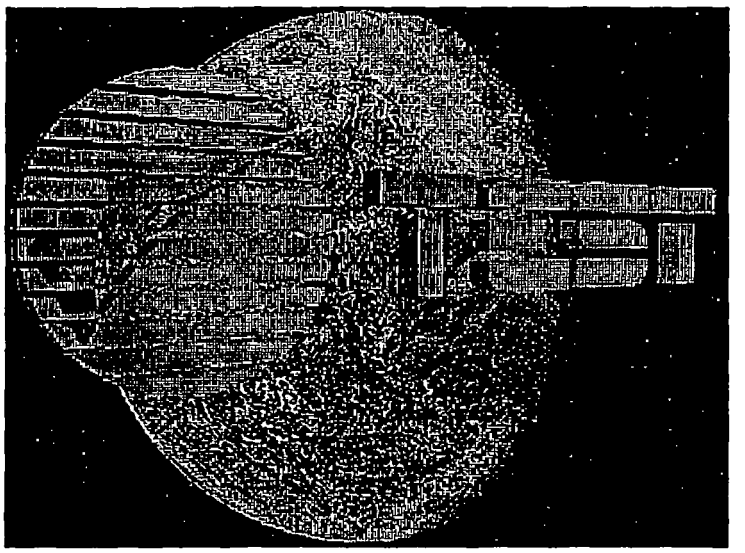


SCADAWARE™

T S P
TEST SCADA PROTOCOL
COMMAND
REFERENCE

Document 1020-03

Revised: December 1994

Cover art: Nick Niklaus

CONTENTS

1 INTRODUCTION TO TSP

TEST's SCADAWARE is a Personal Computer based, real-time data acquisition and monitoring system specifically designed for oil field, offshore, and industrial operation. SCADAWARE has advanced communication abilities which enable it to operate in a wide variety of installations. It also supports a number of data channel types, including PID and AGA3 gas meter calculations. Like other SCADA systems, TEST's units must have some means of configuring and programming the various aspects of the system. This manual describes the powerful command system called TEST SCADA PROTOCOL (TSP) which drives all aspects of system configuration and operation.

TEST's system is unique in that its programming language, TSP, is also its SCADA Protocol. TSP is a collection of commands, methods, and procedures which control basic operation of the system. TSP also permits the preparation of simple programs that extend its capabilities to meet the needs of specific situations. The availability of TSP is what makes TEST's SCADA System so powerful yet easy to configure and use.

TSP is similar to an interpreted programming language, such as BASIC, where the computer "reads" text lines, decodes them on-the-fly, and then executes the meaning of the line. Single lines can be entered and executed right from the keyboard. More often, the programs are written line after line in a simple Text file format and stored for later execution. The software, through TSP, processes these files to perform a complex sequence of instructions. If you are familiar with BASIC programming, TSP will be very easy to understand.

SCADAWARE is a generic program that comes with most of the capability needed to install and startup a working SCADA system. However, most locations will need a few special procedures to handle output control, daily reporting, custom displays, special calculations, and other tasks that are associated with operation of a facility.

This reference is not a low level instruction manual on computer programming. Although the user is not expected to be a professional programmer, some familiarity with personal computers and telemetry concepts is assumed. This documentation describes the TSP commands that can be used to setup and operate the software program. Information on the specific hardware used in each system is contained in documents furnished with each system. This manual makes no assumptions as to hardware capability other than assuming that an IBM-PC/AT (or compatible) type system will be used.

A companion document, SCADAWARE SYSTEM DESIGN CONCEPTS, should be referenced for information on basic system concepts, features, and hardware configuration details. Other documents related to GRAPHICS and DATA BASE operations should be used for information on these advanced topics.

Note: This version of the TSP COMMAND REFERENCE contains information on the software as of Dec 1, 1994. Versions prior to this date may not have all of the features described in this manual. Contact TEST for information on obtaining a SCADAWARE update.

2 COMMAND LINE INTERFACE

2.1 INTRODUCTION TO TSP

TEST SCADA PROTOCOL (TSP) processes all commands as simple ASCII text lines. Each line begins with a special unique keyword, and is optionally followed by various sub-keywords, parameters, and option codes. TSP is built upon a command line interface that is usable by both people and computers. The form of the commands is identical in either mode. Human communications uses the plain text form of the command. RTU communications uses an advanced format which includes message routing and error checking information. However, this additional overhead is handled automatically by SCADAWARE and need not be of concern for the average user.

All command components consist of simple letters and numbers. No special computer codes or numeric formats are used in any visible part of a TSP command message. Many commands and options can be abbreviated for convenience, and all entries are case-insensitive. The computer processes the text lines and converts them into the internal language of the computer. The use of simple Text messages places a higher processing burden on the computer while making the overall system less complex for the user, technician, or troubleshooter.

2.2 SCADAWARE COMMAND PROMPT

When in human interface mode, the program displays a COMMAND PROMPT that indicates it is ready to process another command line. The prompt contains the name of the computer system followed by the name of the currently selected RTU, and then the task number processing the entries. For example, a prompt for system named MASTER, while selecting RTU1, for task 0 would look like

```
MASTER-RTU1 0>
```

The 0 indicates the local user task number. Serial port tasks, which typically handle remote communications, are numbered 1 and higher.

NOTE: If the program in running in its demonstration mode, the prompt will be preceded by the keyword DEMO! When in demonstration mode the serial communications , AGA3 calculations, and I/O drivers are disabled. Demo mode operation will only run for about an hour before rebooting.

Normally, the CALC keyword has to be specifically used on command lines which will access the expression evaluator. It is possible to put each task in CALCULATOR MODE, meaning that it can assign values directly to channels. When in this mode, the prompt will be preceded by the keyword [CALC] to alert the operator that keyboard entries will automatically affect data in the system.

Commands are entered as typed text and ended with a carriage return (ENTER key). This mode is used for operator based command response as well as command file processing. The backspace key can be used to back over and erase typing mistakes, and the standard DOS cancel (control-C) key can be used to terminate an entire line.

Previously entered lines can be accessed with the up and down arrow keys, or the ctrl-E and ctrl-X keys. This allows recall and editing of recently used lines without the need to completely retype them.

Local tasks are always ready to accept command lines and are said to be in TERMINAL Mode. Serial port tasks, which normally handle remote communications, are normally set up for computer based communications and are said to be in RTU MODE. The only way to manually get out of RTU mode is to enter the line [CRTL-A]*TERMINAL[ENTER] (all upper case). Note that the [CRTL-A] and [ENTER] are a single keystrokes, so a total of 11 keystrokes are necessary to escape from RTU mode. This special *TERMINAL key sequence will place the task in TERMINAL mode, allowing direct manual operation. The task will automatically fall back to RTU mode upon disconnect.

The local keyboard and terminal are always assigned to RTU task 0, which is in terminal mode at all times. So, the special escape sequence is only needed when calling into the computer from a remote location over a serial port.

2.3 COMMAND LINE FORMATS

TSP command lines begin with a unique keyword which identifies the operation to be performed. Most commands have additional pieces of information, called COMMAND LINE PARAMETERS, which follow the keyword. These parameters provide additional information for the command, and they come in a variety of forms including numeric, text, and variable forms.

The physical format of TSP command lines is slightly different depending on whether the program is in TERMINAL mode (human operations), or RTU mode (computer to computer operations). RTU mode is never used manually because the error checking procedures are awkward for a person to use. The computers, however, handle the overhead easily and do all the processing necessary for RTU-to-RTU transfers.

The command format for RTU mode is more complicated because it provides error checking and other communications overhead. All text lines in RTU mode must begin with a special START OF HEADER code, SOH, which is CONTROL-A. In RTU mode, nothing is processed until the SOH is seen. This is why the CONTROL-A must be entered when switching from RTU mode to TERMINAL mode as described above.

TSP has a very sophisticated communications protocol that is designed to be efficient, powerful, and reliable in real-life communications situations. Refer to the SCADAWARE SYSTEM DESIGN CONCEPTS manual for a detailed description of the TSP command protocol in both human and RTU mode.

2.4 COMMAND LINE PARSING

The RTU uses a consistent method of command line "parsing", which is the way it separates the various pieces of information found on a line. Either a comma or a space can be used to separate entries on any line, but both cannot be used on the same line (with one exception). If a comma is detected on the line, then the comma is used as the primary separator. If no comma is detected, then a space is used. The reason for having two ways of separating entries is to allow user flexibility and also to allow spaces in certain entries (like channel descriptions).

The only exception is that a space can be used as the first delimiter even if commas are used thereafter on the line. This allows for a more conventional entry format where the keyword is followed by a space, and the subsequent parameters are separated by spaces. The rule here is that if a comma is detected and a space is detected, the space must precede the comma. Otherwise, the comma is the only separator allowed on the line.

```
SET  HORN S32      ; Space delimited line
Set, Horn, S32     ; Comma delimited line
Set  Horn, S32      ; Alternate use of space then commas
```

Pairs of double quotes "---" can be used to combine a group of words into a single parameter. The parser will look for the presence of these quotes and will take everything in between a pair as a single element. For example, the following command lines are equivalent:

```
Name Global Oil Corp HI 272 JA    ; INCORRECT!
Name, Global Oil Corp HI 272 JA   ; Correct Comma Delimited
Name "Global Oil Corp HI 272 JA"   ; Correct with quotes
```

The program would parse the first line as 7 separate entries, starting with the keyword NAME and ending with JA. The remaining lines would be correctly parsed as two elements.

2.5 COMMAND PROCESSING

All command lines begin with a TSP keyword indicating the nature of the command. The keywords are descriptive of the command purpose and are designed to be easily learned by the user. Examples are words like SET, CALC, INPUT, and DISPLAY. The command processor is case insensitive, meaning that any combination of upper and lower case characters can be entered. The keywords CONFIG and ConFig are processed exactly the same. Keywords must be entered *exactly* as they are shown in the command reference sections. The only time abbreviations are allowed is when a specific alternate form of the command is available. This is done for the most frequently used words like DISPLAY (disp), CONFIGURE (config), LIBRARY (lib), and PROGRAM (prog). In effect, the shortened version of the command is a duplicate of the full length word.

SCADAWARE tries to match the TSP keyword found at the start of the line. If not found, the system will normally return an error message. It is possible to have the system search for alternative commands which are stored in TSP procedure files, called command files, which effectively allow the user to extend the TSP language. To use this feature, the SET RTU ON command must be used which tells the system to look for an RTU file with the same name as the unmatched keyword, and if found, execute it immediately.

2.6 COMMAND LINE PARAMETERS

Most lines require *command line parameters*, which are the pieces of information that follow the initial keyword. The number and form of the parameters is dependent on the nature of the command. Some will assume parameters if none are given, while others must have parameters in order to be processed. TSP is very forgiving for the spelling of the parameters because they can often be determined properly with only one or two letters. For example, if a keyword has only two possible parameters like OFF or ON, TSP will simply check two letters of the parameter. In that case, selection of *off* can be done with either *OF* or *OFF*. Of course, there are some instances where an abbreviation or misspelling cannot be allowed because the result may be ambiguous. If it makes sense to you that an abbreviation would be acceptable, then it is probably okay with TSP as well. The goal is to make the system as easy to use as possible by eliminating as many typing mistakes as possible.

Command line parameters can be fixed at the time the line is prepared, or they can be created at runtime using special codes which are replaced at the time the command is executed. These replacement codes allow for run-time values to appear on the line just as if they had been entered on the original line. Refer to the SCADAWARE System Design Concepts manual for more information.

2.7 MULTIPLE COMMANDS PER LINE

The program allows for multiple TSP commands to appear on the same physical line. There are limitations in the use of this because of the interactive nature of TSP. But for many applications, such as simple procedures, the multiple statement capability will clean up text and make for more compact libraries.

The basic idea is to allow several TSP commands which do not require an ACK or user input to be executed in one pass of the command processor. For example, consider the following line that contains four separate commands:

```
horn on : sleep 2 : horn off : msg This is a test
```

Note that the default command separator is a colon and a space, not just a colon. Anytime the parser sees a colon followed by a space it will chop off the beginning of the line and process it separately from the rest of the line. It then continues to parse the line from the point where the chop occurred. The default separator character is a colon, but this can be changed with the SET MULTI command. Other characters that can be used as the separator character are !, |, \, and ~. For example, the command

```
SET MULTI !
```

would set the separator character to be an exclamation point rather than a colon.

The multiple command capability can be turned on or off with the SET MULTI command. The format of the command is

```
SET MULTI ON
or
SET MULTI OFF
```

The default is OFF, so to be able to use this feature the command SET MULTI ON would have to be processed. This is a global setting which affects all tasks.

There are times when the multi-statement parsing is not desired. Sometimes we want the parser to treat the entire line as a single chunk of text. This is most likely to occur with the FORCE and BLOCK commands where text is sent to another task or to another unit. The parser can be instructed to leave the text line alone by starting the command with the slash character. Consider the following example:

```
FORCE 1 READ PROC1 : READ PROC2 : READ PROC3 : BYE
```

At first glance you would think this statement would cause the line

```
READ PROC1 : READ PROC2 : READ PROC3 : BYE
```

to be sent to the input message queue for task 1 because of the FORCE 1 at the beginning of the command. However, this is NOT what would occur in this case because the parser will see the line as follows:

```
FORCE 1 READ PROC1          ; Statement 1 executed by current task
READ PROC1                  ; Statement forwarded to task 1
READ PROC2                  ; Statement 2 executed by current task
READ PROC3                  ; Statement 3 executed by current task
BYE                         ; Statement 4 executed by current task
```

This sequence of events is probably not what was intended by the programmer. The original intent was to send the entire sequence of commands to task 1 for execution. In order for the entire line to be treated as a single statement the following command would have to be used:

```
/FORCE 1 READ PROC1 : READ PROC2 : READ PROC3 : BYE
```

With this syntax, the entire line would be treated as a single statement and everything from READ PROC1 through BYE would be sent to task 1 for processing.

2.8 AUTO CMD EXECUTION FOR ALL RTUS

A shorthand method is available to automatically repeat a TSP command for all logical RTUs on a single system. Preceding any command line with the greater-than symbol (>) will cause the command line processor to execute the command once for each RTU in the list for that system. For example, the ACK command which is used to acknowledge all alarms for an RTU would have to be used several times, once for each logical RTU, in order to clear all alarms on a Host computer. Rather than do the sequence:

```
Sele RTU1
ACK
Sele RTU2
ACK
|
|
Sele RTUn
ACK
```

The user could instead enter

>ACK

2.9 MENUS

Most of the typing in everyday use can be eliminated by the use of the powerful built-in pop-up menu system. The menu systems allow many commands to be processed by making a simple key or mouse selection from a menu rather than entering the commands by hand. Menus can be programmed with the built-in editor of the program, so new ideas and changes can be performed and tested right in the field. Once a system is setup, the everyday operators will only select menu entries and enter a few numeric values (like meter plate sizes) as part of their daily routine. The typing they would normally perform is saved in the menu system for activation by the simple menu selections.

2.10 TASK REFERENCES

Some commands require a task number to be specified as one of the parameters. There are several ways that a task number can be specified in a command. The physical task number, such as 0, 1, or 2, can be used as a direct reference. This is fine for very simple systems that have no dynamic reassignments of tasks and RTUs. A better method is to refer to tasks by the ID name provided for them in the DAT file which is read when the system starts up. Using the ID name removes dependence on the exact layout of tasks within a system, and eliminates reprogramming when changes are made in the system configuration.

Tasks can also be indirectly referenced through a valid RTU name. If the text provided for the Task name cannot be matched, an additional search is made of the RTU list. A match there will cause the task number to be determined by the task responsible for servicing the default link for that RTU. This provides a very powerful and flexible way to determine the task which must be accessed in order to affect an RTU. In effect, sending a message to an RTU is the same as sending it to the task that manages that RTU by default. This is a good idea for systems that dynamically reassign default links to an RTU, possibly for alternate radio and phone communications paths.

2.11 COMMAND RESPONSES

Most commands have some form of response, even if it is just a simple "OK" to indicate that the command was processed properly. Errors are detected whenever possible and some form of error message is displayed. In TERMINAL mode, the messages are fairly descriptive. In RTU mode, which is used in computer to computer connections, the message is less informative but does indicate a problem was encountered. The response in RTU mode is a completion code, or CCC line, followed by a number. Numbers 0 and 1 indicate completed executions. Other numbers indicate some sort of processing error.

For example, the command TIME 08:12:00 entered in TERMINAL mode will cause the time to be set and the program will respond with "Time is 08:12:00". In RTU mode, however, the time would be set but no response would be given. The communications error checking would insure that the command had been received properly and there is no need for the program to furnish a reply.

2.12 COMMAND FILES AND PROCEDURES

It is common to place frequently used command sequences into text files which can be processed when required. These are called COMMAND FILES, and they consist of standard DOS ASCII text files prepared with any text editor, including the one built into SCADAWARE. The assumed file type is .RTU, although any valid DOS file type can be used.

Each file contains a separate function, such as a custom display screen, a download procedure, or an output control routine. Most command sequences are fairly small, between 3 and 10 lines, and there they tend to accumulate quickly to clutter the directory with a large number of small disk files. An alternative to separate files is to put many separate TSP procedures into a single file, called a library, which is loaded into the computer's memory in a single pass. This memory resident library provides a

number of benefits:

1. Much faster TSP procedure execution.
2. More efficient use of limited disk space, especially on RAM disk drives.
3. Easier procedure management by reducing number of disk files.
4. Standard Procedure Development through Utility and other Libraries.

Refer to the LIB command for more information on library loading and processing.

2.13 COMMAND LINE COMMENTS

The semicolon (;) character is used throughout the program as a program comment character. The parser will look for this character and ignore anything that follows it *on each line*. This allows for non-effective comments to be placed in command files. This is a good programming practice that assists in future modification of the various program files because descriptive comments help explain the purpose of program lines.

Comments can be placed in data and RTU files. Anywhere that the program will parse a line, the comment character is valid. Many of the examples provided in this documentation use comments just as they can be used in actual files.

3 TSP COMMAND SUMMARY

This section provides a *quick reference* to all SCADAWARE TSP commands. The keywords which do not have abbreviations are shown with in upper case. Keywords which are available as whole words or abbreviations have the significant portion of the keyword in upper case, and the optional portion in lower case. For example, the command ATTACH can be typed as either ATTACH or ATTA, but cannot be typed as ATTAC because this matches neither of the acceptable formats.

This list is for quick reference only. Use the detailed command descriptions which follow for more information on the use and options of each command.

3.1 ALPHABETICAL COMMAND SUMMARY

ACK	Clear a first out alarm condition.
AGA3	Forces recalculation of a range of AGA3 meters.
AGENDA	Control and display time of day event programming.
ALERT	Signal from One unit that it needs to be polled
ATTAch	Attach your console to another port.
BLOCK	Send a message in error-checked block format.
BACKdrop	Graphic Screen Display Initialization
BOX	Draw box on local CRT.
BREAK	Send a communications break for specified seconds.
BYE	Terminate the RTU communications session.
CAL	Calibrate channel with rapid display.
CALC	Do an expression calculation when not in CALC mode.
CALL	Call and connect to another unit.
CHANGE	Change the alarm mode setting for a channel range.
CLEAR	Clear totalizer and counter channels. Close text files for a task.
CLREOL	Clear Display to End Of Line
CLS	Clear The terminal screen.
CONFIG	Screen oriented channel, link, task, and system programming.
COPY	Duplicate DOS files.
CURSOR	Control and position the CRT cursor on the screen.
DATA	Specify data values for a range of channels.
DATE	Set the date for DOS.
DELete	Delete DOS files with confirmation.
DIAL	Cause task to dial-out to other unit.
DIR	Display DOS file directory.
DISABLE	Disable processing of a channel.
DISCOnnect	Disconnect radio modem.
DISPlay	Display channel and TASK data on screen.
DRAW	Draw Graphic Line On Screen
DUMP	Show channel and TASK configuration data
ECHO	Send the rest of the line to the communications line.
EDIT	Invoke the Text Editor program.
ELSE	Alternate conditional statement processing.
ENABLE	Re-Enable a Disabled channel.
ENDIF	End of conditional processing.
ENTRY	Screen Oriented Data entry and Display
EXEC	Execute a DOS command, including DOS itself.
FAX	FAX Report Generation and Control
FILE	Open, close, and delete output files.
FLAG	Manipulate System Wide Control Bit Flag

FONT	Graphic Font Control
FORCE	Send a command line to another task.
GOSUB	Start processing command file, don't close current file.
GOTO	Jump to a program label.
GROUP	Program and display Callout Group Info.
HALT	Stop the RTU program and return to DOS.
HANGUP	Disconnect the communications line. HAYES Process a Hayes Modem command line.
HAYES	Process Hayes Modem Command Line
HELP	Get help on RTU commands.
HOLD	Set a channel to Holding Status.
HORN	Turn local horn off and on.
IF	Conditional Statement Processing.
IMAGE	Automatic value saves.
INPUT	Enter channel or string data from console.
KEY	Temporary Control of Multi-Drop Line
LET	Does same as CALC.
LIBrary	Load command files into a procedure library in memory.
LINK	Program and display Communications Link info.
LOCAL	Declare local variables for use in calculations.
LOG	Control and Display of Alarm and Data log.
LOGON	Access the program through the password system.
MAP	Open Architecture Data Channel Mapping Control
MENU	Display main menu or load user menus.
MONITOR	Set port for watchdog timer monitor, and off-on control.
MOUSE	Microsoft Compatible Mouse Control
MSG	Display a message on the local console.
PAINT	Transparent Graphic Image Display
PASSWORD	View and modify passwords.
PAUSE	Prompt and accept input for command file continuation.
PHONE	Set the dial out phone number or Radio Call Sign.
PLAY	Audio Subsystem Control
POLL	Control callout polls.
PORT	Set output port and display input port values.
PRC	Process a Packet Radio Controller command line.
PROGram	Program a data channel.
PUBLIC	Declare global (public) variables for calculations.
PURGE	Purge waiting input characters from communications channel.
READ	Start processing of a command file.
REBOOT	Force a cold boot of the computer without going to DOS.
RELEASE	Free variable space after calculations.
REPORT	Generate an RTU data report to the printer or a file.
REREAD	Cause the current RTU file to be restarted from the top.
RESET	Reset alarm conditions on a channel or all channels.
RESYNC	Reset incoming and outgoing block counters.
RETURN	Stop execution of a command file.
RFLAG	Manipulate RTU Control Bit Flag
SAVE	Save current RTU Channel information to DOS Text File
SAY	Print a message or channel value on the screen.
SCAN	Request a range of data values from the RTU.
SELECT	Select a logical RTU as current one.
SET	Set RTU operational parameters.
SETDB	Set and/or display a channel's deadband.
SETHIgh	Set and/or display a channel's high alarm setpoint.

SETLOw	Set and/or display a channel's low alarm setpoint.
SETWAIT	Set and/or display a channel's alarm delay period.
SHELL	Shell out to DOS, just like EXEC.
SLEEP	Suspend console's task operation for a number of seconds.
STATus	Display Communications port and Task Status.
STOP	Stop execution of all command files for a task.
STORE	Write channel configuration data to standard output.
STUFF	Insert characters into another Task's input buffer.
TAB	Position Text Output to Specific Column
TAP	Passive TSP Multi-Drop Communications Control
TASK	Task control and information display.
*TERMINAL	Put program in human interface mode (manual operation).
TIME	Set DOS time of day.
TRANSFER	Copy Values from One Group of Channels to Another
TYPE	Display a DOS text file on the terminal.
UNHOLD	Remove a channel's Holding status.
USER	Display local CRT user menu.
VER	Display SCADAWARE Program Version and Settings
WAIT	Wait for event to occur with timeout.
WATCH	Monitor input and output characters for another task.
WIN	Graphic Window Control
WRITE	Output Delimited Text to Screen or File
XFER	Copy Values from One Group of Channels to Another (Transfer)

3.2 SCADAWARE LITE RESTRICTIONS

The DPMI Protected Mode Version of SCADAWARE has all of the features described in this manual. SCADAWARE LITE is missing some features, most notably the graphics, database, and open architecture features.

The Lite version is further restricted in that some commands are not available to tasks other than the local user's Task 0. This is because certain SCADAWARE LITE commands are contained in the program overlay file. These commands must be loaded from the overlay file (or from EMS memory if present) before they can be executed. Program limitations prevent overlay use by background tasks, thereby limiting access to these specific commands.

A few commands can be optionally loaded (with the LOAD command in the DAT file) at program startup to make them available for other tasks. When this is done, they are effectively removed from the overlay system and made part of the main program code. These loadable commands are PROG, COPY, DISP, DBASE, and REPORT.

4.0 DETAILED TSP COMMAND DESCRIPTIONS

The following section contains detailed descriptions of each RTU program command. This serves as a reference to each command describing its format and syntax. A summary of all commands is included in the previous section.

All letters of each command must be present and correct. However, certain common commands can be abbreviated to simplify manual entry of frequent commands (like using DISP for DISPLAY). All mandatory characters of a command are shown in upper case. If a command can be abbreviated, the optional part is shown in lower case. When processing a command, the RTU program does not distinguish between upper and lower case.

The parameters that follow the keyword may usually be abbreviated if desired. Often, only the first letter is significant. Some parameters, like OFF and ON, require at least 2 letters. If in doubt, simply enter the entire keyword.

Each command is listed separately with the exact command name, any allowed abbreviation, and a short description of the command's function. The number at the end of each line is the security level required to execute the command in computers which use SCADAWARE's password system.

ACK Clear First Out Alarm (2)

The ACK (acknowledge) command is used to remove the RTU's first out alarm condition. When a new alarm comes in, the program will attempt to report the alarm (and possibly blow a local horn). When ACK is used, the alarm condition for *all NEW alarms for the current RTU* is removed and the program is set to accept a new subsequent alarm. The horn activity is also stopped.

Alarms that return to normal after ACK, and are set to auto-reset, will return to non-alarm status right away. Others alarms will go into reset status and will require a RESET to occur (either by command or by push-button) to clear.

Note that ACK does not clear alarm conditions as is done with the RESET command unless the points are set to auto-reset. Also, any callout requirements caused by an alarm condition are not affected by the ACK command.

Example: ACK

Related: RESET

AGA3 Recalculation of AGA-3 Meter Channels (1)

The AGA3 command is used to force a complete re-calculation of a range of meters. This is required for meters that are not local to the system processing a command, or for meters that have long partial recalculation times. TEST's SCADA system operates as an *AGA3 gas flow computer* by calculating gas flow rates according to the American Gas Association Report Number 3. This is the recognized standard for gas flow throughout the worldwide oil and gas industry.

The manner in which the AGA3 calculation is performed on a particular system will depend on the availability of local input devices, the intelligence of the remote units sending the data to a host, and on the power of the computer being used to run the SCADA program. Management of the AGA3 calculations is based on reducing the amount of processor power dedicated to the meters such that adequate values are obtained without wasting the computer's resources.

