was less than 1 Vrms and varied in frequency from 185 to 260 kHz, depending on the channel being transmitted. The frequencies are well out of range of citizen's band and other common frequencies which might cause interference.

Each flow control valve was equipped with one Southwood Electronics Model R8-24 receiver. The receivers use filtering and narrow-bandwidth techniques to detect the appropriate signal. When the correct signal is on the transmission line, the receiver turns on an internal relay, which in this system switches on power to the solenoid valve.

Each pair of flow control valves connected to a single riser was operated on the same transmitter channel. Each, however, was supplied by a different 24 VAC transmission line, so switching between the two receivers on the same channel could be accomplished by switching between the two transmission lines with a 24 VAC SPDT relay (see Figure 7). Both power lines used a common ground, resulting in the need for a 3-wire cable (type 12/3 UF).

Since during our testing period the field was equally divided between corn and wheat, no more than 6 pairs of flow control valves would be irrigated during a single irrigation. For this reason, only 6 transmitter and receiver channels were purchased and designed for in the timing devices. The transmitters and receivers were purchased from Southwood Electronics, Inc., of Greenwood, Indiana.

Irrigaton Practices Implemented

The irrigation practices that were implemented were surge, cutback, and continuous irrigation. Surge irrigation, as mentioned previously, is the practice of alternating the flow of water down two irrigation sets on relatively short time intervals

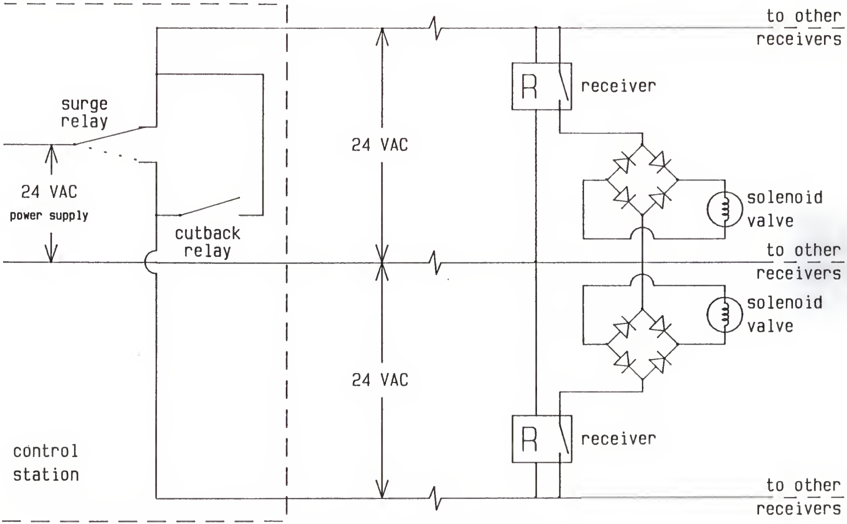


Figure 7. Transmission Line Wiring.

(from about 20 minutes to over 3 hours on each set). Surge was accomplished by switching the power from one transmission line to the other with a 24 VAC SPDT relay. In this way, only one of the pair of receivers on that channel would be receiving power and could be on.

Cutback head irrigation, as implemented in this design, was accomplished by opening two flow control valves and therefore dividing the flow of water down two irrigation sets, effectively cutting the flow of water down each furrow in half. Another 24 VAC SPDT relay was used to provide power to both transmission lines to provide cutback irrigation.

Continuous irrigation was accomplished by surging with only one cycle on long time intervals or by using a cutback head for a long period of time without any surge. The control algorithm implemented on each of the timing devices was a period of surge irrigation followed by a period of cutback head. Figure 8 is a flowchart of this algorithm.

Timing Devices

A transmitter will impose its signal on the line when it is connected across the line. Therefore, the timing device needed to only switch power on and off to the appropriate transmitters. And, since two power lines provided the power to the receivers, surge irrigation was accomplished by switching between the two power lines while keeping the same transmitter channel on. Cutback irrigation was accomplished by turning both power lines on.

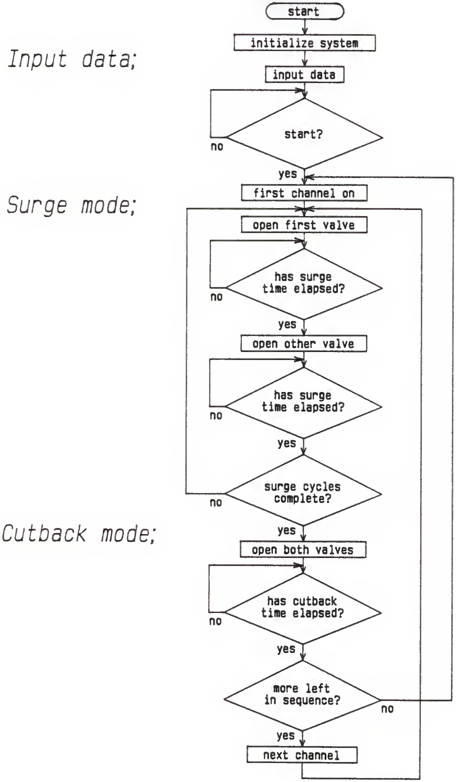


Figure 8. Flowchart of the Control Algorithm.

This timing function was provided in the initial design by two different methods: two Rainbird AG-7 Automatic Irrigation Controllers, and a Synertek Systems SYM-1 Single Board Microcomputer. Each is discussed separately below.

Rainbird Controller. The Rainbird AG-7 is an electro-mechanical device which uses 24 VAC clock motors to perform the timing function. Two controllers were used, each with 6 usable stations (a seventh was used to transfer control to the next controller) for a total of 12 stations. The first controller was assigned transmitter channels 1-3 while the second controller was assigned channels 4-6. On each controller, the odd numbered stations were surge irrigation and the even numbered stations were cutback. The times set on each station dial determined the total amount of time in that mode. A separate equal-on/off timer, completely independent of the Rainbird controller, determined the time of application on each irrigation set for surge irrigation. Its function was to simply provide equal on and off times; its outputs were connected to the coil of the SPDT relay that toggled between the transmission lines.

Following Table 1 as an example, typical operation with this controller was somewhat inflexible. Beginning with station 1 of the first controller, surge irrigation would be performed for the time set on the station 1 dial, and would be performed on the valves whose receivers were tuned for transmitter channel 1. The actual surge time on each irrigation set was determined by the dial setting on the on/off timer. After the duration on station

1 is complete, the controller transfers to station 2, which is still channel 1 but a cutback mode. Both valves assigned to channel 1 would then be open. After the time set by station 2 elapses, the controller passes control to station 3, which repeats the above process for transmitter channel 2. Table 1 is a complete table of the controller stations and their associated transmitter channel numbers and modes.

Table 1. Rainbird Controller Station Assignments.

Controller	Station	Transmitter channel	Irrigation mode
A	1	1	surge
A	2	1	cutback
A	3	2	surge
A	4	2	cutback
A	5	3	surge
A	6	3	cutback
A	7	-	passes control to controller B
B	1	4	surge
B	2	4	cutback
B	3	5	surge
B	4	5	cutback
B	5	6	surge
B	6	6	cutback
B	7	-	passes control to controller A

SYM-1 Single Board Computer. In an effort to provide more flexibility in the control algorithm, a SYM-1 Single Board Computer was programmed to perform the timing function. The program was written in assembly language and burned into an EPROM (Erasable Programmable Read-Only-Memory) using an Apple II+ microcomputer in the Agricultural Engineering Department. (A complete listing of the program is in Appendix B.) The SYM-1 was configured so that the program began running immediately when power was turned on.

It was desired to be able to control each transmitter channel independently, so each output bit of an output port was designed to correspond with a transmitter channel. Since the output port contains eight bits, two bits remained to determine the controller mode. One of the remaining bits controls the SPDT relay which toggles between power lines to accomplish surge irrigation. The toggling action was then performed through software control rather than the equal-on/off timer used with the Rainbird controller. The last remaining bit was designed to control the relay which connects power to both lines, resulting in the cutback mode.

Data for the timing algorithm was programmed to be entered when power was turned on. Simple prompts informed the user (with the aid of an informative chart) which data to enter next. The keyboard on the SYM-1 was re-labelled to help the user identify key functions as they relate to the control program. Very little information on controller status was fed back to the user since the only means of doing so was through the on-board 6-digit LED.

Two separate interface boards were made to switch the 24 VAC to the transmitters using the output port from the SYM-1. One consisted of electronic relay drivers (since the output port could not directly source enough current to drive relays), eight relays (one for each bit of the port), and eight LED's to indicate which relay (and corresponding output port bit) was turned on. This board simplified troubleshooting since it provided immediate feedback on controller output status through the LED's. The second board contained optical isolators and TRIAC's to perform the switching. The components on this board were entirely solid-state, which should lead to more reliable operation in extremely dusty environments. Appendix A contains the schematics for these interfacing boards.

A functional block diagram of the automated system is shown in Figure 9.

Results of Initial Design Field Tests

The initial design of the system was tested during August of 1983, and additional testing was done during March of 1984. The system performed with some success, as numerous changes were made without problems. However, several problems did exist, and efforts were made to identify the sources of the problems and to make modifications in future designs to correct them.

Probably the most difficult problem was that some receivers were turning their outputs on even when their associated transmitter was not sending a signal. In fact, further testing indicated that the receivers remained on even when no transmitters were on. Since both of the receivers for channel 1 were malfunctioning, noise on the line was suspicioned to be the source of the problem. In an effort to reduce possible power

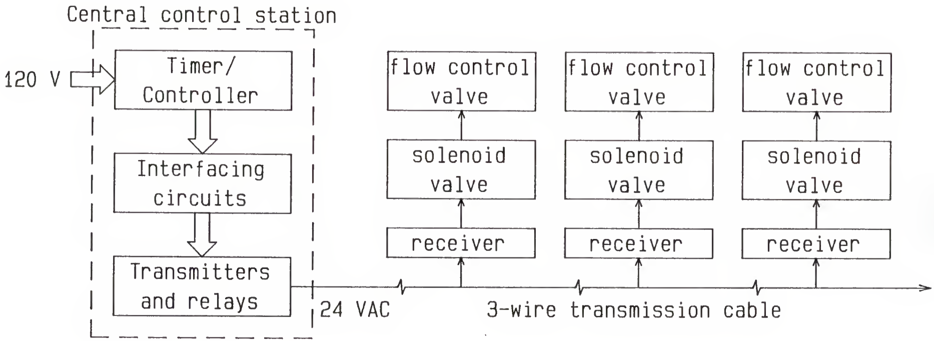


Figure 9. Functional Block Diagram of Automated System.

line noise, a power line filter was used on the 110 VAC line ahead of the 24 VAC transformer. The filter did not seem to have an effect on the errant receivers. One of the channel 4 receivers later developed the same problem. Since these receivers were on separate channels which did not operate on adjacent frequencies, the possibility of noise causing the problem was ruled out. Faulty receivers were apparently the cause, so they were returned to the company for replacement.

