TOTAL ENGINEERING SERVICES TEAM, INC.
(TEST Inc.)

A Weatherford Enterra Company

SCADAWARE™

Trinmar Limited
SOLDADO SCADA SYSTEM

TEST Project 0600-QF4

Original Document Prepared by:
Arthur M. Zatarain, PE
TEST Inc.

First Issued: August, 1996
Revised: Aug 22, 1996

CONTENTS

INTRODUCTION

This document describes particulars of the TEST SCADA system installed for Trinmar Limited in the Fall of 1995. This is a technical summary of the system's unique configuration and features. It contains configuration information for the programmers and technicians who use, maintain, and configure the system.

BASIC TERMINOLOGY

The following SCADA terms are listed in logical order, starting at the top of the entire system. Reading the list in this order will acquaint a novice user with the basics of this SCADA system.

1. **SCADA System** - Supervisory Control and Data Acquisition - the total integration of computers, software, communications, and process sensing hardware to perform industrial monitoring and control functions. SCADA is not a thing - it is a concept.

2. **Host Computer** - The main computer in a SCADA system which collects and stores information from the field locations. It is possible to have more than one host in a single system, each of which performs different host related functions. The Trinmar system has hosts at onshore and offshore locations. A host is sometimes called a Master Terminal Unit (MTU) in older systems.

3. **Marine Base Host** - The central computer of the Trinmar SCADA system. It connects to the offshore units via radio as well as the onshore network computers. This computer is dedicated to this task.

4. **Viewnode** - A PC connected to the Host computer via PC network hardware. The Viewnode can display data for remote locations. In the Trinmar System, one viewnode is provided for each production field. These computers are not essential to system operation.

5. **Communications Link** - A combination of physical communications hardware and program functions which permit inter-computer data transfer. In the Trinmar system, these are both Radio paths and network paths. The network links are between office computers only. The Radio paths, from the Host computer to the outlying remote stations, are done with UHF (450mhz) class radios. The radio system operates in a single channel, half-duplex, multi-drop mode in which all units hear all transmissions. This configuration simplifies multiple-host and network operations.

6. **Polling** - Data requests from one unit to another. In TEST's system, any smart unit (Host or RTU) can initiate polling actions. Trinmar's system normally polls from the Host, where the Host requests data from the RTU, or sends configuration data to the RTU. All polling is done with the TSP protocol (explained below).

7. **Remote Terminal Unit (RTU)** - A field located computer which connects to local sensing and control devices. The RTU sends this information (normally via radio) to a host computer. Trinmar's system uses Type 1200 Smart RTU's, and smaller Type 2200 SCADA Node RTU's.

8. **Smart RTU** - TEST's Type 1200 PC based RTU which can perform local

displays, calculations, and special functions including gas flow measurement. The Smart RTU is capable of stand-alone operation, and can also act as a Host computer if required. The Type 1200 RTU is used at most locations in the Trinmar system.

9. **HOST/RTU** - An enhanced Type 1200 Smart RTU fitted with a faster processor, additional memory, and a hard disk so that it can serve as both an RTU and a Host. These units monitor data from the local platform as well as data for remote locations.

10. **SCADA Node (Dumb RTU)** - TEST's Type 2200 RTU which sends data to the Host, but cannot perform any local operations on its own. These are used at the Riser Platforms.

11. **SCADAWARE** - TEST's software which operates both the Smart RTU and the host computers. It is not used on the T2200 SCADA Node RTU itself.

12. **Task** - A processing unit within the SCADAWARE program. Tasks perform specific functions such as radio communications, local data displays, alarm detection, and I/O point scanning. Each SCADAWARE computer has 3-10 tasks, depending on the requirements of that particular system.

13. **Data Channel** - A data point which contains several attributes such as description, tag name, value, and alarm settings. SCADAWARE supports nine channel types: **S**tatus In, Status **O**ut, **V**alue, **A**nalog In/Out, **C**ounter, **T**imer, **M**eter, **T**otalizer, and **F**unction. The highlighted letter in each title is combined with a number to uniquely designate a channel on each RTU. For example, A12 represents Analog point 12.

14. **Local Channel** - A data point which originates on the computer on which it is processed. Points connected to an RTU are local to that computer.

15. **Remote Channel** - A data point which originates at another location in the system. Data sent from an RTU is a remote point on the Host.

16. **Status Data** - Off-On signals which indicate equipment or well status. Sometimes called Digital In and Out on other systems.

17. **Analog Data** - Value type signals which represent a numeric quantity such as pressure, temperature, or level.

18. **Counter Data** - Event signals which represent totalized transitions of an off-on signal. These are typically used for liquid meter inputs, where the pulses come from the meter device each time a unit of measure is metered. Sometimes called accumulator on other systems.

19. **Logical RTU** - A named, imaginary unit that contains a specific collection of data channels. Normally, a logical RTU corresponds to a physical RTU in the field. Each logical RTU has an 8 character (max) identifier, such as PL21 for platform 21. The RTU name must be unique within the entire system. All data channels on that RTU must have unique names within the scope of that RTU. Each RTU also has certain attributes such as the data validity timestamp, and the default communications link.

20. **Current RTU** - The default logical RTU used for SCADAWARE and TSP procedures. Each program task has a current RTU. Unless some form of override is applied, the data for the task's current RTU will be used in all operations. The current RTU is changed in procedures with the SELECT statement. In manual mode, it is selected with menu hits.