Unlike most RTU's and PLC systems, TEST's SCADS system performs the *entire* AGA3 calculation in real-time as often as once per second. The entire crunch includes complete calculation of 'C-prime' using the entire AGA3 report tables and formulas. The complete crunch is rarely necessary in real-time because parameters like pipe size, gas gravity, and other components don't change at all, much less once per second. Because the AGA3 calculation takes quite a bit of computer power, performing constant calculations can waste this power by crunching numbers that never change.

To reduce the PC processor load, the program allows for a "pre-calc" to occur when any of the normally fixed parameters are altered. This is done whenever any AGA3 channel configuration parameter is changed, or with manual or automatic execution of the AGA3 command for a particular meter range. The partial calculation (that always occurs in real-time) accounts for pressure, differential, and temperature on each pass. The full calculation does the entire AGA3 report, including NX-19 gas supercompressibility (unless overridden by the channel configuration). The user *never* has to look up anything in an AGA3 reference in order to configure or operate the AGA3 gas flow calculator.

The AGA3 command is needed to force an immediate calculation of a remote meter when new input data is received. Otherwise, the complete calculation would be delayed for the number of calculation cycles specified in the meter's setup. For example, a host computer polling a dumb RTU only

needs to re-calculate when new data comes in over the communications line. In this case, the meter would be programmed to *never* pre-calc and an AGA3 command line can be placed in the BYE file so that only one re-calc takes place per transmission.

The AGA3 command may also be used in a command file that requests information on the gas gravity, plate size, or gas quality (CO2, N2) on an input screen. When the entry is complete, the program should do an AGA3 command on the affected meter so that the "per-calc" will occur immediately and the new values will be taken into account as part of the partial calculation stored in the meter's setup.

Examples:

```
AGA3 M1
AGA3 M2:M3

; Program stub to enter new meter date for meter M1
input, Enter New Gas Gravity, V4
input, Enter New CO2, V5
msg Doing the AGA3 pre-calc
AGA3 M1        ; force immediate pre-calc
```

AGENDA Time of Day Procedure Control (2)

The AGENDA system allows the program to perform pre-programmed tasks at specific times of the day. The program monitors the time of day in order to send commands to tasks at the specified times. These commands may only contain a single line, but that line can be a READ command which will cause a more extensive command file processing sequence.

The agenda system is checked once per second, so commands can be timed to occur on one second intervals. Each task can receive as many commands as desired at any time specified, and more than one task can receive commands at the same instant.

The AGENDA LIST command will display all entries in the agenda list and also indicate which entry is the next to be executed for each task. An optional task number can be specified after the LIST keyword which will cause the display to include only the entries for that task. If the AGENDA command appears by itself on the command line then the parameter of LIST is assumed.

The agenda process can be halted and re-started with OFF and ON options. The Agenda CLEAR will eliminate all entries and turn processing off. The Agenda RESET command will restart processing of the agenda list from the top (midnight) if no time is specified, or from a specified time if one is given. This is helpful in testing an Agenda list because you can trick the computer into thinking it is earlier or later (with the Time command) and then moving the internal Agenda pointer with the RESET option. This beats hanging around till 6:00 AM the next day just to test an Agenda activity.

Entries into the agenda list are also made with the agenda command that specifies the task to receive the message, the time to execute, and the message itself. Normally, commas are used to separate the entries on the line so that the message portion can contain embedded spaces. Entries for each task in the agenda list *must be entered in the order of time to be processed.* The agenda list does not automatically get sorted by time. For example, consider the following incorrect agenda list entries:

```
agenda, 1, 09:00:00, Read UPDATE.RTU  ; WRONG ENTRY!!!
agenda, 1, 06:00:00, Poll Now
agenda, 2, 06:00:00, Poll Now
agenda, 1, 10:00:00, Report
```

When the agenda is started it will search down the list for the "next" entry to process for each task. It will find that entry 3 is the next entry to be processed by task 2. Since this is also the only entry for task 2 there can be no problem with the agenda for task 2. Processing the agenda entries for task 1 can be a bit more troublesome.

When searching for the "next" entry to process for task 1 the agenda system will first find entry 1 and wait there until it is time to process it. When that entry gets processed the agenda will continue down the list searching for the "next" entry to process. It will then find entry 2. Since the time for entry

2 to be processed has already passed, the agenda will immediately process the entry and look for the next entry. It will then find entry 4 and wait there until 10:00:00 to process it.

Now consider the following example in which agenda entries are properly listed:

```
agenda, util, 06:00:00, Read DAILY.RTU
agenda, util, 06:20:00, Report
agenda, 1, 06:05:00, Poll RTU smi9
agenda, 1, 06:06:00, Poll RTU vr253
agenda, 2, 06:07:00, Poll RTU hi92
```

Because of the way that the agenda system works, it is up to the user to insure that all agenda entries are listed in the proper order. Usually, all agenda entries are listed in a single file (named AGENDA.RTU) and the file is read to load the agenda. If an agenda entry needs to be added to the list it can first be added to the file, the agenda then cleared, the file read again, and the agenda turned back on. The name of the file that contains the agenda entries can be anything but usually AGENDA.RTU is used. The edit, clear, and reload actions are also normally pre-programmed on a utility menu supplied with the system.

Note that the RESET option will process all agenda entries up to the time that the agenda is reset to. The ON option will activate the agenda by starting it at the current time. Turning the agenda on will not cause any entries prior to the current time to be processed.

Examples:

```
AGENDA      ; display all entries in agenda list
AGENDA LIST   ; display all entries in agenda list
AGENDA LIST 2  ; display all entries for task 2
AGENDA ON     ; activate agenda process
AGENDA OFF    ; stop agenda process
AGENDA CLEAR  ; stop agenda process and clear list
AGENDA RESET  ; restarts agenda from midnight
AGENDA RESET 12:00:00  ; restarts agenda from noon
AGENDA, 0, 09:45:00, read UPDATE.RTU ; for task 0
AGENDA, util, 23:00:00, report     ; for util task
```

ALERT Signal that RTU Needs Attention (1)

ALERT is a command sent from non-PC based RTU's for call-on-exception processing. A small RTU such as the Multi-Drop Type 2200 SCADA Node will detect switch changes and signal that it needs attention by broadcasting an ALERT command followed by its RTU ID. The SCADAWARE computer processing the ALERT will scan its LINK table for a match between the Alert RTU ID and the Link's Phone Number/Call Sign field. If a match is made, the associated LINK is activated and processing continues normally.

The small RTU sending the ALERT command will continue to do so every minute for one hour, or until a proper message is received by that unit. The Alert status is normally cleared by polling an RTU, although any successful message into the RTU will clear the Alert.

Only one SCADAWARE computer on a multi-drop link can respond to ALERT messages. Therefore, the SET ALERT OFF/ON option is provided to allow tasks on each computer to respond to Alert messages. The SET ALERT ON should be placed in the STARTx procedure for the affected task.

The ALERT action can be tested by simply entering ALERT RTUNAME at the local console. The program will scan the link table trying to match RTUNAME, and if found, will activate the appropriate link.

Related: Set Alert ON

ATTAch Attach Console to Another Task's Serial Port (2)

The local keyboard and CRT are "connected" to task 0 of the SCADA program. Any keystrokes entered affect task 0 directly. Modems and other communications devices are connected to very similar tasks inside the program. However, the characters sent to and from the other devices are invisible because tasks have no access to the local CRT. These other tasks control their serial line in the same way that task 0 controls the local CRT and keyboard. They are separate sessions within the program that normally have no reason to cross-connect with one another.

Sometimes it is convenient to use the CRT/Keyboard to "talk" to a communications device connected to another task. This would be like getting inside the program for a moment in order to send and receive characters that are normally destined for another task. For example, it may be helpful to send keystrokes to a modem (to make it dial) and see the responses that are returned from the comm line. Without the ability to temporarily connect to the modem, a separate communications program would have to be used to perform the test.

The ATTACH command is used to connect the local console to the comm port of another task. This is handy when the operator wants to manipulate the communications channel for diagnostics or other special purposes. In effect, this is a mini-terminal emulation program built into the program itself.

Because each serial channel is normally connected to an RTU task, ATTACH puts the other task to sleep for the duration of the ATTACH command. While attached, the local console keystrokes go out of the affected port and all incoming characters are displayed on the local CRT. This condition will remain until the attachment is ended with the special key sequence ESC-X (two keystrokes). When the attachment is ended (by hitting ESC-X), the suspended task is restarted just where it left off. The affect on that task is unpredictable, especially if the task was not operating in the fist place when the ATTACH was started.

Commands that are processed in RTU mode contain extra characters, such as checksum characters, which are used for error checking. When attached to a task in RTU mode, all error checking requirements that are usually added to a command automatically must now be typed in manually. Entering the command ^A*TERMINAL will put the program into terminal mode. This will allow commands to be entered without the error checking requirements.

Example:

```
ATTA 1
ESC-X    (key sequence to end the attachment)
```

Will suspend task 1 and attach its serial port to the current console.

BACKDROP Graphic Screen Display Initialization (1)

Graphic screens can be built on the "backdrop" of a graphic screen contained in a PCX or GIF file. They can also be built on blank screens filled with any color. The BACKDROP command provides the means to initialize the graphic screen with any file or color, and to optionally frame the screen in any standard graphic frame type.

The command provides offset options to locate the file at any X-Y position on the screen. Normally the entire screen is used, but this option is provided for special cases such as providing a rectangular area filled with color at random positions on the display.

Refer to the Graphics System Documentation for more information.

Examples:

```
Backdrop None Blue    : plain blue background
backdrop FLOWMET 0 0 double  : display FLOWMET.PCX with a double edge frame
backdrop flowmet 10 20 single   ; flowmet.pcx offset 10 pix right 20 down with single
frame
```

Related: WIN, PAINT

BLOCK Transmit Command To Remote Unit (2)

The basis of TSP is that text lines are processed by an RTU type task. Sometimes the local computer needs to process a line, and other times we want to send a line to another unit so that the line will be processed *over there.* The command lines are very similar. The only difference is where they get executed.

While in RTU mode, the SCADA system uses error-checked block transmissions (TSP Packets) for all data transfers. The BLOCK command is used in a command file to send a TSP command to another system. For example, while processing a download file at a Host system it is likely that several BLOCK SCAN commands will be processed. If only a SCAN command was used, the local computer would process the line (and send the result to the RTU). By using a BLOCK SCAN, the Host will send everything after the BLOCK keyword to the RTU where the line will then be processed. The resulting DATA line will automatically be blocked over from the RTU to the Host as a response.

One consequence of the BLOCK command is that the system will wait for a communications response from the other unit indicating that the message was received. When the response is received, execution continues normally with the next line of the file. If the response is not received, the transmission is repeated several times according to the TSP protocol controls in effect. If still no luck, the communications link and command file processing are automatically terminated. Execution continues, but other commands in the RTU file that rely on in-tact communications will not process properly. The SET ONLINE command can be used to have better control over a problem like this.

Lines processed by the block command are "expanded" before transmission to the other unit. This expansion will do a special code replacement similar to the ECHO command. This allows for system or run-time information to be placed on the line automatically just before the line is transmitted. One common use of this feature is to set the time and date on the other unit with BLOCK TIME $T and BLOCK DATE $D commands. At run time, the $T and $D will be replaced with the actual time and date of the sending unit.

The receiver of the BLOCKed transmission can supply a response line along with the required ACK. This allows for a system to BLOCK over a SCAN command and wait for the normal DATA response before proceeding with local command file processing.

All comments are stripped from a line prior to transmission to the other system. This shortens the amount of characters that must be sent when doing block transmissions.

Example:

```
Block SCAN A1:a10 E S1:S16 R ; request data from a remote
Block DATA RTU1.A1:A3 E 100 200 300 ; send data to a remote
BLOCK SELE GA343A  ; select another RTU
BLOCK CALC T2 = 30 ; pulse timer on remote unit
```

Related: ECHO, PRC, HAYES, SET ONLINE

BOX Draw Textmode Box on Display (1)

This command can be used only by the local task. It is used to draw a box on the screen which is most useful for enhancing custom displays. The command can be used in a command file along with the CURSOR and SAY commands to create a custom display screen. The x (column number) and y (row number) coordinates for the upper left corner and the lower right corner of the box must be specified with this command. The upper left hand corner of the screen is at position 1,1. An optional 5th parameter is allowed which will be displayed as a string in the title of the box.

Examples:

```
BOX 5,10,60,22        : box at col 5 starting on line 10 thru line 22
BOX 5, 5, 75, 25, This is the title
```

Related: CURSOR, SAY

BREAK Send Timed Break Signal Over Comm Line (2)

Normally, a communications channel sits idle in a MARK, or 1 state when not transmitting data. The data line is never low more than one character time, even while transmitting a character that is all 0's because of the stop character that is present at the end of each async character. A BREAK signal is defined as the presence of a SPACE (0) for more than one character time.

TEST's Modem Monitor device can time a break signal and initiate a CPU reset if the specified time is exceeded. The BREAK command is used to send the required break signal. The only parameter is the *number of seconds* that the break will occur.

A typical sequence is to DIAL an RTU, WAIT for a carrier detect, and then use BREAK x to perform the reset.

Example:

```
DIAL
WAIT 30 CONN
BREAK 5  ; send a 5 second break to cause a reset
```

BYE Terminate RTU Communications Session (1)

The proper way to terminate an RTU session is with the BYE command. Although the program will normally time out after a specified period of inactivity, using the BYE command will provide an orderly shutdown of the session. BYE allows faster processing of multiple connects because the normal delays associated with timeouts can be eliminated. It can also reduce Cellular costs that are charged in chunks of 5 or 10 seconds because the normal 30 or 60 second timeout can be eliminated.

BYE affects the physical and logical on-line state of the task processing the command. Many communications media use the physical carrier detect (DCD) line as a means of verifying the state of the comm line. Others, like multi-drop, must rely on internal timing and other parameters to determine the logical state of the connection. The Bye command affects all aspects of the on-line state. A task processing a Bye command will go through various steps to disconnect whatever communications device is connected. It will also reset the command counters for both incoming and outgoing messages so that a fresh communications session can begin with the next call.

The BYE command handles ACK responses in a special manner. This is because the comm line is in transition at that point, and sometimes a response to a BYE command will confuse the issue. The basic rule is that the BYE command suppresses the normal TSP ACK requirement *unless the BYE originated from the incoming communications line.* So, a BYE processed in a command file will cause an immediate hangup, while a BYE sent from another unit will cause an ACK response (like any other command) prior to the disconnect.

When BYE is processed, the RTU will disconnect from the communications line and will process a special command file if it exists. This file is named BYEx.RTU, where the x is replaced with the task number. For example, BYE1.RTU is the bye program for task 1. This program can consist of special communication line setups or whatever else the operator wants to have happen each time the program terminates communications. For example, a typical BYE file might appear as follows:

```
set echo off  : make sure task is full duplex
prc echo off  : set PRC to full duplex
prc maxframe 1 : no outstanding PRC frames allowed
prc tclear    : dump any chars waiting to transmit
prc mycall smi4 : my name will be SMI4
msg Task 1 Disconnected : tell local console what happened
```

Related: RTU, TERMINAL, SET BYE

CAL Calibrate Channel With Rapid Local Display (1)

When calibrating an analog channel, it is often desirable to have a quick-response display to assist in the adjustments. The CAL command provides a rapid, one line display of a single channel. Normally, analog inputs are used although any channel type can be specified.

The parameters on the line include the required channel identifier (ID number or tag), and an optional wait count. The wait count is the number of system ticks (1/18 sec) that the program will wait between displays. If none is specified, then a value of 9 is used providing a display that is updated about 2 times per second. Lower waits provide quicker displays, but occupy more processor time. Slower displays may often be used because analog inputs are only updated at the speed of the DRIVER task anyway. Displaying faster than the update rate will not produce any additional speed in the display. Therefore, it might be desirable to use a command such as TASK DRIVER DELAY 1 to speed up the driver task during calibration.

A counter is provided to show how often the display is being refreshed. This counter has no effect on the actual value being displayed.

Note that it may be desirable to temporarily clear any negative number restrictions on analog channels during the calibrate process. The change is made on the Analog Channel Config Screen for the affected channel. Eliminating the negative value restriction will allow easier adjustment of the zero point. Remember to restore the correct negative flag for the channel after calibration.

Examples:

```
CAL A1       ; accept default update rate
CAL A1 18    ; wait about a second between update.
CAL A1 1     ; very fast display
```

CALC Access TSP Expression Evaluator (2)

Normally, the expression evaluator will not process a line unless the user is in CALC mode. Single lines can be processed by preceding the line with the CALC command. For example:

CALC TOTAL = TOTAL + DAYTOTAL

would cause the contents of the variable (or channel) DAYTOTAL to be added to the variable TOTAL. Without the keyword CALC, the command processor would not be able to complete the line because the parser would think the desired command is called "TOTAL". Any valid TSP expression can be put on the right of the equal sign. Several special keywords are also allowed a mentioned below.

If a number of expressions are being processed, the SET CALC ON command can be used. This will allow automatic execution of any line that is not processed as a normal command. When in effect, the user's prompt indicates [CALC] to warn the user that any lines entered will be processed directly. Although this method is sometimes convenient, it is quicker to process commands when the keyword CALC is specified.

All channel types can be set to any value using the CALC (or DATA) command. Also, channels that have time and dates associated with them (i.e. counters and totalizers) will have the time and date automatically set to the current time and date when ever these channels are set to exactly 0. This allows for daily processes to clear the counters and totalizers and reset the starting time in a single command.

When using the CALC command, the keywords OFF and ON can be used to set a channel or variable to the values of 0 and 1 respectively. Setting Timer Channels Off will stop them from counting and bypass any alarm action that would have occurred when the timer reached 0 by itself.

String variables can also be set using the CALC command. String variables begin with the percent sign, and are referenced by number: %0 through %9. String expressions can be placed in a string variable. Each parameter on the line after the = sign is added to the string. Quotes can also be used to prevent text from being expanded. For example:

```
CALC %1 = test 123
```

would produce a string like test 123, while the line

```
CALC %1 = test 1  2  3
```

would produce a string like test 1 2 3.

The evaluator will display error information if any problems are encountered when processing a line. For example, if the "target" variable on the left of the = sign is invalid, the system will indicate this with a "TARGET RANGE INVALID" or "COULD NOT SOLVE" line being sent to the task's console (if it has one). The user can control further file execution after any errors with the SET ERROR command. Setting this ON will cause the file to stop on any evaluator error. The default setting is OFF, so errors will not normally stop file execution.

Examples:

```
CALC O1:o8 = 0   ; Turn off all outputs
CALC %1 = SYSTEM $S
CALC T5 = 0   ; stop timer and cancel any alarm action
```

Related: LET, SET CALC, SET ERROR

CALL Attach With Local Echo (2)

CALL is similar to the ATTACH command except that the program processes local characters slightly differently. In CALL mode, the program will locally echo the user's keystrokes rather than waiting for them to come back from the other unit. This assists operation in half-duplex radio installations where there may be a 10 second delay in the PRC transmissions.

CALL mode also monitors the state of the Carrier Detect line from the communications port. If the CD is lost, the CALL mode will automatically terminate and the user is returned to normal command mode.

As in the ATTACH command, the task normally connected to the serial port is suspended until termination of this command. The results of stopping the task are unpredictable.

Terminate the CALL command with an ESC-X sequence.

Example: CALL 1

Related: ATTACH, DIAL

CHANGE Modify Mode Settings Options for Channel Range (2)

Every channel has an "alarm mode" setting, which actually controls more than just its alarming actions. This setting determines how the channel behaves to abnormal conditions. Internally, the setting is a 16 bit word with each bit having a specific purpose. The alarm mode is one of the first things that is programmed during the channel program sequence.

The Skip Reports, REMOTE, HOLD, and ENABLE settings for each channel are also modified with the CHANGE command.

The CHANGE command allows individual settings to be set or cleared for a range of channels. This can be helpful when transferring a channel setup from an RTU to a HOST. For example, it may be desirable to set all channels to NOT call out at the host station. This can be done easier with the CHANGE command than by re-programming each channel individually just to change a single setting.

The command works by taking the specified channel range as the second parameter and a number of "off-on" switches on the remainder of the line. These switches are 1 or 2 letter abbreviations for the setting to be affected. Entering no switches will cause the command to display a quick reminder of the valid options. The options for the CHANGE command are:

OPTION	EFFECT IF SETTING IS ACTIVE
AC	Set channel to be an Alarm type channel.
CA	Call on abnormal conditions.
CR	Call on return to normal (reset needed).
AR	Set channel to auto reset on return to normal.
CF	Execute command file on abnormal.
LO	Log all abnormal conditions.
HO	Activate local horn on abnormal.
SR	Skip this point during report generation
PL	Play Message on abnormal
H	Hold status for the channel.
R	Remote status for the channel.
E	Alarm Enable status for the channel.
D	Dump (Display) current settings (does not set anything).

Plus and minus signs are used to determine if the option is a set or clear operation. For example, entering -HO will set the specified channels to NOT blow the horn on alarm. Using +CR will set the channels to call on reset. The plus and minus can lead or follow the option code. No plus or minus will default to plus. Thus, +CR is the same as plain CR. At lease one space must appear on the line between each option.

The DUMP option allows a short display of the mode settings for a channel range. The option, "DUMP", can be specified anywhere on the command line where a change option is allowed. The dump will represent the state of the range at the time the DUMP parameter is encountered. Thus, any changes caused by options specified after the DUMP keyword will not be reflected in the display.

For example, entering CHANGE V7:V15 CA- CR- DUMP will cause channels V7 through V15 to be changed to NOT call on alarms or resets. After the changes are made, a display will be generated showing the complete alarm mode settings for this channel range.

Although the change command is most often used directly at the keyboard, it can be useful in command files to change the behavior of some channels on-the-fly. For example, it may be desirable to turn off the horn status of certain channels at night. This would prevent awakening the operator for non-critical alarms. To do this, a command file would be run (by hand or as an Agenda item) to change the affected points to HO-, and another file could be run to change them back to HO+ at the appropriate time. A single file could also be used by passing the desired condition (HO+ or HO-) as a command line parameter from a menu hit.

Examples:

```
Change S1:s8 +AC CR- CF+ +HO
Change A3:a5 HO+ DUMP
Change Q1:q4 H-  ; un-hold some channels
Change A1:A6 $1 S1:s4 $2 ; Use command line parameters
```

Related: PROGRAM, HOLD, UNHOLD, CONFIG

CLEAR Clear Totalizers and Counter Channels (2)

There are two options that can be used with this command. The keyword ALL can be used to clear all totalizer and counter channels to 0. This is basically just a convenience to clear all accumulator type inputs at the same time. Individual channels can be set with direct assignment statements such as CALC Q1 = 0.

The keyword FILES can be used to close all open text files for a task. This command is mainly provided so that files can forcibly be closed in the event of an error.

Example:

```
CLEAR ALL
FORCE 1 CLEAR FILES
```

CLREOL Clear Display to End Of Line (1)

Clears the current display line from the cursor position to the end of the line. SCADAWARE uses the proper screen control codes depending on the display type specified in the SET CRT option for each task. The cursor remains at its current position, but all character positions to the right are filled with blanks.

Example:

```
cursor 10,12  ; col 10, line 12
clreol        ; clear to the right
say a1        ; print value of channel A1
```

Related: SAY, CURSOR, CLS

CLS Clear Display Screen (1)

This is essentially the same as the DOS CLS (Clear Screen) command except that it also works with remote ANSI CRTs as well. The terminal or console screen is cleared and the cursor is positioned at the top line, left corner. The screen control codes used for this command (and other CRT related commands) depend on the type of CRT installed with the SET CRT command.

If it is desired to "home" the cursor to the upper left hand corner but not clear the screen, use the CURSOR 1,1 command instead of CLS.

Related: SET CRT, CURSOR, SAY

CONFIG Screen Oriented System Configuration (3)

CONFIG allows screen editor type configuration on the local CRT screen. CONFIG can only be used on the local CRT and keyboard. The CONFIG command can be used to program channels, links, tasks, database, entry screens, and system variables. When using CONFIG, the user can move from field to field making changes and then save the changes with a single keystroke. A Microsoft (C) compatible mouse is also supported in many editing functions. Note that the CONFIG command can be easily accessed from the built-in menu system to get direct access to any configuration command option. The menu system simply types the proper CONFIG command for you.

To configure a channel using this command, enter CONFIG followed by a channel identifier (such as CONFIG a3). The program will display all of the parameters for the channel, and allow editing of each parameter individually. The cursor can be moved using the arrow keys, and can be moved between channels with the PGUP and PGDN keys on the keyboard. The ESC key is used to exit the configuration mode without saving the changes. The F2 key is used accept the changes in memory and exit the configuration mode.

NOTE: Saving most changes in memory will allow the system to remember the current setup only as long as the program is running. If the program is halted for any reason (purposely terminated, power failure, etc.) the current setup will be forgotten unless it is first saved to disk. Changes made during a configuration can be permanently saved to disk by using the SAVE command or selecting SAVE CONFIG from the main menu. This type of save will store the current configuration in a disk file that the program reads each time it is started.

If the system has a mouse attached, it can be used to select a field by placing the mouse cursor on the desired field and pressing the left button. In the case of Y/N fields, the current value can be toggled by pressing the right button.

To configure a link, a task, or the system variables the CONFIG command works just like it does for configuring channels. The options available for use with the CONFIG command are:

```
chan id      ; program a channel such as s1, v2, etc.
LINK [xx]    ; default is link 1
TASK [xx]    ; default is task 0
SYSTEM       ; program variables used by the program
ENTRY        ; user defined entry screens
```

The following CONFIG options are not included in SCADAWARE Lite:

```
DBASE or DB  ; Database Fields
ANN          ; Annunciator Graphic Object
MET          ; Meter Graphic Object
VBA          ; Vertical Bar Graphic Object
HBA          ; Horizontal Bar Graphic Object
SCR          ; Scroll Graph Object
GRAPH        ; Graph of current window type
```

Note that this command only works on the local CRT. To program channels or links remotely over a serial line, the older PROG command must be used. *Note: In SCADAWARE Lite, the PROG command is an overlay and must be specified with a LOAD command in the system's DAT file if it is to be used from a remote terminal.* Just as with the PROG command, changes made with the CONFIG command take effect immediately, but are not saved until a SAVE or LINK SAVE command is issued.

Examples:

```
CONFIG v1
CONFIG LINK 2
CONFIG SYSTEM   ; configure program variables
CONFIG TASK 2
```

Related: PROG, LINK, SAVE

COPY Duplicate DOS Files (2)

This is a simple, single file copy utility provided for use within SCADAWARE. The syntax is similar to a DOS Copy, except that wild card characters are not supported. To avoid confusion, the source and destination file names should be completely specified. If a file extension is not specified, the default extension .RTU will be used. This default is similar to the program's DIR command. A file can be copied to a particular drive by specifying the drive in the destination file name.