On two occasions the valves intended to open did not operate, resulting in all of the valves being set to close. One of these malfunctions was due to technician error, as the transmission wires had been inadvertently disconnected near the control station. The other malfunction was due to a solenoid valve being stuck. Tapping on the valve allowed it to again operate normally.

Both of these malfunctions should have resulted in the safety valve opening to release the pipe pressure. However, the safety valve did not respond, and the result was that pipe pressure pushed some of the pipe sections apart. Apparently the pitot tube was supplying enough water to the diaphragm to replace what was exiting through the safety hose. A larger hose was then installed as the safety hose for field testing in March, which allowed water to exit faster than it entered from the pitot tube. During field tests in March, the safety valve was observed to be operational. However, the cooperating farmer indicated that he had observed water coming out the system standpipe, indicating that on at least one occassion the safety valve did not work properly.

On two other occasions solenoid valves were stuck in the open position, resulting in flow control valves remaining open. Again, tapping on the valves resumed normal operation.

A minor problem was encountered with the on/off timer used to set the surge time intervals. The time interval was set using a simple potentiometer, so setting the time interval to a precise time was extremely difficult. The repeat accuracy was excellent, so that once a time was set it was repeated accurately, but the initial time setting was unreliable. Although this problem did not result in malfunction, it did result in considerable inconvenience and a lack of reliability.

Revisions in Design

Attempts were also made to increase the reliability of the safety valve by using a larger safety hose. It was hoped that this would help prevent the water from entering the diaphragm through the pitot tube as fast as it tried to exit through the safety hose.

Since the original on/off timer was inflexible and difficult to set, a new timer was designed and built to meet our requirements. The new timer used a quartz crystal time-base for accuracy and thumbwheels to allow the setting of a precise data time. Momentary pushbuttons allowed setting and resetting the timer (setting restarts the timer with the output on, while resetting restarts the timer with the output off). The new timer was used in lieu of the commercial timer in the remaining field tests. A design error was found during tests in March, and was corrected for testing in May.

The Rainbird irrigation controller proved to be very reliable, but also very inflexible. Sequencing through the irrigation sets in a different order required reversing some wires, while on the SYM-1 it was part of the data input sequence. Also, the total surge time was set as a time on the Rainbird, while it was entered as a number of surge cycles on the SYM-1. In most applications, an operator would estimate the number of cycles required for water to reach the end of the field; this would then determine the total surge time.

Since one of the objectives of the research was to provide the flexibility in the controls, another controller was developed for testing. A Commodore VIC-20 microcomputer was set up to perform the timing function, as shown in Figure 10. The program was written to follow the same control algorithm as the SYM-1, and the input variables were much the same. However, since the VIC was programmed in BASIC and a full screen was available for feedback to the operator, the program was written to display a variety of system variables as an indication of system status. Appendix C is a full listing of the program.

The program was stored on cassette tape, so when power was

Figure 10. Microcomputer at Control Station.

Figure 11. Receiver and Mounting Board.

turned on the operator needed to load the program from tape to memory. During program execution, the VIC displayed the current time of day, time of the next change, current transmitter channel number, and the system status (either running/timing or holding/stopped). In addition, the lower portion of the screen displayed a menu, giving the operator the options available to him. The entire VIC controller, including monitor and tape recorder, was comparable in cost to either the Rainbird or the SYM-1, and provided much more flexibility. The menu-driven program virtually eliminated the need for an operator's manual for the controller. Operation of either the Rainbird or the SYM-1 was complicated enough that lengthy instructions needed to be given to technicians or other operators.

Also, the wiring connections at each receiver were originally made using wire-nut wire connectors. These connections proved to be awkward and inconvenient. In addition, wiring diagrams were necessary to insure that proper connections were made at each receiver when the equipment is set up. The receivers were later mounted on a small board equipped with a terminal strip for easy connection, as shown in Figure 11. The rectifiers used to convert the 24 VAC to a DC wave were originally mounted on this board, also. In the final design, these were mounted in the plastic receiver boxes. This helped reduce the chance for electrical short circuits and helped prevent leads on the rectifiers from breaking off.

Results of Final Design Field Tests

Field tests were planned for May of 1984, but an unusually wet spring eliminated the need to irrigate the field. A simulation was then set up to test the controls at the Ashland Research Unit No. 1 of the Department of Agronomy and the Agricultural Experiment Station, while the safety valve was tested in the Agricultural Engineering Department Hydraulics Lab.

The safety valve was set up in a manner to simulate its performance in an irrigation system. The same flow control valve that was used during field tests in March was set up using the same safety hose, solenoid valve, and fittings. The valve performed as expected without modifications. Apparently, the malfunction in March was a result of both the distance between the safety valve and the pump and the reaction time of the safety valve. Because it takes several seconds for the safety to begin opening, the water being pumped from the well developed a sufficient pressure in the pipeline to push water out of the standpipe before the safety could open. In addition, the farther the safety valve was from the pump, the higher its elevation; therefore, more pipeline pressure at the well was needed to activate the valve. Also, more time was required to develop the pressure at the safety since the pipeline itself initiated a time delay.

In testing the controls, the equipment was set up along a narrow strip of grass beside a dirt road. Power was provided at the north end and the transmission lines were laid out as far south as possible, then reversed and laid out to the north for a total distance of about 850 m (2800 feet). The receivers were located about every 92 m (300 feet) just as they would have been in the field.

The system was set up with the VIC-20 as the controller and was programmed to cycle through one surge cycle on each channel, then proceed to the next channel. One-gallon buckets were mounted on posts and were filled with water. The solenoid valves were then connected to fittings on the bottom side of the buckets, and a smaller can was place underneath the solenoid. When the receiver detected the signal and opened the solenoid, water would drain to the lower can, allowing a means of detecting if a receiver had been on when it shouldn't have been.

A minor problem was found with one of the receivers; it was turning its outputs on when no signal was present. The problem was found to be in the construction of the mounting boards for the receivers. Two extra wires from the receiver were left intact and came into contact with one another, causing the output of the receiver to turn on. The problem was remedied by cutting the loose ends off of the wires and wrapping a short piece of electrical tape around them. This, however, could not have caused the receiver malfunctions in March because the mounting boards were wired differently.

The cause of this malfunction was careless construction of the mounting boards and was not a result of inoperative controls;

the controls performed without fail for a period of time equivalent to irrigating roughly 80 hectares (200 acres) on 60-minute surge intervals. Measurements with an oscilloscope indicated no distortion or reduction in signal strength over the 850-m (2800-foot) length of transmission cable. This was true both with and without loading at the farthest end.

DISCUSSION

The performance of the final system design indicates that automation of gated-pipe systems is feasible and can be made to be reliable. Long-term durability of components still remains a question since these tests were of relatively short duration.

Blume (1979) experienced problems with damage to components from a passing electrical storm. Since no electrical storms occurred during these tests, no conclusions can be made about the effects of nearby lightning on the system components. Other types of adverse weather, such as blowing dirt and high temperatures, might cause severe problems with the microcomputer and relay contacts in the system.

Automated gated-pipe systems will permit the use of improved water-management practices that were in the past considered too labor intensive to be practical. Cutback irrigation has been suggested (Garton, 1966) as a means for improving efficiencies. Water is applied at normal or above normal rates until runoff occurs, at which time flow is reduced to decrease runoff. Implementation of this requires either automation or plentiful and dependable labor. An automated system could include the ability to recognize when runoff occurs with devices similar to those developed by Bowman (1969). Microcomputers would lend themselves well to this type of automation by then calculating the amount of water needed to complete the irrigation and providing a cutback head for that length of time.

Surge irrigation is another practice that would not be feasible without some degree of automation. Bishop et al. (1981) found that advance rates for surge flow methods were dramatically greater than those for continuous irrigation. Coolidge et al. (1982) confirmed these results and suggested that efficiencies might be higher with surge irrigation, since with faster advance rates the entire furrow length has more uniform opportunity times. Again, microcomputers could perform more complex methods, such as increasing the surge time with each successive cycle or variable duty cycles (such as 1/3 on and 2/3 off time). Dependable controls are necessary to research the advantages and disadvantages of these new labor-intensive methods.

Labor Requirements

The labor requirements for conventional and automated gated-pipe systems and sprinkler systems were estimated and are shown in Table 2. Estimates were based upon four irrigations per season, including one pre-irrigation, on a 60-ha (150-acre) field using 800-m (2600-foot) runs.

Labor requirements for surface irrigation are increased dramatically if pre-irrigation is used since it requires setting up and taking down the gated pipe twice during the season. Both sprinkler and automated surface irrigation have labor requirements which are dependent on the number of days of irrigation more so than the number of irrigations.

Table 2. Estimated Annual Labor Requirements for Selected Irrigation Systems.

System	Preparation	Operation	Total	Per ha.
Conventional gated-pipe	68	120	188	3.1
w/o pre-irrigation	36	90	126	2.1
Automated gated-pipe	80	40	120	2.0
w/o pre-irrigation	46	30	76	1.3
Sprinkler	16	40	56	0.9
w/o pre-irrigation	16	30	46	0.8

The preparation time is slightly higher for automated systems than the others, since some preparation is required for the controls. Preparation times were estimated from the time required to set up this system for field tests. Operation time for a conventional system is very high as a result of the time required to change individual gates each time an irrigation set is changed. Operation time for the sprinkler and automated surface systems was based upon one visit per day at one hour per visit, with each irrigation consisting of ten days. Harrer and Wilfert (1983) describe a study which found that decreasing the length of run from 800 m (2600 feet) to 400 m (1300 feet) resulted in an increase in annual labor requirements of 30-40%. This was due primarily to the increased preparation times.

Annual labor requirements for typical systems were found to be 3.1, 2.0, and 0.9 man-hours per hectare (1.2, 0.8, and 0.4 man-hours per acre) for conventional gated-pipe, automated gated-pipe, and sprinkler systems, respectively. Clearly, automation will not reduce labor requirements of gated-pipe systems to levels required by sprinkler systems, but it does cause a substantial reduction.

Economic Requirements

Estimated costs of components of an automated system similar to this one are shown in Table 3. These costs are based upon a 60-ha (150-acre) field, and include all component costs not related to the well and pump or land preparation. Price estimates were obtained from Geis Irrigation, Sublette, Kansas.