21. **Timestamp** - A time and date associated with a data update from a specific logical RTU. Besides providing a reference point for data validity, setting of the timestamp also triggers certain automatic update functions within SCADAWARE.

22. **Ambiguous Channel Tag** - An identifier for a data channel that is unique on that logical RTU, but may be ambiguous within the scope of the entire system. The ID can be a physical reference to the overall list, such as M3, which means Meter 3 on that RTU. It can also be an alpha tag name, such as BattVolt, which is an 8 character (max) code name that must also be unique on that RTU. The reason these tags are ambiguous is that unless the particular RTU is specified, it is possible for a tag reference to appear on more than one RTU within a system. SCADAWARE has mechanisms to deal with the ambiguous nature of tag names.

23. **Unambiguous Channel Tag** - A tag name combined with an RTU name that forms a *totally unambiguous* identifier within the entire system. A channel tag can be forced to apply to a specific RTU by preceding the tag with the RTU name and a dot. For example, BS16.A12 designates analog point 12 on Block Station 16. This method overrides the current RTU for that task. Note that is possible (and likely) for different RTU's within a single system to have points with the same name. The only restriction on data channel

names is that they must be unique within that particular logical RTU.

24. **Communications Protocol** - Protocols are the languages of computer communications. They provide a coordinated method of exchanging information on a variety of transmission media. Many protocols are available for industrial applications, a subset of which apply to SCADA systems. The Trinmar system uses a method called TSP, described below.

25. **TEST SCADA Protocol (TSP)** - A text-based communications protocol designed by TEST specifically for SCADA applications. It is based on plain text commands and messages, and forms the programming language used to operate the various computers which make up the overall system. TSP supports all SCADAWARE data types, using both integer and floating point format. It also provides specialized services such as logical RTU names, data timestamping, remote command execution, and data broadcasting, none of which are supported in other protocols. TSP also provides open architecture access to other standard protocols (such as Allen Bradley and Modbus) for connection to outside systems.

26. **MODBUS Protocol** - An "open architecture" protocol which permits low-level data transfers among computers from various manufacturers. Unlike TSP, Modbus itself only supports bit oriented (off/on) data, and integer (0-64K) whole number data. Other types of data, such as floating point information for analog data, is not native to Modbus. Values such as flow rates must be exchanged with special, non-standard methods, effectively making Modbus proprietary for each vendor when used at that level of complexity.

27. **TSP Procedure** - A series of TSP statements which are stored in a single named DOS file. The procedure is executed with the READ and GOSUB commands, which causes the affected task to process the statements contained in the file. Due to DOS file name restrictions, each procedure name is limited to 8 characters. The DOS file type can be specified if desired, and defaults to RTU. Processing is very similar to a DOS batch file, except that the statements are executed by SCADAWARE, not DOS. TSP procedures can be written for a single specific purpose, or they can be generalized to perform slightly different functions which can be controlled at runtime. Procedures are read from disk, line by line, as they are interpreted by the software.

28. **TSP Library** - Multiple TSP procedures can be grouped into a single library file, which is simply a DOS text file with a default file type of LIB. The

procedures within the library each have a unique name which can be up to 16 characters long. Procedures which are similar in function (such as update procedures for each RTU) are often placed into a single file to simplify maintenance, and to speed runtime operation. Unlike TSP procedures which are stored in their own files, a Library procedure is loaded once during system startup and then executed from memory.

29. **LINK Procedure** - A special TSP procedure associated with a particular communications link. The download mechanisms for each RTU are controlled by these files, which are contained in the LINK.LIB library file. The Link procedures are named LINKxx, where xx corresponds to the link number, as in LINK21 for Link number 21.

SYSTEM OVERVIEW

The primary function of the Trinmar SCADA system is to monitor production information on offshore platforms, onshore metering stations, and storage facilities. It consists of TEST Inc. Remote Terminal Units (RTUs) located at approximately 25 field locations. All remote units communicate via radio to a central Host computer at the Marine Base. This central computer collects and displays alarm, status, and production information from the remote units. It makes the data available to several other computers via a PC local area network (LAN).

An additional, separate PC at the Tank Farm serves as a backup Host computer. This computer is not on the overall Trinmar network, and therefore cannot receive the SCADA broadcasts from the Host computer. The Tank Farm receives updates by "tapping into" routine radio communications to and from the main Host computer. If required, the Tank Farm can be designated as the primary Host by taking over the routine polling normally done by the Marine Base Host.

Data reports from the Marine Base computer are also made available to the general Trinmar PC network. These reports can be provided in text output format suitable for direct viewing, or as spreadsheet import files for use in other software programs. The content of these reports is controlled by TSP procedure files on the main computer.

MARINE BASE HOST

The Marine Base Host computer acts as the central hub of the entire system. It performs all of the routine polling, data storage, and network operation required to update system data for all display stations except the Tank Farm. The Host is a standard IBM compatible personal computer operating DOS and Novell's Personal Netware. TEST's SCADAWARE program provides the services of data display, storage, polling, and diagnostics.

In addition to operating as the overall system coordinator, the Marine Base Host also serves as a viewnode for the Main Field. This was done to eliminate a dedicated host, thereby reducing the number of required computers. This double-duty has very little effect on the overall performance of the host system.

