TOTAL ENGINEERING SERVICES TEAM

SUPERVISORY CONTROL AND DATA ACQUISITION

(SCADA)

TEST SCADA PROTOCOL (TSP) TUTORIAL

Feb 1, 1993

INTRODUCTION

TEST's SCADA system, like any other SCADA or telemetry system, must have some method of exchanging data and commands among the units connected to the network of devices. These methods are collectively called a *PROTOCOL*, and there are many different ones in existence. Some have evolved from the early days of telemetry, while others have been developed in the Programmable Logic Controller (PLC) industry. TEST has developed its own protocol that is designed to best handle the functions actually required for this type of equipment in typical remote SCADA applications. Admittedly, this protocol is quite a bit different from the traditional protocols used by other developers. This document provides a background on the protocol, called TEST SCADA PROTOCOL (TSP), and offers explanations and examples of how well it works in actual practice.

This presentation is an historical look at TEST's SCADA Systems and the communications protocol it uses. Although it is technical in nature, this *informal* discussion is presented at an introductory level and concentrates on the philosophy and methods rather than the small technical details. For a more complete description of the protocol and other command features, please refer to TEST's SCADA system documentation.

NOTE: The reader is assumed to be somewhat familiar with RTU and PLC systems, although a non-technical user can learn a lot about TEST's systems as well. Please refer any specific questions to TEST's New Orleans office.

TEST SCADA HISTORY

TEST has been installing various types of control and telemetry equipment since 1970. In almost all cases, the units prior to 1988 were made by a major manufacturer and were simply configured by TEST for each particular installation. Often, the system was affected by compromises that were the result of using off-the-shelf components that best fit the application. It was difficult and expensive to get these standard systems to work in an offshore environment. Sophisticated installations required pre-engineered systems using extremely expensive PLC and mini-computer based software that were tailored to meet the exact needs of the end user. These systems formed the backbone of the large SCADA systems used

by the major oil companies. But the smaller companies could not afford to get into the business of developing a system that would meet their unique needs.

The design of TEST's SCADA systems began in 1987 as an effort to provide a reliable, affordable, and flexible system for typical offshore oil production applications. TEST sought to find a way to allow the development time and costs to be shared by a larger number of users. This would decrease the per-unit cost because the end users would install TEST's system rather than one specially developed for each company.

An effort was made to select an existing, standard SCADA system that TEST could adopt as its own system. Although there were many existing systems on the market at the time, this turned out to be a problem rather than a benefit. It was impossible to find a single product or product line that would work in the wide (and I mean WIDE) variety of installations that TEST must handle. The problem was that they were all too different and required unique engineering and programming for each installation. Each also had a different method of communicating with remote devices. Even different products from the same vendor had different communications requirements. Each supplier worked on his own and provided the features that directly affected his product and its applications without much regard for maintaining compatibility with previously installed systems.

The existing systems had all been based on low level electronic units (RTUs) or control oriented Programmable Logic Controllers (PLCs). These systems had "life cycles" of one to three years, after which they were replaced with a newer model. Although these newer models offered cost and performance advantages over their predecessors, they were usually incompatible in both hardware and software. Therefore, considerable manpower was needed to phase out the old system and integrate the new one. It was actually less expensive to install the more costly older equipment because of the engineering time involved to switch to the newer designs. There are many examples of huge SCADA systems that still exist with relic equipment because the cost of upgrading to newer (but incompatible) equipment is far greater than continuing to buy the outdated RTUs.

With over 50 installations, TEST has been successful in meeting the SCADA needs of both small operators and large multi-national corporations. The design philosophy, based on the PC, that was once considered radical has become accepted as the obvious trend in telemetry systems. TEST is fortunate to have taken this path, and is anxious to share its experience and technology with SCADA users from all industries.