Note: In SCADAWARE Lite, COPY is usable only by the local task.

Examples:

```
COPY shutin.rtu wellopen.rtu
COPY shutin wellopen      ; same as above
COPY shutin.bak          ; copy shutin.bak to shutin.rtu
COPY shutin A:          ; copy shutin.rtu to drive A
```

Related: TYPE, DEL

CURSOR Locate and Control Display Cursor (1)

The CRT's cursor is the blinking line or block that indicates the current position on the screen. When writing script files, it is sometimes desirable to locate the cursor at a specific location prior to printing a message.

The basic syntax is to provide the X=COLUMN and Y=ROW coordinates for the cursor. If an entry is invalid, the location of the cursor is unchanged. All positions start with 1, and continue to the maximum supported by the user's CRT. So, position 1,1 is the top left corner of the screen. Position 80,1 is the upper right corner, and 80,25 is the bottom of the typical PC screen.

The number of columns is 80 for all currently supported CRT types, but the number of lines depends on the CRT type. The local CRT can be 25, 43, or 50 for MONO-CGA, EGA, and VGA screens respectively. Remote ANSI terminals usually have 24 lines.

An optional keyword can be used to cause the cursor to be hidden or returned to normal. Thus, CURSOR OFF will cause the cursor to become invisible, and CURSOR ON will restore the standard cursor for that CRT type. This option may not work on certain types of terminals and the cursor will always remain on.

If more than 3 parameters are on the line, the fourth (and remaining parameters) is displayed as a text message, an expression evaluation, or as a channel. Text messages can be enclosed in double quotes to maintain spaces or commas within the text. TSP Expressions must be enclosed in parentheses, and channel references can be either the channel tag or number. Formatting characters can also be used to control a display on the screen. These codes follow the parameter with an accent (') character. See the SAY command for more information on how to display information using this command.

TEXT IN GRAPHICS MODE

The protected mode version of SCADAWARE can display text in any supported graphics mode. The CURSOR and FONT commands are used to style and position the text output. In graphics mode, the column and row parameters are specified as either a percentage of the entire screen, or as pixel values. As with all SCADAWARE graphic coordinates, position values less than 1 are considered percent of screen. Values of 1 and greater are considered pixel coordinates.

For percentage placement, position 0.00,0.00 is the top left corner of the screen, while 0.99,0.99 is bottom right corner. SCADAWARE will calculate the proper pixel coordinate based on the current screen resolution. In pixel mode, 1,1 is the top left, while the bottom right will depend on the current graphic resolution. For normal VGA, the bottom right corner is at 640,480.

The size and style of the graphics characters is determined with the FONT command. Various options for font style, size, color, and justification are available.

Examples:

```
CURSOR 1,10  ; line 10 column position 1
CURSOR off
CURSOR on
CURSOR 1 1 "The current time is $T"
CURSOR 10 5 V3'R10  ; chan V3 right justified in field of 10
```

Related: CLS, SET CRT, SAY, FONT

DATA Reply from SCAN for Data Values (2)

DATA lines are the primary means of sending RTU channel information from one unit to another. DATA lines are normally generated as a response to a SCAN line, although they can also be entered as simple text lines. When an RTU is uploading channel data to another computer (in response to a SCAN command), it sends each line of information in the form of DATA lines. These lines indicate a range of channels followed by the appropriate values. The SCAN command is given with a range of channels, and the DATA command represents a response with that range. The values are listed one after another on the line in ascending channel order. If no options are given in the SCAN command, the values are returned in engineering units representing the current value of each channel.

DATA commands can also be used in a command file to allow setting of channels to specific values, just as if they had come from another unit. This is helpful is establishing initial values as well as testing without actually connecting to another unit.

A special form of the command is DATA @ENDS. This indicates the end of an upload process and causes the receiving system to set the update time and date to the current time. A specific RTU can be indicated after the @ENDS keyword. If not specified, the current RTU for the processing task will be used. A time and date can follow the RTU parameter to specifically set the timestamp instead of letting it default to the receiving system's current time and date.

Multiple channel types can be present in the same line. Each group (except the first) begins with a back slash (/) character. Otherwise, the extra groups are similar to the first one. This multiple grouping allows for faster transfers in half-duplex radio systems where transmission overhead is a significant portion of the download time. The only restriction is that all channel references in the same line must refer to the same RTU. This RTU is either specified in the first range or assumed to be the current RTU if none is specified. Combining channel types cuts down the number of transmissions required to complete a download.

Refer to the description of the SCAN command for details on the various options and formats for the DATA command.

Examples:

```
DATA RTU1.S1:S24 R 4 16 8
DATA RTU2.S1:s8 R 12 /01:o8 R 4 /A1:a2 E 100 200
DATA RTU1.A1:A8 R 1022 450 344 756 124 544 0 77
DATA RTU1.A1:A4 E  12.3 16.3 17.3 122.5
DATA @ENDS
DATA @ENDS RTU1 09:00:00 06/12/91
```

Related: SCAN

DATE Set System DOS Date (2)

This sets or displays the DOS date which is used by the program and any other DOS function. Note that this command does not always set the "watch" that maintains time in the computer when power is removed. It appears that for most AT type computers and above, operating under DOS version 3.3 or later, the watch does get set simply by using the DATE command. However, on some machines, setting of the watch requires a separate DOS program that is machine dependent.

For example, PC type systems often use the PCCLOCK program, while AT class computers use the ATCLOCK program to directly access the built-in timekeeper. Either of these programs are run as a DOS command by shelling out to DOS using the EXEC or SHELL commands. To find out if a computer requires one of these separate programs, set the date using the DATE command, turn the computer off and then on again, and use the DATE command to see if the new date was saved. If so, a separate program to set the watch is not needed.

Many 8088 PC level computers use a Dallas Semiconductor "Smartwatch" device to maintain the system time and date. These devices are accessed outside SCADAWARE by a program such as PCCLOCK. SCADAWARE has an option to access the Smartwatch during normal operation, making the Smartwatch act similarly to the AT style CMOS clock device. To use this option, a LOAD SWATCH line must be installed in the DAT file for processing during startup. This option does not apply to 286 and higher machines which have built-in CMOS time chips.

Entering the command DATE with no options causes the current value to be displayed. Entering a properly formatted date as a parameter will cause the date to assume that value. As a second parameter, a properly formatted time can be specified which will cause the time to assume that value. This allows both time and date to be set with a single command line.

Systems being used in different countries may need to include a line in the config.sys file to control the date format. The following commands can be used in the config.sys file to cause the dates to appear as follows:

```
COUNTRY = 001  <-- United States  mm/dd/yy (DOS Default)
COUNTRY = 003  <-- Latin America  dd/mm/yy
```

Example:

```
DATE 01/01/88          ; will set DOS to New years day 1988.
DATE 01/01/08 12:00:00       ; will set DOS date and time
```

Related: Time

DEL Delete DOS File (2)

This is similar to a DOS Del or Erase command, including support for wild card characters. Unlike DOS, the program will prompt for "OK to delete" for each file name that matches the specification provided as the first parameter. This confirmation can be overridden by using the switch /N after the file specification.

Examples:

```
DEL ab*.rtu ; delete all rtu files starting with AB
Del *.RTU /N ; del all rtu files without confirmation
```

Related: Type, copy

DIAL Make Dial-Up Connection to Remote Unit (2)

This command is normally used in file processing to have a task begin a connection to another unit. A parameter can be provided after the DIAL keyword to specify the phone number, call sign, or Multi-Drop ID to be accessed. If no parameter is given, the default phone number or radio call sign for the currently selected RTU is used. The default phone number and call sign is set during the Link configuration, or optionally with the PHONE command.

The DIAL command will start the connection sequence for any type of communications media, including those that do not actually "dial" anything. The action depends on the setting established with the SET MEDIA command. The required delays for the modem will be provided as in any normal dialout procedure. Direct connect and multi-drop connections do not dial at all but do set their modem control signals and internal controls to start a communications session. In any case, processing will continue immediately after the DIAL is complete. The WAIT command is normally used just after the DIAL command to allow time for the call to go through.

Example:

```
DIAL          ; Phone to default number
DIAL 837-3699     : Phone dial out
DIAL MP36A      ; Radio type dial out
DIAL 0              : special Broadcast mode for multi-drop
```

Related: SET MEDIA, WAIT, PHONE, HANGUP

DIR Display DOS File Directory (1)

This is very similar to the traditional DOS Directory command, except that the output can be paged for easier terminal viewing. The only parameter is a file specification like *.dat to request a directory display of a particular set of files. The default file extension is RTU. So, entering DIR VR* is the same as entering DIR VR*.RTU.

Examples:

```
DIR *.RTU
DIR START*
```

Related: SET PAGE

DISABLE Temporary Alarm Disable for Channel Range (2)

Sometimes it is desirable to temporarily disable the alarm action of a channel. For example, if a piece of monitored equipment is being repaired many nuisance trips may occur. The DISABLE command allows the point to be ignored and not be processed by the system. Disabled points are indicated as disabled in reports and on displays, but no other action is taken on them.

It is a good idea to disable all points that are spare so that they will not clutter displays or cause nuisance alarms. The disabled state is not stored in the channel configuration file with the SAVE command, so disabling a point and then SAVING will not cause the channel to be permanently disabled. However, the disabled state does get stored in the data image if image saving is turned on. This will cause the channel to remain disabled, even after system restarts. See the IMAGE command for more details on saving the data image.

A range of channels can be specified to disable a number of points in a single command. If required, several ranges can be specified on the same line.

Example: DISABLE S10:S16 m1:m3 a4:a5

Related: ENABLE, CHANGE

DISCONNECT Packet Radio Modem Disconnection (1)

This command is implemented to handle the DISCONNECT response from a packet radio controller (PRC). Normally, all responses are ignored. However, the DISCONNECT response is detected and processed in a manner similar to the BYE command. This command should never have to be used by anyone and is only there to avoid problems with disconnects from some brands of packet controllers.

Related: BYE, PRC

DISPlay Display Channel Data (1)

This is the primary on-line data display command. The parameter required is the starting channel to be displayed. Only about 20 lines can be displayed on most terminals, and the program will limit the display to this number if more are specified. Note that all display actions can be accessed via the built-in main menu system reached with the F9 function key.

The normal use of the command is DISPLAY followed by a channel type. For example, DISPLAY T will put up a timer channel display starting with the first timer channel. To display a specific channel use the DISPLAY command followed by a channel tag. A display can also be started from the main menu or by pressing the F7 key.

The DISPLAY command can also be used to locate new alarms that have not been acknowledged by the operator. The DISPLAY FIRST command will display the channel that first went into alarm for the current rtu. After the First Out alarm is acknowledged, the DISPLAY FIRST command will cause the system to search the current RTU for new alarms using an internal channel order. When no new alarms for the current RTU are found, the system will scan other RTUs for pending alarms. When no new alarms exist, a message is displayed indicating there are no new alarms. The DISPLAY FIRST command can also be initiated by pressing the SHIFT-F7 key.

Normally, the DISPLAY command will operate in a continuous display mode where the screen is updated as fast as possible. If the SET CONTinuous OFF command is used, only a single screen will be displayed per entry of the command. Local terminals (i.e. high speed) will normally have continuously updated displays, while slow remotes will get a single display screen by setting continuous off. The display is terminated with the ESC key.

During continuous displays, the operator can switch channel types by entering a letter to indicate the desired channel type. The selections are shown in a prompt bar at the bottom of the display. Only channel types available on the particular configuration will be shown in the prompt.

The + and - keys can be used during continuous displays to move among the various RTUs on the system. However, the program will not allow movement into RTU 0 using these keys. If displays are longer than a single page, the U or PGUP keys can be used to move up, and the D or PGDN keys can be used to move down. Also, the up and down arrows can be used to scroll a single line at a time.

Each channel on a system has an alarm time and date associated with it that is used to keep track of the last time the channel went into alarm. During continuous displays the [ENTER] key can be used to toggle between several display modes. One mode will display the alarm time and date and the other modes will display other relevant channel information. For Output type channels, the ENTER key has no effect on the display. No toggling will occur because all relevant information, including the alarm time and date, is able to fit on the screen at the same time.

Initially, each channel's alarm time and date are set to midnight on January 1, 1900. This time and date will appear in the display for any channel that has not yet gone into alarm. When a channel goes into alarm its alarm time and date will immediately be set. That time and date will remain until the channel goes until alarm again and a new time and date is set. The alarm time and date do not clear simply by returning to a normal state.

Example:

```
DISP          ; Default to S1
DISP A        ; Display analog channels starting at 1
DISP V22      ; Put up display containing channel V22
DISP FIRST    ; Display first out alarm channel
```

DRAW Draw Graphic Line on Display (1)

Draw provides a simple line drawing facility that is used when in graphic mode. The beginning and ending positions of the line are provided as either pixel coordinates or percentage of screen.

Refer to the Graphic System Documentation for more information.

Examples:

```
Draw 10 20 100 200   ; from col 10, row 20 to col 100, row 200 in pixels
draw 0.1 0.2 0.8 0.9  ; from 10% X 20% Y  to 80% X 90% Y
```

Related:

DUMP Show I/O, RTU, Variable Data, etc. (1)

This command is used to get a configuration display on the I/O point system, individual RTU setups, specific task data, and more. The options are:

TASK x	Current or specified TASK (by number).
RTU x -	Current or specified RTU (by number).
LINK -	All comm links.
GROUP	All comm link groupings.
IO x -	All or specified I/O drivers and maps.
VAR -	Task variables used by expression evaluator.
COMM -	All comm ports.
CHAN x	Text line for a single channel or channel range.
AGA x -	Specified meter channel internal values.
PID x -	Specified PID Channel Internal Values.
FILE -	All open files.
ATTR -	CRT attributes, using example displays.
SCREEN	Sends copy of screen to printer.
FLAG -	Shows status of all globally defined flags.
RFLAG -	Shows status of all flags for the current RTU.
DB or DBASE	All open databases and current database information.

DUMP SCREEN is a command that allows for custom displays on the local CRT to be printed. To use this command, the SET PRINT ON command must have been entered at some time to allow local printing. The command will use the system's BIOS print screen driver, and will follow the screen image with a line telling the system name, rtu name, and time and date of the printout.

This command will be useful for a command file that clears the screen, paints a custom display using SAY and CURSOR commands, and then does the print screen.

Example:

```
DUMP TASK 3
DUMP RTU 2
DUMP CHAN C1:C8
DUMP IO
DUMP AGA M2
```

ECHO Send Text String to Comm Line (1).

When the program is reading a command file, it can use this command to send a line of text directly to the communications channel. In the case of the local console, this is useful for displaying status messages from within a command file. For remote terminals, this command can be used to send non-standard configuration information to the communications controller (modem or PRC). Many of the code conversion features of the ECHO command are also used in the expansion of lines being processed by other commands.

ECHO sends the line unaltered except for numeric values entered with the # key and special codes beginning with a dollar sign ($). Values following the # key are converted to a single ASCII character, normally a special control character. For example, #3 is a Control-C, #13 is a carriage return. These decimal values represent the numeric value of the ASCII code for these control codes. If you do not know what ASCII values are, you probably should not be using this command.

The special dollar parameters can be specified in an echo line by using the dollar ($) sign followed by a key letter. These special codes will be expanded at run-time to provide information within the text of the line. A literal string can be contained within double quotes which will be displayed without the quotes at runtime. Note that some codes are not available in SCADAWARE Lite. The special codes are:

$S	System name, set in the NAME line of the DAT file.
$L	Current link number.
$T	Current time of day.
$D	Current date.
$#T	Time of current database record.
$#D	Date of current database record.
$.T	Last update time for the current RTU.
$.D	Last update date for the current RTU.
$@	Last update time and date for the current RTU.
$@RTUID	Last update time and date for specified RTU.
$N	Text name of the current RTU.
$R	8 character ID of the current RTU.
$$	A single dollar sign.
$U	User's identification from password system.
$0-9	Command file command line parameters.
$%0-9	Task string variables.
$( )	Any valid TSP expression.
" "	Return string inside of quotes, with or without a $.

Unless the special accent mark (') is used at the end of the line, the ECHO command will send a carriage return and line feed character at the end of the text message. If the special character is used, the line is sent without the CR/LF.

Examples:

```
ECHO #7 TASK 1 Setup Complete
ECHO Alarm at $T on $D
ECHO Current link is $L
Echo WC45 updated at $@WC45  ST10 Updated at $@ST10  ; specified RTU name
```

Note: The #7 is the ASCII value for the BELL, or terminal beeper.

Related: HAYES, PRC, FORCE, MSG

EDIT Access Internal Text Editor (2)

SCADAWARE has a simple text editor which can be used to create and modify all of the text files used to configure and operate the system. A separate manual provides details on operation of the editor. This section provides only a brief overview of the feature and its access from within SCADAWARE.

This command can be used to edit configuration files, RTU command files, MENU files, or any other DOS text file. The editor can only be used on the local CRT console, not on remote displays. The editor has many features in addition to the usual insert and delete functions. Some of these features include find and replace, block moves, block copies, block erase, text printing, and word wrap. The control keys used to drive the editor are similar to the standard DOS EDIT, DBASE, and other DOS programs. An on-line help file (called RTUEDIT.HLP) can be displayed by pressing F1 while in the editor. This will cause a small screen to appear with hints on the various editor functions.

Entering EDIT by itself will cause a list of files to be displayed. The arrow keys (or mouse) must be used to select the file to edit. The list of files to choose from is determined by a file specification that is remembered by the text editor. When the program is started the default file specification is *.RTU. Whenever a file specification containing a star (*) character is entered along with the EDIT keyword, that file specification will become the default for any future edits.

For example, simply entering EDIT once the program is started will cause a list of all the RTU files to be displayed. *.RTU will remain the default file specification until changed. Entering EDIT *.LIB will cause *.LIB to become the default file specification and a list of all the LIB files to be displayed. However, entering EDIT MYFILE.LIB will not cause the default file specification to change because a wild card character (*) was not specified.

Entering EDIT followed by a complete file name will cause the specified file to be loaded into the editor. Entering a file name such as START*.RTU will cause all of the STARTx.RTU files to be displayed. Entering EDIT *.DAT will cause all of the DAT files to be displayed.

The editor provides a prompt for a file name when doing a block read or block write. This can be used to read existing RTU files into a single LIB file by doing control-K-R to request a block read. Simply specify each existing RTU file to read them into the single LIB file.

While in the editor, some of the normal function key functions are not available because they are used by the editor. Some of the keys, like F1 silence, can be accessed by pressing the ALT key when pressing the F1 key. So, ALT-F1 in the editor is the same as F1 outside of the editor.

The key assignments of the editor are similar to those of DOS BASIC and Edit, Dbase, Borland Products, and many other traditional DOS programs.

PRIMARY	ALTERNATE	FUNCTION
F1		Request help.
F2	^KS	Save and continue editing.
F7	^KB	Mark start of block.
F8	^KK	Mark end of block.
	^KD	Save and switch files.
ESC	^KQ	Quit without saving.
INS	^V	Turn insert mode off and on.
DEL	^G	Delete to right of cursor.
BSP		Delete to left of cursor.
ARROWS	^SEDX	Move up, right, down, left.
PGUP	^R	Move back a page.
PGDN	^C	Move down a page.
HOME	^QR	Top of file.
END	^QC	End of file.

AUTOMATIC FILE BACKUP

A SET command option is available to determine if the editor creates a backup file every time a file is saved. On systems with limited disk space, the accumulation of backup files causes a storage space problem. Using SET BACKUP OFF will cause the editor to skip the generation of backup files. The default is OFF. See the separate editor manual for a more detailed description of the editor.

ELSE Conditional Statement Processing (1)

The IF statement will allow sections of a command file to be skipped if the associated expression evaluates to 0. The ELSE command allows a section of code to be processed when the IF's expression does equal 0. See the description of the IF command for more information.

Example:

```
IF O1 = 0
 msg Output 1 is not on
else
 msg OUTPUT 1 IS ON NOW
endif
```

Related: IF, ENDIF

ENABLE Re-enable Channels Affected by Disable Command (2)

Channels that have been disabled (see DISAble above) are activated with the ENABLE command. This command can accept a range of channels, so a number of similar type channels can be enabled with a single command. Multiple channel ranges can also be entered in a single command line.

When channels are enabled, they regain all of the attributes they had before being disabled except for their alarm status. All points being enabled are set to the NO ALARM state, so that their current state will cause the alarm mode to be accurately set at the time they are enabled.

Examples:

```
ENABLE s3:s6
ENABLE S1:s4 a4:a6 o3:02
```

Related: DISABLE

ENDIF End of Conditional Statement Processing (1)

Every IF statement must have a matching ENDIF statement to mark the end of the conditional statements. See the IF and ELSE commands for further information.

Example:

```
IF V1 > 10
  msg Value 1 is greater than 10
ENDIF
```

Related: IF, ELSE

ENTRY Screen Oriented Data Entry and Display (1)

This command is used to set up an entry screen and allow the operator to edit existing values. Channel values, database variables, local variables, public variables, and string variables may be edited using this command. When using an entry screen to edit variables the user can move from field to field making changes and then either save all variables or abort all changes with a single keystroke. ENTRY is a more powerful way to edit system variables than can be done with the simpler INPUT command.

The ENTRY command must be processed from within a command file and can only be processed by task 0. Therefore, to setup an entry screen the user must create a command file which consists of several ENTRY commands. The ENTRY commands will define the entry screen title, the variables to edit, the prompts to display for each variable, and the screen position for each field. After all fields are defined the command CONFIG ENTRY is used to actually display the entry screen and allow the variables to be edited.

Of all the variables that can be edited, only the string variables (denoted by the preceding % sign) will accept text as input. All other variables are defined as real numbers and will not accept anything other than digits and decimal points. The ENTRY setup allows for range checks to be put on numeric entries so that only values within the range will be accepted.

The following is a list of keywords and parameters that can be used with the ENTRY command to define an entry screen:

```
TITLE TitleString
ADD PrCol PrRow PrStr Var [FldCol FldRow LoVal HiVal DecPlaces FldWidth]
TEXT Col Row TextString
CLEAR
```

The keyword ADD is used to add a variable to edit to an entry screen. The parameters that must be specified when using this keyword are the column and row of where the prompt will be positioned, the prompt string itself, and the variable to edit. Other parameters can also be specified when using this keyword. These optional parameters are the ones listed inside the brackets. Default values will be used if these parameters are not specified in the command. The default location for an entry field is right behind the prompt on the same row. No range checking is performed when editing real type variables if a high and low limit are not specified in the command. Also, a default of 2 decimal places and a field width of 8 are used for the format of real type variables if not otherwise specified.

For most applications the default field width of 8 will work fine. This is because there are usually only a few values that are put on one screen and there is plenty of room to put the variables being edited. However, on screens where many variables must be edited the default width can make the screen overly crowded. Space on such screens is sometimes wasted on numeric value fields that contain only 2 or 3 digits. In these situations, the total field width of the formatted number can be specified by the user in the ENTRY command. It must be at least two more than the value specified for the number of decimal places.

The keyword TEXT is used to position a text string on an entry screen without adding a variable to edit. The text string can contain the special $ codes contained in the ECHO command. The keyword CLEAR is used to deallocate all memory used by the current entry screen. The CLEAR option should never have to be used because all memory is automatically deallocated upon termination of the data entry screen, but its use is recommended to be compatible with planned extensions to the ENTRY command . The only time it might be necessary to use the CLEAR option is if fields were added to an entry screen but the entry screen was never displayed by using the CONFIG ENTRY command.

A maximum of 64 variables can be added to any one entry screen for editing. There is no limit to the number of text lines that can be added to an entry screen.

```
: sample Entry screen file
entry clear ; clear any existing entry setups
Entry, title, SAMPLE ENTRY SCREEN
entry add 5,6, Gas Gravity,V5,20,6,0.50,0.80,3
entry add 5,8,Percent N2,V6,20,8,0,10,3
entry text,20,21,Enter the New Values
config entry ; process the waiting screen screen
```

This small file will put up two fields with prompts and put the local user into the entry mode. Although only 2 entries are in the example, actual programs can use up to 64 different fields on a single "logical" screen. If the row and column positions are off the physical screen, the ENTRY processor will automatically scroll the screen to make the fields visible on the screen when the cursor moves past the screen boundary.

Notice how the numeric entries have the optional range specified on their setup line. This prevents the user from entering bogus values because ENTRY will not accept values beyond the specified range.

Related: INPUT, CONFIG

EXEC Execute a DOS Command, Including DOS itself (2)

This command allows access to the DOS operating system either directly (using the standard command.com) or by running the specified program. The RTU program is capable of executing a DOS program as a subtask. The only limitation is the available DOS memory for the Shell or Exec operation. A command line option (/E=) is available to reserve memory for an Exec operation. This must be specified when SCADAWARE is started as run-time adjustment of the program's memory is not possible.

EXEC is handy for "shelling out" to DOS to run small programs such as PCCLOCK or ATCLOCK or any other well behaved small DOS program. The problems arise when the programs do things directly to the computer's hardware (sort of like the RTU program itself does!). Many programs are simply not designed to run with other hardware intensive programs, and many will hang the system.

This command requires a certain amount of available memory to be able to execute successfully. The more channels a system has the more memory it uses when running the SCADA program. This means that there is less memory available for doing other things such as shelling out. If there is not enough memory available to process this command successfully, a message will be displayed and the command will be canceled.

Another complication is the use of the serial port to operate DOS. Although this is acceptable according to DOS technical information, in practice there are many restrictions. When using COM ports, special handling of input and output data must be provided to allow programs that normally access the local keyboard and screen to instead access the remote terminal. DOS itself is not too bad, and requires the addition of redirection codes on the command line (like < COM1 > COM1). These codes are added automatically when this command is used from a serial port task. Programs not using DOS for console I/O will probably not operate properly because their output will end up on the local CRT rather than the user's remote terminal.

This may be tolerable as long as the program does not require any operator input and output. For example, PCCLOCK and ATCLOCK programs used to set the internal timekeeper do not require any operator input, and can be run fine. Other programs are not as easy.

Another problem is that the RTU TASK dispatcher (the program that allows multi-tasking) does not want to swap tasks that are currently running a DOS operation. Shelling out to DOS or a DOS program will stop all background tasks that operate as normally scheduled tasks. Tasks that operate directly on each system tick (like the counter channel processor) will continue to operate during the DOS operation, but all others will be suspended.

Experiment fully with all programs before running them on a remote unit to avoid system lockups.