These estimates are based upon the system that was tested; a commercial version would probably integrate the receiver/solenoid valve/connectors combination to simplify set-up, and might possibly design a valve combination to replace the riser, tee, and flow control valves (several manufacturers currently have units which combine the tee and flow control valves). The shelter cost is included with the computers because a small shed would likely be built to house the components to help protect them from the environment.

Total system cost was found to be $30,594 (for the VIC system, which was the most versatile of the three), as compared to about $17,200 for a conventional gated-pipe system and about $58,000 for a center-pivot sprinkler system equipped with a corner unit (including added pump bowls for increased pressure requirements and higher pressure underground pipe).

Table 3. Estimate of Costs for an Automated System.

Qty.	Description	$/ea.	$ Total
2640'	14" buried pipe, installed	3.45/ft	9108.00
10	14" to 12" risers	150.00	1500.00
10	12" to 10" hydrants	210.00	2100.00
8	10" tees	67.50	540.00
18	10" flow control valves	250.00	4500.00
90	10" gated pipe, 30' sections	66.00	5940.00
1040	irrigation soks for 10" pipe	2.85	2964.00
18	10" end caps	18.50	333.00
2640'	12/3 UF cable	.22/ft	580.80
9	Southwood T8-24 transmitters	25.00	225.00
18	Southwood R8-24 receivers	48.00	864.00
18	24 VAC solenoid valves	46.00	828.00
	Misc. (connectors, wires, etc.)		180.00
	Subtotal		$29,662.80
	PLUS one of the following timers:		
3	Rainbird AG-7 Automatic Controllers	379.00	1137.00
	OR		
1	Syntertek Systems SYM-1 Single-Board Microcomputer	189.00	189.00
1	Power line filter w/ surge protection	69.95	69.95
	+ Interfacing components		80.00
	+ Shelter		500.00
	OR		
1	Commodore VIC-20 Microcomputer	97.00	97.00
1	Commodore Datasette recorder	72.97	72.97
1	Computer monitor	112.00	112.00
1	Power line filter w/ surge protection	69.95	69.95
	+ Interfacing components		80.00
	+ Shelter		500.00
	TOTAL (complete system using VIC)		$30,593.92

CONCLUSIONS

In review of the project objectives:

1. A gated-pipe irrigation system was successfully automated using wire telemetry controls.

2. Flexibility was provided in the controls through three separate means:

   a. A Rainbird Automatic Irrigation Controller.

   b. A Synertek Systems SYM-1 Microcomputer.

   c. A Commodore VIC-20 Microcomputer.

3. The performance of the wire telemetry controls was excellent, as the final design completed 100% of the scheduled changes.

4. The automated system reduced labor requirements substantially, and at a cost considerably less than the cost of converting to a sprinkler system.

SUMMARY

Decreasing water tables and dwindling streams have caused concern recently over the future of U.S. water supplies. As a result, surface irrigation and other low efficiency users of water are under pressure to decrease consumption. Automation is one way to increase the efficiency of surface irrigation while decreasing the labor requirements. This research investigated the use of wire telemetry in the automation of surface irrigation.

The objectives of this research were to automate a gated-pipe irrigation system using wire telemetry controls, to provide flexibility in the controls to allow more efficient irrigation methods, to evaluate the performance of the controls, and to evaluate the economic and labor-reducing potential of wire telemetry controls in gated-pipe irrigation systems.

A 65-ha (160-acre) field in southwestern Kansas was successfully irrigated using wire telemetry controls. The existing underground pipeline was used to deliver water to flow control valves. Each flow control valve had independent control over a complete irrigation set. These valves used pipeline water pressure to inflate a diaphragm, which would press against a lip in the valve body to effectively shut off the flow of water in the pipe. Three-way solenoid valves directed the flow of water into or out of the diaphragms.

Transmitters were located in a control station near one corner of the field. The transmitters superimposed high-frequency signals on 24 VAC transmission lines running along the upper end of the field. A receiver located at each flow control valve detected the appropriate signal and turned power on to the solenoid valve. The valves and receivers obtained their power from the 24 VAC transmission line.

An automatic irrigation controller was originally configured to control the system. It proved to be reliable but inflexible, and was replaced with two different microcomputers. A SYM-1 single-board microcomputer was programmed using assembly language and proved in laboratory tests to be more flexible, but allowed little feedback to the operator. A VIC 20 microcomputer was then programmed using BASIC and proved to be extremely flexible during field tests. The menu-driven program gave continuous feedback on system status, and virtually eliminated the need for an operator's manual. Further testing is needed to determine the long-term durability and reliability of this and similar devices.

Initial field tests were performed in August of 1983. Problems experienced with the wire telemetry components were attributed to faulty receivers. After replacement by the manufacturer, all the receivers worked satisfactorily. Other problems were experienced with sticking solenoid valves and unreliable timers. Timers were repaired and posed no further problems. The solenoids required annual maintenance which consisted of dismantling the valve to remove debris which collected during the irrigation season. The final system design completed 100% of the scheduled changes.

Labor requirements were found to be substantially reduced by automation. Annual labor requirements were estimated at 3.1, 2.0, and 0.9 man-hours per hectare (1.2, 0.8, and 0.4 man-hours per acre) for conventional gated-pipe, automated gated-pipe, and sprinkler systems, respectively. Much of the labor in conventional systems was due to operation time, while automated gated-pipe systems had a higher preparation time.

Automated surface systems have been found to increase efficiencies to values comparable to sprinkler irrigation, and at costs comparable to sprinkler systems. Although automation has been proven to be reliable, further tests are needed to determine durability in the environments that irrigation systems are normally exposed to.

SUGGESTIONS FOR FUTURE RESEARCH

Although field tests for wire telemetry controls were successful, additional development and testing is still needed before a commercial system could become available. The use of such controls appears to be feasible, but years of research will be required to determine their long-term reliability and durability.

With the research and development of any experimental control system will come equipment failures. The safety device used to protect this system was itself prone to inoperation. Further work needs to be done in the development of dependable, resettable safety devices for gated-pipe systems. The device should have as few moving parts as possible and should not have significant operation delays.

There are numerous ways to reduce the possibility of computer malfunction due to power failure. A simple battery back-up circuit similar to the one shown in Figure 12 would provide power to the entire system for up to one day in case of temporary power outages. The cost of such a circuit is minimal (<$1000) compared to the cost of the system, and would be a feature welcomed by most operators. Additional benefits would be gained by storing the program in ROM (Read-Only-Memory); it would eliminate the need for loading the program from tape.

For many applications it might be desirable to use a separate channel for each receiver rather than grouping them in

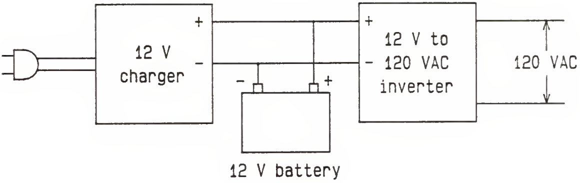


Figure 12. Suggested Battery Back-up Circuit.

pairs. The Rainbird controller used in this system would have been extremely difficult to implement using separate channels. The computers, however, could easily be adapted to this or most any other arrangement. With such a setup, only two transmission wires would be needed, and the two relays on the transmitter panel could be eliminated. The computer control program would be slightly more complex, but the remainder of the system (the only parts the operator would be concerned with) would be somewhat simpler to understand.

Haise et al. (1980) contended that farmers will need to exercise special caution in the care and maintenance of automation equipment, and until they learn to do so, will have to be content with their present labor-intensive systems. One must keep in mind that few farmers will change their habits to meet the needs of special systems; rather, successful systems will adapt to the needs and habits of farmers. Durability, reliability, and simplicity will be the key selling points of any successful automated system.

REFERENCES

Bishop, A. A., W. R. Walker, N. L. Allen and G. J. Poole. 1981. Furrow advance rates under surge flow systems. Journal of the Irrigation and Drainage Division, ASCE 107 (IR3): 257-264.

Blume, H. R. 1979. Radio controls for gated pipe irrigation systems. Unpublished M. S. Thesis. Kansas State University Library. Manhattan, Kansas.

Bowman, C. C. 1969. Radio controls for automating irrigation. Agricultural Engineering 50 (5): 305.

Coolidge, P. S., W. R. Walker and A. A. Bishop. 1982. Advance and runoff - surge flow irrigation. Journal of the Irrigation and Drainage Division, ASCE 108 (IR1): 35-41.

Edling, R. J., H. R. Duke and M. L. Payne. 1978. Utilization of a portable solid state furrow irrigation valve controller. ASAE Paper No. 78-2546. Presented at the ASAE Winter Meeting. Chicago, Illinois

Fischbach, P. E., T. L. Thompson, and L. E. Stetson. 1970. Electric controls for automatic surface irrigation system with reuse system. Transactions of the ASAE 13 (3): 286-288.

Fischbach, P. E. and R. Goodding II. 1971. An automated surface irrigation valve. Agricultural Engineering 52 (11): 584-585.

Fischbach, P. E. and B. R. Somerhalder. 1971. Efficiencies of an automated surface irrigation system with and without a runoff reuse system. Transactions of the ASAE 14 (4): 717-719.

Fisher, H. D., A. S. Humpherys and R. V. Worstell. 1978. Electronic timers for automated surface irrigation systems. ASAE Paper No. 78-2544. Presented at the ASAE Winter Meeting. Chicago, Illinois

Garton, J. E. 1966. Designing an automatic cutback furrow irrigation system. Oklahoma Agricultural Experiment Station Bulletin B-651.

Haise, H. R., E. G. Kruse and N. A. Dimick. 1965. Pneumatic valves for automation of irrigation systems. ARS 41-104. U.S. Department of Agriculture.

Haise, H. R., E. G. Kruse and L. Erie. 1969. Automatic surface irrigation. Agricultural Engineering 50 (4): 212-216.

Haise, H. R., E. G. Kruse, M. L. Payne and H. R. Duke. 1980. Automation of surface irrigation: 15 years of USDA research and development at Fort Collins, Colorado. Production Research Report No. 179, U.S. Department of Agriculture.

Haise, H. R. and M. L. Payne. 1969. Self-closing irrigation pipe valve. Journal of the Irrigation and Drainage Division, ASCE 95 (IR4): 503-516.

Harrer, B. J. and G. L. Wilfert. 1983. Technical descriptions of ten irrigation technologies for conserving energy. Pacific Northwest Laboratory, Battelle Memorial Institute.