The Host polls the remote locations via radio. If the radio and modem were physically close to the Host PC, they could connect directly to the serial port of the computer. However, the radio is located at the main communications tower at the Marine Terminal, several hundred feet away. In order to transmit and receive the serial data signals, the Host PC and Radio modem are fitted with special line drivers. These drivers amplify the serial data signal running to and from the office to the antenna tower shack. These line drivers convert RS-232 to the higher performance RS-422 standard, which is commonly used for long distance transmission over wire.

DATA UPDATE PROCESSES

The host computer uses local timers to trigger a routine polling operation. The interval is user adjustable, normally set to 20 minutes. The poll operation executes on the RADIO task of the Host PC to contact each RTU and requests a data update. As the host completes the update from each RTU, it stores the data on the Host's data base. It also broadcasts the update over the local area network to update the viewnodes.

A database is configured on the Host for each separate RTU. Each file contains specific data channel information (fields) stored in records associated with date and time of day. The actual data storage takes place as part of the UPDATE process which occurs when each RTU is timestamped at the host.

The timer which controls the periodic poll is tagged POLLRTUS, and runs at the Host on a simulated RTU called BASE. When the timer reaches 0, it initiates execution of a procedure with the same name. This procedure uses the built-in

mechanisms of SCADAWARE to initiate a poll to all RTU's. The procedure code is:

```
; This runs when the poll timer runs down
proc pollrtus          ; automatic poll of all RTUs
  > rflag 115 on       ; indicate quick download from each RTU
  sele base        ; BASE RTU is default
  if no_host = 0       ; if we are still acting as the main host
    > Poll now           ; initiate poll to all the RTUs
endif
```

This simple procedure does all that is required to kick off a poll to every RTU which has a valid comm link. Note the use of the > symbol which causes the associated statement to be executed once for each RTU. The rflag statement sets a control bit for each RTU indicating a quick download is required. The POLL NOW statement sets the default link for each RTU to the ACTIVE state. This causes the radio task to perform a poll of each RTU until its link is cleared, which occurs when a download is complete.

The RADIO task uses the comm link list to determine which RTUs require an update. The actual process required to perform each download is contained in LINKxx procedures, where xx is replaced with the link number. Thus, the code for comm link 10 is contained in a procedure named LINK10. This procedure would be have the unique download requirements for the RTU associated with Link number 10. All of the link procedures are contained in a single TSP library file named LINK.LIB.

A typical link file for a smart RTU is as follows:

```
proc link6                  ; call to PL14
  sele pl14
  set upfile off
  set online on
  if @rflag(108) = on ; need to send data to the RTU before getting update
    scan a1:a11 l           ; send setpoints to the RTU
    scan a12:a22 l
    scan a1:a11 h
    scan a12:a22 h
    scan a1:a11 w
    scan a12:a22 w
    scan a1:a11 d
    scan a12:a22 d
    scan m1:m2 l m1:m2 h
    scan m1:m2 w m1:m2 d
    scan v1:v6 e            ; send gas meter parameters to the RTU
    rflag 108 off
  endif
  gosub start_dl 1    ; long analog scan
  set online off
```

This typical procedure is for Link 10, which happens to apply to Platform 14. It begins with a check of rflag 108 which would be true if the RTU required a data update from the Host. As this is rarely the case, the code will normally jump over that section and proceed with the line GOSUB START_DL 1. This calls common code which proceeds to get the data from the RTU in a generic manner. The code for this procedure is:

```
; Generic Download initiate for Smart unit. Rflag 115 controls short or
long
; Base Output point q_scan conrols amount of data downloaded.
; Base output point dl_style determines if remote sends short or long data
list

proc start_dl  ; start download from smart rtu.
               ; Param 1 controls second scan line for analogs if needed
  if base.q_scan = 1   ; if in quick scan mode
    set dumb on
;; Use multi scan lines to insure none are too long
    block scan $R.s1:s32 r m1:m8 e q1:q8 e c1:c8 E
    if (@hours(0) =7)  ;if morning report
      block scan $R.v1:v24 E
    endif
    block scan $R.a1:a16 E@ f1:f6 E   ; @ on this line fow low Ain count
units
    if $1 = 1
      block scan $r.a16:a30 E      ; need 2 lines.
   endif

; IMPORTANT! You need the BLOCK BYE at the end of this process to
; clear the smart RTU. It is waiting for an ACK from us, and it we
; dont send something, it will resend its last reply. This messes up
; our next xmit when moving on to another RTU. SO KEEP THE BYE COMMAND!

    block bye   ; need to ack their final xmit because of dumb mode
    bye   ; let the other end time out and do its own bye
    return
  if base.dl_style = 0
    block read download 1  ; get quick update from the RTU
    return
  endif
  if @rflag(115) = on
    block read download 1  ; get quick update from the RTU
  else
    block read download 0  ; get complete update from the RTU
  endif
```

The generic download will check rflag 115 to determine if a short or long download style is desired. It also uses a command line parameter (set when called by the Link procedure) to determine how many channels must be downloaded from this particular RTU.

ROUTINE POLLING CONTROL

The BASE PC polls all remote units on a timed basis. The time interval can be set with a menu entry on the BASE PC POLLING CONTROL MENU.

RTU's can also be manually polled by selecting that RTU and pressing F5. The graphic screens also have a POLL function to initiate callouts to the current RTU.