Examples:

```
EXEC PCCLOCK/S
EXEC COMMAND.COM
```

Related: SHELL

FAX FAX Report Generation and Control (1)

The full version of SCADAWARE can generate and transmit FAX reports over phone lines using a standard PC FAX Modem. Use of this feature requires proper setup and configuration of the modem as one-time setup with the OPTION form of the FAX command. Other forms of the command are used to generate and send the report in a multi-step sequence consisting of:

1. Produce standard text form of report using SCADAWARE features.
2. Generate FAX image from text file using FAX MAKE command
3. Use FAX SEND command to send fax image

The FAX OPTION statement is used to provide setup information needed by the fax processor during transmission. This information can be specified during startup or at any other time, but it must be done at least once prior to sending of a fax. All options are preceded by a slash /, or dash -, and followed by a parameter. More than one can appear on a single line, although this is not encouraged for clarity reasons. All of the following would follow the command sequence FAX OPTION

```
/I   Station identifier ; provide transmitting fax ID which appears on top each page
/N Number      ; provide complete phone number to send fax to
/C n           ; Com Port number (1-4)
/Q n           ; interrupt (IRQ) number if non standard.
/B nnn         ; modem baud rate, usually 9600 (this is not fax data rate)
/F nnn         ; Fax data rate (usually 9600 or 14400)
/M xxxxxxx     ; Modem Init String unique to each fax modem situation
/T Title String      ; Title used on top each page
/H             ; enable high-level fax functions if available
/S 1|2|A|C     ; set fax class to 1, 2, Auto, or Cas. Default is 2
```

Examples for typical options are as follows:

```
FAX Option -C 3 -Q 9     ; set port to 3, IRQ to 9      Shows multiple options on 1 line
FAX option /F 14400      ; fax data rate
FAX, option, Number, 1-800-555-1212  ; set fax dialout number. Note commas for delims
Fax  option /S 2         ; fax modem interface class 2
```

The FAX command has several other sub-commands in addition to OPTION. They are:

FAX MAKE fname	Create a fax image file from ASCII file FNAME. No file type is assumed for the input file, but the output will be the same name with a type of APF
FAX SEND fname	Send the previously created fax image file FNAME with assumed file type of APF.
FAX DUMP	Provide display information of the current fax setup.
FAX SHOW	Show real-time status of Fax sub-system.

The following simple TSP procedure illustrates the steps necessary to send a typical fax report.

```
; Demo to make and send a fax report
msg Generating Text Report
set eject off  ; no form feed at the end of a report
set delim space  ; put spaces between writeln outputs
sele kissa           ; make KISSA current RTU
report to faxtest.txt  ; write to a ASCII text new file

sele kissb           ; move on to the next RTU
report append faxtest.txt  ; add on to existing file created above

file append faxtest.txt   ; open up file for additional message
```

```
writeln                          ; blank line to the file
writeln "Thats All Folks!!"
file close

msg SCADA Report Generated.  Adding text comments to end
file append faxtest.txt
writeln
writeln
writeln, This is a text comment added to the end of the report
writeln, at $T on $D.
writeln
writeln, The value of Analog Channel 1 is,a1'####.##
writeln, Text Text Description of S1 is , s1'z
writeln
writeln, That is all.
file close

fax make faxtest.txt        ; convert text file to fax image format
msg Fax Created.
fax option, /i, ARTZ COMPUTER, /C,2, /T,SCADAWARE FAX REPORT,
msg Sending Fax now to 9,555-1212
fax option, /n, 9^340-7030  ; set number in modem. Note use of ^ in place of comma
fax dump      ; tell us what's been setup
fax send faxtest.apf    ; send the fax format file using fax modem
fax show  ; display realtime status of fax progress
```

NOTE: During conversion from text to fax format, the program uses a default font style that is contained in a separate file titled APFAX.FNT. This file must be present in the default SCADAWARE directory in order for the conversion to be successful.

FILE DOS File Output and Management Functions (2)

The RTU program normally sends all responses to the local CRT or serial port operating the task. The program can also send output to disk files (or logical devices that look like files, i.e. PRN) by using this command to set up and control the file. The available operations are:

OPEN	Create a new file, erasing any old ones with same name.
APPEND	Create or add on to existing file.
CLOSE	Stop sending output to the file.
DELETE	Delete a file from the disk device.
S+ and S-	Set file more to READ Only or READ/WRITE

FILE is a very powerful command, especially when used within a TSP command file. For example, the following script can be used to send a complete data log to a disk file for later processing:

```
FILE OPEN DATA.LOG  ; open a file called data.log for output
SCAN s1:s18 E     ; Generate DATA commands for current
SCAN a1:a8 E      ; status of the system
SCAN q1:Q4
FILE CLOSE       ; Stop output to the disk file
```

Use of the PRN: device as the file name allows for messages or other information to be directed to the local printer if desired.

Examples:

```
FILE OPEN ERROR.LOG
FILE APPEND OLDFILE.LOG
FILE DELETE BADFILE.LOG
```

FLAG Manipulate System Wide Control Bit Flag (2)

This program contains a predefined array of 512 flags that can be accessed by any task. A flag is very similar to a public variable with the exception that a flag can only be set to 0 or 1. The last 16 flags are reserved for system use and are altered with the SET command instead of the FLAG command. The reserved system flags are:

FLAG	USE
506	Ticking Enabled
507	Warbling Now
508	Warble Enabled
509	Edit Backup
510	Page Eject
511	Alarm Delays
512	Reset Files Activated

System flags are set off and on with the FLAG command. To set a flag, specify the number of the flag (1 - 496) followed by either a 0, a 1, or a keyword. Allowable keywords are:

TRUE	value of 1
FALSE	value of 0
ON	value of 1
OFF	value of 0
FLIP	invert current value

Multiple flags and flag ranges can be set using a single FLAG command. This can be done either by listing each flag and its value, or by specifying a range of flags followed by a single value. A range of flags can be specified by using either the ".." or ":" notation. All flags can be cleared by using the keyword RESET. The DUMP FLAG command can be used to display the status of all flags. The status of a flag, including the special reserved flag bits, can be checked from within a command file by using the @FLAG(x) function. A special test for @FLAG(0) will return true if any flag is currently turned on.

Note that a separate set of flags is available for each RTU through the use of the similar RFLAG command. Refer to the System Design Concepts manual for more information on FLAGs and their use in SCADAWARE applications.

Examples:

```
FLAG 1 0              : set flag 1 to 0
FLAG 6 TRUE           : set flag 6 to 1
FLAG 1 ON 2 OFF       : set flags 1 and 2 to 1
FLAG 1..10 ON : set flags 1 through 10 to 1
FLAG 40 FLIP  : change value of flag 40
FLAG RESET            : clear all flags (set to 0)
```

Related: RFLAG, PUBLIC

FONT Graphic Font Control (1)

The FONT command provides complex control over the graphic font size and style used in graphic based text output procedures. The fonts are contained in the system video ROM and also in a disk based file called RTUFONT.LIB. SCADAWARE has a series of predefined text fonts as follows:

FONT NUM	FONT NAME	PIXEL SIZE	FONT Description
0	SYS	?	Default System Font
1	Small	8x8	Small ROM Font
2	Medium	8x14	Med ROM Font (EGA standard)
3	Big	8x16	Largest ROM Font (VGA Standard)
4	Sanserif	1-20	Scalable Library Font. Size 3 is aprox 24 pixels
5	Roman	1-20	Scalable Library Font. Size 3 is aprox 24 pixels
6	LED	24	LED Style Font
9	Vertical	8	Rotated Small font

The fonts with fixed sizes will always appear the same size on the screen. The scalable fonts (also called stroked fonts) are generated at run time to the size specified in the most recent FONT SIZE command. A size of 3 produces a size of approximately 24 pixels, or 3 times the normal small character height on the screen.

Options for the FONT command include Style selection, color selection, alignment, and size as follows:

```
FONT STYLE   font_name     ; specify the font to use in subsequent output
FONT SIZE    multiplier    ; set scale factor for scalable fonts
FONT Alignment position    ; right, left, center alignment from specified X-Y position
```

Examples:

```
Font type Roman size 6  ; double sized from normal
font color blue alignment center
cursor 0.50, 0.50, This is Centered on the Screen
```

Related: Cursor

FORCE Send Command Line to Another Task (2)

RTU tasks can receive TSP messages from several sources. One source is inter-task messages. This is an internal mechanism that allows one RTU task (and several built-in functions) to send messages into the message queue of another task. With this capability, RTU tasks can control the actions of another task. For example, it may be necessary for the local CRT task to send a message to another task telling it to poll an RTU. The TSP command line is generated by the first task, sent to the second task, and then processed when the message is finally read from the message queue. Note that sending the message does not always cause an immediate reaction from the receiver because it may already be busy processing a previous message or command file.

Forcing a line to another task will place the line in a special queue that is checked before any other type of input is processed. However, if a task is currently processing a command file, the queue will not be checked until the file is finished being processed. Therefore, forcing a line to a task that is running will cause that task to process the command as soon as possible. This command is most useful to send commands to serial port type tasks. The most common use of this command is to start procedures from the local task and have them processed by a communications task that actually drives a serial port.

Another use of this command is to send configuration lines (i.e. SET commands) directly to tasks from the keyboard. Note that the task must be currently running in order to process the line. If the task is stuck or otherwise occupied, it can be restarted via the TASK START command. However, this has the side effect of erasing any waiting messages for the task.

The first parameter on the line is either a special keyword, or a task number (or name). The special keywords allowed are CLEAR and DUMP. CLEAR will erase all messages from the input queue of the specified task. DUMP will provide a listing of all messages waiting for all tasks.

Example:

```
FORCE 1 SET CRT ANSI
FORCE 2 PHONE 555-1234
FORCE clear 0       : kill messages for task 0
FORCE Dump
```

Related: TASK

FORM Real-Time Screen Form Display (1)

SCADAWARE has an easy to use screen form function to simplify the design of custom display screens and user menus. The form operates as a combined display and menu hit processor where realtime data can be presented while menu selections are made with keystrokes or a mouse. This is a completely new feature and is subject to revision at this time. Future enhancements will allow for printing of the forms, but this capability is not present at this time other than with PRINT SCREEN functions.

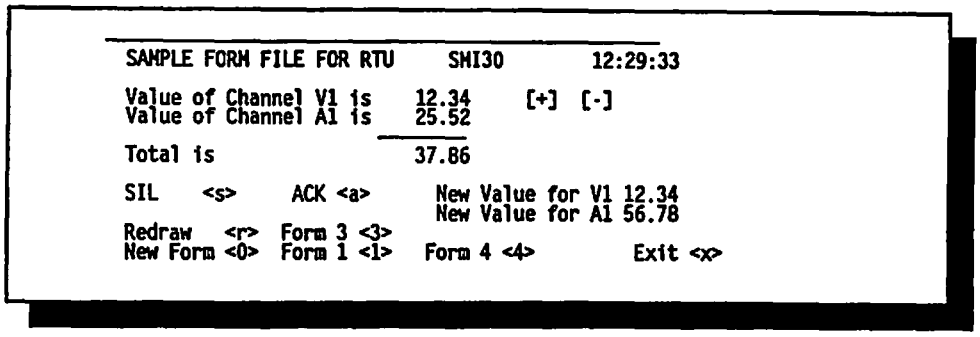
The form is created with the text editor and should have a file type of FRM. The main menu has been modified to allow access to FORM display and editing from the position formally occupied by the USER selection.

The form text file consists of plain text that will form a backdrop for the screen display. Special identifiers are placed on the form to indicate where data values and menu selections will appear. After the form is defined, additional information is provided in the form file to tell the system what is supposed to appear in the data positions, and what is to happen when menu selections are made. Here is a simple form file:

```
SAMPLE FORM FILE FIR RTU     [010]              [011]

Value of Channel V1 is    [012]     [+] [-]
Value of Channel A1 is    [013]

Total is                  [014]


SIL    [s]     ACK [a]       New Value for V1 [1]
                             New Value for A1 [2]
Redraw   [r]  Form 3 [3]
New Form [0]  Form 1 [4]   Form 4 [5]          Exit [x]
?end
?Color blue/gray
010  $R    : insert RTU name automatically
011  $T    : update time on display
012  v1'###.##    : show V1 to 2 decimal places
013  a1'###.##
014  $(V1+a1)'###.##  : total of 2 channels

a    ACK
s    horn off
X    ? Msg Form Exited Normally
r    ?paint

+    calc v1 = v1 + 1
-    calc v1 = v1 - 1
1    cursor 1,24 : input New V1,v1 : cursor 1,24 : clreol
2    cursor 1,24 : input New A1,A1 : cursor 1,24 : clreol
3    ?GOSUB FORM3    : gosub to a new form called FORM3.FRM
4    ?RETURN  : return to previous form
5    ?FORM5   ; branch to form 5 at same nest level
0      ? Form NEWFORM ;quit this form system and start over
;---------------  end of FORM file -------------
```

This little text file does a lot of processing once loaded by the FORM statement. The layout of all the prompts and other text will appear just as it looks in the template above. No X-Y coordinates must be calculated by the programmer. The form designer need only write the text and locate the variable locations with the bracketed numbers, as in [020]. The bracket notation is simple. Single

letters, characters, and digits are place holders for menu hits. Numbers 10 and above are realtime variable locations. The above file would generate a real-time Form display as follows:



The amount of information and menu items available for each screen vary between SCADAWARE and SCADAWARE Lite.

	LITE	SCADAWARE
Screen Value Elements	60	200
Screen Menu Items	30	200
Text Lines	30	100
Nested Screens	4	8

So, each screen uses single letters or numbers (0-9, a-z, +,-,=, etc) for menu selections, and a multi-digit numbers (010 and greater) for real time variables.

The space occupied by the real-time values will be determined by their actual size when formatted per system standards. If no format codes are provided, the default decimal places will be used. However, it is best to individually format each value so that no surprises occur when things change later. The form processor will right justify values so that they align on the right bracket. Space to the left is taken up as needed, so leave enough room between the text and data place holder.

Menu hits always take 3 character positions. A menu selection item such as [a] will appear on the screen as <A>, converted to upper case. Selecting the item can be done by pressing "A", or by mouse clicking on any of the 3 character positions in <A>. When the menu selection is made, the form processor immediately executes the statement associated with that menu hit. The form remains on the screen during execution.

The special control characters used within the form file all start with the question mark. Therefore, the question mark cannot be used as a menu hit identifier. The special control codes which can appear in the form file are:

?END	Signals the end of the on-screen portion of the file
?COLOR	identify new screen colors, similar to SET ATTRIB
?PAINT	Menu option to redraw the Form
?EXIT	Menu option to quit the form
?GOSUB	Nested call to another form
?RETURN	Nested return to calling form
?fname	Form name to transfer without nesting
?TSP	Process a TSP Command Line During Form Load
?SEL	Select logical RTU during Form Load

Menu selections with the ? mark are processed differently than ones without the mark. A ? followed by a space tells the form processor that the user wants to exit the form and process the remainder of the menu definition from the normal command prompt. This allows a way to menu-hit out of a form and do something else, including load another form. Use the ?-space combo to leave the form and return to the normal TSP command prompt.

The ?PAINT statement in a menu hit tells the form processor to redraw the current form. This

may be necessary if text has somehow gotten onto the screen, perhaps as a result of an internally processed command.

The ?GOSUB FNAME statement allows another form to be loaded as a subroutine. This is similar to the MENU GOSUB functions in the USER MENU system. If a ?GOSUB is done, the name of the current form is placed in a queue so that it can be reloaded when the subsequent form executes a ?RETURN form statement. This nesting of forms can occur up to 8 deep.

The ?FNAME form statement allows transfer to another form at the current nesting level. A ?RETURN executed in the subsequent form will go back to the original form just as if a transfer had not occurred.

Note that the ESCAPE key can always be used to exit the entire form system at any time. Any form nesting is lost when the user returns to the command prompt. It is not possible to execute another FORM command within a form itself. To start a new form system, the user must exit the current form and return to the command prompt or a USER menu.

FORMs can be started from the main menu with the revised USER/FORM selection. They can also be started from the command line or by any other means with the new TSP command FORM followed by the form file name. For example:

FORM MYSCREEN

will start a form called MYSCREEN.FRM. A file type other than "FRM" can be used, but this is not recommended. For Task 0, the simple TSP command FORM (without a form file name) can be used. This causes a directory menu to be presented of all files with the FRM filetype. An existing form file can be selected with key or menu.

GOSUB Subroutine Call (2)

The GOSUB command allows processing of command files in subroutine style program flow operations. It is very similar to the READ command. The READ command causes termination of the current file, and continues processing at the start of the named file. The GOSUB command holds processing of the current file and begins execution of the named file. When processing of the new file is complete, the program continues execution at the line following the GOSUB command. This command simplifies the use of standard routines by allowing them to be "called" from within other command files. Command files can be "nested" up to 16 deep.

If no file type is specified with the file name, the GOSUB command will search the procedure library for the file. If the file does not exist in the library, the GOSUB command will assume a file type of .RTU and then search on disk for the file. The default file type may be overridden by specifying a file type. If a file type is specified, the GOSUB command will skip the library search and look only on disk for the file.

Command line parameters can be passed with the GOSUB command just as they are done with the READ command. Parameters are only local to a single file. In order for a called program to get the parameters, they must be passed on the command line. Consider the following 2 programs started with the command line READ FILE1 PARAM1 PARAM2 PARAM3:

```
;-----------------------------------------------------
 ; Command file FILE1
MSG This is the start of program 1 $1 $2 $3
gosub file2 $3 $2 $1
msg Back to the main file
return
-----------------------------------------------------
; command file FILE2
msg This is from program 2 $1 $2 $3
return
-----------------------------------------------------
```

When FILE1 is started, it receives the three parameters that become local to that file. When

FILE1 calls FILE2, it passes three text parameters to FILE2 in the reverse order because they are defined in reverse order at the time of the call. Running FILE1 as described produces the following output:

```
This is the start of program 1 PARAM1 PARAM2 PARAM3
This is from program 2 PARAM3 PARAM2 PARAM1
Back to the main file
```

Note that the RETURN statement can be used to stop execution of the current command file. Execution also stops when the end of file is reached, so RETURN is only needed if termination is needed within the body of the file. A RETURN from the "topmost" file will return processing to the command line.

Examples:

```
GOSUB file1
GOSUB file2.xxx param1
```

Related: READ, RETURN

GOTO Branch to Program Label (1)

While processing command files, it is often desirable to have the processing continue at another section of the file. The GOTO allows for a jump similar to a DOS Batch File. The jump point is called a label, and is identified as any word, up to 32 characters in length, that begins with a colon character (:).

One side affect of the GOTO is that all pending IF statements are canceled. This allows for a branch to be made with a conditional IF without having to keep track of the nested IF level. Consider this small sample file:

```
; Program to test GOTOs
calc v1 = 0      ; initial value
cls          ; clear the screen
cursor 10 1 "GOTO TEST FILE"
cursor 10 10 "Value of V1="
:START           ; Loop Point
calc v1 = v1 + 1     ; increment v1
cursor 25 10 V1      ; display new value
sleep 1          ; let value display for 1 second
if (v1 < 10)         ; test v1 for over 10
 goto start        ; if not 10 yet, loop
endif
msg Program over
```

Labels must begin with a colon in order to be properly found by the GOTO search. In searching, the file is reset to the beginning without the normal DOS close and open operation (similar to REREAD command). This speeds file processing, especially on small files. The search starts at the beginning of the file, so labels that are used more than once in a single file will result in only the first one begin matched.

Example: GOTO start

Related: READ, REREAD

GROUP Callout Group Control (2)

A callout Group is a collection of communication links that are all activated when channels using the group require a callout. Each channel point has a callout group associated with it, and will use this group whenever a callout is required for either an abnormal or return-to-normal condition. The requirement to call is determined by settings in the point's alarm mode (explained elsewhere).

The GROUP command requires that the specific group being referenced be provided as the first parameter on the line. After the group number, any number of links can be referenced. If the link number is entered as a simple number (i.e. 3), then the link is added to the existing links that make up the group. If the link is entered as a minus number (i.e. -3), then the link is removed as a member of the group. This allows for links to be dynamically added or removed from a specific group under program control.

To completely clear out a group, the keyword RESET is used just after the group number.

If only the group number is entered, then the system will display the links associated with the group. This allows for a quick way to verify the current contents of the group.

Examples:

```
GROUP 3 RESET
GROUP 2 1 3 5 ; Make links 1,3, and 5 part of group 2
GROUP 3 -4    ; Remove link 4 as part of group 3
```

Related: LINK

HALT Terminate SCADAWARE and Return to DOS (2)

The HALT command is used to properly terminate the program. If the command is used without any parameters (or if QUIT PROG is selected from the main menu), the prompt "Really want to quit?" will be displayed. A "Y" response will cause task 0 to look for a library procedure or RTU file named HALT and execute it if found. Although this procedure or file does not have to be used, it does allow a more controlled program stop. Typical uses for the HALT procedure or file are:

1. Putting outputs in a particular state.
2. Making final saves to image files.
3. Communications device deactivation.
4. Saving special data to text files.
5. Controlled stop of all tasks.

It is probably better to make HALT.RTU a separate file rather than a procedure in the library. This will ensure its availability upon program termination regardless of the state of the library. A typical HALT.RTU file might look like this:

```
task 1 stop
task 2 stop
task 3 stop
task 4 stop
image save
```

The keyword HALT (in caps) can be used as a parameter in the HALT command to bypass execution of the HALT file or procedure. Use of the HALT HALT command will also cause the program to be terminated immediately without issuing the prompt to really quit.

Examples:

```
halt
halt HALT
```

Related: REBOOT

HANGUP Disconnect Communications Line (1)

The hangup command provides a convenient way to disconnect a radio or telephone communications link. This is slightly different from the BYE command which causes a change in the logical connect state of the current task. The BYE command also causes processing of the BYEx.RTU file to occur. The HANGUP command does none of this. It only processes the disconnect portion of the BYE operation. All other aspects of the task are left alone.

This command can be used in special command files that will do non-standard communications operations. A typical example is a command file to call an RTU several times to initiate a modem-monitor reset. This requires several sequential calls in rapid succession, which can be done with the DIAL, WAIT, and HANGUP commands.

Example: HANGUP

Related: DIAL, WAIT, BYE

HAYES Process Hayes Modem Command Line (1)

If the communications channel is handled by a HAYES (C) compatible modem, the HAYES command can be used to send configuration lines directly to the modem by placing it in local mode. This requires a combination of time delays and special characters (normally + + +), and the HAYES command takes care of doing this automatically. Using this line in a command file will allow sending setup information directly to the modem at startup or any other time.

This command skips the normal command line parsing so that commas can be used within the hayes string. This command uses everything after the word HAYES as the hayes command string, regardless of the presence of commas or spaces.

Typical HAYES commands are as follows:

H0 Hang up
E0 Echo Off
Q1 Quiet mode on
S0=x Register for number of rings before answer
S6=x Seconds to wait for a dial tone before aborting dialout
S7=x Seconds to wait for carrier detect on a callout
S8=x Seconds that each comma represents within a phone number
S9=x 1/10th seconds that a carrier must be present to be valid
S10=x 1/10th seconds that a carrier must be lost to abort
S11=x milliseconds for touch tone duration and spacing.

Within a phone number, several characters have special meaning:

comma Wait for approximately a second (see S8 above)
W Wait for a dial tone before proceeding
P Use pulse dialing rather than tone
T Use touch tone dialing rather than pulse
@ Wait for silence before continuing
! Flash (off-on hook toggle for 1/2 second)
- Ignored, but makes numbers easier to read

A phone number may look like PW9,T1,504-123-4567

This number will be processed as use pulse dialing, wait for a dial tone, dial a 9 (to get an outside line), wait a second, change to tone dialing, dial a 1 (long distance access), wait a second, then send out 5041234567. Note that the dashes in the phone number are permissible but ignored.

Because the comma character is used as both a Hayes pause code as well as the program's parameter separator, the program will translate any caret ^ characters into commas at the time the phone number is processed. This allows for pauses in numbers to be entered with the ^ character rather than the normal comma. For example, the number 9^555-1212 will be treated as if it were 9,555-1212 when the number is sent to the modem.

Examples:

```
HAYES S0=1 S1=7
HAYES E0 Q1
```

Related: PRC

HELP Display Help on RTU commands (1).

The RTU program is designed to operate on a wide variety of computer configurations, and the amount of on-line help available will depend on the amount of local storage available. In small remote systems, the help display may be limited to a list of commands. In larger hard-disk based systems, detailed help files may be available for every command. The contents of the help files is left to the user's discretion.

The HELP command uses a file name as a parameter. The file must have the assumed extension of .HLP. For example, the file DIR.HLP will contain help information on the DIR command.

If no command is specified as a parameter on the line, the HELP command will display a list of files with the extension HLP.

The HELP files are simple text files prepared with any editor or word processor that can produce plain ASCII text files. They are sent to the console un-altered, except for lines that begin with the slash character (/). If this character is the first on the line, the line will not be displayed and the command will pause until the operator hits the ENTER key or a time-out period expires. This allows for the file to be broken into logical pages for viewing.

Example:

```
HELP
HELP PROGRAM
HELP AGA3
```

Related: TYPE

HOLD Set Channel Holding Status (2)

Each channel has a Hold status that prevents any low-level value conversions from occurring. This is useful when calibrating a channel or when the channel must be manually set for some reason. A common use is to set an AGA3 meter to HOLD, and then calibrate its input channels. During the calibration, the meter value will remain constant. Any totalizer channels reading the meter channel will continue to add the constant value rather than an incorrect value resulting from the calibration process.

Examples:

```
HOLD M1
HOLD m1:m2
HOLD M1:M3 A2:A7 Q1:Q5
```

Related: CHANGE, UNHOLD

HORN Annunciator Horn Control (2)

The program uses the PC's internal speaker as an alarm horn or "warble." The program also has the built-in ability to manipulate any channel as an alarm horn. Whenever an un-acknowledged alarm is present, the system will set the specified horn channel to 1. When the ACK occurs, the channel is set to 0. Normally, a status input or output is specified as the horn channel, although any channel can be used. The channel is specified with the SET HORN nn command.

The HORN command is used to provide generic access to the horn channel without knowing exactly what it is. The only options are HORN ON and HORN OFF. The exact application is up to the user. The horn is the local PC speaker as well as any channel designated with the SET HORN command.

The SET WARBLE command is used to control the use of the local speaker as an alarm horn. Use SET WARBLE ON to enable the local speaker and SET WARBLE OFF to disable it. If WARBLE is turned off, the speaker will not sound even if the horn is turned on. Consider the following program stub that manipulates a horn output in two different ways:

```
; Set up output as a horn and turn it on
set Horn O8  : output 8 is now the horn control
calc O8 = ON  : directly turn on horn
calc o8 = 0  : turn off the physical output
HORN ON    : turn on horn output (and speaker) indirectly
HORN OFF    : turn off horn (and speaker)
```

NOTE: The RTU version of SCADAWARE is required to operate a local input/output channel.

Related: SET HORN, SET WARBLE

IF Conditional Processing Statement (1)

Logical "IF" processing can be used in a TSP command file to control program flow. The IF statement allows a TSP expression to be evaluated to either a 0 or non-0 (false or true) state. If the result is true, then the lines in the command file following the IF line are executed. If the result is 0, or false, then the lines following the IF line are skipped until an ENDIF line is processed.