Humpherys, A. S. and R. L. Stacey. 1975. Automated valves for surface irrigation pipeline systems. Journal of the Irrigation and Drainage Division, ASCE 101 (IR2): 95-109.

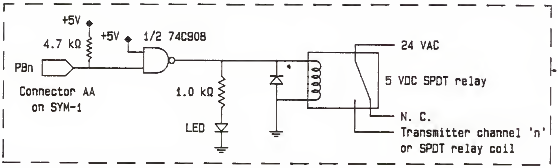
Lamb, J. A., G. E. Meyer and P. E. Fischbach. 1982. Microprocessor automation of a solid-set irrigation facility for research and demonstration. ASAE Paper No. 82-2533. Presented at the 1982 Winter Meeting. Chicago, Illinois.

Lillevik, S. L. 1982. Discrete-time, cutback furrow irrigation. Transactions of the ASAE 25 (6): 1646-1650.

Manges, H. L., D. K. Matteson and K. G. Butler. 1980. Radio controls for gated-pipe irrigation. ASAE Paper No. 80-2557. Presented at the ASAE Winter Meeting. Chicago, Illinois.

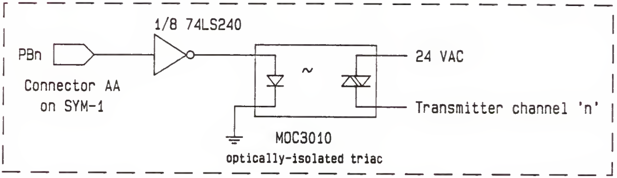
Pair, C. H. 1961. Automation near in irrigation. Agricultural Engineering 42 (11): 608-610.

APPENDIX A: INTERFACING DIAGRAMS



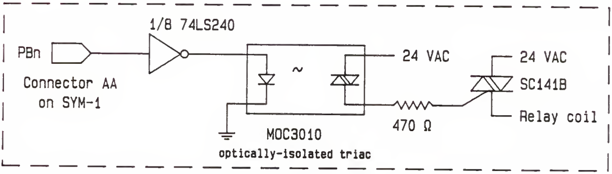
X 8 (one for each of 6 transmitter channels
plus one for each of 2 relays)

Figure 13. Relay Interface Board Schematic.



X 6 (one for each of 6 transmitter channels)

Figure 14a. Triac Interface Board Schematic for Transmitters.



X 2 (one for each of 2 relays)

Figure 14b. Triac Interface Board Schematic for Relay Coils.

APPENDIX B: ASSEMBLED SOURCE CODE FOR SYM-1

```
0010 ;FILE "IRRIG.C"
0020
0025 ;CONTROL FILE FOR ASSEMBLER.
0026
0030            .CE
0040            .CT
0050            .OS
0060            .ES
0070            .PR "BEGINNING ADDRESS AT $C000"
0080            .FI "SYM.L"
```

0298 30FC-3394 SYM.L

```
0010 ;FILE "SYM.L"
0020
0030 ;MONITOR ADDRESSES FOR SYM
0040
0050 ACCESS     .DE $8B86
0060 BLKMOV     .DE $8740
0070 CONFIG     .DE $89A5
0080 FILL3      .DE $8718
0090 GK         .DE $88CF
0100 INBYTE     .DE $81D9
0110 INCHR      .DE $8A1B
0120 KSCONF     .DE $89A3
0130 L2         .DE $84D3
0140 MONITOR    .DE $8003
0150 NACCESS    .DE $8B9C
0160 OUTBYT     .DE $82FA
0170 OUTCHR     .DE $8A47
0180 OUTNIB     .DE $8A44
0190 OUTXAH     .DE $82F4
0200 PARM       .DE $8220
0210 RESALL     .DE $81C4
0220 RESTART    .DE $8000
0230 SAVE2      .DE $87EA
0240 SAVER      .DE $8188
0250 SCAND      .DE $8906
0260 SPACE      .DE $8342
0270 VECSW      .DE $8BB7
0280
0290 DISBUF     .DE $A640
0300 GOVEC      .DE $A659
0310 ID         .DE $A64E
0320 INTVEC     .DE $A678
0330 PARN       .DE $A64A
0340 PARNR      .DE $A649
0350 RMDIG      .DE $A645
```

```
0360 SDBYT        .DE $A651
0370 TAPDEL       .DE $A630
0380
0390 VIA1         .DE $A000
0400 VIA2         .DE $A800
0410 VIA3         .DE $AC00
0420
0430 ;FILE "SYM.L"
0090              .BA $00
0100              .FI "ZPVARS.B"
```

```
03FC  30FC-34F8  ZPVARS.B

               0010 ;FILE "ZPVARS.B"
               0020
               0030 ;ZERO PAGE VARIABLES
               0040
               0050 ;VARIABLES FOR GENERAL PROGRAM USE
0000-          0060 VERIFY      .DS 2          ;$A55A ON RESTART
0002-          0070 SCRATCH     .DS 1          ;USED BY CHANGE ROUTINE
0003-          0080 PORTDATA    .DS 1          ;OUTPUT PORT ON VIA2
0004-          0090 COUNTER     .DS 1          ;VALVE INDEX COUNTER
0005-          0100 TOTAL       .DS 1          ;TOTAL # OF VALVES INPUT
0006-          0110 FLAG        .DS 1          ;$FF FOR ON, $00 FOR OFF
0007-          0120 MASK        .DS 1          ;$FF FOR MIN, $00 FOR SEC
0008-          0130 DAY         .DS 6          ;USED BY CLOCK ROUTINES
               0140
               0150 ;VARIABLES GENERATED BY INPUT ROUTINE
000E-          0160 VALVE       .DS 10         ;VALVE #'S IN ORDER
0018-          0170 CYCLES      .DS 10         ;# OF SURGE CYCLES INPUT
0022-          0180 TSRGHRS     .DS 10         ;TIME (HRS) ON SURGE
002C-          0190 TSRGMNS     .DS 10         ;TIME (MINS) FOR SURGE
0036-          0200 TCUTHRS     .DS 10         ;TIME (HRS) FOR CUTBACK
0040-          0210 TCUTMNS     .DS 10         ;TIME (MINS) FOR CUTBACK
               0220
               0230 ;COUNTERS - DECREMENTED TO ZERO
004A-          0240 MINCNT      .DS 1          ;MINUTES REMAINING
004B-          0250 HRSCNT      .DS 1          ;HOURS REMAINING
004C-          0260 CYCCNT      .DS 1          ;CYCLES REMAINING
               0270
               0280 ;FILE "ZPVARS.B"
               0110             .BA $C000
               0120             .MC $1000
               0130             .FI "MAININ.S"
```