A poll summary screen can be obtained with Shft-F5. This causes the form UPTIME.FRM to be displayed. The form shows the last update time for all RTUs on a single screen.

MARINE BASE NETWORK SETUP

The Main Host unit is set up as a network server which can be read from and written to by the viewnode computers. The Network Oprating System (NOS) on all the SCADA computers is Personal Netware by Novell, running on top of Microsoft DOS. The viewnodes are connected via Ethernet twisted pair network (10Base-T) to the Trinmar network. The Trinmar network, which uses Novell Netware, is compatible with the network software provided on the Marine Terminal computers of the SCADA system.

Accessing the SCADA server computer requires that drive assignments take place on the viewnodes. As part of the normal viewnode startup, drive path C:\ on the host is set to be drive D: on the viewnodes. This relationship is established on the viewnodes with statements in their AUTOEXEC and START.BAT files which are processed during initial system boot. Additional statements are also included in the SCADAWARE startup batch file.

The Marine Base PC is also connected to the Trinmar Netware Network. A Trinmar server disk location (defined by Trinmar's network administrator) appears as drive F: on the SCADA computers. Periodic reports are placed on drive F: by the viewnodes for access from other Trinmar computers outside the SCADA network. Access to the server disk location is controlled by Trinmar's computer support group and is outside the SCADA system setup.

The Network Names (for Netbios reference) are all set to SCADA*. This permits multi-drop operation for SCADAWARE because a network broadcast system is used instead of point-to-point. Each Node transmits to all systems named SCADA*, which will be all SCADAWARE computers. All these computers

process the command, but only those with the proper SCADAWARE system name will respond. Thus, the SCADAWARE multi-drop methods are identical to those of radio based systems.

HOST UPDATES

When the update process occurs, the host's network task will receive a message to read a specific file which performs the desired update process for each RTU. The procedure file is called RTUNAME_UP, where RTUNAME is replaced with the identification name of the particular RTU, as in PL15_UP, PL4_UP, etc.

Although each RTU has its own procedure, they are all similar in function. These procedure files are all stored in the single library file named UPDATE.LIB. Each Update procedure selects the proper RTU, then run a generic procedure called T1200_update. T1200_update tests RTU flag 115 which will be set if it is time to do a data base update. This flag is set hourly by an agenda activated procedure named DBUpdate.

If it is time for an update, as determined by RFlag 115, a procedure called Do_File_Update will be executed to do the actual database access. Any functions related to periodic updates shoult be done within that procedure.

VIEWNODE UPDATES

The Viewnode computers cannot directly access the radio-linked remote units. The viewnodes receive updates only via network broadcasts from the Host computer, which actually performs the radio data exchanges with the RTU's. The network update is part of a built-in process within SCADAWARE which occurs automatically whenever a data timestamp takes place at the host.

On the host, the RTU updates are initiated by communications task 1, which is named RADIO. Whenever this task receives a data update from an RTU, it initiates an automatic process on another task which has been designated as the UPDATE task. On the Trinmar Host computer, the NET task has been designated as the UPDATE processing task.

The update broadcasts are done over the network with the TSP SCADA protocol using NetBIOS broadcast network messages. In this way, each online viewnode is automatically and simultaneously updated each time the host receives information from an RTU. ViewNodes which are not online will gradually get up-to-

date information as each unit is polled by the main computer. A menu option on the ViewNode is available to force an immediate update from the Host over the network. Another option is available on the Viewnode to force a fresh data update by the host from any RTU.

VIEWNODE POLLING

Although ViewNodes cannot directly link to an RTU via radio, they can request the Marine Base to obtain an update which will be broadcast over the net when complee. The poll request is done by pressing F5 (the normal Poll key). Instead of actually polling the RTU, the viewnode sends a message (via the net) to the Host signaling it to do the actual poll of the RTU.

Each RTU has a default link associated with it that is used to poll for data. In the case of RTU's on a Viewnode, the link is set to NET instead of a physical Link number. This NET link designation tells the Viewnode to process a procedure named NETPOLL on the Viewnode whenever the Poll key is pressed. In the case of the Trinmar system, NETPOLL sends a network message to the Marine Base requesting an update for the specified RTU. The NETPOLL procedure is very simple, as shown below:

```
; generic network poll request. First param is the rtu name
dial host
block poll now $1
bye
```

This procedure is executed by the NET task on the viewnode whenever the user presses F5 on the viewnode. The actual command line processed by NET is READ NETPOLL RTUNAME, where RTUNAME is replaced with the actual tag name of the RTU. This name will appear as $1 in the procedure, allowing this generic code to be used for any network RTU.

Note that all this procedure does is to send a message to the Host telling it to poll the requested RTU. When this happens, the Host will proceed as if the F5 key had been pressed at the Host itself. After the update occurs, the Host will send the update data to all viewnodes, completing the update process.

HOST AND VIEWNODE DIRECTORY SETUP

The Host computers all have a similar directory setup which separates the system files into managable groupings. The main Host setup is as follows:

C:\SCADA

\BASE	Main directory with procedure and library files.
\SCREENS	All PCX and GIF graphic overlay files
\DB	All data base files

The viewnodes are similar, except that the DB directory isn't required because the viewnode retrieves Database information from the server.