TEST SCADA DESIGN

TEST's design strategy was based on the idea that it was time for a radical change in RTU and SCADA system design. The decision was made to start with a "clean slate" and to design a system that would meet the following goals:

1. Cost Effective SCADA in a wide variety of oilfield and industrial installations

2. Low engineering and programming costs per installation

3. Limited hardware obsolescence through the use of generic hardware

4. Future expansion without re-engineering or re-programming

5. Extensive field programming and diagnostic capability

6. Ability to use low cost, dial-up radio and phone communications

The design that emerged from the initial study was an RTU and SCADA system based on Personal Computer (PC) technology. Although this seems obvious now (in 1992), it was fairly radical in 1987 when the design was started. Basing the system on the PC provided a level of field processing power that was unheard of in RTU systems. Because the PC had become a generic item, the availability of components was assured well into the next century. And the continued development of the PC into lower cost and higher performance models meant that a system based on the PC would continue to grow along with the industry.

Once the hardware side of the design was settled, the software options became endless. With the processing power of a PC, almost any type of SCADA protocol could be implemented. A review of past and pending projects indicated that none of the existing protocols provided anywhere near the level of performance that could be derived from a PC based system. The existing protocols had come from the world of discrete electronics, relays, and primitive computer systems. Although they provided some of the needed features, they were totally inept at solving the problems TEST faced on actual field installations. The decision was made to implement a new protocol that would take advantage of the PC to provide a simple, flexible, and reliable SCADA communications system.

This may sound like a harsh attitude in that there were many SCADA systems in operation that worked fine with the older protocols. The main difference in these systems and the ones envisioned by TEST is that the older systems were operated and maintained by skilled technicians. The new TEST system was to be used like the typical PC and would be maintained and operated by normal everyday people. The existing protocols were extremely complex and served the equipment rather than the users. They were designed to allow primitive hardware to work well at the expense of programming time and maintenance nightmares. The new protocol (TSP) would put the PC to work and provide many new features designed to reduce the real cost of the systems: engineering, programming, and maintenance.

TSP started with a small command set that provided the basic data transfer and command functions. But the design was such that additional commands could be added in the future that would not make any existing systems obsolete. This "upward compatible" philosophy has dominated TEST's SCADA systems since the beginning. We constantly add features to the software (and to TSP), but maintain compatibility with any existing system at the same time. So, any system can operate the newer program provided that the installed hardware can support the demands of the application. If it cannot, than a minimal amount of hardware can be changed to provide the additional processing power. The end result is a SCADA system that is much like a PC based Word Processor or Spreadsheet. It just keeps getting better all the time.

TSP BASICS

TEST SCADA PROTOCOL (TSP) is a command based language that offers about one hundred different data transfer and command functions. Many systems only need a handful of the options, while others make use of almost all of the capability. There is only a single version of the program that is updated constantly, and the protocol is constantly extended to include new features. This is considerably different from other RTU and PLC protocols that are very inflexible and cater to only a specific application. TSP is more like a programming language that can be reshaped with its own built-in resources to suit each individual application. Therefore, although all installations are similar they each posses unique functions that are provided by the TSP language.

TSP works with command and response lines that are always simple ASCII text lines (well, almost always). The command processor accepts these lines, executes a function based on the contents of the line, and often generates a response line. This response line may actually serve as a command line to another unit. In this way, various copies of the software communicate with other simply by sending commands to one another. These lines can be easily viewed by an operator or technician so that all activity of the system can be monitored and diagnosed.

This is a very simple concept, one that is difficult for veteran Programmable Logic programmers to understand. Older protocols were based on complex command and response codes, often involving only HEX or Binary coded numbers. While these are convenient for the hardware, they are impossible for a person to understand without a lot of assistance from either a computer or a special piece of programming hardware. It is not uncommon to hear of programmers taking a week to connect two systems located five feet from one another. A large part of this is due to that fact that the programmers cannot easily deal with the computer-to-computer transmissions. The heart of TSP is that it is just simple TEXT messages that make sense to both the computers *and* the people who use them. Although this places a much higher burden on the computer, it makes the user's job much easier. And this reduces the most expensive component in the system: manpower.

So, we agree that TSP uses simple TEXT and that it is easy to read. So what can we do with it? Well, imagine yourself sitting with a blank piece of paper and getting to write down all the things you wished your SCADA system could do. That is what TEST did (and is still doing), and the results are found in TSP. There are commands related to data transfer, calculations, I/O configuration, communications control, system setup, alarm functions, and numerous other functions. Each command line starts with a simple keyword such as SCAN or DATA. The rest of the line provides information to or from a built-in function. In essence, the software reads these text instructions and returns results in simple words and numbers. No computer voodoo, no magic. It is so simple, it is understandable that someone who has spent years dealing with older PLC and RTU systems would be skeptical of TSP's power.