The end of a command file is also treated as an ENDIF. Any pending IFs are cleared whenever a command file terminates. IFs can be nested up to 32 deep in any single command file. The program tracks ELSE and ENDIFS using conventional programming logic where each ELSE or ENDIF applies to the most recently processed IF statement.

The expressions used in IF statements are processed exactly as they would be in a CALC line except that no assignment to a target variable is allowed. Use parentheses to insure that the desired logical grouping is obtained. The IF and its associated evaluation can be tricky, so experiment with various combinations before releasing programs for general use.

File processing in the event of an evaluator error is controlled by the SET ERROR command.

Examples:

```
IF (V1 > 10) | (v2 > 10) ; OR test
  if v2 > 10
    msg Both V1 and V2 are greater than 10
  else
    msg V1 only is greater than 10
  endif
else
  msg Neither V1 or V2 is greater than 10
endif
```

Related: CALC, ENDIF, ELSE

IMAGE Automatically Save Values to Disk (1)

The program has an automatic, transparent data save system that continuously saves many dynamic values while the program is running. This feature creates an "image" of the data to be saved in the actual internal format of the program. The IMAGE command is used to save the image to disk, load the image from disk, or have the image automatically saved every so many seconds. The name of the file that contains the saved image defaults to the same as the system name and has a file type of ".IMG". Therefore, a system called VR167 will have an image file called VR167.IMG. The IMAGE FILE command can be used to specify an alternate disk drive and/or file name. The image command options are:

```
IMAGE DUMP  ; display image status and contents
IMAGE ; same as image dump
IMAGE LOAD  ; load variables from data image file
IMAGE SAVE  ; force an immediate image save
IMAGE ON [xx]          ; start automatic image saving
IMAGE OFF     ; stop automatic save and close image file
IMAGE FILE filename    ; override default image file name
IMAGE TASK xx          ; specify which task will save the image
IMAGE SELECT xx        ; specify the current image slot for a task
```

Loading of IMAGE data is usually only done during system startup. This allows a system to pick up where it left off before the program terminated because most of the dynamic conditions of the program are immediately restored to the values that existed the last time the IMAGE SAVE was processed. This presents a few questions at startup because the user may have special considerations for some values. For example, it may be desirable to have all outputs turned off whenever the system starts regardless of the output states when the program terminated. Simply loading the IMAGE file will restore the outputs as they were when the last image save occurred. To work around this, it is possible to coordinate the RTU configuration load, the image load, and the DRIVER TASK startup such that inadvertent conditions are avoided. Consider the following START0 procedure stub:

```
; Portion of a START0 process
image load
if @image(0) = 0
  msg Image Loaded Properly
else
  gosub VR123          ; load in the normal RTU configuration data
  gosub vr123.lin      ; link data
  image save           ; initial data save
endif

calc o1:o8 = off ; make sure all outputs are off
task driver start ; now let the I/O system run
image on 30     ; start saving every 30 seconds
task 2 start
task util start
task scan start
```

The IMAGE ON command is used to start saving the data image to disk every so many seconds. The default is 60 seconds. An additional parameter can be specified with this command to change the number of seconds between image saves. Note that there are differences in the image file formats of the Full and Lite Versions of SCADAWARE. The image files are not compatible with each other. For more information see Automatic Value Saves in the System Design Concepts Manual

Changing of the image file name can only be successful when the file is not already opened. Therefore, the IMAGE FILE command must be used prior to any IMAGE LOAD or IMAGE SAVE actions.

Examples:

```
IMAGE                 : display image status
IMAGE ON              : start automatic saving
IMAGE ON 10           : start saving automatically every 10 secs
IMAGE FILE B:MP36     : set image file name to MP36.IMG on drive B
IMAGE SAVE            : save the image right away
```

INPUT Enter Channel Value or String Data (1)

This command is used to prompt the operator and allow the entry of a new values for channels, user defined variables, and strings. INPUT uses the same line editor as the PROG command described below. Note that many applications previously served by the INPUT command can be better handled by the newer ENTRY screen system. However, INPUT is still desirable for any procedures that must operate over a remote console through a serial port.

The command requires a prompt, even if it is blank, followed by a channel or variable identifier. By default, the command will time out in about 10 minutes. The timeout period can be changed by specifying the number of seconds after the variable identifier. The minimum timeout value is 10 seconds.

The new value takes effect immediately after completion of the command.

Examples:

```
INPUT, Enter New Plate Size,V1  ; use default timeout
INPUT, Enter New Flow Rate,V5,300 : 300 second timeout
INPUT, Enter Report Title, %1
```

Related: PAUSE

KEY Temporary Control of Multi-Drop Modem (1)

KEY lets the local user control the multi-drop modem or comm link for test purposes. These devices use the RTS or DTR line to control the carrier or line driver which are automatically activated during normal communications. Testing requires that these control lines be manually driven, and the data line be given test data to transmit. All of these functions can be done with the KEY command and its various options. The basic syntax is as follows:

```
KEY port [RTS|DTR] [Period in Seconds]
```

The port is the number of the communications port, such as 1 or 2. RTS or DTR must be specified to tell the system which control line to use. The optional timing period is the number of seconds (defaults to 30) and can be any number. This determines how long the local task will take over the port for purposes of the test.

Once KEY takes control, the comm port's control line is activated and a steady 0 signal is transmitted. A timer starts which will time out if the KEY function isn't terminated earlier by the user. During the timing period, the following keys have meaning:

ESC	Quit the KEY procedure and restore normal port operation
Space	Toggle the output bit from 0 to 1 to 0.
K	Turn the output control (RTS or DTR) off and on.
T	Send a short test burst out the port.

Examples:

```
KEY 2 RTS  60 : Use port 2, pin RTS, for up to 60 seconds
KEY 1 DTR           ; Port 1 DTR for default timing period
```

LET Alternate to CALC Command (2).

The LET command can be used in place of the CALC command to cause execution of the expression evaluator. LET is provided to simulate BASIC programming commands. Refer to the CALC command for more information about this command.

Example: LET TOTAL = DAYTOTAL + WEEKTOTAL

Related: CALC

LIBrary Load Command File Library (1)

This command is used to load command files into a buffer in memory which acts as a Library of procedures. This command is also used to monitor the files in the Library. The Library system allows loading, merging, and listing of the procedures contained in the Library. For more information on how this works refer to the section called PROCEDURE LIBRARY in the system design documentation.

The LIBRARY command options are:

```
LIBRARY LOAD filename      ;load a file of procedures into memory
LIBRARY MERGE filename     ;load entire file into memory as a single procedure using the file name
                           (without the extension) for the name of the procedure.
LIBRARY CLEAR      ;release all procedures from memory
LIBRARY DUMP       ;display list of all procedures in memory
```

If filename is not specified in the LIBRARY LOAD command the default file name RTU.LIB is used. If no extension is given in the filename for the LIBRARY LOAD command the default extension used is '.LIB'. If no extension is given in the filename for the LIBRARY MERGE command the default extension used is '.RTU'.

Examples:

```
LIBRARY LOAD file1 - load in procedures from file1.lib
LIB MERGE x.rtu   - load file x.rtu and call it procedure x
LIB CLEAR      - delete all procedures from memory
LIB DUMP       - show what procedures are in memory
```

LINK Program and Control Comm links (2)

The LINK command allows for programming and control of the communications links to other units. Basically, LINKs are a combination of a task (and its comm port),a phone number, and some control parameters. The link, once programmed, becomes a means of reaching another unit by referencing the link number rather than a specific port and phone number.

The LINK command is very similar to the POLL command in that both are used to control communication links. The difference is that the link command is concerned with a specific link whereas the poll command is concerned with a specific RTU. When using the link command the specific link to reset, retry, or activate must be given. When using the poll command only an RTU name needs to be given to reset, retry, or activate a link. The computer will look up the default link for that RTU and use that link to perform the same functions that this link command does.

The following is a list of parameters that can be used with the LINK command:

```
link definition   <- program a link
RESET             <- reset currently active link to idle state
RESET xx          <- reset specified link to idle state
RETRY xx          <- set failed link to active state
NOW xx            <- activate a link
SAVE              <- save link and group data to default file
```

SAVE TO fname <- save link and group data to specific file
xx <- dump link info for specified link

To program a link using the link command the following parameters are required in the specified order:

1. Which Link is being referenced.
2. Which task will use this link on its comm port.
3. The phone number or radio call sign to use in dialing.
4. The maximum number of re-tries to make the connection.
5. The number of seconds to wait between callouts.
6. The number of seconds to wait for a connect when calling.
7. Yes or No for whether or not this is a host.
8. Link Description
9. Alternate Link Number
10. Number of tries to rapidly callout
11. Max callout tries without resetting counter to 0

Note that commas can be placed in a phone number (for Hayes command processing) by using a caret (^) character in place of the comma. The LINK command will replace the ^ with commas in the phone number field. For example, the line

```
LINK 3 2 9^1^555-1212 5 30 120  ; Link3 on comm 2
```

will produce a phone number that is actually 9,1,555-1212.

The keyword RESET is used to return a link to the idle state regardless of the current state of the link. If RESET is used with no other parameters, then the link to be reset is the link currently in use by the task processing the LINK RESET command. Issuing a LINK RESET can be used to reset the callout request at the sending unit without having to know which of the sender's links was in use. Issuing a LINK RESET for a task that is not currently processing a callout will do absolutely nothing. A likely place to use this command is at the end of a download file. As an alternative, a link number can be specified with the RESET parameter which will cause the specified link to be reset.

The keyword RETRY is used to reactivate a failed link. This would be a link that has exceeded its normal retry count. When in the failed state, subsequent attempts to activate the link are ignored until the link is reset (via a LINK RESET command or an F5 Key press). This option, LINK RETRY, will cause the specified link (or the current link if not specified) to be made active again. An example application would be to have an agenda command that periodically re-activates all failed links. The links may have failed because the communications equipment was inoperable, so a periodic RETRY command will kick off the link process. The LINK RETRY only affects links that have failed. Links that are idle or active will not be affected by this command.

The keyword NOW is used to activate a specific link. Nothing will be done if an attempt is made to activate a link that is currently processing a callout.

If only the link number is provided on the line, then the system will display a description of the link. This provides a quick way to check the current status of a link.

An optional keyword ALL is available to be used wherever a link number can appear. This will cause the command to apply to all links rather than just one link. For example, to activate all links you can enter LINK NOW ALL. This can replace the several lines previously needed to activate all of the links in a host system. It will most likely be used in agenda processing where all RTUs are polled at the same time each day. Instead of listing all of the RTUs on separate lines, a single line can be used. Any additional RTUs added to this HOST will automatically be processed by the LINK command when the ALL option is used.

The SAVE keyword causes the system's link and group status to be saved to a separate file named after RTU 0 with the file extension of ".LIN". For example, a system called VR167A will have its link setup stored in VR167A.LIN. It is possible to specify a different file name by using the TO option on the same line.

Examples:

```
LINK 3 2 555-1212 5 30 120 ; Link3 on comm 2
LINK RESET                 ; Reset current link
LINK 2 RESET               ; Reset link 2 only
LINK RESET ALL             ; Reset all links
LINK 3                     ; Tell us about link 3
LINK SAVE                  ; Save link data to default file
LINK SAVE TO ga694.lin     ; Save link data to specified file
```

Related: POLL, GROUP, PHONE

LOCAL Declare Local Variables (1)

This command is used to declare variables for a task that will be accessible only by the procedure that declares them or by subsequent procedures called with the GOSUB command. For example, if a command file declares a local variable then that file has complete access to that variable. Any command files called from that file with the GOSUB command will also have access to that variable. When the processing of the file that defined the variable terminates the variable is released. See the PUBLIC command for further information on the use of variables.

Example: LOCAL TOTAL1 TOTAL2

Related: RELEASE, PUBLIC, GOSUB

LOG Control and Display of Logged Data (2)

The SCADAWARE LOG SYSTEM stores alarms, data values, and text messages in DOS text files for later retrieval. This system receives input from all alarm points that are programmed to log abnormals. Points with this attribute will generate an alarm message containing the point number, value, time, and date of the abnormal condition. Points that do not have this attribute set will not automatically generate logs. Note that data points do not have to be alarm type channels in order for abnormal conditions to be logged.

Return-to-normal conditions can also be logged if the SET RLOG command is issued. This global setting affects all logged points for all RTUs.

Channels can also be "logged" via the log command so that their current values will be sent to the log system for later review. This is similar to the action caused by an abnormal condition except that it can be done at any time regardless of the state of the channel. This is useful for totalizer type channels and other channels that must have their values periodically recorded.

Logging of messages is done to store text information with time and date stamps. A few events cause logging to occur automatically, but messages can also be stored in command files by using the LOG MSG command. The text string that is saved is everything after the MSG keyword. The message is expanded using the same rules as the ECHO command, so the $T, $D, and other special operators can be used within messages. This is normally not necessary because the log system adds time and date automatically when the message is logged.

Logging is done internally by the program in two separate steps. The first is to send the data to a memory queue that holds the messages until they are sent to the disk. The second step is transferring the data from the memory queue to the disk file. The timing on these actions is adjustable to match the performance of the logging system to the hardware. Messages in memory will be lost if the system is reset or loses power. Messages in the log file will be more permanently stored in a standard DOS disk file, with a separate file for each RTU.

Each logical RTU set up on a single CPU has its own log file, which is the name of the RTU plus the ".LOG" extension. Log lists only apply to the currently selected RTU.

The logging options are as follows:

```
LOG                          <- Display contents of the log file and memory log
LOG FILE [filename]          <- Set or display default log file
LOG EVERY [xx]               <- Set or display logging interval
LOG CLEAR                    <- Erase disk and memory log
LOG RESET                    <- Same as LOG CLEAR
LOG MSG any message <- Log any text message
LOG q1                       <- Log single channel's current value
LOG a1:a8                    <- Log a range of channel's cur value
LOG TASK [xx]                <- Set or display task responsible for logging
LOG FIRST [rtu]              <- Log first out alarm for current or specified RTU
LOG SAVE [filename]          <- Save log data to disk file for the current RTU
LOG SAVE ALL                 <- Save log data to disk files for all RTUs
LOG FROM logfile TO outfile FOR range  <- display channels from a log file
```

The LOG FOR option is used to display a range of channels from the log file. The FROM, TO, and FOR parts are optional and may be listed in any order.

Note that setting the logging interval does not cause any data to be logged. This only controls how often the contents of the memory log are transferred to the log file. If the logging interval is set to 0, the memory log is never sent to disk.

If a disk error occurs during a log save, the interval is automatically set to 0 to prevent further attempts. Additionally, a command to read a file called LOGERROR.RTU is sent to the utility task. This file can cause any sequence of events to occur, including resetting the log interval. The task which processes the LOGERROR command can be changed with the LOG TASK option.

See the SCADAWARE System Design Concepts manual for more information.

Examples:

```
LOG every 30   <- save any data every 30 seconds
LOG file myfile <- set new default log file name
LOG msg Task 1 communications.started
LOG a1:a2
LOG from OLDDATA.LOG <- look at an old log file
```

LOGON Access Password Security (0)

In systems using passwords, the LOGON command is used to gain access by entering a valid password. If the keyword LOGON is entered by itself, the user will be prompted for an access code. An optional access code can be entered directly after the LOGON keyword to eliminate the prompt. This can also be used in command files to change users.

If the access code is found in the current Password file, then the user is allowed access to the system at the security level assigned at the time the user's code was entered into the computer.

Examples:

```
LOGON Bingo
LOGON
```

Related: PASSWORD

MAP Open Architecture Data Channel Mapping Control

SCADAWARE uses a data mapping feature to allow transport of most TSP data values and alarm settings over alternate protocols such as Modbus. The mapping facility allows numeric data values, alarm conditions, setpoints, and other TSP data to be translated into the other protocol using binary bits, words, and IEEE floating point values. The exact syntax of the mapping depends on the protocol being emulated (Modbus, Fisher-ROC, Allen Bradley, etc).

Although the concepts among protocols are similar, the exact details will depend on the implementation being performed by SCADAWARE to emulate or enhance the alternate protocol. SCADAWARE can emulate multiple protocols on the same communications channel, allowing it to coordinate communications among diverse units. The various maps are setup and selected using the MAP command and its various options.

Refer to the TSP protocol conversion manual for each system for more information.

MENU Control Main and User Menus (1)

There are two menu systems available to the local user that are controlled by this command. These menus are referred to as the MAIN menu and the USER menu. The Main menu is a PC type "pull down" menu that is used to perform many of the typical SCADA program operations. The Main menu is a standard menu that is the same at each location.

The User menu is a "pop-up" type menu system that allows for quick selection of pre-programmed actions. The User menu can be modified to perform custom functions unique to each location. This is done with the use of menu files. See the section on FUNCTION KEYS and MENUs in the System Design documentation for more information on the menu system.

When the MENU command is entered by itself, the Main menu is displayed. This is the only case where the menu command refers to the Main menu. When the menu command is entered followed by a keyword, the command refers to the User menu. The User menu is displayed by using the USER command. A list of available keywords used with the MENU command is as follows:

LOAD [menufile]	load a new user menu file into memory to replace the current one. The LOAD option allows an optional file name to be specified. This can be any legal DOS file name, although one with a ".MNU" extension is recommended. If a filename is not provided, it defaults to RTU.MNU.
READ [menufile]	transfer to a new menu without nesting
GOSUB [menufile]	nested transfer to a new menu
RETURN	go back to previous menu
DUMP	display information about the current menu file

Menu files can be contained within the procedure Library just like RTU command files. The menus are identified with the PROC keyword followed by the name of the menu. The PROC keyword is used even though these files are not actually procedures. The keyword simply serves to identify and separate each section of the Library file.

When using menus in the Library file, it is important to not use a .MNU extension when referencing the file. Just as is done with RTU command files, the file processor will skip the Library if the referenced file has a specified file type. So, entering MENU LOAD MYMENU.MNU will force the program to look directly at the current disk, bypassing the Library. However, entering MENU LOAD MYMENU will allow the program to first check the Library. If not found there, the program will append the .MNU file type to the name and look on the disk. For more information on using the Library see PROCEDURE LIBRARY in the System Design Concepts documentation.

Examples:

```
MENU          ; display main menu
MENU LOAD SHUTIN.MNU ; load the shutin menu file
MENU READ HURRIC   ; make hurric current menu file
MENU GOSUB EXAMPLE  ; nested transfer to example file
MENU RETURN       ; go back to previous user menu
MENU DUMP
```

Related: USER, SET MENU ON

MONITOR Watchdog Timer Monitor Control (2)

The monitor function of the program toggles the RTS or DTR pin on the serial port once per second as long as all is well within the system. Various functions are monitored by the task manager which will cease the toggle if an error is detected. The serial port pin can be connected to a timeout monitor (watchdog timer) to detect system failures and reset the system. Systems using this feature must have modems that do not rely on the state of the line being toggled. If necessary, remove the pin (usually pin 4 or 20) of the DB25 connector at the modem to prevent any problems.

In all configurations, the monitor function operates automatically on COM1. If this causes a problem with certain equipment, then commands should be included in the STARTO file to change the port or the status of the monitor signal. Multiple parameters can be entered on one command line.

```
MONITOR ON          ; Enable the monitor output
MONITOR OFF         ; Stop the monitor output
MONITOR 1           ; Use com1 RTS line
MONITOR 2           ; Use com2 RTS line
MONITOR HI          ; Set RS-232 signal HI
MONITOR LO          ; set RS-232 signal LO
MONITOR RTS         ; Use Pin 4 RTS
MONITOR DTR         ; Use pin 20 DTR
MONITOR FLIP        ; Invert the control line
```

Examples:

```
Monitor ON
Monitor ON 1 DTR
```

MOUSE Microsoft Compatible Mouse Control (1)

The mouse command provides control over basic and advanced functions of a Microsoft compatible mouse installed on a SCADAWARE computer. The actual device may be a touch screen or trackball, but the interface is the same and can use this command. Options provide control over the *graphic* mouse shape, X-Y coordinate display, and low level software reset.

```
MOUSE  OFF|ON       ; overall mouse enable
MOUSE ACK off|on    ; Automatic dialog box fo confirm graphic mouse click
MOUSE BLIP OFF|ON   ; Audible blip on mouse clicks
MOUSE INIT          ; Low level mouse initialization
MOUSE XY OFF|ON     ; control display of graphic X-Y coordinate update
MOUSE STYLE style_name     ; Sets mouse shape to one of the following:
                ARROW      Standard Arrow
                CHECK      Check mark
                PLUS       Plus or Cross symbol
                BOX        Box shape
                RTHAND     Right Hand Symbol
                UPHAND     Up pointing Hand
                STOP       Stop Sign
                QUEST      Question Mark
                HOUR       Hour Glass
```

MOUSE COLOR color ; set mouse fill color

Note that no options are valid for the text mode mouse other than the RESET function.

Examples:

```
Mouse XY ON
Mouse Style Uphand
Mouse color Blue
Mouse style arrow
Mouse reset
```

MSG Display Message on Local Console (1)

During command file execution, it is often desirable to display messages on the local computer's CRT to indicate progress in execution. The MSG command acts just like a FORCE 0 ECHO command, except that MSG lines processed by the local CRT task are displayed immediately rather than being placed as ECHO commands in task 0's input message buffer.

Typical applications would be to indicate connections, logon procedures, or any other action that should be noted on the local CRT.

Example: MSG Task 1 Connected at $T

Related: FORCE, ECHO

PAINT Transparent Graphic Image Display (1)

PAINT is used to place scalable graphic objects contained in disk based GIF and PCX files onto the graphic display screen. Objects (images) can be scaled up and down, and can be placed at any pixel coordinate or percentage of screen X-Y. Refer to the Graphics System Documentation for more information on PAINT.

Related: Backdrop

PASSWORD Access Password Setups (3)

The Password system provides a means of controlling access to the system by requiring security codes to be entered by each user. The PASSWORD command is used to display and edit the list of eligible users. Each user has a password and a security level assigned with this command.

Only level 3 (supervisor) personnel can access this command and make revisions to the current password setup.

See the section on Passwords in the System Design Concepts manual for further information on the password system.

PAUSE Display Message and Pause for Confirmation (0)

Command file execution can be controlled manually with the PAUSE command. The keyword PAUSE by itself will cause the message

HIT ANY KEY TO CONTINUE..

to be displayed for 30 seconds or until a key is pressed. In either case, processing will continue with the next line in the command file.

There are several optional parameters that can be used with the PAUSE command. These parameters are:

1. Prompt Message
2. Default Response (Yes or No)
3. Timeout in Seconds
4. Continue File Processing on Negative Response (Yes or No)

Not all parameters must be specified but the order must be correct. Commas should be used as delimiters in the command so that the Prompt Message can have embedded spaces and omitted parameters can be indicated.

If a parameter other than the Prompt Message is provided, a YES or NO will be displayed after the prompt message which indicates a default positive or negative response, respectively, that will be generated by the PAUSE command. This response is used to determine if processing of the current command file should continue or be terminated.

The default response will be accepted by the PAUSE command if the user simply presses the ENTER key or allows the command to time out. The default response that is displayed is controlled by the second parameter in the PAUSE command. If this parameter is not specified a NO will be assumed. The user can override the default response by entering YES, ON, or 1 to generate a positive response or by entering NO, OFF, or 0 to generate a negative response.

Normally, a negative response will terminate file processing and a positive response will allow processing to continue. However, this is only the default setting and can be changed in one of two ways. First, the SET PAUSE command can be used to control file processing. The default setting for each task is SET PAUSE ON, which allows file processing to continue only when a positive response is generated. As an alternative, the SET PAUSE OFF command can be used to allow file processing to continue regardless of the response generated by the PAUSE command. Remember, the SET PAUSE command is used on an individual task basis, not as a global setting.

Second, the fourth parameter on the line of the PAUSE command can be used to override the current SET PAUSE setting. Specifying a YES for this parameter will allow file processing to continue regardless of the response generated by the command. Likewise, specifying a NO will allow file processing to continue only when a positive response is generated.

If command file execution is to continue on a negative response , the user must be able to determine if the response was a Yes or a No. This can be done with the @yes(0) and @no(0) functions as shown below.

```
Example 1:
set pause off        ; do not stop on a No response
pause, Enter Y or N, y, 10  ; wait 10 secs, default resp is Yes
if @yes(0)
 msg User entered Yes
else
 msg User entered No
endif

Example 2:
set pause on         ; stop on a No response
 pause, Enter Y or N, y, 10
if @no(0)
```

```
 msg This line will never appear because processing stops on No
else
 msg This line will appear if Yes is entered
endif

Example 3:
pause, Enter Y or N, n, 10, y  ; last y will cause file
                  ; processing to continue no matter
                 ; what the response
if @yes(0)
 msg User entered Yes
else
 msg User entered No
endif

 Examples:
PAUSE
PAUSE, DO YOU WANT TO CONTINUE?
PAUSE, WANT TO CONTINUE?, N, 10   ; wait only 10 seconds
PAUSE, OK TO GO ON?, YES
PAUSE, WANT TO CONTINUE?, Y, 45, Y
```

PHONE Set the dial out Phone Number or Radio Call Sign (2)

The PHONE command was used in earlier versions of the software to set the phone number for the currently selected RTU. This has been replaced by the complex LINK system (explained elsewhere) that allows for many phone numbers to be used in each computer. The PHONE command has been retained, although its action is different than earlier (pre 1989) versions.

Each RTU has a default link assigned to it in the main configuration file when the program is first started. The PHONE command is now used to set the phone number for the default link number assigned to the current RTU.

A typical use of the PHONE command is similar to the earlier versions, where a command file will contain a SELECT RTU command followed by a number of RTU related commands. In this context, the PHONE command will work as before.

Examples:

```
PHONE 555-1212
PHONE WC433A     ; radio example
```

Related: LINK

PLAY Audio Subsystem Control (1)

PLAY is used to playback prerecorded sounds over the PC sound sub-system. Installation of the sound system requires system-specific drivers to be loaded prior to the start of RTUMON3. SCADAWARE allows playback of multiple sounds in sequence such that separate discrete phrases can be joined to make meaningful messages. Once the required individual phrases are available as WAV files on disk, they can be combined into various strings to announce any desired message, including those with numeric runtime values. Phrase playback can be manually initiated from the keyboard or from a menu selection. They can also be activated automatically from a TSP command file, or by an alarm condition for any data channels set to Play Sounds on abnormal.

PLAY can also automatically control a Push-To-Talk (PTT) output control to key a radio or PA system. If specified with the SET PTT xx command, the PTT output channel will be activated and a slight wait inserted before playback of the audio message. After the message is complete, another slight delay is inserted before the PTT is deactivated.

PLAY occupies complete attention from the task executing the commend, although other tasks are allowed to run during playback. If the local console executes PLAY, it will be held up until the message is complete. The utility task can be used instead to allow playback in the background. In

complex systems, a separate PLAY task can be installed to provide separate processor time specifically for audio playback.