C:\SCADA

\VIEWNODE	Main directory with procedure and library files.
\SCREENS	All PCX and GIF graphic overlay files

VIEWNODE CONFIGURATION

Each viewnode maintains its own set of program and setup files for everything except the DataBase files. The separate files permit each Production Field to maintain its own configuration in terms of alarm conditions. The local files also reduce the network traffic which would result if TSP procedures for screen generation were constantly executed over the net.

All the files for each ViewNode are stored in a directory path starting with C:\SCADA. The common database files are all stored on the Main computer in C:\SCADA\DB, but they appear to be on drive D: of the ViewNodes. With this scheme, all units maintain their own realtime database values, but extract historical information from a common data pool stored on the Marine Base PC.

Because each computer maintains its own configuration, they all must be kept in sync with any changes made to the field units. Any system level changes, such as the number of RTUs or the RTU I/O point layout, must be made to all viewnode locations. Each unit must have separate but similar setups so that they can share the common Marine Base Database, yet maintain separate alarm settings for all points in the system. This allows each field to determine which points it will monitor for alarms.

NEW VIEWNODE SETUP

Establishing a new viewnode can be easily by done copying the files from any existing viewnode as follows:

1. From the old viewnode, creat a new directory structure on the server (D:) to hold a copy of the viewnode setup. Storage directories exist for each of the present viewnodes, so creating one for the new viewnode should follow that example.

2. Copy the directories from the Viewnode to the temp storage on the server. There will be 2 important directories: the main one, and SCREENS.

3. Configure the new Viewnode to have network links to the SCADA server. The code for this will vary for each machine, and will involve settings in the autoexec.bat and config.sys files on the viewnode. The end result should have the SCADA server appear as drive D: on the viewnode. **Important:** The network operating system (NOS) on the viewnode must support NETBIOS or NETBEUI modes of network communications. This is standard on some systems, like Vines or Lantastic, but optional on others, including Novell.

4. Set up the required directories on the viewnode. The structure will be identical to any existing viewnode.

5. Copy from the temporary storage on the server into the local disk of the viewnode.

6. Edit the system data file, VIEWNODE.DAT, and change the reference to the old viewnode to the new one. This would involve, for example, changing the name EAST to MYPC, or whatever the new viewnode name will be.

7. Edit START.BAT to change the NET LOGIN line to reference the network ID of the viewnode. If it follows the convention of the existing machines, the network ID is the same as the Viewnode ID.

Delete the old image file, if present, by entering DEL *.IMG. This will ensure that the system starts up with a fresh binary image. Then type START, which should bring up the new viewnode software.

Once a system is operational, the user can configure the various data point attributes, such as alarming status, to meet individual needs. These will be stored

on the viewode hard drive whenever a SAVE command occurs, or when the system terminates normally. On restart, the program will load the configuration information from the IMG file on the local drive, restoring the users individual settings which will differ from those of the main computer, and from the other viewnode users.

HOST AND VIEWNODE STARTUP

During startup of the host computers, the autoexec will complete by switching to the main directory (i.e. C:\SCADA\HOST) and then executing START. All SCADAWARE systems use a standard startup batch file named START.BAT. This batch file will make any required network links before starting the actual SCADAWARE program (RTUMON3.EXE).

PRIMARY HOST DESIGNATION

The Marine Base unit is the primary Host computer for the entire system. It does all the routine polling, and responds to ALERT notifications from the SCADA Node units. The Tank Farm acts as the backup system to the Marine Base. However, only one system can be the primary host at any instant.

To avoid conflicts, the main Host and Tank Farm computers are set up with local control bits (in the BASE RTU) to determine which system acts as the primary host. The bits are switched by manual menu entry on each unit. It is important to ensure that both units are not set to be the primary host. If this occurs, their transmissions will likely collide causing a loss of radio communications.

REPORT FILE GENERATION

The BASE PC generates report files which are placed on both the BASE PC hard disk, and on a network disk setup as F: on the BASE PC. Generation of these files takes place during processing of a procedure called HOURLY which is triggered by SCADAWARE's AGENDA system. All periodic report functions should go into this procedure for easy maintenance.

The report files are in two formats. One is a simple text output which can be viewed directly by any file viewer in DOS or Windows. The second is a comma

delimited file designed for import by a spreadsheet. Each field has its own report pair. The text report has the file type of TXT. The spreadsheet data file has the type CSV for direct import into Excel.

In addition to the SCADA data, these files can also contain text data sent from the offshore Host locations. These messages are sent in individual files which have the same name as the offshore host (and tank farm), with the extension of TXT. It is possible to have the contents of these files added to the report files so that comments from the field can appear in the network accessible reports.

The initial report generation files prepared by TEST are for the East field. The file named EASTREP.RTU generates the text format report. The spreadsheet data file is produced by procedure file EASTDAT.RTU. These are the files that will be edited to change the contents of the EAST field reports. Similar procedures will be required to produce reports for the other fields.

BASE PC PRINTER CONTROL

The BASE PC will have a printer attached to it which can print a variety of alarm and data reports. If the printer is not available, some system functions will be affected which can slow or stop the entire system. SCADAWARE attempts to detect printer problems and sets the print functions offline automatically. If the printer is suspended, it can be restarted with a menu entry on the PRINTER CONTROL menu of the base PC.

Options are also provided on this menu to control automatic printing of alarm summaries, and to select the printer type for graphic printouts.