12C3 30FC-43BF MAININ.S

```
                  0010 ;FILE "MAININ.S"
                  0020
                  0030 MAIN
C000- 20 1D C0    0040                JSR INIT             ;INITIALIZE
C003- A5 00       0050                LDA *VERIFY          ;RESTART OR RESET
C005- C9 A5       0060                CMP #$A5
C007- D0 0A       0070                BNE @MAIN1
C009- A5 01       0080                LDA *VERIFY+1
C00B- C9 5A       0090                CMP #$5A
C00D- D0 04       0100                BNE @MAIN1
C00F- 58          0110                CLI
C010- 4C E1 C1    0120                JMP REBEGIN          ;MUST NOT BE A RESET
C013- 20 2B C0    0130 @MAIN1         JSR INIT1            ;RESET, SO COLD BOOT
C016- 58          0140                CLI
C017- 20 4B C0    0150                JSR INPUT
C01A- 4C 23 C1    0160                JMP SCAN
                  0170
                  0180 INIT
C01D- A9 3A       0190                LDA #L,INTERRUPT     ;STORE INTERRUPT VECTOR
C01F- 8D 78 A6    0200                STA INTVEC
C022- A9 C2       0210                LDA #H,INTERRUPT
C024- 8D 79 A6    0220                STA INTVEC+1
C027- 20 57 C3    0230                JSR SETUPCLOCK
C02A- 60          0240                RTS
                  0250
                  0260 INIT1
C02B- A2 7F       0270                LDX #$7F
C02D- A9 00       0280                LDA #0
C02F- CA          0290 @MEMINIT       DEX                  ;INITIALIZE MEMORY
C030- 95 00       0300                STA *$00,X
C032- D0 FB       0310                BNE @MEMINIT
C034- A9 FF       0320                LDA #$FF
C036- 8D 00 A8    0330                STA VIA2             ;SETS REGISTER TO ALL 1'S
C039- 8D 02 A8    0340                STA VIA2+2           ;SETS REGISTER AS OUTPUT
C03C- 85 03       0345                STA *PORTDATA        ;SETS TO ALL 1'S
C03E- A9 00       0350                LDA #$00
C040- 85 06       0360                STA *FLAG            ;STOPS TIMER
C042- A9 A5       0370                LDA #$A5
C044- 85 00       0380                STA *VERIFY
C046- A9 5A       0390                LDA #$5A
C048- 85 01       0400                STA *VERIFY+1
C04A- 60          0410                RTS
                  0420
                  0430 INPUT
C04B- A9 20       0440                LDA #'
C04D- 20 47 8A    0450                JSR OUTCHR
C050- 20 47 8A    0460                JSR OUTCHR
C053- 20 47 8A    0470                JSR OUTCHR
C056- A2 41       0480                LDX #'A              ;OUTPUT 'A' AS PROMPT
C058- 20 0F C1    0490                JSR READY
C05B- 20 CF 88    0500 @MASK          JSR GK
```

```
C05E- E0 00                CPX #$00          ;IF ZERO ENTERED
C060- D0 05                BNE @ONE
C062- 86 07                STX *MASK         ;MASK = 0; TIME IN SEC
C064- 4C 6F C0             JMP @OK
C067- E0 01     @ONE       CPX #$01          ;IF ONE ENTERED
C069- D0 F0                BNE @MASK
C06B- A2 FF                LDX #$FF
C06D- 86 07                STX *MASK         ;MASK = $FF; TIME IN MIN
C06F- 20 47 8A  @OK        JSR OUTCHR
C072- 20 E3 C0             JSR SEQ           ;GETS VALVE #'S IN SEQ.
C075- A9 00                LDA #0
C077- 85 02                STA *SCRATCH      ;START AT BEGINNING
C079- 20 7D C0             JSR INDATA        ;GETS REMAINING DATA
C07C- 60                   RTS

                INDATA
C07D- A2 43                LDX #'C           ;INPUT CYCLES DATA
C07F- 20 0F C1             JSR READY
C082- 20 D9 81             JSR INBYTE        ;ACCEPTS 2 DIGITS
C085- 20 7C C3             JSR BINARY        ;CONVERT TO BINARY
C088- 0A                   ASL A             ;MULT. BY 2 FOR CYCLE COUN
C089- A6 02                LDX *SCRATCH
C08B- 95 18     @CYCLES    STA *CYCLES,X
C08D- E8                   INX
C08E- E4 05                CPX *TOTAL        ;TOTAL = # OF TIMES THRU
C090- 30 F9                BMI @CYCLES
C092- A2 44                LDX #'D           ;INPUT SURGE HOURS
C094- 20 0F C1             JSR READY
C097- 20 D9 81             JSR INBYTE
C09A- 20 7C C3             JSR BINARY
C09D- A6 02                LDX *SCRATCH
C09F- 95 22     @TSHRS     STA *TSRGHRS,X
C0A1- E8                   INX
C0A2- E4 05                CPX *TOTAL
C0A4- 30 F9                BMI @TSHRS
C0A6- A9 2D                LDA #'-           ;INPUT SURGE MINUTES
C0A8- 20 47 8A             JSR OUTCHR
C0AB- 20 D9 81             JSR INBYTE
C0AE- 20 7C C3             JSR BINARY
C0B1- A6 02                LDX *SCRATCH
C0B3- 95 2C     @TSMIN     STA *TSRGMNS,X
C0B5- E8                   INX
C0B6- E4 05                CPX *TOTAL
C0B8- 30 F9                BMI @TSMIN
C0BA- A2 45                LDX #'E           ;INPUT CUTBACK HOURS
C0BC- 20 0F C1             JSR READY
C0BF- 20 D9 81             JSR INBYTE
C0C2- 20 7C C3             JSR BINARY
C0C5- A6 02                LDX *SCRATCH
C0C7- 95 36     @TCHRS     STA *TCUTHRS,X
C0C9- E8                   INX
C0CA- E4 05                CPX *TOTAL
C0CC- 30 F9                BMI @TCHRS
```

```
C0CE- A9 2D     1040             LDA #'-         ;INPUT CUTBACK MINUTES
C0D0- 20 47 8A  1050             JSR OUTCHR
C0D3- 20 D9 81  1060             JSR INBYTE
C0D6- 20 7C C3  1070             JSR BINARY
C0D9- A6 02     1080             LDX *SCRATCH
C0DB- 95 40     1090 @TCMIN      STA *TCUTMNS,X
C0DD- E8        1100             INX
C0DE- E4 05     1110             CPX *TOTAL
C0E0- 30 F9     1120             BMI @TCMIN
C0E2- 60        1130             RTS
                1140
                1150 SEQ
C0E3- A2 42     1160             LDX #'B         ;INPUT VALVE SEQUENCE
C0E5- 20 0F C1  1170             JSR READY
C0E8- A0 00     1180             LDY #0
C0EA- 84 05     1190             STY *TOTAL
C0EC- 20 CF 88  1200 @NEXT       JSR GK
C0EF- E0 0E     1210             CPX #$0E        ;ENTER KEY PRESSED?
C0F1- F0 1B     1220             BEQ @ENDSEQ
C0F3- E0 07     1230             CPX #7
C0F5- 10 F5     1240             BPL @NEXT       ;IF VALVE # > 6, INVALID
C0F7- A4 05     1250             LDY *TOTAL
C0F9- 96 0E     1260             STX *VALVE,Y
C0FB- E6 05     1270             INC *TOTAL
C0FD- 20 47 8A  1280             JSR OUTCHR
C100- A2 09     1290             LDX #$09
C102- E4 05     1300             CPX *TOTAL      ;IF 9 VALVE #'S ENTERED
C104- 30 08     1310             BMI @ENDSEQ     ;END
C106- A9 2D     1320             LDA #'-         ;OUTPUT DASH, GET ANOTHER
C108- 20 47 8A  1330             JSR OUTCHR
C10B- 4C EC C0  1340             JMP @NEXT
C10E- 60        1350 @ENDSEQ     RTS
                1360
                1370 READY
C10F- A9 20     1380             LDA #'          ;OUTPUTS PROMPT
C111- 20 47 8A  1390             JSR OUTCHR
C114- A9 2E     1400             LDA #'.
C116- 20 47 8A  1410             JSR OUTCHR
C119- 8A        1420             TXA
C11A- 20 47 8A  1430             JSR OUTCHR
C11D- A9 2E     1440             LDA #'.
C11F- 20 47 8A  1450             JSR OUTCHR
C122- 60        1460             RTS
                1470
                1480 SCAN
C123- A9 00     1490             LDA #$00
C125- 85 06     1500             STA *FLAG
C127- A9 FF     1503             LDA #$FF
C129- 8D 00 A8  1506             STA VIA2        ;SETS OUTPUTS AS 1'S
C12C- 20 86 8B  1510             JSR ACCESS
C12F- 78        1520             SEI             ;DISABLE INT. WHILE INIT.
C130- 20 2F C2  1530             JSR INITCTRS    ;INITIALIZE COUNTERS
C133- 58        1540             CLI
```

```
C134- A9 20     1550          LDA #'          ;OUTPUT " dead."
C136- 20 47 8A  1560          JSR OUTCHR
C139- A9 44     1570          LDA #'D
C13B- 20 47 8A  1580          JSR OUTCHR
C13E- A9 45     1590          LDA #'E
C140- 20 47 8A  1600          JSR OUTCHR
C143- A9 41     1610          LDA #'A
C145- 20 47 8A  1620          JSR OUTCHR
C148- A9 44     1630          LDA #'D
C14A- 20 47 8A  1640          JSR OUTCHR
C14D- A9 2E     1650          LDA #'.
C14F- 20 47 8A  1660          JSR OUTCHR
C152- 4C 7B C1  1670          JMP @HOLD
                1680 HOLDING
C155- A9 00     1690          LDA #$00
C157- 85 06     1700          STA *FLAG
C159- A9 FF     1703          LDA #$FF
C15B- 8D 00 A8  1706          STA VIA2        ;SETS OUTPUTS AT 1'S
C15E- 20 86 8B  1710          JSR ACCESS
C161- A9 20     1720          LDA #'          ;OUTPUT "  off."
C163- 20 47 8A  1730          JSR OUTCHR
C166- 20 47 8A  1740          JSR OUTCHR
C169- A9 30     1750          LDA #'O
C16B- 20 47 8A  1760          JSR OUTCHR
C16E- A9 46     1770          LDA #'F
C170- 20 47 8A  1780          JSR OUTCHR
C173- 20 47 8A  1790          JSR OUTCHR
C176- A9 2E     1800          LDA #'.
C178- 20 47 8A  1810          JSR OUTCHR
C17B- 20 CF 88  1820 @HOLD    JSR GK
C17E- E0 11     1830          CPX #$11        ;MONITOR KEY?
C180- F0 52     1840          BEQ GOMON       ;JUMPS TO MONITOR
C182- E0 0A     1850          CPX #$0A        ;CHANGE KEY?
C184- F0 72     1860          BEQ CHANGE      ;GETS NEW DATA FOR A VALVE
C186- E0 0B     1870          CPX #$0B        ;RSQ KEY?
C188- F0 68     1880          BEQ RESEQ       ;GETS NEW VALVE #'S IN SEQ
C18A- E0 0C     1890          CPX #$0C        ;KILL KEY?
C18C- F0 95     1900          BEQ SCAN        ;RESTARTS CONTROL SEQ. AT
C18E- E0 0F     1910          CPX #$0F        ;GO KEY?
C190- F0 03     1920          BEQ @TIMING     ;IF SO, START TIMING
C192- 4C 7B C1  1930          JMP @HOLD       ;ELSE, HOLD MORE
C195- A9 FF     1940 @TIMING  LDA #$FF
C197- 85 06     1945          STA *FLAG
C199- A5 03     1950          LDA *PORTDATA
C19B- 8D 00 A8  1955          STA VIA2        ;SETS APPROPRIATE OUTPUTS
C19E- 20 86 8B  1957          JSR ACCESS
C1A1- A9 FF     1960          LDA #$FF
C1A3- 85 06     1970          STA *FLAG
C1A5- A9 20     1980          LDA #'          ;OUTPUT "   on."
C1A7- 20 47 8A  1990          JSR OUTCHR
C1AA- 20 47 8A  2000          JSR OUTCHR
C1AD- 20 47 8A  2010          JSR OUTCHR
C1B0- A9 30     2020          LDA #'O
```

```
C1B2- 20 47 8A  2030           JSR OUTCHR
C1B5- A9 4D     2040           LDA #'M
C1B7- 20 47 8A  2050           JSR OUTCHR
C1BA- A9 2E     2060           LDA #'.
C1BC- 20 47 8A  2070           JSR OUTCHR
C1BF- 20 CF 88  2080           JSR GK              ;GET A KEY
C1C2- E0 11     2090           CPX #$11            ;MONITOR KEY?
C1C4- F0 0E     2100           BEQ GOMON           ;JUMPS TO MONITOR
C1C6- E0 0C     2110           CPX #$0C            ;KILL KEY?
C1C8- F0 07     2120           BEQ @SC             ;GO TO SCAN
C1CA- E0 0D     2160           CPX #$0D            ;HOLD KEY?
C1CC- F0 87     2165           BEQ HOLDING         ;GO AND HOLD
C1CE- 4C 95 C1  2170           JMP @TIMING         ;ELSE, CONTINUE TIMING
C1D1- 4C 23 C1  2180 @SC       JMP SCAN
                2210
                2220 GOMON
C1D4- A9 E1     2230           LDA #L,REBEGIN      ;GO TO SYM MONITOR
C1D6- 8D 59 A6  2240           STA GOVEC
C1D9- A9 C1     2250           LDA #H,REBEGIN
C1DB- 8D 5A A6  2260           STA GOVEC+1
C1DE- 4C 03 80  2270           JMP MONITOR
                2280
                2290 REBEGIN
C1E1- A5 03     2292           LDA *PORTDATA       ;RESET WAS PRESSED ...
C1E3- 8D 00 A8  2294           STA VIA2
C1E6- A9 FF     2296           LDA #$FF
C1E8- 8D 02 A8  2298           STA VIA2+2
C1EB- A5 06     2300           LDA *FLAG
C1ED- D0 A6     2310           BNE @TIMING         ;TIMING IS ON
C1EF- 4C 55 C1  2320           JMP HOLDING         ;TIMING IS OFF
                2330
                2340 RESEQ
C1F2- 20 E3 C0  2350           JSR SEQ             ;INPUT VALVE SEQ. AGAIN
C1F5- 4C 23 C1  2360           JMP SCAN            ;GO BACK TO BEGINNING
                2370
                2380 CHANGE
C1F8- A2 46     2390           LDX #'F
C1FA- 20 0F C1  2400           JSR READY           ;PRINTS .F. TO PROMPT
C1FD- 20 CF 88  2410 @B1       JSR GK              ;GET A KEY
C200- E0 0E     2420           CPX #$0E            ;ENTER KEY?
C202- F0 28     2430           BEQ @ENDCHG
C204- E0 07     2440           CPX #7              ;V # MUST BE <=6
C206- 10 F5     2450           BPL @B1             ;>6, SO NOT VALID ENTRY
C208- A8        2460           TAY                 ;PUTS ASCII IN Y
C209- 8A        2470           TXA                 ;PUTS V # IN ACC
C20A- A2 00     2480           LDX #0
C20C- CA        2490           DEX
C20D- E8        2500 @FIND     INX
C20E- D5 0E     2510           CMP *VALVE,X        ;FIND THAT VALVE # IN TABLE
C210- F0 07     2520           BEQ @FOUND
C212- E4 05     2530           CPX *TOTAL
C214- 30 F7     2540           BMI @FIND
C216- 4C FD C1  2550           JMP @B1
```

```
C219- 98          2560 @FOUND      TYA                      ;FOUND VALVE #, CHANGE DATA
C21A- 20 47 8A    2570             JSR OUTCHR
C21D- 86 02       2580             STX *SCRATCH
C21F- A5 05       2590             LDA *TOTAL
C221- 48          2600             PHA
C222- A9 01       2610             LDA #1
C224- 85 05       2620             STA *TOTAL
C226- 20 7D C0    2630             JSR INDATA
C229- 68          2640             PLA
C22A- 85 05       2650             STA *TOTAL
C22C- 4C 23 C1    2660 @ENDCHG     JMP SCAN                 ;GO BACK TO SCANNING KEYBD
                  2670
                  2680 INITCTRS
C22F- A9 00       2690             LDA #0                   ;INITIALIZE COUNTERS
C231- 85 4A       2700             STA *MINCNT
C233- 85 4C       2710             STA *CYCCNT
C235- 85 4B       2720             STA *HRSCNT
C237- 85 04       2730             STA *COUNTER
C239- 60          2740             RTS
                  2750
                  2760 ;FILE "MAININ.S"
                  0140             .FI "INTERRUPT.S"
```

```
08BF  30FC-39BB  INTERRUPT.S

                  0010 ;FILE "INTERRUPT.S"
                  0020
                  0030 INTERRUPT
C23A- 48          0040             PHA                      ;OCCURS ONCE PER SECOND
C23B- 8A          0050             TXA
C23C- 48          0060             PHA
C23D- 98          0070             TYA
C23E- 48          0080             PHA
C23F- 20 F1 C2    0090             JSR READCLOCK
C242- A9 FF       0100             LDA #$FF
C244- 8D 02 A8    0110             STA VIA2+2               ;ASSURES PORT IS OUTPUT
C247- A5 06       0120             LDA *FLAG
C249- D0 08       0130             BNE @INTGO
C24B- A9 FF       0140             LDA #$FF
C24D- 8D 00 A8    0150             STA VIA2
C250- 4C 61 C2    0160             JMP @ENDINT
C253- A5 0D       0170 @INTGO      LDA *DAY+5               ;LOCATION OF SECONDS
C255- 25 07       0180             AND *MASK                ;MASK IS $FF FOR MIN. AND
C257- D0 03       0190             BNE @NOTTIME
C259- 20 67 C2    0200             JSR EVERYMIN             ;ONLY IF SEC = 0
C25C- A5 03       0210 @NOTTIME    LDA *PORTDATA
C25E- 8D 00 A8    0220             STA VIA2                 ;OUTPUT PORT
C261- 68          0230 @ENDINT     PLA                      ;RESTORE REGISTERS
```

```
C262- A8          0240              TAY
C263- 68          0250              PLA
C264- AA          0260              TAX
C265- 68          0270              PLA
C266- 40          0280              RTI
                  0290
                  0300 EVERYMIN
C267- C6 4A       0310              DEC *MINCNT
C269- 10 23       0320              BPL @ENDMIN        ;IF MINUTES NOT UP, RTI
C26B- C6 4B       0330              DEC *HRSCNT        ;MIN = 0
C26D- 30 07       0340              BMI @DECCYC        ;IF HOURS  = 0, DEC CYCLES
C26F- A9 3C       0350              LDA #60
C271- 85 4A       0360              STA *MINCNT        ;PUTS 60 IN MIN COUNTER
C273- 4C 67 C2    0370              JMP EVERYMIN
C276- C6 4C       0380 @DECCYC      DEC *CYCCNT
C278- 10 06       0390              BPL @SAMEVLV       ;IF CYCLES NOT UP, STAY ON
C27A- 20 A6 C2    0400              JSR NEWVALVE       ;ELSE, TURN ON NEW VALVE
C27D- 4C 67 C2    0410              JMP EVERYMIN
C280- F0 06       0420 @SAMEVLV     BEQ @CUTBK         ;IF CYCLE COUNT = 0, THEN
C282- 20 94 C2    0430              JSR TOGGLE         ;CYCLES NOT 0
C285- 4C 67 C2    0440              JMP EVERYMIN
C288- 20 D9 C2    0450 @CUTBK       JSR CUTBACK        ;OPENS BOTH
C28B- 4C 67 C2    0460              JMP EVERYMIN
C28E- A9 07       0470 @ENDMIN      LDA #7             ;BEEPS
C290- 20 47 8A    0480              JSR OUTCHR
C293- 60          0490              RTS
                  0500
                  0510 TOGGLE
C294- A5 03       0520              LDA *PORTDATA      ;OUTPUT VARIABLE
C296- 49 40       0530              EOR #%01000000     ;TOGGLE X BIT
C298- 85 03       0540              STA *PORTDATA      ;PUT BACK OUT
C29A- A4 04       0550              LDY *COUNTER
C29C- 88          0560              DEY                ;GETS LOCATION OF CURRENT
C29D- B6 2C       0570              LDX *TSRGMNS,Y     ;LOAD MINUTES
C29F- 86 4A       0580              STX *MINCNT
C2A1- B6 22       0590              LDX *TSRGHRS,Y     ;LOAD HOURS
C2A3- 86 4B       0600              STX *HRSCNT
C2A5- 60          0610              RTS
                  0620
                  0630 NEWVALVE
C2A6- A4 04       0640              LDY *COUNTER       ;GETS CURRENT VALVE #
C2A8- C4 05       0650              CPY *TOTAL
C2AA- 10 16       0660              BPL @DONE          ;IF TOTAL = COUNTER
C2AC- E6 04       0670              INC *COUNTER       ;GO TO NEXT VALVE
C2AE- B6 0E       0680              LDX *VALVE,Y       ;LOADS CURRENT VALVE #
C2B0- 20 C7 C2    0690              JSR NEWBIT
C2B3- B6 22       0700              LDX *TSRGHRS,Y     ;LOAD VALVE DATA
C2B5- 86 4B       0710              STX *HRSCNT
C2B7- B6 2C       0720              LDX *TSRGMNS,Y
C2B9- 86 4A       0730              STX *MINCNT
C2BB- B6 18       0740              LDX *CYCLES,Y
C2BD- 86 4C       0750              STX *CYCCNT
C2BF- F0 18       0760              BEQ CUTBACK        ;IF CYCLES = 0
```

```
C2C1- 60          0770               RTS
C2C2- A9 00       0780 @DONE         LDA #$00
C2C4- 85 06       0790               STA *FLAG
C2C6- 60          0800               RTS
                  0810
                  0820 NEWBIT
C2C7- A9 00       0830               LDA #0
C2C9- E0 00       0840               CPX #0
C2CB- F0 05       0850               BEQ @NOVALVE
C2CD- 38          0860               SEC
C2CE- 2A          0870 @AGAIN        ROL A                ;NEXT VALVE
C2CF- CA          0880               DEX
C2D0- D0 FC       0890               BNE @AGAIN           ;LOOP X NUMBER OF TIMES
C2D2- 49 3F       0900 @NOVALVE      EOR #%00111111       ;TOGGLE #'D BITS FOR ON=0
C2D4- 09 C0       0910               ORA #%11000000       ;SET X & Y BITS
C2D6- 85 03       0920               STA *PORTDATA
C2D8- 60          0930               RTS
                  0940
                  0950 CUTBACK
C2D9- A4 04       0960               LDY *COUNTER
C2DB- 88          0970               DEY
C2DC- A2 00       0980               LDX #0               ;RESET COUNTERS
C2DE- 86 4C       0990               STX *CYCCNT
C2E0- B6 36       1000               LDX *TCUTHRS,Y
C2E2- 86 4B       1010               STX *HRSCNT
C2E4- B6 40       1020               LDX *TCUTMNS,Y
C2E6- 86 4A       1030               STX *MINCNT
C2E8- A5 03       1040               LDA *PORTDATA
C2EA- 29 7F       1050               AND #%01111111       ;CLEAR Y BIT (TURNS Y ON)
C2EC- 09 40       1060               ORA #%01000000       ;SET X BIT (TURNS X OFF)
C2EE- 85 03       1070               STA *PORTDATA
C2F0- 60          1080               RTS
                  1090
                  1100 ;FILE "INTERRUPT.S"
                  0150               .FI "CLOCK.S"
```

```
0666  30FC-3762  CLOCK.S

                  0001 ;FILE "CLOCK.S"
                  0002
                  0079 ;SUBROUTINE TO READ TIME FROM CLOCK
                  0080 ;AND STORE IT IN 6 BYTES AT DAY
                  0081
                  0082 READCLOCK
C2F1- A9 06       0083               LDA #$06             ;HOLD & READ
C2F3- 20 34 C3    0084               JSR @STOPCLOCK
C2F6- A2 05       0085               LDX #5
C2F8- A9 00       0086               LDA #0
```

```
C2FA- 8D 01 A0                STA VIA1+1
C2FD- 20 15 C3                JSR @READ3          ;READ HH:MM:SS
C300- EE 01 A0                INC VIA1+1          ;SKIP DAY OF WK
C303- 20 15 C3                JSR @READ3          ;READ YY:MM:DD
C306- A5 0B                   LDA *DAY+3          ;CLEAR 24 HR FLAG
C308- 29 3F                   AND #$3F
C30A- 85 0B                   STA *DAY+3
C30C- A5 0A                   LDA *DAY+2          ;CLEAR LEAPYEAR FLAG
C30E- 29 3F                   AND #$3F
C310- 85 0A                   STA *DAY+2
C312- 4C 57 C3                JMP SETUPCLOCK

               ;READ 3 BYTES OF DATA FROM THE CLOCK
               ;AND PUT IT AT DAY,X TO DAY,X-2
               ;LEAVE X = X-3.

               @READ3
C315- A0 03                   LDY #$03
C317- AD 01 A0                LDA VIA1+1
C31A- EE 01 A0                INC VIA1+1          ;NEXT NIBBLE
C31D- 4A                      LSR A
C31E- 4A                      LSR A
C31F- 4A                      LSR A
C320- 4A                      LSR A
C321- 95 08                   STA *DAY,X          ;LOWER NIBBLE
C323- AD 01 A0                LDA VIA1+1
C326- EE 01 A0                INC VIA1+1          ;NEXT NIBBLE
C329- 29 F0                   AND #%11110000      ;HIGH NIBBLE
C32B- 15 08                   ORA *DAY,X          ;MERGE NIBBLES
C32D- 95 08                   STA *DAY,X
C32F- CA                      DEX
C330- 88                      DEY
C331- D0 E4                   BNE @READ3+2
C333- 60                      RTS

               ;THIS ROUTINES DISABLES THE INTERRUPTS
               ;AND SETS THE OUTPUT LINES AS SET IN A
               ;2 PB IS SET TO OUTPUT IF WRITE FLAG IS SET

               @STOPCLOCK
C334- 78                      SEI
C335- 85 08                   STA *DAY
C337- A9 10                   LDA #$10            ;DISABLE CB1 INT
C339- 8D 0E A0                STA VIA1+14         ;DISABLE INTERRUPTS
C33C- AD 00 A0                LDA VIA1
C33F- 09 07                   ORA #$07            ;SET STATUS BITS
C341- 45 08                   EOR *DAY            ;INV NEW STAT BITS
C343- 8D 00 A0                STA VIA1
C346- A9 02                   LDA #$02            ;READ BIT
C348- 24 08                   BIT *DAY            ;SET FOR READ?
C34A- D0 05                   BNE =+6
C34C- A9 FF                   LDA #$FF
C34E- 8D 03 A0                STA VIA1+3          ;ALL OUTPUTS
```

```
C351- A2 1E                LDX #30          ;DELAY FOR HOLD SETUP
C353- CA                   DEX
C354- D0 FD                BNE =-2
C356- 60                   RTS

            ;ROUTINE TO SETUP THE VIA'S FOR THE CLOCK
            ;SUBROUTINES

            SETUPCLOCK
C357- AD 00 A0             LDA VIA1
C35A- 29 F8                AND #%11111000   ;CLR STAT
C35C- 09 05                ORA #%00000101   ;SET NEG STAT
C35E- 8D 00 A0             STA VIA1         ;SET READ ONLY MODE
C361- AD 02 A0             LDA VIA1+2
C364- 09 07                ORA #%00000111   ;STAT BITS OUT
C366- 8D 02 A0             STA VIA1+2
C369- A9 0F                LDA #$0F
C36B- 8D 01 A0             STA VIA1+1       ;SET FOR INTERRUPTS
C36E- 8D 03 A0             STA VIA1+3       ;DATA DIRECTION REG.
C371- A9 10                LDA #$10         ;CB1 INT
C373- 8D 0D A0             STA VIA1+13      ;RESET INTERUPTS
C376- A9 90                LDA #$90         ;CB1 INT
C378- 8D 0E A0             STA VIA1+14      ;ENABLE INTERRUPTS
C37B- 60                   RTS

            ;FILE "CLOCK.S"
                           .FI "BINARY.S"
```