One immediate reaction from new users is "Hey, I can't type! I can never use this thing!" Well, we agree that most field users cannot type and should not be expected to do so on a routine basis. So, all of the TSP functions can be easily automated through the use of simple TEXT files (called command or RTU files) that permanently store the often used functions. These files are activated by numerous mechanisms such as a menu selection, alarm condition, communications

requirement, time of day, and several more technical matters. But the end result that is no matter how complex the operation, the software is simply "reading" some text. The text can come from the keyboard, a DOS file, a command library, a built-in message system, or from another unit. All of these methods have their particular application, but they all work the same. So, if you can do something by hand at the keyboard, you can probably do it to a remote system by having one copy of the software send the messages to another copy over a phone or radio.

That is the essence of TSP. It is simple for the operator to use and understand yet provides unlimited capability through the power of the PC. There is no other protocol that even comes close to providing the level of features that now exist in TSP, and the list of features will continue to grow.

SCADA SOFTWARE BASICS

This document is intended to detail TEST's protocol, but this is difficult if the reader is not familiar with the basic software that operates the system. A brief explanation of the system is necessary in order to appreciate the power of the TSP commands to be discussed later. So, lets take a quick look at the way TEST SCADA system operates internally.

Each system is a PC computer that may or may not have local I/O devices connected to it. Every system has some number of data points called CHANNELS. Unlike other SCADA systems, each channel is not simply a value or off-on state. Channels have many components such as an ID number, tag name, text description, alarm status, high and low setpoints, and many other attributes. Channel types are status input, status output, analogs, counters, timers, values, AGA-3 meters, totalizers, and special functions.

These channels can be thought of individually (such as status channel number 1, or *S1*), or they can be thought of as a group (analogs 1-16 is A1:16). Therefore, an RTU with 16 status and 16 analogs will have over 100 total data values associated with it, each accessed as a separate part of a particular channel. The values are the data itself as well as the setpoints, last alarm time, and other internal values that often take separate storage space on other RTU systems. This often confuses new users who are accustomed to allocating separate "registers" in a PLC to provide some of the features that come automatically with every TEST SCADA channel.

Most of the processes that take place in a SCADA system relate to entering data into channels or transmitting the data to another unit. The most basic maneuver is to SCAN a range of channels. Other SCADA systems use a "SCAN" command to pick up simple data values, often as raw binary numbers. In the TEST system, the SCAN command can request information on any of the individual components of the channel, including the current value in *engineering units*. This may sound like a small feature, but the ability to have an RTU generate completed values (rather than raw binary information) is a huge leap over the older designs. Also, the ability to scan high and low setpoints, alarm status, alarm delays, and other low level information with a single command is an example of the power of TSP.

The data channels are combined into groups that are referred to as an RTU, which is an old term for Remote Terminal Unit. TEST's RTU's are not really RTUs, but complete computer systems operating in a remote location. The term RTU is

used because that is what the industry calls the remote device. Each computer can have a single RTU, or it can have numerous groups of information all located on a single system. This often occurs at a HOST location, where data from several remote systems is gathered into a central system. The software keeps all of the RTUs separate, and each one seems to be a stand-alone system on the Host. Actually, there is no difference in capability in the software at the RTU and the Host. The only difference is that an RTU gets its I/O from field devices, while an Host gets its information over a communications channel. *All other aspects are the same.* Therefore, anything that can be done at a powerful Host location can also be done at a small RTU located in the field.

Each "logical" RTU has certain aspects of its own, such as a unique name and an update time and date. When data is transferred from one unit to another, the time and date at which the information was valid also goes with the data. This makes it possible for one system to pass data to another system on a convenient basis. The system that eventually gets the news will also have a timestamp indicating when the data was originally collected.