PAGER (BEEPER) OPERATION

The pager (beeper) activation process is performed in SCADAWARE with a modified application of the standard LINK feature. Links are the means by which SCADAWARE dials out to a remote unit to perform a specific function such as a data download. In the case of the Pager, all the Link needs to do is dial a specific sequence of numbers which program the paging system to activate the beeper. The dialing sequence is identical to that done during a manual dial. Any pauses required in the sequence are handled by commas in the dialing string.

Links normally remain active in SCADAWARE until they are reset during a

data download, or until they reach a FAILED state when the retrys are exhausted. In the case of the pager link, they are not naturally reset because no real data communication takes place. They always fail after the specified tries, which activates an automatic failure procedure tied to the link number. These procedures are named CFAILxx, where xx is replaced with the affected link number.

Each data point (channel) in the system has the ability to activate one or more callout Links. Each channel is assigned to a specific allout group, and the group can contain one or more link numbers to be activated. Each SCADAWARE system can have many groups, each of which can have one or more links associated with it. Note that a channel cannot activate a link directly. It must activate all the links in the group assigned to it. However, the group can contain only one link, so the end result is the same as if the channel was directly tied to a link.

The method used to activate pagers is to assign each pager to a specific link, and then to assign that link to a specific group. If certain points require that multiple pagers be alerted, then additional groups can be established which contain the links associated with those pager numbers.

When the pager link is activated, it blindly dials the number sequence stored in it. SCADAWARE cannot tell if the call went through, or if the pager unit actually received the signal. Because the pager system is not foolproof, the number of trys in the link setup must be set high enough to reasonably insure that one of the callouts will get through, and that the page will be received by the beeper unit.

The phone number in the link contains commas, which tell a Hayes type modem to pause. These are necessary on pager callouts to allow time for the system to answer, announce that an ID is required, and then announce that a phone number is required. The modem is altered during task 1 startup so that each comma equates to a 4 second delay. To get 12 seconds, 3 commas are required in the phone number text. For example, to callout and wait for two voice prompt messages, the form of the number would be:

9,123-4567,,,4321,,,555-1212

The 9 gets an outside line. The number 123-4567 is the phone number of the pager service. The next 3 commas are a wait to let the system say "Please enter the ID number of the person...." The modem will then send 4321 as the personal ID we which to page. The system will then announce "Please enter your

phone number which will be displayed on..." The modem then enters 555-1212 as the number you want the person to call.

The actual numbers used will of course depend on the pager service, the personal ID, and the return phone number. For SCADA applications, a real phone number isn't required for the return number. Instead, a code (such as 99999) can be entered to tell the receiver that it was the SCADA system calling.

HOST PC FILES

The Host PC contains many separate procedures which, as a group, comprise the custom programs prepared for this system. To simplify the design and maintenance, several standard methods of writing and accessing the procedures were implemented. These methods generally combine the RTU name with other alpha characters to form a specific procedure name to be executed.

For example, all of the graphic screen procedures are stored in individual files which have a file type of SCR (for screen). This is used in place of the default RTU file type normally associated with disk based procedures. Each RTU has its own screen file which is comprised of the name of the RTU followed by .SCR. Thus, the screen filename for Platform 21 is PL21.SCR.

The following is a list of the naming conventions used in this system:

1. SCR - GRAPHIC SCREENS - All screen procedures are given the SCR file type. Each RTU has a separate file.

2. XXX_UP - UPDATE PROCEDURES - Each RTU has an update procedure which consists of its RTU name followed by _UP. For BS16, the update procedure would be named BS16_UP. All of the update procedures are stored in a single library named UPDATE.LIB.

3. FRM - DISPLAY FORMS - The text mode FORM screens which display summary data for each RTU are stored in individual files with the type FRM. The form file for PL20 is in file PL20.FRM

4. RTU - CHANNEL DATA - Each RTU stores its channel data setups in separate files, each with the filetype RTU. When a SAVE command is executed, SCADAWARE stores a text formatted version of all channel data into this file. These files are not normally used during startup because channel data is also stored in binary format in the system IMAGE file.

However, if the image file is not available, the program will automatically read the RTU data file to reload all important channel data. The data file for BS16 is named BS16.RTU

5. LOG - DATA LOGS - Any channel can be set to Log alarm information. This information can be viewed with the SCADAWARE Log command. The log data is stored for each RTU in separate files which have the file type of LOG.

6. RTG - REAL TIME GRAPHIC - The TSP code necessary to generate individual graphic blocks are stored in separate disk files with the type RTG. These files are produced automatically when the Graphic Configuration screen is used in SCADWARE.

GRAPHIC SCREENS

SCADAWARE graphic screens are stored in TSP procedure files with a file type of SCR. Each RTU has its own screen generation file. All procedures are similar and perform these basic steps:

1. A startup section executes to prepared the screen and program. Because the setup executes only once, it is placed at the end of the procedure to save processing time after startup is complete. The setup will clear the screen and put the CRT into graphics mode at 256 colors, 800x600 dots. No graphic backdrops are used, so a solid blue background with a beveled frames is used instead.

2. After initializing the CRT, Startup will begin loading the configuration for each graphic element for the screen. Some of these are loaded via GOSUB to an RTG file which contains compact code for graphic elements prepared with the CONFIG command. Other screen elements are loaded directly in the SCR file using RTG (real time graphic) command lines.