008D 30FC-3189 BINARY.S

```
            BINARY

            ;CONVERTS 2 ASCII HEX DIGITS IN 'A'
            ;TO ONE HEX BYTE

C37C- 48                   PHA
C37D- 4A                   LSR A
C37E- 4A                   LSR A
C37F- 4A                   LSR A
C380- 4A                   LSR A
C381- AA                   TAX
C382- 68                   PLA
C383- CA                   DEX
C384- 10 01                BPL =+2
C386- 60                   RTS
C387- 38                   SEC
C388- E9 06                SBC #$06
```

```
C38A- B0 F7     0150            BCS =-8             ;ALWAYS
                0160
                0170 ;FILE "BINARY.S"
                0250            .EN
END OF MAE PASS!

 ---  LABEL FILE:  ---

@AGAIN =C2CE            @B1 =C1FD               @CUTBK =C288
@CYCLES =C08B           @DECCYC =C276           @DONE =C2C2
@ENDCHG =C22C           @ENDINT =C261           @ENDMIN =C28E
@ENDSEQ =C10E           @FIND =C20D             @FOUND =C219
@HOLD =C17B             @INTGO =C253            @MAIN1 =C013
@MASK =C05B             @MEMINIT =C02F          @NEXT =C0EC
@NOTTIME =C25C          @NOVALVE =C2D2          @OK =C06F
@ONE =C067              @READ3 =C315            @SAMEVLV =C280
@SC =C1D1               @STOPCLOCK =C334        @TCHRS =C0C7
@TCMIN =C0DB            @TIMING =C195           @TSHRS =C09F
@TSMIN =C0B3            ACCESS =8B86            BINARY =C37C
BLKMOV =8740            CHANGE =C1F8            CONFIG =89A5
COUNTER =0004           CUTBACK =C2D9           CYCCNT =004C
CYCLES =0018            DAY =0008               DISBUF =A640
EVERYMIN =C267          FILL3 =8718             FLAG =0006
GK =88CF                GOMON =C1D4             GOVEC =A659
HOLDING =C155           HRSCNT =004B            ID =A64E
INBYTE =81D9            INCHR =8A1B             INDATA =C07D
INIT =C01D              INIT1 =C02B             INITCTRS =C22F
INPUT =C04B             INTERRUPT =C23A         INTVEC =A678
KSCONF =89A3            L2 =84D3                MAIN =C000
MASK =0007              MINCNT =004A            MONITOR =8003
NACCESS =8B9C           NEWBIT =C2C7            NEWVALVE =C2A6
OUTBYT =82FA            OUTCHR =8A47            OUTNIB =8A44
OUTXAH =82F4            PARM =8220              PARN =A64A
PARNR =A649             PORTDATA =0003          READCLOCK =C2F1
READY =C10F             REBEGIN =C1E1           RESALL =81C4
RESEQ =C1F2             RESTART =8000           RMDIG =A645
SAVE2 =87EA             SAVER =8188             SCAN =C123
SCAND =8906             SCRATCH =0002           SDBYT =A651
SEQ =C0E3               SETUPCLOCK =C357        SPACE =8342
TAPDEL =A630            TCUTHRS =0036           TCUTMNS =0040
TOGGLE =C294            TOTAL =0005             TSRGHRS =0022
TSRGMNS =002C           VALVE =000E             VECSW =8BB7
VERIFY =0000            VIA1 =A000              VIA2 =A800
VIA3 =AC00
//0000,C38C,138C
```