Each Logical RTU stands alone in the entire network of the SCADA system. For example, analogs 1-5 for RTU1 are uniquely identified with an RTU name and a channel position (noted as RTU1.A1:5) This unique identifier allows systems with different internal configurations to exchange data. Traditional systems require *exact* alignment of all units in the network because data is exchanged at such a low level. TEST's system allows the data to be *exchanged at a high level* that avoids all of the configuration details of each unit. For example, a Host requesting a pressure value does not have any involvement in how the value was generated. A hardware change in the actual pressure transmitter at the RTU is transparent to other computers because the pressure is a calculated number, not a raw binary input.

TEST's SCADA system is also multi-tasking. This is a much abused term that basically means that the computer can do several different things at the same time. Through the use of the multi-tasker, TEST's SCADA system can provide displays, calculate gas flow rates, communicate with a remote unit, do network communications, and update a data base *all at the same time.* This power comes at the price of complexity, but this complexity is hidden from the user. The end result is a single computer that does *all* of the SCADA chores, including I/O, without the use of external devices such as PLC's or RTU's. TEST's multi-tasking computer *is* the RTU, or the HOST, or a combination of both in a single unit.

TEST's SCADA has an extremely flexible communications system that works with land phones, cellular phones, data radios, voice radios, fiber optics, microwave links, and direct wire. The features needed to make, maintain, and disconnect all of these devices are built into the TSP language. Every aspect of communications control such as cycle timing, phone numbers, timeout delays, retry counts, and other parameters are all easily controlled with the protocol. TEST's capability to handle complex communications is due to the ability to directly control all of these communications systems *at the RTU itself.*

So, the TSP commands to be discussed below all relate to channels, tasks, and communications links. Each command provides a specific function as detailed in the COMMAND REFERENCE document.

SIMPLE EXAMPLES

Although the reader may not be very familiar with TEST's SCADA design, we can illustrate the use of TSP with a simple example. Consider a group of analog input channels that can be referenced with the notation A1:a6. This means the first 6 analog channels on the currently selected RTU (if an RTU other than the current one was desired, it would be specified ahead of the channel range as in RTU12.A1:A6). A command line that would request the engineering values of these channels would be `SCAN A1:a6 E` . SCAN is the keyword, A1:A6 is the channel range, and E requests Engineering Units. When a system reads this line, it would generate a DATA line in the form of:

```
DATA RTU12.A1:A6,E,100.1,200.22,300.333,400.4,500.55,600.66
```

This may look like a complex line, but it is actually very simple and infinitely easier to understand than the encoded response from an RTU. The DATA keyword indicates that data values are to follow, and RTU12 indicates that the information comes from a unit named RTU12. The data will be for analog channels 1 through 6. The data values (these are simple examples) are 100.1, 200.22, and so on. The actual precision of the numbers (decimal places) is set on each system for each individual channel.

The first example showed how data values can be sent from one unit to another. Sending low set points is almost identical because the *E* (for engineering units) is replaced with a *L* (for Low setpoint). The data line would be very similar. Always in text, easy to understand, easy to modify.

So, we see that a SCAN line generates a DATA line. This simple process is used to determine what information is transmitted from one unit to another, and certain circumstances may require different groups of information to be exchanged. Normally, the required lines are typed into a text file that is processed at the time the data needs to be transferred. The default name for this file is DOWNLOAD, so one unit can call another unit and tell it to read its DOWNLOAD file. The SCAN lines will be read at the RTU, and the DATA lines will be processed by the calling unit.

To expedite communications, TSP allows for multiple scans on a single line. For example, the line `SCAN A1:a5 E  S1:s16 R  O1:O8 R@` may be all that is needed to send all of the data for a typical RTU. This line asks for analogs 1 through 5, status 1 through 16, and outputs 1 through 8. The @ sign requests a timestamp to identify the time that the download occurred. That one line does what pages and pages of programming do in a conventional SCADA system.

Other simple TSP commands consist of `DIAL`, `HANGUP`, and `ACK`. These and many others are used in typical command files to perform a variety of SCADA and control related functions. Advanced commands allow for screen displays, graphics, data base, program flow control, and system configuration. The commands can all be entered directly into the RTU, or stored in a file for later processing.