3. A common graphic menu is also loaded which uses button icons across the bottom of every screen.

4. Setup makes up the majority of the code in each screen file. When setup is complete, all graphic elements have been loaded into the processing system, but have not displayed because no update has yet occurred.

5. The program jumps back to the top at :LOOP. This label forms a reference point on which the display screen will loop until terminated.

6. During the first few passes of the main loop, the trend screens are drawn with updates to the graphic windows containing the trends. After the trends are complete, this section of the code is skipped during subsequent loops.

7. All passes of the main loop perform updates to the non-trend graphic elements. This includes the vertical bar graphs as well as the graphic style text boxes. The code for the update sections will vary slightly depending on the particular screen arrangement. A typical update section is as follows:

```
:loop          ; branch point

; Draw inside trends during first 25 loops only
if (@dberr(0) = 0) & (z < 25)
 win sele 1
 win update
 win sele 2
 win update
 calc z = z + 1
 db next
else
 db close
endif
 win sele 3
 win update
 win sele 4
 win update

 rtg update        ; do all updates in one hit

;---------  start common routine
 if key = 0    ; if key not set already by a mouse hit
   calc key = @key(0)
 endif

 if key = 0
  goto loop
 endif

; Fall out and exit the program
  win close
  db close
  previous
;---------------   end common routine
```

8. The above example demonstrates how "Win" style graphic updates are done for each active window. It also contains an RTG update which occurs with a single statement. The screen update loop continues until a key press (or mouse hit) which causes the loop to terminate.

9. Note that the procedure terminates with a PREVIOUS statement rather than a RETURN. This causes a return to the *start* of the calling procedure so that it can completely redraw itself and overwrite this display.

SMART RTU SETUP

Each Type 1200 RTU consists of an industrial personal computer and special Input/Output hardware to connect the field sensors. Most field locations have a solid state disk (SSD) which uses low power RAM chips instead of a rotating hard disk. For human I/O, the RTU has a standard PC display and keyboard. On the unmanned platforms, the CRT and keyboard are removed except for maintenance. For the Manned (Host) locations, a color CRT, keyboard, and mouse are provided for routine operation.

The PC hardware for the T1200 has several special features which require attention during the setup of the RTU. In particular, the SSD must be specially formatted in order to operate as the DOS system disk. The instructions for this setup are detailed in the CPU board documentation. Once the SSD is configured as drive C:, it can be used much like a normal hard disk.

A diskette set was provided for each RTU which contains all the files and programs required to set up an RTU. After formatting the hard disk, DOS can be installed onto it using the standard DOS SYS command function. A minimal amount of DOS software must be installed on the hard disk in a subdirectory named C:\DOS. At that point, the RTU should be able to boot from the SSD.

All files required to operate SCADAWARE are installed into a single subdirector named C:\RTU. This directory will contain the RTUMON program, the system configuration files, and the procedure files and libraries. Keeping all the files in a single directory is possible on an RTU because of the small size of the files. All files can normally backed up to a single 3-1/2" floppy disk. Restoring the system after a disk reformat can be easily done by copying the backup disk into the C:\RTU directory.

During initial system configuration, several special files must be edited to detail the exact configuration for each RTU. The generic I/O hardware used to interface with field sensors must also be defined for each particular setup. This information is stored in a simple DOS text file which has the name of the RTU, and a file type of GIO (for generic I/O). This file identifies the type, quantity, and

addressing of the I/O hardware installed in the RTU computer. This file is processed during startup of SCADAWARE to establish the link between the physical I/O devices and their associated SCADAWARE data channels.

Another file unique to each system is the system data file, or DAT file. This file details the number of I/O points, the RTU name, and operating tasks for each system. It too is a simple DOS text file.

Other files will be very similar among the RTU's, with slight differences in location names and update procedures. In each case, a single library file is used to hold all the procedures for each RTU. The file is named for the RTU, with a file type of LIB.

NEW RTU SYSTEM SETUP

The best way to set up a new RTU is to take the files from an existing one and make a few changes. The required changes can be summarized as follows:

1. Copy all files from the old RTU into a separate diskette. Copy this diskette to the proper directory (C:\RTU) on the new RTU.

2. On the new RTU, edit START.BAT and change the /F=oldname to /F=newname. This specifies the default system file names for the new system.

3. Delete the image file (DEL *.IMG) and any other files which are clearly not required in the new system.

4. Rename all the RTU related files to the new RTU name. For example, if the old RTU was named BS16, and the new one is PL30, then the DOS command:

   REN BS16.* PL30.*

   could be used. Make sure this is done on the new machine, not on the one which is being copied.

5. If it is desired to keep the channel configurations of the old RTU, then edit the PL30.RTU file to change the first line which contains the SELE statement to reference the new RTU name. If mostly new designations are required, then delete the PL30.RTU file instead so that SCADAWARE will install default channel configuration information.

6. Edit the DAT file and change the name reference from the old name to PL30. Also edit the com port and task setups to meet the needs of the new system. Finally, edit the data channel sizes to suit the new I/O.

7. Edit the LIB file(s) and change name references from the old RTU to the new one. The details of this will vary greatly from system to system, and may involve complete replacement of specific procedures. The only procedure which will appear in most systems is DOWNLOAD, which controls default data transfers from an RTU.