APPENDIX C: VIC-20 PROGRAM LISTING

NOTE: The original code is free of comments and extra spaces due to memory limitations. Comments added for this listing begin with "**". Also, since not all of the character representations used by the VIC-20 were available on this printer, the following substitutions were made:

- C = Clear screen
- H = Home cursor
- D = Cursor down
- U = Cursor up
- L = Cursor left
- V = Reverse video on
- O = Reverse video off

** Initialize system

```
 10 POKE 36879,8:                    ** set screen to black w/ white
    PRINT CHR$(5):
    PD = 255:                        ** all 1's
    PO = 37136:                      ** output port location
    POKE PO+2,PD:                    ** data direction = outputs
    POKE PO,PD:                      ** outputs = 1's
    CC = 0:                          ** cycle count
    IV = 0:                          ** index counter for valves
    FL = 1                           ** flag.  If 1, time in seconds
                                     ** If 100, time in minutes
```

** Input system data

```
100 PRINT "CENTER THE FOLLOWING:  (WITH TIMES AS HH:MM)"
110 PRINT "DCURRENT TIME OF DAY?  (24-HOUR CLOCK)":
    GOSUB 9000                       ** get time subroutine
130 TI$ = RIGHT$(STR$(1000000+HR*10000+MN*100),6)
200 PRINT "VALVE NO.'S IN ORDER? (PUT COMMAS BETWEEN)":
    GOSUB 9300                       ** get valve sequence subroutine
300 PRINT "SURGE TIME PER LEG?":
    GOSUB 9000:                      ** get time
    FOR I=1 TO MX:                   ** MX = total number of valves
    SG%(I) = 100*HR + MN:            ** assign same surge time to each valve
    NEXT
400 PRINT "SURGE CYCLES PER SET?":
    GOSUB 9500:                      ** get surge cycles subroutine
    FOR I=1 TO MX:
    SC%(I) = A + 10*M:
    NEXT
500 PRINT "CUTBACK TIME PER SET?":
    GOSUB 9000:
    FOR I=1 TO MX:
    CB%(I) = 100*HR + MN:
    NEXT
```

** Display top half of display -- system status

```
 600 PRINT "CURRENT TIME":
     PRINT "D* SYSTEM IS     VINGO *":
     PRINT "DSWITCH TIME"
 630 PRINT "DCYCLES COMPLETED":
     PRINT "DCURRENT VALVE NO.D":
     NT = VAL(TI$)+2                   ** NT is time of next switch
```

** SYSTEM HOLDING MENU

```
1000 PRINT "HDD"SPC(12)"VHOLDO":
     GOSUB 6500:                       ** print hold menu choices
     TR = NT - VAL(TI$)                ** time remaining
1010 GET A$:
     IF A$="" THEN NT=TR+VAL(TI$):     ** if no key pressed,
     GOSUB 9800:                       ** time valid subroutine
     GOSUB 5000                        ** update system status
1100 IF A$="R" THEN 2000               ** if run, start timing
1110 IF A$="I" THEN 100                ** if input data, warm boot
1120 IF A$="B" THEN CC=999:            ** so that it will switch next time
     IV = 0:                           ** restart with first valve
     GOSUB 8000:                       ** switch routine
     GOTO 2000                         ** start timing
1130 IF A$="C" THEN GOSUB 7500:        ** change data subroutine
     GOSUB 6500                        ** hold menu
1140 IF A$="D" THEN GOSUB 7000:        ** display valve data subroutine
     GOSUB 6500
1900 GOTO 1010                         ** invalid key, so get another
```