TYPICAL TSP COMMANDS

Every action taken in a TSP procedure occurs as a result of a command line. The source of the command line is not important. What is important is the keyword that starts the line and the various pieces of information contained on the rest of the line. The format of the command lines is not rigid, but allows for variations that suit the instance in which the line is used. For example, either spaces or commas can be used to separate items on the line. This is because it is easier to enter `SET HORN ON` rather than `SET,HORN,ON`. Either will work, but the selection of commas or spaces is left to the user. The system also accepts abbreviations in many cases where the actual spelling is not important. For example, an output can be turned off with the command `CALC OUTPUT1 = OF` (not that OFF was abbreviated to OF). This is because the system is smart enough to know that the only choices are OFF or ON, and the first two letters is all that is needed to tell the difference. These are tiny examples of the type of intelligence built into TSP that make its use so fast and convenient in actual practice.

Lets look at a few more examples to demonstrate the power of TSP.

```
CALC TOTAL1 = TOTAL2 + TOTAL 3
```

The CALC command allows access to a built-in expression solver (calculator) that provides the normal math functions as well as special RTU related functions. This example lets two channels (total2 and total3) be summed and the result stored in a channel called TOTAL1. The channel types are not important, as TSP takes care of all of the internal lookups and data conversions.

```
SET HORN SIREN1
```

This SET command sets the logical horn to a channel called SIREN1. This channel is probably an output channel that will turn on a siren whenever the program wants to blow the horn. Anytime an alarm occurs, SIREN1 will be turned on. When the horn is silenced, SIREN1 will be turned off.

```
PHONE 555-1212
```

The phone command is used to set the default callout number for the currently selected RTU. Normally this is set once and forgotten, but sometimes we need to automatically change the number to account for different shifts of operators. The numbers can be pre-programmed and changed with any event such as time of day or an operator keypress. Or, this command can be sent from one unit to another to remotely reprogram the phone numbers.

```
CURSOR 5,8,"The Pressure is ",$(PRESS1)
```

The cursor command is used in script files to paint images in real time and have current values positioned on the screen. This example places the cursor at screen column 5, row 8, and presents the message "The Pressure is" followed by the current value of channel PRESS1.

```
POLL WC345 REPORT
```

A POLL is a SCADA term for sending data, usually from a remote to a host location. This example tells the system to call unit WC345 and get a download, and then print a report when done. The system takes care of all the details of

calling, transferring, retrying if necessary, and finally printing the report. This command is typically used in the daily AGENDA that executes commands at a specific time of day.

```
BLOCK CALC TIMER1 = 30
```

The BLOCK command causes the remainder of the text line to be sent to another unit for processing. This is how one unit controls another one with a remote connection. The rest of the line is processed at the remote as if it was entered locally at that unit. In this example, a channel called TIMER1 will be set to a value of 30 at the remote unit. This reference by name (rather than by point ID) allows for host systems to access remotes without specific knowledge of their configuration. Only the channel names are important, not their position in a list of I/O points.

TSP COMMAND FILES

The power of these simple commands really comes though when they are combined in an TSP command file. These files are used to perform any routine function that can be anticipated for the system. Commands are simply entered in the order they will execute, and comments can be entered following a semicolon anywhere in the file. For example, a host system may call an RTU to initiate a facility shutdown with the following simple file:

```
;  Typical Remote Shutdown file (very simple example!)

        Select RTU10                    ; Select RTU to be polled
        DIAL                            ; connect by radio, phone, or whatever
        WAIT 32 Connect                 ; wait up to 32 seconds for a connection
        If @online(1)                   ; If the RTU connected
          msg RTU is online.  Sending Timer Command.
        else                            ;  alternate action if RTU did not connect
          msg RTU did not answer in time.  Retry later.
          return                        ; exit this process
        endif
        block CALC ESD_TIME = 60        ; send Pulse command to the remote unit
        msg Waiting for Platform to Shutdown
        sleep 30                                ; wait for the ESD to complete
        block read download                     ; Request download from the remote
```

This simple TSP file will select the proper RTU, call it, provide a status message indicating the state of the connection, transfer the CALC command that loads the remote timer, wait for the facility to shutdown, and then request a data transfer (download) from the remote. Note that the unit executing the file only needs to know one thing about the remote: the name of the timer channel that need to be pulsed. In this case, the channel is called ESD_TIME, but this is of course an arbitrary name. The remote system will take care of the actual ESD details, and will also provide the command sequence necessary for the download.

