Programmable Controls™ The user magazine

AN ISA PUBLICATION

MARCH-APRIL 1987

Detroit Show Preview

PLC Controls Dancing Fountain
Manufacturing Cell Control
User Applications

FOR USERS AND SPECIFIERS OF PROGRAMMABLE CONTROLLERS AND INDUSTRIAL PCs

Detroit IPC Show Preview

Industry News

Events & Training

Reader I/O

Technology

Products & Services

Control Systems Houses

About the cover: The Grand Haven, Michigan musical fountain has 124 water valve circuits and 40 lighting circuits PLC-controlled in synchronization with music played through 32 speakers (9.2Kw). See related article, "Controlling a Dancing Fountain," on page 69.

Programmable Controls™

Editorial Director
Kenneth E. Ball

Managing Editor
Richard T. Simpson

Associate Editor
Paul S. Baur

Associate Editor
Charles V. Robinson

Editorial Assistant
Harriet H. Perry

Production Assistant
Debbie F. Jackson

Editorial Consultant
Jack Grenard

Programmable Controls, is published bimonthly by ISA Services Inc. Editorial and advertising offices are at 67 Alexander Drive, P.O. Box 12277, Research Triangle Park, NC 27709, (919) 549-8411.

Programmable Controls is regularly indexed in Engineering Index Service and Applied Science & Technology Index, and is microfilmed by University Microfilms, 300 N. Zeeb Road, Ann Arbor, MI 48106. Founded in 1982 by Jack Grenard and Michael Hopper.

ISSN 0747-4458

Postmaster: Send Form 3579 to Programmable Controls, 67 Alexander Drive, P.O. Box 12277, Research Triangle Park, NC 27709. Second class postage paid at Durham and at additional mailing office.

The free exchange of facts and ideas is fundamental to professional development. Articles, news, and other features are selected for publication in *Programmable Controls* in accord with the philosophy of free press. Choices are made solely on the basis of likely value and interest to individuals in the field of instrumentation and control throughout the world. *Programmable Controls* is published to serve the information needs of the profession. Opinions expressed or implied are those of persons or organizations contributing the information and are not to be construed at those of ISA Services, Inc. or the Instrument Society of America.

Designing controls for choreographed fountains is obviously a quite limited market. However, the interfaces and the eventual system are worth noting as the methods are widely applicable.

Controlling a Dancing Fountain

TERRY STEVENS
Controlco Systems, Inc, Muskegon Heights, MI

The Grand Haven Musical Fountain (Fig 1 and Cover) is 160 feet long, pumps up to 4500 GPM of water as high as 125 feet, has 124 water valve circuits, 40 lighting circuits, and 9.2 kw of power driving 32 speakers. The fountain is manually programmed, computer assisted, and PLC operated. The fountain is choreographed to music by changing water formations and illumination patterns.

Primary concerns in updating obsolete controls were ease of programming, minimal maintenance, and minimizing expense. The system developed consists of a personal computer front end interfacing with an audio tape system which in turn interfaces with a PLC

[This article is based on the paper, "A Poor Man's Data Highway," by Terry Stevens presented at the 1986 ESD/SMI Programmable Controllers Conference; Detroit, MI. The paper appears on Page 355 of the proceedings.]

FIGURE 1. A second view (see cover) of the Grand Haven Fountain in action.