The syntax of the PLAY command provides for direct specification of the desired phrases. It also allows for a tag reference to be made which causes the command to search a text file which associates predefined phrases with specific tag names. This permits direct announcement of specific phrases, or indirect announcement of phrases associated with a specific data point.

To play a direct sound or list of sounds, the PLAY command is used with the desired phrases on a single command line, as in:

```
PLAY COMPRES SHUTDOWN
```

This simple example causes the PLAY subsystem to look for two audio files, COMPRES.WAV and SHUTDOWN.WAV, and play them in sequence.

Phrase sequences can also be executed by reference to a tag name. The file which contains the predetermined phrases is a simple ASCII text file for each logical RTU on a system. The default file extension is .PLY. Each line begins with the tag name for the point to be defined, followed by the list of phrases associated with this point. A short file for an RTU named GASMET1 may look like this:

```
; FILE GASMET1.PLY
; Play List for RTU at Gas Meter One (GASMET1)
DIFF1  Low Flow Rate
COMPSD Compress Shutdown
METER1 Low Meter 1 Rate $(M1)
```

This file defines message strings for three separate RTU data points: DIFF1, COMPSD, and METER1. The words following the tag names are WAV sound file names which are created with software provided by the sound system vendor. The sound files required for this sample are:

LOW.WAV	Low
FLOW.WAV	Flow
RATE.WAV	Rate
COMPRESS.WAV	Compressor
SHUTDOWN.WAV	Shut Down
METER.WAV	Meter
1	One

Note that several WAV files are used more than once, demonstrating how sound files can be combined and used for more than one complete message. Note also that a runtime expression $(M1) is used for the METER1 message. The $(M1) will be expanded at the time of announcement into a decimal number, such as 16.5, and will be automatically announced as "ONE SIX POINT FIVE." Any numeric value in the PLAY string will be said as a series of numbers. This requires that the numbers 0-9, and the word POINT exist in the Library containing the prerecorded messages.

The audio WAV files are normally stored in a separate directory which must be specified with the SET PATH SOUND XXXXXX command. Once specified, the PLAY command will use this directory as the default source for all sound files. To override this specification, enter the complete file name for the desired file, as in:

```
PLAY C:\AUDIO\HELLO
```

The presence of \ signals PLAY to not search the default directory. Note that the default file type of WAV is assumed and need not be entered as part of the phrase file name.

The command line options for the PLAY subsystem all begin with the slash / character. Only the first letter is significant, although other letters can be provided for clarity.

```
PLAY /TAG xx   ; Play message sequence for tag name xx in current or specified RTU
PLAY /PATH xxx ; specify default path for Sound files
PLAY /HEADER xx       ; display file data for sound file xx.wav
PLAY /STOP            ; cancel any pending sound playback
```

When a point set to PLAY on Abnormal first goes into the abnormal state, a message is sent to the Utility task in the form of:

PLAY /TAG RTUNAME.TAGNAME

This causes the current utility task to start the PLAY command in /TAG mode where it expects to find a file named RTUNAME.PLY with an entry for point TAGNAME. If the file exists, and the tag name is matched, then PLAY will work with the sequence of phrase messages specified in the file. If no matches are made, then the command is ignored.

It is useful to monitor the Util task (WATCH UTIL) during playback to see errors and status reports for the playback system.

Example:

```
PLAY This is a test  ; 4 phrases - "This", "Is", "a", and "Test"
PLAY /TAG COMPSD  ; play phrases associated with tagname COMPSD
PLAY /STOP           ; stop all playback activity
```

Related: SET PTT

POLL RTU Poll Control (2)

The POLL command is used to control the default link for an RTU. An RTU name can be specified with this command to affect the link for a particular RTU. If the rtuname option is not provided, the system will assume the currently selected RTU. This command can reset a link, activate a link, or reactivate a failed link. See the LINK command for more information about controlling links.

The POLL command has the following options. The [rtuname] code indicates that an optional RTUNAME can be specified to override the choice of the current RTU for the command action:

```
RESET [rtuname]     <- reset default link to idle state
RETRY [rtuname]     <- set default link to active if failed
NOW [rtuname]       <- activate default link for an RTU
REPORT [rtuname]    <- activate link and print a single report
```

The keyword RESET is used to return the default link for an RTU back to the idle state regardless of the current state of the link.

The keyword RETRY is used to return the default link for an RTU back to the active state if it has failed. This will allow the callout attempt to be restarted. This command is used to do a poll only if it has previously failed.

The keyword NOW is used to activate the default link for an RTU.

The keyword REPORT is used to activate a link and then automatically print a report at the time the data transfer is time-stamped. Using this option, a report will be printed as soon as a successful transfer is complete, but will not occur if the transfer cannot be done. This eliminates reports that contain old data.

Examples:

```
POLL RESET    ; Reset default link for current RTU
POLL RESET wc458 ; Reset default link for wc458
POLL RETRY    ; Re-poll current RTU if link failed
POLL NOW      ; Poll current RTU
POLL REPORT wc458 ; Poll wc458 and print report
```

Related: LINK

PORT Set and Read PC Output Port (2)

This command is used to read the input value of a port and to send byte values to a port on the PC's I/O port bus. This can be used to control special function cards installed in the system (like small relay alarm cards). The syntax is PORT followed by the port number specified in either decimal or hexadecimal format. This will read the input value of the port and display the value in both decimal and hexadecimal notation. An additional parameter consisting of a byte (0-255) value can also be specified as a third parameter. If a value is specified, the value is sent to the port number provided as the second parameter. Although the address is the same, the physical registers that are read from and written to may be different.

As a useful example, consider a Host RTU that does not have an RTU driver installed. The SET HORN command alone will not work on this Host because it does not have the ability to automatically control physical outputs like an RTU. A simple relay card can be installed to provide a status output to control a local horn. But, a specific TSP program must be prepared to control the horn when the specified HORN output is turned on.

```
; This runs whenever output point 08 turns on
; Assume relay card at address 768.
; board needs the codes 10 for horn on, 12 for off
; This also assumes that the SET RFILE option is in effect

if $1 = 1
 PORT 768 10  ; turn on the horn
else
 port 768 12  ; turn the horn off
endif
```

Examples:

```
PORT 768 2  ; send the value 2 to port 768
PORT 770    ; display value of port 770
```

PRC Process Packet Radio Controller Command (1)

Systems equipped with packet radio controllers (PRC) can have commands sent to the unit via the PRC command. The command will send the text line following the PRC keyword to the unit after placing it in command mode. If the unit was on-line prior to issuing the command, and the command is not a DISCONNECT, then the PRC will be placed back on line after the command is issued.

The program sends a break signal to the PRC to place it in command mode. After sending the command to the PRC, the program will place the PRC back in Transparent mode for further data transmissions.

Common PRC commands are as follows:

CONMODE T	Set connect mode to transparent
ECHO off	Set local echo off or on
FLOW on	Stop echo to screen while typing
MONITOR 0	Select data transmission monitor level
MYCALL xxxxxx	Set station call sign (normally RTU name)
RETRY 3	Max number of frame transmission retries
STATUS off	Controls status codes sent to terminal
TBAUD 1200	Sets terminal (i.e. RTU) baud rate
TCLEAR	Clear the transmit buffer
TXDELAY 50	10s of Milliseconds between push to talk and xmit
XFLOW off	Software xon-xoff flow control.

Examples:

```
PRC ECHO OFF
PRC FLOW ON
PRC MYCALL SMI174
```

Related: HAYES, ATTAch

PROGram Program Data Channel or Comm Link (3)

PROGRAM is used to change parameters associated with data channels and comm links. It is similar to the CONFIG command, but can be used by any RTU type task and not just task 0. The user interface is line-by-line rather than screen-oriented. The system will prompt the operator with the required information and the current value for each parameter. The operator can then enter the new information, or simply hit [ENTER] to accept the current value. Entering a ^C (Ctrl-C) will abort the sequence. See the System Design Concepts manual for further information on channel parameters.

If the first parameter is the keyword "LINK", then the system expects the link number as the third parameter. The user is then placed into a prompted program sequence for the specified link.

This command can be used over a serial communications line to program a channel or link at a remote location. In SCADAWARE Lite, PROGRAM is an overlaid command, and must therefore be locked in with a LOAD PROGRAM statement in the DAT file if remote operation is desired. To program a channel or link locally use the CONFIG command.

Examples:

```
PROG S2
PROG LINK 2
```

Related: CONFIG

PUBLIC Declare Globally Accessible Variables (1)

In addition to the normal channel values, the expression evaluator supports two types of variables which are declared as either PUBLIC or LOCAL. When a variable is declared it is associated with a particular task. The number of variables allowed for a task is determined by a VARIABLES line in the main configuration (DAT) file. The default number of variables allowed for each task is 0.

A LOCAL variable is only accessible by the procedure that declares it or by any nested procedure called with the GOSUB command. When the procedure that declared a LOCAL variable is terminated the variable is released. PUBLIC variables are accessible from any procedure and are released only by using the RELEASE command. Public variables are always located at the highest nesting level regardless of the subroutine level at which they were defined. Unlike local variables, public variables cannot be re-defined at a lower level as a separate data item.

Variables with the same name can exist for different tasks without conflict. Several variables for a single task can also exist with the same name as long as they are declared at different levels. Consider the following three example files:

X.RTU

```
PUBLIC A
LOCAL B
CALC A = V1
CALC B = V2
READ Y
```

Y.RTU

```
LOCAL B
LOCAL C
CALC B = V3
CALC C = A + B
GOSUB Z
CALC D = C * 10
```

Z.RTU

```
PUBLIC D
LOCAL E
CALC E = A + B
```

If the file X.RTU is processed this is what would happen. Variables A and B would be declared and set to the values of channels V1 and V2. The READ Y command would then terminate the processing of X.RTU, releasing local variable B, and begin processing the file Y.RTU. Within this file the

local variables B and C would be declared. (This variable B has nothing to do with the variable B declared in X.RTU.) After setting B to the value of channel V3, C would be set to the sum of variables A and B. Accessing variable A from within this file would be allowed because it is a public variable and can be accessed by any procedure. The GOSUB Z command would then transfer control to the file Z.RTU. The GOSUB command would only suspend the processing of the file Y.RTU, not terminate it. Therefore, the local variables B and C would still be declared and could be accessed within Z.RTU. After reaching the end of the file, Z.RTU would terminate and release variable E. Control would then return to Y.RTU which would set variable D to C*10. Y.RTU would then terminate and release local variables B and C. At this point all local variables would have been released and only the public variables A and D would remain declared. These two variables could then be used by other commands (or other command files) or freed with the RELEASE command.

Whenever a variable is referenced, the evaluator begins a search to locate the variable or channel. The list of RTU channels is searched first. If no match is made, the list of LOCAL variables is searched. If still no match is made, the list of PUBLIC variables is searched.

Example: PUBLIC TOTAL1, TOTAL2

Related: RELEASE, LOCAL

PURGE Purge Characters in Communications Input Buffer (1)

Within command files, it is often necessary to remove any garbage characters that may be waiting in the input buffer as the result of a communications failure or other unusual link termination. The PURGE command is used to empty the input buffer so that any waiting characters will not be processed.

Example:

```
if @online(1)
 msg RTU is Online
else
 PURGE
 msg RTU disconnected
 return
endif
```

READ Start Processing Command File (2)

The RTU program normally accepts its input from a communications channel or local keyboard. Certain operations can be automated by placing the commands to be executed in a text file (or Library procedure) and telling the program to begin reading the file. Input from the file is processed as if it was typed in by the user. TSP file processing via the READ command is the primary means of automating SCADAWARE processes.

Any command file can be loaded into a fixed memory buffer which acts as a Library of procedures. TSP command files can then be executed from RAM memory rather than disk. See the LIBRARY command for details on how to load command files into the Library.

To have a command search the Library for a command file the file type must be omitted from the file name in the command. For example, consider the command READ MYPROG.RTU. When this command is processed, the program will search for the file on disk and execute it if found. If the file type is omitted and the command READ MYPROG is used instead, the program will first search the Library for a procedure named MYPROG and process it if found. If however, no Library procedure with that name is found, the program will assume a default file type of RTU and search on disk for the file.

If the READ command is processed from within a command file (or Library procedure), the current file is terminated and processing continues at the start of the named file. Refer to the GOSUB command for a way to temporarily suspend processing of the current file while the called file is being processed.

When a READ command is processed, the system picks up command line parameters beginning with the file name. This parameter, and those following it, are referenced within a command file as special $ codes similar to the ones explained in the ECHO command. Each parameter is referenced by number, with $0 being the current file name, $1 being the first parameter, $2 the second, and so on. For example, the line:

```
READ SETOUT o1 RTU1
```

has parameters as follows:

```
$0 SETOUT
$1 o1
$2 RTU1
```

Note that the highest parameter number allowed is $9, and all parameters are cleared when a new file is read.

Examples:

```
READ CLEAROUT
READ CLEAR.TXT
READ SHUTIN RTU1 30 ; pulse RTU 1 shutin for 30 secs
```

Related: GOSUB, REREAD, PAUSE, SET ONLINE

REBOOT Force Cold Boot of Computer (2)

It is sometimes necessary to completely reset a computer to clear some bogus condition that cannot be cleared any other way. This is often done by pushing the reset button, but this can also be done remotely if the computer is still processing commands.

The REBOOT command causes the computer to execute the same code that occurs when the reset is pressed. In order to execute, the system will present a prompt which must be entered by the operator exactly as described by the command. This will prevent inadvertent operation of this command. It is likely that the communications channel will be lost when this command executes, so a re-dial will be needed after using it.

Systems equipped with TEST's Modem Monitor device normally do not need to use the REBOOT command but can reset the system remotely using this device.

RELEASE Remove Temporary Variables (2)

Variables created with the PUBLIC and LOCAL commands can be removed with this command. After they are released, they will no longer be available for any purpose. The memory previously used by the variables will be made available for later variable definitions.

This is most often used to control memory allocation of temporary variables used within a single operation. When the operation is complete, the variables should be released so that later operations will not run out of variable space. The number of variables allowed for a task is determined at system startup by an entry in the main configuration (DAT) file.

The keyword ALL can be used with the RELEASE command to remove all variables that are currently defined for a task.

Examples:

```
RELEASE ALL      : release all variables for a task
RELEASE TOTAL1 TOTAL2  ; release variables TOTAL1 and TOTAL2
```

Related: PUBLIC, LOCAL

REPORT Generate RTU Data Reports (2)

The RTU data can be sent to the PC printer or to a disk file for later printing. REPORT by itself generates a report for the current RTU which goes to the printer. REPORT followed by only a disk file name will cause the report to be sent to that file. An existing file with the same name is overwritten.

Report entered from the keyboard or command file will cause the current task to print the report. Hitting the F4 print key will set a flag which causes a REPORT command to be sent to the Utility task (it one exists). In the SCADAWARE LITE program, the command LOAD REPORT must be used in the main configuration (.DAT) file in order for the REPORT command to be processed by any task other than task 0.

CAUTION: Off-line printers can hang the system, so be sure that the printer is always turned on and powered up.

An option is available to use a FOR and/or a TO phrase to specify which RTU to print as well as the destination for the report. This allows more flexibility in printing reports for multi-RTU installations. For example:

```
REPORT FOR SMI7 TO SMI7.REP
```

This example specifies SMI7 as the RTU to be reported, and the destination is disk file SMI7.REP. If the FOR is not specified, then the current RTU is assumed. If the TO is not specified, then the printer is assumed. Either option can also be used by itself.

The REPORT command also has the option to print only the points that are currently in alarm (i.e. the new alarm or in-alarm state). To use this feature the keyword ALARM must be specified somewhere in the REPORT command, such as:

```
REPORT FOR SMI7 ALARM
```

Reports can also be printed in any of the 3 display styles available on the normal F7 display screens. The default will be the natural format for each channel. To generate a report using a specific style, the keyword STYLE must be specified in the REPORT command followed by an additional keyword. The keywords available for specifying a particular style are NORMAL, ALARM, and SETPOINTS.

Examples:

```
REPORT       ; current RTU to printer
REPORT Data.txt  ; specify the disk file for cur RTU.
REPORT TO daily.rep for hi356a ; specify both
REPORT FOR SMI7 STYLE NORMAL
REPORT FOR SMI7 STYLE ALARM
REPORT FOR SMI7 STYLE SETPOINTS
REPORT FOR SMI7 ALARM STYLE ALARM
```

Take a careful look at the last line from these examples. Note that the first ALARM keyword on the line is a stand-alone keyword which requests only the points that are in alarm to be included in the report. The second ALARM keyword is a directive to the STYLE keyword which requests a specific printout format.

Related: SET REPORT, SET PRINT, SET EJECT

REREAD Restart Execution of Current Command File (0)

Command files often need to repeat over and over in a continuous loop. The REREAD command can be placed anywhere in the file to cause execution to start from the top. This would be similar to placing another READ command in the file except that the name of the file must be coded directly into the command. REREAD avoids this by letting the processor simply reset the file pointer and start reading from the top. This also saves the DOS overhead of closing and then re-opening the file.

A bit of caution is worth noting when using this command. When a task is processing a command file, it does not allow commands from any other sources (such as other tasks or a communications channel) to be processed until the file is completed. Because the REREAD command continues to process a file without ever closing it, queued messages won't be processed as long as a file is being reread. If this is a problem, there are alternatives to this command. For example, instead of using the REREAD command, use the FORCE command to have the task queue up a message to itself to read the file again. This will allow any pending messages to get processed before the command file begins processing again.

Using REREAD does not clear out the command line parameters entered when the file was first started.

Example:

```
If loops < 10
  reread
endif
```

Related: READ, SET ONLINE

RESET Reset Misc. Values and Conditions (2)

The RESET command provides different keyword parameters to affect different things. If no parameter is specified then the keyword ALARM is assumed.

If a point is set to be manually reset (i.e. not auto reset), then either the RESET command or the RESET function key must be used to restore the channel's alarming ability. When a point has been acknowledged, it can be reset with the RESET button. Points that have not been ACK'd can be ACK'd and reset with the RESET command. This allows for command files to specifically clear any points listed after the RESET command without knowing about the point's alarm state. All channels, a channel range, or a list of channel ranges can be specified to reset alarm channels. The special keywords are :

ALL	<-	Resets all cleared alarms channels and resets all counter channels to 0
ALARMs	<-	Reset all cleared alarm channels.
xx xx xx	<-	Channel identifiers like S1, O10, etc. Resets channels that have cleared alarms

Other keywords used with the RESET command are :

FILES	<-	Close all open files
OUTputs	<-	Reset local output channels to normal state
CALLouts	<-	Reset the link for the current RTU and remove current need to do a callout
COMM	<-	Reset communications channel for current task

Examples:

```
RESET ALL
RESET S1:S32
RESET A1..A10 S1..S8 01..08
RESET CALL
RESET OUTPUTS
```

Related: ACK, CLEAR

RESYNC Reset Incoming and Outgoing TSP Block Counters (2)

When the program is operating in RTU mode (computer to computer operations), TSP command lines are automatically formatted to allow for communication error checking. As part of the error checking system, each TSP message sent to another unit is sequentially numbered. Each message received from another unit is also numbered. If the sequence number of an incoming message has already been processed then the duplicated message will be ignored. This type of error checking is useful because it prevents messages from getting out-of-sequence due to poor or slow packet communications.

In situations of continuous communication the counters for these incoming and outgoing messages may exceed their limits. If this happens, the counters will roll over and start counting again from 0. This will cause the next message that is transmitted to have a sequence number less than the previous message. Subsequently, the message will be ignored. To keep this undesirable situation from occurring the RESYNC command can be used.

The RESYNC command is used to reset the incoming and outgoing block counters. In order for the counters on both units to be reset and stay synchronized the RESYNC command must be used in two steps. First process a BLOCK RESYNC command to reset the message counters at the other unit. Then process a RESYNC command that will reset the message counters on the local unit.

Example:

```
 ; Continuous download file
set online on  ; only continue if connected
block resync ; tell other unit to start over (see below)
resync ; reset local counters
scan A1:10 E
scan S1:s16 R o1:o16 R
reread
```

IMPORTANT RESYNC NOTE: Older (pre 10/92) software required that a BLOCK RESYNC be sent to a remote unit prior to doing a local resync. TSP has been enhanced to allow an automatic resync by the remote unit *whenever message number one* is received. Therefore, the BLOCK RESYNC is no longer needed to get a remote unit in sync with the sending unit because the receiver will automatically resync when it detects that the sender has reset its counters. Therefore, the BLOCK RESYNC in the above example would no longer be needed. However, the block resync may still be needed when communicating with systems with older software.

RETURN Terminate Execution of Command File (1)

Execution of a command file normally continues to the end of file before termination. If it is desirable to stop a file in the middle somewhere, the RETURN command can be used. File execution is stopped and the command processor returns to the command mode just as if the file had ended. If the processing of the file was started from within another file then the processor returns to the calling file.

Example:

```
if (v1 > 10)
  msg Terminating the loop now
 return
endif
```

Related: STOP, GOTO, REREAD

RFLAG Manipulate RTU Control Bit Flags (2)

Each RTU defined on a system contains a predefined array of 512 flags that can be accessed by any task. These flags are very similar to public variables except for the fact that each flag can only be set to 0 or 1. This set of flags is also unique to each RTU. The RFLAG command can be used to set RTU flags for the currently selected RTU. A similar command, the FLAG command, can be used to set flags from a group that are common to all RTUs. To set a flag, specify the number of the flag (1 - 512) followed by either a 0, a 1, or a keyword. Allowable keywords are:

TRUE	value of 1
FALSE	value of 0
ON	value of 1
OFF	value of 0
FLIP	invert current value

Several flags can be set using a single RFLAG command. This can be done either by listing each flag and its value or by specifying a range of flags followed by a single value. A range of flags can be specified by using either the ".." or ":" notation. All flags can be cleared by using the keyword RESET. The DUMP command can be used to display the status of all flags for the current RTU. The status of an RTU flag can be checked from within a command file by using the @RFLAG(x) function.

Examples:

```
RFLAG 1 0              ; set RTU flag 1 to 0
RFLAG 6 TRUE           ; set RTU flag 6 to 1
RFLAG 1 ON 2 OFF       ; set RTU flags 1 and 2 to 1
RFLAG 1..10 ON         ; set RTU flags 1 through 10 to 1
RFLAG 40 FLIP          ; change value of RTU flag 40
RFLAG RESET            ; clear all RTU flags (set to 0)
```

Related: FLAG, PUBLIC

RTG Real Time Graphic Control (1)

RTG is used to describe and control Real Time Graphic objects in the graphic display subsystem. Refer to the Graphics System Documentation for more information on RTG.

SAVE Save Current RTU Setup Information To Disk (3)

This SAVE command is used to send a logical RTU's channel configuration to a disk file. An optional file name can be specified after the keyword TO. If none is provided, then the name of the RTU plus the ".RTU" extension is used. This saves all of the channel configuration data and current value for each channel in the current RTU. The file that is generated by the SAVE command is a simple DOS Text file that can be easily transported to another unit via diskette or network connection. The lines in the file contain the setup information for each channel in the RTU. The file also has a SELECT line at the start to insure that the computer processing the file selects the correct logical RTU prior to processing the contents of the file.

Examples:

```
SAVE              ; save to standard rtuid.rtu file
SAVE TO C:SETUP.RTU ; save to alternate file
```

Related: LINK SAVE

SAY Display Expression Value or Text String (1)

The SAY command can be used in command files to display evaluator expressions, channel values, string variables, or text messages on the CRT. It is normally used after positioning the cursor with the CURSOR command. The process of evaluating what to display using this command is exactly the same as it is for the CURSOR command. The only difference is that the CURSOR command specifies the coordinates of where the display will be located on the screen and this command does not.

An expression may be displayed by enclosing it in parentheses. A channel value may be displayed by referencing a channel by number or by tag. String variables may be displayed simply by using the % sign followed by a number 0-9. Any parameter that can not be identified as an expression, channel name, or string variable is considered to be a text message. All text messages should be enclosed in double quotes. Although this is not necessary, it is good practice and allows commas and spaces to be embedded in the messages. Parameters should be separated from one another with commas.

When text is displayed, it is expanded as is done in the ECHO command. This allows the special $ values to be displayed on the CRT from within a command file. Individual parameter formatting can also be performed by attaching a format string (sometimes referred to as a "picture mask") to the end of an expression, channel variable, string variable, or text message. The format string consists of the accent character (') followed by a string of characters which each represent a specific format action. Valid characters that can be used in a format string and their associated actions are as follows:

A — If formatting a channel variable, this will cause the display to use the channel's alarm video attribute. The A is stripped out of the picture mask and does not occupy a character position.

X — If formatting a channel variable, this will cause a string to be displayed which is dependent on the type of channel being formatted. If formatting a Status Input or Output type channel, the channel's Abnormal or Normal Text will be displayed, depending on the current state of the channel. For any other type channel, this will cause the channel's Units to be displayed. The X is stripped out of the picture mask and does not occupy a character position.

Z — If formatting a channel variable, this will cause the Channel's Description to be displayed. The Y is stripped out of the picture mask and does not occupy a character position.

Cxx — Center the resulting string in a field that is xx spaces wide. Note that 2 digits must be provided, as in C10 to indicate a field of 10 or C06 for a field of 6. The Cxx is stripped out of the picture mask and does not occupy any character positions.

Lxx — Left justify the resulting string in a field that is xx spaces wide. Note that 2 digits must be provided, as in L10 to indicate a field of 10 or L06 for a field of 6. The Lxx is stripped out of the picture mask and does not occupy any character positions.

Rxx — Right justify the resulting string in a field that is xx spaces wide. Note that 2 digits must be provided, as in R10 to indicate a field of 10 or R06 for a field of 6. The Rxx is stripped out of the picture mask and does not occupy any character positions.

^ — This character is used to indicate a comma position in the format string. Because the program's command line parser is sensitive to the use of commas and spaces, it is necessary to use a non-comma character to indicate a desired comma position in the format string. To be consistent with other parts of the program, the up-caret character (^) found above the 6 key is used for this purpose. Any ^ characters found in the format string are automatically replaced by commas.

The following characters should only be used in a format string when an evaluator expression or a channel variable is being formatted and the resulting string represents a number.

#	This represents a digit position. Unused digits are left blank if the picture mask does not contain a * or @ character. A floating minus sign is displayed in front of the number if the picture mask does not contain a + or - character and the number is negative.
*	This represents a digit position. Unused positions are set to * instead of blanks. Only one * is needed in the picture mask to set this option. The sign will not be displayed unless the picture mask has a + or - character in it also.
@	This represents a digit position. Unused positions are set to 0 instead of blanks. Only one @ is needed in the picture mask to set this option. The sign will not be displayed unless the picture mask has a + or - character in it also.
$	This represents a digit position. A floating dollar sign is placed in front of the resulting number.
+	The resulting string will contain a plus sign in the position designated by the + character in the picture mask, provided the number is positive. If the number is negative, the - character is displayed in this position instead.
-	The resulting string will contain a minus sign in the position designated by the - character in the picture mask, provided the number is negative, If the number is positive, this position is left blank.