8. Edit the MNU file(s) which control the user menus.

After making these changes, type START to get the system going. It should begin execution and start operation. The built in editing and configuration features can then be used to finish the system setup. When done, a total backup of the new configuration should be made.

ADDING AN RTU TO THE HOST

The host computer must be reconfigured whenever a new RTU is added to the system. The configuration can be done as follows:

1. On the RTU, type SAVE to make a text mode copy of the channel configuration data. Copy this file (RTUNAME.RTU) to a floppy for transport to the Host.

2. On the Host, edit the system DAT file to include the additional RTU. New units should be added to the end of the list. The easiest way is to block copy a similar RTU and make the minor changes.

3. Edit the startup procedure (usually in the main library) to include a load of the additional unit. This will occur in the section which executes if an image file cannot be loaded. The complete load (instead of an image load) is required when any significant changes are made to the overall system setup. Additional changes will also be required to the link library and possibly other libraries, depending on the needs of the new RTU. These can be deferred it until after the RTU has been brought on line.

4. Save all RTU configuration data on the Host in TSP text format by entering >SAVE. The > sign will cause the save to occur for all RTU's with one command. The text mode configurations of all RTU's will be reloaded when

the system is restarted.

5. Terminate SCADAWARE and return to DOS. At the dos prompt, delete the system image file by entering DEL *.IMG. This will force a reload of all RTU configurations, including the new one, when the system is restarted.

6. Restart SCADAWARE. You should notice that all RTU's are loaded during processing of the startup procedure. After the system is loaded, the new RTU should appear in the F10 RTU selection menu.

7. Insert a diskette containing the RTU setup file. This text file was generated by SCADAWARE on the RTU with the SAVE command. The diskette file can be loaded onto the Host by entering READ A:RTUNAME.RTU , where RTUNAME is replaced with the actual RTU tagname.

8. After the RTU config file is processed, the Host will have an identical channel setup to that of the RTU. This will not be quite correct because of different alarm settings between RTU and Host setups. Use the Config command (or change command) to revise all channels for host mode. This usually involve changes to the CALL ON ALARM and EXECUTE RTU FILE settings in each channel. The CHANGE command allows all channels in each type (Status, analog, etc) to be changes at once without use of the config screen. Note that a procedure named MAKEHOST has been provided to simplify the conversion from RTU to host. This is executed by entering READ MAKEHOST at the SCADAWARE prompt.

9. Store the revised channel configurations to the hard disk by entering SAVE at the SCADAWARE prompt.

10. Make similar changes to the Viewnodes and other Hosts so that they will contain the revised RTU information.

TROUBLESHOOTING

The Trinmar SCADA system is a collection of components connected in a variety of ways. In general, data flows from the field transmitters through the RTU's over the radio to the Host, and then on to the viewnodes. The key to troubleshooting is to determine where in this chain that things have gone wrong. SCADAWARE and the RTU's have many facilities designed to assist in troubleshooting. These are explained in the standard manuals. What will be covered here are the specific points of interest along the data path, and what can

be observed at each station. The path starts in the field at the sensor and continues all they way to the display screen.

	LOCATION	MONITORED DATA OR SIGNAL	DIAGNOSTIC RESULT
1	Field End Device	Physical signal level such as voltage from a switch, or current from a transmitter.	Confirms device operation and supply voltage from the RTU.
2	RTU Termination Point	Return signal from the end device. For analog signals, the scaled voltage can also be read on the MAT or CAT resistors. Status inputs can be verified by LED indicator.	Confirms wiring back to the RTU. Also confirms lowest level operation of the RTU I/O interface.
3	RTU Display and Data Statements	Converted status or analog data as processed by SCADAWARE within the RTU.	Shows the RTU's interpretation of the I/O, confirming Norm Open - Closed settings, and analog scale settings.
4	Serial Data from the RTU	Bit stream coming from serial port to the modem. Indicated by LED tap or scope measurement	Any activity shows that RTU is trying to transmit. Tapping with another PC can confirm data accuracy.
5	Audio from Modem	Audio type "noise" signals with separate tones for 1 and 0 bits of serial data. Can be heard with a small amplifier.	Familiar tones indicate proper operation of the modem. Signal level must be adequate for radio input requirement.
6	RF from Radio	Noise heard on scanner radio. Should be clear and consistant.	Indicates that Radio is outputting suitable signals. Can be digitally monitored with modem and PC connected to scanner audio output.
7	Audio from receiving radio	Noise heard coming from receiver. Should have brief squelch period followed by data.	Proves receiver is getting adequate RF at radio input.
8	Serial data from receiving modem	RS-232 data which should be identical to what is seen on RTU display. Can be monitored with serial data tap and another PC.	Proves modem input signal is decodable.
9	Receiver Comm task data input	Processed data line received by the Host computer radio task. Monitored with WATCH command on the Host CRT.	Indicates data line processed through complete communication system.

10	Host Display Screen	Data from live data table on the Host.	Indicates proper insertion of RTU data into proper position in Host data table.
11	Host Database File	Data stored by update process on Host.	Verifies proper operation of update procedures
12	Viewnode Display	Data stored in local Viewnode data table.	Verifies network updates from host to viewnode
13	Printed Report	Data sent from Host to Report Files	Verifies report generation process

-- End --

AMZ/nt TRINDOC.WP