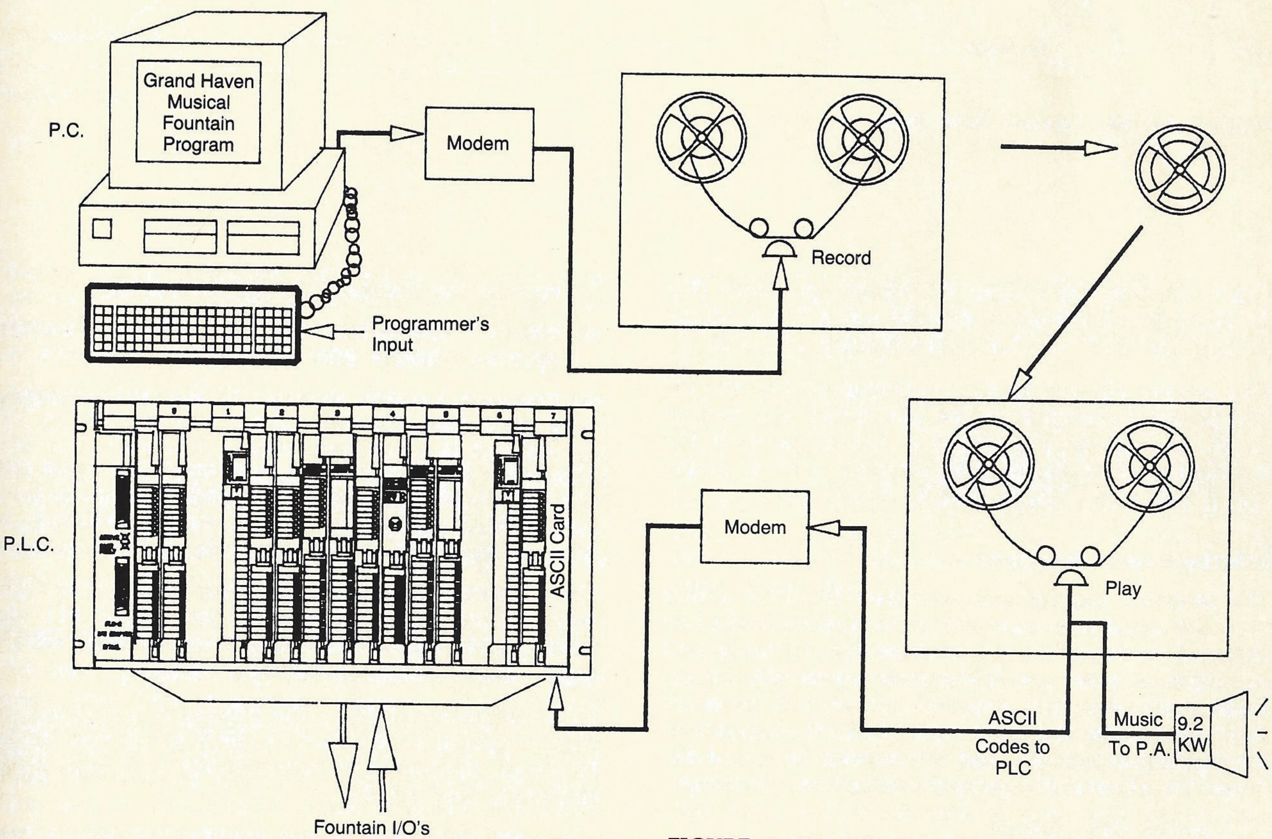

FIGURE 2. Fountain controls are programmed and taped via a personal computer interface. The tape drives a PLC to execute function commands.