Some formatting examples of the number 123456.789 are:

PICTURE MASK	RESULT	
######	123457	Note Rounding
######.#	123456.8	Note Rounding
###^###.##	123,456.79	Comma and Decimal
+######.###	+123456.789	
@@@@@@@@@@.###	0000123456.789	Leading Zeros
##########.###	__123456.789	Leading Spaces
A######	123457	Alarm Video
L12######.#	123456.8__	Left Justified
R12######.#	__123456.8	Right Justified

If no picture mask is specified for evaluator expressions and channel variables, the format of the displayed numbers will be determined in the same manner as is done for channel displays and reports. The default number of decimal places to be displayed is controlled with the SET PLACES command. This is a global setting which affects the display of all expression results and channel values. However, each channel can be configured to display a number of decimal places that is different than the default. If this channel parameter is set to something other than a negative number, the default number of decimal places is overridden and the specified number of decimal places will be displayed.

Examples:

```
SAY "The current time is ", $T
SAY "The value of V1 is ", V1
SAY "The value of string variable 0 is ", %0
SAY V3'R10
SAY (SQRT(V1)+1)
SAY V1'A
SAY V1'AC10
```

Related: CURSOR, DISPLAY

SCAN Request Range of Data Values for an RTU (2)

The SCAN command is the primary means of requesting channel data from a SCADA system for transmission to a remote unit. The SCAN command can access simple channel values as well as the internal settings of the channels that control their configuration or alarm action. When getting data values for use in a computer, it is desirable to get them in a condensed format if possible. The SCAN command specifies a range of channels to be sent via a DATA command that the receiving RTU will provide.

Besides the required channel range, various code letters are used to indicate the format and content of the data to be scanned. These codes go on the SCAN command line just after the channel range specification and are repeated in the resulting DATA line that the SCAN command generates. Most options are used one at a time, while a few can be used in conjunction with other codes. The available code letters are:

CODE	MEANING
M	Alarm Mode (16-bits representing alarm mode settings)
W	Time delay (Wait) till alarm, whole seconds only
A	Alarm time and date
E	Engineering Units data format
R	Raw data format
L	Low Setpoint (Engineering units)
H	High Setpoint (Engineering units)
D	Deadband (in engineering units only)
G	Callout Group (whole numbers only)
T	Time and Date that channel was started
@	Time stamp, link reset signal, and remote channel log
#	Time stamp clear
+	Add to existing value
[	Julian time and date format
]	Compress duplicate times and dates on a single line

Not all options apply to all channels types. The following matrix identifies the options which are not universially available. The ● symbol indicates that the option for that column is available for the channel type of that row.

	R	D	T
STATUS IN	●		
STATUS OUT	●		
VALUE		●	
FUNCTION		●	
TIMER		●	
AGA3		●	
ANALOG IN	●	●	
TOTALIZER		●	●
COUNTER	●	●	●
PID	●	●	

The # code can be used with other codes and is usually placed on the first SCAN line. The # code will cause the system processing the DATA line to clear out the update time and date. This allows transfers that are terminated abnormally to be indicated by an invalid time and date. Otherwise, the

previous time and date would remain as the valid update time stamp.

The @ code can also be used with other codes and is usually placed on the last SCAN line. The @ code will cause the system processing the DATA line to do several things. First, the time and date of the RTU for which the command is processed will get updated using the current time and date. The default link for that RTU will then be reset. The current link for the system is also reset. This may or may not be the same as the default link that gets reset. A report will be printed if set to do so using the SET REPORT ON command or if the SCAN command was processed as a result of the POLL REPORT command. Next, any channels that are in alarm and require logging will get logged. The reason for the delay of the logging process until the end of a transfer is to allow the alarm time and date for each channel to be scanned before doing the log (using the A option). Finally, a command will be queued up to read a file with the same name as the updated RTU and the extension ".UP". The "UP" file can be used to perform any function desired at the end of a data transfer. For example, on systems using databases the "UP" file can be used to update a database with the newly received data.

The + code can be used with the E and R codes to cause an incremental update of a channel's value. Normally, when a SCAN is done for a channel's value using engineering units, the system that processes the resulting DATA line will replace the channel's current value with the new value. Using the + code along with the E code will cause the new value to be *added* to the current value. This is most useful for counter type channels that are read then cleared on every poll.

The [code is used in conjunction with codes that cause times and dates to be transferred. This code causes times and dates to be transferred as integer values rather than formatted time and date strings. For time, the integer value represents the number of seconds since midnight. For dates, the integer value represents the number of days since 1/1/1900. Transferring times and dates in integer format reduces the number of characters that must be transmitted from one unit to another.

The] code is also used in conjunction with codes that cause times and dates to be transferred. This code causes duplicate times and dates in a DATA line to be transferred simply as T and D. This reduces the amount of data to be transferred between two units when times and dates are the same for consecutive channels in a range. The optimal choice is to use both [and] when scanning for a time or date because together these codes will generate the smallest DATA line possible.

The L and H codes generate different results when scanning different types of channels. The high and low setpoints for value type channels (all except STATUS and OUTPUT) are sent as real numbers representing the high and low alarm values. The low setpoint for a digital channel is sent as either a 0 or 1 to indicate the normal state of the channel. The high setpoint for a digital channel is sent as either a 0 or 1 to indicate the alarm condition for the channel. It is unnecessary to send both High and Low for digital channels. Either one will do.

The M code will generate a DATA line containing the alarm mode settings of the channels scanned. Normally, HOST and RTU systems have slightly different mode settings. The differences are usually related to Call on Alarm, Call on Reset, and the Sound Horn setting. In order to change the alarm mode setting at the RTU, the user will re-program the HOST to have the desired settings for the RTU. After sending the mode to the RTU, simply re-load the HOST's setup file for the affected RTU. This will restore the original HOST settings. This is done by using the READ command followed by the name of the RTU.

The R code letter is used to send raw (i.e. direct I/O data) rather than engineering unit data from one unit to another. In this case, the DATA command will represent raw input values rather than processed engineering values. A SCAN for raw data for Analog and Counter channels will generate a DATA line that contains an integer for each channel scanned. Scanning a Status or Output channel for raw data generates a different response. Status and Output channels can receive the off-on status in a packed format, with 8 channels per entry, rather than a separate entry for each channel.

For example, issuing a SCAN S1:S32 R will cause the system to respond with DATA RTUNAME.S1:S32 R a b c d where a b c d will be decimal equivalents of 8 bit values. These values represent 8 points of status input. Values for Status Channels 1-8 will be in the first number, channels 9-16 in the second, and so on. Using the RAW mode allows for faster data transfers than engineering mode.

Note that the first channel identifier in the DATA response includes an RTU name in "dot"

notation. This way, the receiver knows the RTU from which the data came so it can place it properly in its own data table. The channel numbers are relative to the sending RTU, so the receiver has to know which RTU is sending the data. The dot notation is a prefix that is allowed in may instances where a channel name is required.

The program takes the values presented in a DATA line and places the values in the system data table as the current point values. Of course, there can be no local I/O scanning taking place or the values will be quickly overwritten by local I/O points.

If the time option is specified for a Counter or Totalizer channel, the time and date will be sent as single entries separated by a space. Time always comes first in the pair.

Example:

```
SCAN Q1:Q2 T
DATA, RTU.Q1:Q2,T,11:22:33 01/02/88,10:20:30 01/02/88
```

Note that commas are always used as entry separators in the data response so that spaces can be used in the time and date field. The format of how data is transferred is up to the user.

SPECIAL @ENDS KEYWORD

The keyword @ENDS can be specified in a SCAN command to generate a DATA command indicating the update time and date for an RTU. The format of the command is SCAN @ENDS. An RTU name can be specified after the @ENDS keyword. If no RTU is specified the current RTU is used. The format of the response is DATA @ENDS RTUNAME TIME DATE. The system that processes the DATA @ENDS command will do everything that the @ option described above does. The only difference is that the update time and date are specified in the DATA @ENDS command instead of simply using the current time and date.

When an RTU with local channels processes the SCAN @ENDS command, the returned DATA @ENDS command contains the current time and date. When a unit with remote channels processes the SCAN @ENDS command, the returned DATA @ENDS command contains a time and date representing the last time-stamp for the RTU. This command is intended to be used at the end of a host to host download.

SCANNING FIRST-OUT ALARMS

A first-out alarm is the first channel to go into an alarm condition as long as another first out is not pending. Once an alarm is made the first out, no other alarm can become the first out until the current one is acknowledged. Subsequent alarms will still show as NEW alarms, but they will not be the first out.

The log system is used to track the first out alarm (lets call it the "FO" alarm). Log table entry number 0 is used to hold the first out entry, and it can be examined at any time by entering LOG FIRST.

The current FO alarm can be scanned by using the SCAN FIRST command. This will cause the system to generate a special DATA line that contains the FO channel number, FO time, FO date, and FO value. This DATA line will be processed by placing this information into the FO table entry for the receiving RTU.

When scanning the FO channel, always make it the last scan. It must be on a line by itself as well. It must be the last scan because it is possible that previous scan lines caused alarms that quickly became the FO channel. This may not really be the FO because the order of download is not related to the order that the alarms come in. So, let the normal alarm scan think it found the FO alarm, and let the final SCAN FIRST line override any FO found by the local alarm scanner.

Note that each RTU has a separate log table, and therefore a separate FO entry.

MULTIPLE SCANS ON SAME LINE

Multiple transfers can be done in the same command. This speeds half-duplex radio systems that require a lot of overhead when sending each response. To set up multiple requests, simply enter

the additional ones on the line after the first one. The SCAN keyword is not repeated, but each channel range and keyword must be present for each group. Also, note that all points must belong to the same RTU because only the first entry in the response will have the RTU ID.

For example, the following could be used on a small RTU to transfer all points in one command:

SCAN s1:s8 R o1:o8 R a1:a6 E m1:m1 E

Note that this SCAN command requests data for 4 types of channels in a single line. The response might look like this:

DATARTU1.s1:s8,R,12,/O1:O8,R,2,/A1:A6,E,12.3,4.33,2456.00,0,12.45,22.67,/M1:M1,E,17.65

Multiple scans must be set up so that the resulting line will not exceed 200 characters. This is the maximum line size allowed in all transmissions. In noisy radio environments, shorter lines may produce better overall results due to the difficulty in sending long transmissions.

Each extra group begins with a slash (/) character preceding the channel range code. The DATA statement is programmed to look for this character and separates each group when it matches this code.

Examples:

```
SCAN s1:s8 R
SCAN s1:s8 R a1:a6 E q1:q4 E@ ; simple RTU with time stamp
SCAN a1:a5 #E c1:c4 R+ s1:s8 R  ; counters are added
SCAN Q1:Q5 []T ; totalizer start times, compressed and numeric
SCAN @ends
SCAN FIRST
```

Refer to the DATA command for additional information on the responses to the SCAN command.

Related: DATA, DOWNLOAD

SELEct Select the Current RTU by Name (2)

The program always assumes a current logical RTU for all operations. The current RTU can be changed with the SELECT command by specifying the RTU name. For example:

SELECT VR167A

would cause the program to search the RTU list from top to bottom looking for an RTU named VR167A. If the name cannot be found, then the current RTU remains unchanged and an error message is displayed.

The only exception to this rule is that the system level RTU is selected by number rather than by name. To select RTU 0 and make it the current RTU, the following command must be used:

SELECT 0

The F10 key can also be used to pop-up a pick-list type menu listing all of the RTUs. Use the arrow keys to move to the desired RTU and select with the [ENTER] key. The pick-list will not work if the system has only a single RTU. The pick-list can also be displayed by having Task 0 process the SELECT command without specifying a RTU name.

During data displays, mouse equipped systems can move the mouse cursor to the upper left hand corner of the screen where the current RTU name is displayed. Pressing the left mouse button will cause the same RTU pick list to be displayed. The desired RTU can be selected with either the mouse or with the keyboard arrows as described above.

While in the display mode (F7 key), the plus and minus keys can be used to move from one RTU

to another. This also provides for a wrap around so that the user can move from the last RTU to the first RTU and back again without stepping through all the RTUs on the system.

Example: Select GA343A

SET Set RTU Operational Parameters (2)

SET provides access to a command sub-system that allows control and display of many time-out and configuration values that control system operation. Only the first 4 letters of the SET command option are significant, although others can be provided. Some set commands are no longer required, having been replaced by later program features or functions. To maintain compatibility with older software installations, all options remain from earlier versions whether they are needed or not.

All SET commands have a similar form which begins with the keyword SET followed by a sub-key which specifies the internal parameter to be affected. If no additional parameters are provided, SET will present a display of the current values for the specified SET option. No changes will be made. If additional parameters are provided, SET will take the appropriate action and display the end result of the changes.

Most SET options take a parameter which falls into one of the following categories:

Boolean	Any off/on condition such as 1, 0, Off, On, Yes, No
Numeric	Any valid number or expression which evaluates to a number
Tagname	Any channel tag name reference, with or without RTU override.

Note that each SET option influences specific portions of the overall system. Some affect only the current task, others the entire system, and others only the current RTU. Each entry in the listing below contains code letters in brackets which indicates the scope of the SET option:

[R]	Affects only the current RTU
[S]	Affects entire System
[T]	Affects task executing the SET command
[L]	Affects only the local CRT task

Set Option Example:

DELI	Delimiter [T]	Specify delimiter (separator) used for WRITE and WRITELN
	1st Param	Delimiter code, which is any character or the words SPACE, COMMA, NONE
	Default	Comma
	Example:	SET DELIM SPACE

The set option is DELI, which stands for delimiter. Use of this SET option will affect only the task executing the SET command, indicating that each task has its own delimiter value. The brief description of the option indicates that DELIM controls the separator character placed by SCADAWARE between values output with the Write command. The 1st parameter (which follows DELI) can be any character desired for the separator, or the special words SPACE, COMMA, or NONE.

An alpha listing of all set commands follows. ***Note that only the first four letters of the primary option keywords are significant.***

ACK	Acknowledge [S]	Tagname used for External ACK input
	1st Param	tagname
	Default	S0 (no input)
	Example:	SET ACK S8 ; use status input 8 for the ACK push button
ALER	Alert [T]	Enable/Disable Alert Response for Call-On-Exception
	1st Param	Boolean
	Default	Off (Alerts are not processed)

Example: SET ALERT ON

ATTR Attribute [L] Control CRT display attributes
ATTributes DEFAULT | norm title new old reset
Sets color (or mono) display attributes for local CRT. Each screen attribute has a number, and can be viewed with the DUMP ATTRibute command. The command SET ATTR DEFAULT will set all attributes to their default values. To set a particular attribute, enter SET ATTR followed by the attribute number follower by the foreground/background color combination. Use a + sign to brighten foreground colors and to cause background colors to blink. If more than one attribute color is specified, the additional ones go the subsequent screen attributes.

Examples:
SET ATTR DEFAULT ; sets attributes to default
SET ATTR 1 green+/black ; set normal text
SET ATTR 2 blue+/gray ; set title text
SET ATTR 1 green+/black blue+/gray ; same as both above

The available colors are red, blue, green, cyan, magenta, yellow, black, white, gray, and underline.

BACK Backup [L] Controls text editor backup file generation
1st Param Boolean
Default Off (no backups are made by the editor)
Example: SET Backup ON

BAUD Baud [T] Set baud rate for serial comm port.
1st Param Numeric baud rate, must be 300, 1200, 2400, 4800, 9600
Default Determined by DAT file setting, typically 1200
Example: Set baud 4800

BYE BYE [T] Control auto processing of BYE function
Tasks normally execute a BYE file when communications are complete. Special cases do not need or want this feature, as when polling multi-drop dumb RTUs in one LINK file. SET BYE can be used to disable the auto BYE function.
1st Param Boolean
Default ON
Example SET Bye OFF

CALC Calculator [T] Controls automatic processing of CALC if TSP command not found
Sets auto calculate mode to allow direct assignment of values to channels and data variables.
1st Param Boolean
Default: Off ("CALC" is not assumed at the start of each line)
Example: Set Calc On

CD Carrier Detect [T] Requirement of DCD for valid communications (Same as DCD)
1st Param Boolean
Default ON (DCD is required for data communications)
Example Set CD OFf ; don't need on multi-drop links

COMM Communications Timeout [T]
Sets the number of seconds between comm channel resets. No activity on a comm channel causes a BYE operation to occur to re-program any comm devices. The timer is reset every time a BYE operation occurs.
1st Param Numeric
Default 3600 ; one hour
Example: SET COMM 300 ; set for 5 minutes

CONF Config [L] Allow Limited Configuration For Password Level 2
A restricted version of the Config command can be accessed by users with levels less than the normally required level 3 if this option is selected.
1st Param Boolean
Default Off

Example SET Config ON ; allow Level 2 access to some config parameters

CONT Continuous [T] Selects constant screen display or one-time screen display
Set continuous action by the DISPLAY commands.
1st Param Boolean
Default On for Task 0, Off for others
Example: SET CONT OFF ; update screen only once per command

CRT CRT Selection [T] Define physical CRT type connected to the affected task
Sets type of physical CRT device connected to a task.
1st Param CGA, EGA, VGA, Mono, Ansi, None, QT2
Default: Mono or CGA for Task 0, Ansi for others.
Example: Set CRT VGA ; put console into 50 line mode

DCD Data Carrier Detect [T] Requirement of DCD for valid communications (Same as CD)
See CD above

DELA Delay [S] Turn alarm delay processing off and on
Determines if alarm time delays are in effect for all local channels. Turn off to speed up processing and simplify testing.
1st Param Boolean
Default On
Example Set Delay OFF ; stop delays to speed up alarm response for test

DELI Delimiter [T] Specify delimiter (separator) used for WRITE and WRITELN
1st Param Delimiter code, which is any character or the words SPACE, COMMA, NONE
Default Comma
Example: SET DELIM SPACE

DIR Directory [S] Specify various default directories
1st Param SCREEN, SOUND, RTU
2nd Param Full Path name of file directory for subsystem
Default Same as current directory at program startup
Example Set Dir Sounds c:\WAVFILES\

DITH Dither [L] Graphic File Conversion Dither Method
Graphic files which must be converted to a different resolution or color set are "dithered" according to the rule specified by this option.
1st Param None, Square, Dispersed, Cluster
Example: Set Dither Square

DRIV Drive [S] Set default drive (path) for RTU files
1st Param Path name for RTU procedure files.
Default Current directory at start of program.
Example: Set Drive C:\RTUFILES\

ECHO Echo [T] Turn comm echo off and on
Send characters received into a serial port out the comm channel. This is normally used for human type interface operations where character echo helps coordinate keyboard entry.
1st Param Boolean
Default On for local task, off for all others
Example Set Echo ON ;

EJEC Eject [S] Turn printer page eject after report off and on
Controls form-feed characters at the end of reports. Set On for direct printouts, normally off if print spooler in use. This affects all printouts for all RTUs.
1st Param Boolean
Default Off
Example Set Eject OFF

EMSG ErrorMessage [S] Error Message Trap control
Shows last error trapped by system. Also resets trap for new error.

1st Param Boolean
Example Set EMSG Off

ERRO Error Trap [T] Terminate TSP processing on any error
Controls command file execution for this task on any expression evaluator error. Setting off will allow execution to continue after an error.
1st Param Boolean
Default Off
Example Set Error ON ; stop processing on any expression error

FAIL Failsafe Comm Timeout [T] Comm timeout for each task after connect
Set command time-out (fail-safe) seconds. The task will terminate the comm link if no commands are entered in the specified number of seconds. Set to 0 to disable the automatic timeout and let a task remain online indefinitely.
1st Param Numeric
Default 60 seconds for all tasks except task 0 whose default is 0.
Example Set Fail 20 ; hangup after 20 seconds of inactivity

FAST Fast [S] Enable High-Speed processing for Timers, I/O Scans, Counters
Controls high-speed execution of certain internal operations such as 50 ms timers, status inputs for counters, and 50 ms I/O scans. Not normally used on 8088 type systems.
1st Param Boolean
Default Off
Example Set Fast ON

FIRS First Ack [T] Seconds to wait for first ACK from other unit after connect
Sets the number of seconds to wait for the first command after going online.
1st Param Numeric
Default 30 seconds, but usually set in STARTUP procedure for each task
Example Set FIRST 5 ; fast response for direct connect system

FONT Font Control [L] Specify default graphic font
Set graphic font used in subsequent text output. Replaced by FONT command.
1st Param Any valid Font Name
Default System
Example Set Font Roman

FRAM Frame [L] Specify default graphic frame style and size
Controls automatic placement and size of graphic frames on most objects.
1st Param None, Single, Double
2nd Param Pixel Size of frame edge
Default None

HORN Horn Output [S] Specify channel used for Horn Output
Set channel used to indicate alarm condition and to turn the horn off and on.
1st Param Tagname
Default S0 (no horn output)
Example Set HORN LOUDHORN ; use channel named LOUDHORN

KEYH KeyHit Sensing [T] Determine if any key press will stop TSP file execution
Controls command file processing when a key is hit. This command is effective only when being called from within a file and can be used to exit a loop on any key press.
1st Param Boolean
Default Off
Example Set Keyhit ON

LCD Liquid Crystal Display [L] Turn LCD optimized color set off and on
Monochrome LCD screens often look better with a special set of display colors (attributes). Setting LCD on will automatically convert the current color set to one that looks best on an LCD type screen.

LEVE Level [T] Set security (password) level for subsequent processing

Sets the required security level for further processing of a command file. Processing stops if the current user is not of required level.
1st Param Numeric LEVEL
Default 0
Example Set level 3

LINK Link [R] Set default link number for current RTU
Each RTU had a default link which is activated when any POLL request is made for this RTU. SET LINK allows the default link to be changed at any time for the current RTU.
1st Param Numeric (to indicate line number)
Default Determined by DAT file setting
Example Set Link 5 ; switch to phone line

LPP Lines Per Page [S] Set Lines Per page before form-feed on reports
Set report Lines Per Page (LPP). Setting to 0 will disable all page headings. Setting LPP to a very high number will suppress all but the first page heading.
1st Param Numeric
Default 60
Example Set LPP 45

MD MultiDrop [T] Multi Drop Communications Options
Various Multi-Drop settings are set to control timing and establish ID names.
1st Param DELAY, ID
2nd Param Text Value ID, numeric for Delay
Default ID is same as system name. Delay is 2 ticks

MEDI Media [T] Select Communications medium (technique)
Sets the type of communications media used by a task. Normally, the media is set one time at startup and never changed, although it could be done at runtime if required.
1st Param Phone, Radio, Keyboard, Direct, MultiDrop, or None
Default Determined by DAT file setting
Example Set Media direct ;connect handheld to this port

MENU Menu [L] MAIN or USER menu on F9
Main and User control which menu system is the primary menu system and which is the secondary. The Main menu is the default primary menu. Pressing F9 will display the primary menu and SHIFT-F9 will display the secondary menu. Off and On control the Auto-Menu feature. In Auto-Menu mode, the primary menu will automatically pop-up whenever the user is at a command prompt.
1st Param Off, On, Main, User
Default Main and off
Example Set Menu USER ; Let F9 call up user menu

MULT Multiple [T] Option for Multiple TSP statements per line
1st Param Cmd Separator Character (!,|, \, ~) or keyword OFF
Default Off
Example Set Multi | ; use vertical bar as cmd line separator

ONLI Online [T] Command processing controlled by comm connect status
Controls command file processing requirement for online. This command is effective only when being called from within a file.
1st Param Boolean
Default Off
Example Set Online on ; need DCD to continue processing from this point

PAGE Page Breaks [T] Page break control in reports
Set page break for DUMP, TYPE, and DIR type commands.
1st Param Boolean
Default Off
Example Set Page on

PALE Palette [L] Color Palette control for simultaneous 256 color images

Simultaneous display of multiple color graphic images which use non-standard palettes require some form of compromise to obtain a common color scheme. SCADAWARE converts images as they are loaded with the following options:

None	Use the palette from the most recently loaded image
Convert	Convert subsequent images to initial palette
Map	Take top 16 palette positions for std colors.
1st Param	None, Convert, Map
Default	None
Example	Set palette

PATH Path [S] Set default path for CMD files (Same as Drive)
(Same as the SET DRIVE OPTION above)

PAUS Pause [T] File processing after Negative response to PAUSE command
Controls file processing when the PAUSE command is answered negatively. SET PAUSE OFF allows file processing to continue when a No is answered. This lets the @yes() and @no() expressions to be used for program flow control. SET PAUSE ON causes file processing to terminate when a No is answered.

1st Param	Boolean
Default	On
Example	Set Pause off ; let file processing continue after N response

PLAC Places [T] Sets default decimal places for numeric values
Set the default decimal place count used by all value type channels in displays, reports, and data transfers. Each channel has its own parameter for decimal place precision, but if that parameter is less than 0 then the global setting set by this command is used.

1st Param	Numeric
Default	2
Example	Set Places 3 ; want high resolution on this display

PORT Port [T] Hardware Comm Port number for executing task
Sets the communications port for the current task to the specified number. Use this when linking a task to a port that has a number different from the task itself, as when using COM4 on task number 1.

1st Param	Numeric
Default	Determined by DAT file setting
Example	Set PORT 2 ; switch to phone line port

PRIN Printer [S] Set hardware or logical address for Printer Port
SCADAWARE uses standard DOS and BIOS printer routines for all printer output. To work around the poor handling of printer errors offered by the standard interface, SCADAWARE does internal checks which bypass the limited DOS and BIOS error signals. To do this properly, SCADAWARE must be told the printer port number being used for printouts. If none is specified, then no printout will be allowed. When an error is detected, the program attempts to reset the printer, and if this fails, the program automatically sets the Printer value to 0 to prevent further attempts to print until the problem is fixed and a SET PRINT xxx statement is re-executed.

1st Param	Numeric LPT port number
Default 1	0 (no printing enabled)
2nd Param	Numeric value for mask used to test
Default 2	90h (90 hex, 10010000 binary)
Example	Set Print 38h ; special bits for this strange printer

PTT Push-To-Talk [S] Set control output for Audio Output System
The audio sub-system can control an output for keying radios and PA systems.

1st Param	Tagname
Default	SO (no output channel)
Example	Set PTT O5 I output 5 controls the push to talk

PWAI Printer WAIT [S] Ticks to wait between lines sent to Printer
Set the number of process ticks the task printing a report to the printer should wait between each line. This command will slow down the rate at which lines are being sent to the printer.