** SYSTEM RUNNING MENU

```
2000 PRINT "HDD"SPC(12)"VRUNNO":
     GOSUB 6100
2010 GET A$:
     IF A$="" THEN GOSUB 9800:         ** time valid
2100 IF A$="S" THEN 1000               ** stop here and hold place
2110 IF A$="F" THEN NT=0               ** forces switch next time through
2120 IF A$="V" THEN CC=999:            ** forces start of surge mode
     NT = 0:                           ** forces switch
     IF IV>0 THEN IV=IV-1              ** to maintain current valve #
2130 IF A$="L" THEN CC=CC-1:           ** restarts this leg - 1
     GOSUB 8000:                       ** switches
     GOSUB 8000
2140 IF A$="<" THEN CC=999:
     NT = 0:
     IV = IV-2                         ** go back one valve
     IF IV<0 THEN IV=0                 ** can't have negative index
2150 IF A$="D" THEN GOSUB 7000:        ** display data
     GOSUB 6100                        ** change back to run menu
2800 IF NT<=VAL(TI$) THEN GOSUB 8000 ** if time to switch, then do it
```

```
   2900 GOTO 2010                           ** go get another key


** Update screen subroutine

** print current time
   5000 PRINT "VH"SPC(13)LEFT$(TI$,2)":"MID$(TI$,3,2)":"RIGHT$(TI$,2)
   5005 B$ = RIGHT$(STR$(1000000+NT),6)
** print next switch time
   5020 PRINT "VDDD"SPC(13)LEFT$(B$,2)":"MID$(B$,3,2)":"RIGHT$(B$,2)
** print current cycle count
   5030 B$ = MID$(STR$(100.01+CC),3,4):
        PRINT "VD"SPC(17)B$
   5043 B$ = RIGHT$(STR$(VN%(IV)),1):
        IF CC=INT(CC) THEN B$=B$+"N"
   5047 IF CC<>INT(CC) THEN B$=B$+"S"
   5048 IF CC>=SC%(IV) THEN B$=LEFT$(B$,1)+"B"
   5050 PRINT "VD"SPC(19)B$"Q":
        RETURN


** clear lower portion of screen for menus
   6000 FOR M=7922 TO 8185:                 ** screen locations
        POKE M,32:                          ** put spaces in locations
        NEXT:
        PRINT "HDDDDDDDDDDD":
        RETURN


** print running menu

   6100 GOSUB 6000:
        PRINT "CHOOSE ONE:D"
   6105 PRINT "S -STOP & HOLD PLACE  F -FORCE SWITCH"
   6120 PRINT "V -RESTART THIS VALVE L -RESTART THIS LEG"
   6140 PRINT "< -GO BACK ONE VALVE  D -DISPLAY VALVE DATA":
        RETURN


** print holding menu

   6500 GOSUB 6000:                         ** clear screen
        PRINT "CHOOSE ONE:D":
        PRINT "R -RESTART FROM HERE  B -START AT BEGINNING"
   6600 PRINT "C -CHANGE VALVE DATA DI -*** INPUT DATA ***":
        RETURN


** display valve data

   7000 FOR I=1 TO MX:                      ** do each valve
```

```
     GOSUB 6000:                      ** clear screen
     PRINT "VALVE DATA:":
     PRINT "DVALVE #"VN%(I)"IS"I"OF"MX
7105 A$ = STR$(10000+SG%(I)):
     PRINT "SURGE TIME    "MID$(A$,3,2)":"RIGHT$(A$,2)
7110 PRINT "UUSURGE CYCLES     "RIGHT$(STR$(100+SC%(I)),2)
7115 A$ = STR$(10000+CB%(I)):
     B$ = MID$(A$,3,2)+":"+RIGHT$(A$,2)
7120 PRINT "UCUTBACK TIME  "B$:
     PRINT "DPRESS SPACE TO GO ON"
7130 GET A$:
     IF A$="" THEN GOSUB 5000:        ** update screen
     GOTO 7130                        ** get another
7140 IF A$=" " THEN NEXT:             ** show next valve data
     RETURN
```

```
** change valve data

7500 GOSUB 6000:                      ** clear screen
     PRINT "VALVE # TO CHANGE?";
7520 GET A$:
     IF A$="" THEN 7520               ** stall until key down
7530 PRINT A$:
     FOR I=1 TO MX:                   ** search sequence
     IF VAL(A$)=VN%(I) THEN 7600      ** try to find that valve #
7540 NEXT:
     PRINT "DDVALVE # "A$" IS ILLEGAL":
     FOR I=1 TO 2000:NEXT:            ** short delay loop
     RETURN

7600 PRINT "DNEW SURGE TIME?";:       ** valid valve #
     GOSUB 9000:                      ** get new surge time
     SG%(I) = 100*HR+MN
7700 PRINT "NEW SURGE CYCLES?";:
     GOSUB 9500:                      ** get new surge cycles
     SC%(I) = 10*M+A
7800 PRINT "UNEW CUTBACK TIME?";:
     GOSUB 9000:                      ** get new cutback time
     CB%(I) = 100*HR+MN:
     RETURN
```

```
** SWITCHING SUBROUTINE

8000 PD = PEEK(PO):                   ** put value of outputs into PD
     CC = CC+0.5:                     ** increment cycles counter
     IF CC=SC%(IV) THEN 8200          ** check if done surging
8030 IF CC>SC%(IV) THEN 8300          ** check if done with cutback
** still surging ...
8100 NT = VAL(TI$)+FL*SG%(IV):        ** set next switch time
     GOSUB 9800:                      ** time valid
     IF (PD OR 64)=PD THEN PD=PD AND 191:   ** toggle surge bit
```

```
        GOTO 8400
   8120 IF (PD AND 191)=PD THEN PD=PD OR 64:     ** toggle surge bit
        GOTO 8400
** goto cutback mode
   8200 IF CB%(IV)<>0 THEN NT=VAL(TI$)+FL*CB%(IV):  ** next switch time
        GOSUB 9800:                     ** time valid
        PD = (PD AND 127) OR 64:        ** turn on cutback
        GOTO 8400
** goto next valve
   8300 CC = 0:
        IV = IV + 1:                    ** next valve in sequence
        IF IV>MX THEN IV=1              ** wrap around to beginning
   8310 IF SG%(IV)=0 THEN 8200          ** if no surge, then cutback
   8320 NT = VAL(TI$)+FL*SG%(IV):       ** next switch time
        GOSUB 9800                      ** time valid
   8330 PD = 255-2**(VN%(IV)-1)         ** sets appropriate valve # bit
   8400 POKE PO,PD:                     ** output new data
        RETURN


** get time subroutine
** returns hours in HR and minutes in MN

   9000 B$ = "":
        HR = 0:
        A = 0
   9010 GET A$:
        IF A$="" THEN 9010              ** wait until key is pressed
   9020 IF ASC(A$)=13 THEN 9100         ** if return key is pressed
   9030 IF ASC(A$)<59 AND ASC(A$)>47 THEN B$=B$+A$:
        PRINT A$;
   9040 IF ASC(A$)=20 AND LEN(B$)>=1 THEN B$=LEFT$(B$,LEN(B$)-1):
        PRINT "L L";                    ** erase char. if del. is pressed
   9100 A = 0:
        FOR M=1 TO LEN(B$):
        A$ = MID$(B$,M,1)
   9110 IF A$=":" THEN HR = A:
        A = 0:
        NEXT
   9120 A = A*10 + VAL(A$):
        NEXT
   9130 MN = A:
        PRINT "D":
        RETURN


** get valve numbers in sequence subroutine
** stores sequence in VN% within routine

   9300 M = 0:
        I = 0:
        A = 0
   9320 GET A$:
```

```
     IF A$="" THEN 9320              ** wait until key pressed
9330 IF ASC(A$)=13 THEN 9400         ** if return is pressed
9340 IF A$="," THEN PRINT A$;:
     I = I + 1:
     VN%(I) = A:
     GOTO 9320                       ** get another key
9350 IF ASC(A$)<48 OR ASC(A$)>54 THEN 9320
9360 PRINT A$;:
     A = VAL(A$):
     GOTO 9320
9400 I = I + 1:
     VN%(I) = A:
     MX = I:
     PRINT "D":
     RETURN
```

** get number of surge cycles
** returns number of 10's in M, 1's in A

```
9500 A = 0
9520 GET A$:
     IF A$="" THEN 9520              ** wait until key pressed
9530 IF ASC(A$)=13 THEN 9600         ** if return pressed
9550 IF ASC(A$)<48 OR ASC(A$)>57 THEN 9520
9560 PRINT A$;:
     M = A:
     A = VAL(A$):
     GOTO 9520                       ** get another key
9600 PRINT "D":
     RETURN
```

** time valid subroutine
** ensures that minutes and seconds are < 60

```
9800 IF VAL(RIGHT$(STR$(NT),2))>=60 THEN NT=NT+40
9820 IF VAL(RIGHT$(STR$(NT),4))>=6000 THEN NT=NT+4000
9830 IF TI$ = "000000" THEN NT=NT-240000:
     TI$ = "000001"
9899 RETURN
```

WIRE TELEMETRY CONTROLS FOR

GATED-PIPE IRRIGATION SYSTEMS

by

WALTER JAY BRADBURY

B. S., Kansas State University, 1983

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Agricultural Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1984

ABSTRACT

The objectives of this research were to automate a gated-pipe irrigation system using wire telemetry controls, to provide flexibility in the controls to allow more efficient irrigation methods, to evaluate the performance of the controls, and to evaluate the economic and labor-reducing potential of wire telemetry controls in gated-pipe irrigation systems.

A 65-ha (160-acre) field in southwestern Kansas was successfully irrigated using wire telemetry controls. Flow control valves that used pipeline water pressure to inflate a diaphragm controlled the flow of water into sections of gated pipe. Three-way solenoid valves directed the flow of water into or out of the diaphragms.

Transmitters were located in a central control station near one corner of the field. These transmitters superimposed high-frequency signals onto 24 VAC carried in transmission lines running along the upper end of the field. Receivers located at each of 18 flow control valves detected the appropriate signal and turned power on to the solenoid valves. The valves and receivers obtained their power from the 24 VAC transmission line.

Initial field tests were performed in August of 1983. Problems experienced with the wire telemetry components were attributed to faulty receivers. After replacement by the manufacturer, all receivers worked well. Other problems were experienced in sticking solenoid valves and unreliable timers.

The final system design completed 100% of the scheduled changes. Labor requirements were decreased to periodic inspections, except during installation and removal at the site. Automated systems have been found to increase efficiencies to values comparable to sprinkler irrigation, at costs comparable to sprinkler systems.

Microcomputers were found to work quite well in the automation of irrigation systems. Those computers that use video displays can easily be programmed to provide information on system status, and in more advanced systems may be used in the scheduling of irrigation as well.