0747-4458/87/03/69/05/$2.50© 1987 ISASI

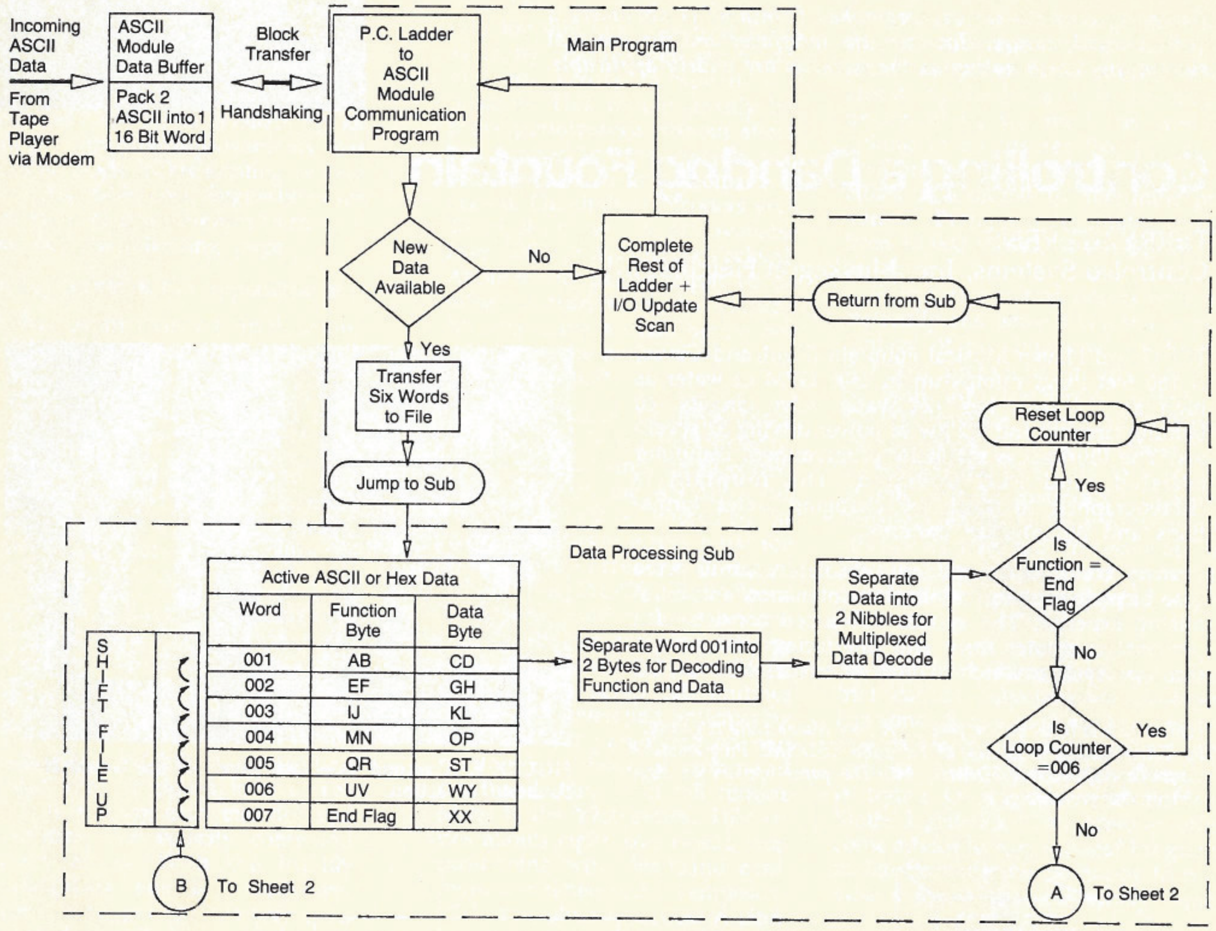

FIGURE 3A, 3B. Program flow chart.

which activates the various functional operations (Fig 2). The personal computer selected was a Tandy Model III and the PLC was an Allen-Bradley PLC-2/30.

This system allowed program development in the PC and program execution by the PLC which was mounted in an adverse location in a bunker beneath the fountain's water basin. The interface was a simple ASCII exchange, RS232 port to RS232 port.

Interface considerations

The interface requirement was data intensive rather than the usual data monitoring need. Data had to be supplied to the PLC in real time for decoding and execution in precise synchronization with the music. Each music program required up to 10K bytes of memory storage and there were some 50 programs in the existing repertoire. The intervening tape unit between the PC and PLC actually served four purposes:

(1) Massive data storage

(2) Assured real time music/fountain coordination by using one channel of the 4-channel music tape to encode the fountain control commands.

(3) Greatly simplified the task of the fountain operator—simply play the tape!

(4) Obviated concerns that the PC might crash during a performance.

The ASCII interface allowed standard components and connections with well established technology. The computer RS232 port connects to a modem which is in turn connected to the record mode of the tape unit.

Other interface methods considered, such as data highway, PC bus extension, parallel port, digital I/O, etc, were quickly abandoned due to overwhelming advantages of the ASCII as outlined below.

System reliability/integrity—the process is non-critical so that no special efforts such as check sums were made. However, thousands of transmissions have been made effectively and reliably.