COMMUNICATIONS SECURITY

Oilfield and industrial systems often have to work in harsh environments, and the available communications are typically below the standards required for standard SCADA systems. With this in mind, TSP was designed with many features to provide secure communications in marginal systems. The philosophy is that the system will have the intelligence to transparently handle the comm link and to make adjustments to accommodate poor quality links. Features such as timeouts, command sequencing, error checking, and data compression are all built into the protocol.

If it is possible to get a message through, TSP will get it delivered. More importantly, if the message *cannot* be delivered TSP will know it and can be programmed to take an alternate action. TSP will not allow a bad message to get through, and always acknowledges every command action. In short, TSP is an industrial strength protocol that knows how to deal with real-life communications problems.

AUTOMATION

The command line system used in TSP is fine for manual operation because the user gets instant feedback when a command is entered. Normally, common actions are stored in Command Files that perform a specific function such as Daily Logging or Emergency Shutdown. No separate compiler or development system is needed to develop these files. The SCADA software is a self-contained unit complete with a text editor for modifying command files. Processes can be automated by entering standard TSP commands into a text file (with the built-in editor) and then causing the file to be processed at some specific event. Therefore, extremely complex operations can be prepared and tested and later activated with a single keypress or alarm action. The events that can cause a command file to be processed are:

1. Local Keyboard or Remote Communications command.
2. Screen Menu selection.
3. Alarm or Reset Condition on any channel.
4. Timer channel timeout (periodic processing).
5. Daily Agenda list (time of day).
6. Communications connect, disconnect, or failure.
7. Data Timestamp (data update process).
8. Internal Errors (logger, printer, etc.)

The automation capability allows TSP to be used for simple control applications because a preprogrammed series of commands can execute on any alarm condition. Therefore, it is possible to have the RTU perform local actions as a result of a local stimulus by setting up commands in a specific file. These actions can also be controlled remotely because of the numerous methods of activating

command files.

DATA STORAGE

A common question regarding data storage is "how long can your system store data?" This question is a relic from the dark ages when RTU's and SCADA systems were primitive electronic devices. TEST's system is a *PC*, and a PC can store data for a long time. The storage in a TEST system is basically limited to the available disk storage, and this is almost unlimited in typical PCs in use today. Small RTUs have 512K solid state disks, while larger Host systems may have 200Mb + hard disks.

TSP stores most data in ASCII text files that can be easily read by other PC based programs. The internal database is a special real-time format that must be read by specially prepared programs. However, a TSP command file can be easily prepared to pull information from the data base and place it into a delimited ASCII DOS file for import into any other program. Again, the theme is simplicity, and data in ASCII DOS files is about as simple as it gets.

TSP ADVANTAGES OVER PLC PROTOCOLS

Programmable Logic Controllers (PLCs) are designed for local high speed control in industrial situations. *Period.* PLCs are not designed for remote operations, although many of them provide some facility to send and receive data scans over a dedicated communications link. They also have no built-in ability to do typical SCADA actions such as dial a phone, key a radio, or perform an AGA-3 gas flow calculation. The PLC protocols such as MODBUS, DATA-HIWAY, or TIWAY are closed proprietary systems designed for high speed, local, binary data transfer. Data transfer covers about 5% of what a SCADA system must do, and the PLC systems have no provision to perform the remaining 95% of the job. They operate at a very low level that is different for each type of PLC, even those in the same product line. The interface problems associated with these systems are infamous and provide considerable job security for people all over the world.

TSP is a command file driven system, and editing and modification of the systems *in the field* is a standard and valuable feature. On the other hand, communications changes on a PLC is either impossible or done only with special equipment. Any incompatibility problems between units, even those from the same manufacturer, must be solved *later* by the factory at their convenience (if they will be solved at all).