If the lines are being sent too fast and the printer cannot keep up, the system can run slowly or hang completely. Each system tick is approx 50ms. The default PWAIT is 20, so the system will delay a task about one second between lines of output. During this second, other tasks will run uninterrupted. Adjust to provide best performance for each computer and printer system.
1st Param Numeric (number of 50 ms ticks)
Default 20
Example Set Pwait 10 ; wait 1/2 second between printed lines

RAW Raw Data SCAN Status and Output channels as physical position rather than logical true/false value. Normally closed switches which are currently closed will scan as 1 instead of 0.
1st Param Boolean
Default Off
example Set RAW ON ; send back physical switch and output positions

REPO Report [R] Automatic Report generation on data update timestamp
Turn on automatic report generation setting on any polls for each RTU
1st Param Boolean
Default Off
Example Set REport ON ; current rtu will not print on every update

RESE Reset [S] Specify External Channel used for RESET Push Button
Set channel to be used as a RESET push-button.
1st Param Tagname
Default S0 (no reset input)
Example Set Reset S5 ; Switch 5 is reset input

RFIL ResetFile [S] Command File Processing when abnormal condition RESETS
Controls execution of command files when a channel returns to normal. Setting OFF will not execute files. Setting ON will execute files on return to normal (reset). This setting affects all channels on all RTUs.
1st Param Boolean
Default Off
Example Set Rfile ON

RLOG Reset Logging [S] Logging of Channel Return to Normal Events
The data logger normally records only changes from normal to abnormal. This setting allows returns to normal to be similarly recorded.
1st Param Boolean
Default Off
Example Set RLOG On

RTU RTU File Lookup [S] Automatic search for RTU procedure or file when cmd not found
Controls automatic search for .RTU files in the event of an unrecognized internal command.
1st Param Boolean
Default Off
Example: Set RTU ON ; let system automatically search for RTU procedures

SILE Silence Input [S] Specify channel used for external Silence Push Button
1st Param Tagname
Default S0 (no silence input)
Example Set Silence S5 ; Switch 5 is silence input

STYL Style [L] Sets default Graphic Font Style
Determines graphic font style used in subsequent text output. Replaced by FONT command.

TAB Tab Setting [T] Specify default TAB spacing used in WRITE command
Writes can be done to integer tab space settings changed by this command..
1st Param Numeric
Default 8
Example Set Tab 10 ; columns every 10 character spaces

TICK Ticker[S] Turn system tick off and on

Set the one second "tick., tick" signal.
1st Param Boolean
Default On
Example Set Tick OFF ; stop the obnoxious ticking sound

TRAC Trace [T] Expression Evaluator Processing Trace Display
Determines if the equation evaluator shows intermediate steps during expression solving.
1st Param Boolean
Default OFF
Example Set TRACE ON

TRANS Transparent [L] Specify background color used for Transparent Graphic Object display
Controls transparent action of PAINT command by allowing the specified color to be the "clear" background which will not be transferred to the graphic screen.
1st Param Color
Default Black
Example SET TRANS WHITE

TRYS Trys [T] Number of Trys for each communications block transmission
Sets the number of tries to send a block and receive an acknowledgement for the task processing the command.
1st Param Numeric
Default 3
Example Set TRYS 5

UPFIle Update File [R] Set and Clear Update File Flag for each logical RTU
Controls whether or not a command to read the "UP" file gets queued up or not when a @ or @ENDS is processed in a DATA command. If UPFILE is off the command will get queued up and UPFILE will be set to on. The command will not get queued up again when a @ or @ENDS is processed in a DATA command unless UPFILE was turned off using this command.
1st Param Boolean
Default Off
Example Set Upfile off ; enable a new update request

VGA VGA [L] Default VGA and VESA video mode
Sets the default VGA or VESA screen mode used with the system switches from ext to graphics mode.
1st Param VGA Mode number
Default 7
Example Set VGA 37 ; select 640 x 480 x 256

Common VGA and VESA Mode Numbers

VGA MODE	CRT Horiz	CRT Vert	Max Colors	
0	320	200	4	
1	640	200	16	
2	320	200	16	
5	640	350	16	
7	640	480	16	(default)
8	320	200	256	
36	640	400	256	
37	640	480	256	(preferred)
38	800	600	16	
39	800	600	256	
40	1024	768	16	
41	1024	768	256	
42	1280	1024	16	
43	1280	1024	256	
51	320	200	32K	
52	320	200	64K	
53	320	200	16m	
54	640	480	32K	
55	640	480	64K	

56	640	480	16M
57	800	600	32K
58	800	600	64K
59	800	600	16M
60	1024	768	32K
61	1024	768	64K
62	1024	768	16M
63	1280	1024	32K
64	1280	1024	64K
65	1280	1024	16M

WAIT Wait [T] Seconds to wait for ACK during normal communications
Sets the seconds the unit should wait for an ACK response from another unit during callouts. This is a separate setting for each Task and is only used in computer to computer operations.
1st Param Numeric
Default 20
Example Set Wait 10 ; 10 second period to before retransmit

WARB Warble [S] Control internal PC speaker horn alarm
Controls use of local speaker as an alarm horn.
1st Param Boolean
Default On
Example Set WARB Off ; disable internal speaker alarm

SETDB Set Deadband for a Single Channel (2)

This command can be used to set the deadband for a single channel without having to go through the entire channel programming process. The command can be used for analog inputs, analog outputs, frequency inputs, counter inputs, AGA3 meters, timer channels, value channels, and totalizer channels. The format of the command is the keyword SETDB followed by a channel tag and possibly a value. If a value is specified, the deadband for the specified channel will be set. Otherwise, the current deadband setting will simply be displayed.

Note that the deadbands can also be set and read with the SCAN D option. This would be more suitable for sending a set of deadbands from one unit to another.

Examples:

```
SETDB A1 2    ; set deadband to 2
SETDB A3
```

Related: PROGRAM, CONFIG, SCAN D

SETHIgh Set High Setpoint for a Single Channel (2)

The high alarm setpoint for a single channel can be set with this command without going through the complete channel programming sequence. The channels that can have high setpoints are analog inputs, analog outputs, frequency inputs, counter inputs, AGA3 meters, timer channels, value channels, and totalizer channels. The channel number is followed by an optional new setpoint. The current value of the setpoint is displayed after the command is processed. If no new value is provided, the command will display only and leave the setpoint unchanged.

Note that the high setpoints can also be set and read with the SCAN H option. This would be more suitable for sending a set of setpoints from one unit to another.

Examples:

```
SETHI A1 12.3
SETHI C1 123.4
SETHI M2
```

Related: SETLOW, PROGRAM, CONFIG, SCAN H

SETLOw Set Low Setpoint for a Single Channel (2)

This is used to set the low alarm setpoint for a single channel. Low alarms can be set for analog inputs, analog outputs, frequency (rate) inputs, counter inputs, AGA3 meters, timer channels, value channels, and totalizer channels. See the SETHIGH command for syntax rules.

Note that the low setpoints can also be set and read with the SCAN L option. This would be more suitable for sending a set of setpoints from one unit to another.

Examples:

```
SETLOW M2 345.6
SETLOW A3
```

Related: SETHI, PROGRAM, CONFIG

SETWAIT Set Alarm Delay Period for a Single Channel (2)

This command can be used to set the alarm delay period for any type of channel without having to go through the entire channel programming process. The format of the command is the keyword SETWAIT followed by a channel tag and possibly a value. If a value is specified, the alarm delay for the specified channel will be set. Otherwise, the current alarm delay will simply be displayed.

Note that the alarm delays can also be set and read with the SCAN W option. This would be more suitable for sending a set of alarm delays from one unit to another.

Examples:

```
SETWAIT V1 10     ; set alarm wait to 10 seconds
SETWAIT A3
```

Related: PROGRAM, CONFIG, SCAN W

SHELL Shell to DOS (2)

This command is exactly like the EXEC command discussed above. It is used to be compatible with other DOS programs that use the Unix term "SHELL" rather than the DOS term "EXEC".

Related: EXEC

SLEEP Suspend Task Operation for a Number of Seconds (2)

It is sometimes desirable to have a task suspend operation for a while, as when waiting for equipment to startup. The sleep command specifies a number of seconds that a task will stop execution. The current task is suspended and will not be eligible for processing until the specified number of seconds has elapsed. This is most useful for programmed pauses in command file execution as when waiting for an output action to take affect.

Example:

```
Sele RTU1
Dial
Wait 30 conn
if @online(1)
 msg RTU is online
 block CALC t1 = 30
 msg Pulse command sent
 sleep 30  ; wait for ESD to occur
 msg getting download
 block read download
endif
```

Related: WAIT

STATus Display Status of Link, Comm Port, and Tasks (1)

The status of all dynamic system variables can be monitored with the STATUS command. The comm port, comm link, task status, and file status can be displayed on a real-time screen. This allows greatly simplified diagnostics because the local CRT can be used to monitor all of the internal workings of the system. This command is used mostly for diagnostics and system setup. The options are as follows:

C - Communications Channel Parameters
L - Communications Link Parameters
T - Task variables and states
F - Shows status of open files

No options will produce a display of C and L combined.

Examples:

```
STATUS
STAT T
STAT L
STAT C
STAT F
```

STOP Stop Command File Execution for a Task (1)

Execution of a command file normally continues to the end of file before termination. If it is desirable to stop a file in the middle somewhere, the STOP command can be used. This is very similar to the RETURN command because processing of the current file is terminated. The difference is that the RETURN command will only stop processing the current command file. The STOP command will stop processing the current command file *and any other files* currently being processed by the same task. In other words, if the processing of the file was started from within another file then both files will be terminated. The STOP command will always return the command processor to the command mode.

Example:

```
if (v1 > 10)
  stop
endif
```

Related: RETURN, GOTO, REREAD

STORE Write Configuration Data to Standard Output (2)

It is necessary to save the various setup information for each system to some sort of storage unit for later recall when powering up or resetting the computer. The STORE command is used to generate command lines that can later be read by the system (or another system) to re-program it to the settings currently in effect. The things that can be stored include channel setups, link setups, and callout group assignments.

The STORE command sends the lines it generates to the standard output device in effect at the time the command is executed. Normally, this is the console or terminal in use by the current task. However, it is possible (using the FILE command) to send output to a disk file or device such as a printer. In this manner, the text lines can be saved for later use or printed out for reference. The SAVE command does the file open and close automatically, although the user can also do this for special cases.

Channel configuration is read and written in a text format that divides each channel into its internal layers, with each layer on a single line. Therefore, the number of lines required to describe a channel depends on the type of channel.

Link information is generated as one line for each link that is currently being used. Unused links do not generate lines. In-use is determined by the link's task number being > 0.

Callout Group information is written for each group that has any links associated with it. Unused groups are not written out.

The allowable keywords that can be used with this command are CHANNELS, LINKS, GROUPS, and ALL. These keywords will designate exactly what data will be stored. If no keyword is given the keyword CHANNEL is used. If the keyword ALL is used, all channels, links, and groups will be stored. Only the first 2 letters of any keyword are significant.

Examples:

```
STORE ALL    ; Store everything
STORE GROUPS
STORE LINKS
STORE CHAN   ; Store all channels
STORE      ; Store all channels
```

Related: SAVE

STUFF Insert Characters Into Another Task's Input Buffer (2)

This command is used to jam characters into the input buffer of a task. Similar to the FORCE command, this can be used to send commands directly to another task. The difference is that text specified on the STUFF line is sent to the task's comm input buffer, which is checked during execution of most commands. The FORCE command places text in a special queue that is read when the task is not processing any other commands.

The STUFF command can be used to send special codes similar to those used by the ECHO command. This allows escapes and other codes to be sent to tasks that may be looping waiting for operator input. If the force command was used, this would place a message that would not be read because the task may be stuck in the middle of a command. STUFF gets characters directly into the input buffer as if they had been received over the normal communications line.

The following abbreviations are allowed to be used in place of actual ASCII codes:

ES	Escape
CR	Carriage return
SP	Space
CA	Control C (CANCEL)
BS	Backspace

This command is mostly useful for diagnostics and troubleshooting.

Example:

```
STUFF 1 ES
STUFF 1 SP
STUFF 1 ES CAN ES CAN
```

Related: FORCE

TAB Position Text Output to Specific Column (1)

TAB is used with the WRITE function to align text output to a specific column. The next logical tab stop is used, which defaults to every 8 characters. The SET TAB command can be used to change the spacing for each task.

Tab can also be used to position the text output stream to a specific column by providing the numeric value of the column after the TAB command.

```
file open report.txt  ; open a new output file
set delim none        ; use nothing as delimiter, use TAB spacing instead
writeln "SAMPLE REPORT"
write A1
tab
write A2
tab
writeln a3   ; finish the line
file close

file append report.txt    ; re-open the file and add to its end
writeln
writeln "MORE DATA"
write a4
tab 20
write a5
tab 40
writeln a6
file close
```

Examples:

```
TAB          ; Move to next tab
TAB 50 ; move to column 50
```

Related: SET TAB, SET DELIM, WRITE

TAP Passive TSP Mult-Drop Communications Control (2)

Multi-Drop systems that have more than one Host on the same line can eliminate redundant transmissions with the TAP option. The feature lets a unit TAP into the multi-drop line for purposes of processing DATA lines only. The unit listening to the line processes the DATA line but does not send an ACK or any other response. This is similar to the normal BROADCAST option in the MultiDrop except that only DATA lines will be processed. BROADCAST allows any line to be processed, and this is not normally desired for most lines.

The TAP feature can also be used to set up redundant Host computers by letting one unit actively control a network, while the other unit taps into the comm line and monitors all data transfers. This lets the passive unit be updated without any direct connection to any other units.

A typical application will have three or more units on a common line where each unit shows its own data (as an RTU) and data from the other units (as a Host). Whenever one unit sends DATA lines to another one, all units can listen in to the transmission and process the DATA line just as if it had been sent directly to it. Therefore, all units can be updated at the same time with DATA lines, but only the addressed unit will process other command such as file reads, calcs, etc.

To use this feature, the affected DAT files must have a statement that specifies the number of RTU's that will be monitored. The syntax of the line is:

TAP n

where n is the number to use. Example: TAP 5 sets up a list of 5 RTU names that will be matched on each line.

The RTU names that can be monitored are entered with one or more command lines that are normally processed during startup. These lines add the RTU names to the acceptable list. For example:

```
TAP Clear  ; Clear out any  exiting TAPS
TAP ADD  WC354  HOST EC311 WD61
```

The status of the TAP can be checked at any time with TAP DUMP.

If a multi-drop task processes a data line which is being sent to a unit on its TAP list, it will process the line just as if the line had been sent to its own computer. However, no ACK will be generated by the TAP system. Tasks being Watched will indicate reception and processing of appropriate TAP acceptable lines just as if they had been directed to the affected computer.

Refer to the System Design Manual for more information on multi-drop communications and the TAP processing option.

TASK Control Task Execution and Settings (2)

This command is the primary means of controlling the status of all tasks defined in the RTU.DAT file. The command uses an optional task identifier (number or nickname) followed by a keyword. Current keywords are:

START	Start a task running with a fresh start.
STOP	Make a task dormant.
WAKE	Return a sleeping task to a run or wait status.
SUSPEND	Put a task to sleep for the specified seconds.
DELAY	Specify ticks to delay between executions.
PRIORITY	Specify sequential ticks a task can run.
DUMP	Display task status.

Tasks have "nicknames' or ID's assigned to them when they are defined in the DAT file. Typical task ID's are as follows:

UTIL	Utility task.
CALC	Calculator task.
SCAN	Alarm Scanner task.
SEC	One second task.
LOCAL	Local CRT task.
DRIVER	I/O driver task.

Starting a task resets the program counter and stack space for the task and sets a task level flag indicating that the task is about to run for the first time. At the next available chance, the task will start running with a "fresh" start. Any previous commands, file status, or other dynamic conditions are all reset with the start command. A task that is locked up for some reason can usually be re-started with this command.

Stopping a task halts it in its present state and sets the task status to dormant. The only way to get the task active again is to use the START command above. Stopped tasks use no processor time or other CPU resources. NOTE, stopping task 0 would cause the multi-tasker to crash so this command can not be used to stop task 0.

Suspending a task for the specified seconds does not adjust the task's default delay or priority setting. The task is simply put to sleep for a period of time, after which it picks up just where it left off.

Because each system has differing processor speeds, task assignments, and other workloads, it is desirable to have control over how much processing time each task has. The TASK command allows the critical timing of each task to be individually controlled at run time.

The priority of each task determines how many sequential clock ticks the task will be allowed to execute before it is suspended. If a task completes before its allotment is used up, the next available task gets an early chance to run. Adjust this number up to allow a task more processing time out of each overall cycle.

The delay of each task is the ticks that the task will pause between each completion of its full execution. For example, an I/O driver that does not need to run constantly can be assigned a high (say 18) delay number so that after completing a run it will wait a at least a second before running again. The priority of the task can be also set high so that when it does get a chance to run it will have a better chance of running to completion all at one time.

Examples:

```
TASK UTIL START  ; start the utility task
TASK scan delay 1
TASK 0 priority 4 ; speed up screen displays
TASK second stop
TASK util delay 2
TASK calc wake
TASK 2 suspend 10
```

*TERMINAL Put Program in Human Interface Mode (0)

This is a very special command because it is immune from the normal error checking requirements that are in effect when the program is in RTU mode. After the command is entered, beginning with the start character ^A (control-A), the program will drop into the terminal mode. This allows entry of commands without the checksum requirements normally associated with RTU data transfers. This command has no effect on Task 0 because Task 0 is always in terminal mode rather than RTU mode.

Remember, the start character ^A must be specified and the command must be completely spelled out in upper case.

Example: ^A*TERMINAL

TIME Set DOS Time of Day (2)

This sets or displays the DOS time as seen by the RTU and other programs. The same operation notes that apply to the DATE command discussed earlier apply to the TIME command. The time string must be a full time specification in 24 hour "military time" format.

Entering the command TIME with no options causes the current value to be displayed. Entering a properly formatted time as a parameter will cause the time to assume that value. As a second parameter, a properly formatted date will cause the date to assume that value. This allows setting of both the time and date in a single command line.

Example:

```
TIME                                ; display current time
TIME 02:32:00                 ; set time
TIME 02:32:00 01/01/88              ; set time and date
```

Related: DATE, EXEC

TRANSFER Copy Values from One Group of Channels to Another (2)

This command provides a quick way to copy the values from one group of channels to another group of channels. To use this command two channel ranges must be given. The first range represents the channels to copy from and the second range represents the channels to copy to. Single channel names can be given instead of channel ranges if the value for only one channel needs to be transferred. Channel values can be transferred between different RTUs on a system by specifying the RTU names in the channel ranges using "dot" notation. Dot notation consists of two parts separated by a period (or "dot"). The first part is the RTU name and the second part is the channel range. The current RTU is assumed if no RTU name is specified in a channel range.

Example:

```
TRANSFER S1:S5 O1:O5                ; transfer inputs to outputs
TRANSFER RTUA.S4:S8 RTUB.S10:S14    ; transfer between two RTUs

; sample Daily file
transfer q1:q4 v11:v14 ; move totals to yesterday's value
calc q1:q4 = 0 ; start totalizing this day
```

Related: CALC, XFER

TYPE Display a DOS Text File or TSP Procedure (1)

This is similar to the DOS type command, except that the output is paged into screen fulls with a timed pause at the end of each page.

The required parameter is the name of the DOS text file to be viewed. If no file extension is provided, a default of RTU is assumed.

Example: TYPE AUTOEXEC.BAT

Related: SET CONT

UNHOLD Remove a Channel's HOLD Status (2)

A channel's HOLD status determines if any low level value conversions are made by the channel scanner task. The HOLD command places a channel on hold, preventing these conversions. This is normally used during calibration times. The UNHOLD command is used to clear the HOLD status for any number of channels or channel ranges.

Examples:

```
UNHOLD M1
UNHOLD m1:m3
UNHOLD m1:m4 q1:a7 a1:a3 C5
```

Related: CHANGE, HOLD

USER Display User Menu (1)

This command is used to display a customized user menu. The user menu is a "pop-up" type menu system that allows for quick selection of preprogrammed actions. The user menu can be modified to perform custom functions unique to each location. This is done with the use of menu files. Before a user menu can be displayed it must first be loaded into memory. See the MENU command for how to load different user menu files into memory. If no menu file is loaded in memory when the USER command is processed, the program will automatically look for a file called RTU.MNU, load it into memory if it exists, and then display the menu.

Example: USER ; display currently loaded user menu

Related: MENU

VER Get SCADAWARE Information (1)

This will produce a program version date as well as information about the memory allocated by the program.

Examples:

```
VER
```

WAIT Delay Some Number of Seconds for An Event (2)

Within command files it is often necessary to hold up processing until certain events occur. This command allows the wait to occur for an approximate timeout period (expressed in seconds). If the event occurs during the timeout period, the wait is canceled and processing continues with the next line in the file. If the event does not occur, file processing is suspended until the timeout period expires. Processing will then continue with the next line in the file.

Using the SET ONLINE command can be used just after the WAIT command to have the program wait for a connection and then continue processing only if the unit is online. The wait function gives time for the phone or radio connection to be made. Without using the WAIT command, file processing would not allow time for the connection to be completed.

If no wait time is provided, a default value is assumed. The default value used is the number of seconds that the current link will wait for a connection before giving up. The local task does not assume a default timeout period because it has no link associated with it. The time can appear anywhere on the command line along with any number of event keywords. If more than one keyword is specified, the wait is terminated when any of the events take place.

The keywords currently supported are:

CONNECT <- Detection of Online status (carrier detect)
DISCONNECT <- Loss of connection status
KEY <- Keyboard or serial line input

Examples:

```
WAIT Key 10     <- Wait up to 10 seconds for a keypress
WAIT Conn 60    <- Wait 1 minute to make the connection
WAIT Key Conn 20 <- Wait 20 secs for keypress or connection


; Sample RTU dialout procedure
sele RTU1
dial
wait 32 conn ; wait up to 32 seconds for a connection
if @online(1)

   --- do something interesting ---
endif
```

Related: DIAL, SET ONLINE, PURGE

WATCH Monitor Input and Output Characters for Another Task (2)

This is a command that will allow a console to monitor the input and output to any other task. This is handy when trying to figure out what another task is doing, and to monitor characters going to and from a communications line. The Watch command is the primary diagnostic for monitoring the progress of any task other than task 0. With the Watch facility, the local user can see every input and response for any other RTU type task. This allows real-time debugging and troubleshooting of the multi-tasking system because the local user can "see" what is happening to another task on the actual equipment processing the programs. No additional equipment is necessary to monitor serial or network communications. The Tasks to be monitored can be referenced by number or name as in any other task related command.

Two parameters can be used to start or stop monitoring a task. The first one is the task to be affected. The second is the word OFF or ON to turn the watch function off and on. If only the task parameter is given the option ON will be assumed. No parameters will provide a listing of the tasks that your task is watching. There is also an option to cancel all watches in progress. Using WATCH OFF (with no task specification) will cause all current watches to be cleared. This is offered as a simple way to remove all diagnostic traces without having to determine which ones are in effect.

CAUTION: Because characters are displayed as they are encountered, the input and output may be mixed making the resulting display difficult to read.

Brackets are used to surround characters that come into a task via command file or a force command. These brackets look like { or [depending on the source of the message.

Additional trace information is also displayed during the WATCH period. This includes messages indicating communications connect and disconnect, as well as block format transmission signals.

An advanced form of the watch command allows the user to specify which message sources are to be monitored. Normally, text from the comm line, command files, and inter-task message queue are displayed along with the progress reports for the monitored tasks. A special option is available when the watch is started that specifies which of the three input sources will be displayed. The option is specified as a single number that is evaluated by the watch processor as a binary bit field. Each bit controls the monitoring of a specific input source as follows:

0 Command Line Input
1 Command Files and Procedures
2 Inter-Task messages

A short table of all combinations is as follows:

Value	COMM	FILE	MESSAGES
1	Y	N	N
2	N	Y	N
3	Y	Y	N
4	N	N	Y
5	Y	N	Y
6	N	Y	Y
7	Y	Y	Y

The sources to be monitored must have their bit set to one in the number that is supplied on the WATCH command line. For example, to monitor only the command lines and inter-task messages we would want to set bits 0 and 2. Bit 0 in binary is worth 1, and bit 2 is worth 4. Therefore, the value of 5 *(1 + 4)* is specified as the mask number as follows:

WATCH 1 ON 5 ; task 1 for com port and messages only

To watch Task 2 for comm line only, we would use

Watch 2 on 1 ; Task 2 for comm line only

If no value is provided the system will default to 7 which will cause the Watch command to show all text messages from any source.

Examples:

```
WATCH 1 ON  ; start monitoring task 1
WATCH 2     ; start monitoring task 2
WATCH Util  ; start monitoring task named Util
WATCH 2 OFF ; stop monitoring task 2
WATCH OFF   ; stop monitoring all tasks
```

WIN Graphic Window Control

Graphic window assignments are done in separate windows, each of which contains a graphic element such as a bar graph or data trend. WIN is used to setup and control these windows in TSP command files. Refer to the Graphic System Documentation for more information.

WRITE - WRITELN Output Delimited Text to Screen or File (1)

The WRITE and WRITELN commands simplify text output for special applications such as database or spreadsheet interfaces. This command allows any channel or variable data to be sent to the standard output device (screen, port, or file) with "delimiters" automatically inserted between each value. The standard delimiter is a comma, but this can be changed with the SET DELIMITER command.

Write and Writeln can be used to output any TSP data. Write sends the data without a line-end (carriage return/line feed). Writeln terminates the line after the last field. Multiple Write lines can be used to output large amounts of data onto one single line as long as the line is eventually terminated with a WRITELN. The most common usage will be to transfer Database information to a text file so that it can be read into another program such as DBASE or LOTUS. The following example shows the sequence necessary to send data base fields to a file:

```
File open OUTFILE.TXT  ; open the output stream
db sele mydata.sdb     ; open the data base
:loop
write  sdb_01 sdb_02 ; first 2 fields with commas
writeln sdb_03 sdb_04 ; last 2 fields and line end
db skip
if @dberr =0
  goto loop
endif
db close
write File generated at $T on $D
file close
msg Data Output Complete
```

The delimiter is normally a comma, but it can be changed with a SET command. The SET DELIM options are:

```
SET DELIM COMMA     ; use a comma
SET DELIM SPACE     ; use a single space
SET DELIM NONE      ; join values together
SET DELIM ascii_value ; pick up ascii numeric value
```

Write can also be used to generate custom reports to any valid DOS filename, including the logical printer (PRN). The following is a small report that is sent directly to the printer:

```
file open PRN
writeln "This is a Sample Report for" $R
writeln "Total Gas Production = ", q1+q2'####.#," MMCF"
writeln "Line Pressure     = ", a5'#####," Psi"
file close
```

XFER Copy Values From One Group of Channels to Another (2)

This command is identical to the TRANSFER command described above. Refer to the TRANSFER command for more information about this command.

File: RTU1020.WP AMZ Printed: December 9, 1994