Speed and throughput—Programs involve only a few changes per second maximum and PLC scan time was minimized by using advanced programming tech-

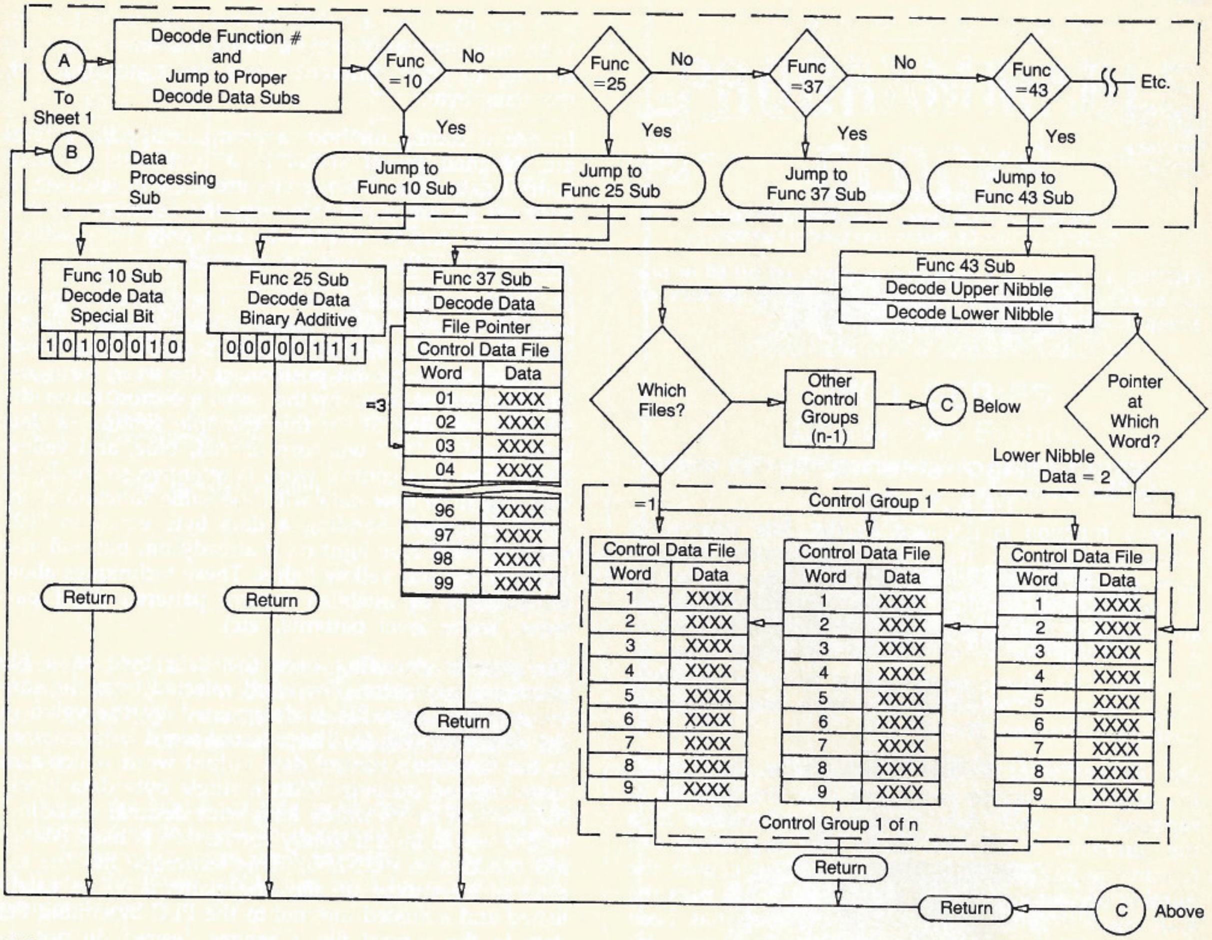

FIGURE 3B.

niques. ASCII codes are executed as six 2-byte commands per PLC ladder scan. With an average scan time of 10 msec, up to 60 commands per second can be executed, well beyond any expected demand.

Software—The ASCII approach entailed no expensive PC or PLC software. Civic volunteers proficiently wrote the necessary PLC ladder diagrams and the fountain language (basic) programs. In most cases, a standard communication software package such as "Crosstalk" could be used for the serial port driver routine, but a custom port driver developed in assembly language met some special needs.