Aside from the hardware interface problems, PLC protocols have no built-in ability to handle most of what TSP does without any additional programming. For example, TSP knows all about radio and phone communications, including those with extensive delays such as packet radio or satellite. TSP allows channel values, setpoints, alarm delays, alarm modes, and all the other channel features to be transferred with almost no programming effort. TSP has a built-in channel type for *true and complete* AGA-3 gas flow calculation that requires absolutely no hand calculation. In a PLC system these feature *do not even exit*, and must be specifically programmed for each and every application. Once they are programmed, a data transfer mechanism must be designed to send data from one unit to another. And functions such as dialing a cellular phone, using an alternate link, daily production totals, and other standard TSP functions are difficult or

impossible on even the most powerful PLC system.

How then are PLC systems advertised as suitable for your RTU application? Often the solution is to put a Personal Computer (PC) or mini-computer in the loop somewhere to handle the communications, gas calculations, data display, and other SCADA related chores. All "PLC based SCADA systems" employ a computer of some sort to handle all of the things that the PLC cannot do by itself. Some PLC manufacturers make a PC module (for about $3000) that must be installed in order to do any of the things beyond normal ladder logic control. TEST's systems eliminates the PLC by letting the *PC do all the work*. The result is a far simpler, less expensive SCADA system that can be maintained by anyone who can operate a personal computer.

ADVANTAGES OVER OTHER RTU PROTOCOLS

Fortunately, most RTU protocols come closer than PLCs in providing true SCADA features because the RTU evolved within the SCADA industry. These systems need to alarm, transfer data, and handle communications systems, and their protocols are oriented to these tasks. PLC's, on the other hand, are designed for local digital control. Most RTU protocols do not assume much intelligence in the field because traditional RTUs were fairly low powered devices that could not do anything on their own. But TEST's system (TSP) is much more advanced than low level RTU systems such as OPTOMUX, HYDRIL, or SYSTRONICS type protocols. TSP works at the TEXT level rather than the machine language level of the older systems. The power of the PC is there to provide a convenient protocol that makes sense to both the computer *and the people who use them.*

TEST's TSP protocol can handle the various communications methods that are encountered in real-life situations. Problems associated with dial-up links (such as the phone system) are handled automatically. Multi-drop situations where numerous units share a common radio channel are also handled with ease. Problems with cellular phones, shared radio systems, and intermittent communications are all handled by TSP. There is no chance of an improper transfer (as in OPTOMUX) because each and every transmission is numbered, routed, identified, and acknowledged. High level data can be transferred (not available in MODBUS) to allow exchange of engineering unit and text data among the RTUs. And TSP's concept of time stamping data is not available in any other protocol.

TSP is also considerably more advanced than other RTU protocols because it is actually a programming language in itself. The basic functions of scan and control have been greatly enhanced to provide over 100 different commands and functions within the language itself. The ability to create command files that are assigned names allows the language to be extended for unique requirements at each location.

PROTOCOL COMPATIBILITY ISSUES

A common question about TSP is "Will TSP communicate with our existing SCADA system?" The answer is "yes" because TEST's SCADA can optionally use the pseudo-standard Modbus PLC protocol. This communications method is common to "Open Architecture" applications where systems from several vendors must connect and share data. The use of Modbus allows TEST's system to appear as a Modbus RTU on any compatible data hiway.

It may be possible to modify TEST's software to communicate with other protocols, and we always offer to look at the problem. But so far, every study has resulted in the decision to pitch the old protocol and move ahead to TSP. This is because the effort required to implement the old protocol will only result in another installation of the old protocol. Most of the advantages of a PC based system cannot be used on the old system because the old system was not as sophisticated as the new program. Users wanting to move on to a more modern system such as TEST's SCADA system realize that their system that started life in 1966 is no longer acceptable in the 1990's. And the cost of installing a number of new TEST RTUs is often less than one years maintenance on an existing "relic" system.

But, TEST is always willing to look at special cases to determine if indeed an older protocol can be implemented. If possible, TEST will provide the modifications to the SCADA software to allow connection to the older program. This is likely to be the best solution for systems with a large number of installed RTUs that must remain in place for some time to come. It is often impossible to modify these older systems because the development tools necessary for the job have long since disappeared from the scene.

For more SCADA Information please contact:

Mitch Wiese - Houston (713) 688-9621
Art Zatarain - Gretna (504) 368-6792
Bob Lee - Lafayette (318) 235-9013
Tom Garic - Ventura (805) 658-0403

AMZ/nt proto1.doc