Programming

Advanced programming instructions are used to minimize scan time, maximize memory utilization, and implement data multiplexing techniques. The PLC program consists primarily of jumps, jump to subroutines, file-to-word transfers, and file-to-file transfers. The use of conventional ladder diagram instructions is virtually non-existent except for linking data control bits to real outputs. The main program "talks" to the ASCII module via the communication program (block transfer with handshaking overhead rungs). When new data is available, various subroutines are called to decode the function number and activate the control data routines (Figs 3a, b).

The ASCII module itself provides data buffering (2 k bytes) between the PLC processor and the incoming data. It allows 8 bit ASCII data to be packed into 16 bit words in many different modes. The program implemented uses 2 ASCII characters (2 HEX characters each) per 16 bit word. The ASCII module also detects modem failure or disconnect, and provides error diagnostics.

When new data is available, up to six commands (2 bytes each) are transferred from the ASCII module data buffer to the data processing subroutine into the "active" file. Each command is broken into two words for decoding (DOS loop). The left byte of the command is transferred to the function word and the right byte is transferred to the data word. The function word tells what type of command (which bank of lights, which group of water valves, etc), and the data word identifies specific action (what color, what water level, etc). A list of commands was established

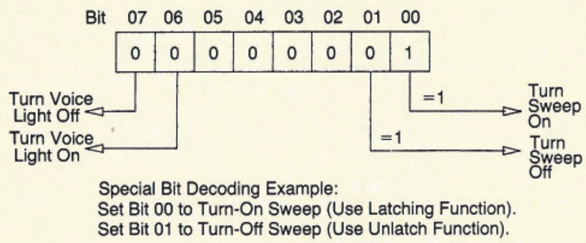

FIGURE 4. Special bit decoding example: set bit 00 to turn on sweep (use latching function). Set bit 01 to turn off sweep (use unlatch function).

to create a fountain programming language originating in the computer.

Once a function is decoded in the data processing subroutine (Fig 3a), the appropriate function subroutines are called to decode the data and activate the control bits. In the PLC processor, subroutines are allowed to call other subroutines or themselves up to 8 levels deep, before a "Return" command must be executed. This allows for extremely powerful programming, versatility, minimum scan time, and maximum memory utilization.

On "Return" from the data decoding subs, the "active" file is shifted up one word and decoding process is repeated. On each data processing subroutine loop the command is checked for an end flag which can terminate the loop. A scan counter also counts the number of passes through the loop. On each pass the scan counter is tested to see if the loop has been executed six times and if so the loop is exited for new data. This procedure is similar to a BASIC program statement "DO FOR I=1 TO 6".

To conserve data transfer and economize memory, data is multiplexed at several layers. By breaking commands into two basic parts, function and data, similar actions can be treated with one command. For example, a function is created to control lights on WATER MODULE 1. The data can determine which color is activated by which bit is set in the data control word. With one command then, a water module is selected and up to eight different colors are controlled with the data byte.

In one decoding method, a group of specific actions are assigned on-off status in a particular functions control data byte. Unique bits are used to latch on or latch off an action. In this way, the functions control data word can be written to and only bit positions with "one" values will be affected (Fig 4).

In another decoding method, a group of common actions (binary additive decoding) are assigned single bit control in a particular function's control data word. A "one" bit in the bit position of the word turns the function on. A zero in the same position turns the function off (Fig 5). In this example sending a data byte equal to "07" will turn on red, blue, and yellow lights. The data control word is retentive so the lights will stay until new data with the same function number is decoded. Sending a data byte equal to "02" will leave the blue light on if already on, but will also turn off red and yellow lights. These techniques allow bit patterns to establish control patterns (light patterns, water level patterns, etc).

File pointer decoding uses the data byte as a file pointer manipulator. The word selected from the control word storage file is determined by the value of the data byte (Fig 6). The selected word is transferred to the function's control data output word which activates desired outputs. With a single byte data block, the file can be 99 words long with decimal decoding, or 256 words long if binary conversion is used (standard function in PLC-2/30). This technique has the additional advantage of allowing patterns to be established and adjusted internal to the PLC by editing the data in the control file. Changes desired do not require re-programming of the computer software. In effect, data "look-up" tables are being implemented in the PLC.

This technique is extended by using the File-to-File transfer instruction. (Fig 7) Control data words are grouped together using consecutive addresses, to create an active control data file. A control data storage file is selected based on the function number and

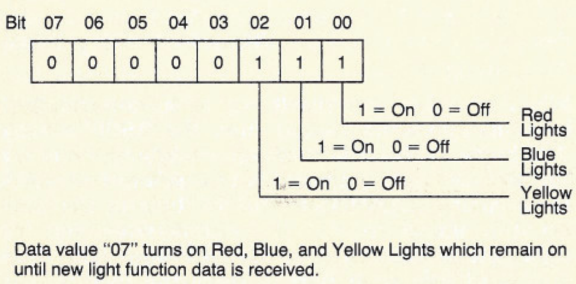

FIGURE 5. Binary additive example: Data 1 in 07 turns on red, blue and yellow lights which remain on until new light function data is received.

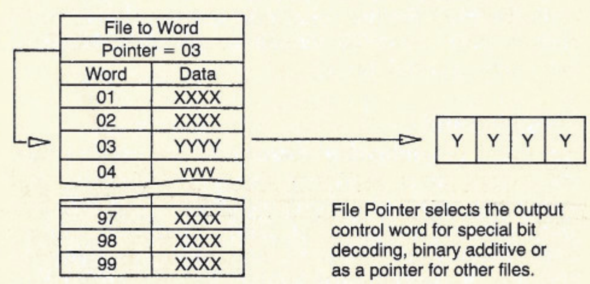

FIGURE 6. File pointer can use output control word for special bit decoding, binary additive or as a pointer for other files.

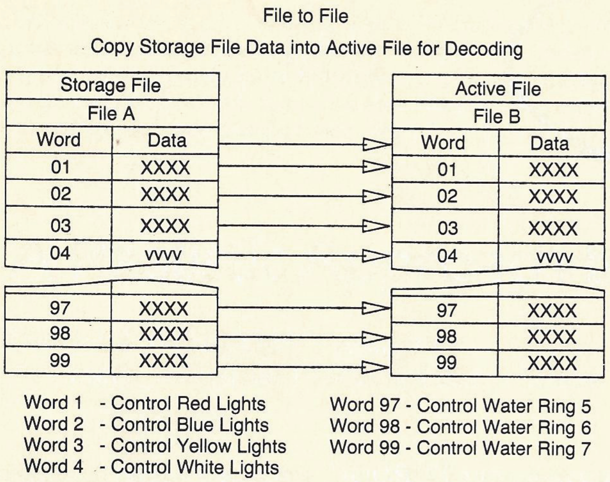

FIGURE 7. File transfer example.

transferred to the active control data file to effect all outputs in the file. With this method, one command can control 1584 outputs (16 bits × 99 words) or by using binary conversion, 4096 outputs (16 bits × 256 words).

Realizing the power of this technique, the data byte is further decoded into an upper and a lower nibble. The upper nibble was used to select groups of files (0-9) and the lower nibble was used to control the file pointers (0-9).

Greater versatility is realized by controlling real outputs through internal control bits. Control data bits are activated by the function/data decoding and, in turn, these control data bits are linked to real outputs through conventional ladder diagram programming. This allows multiple functions to control the same real outputs by placing the control data bits in parallel. Buffering the real outputs is also good practice to allow safeties, interposing interlocks, and opposing conditions to be programmed in series between the control data bit(s) and the real output.

Various combinations of the above techniques were implemented to create desired effects and fountain programming conveniences. The resulting program was extremely short with more data table than ladder diagram rungs.

Debugging the program was straightforward because of the segmentation caused by subroutine usage. The ease of editing the short ladder, the control data words, and the files were greatly appreciated during the commissioning of the project. Documentation was critical due to the number of different codes processed.

Although the Grand Haven Musical Fountain is unique, the programming techniques used to implement it are not. These techniques are fundamentally sound for processing data, whether that data originates from an ASCII interface, some other type of interface, or internal to the PLC from other subprograms or intelligent modules.