Design and Development of an "Optimised Telemetry Control System"

A thesis submitted in fulfilment of the requirements for

the degree of Masters of Engineering

at Auckland University of Technology

by

Avinash Salig

July 2011

Primary Supervisor: Prof. Ahmed Al-Jumaily

Table of Contents

Acknowledgements

I sincerely thank my supervisors for supporting me throughout my thesis with their knowledge, advice and support.

Supervisor:	Prof. Ahmed Al-Jumaily	Auckland University of Technology
Co-Supervisor:	Dr Boon Chong Seet	Auckland University of Technology

I am also grateful to my industrial mentor for his technical advice and guidance throughout the project.

Industrial mentor:	Neil Pearce	CSE-W. Arthur Fisher Limited

I express my sincere appreciation to the management of CSE-W. Arthur Fisher Limited for the opportunity they provided me regarding the thesis topic and associated objectives, expertise and facilities provided.

I express my gratitude to Technology New Zealand for their financial support through the capability contract for education fellowship (Contract Number TP020932). This made a tremendous impact by allowing me to focus on my objectives and achieve my goals.

Special thanks to “Trio Datacom” for loaning the radio equipment that was used for the design of this project.

Abstract

Designing an optimised telemetry control system will improve the quality of service for Supervisory Control And Data Acquisition (SCADA) systems implemented by CSE-W. Arthur Fisher and enable system expansion thus minimising revenue for their future system designs. The telemetry control system ensures a high degree of data reliability and integrity to meet SCADA operational requirements. This thesis presents the design and development of an optimised telemetry control system using Kingfisher Remote Terminal Units (RTU's) with Kingfisher Series 2 protocol.

To determine the system response for data transmission over the bandwidth**,** quantitative research methods were undertaken to evaluate communication blocks within the Kingfisher protocol. There are usually different techniques used to collect data from remote stations. The Kingfisher S2 protocol implements two techniques namely "Exception Reporting" and the "Polling" technique for data acquisition. The polling technique was the most efficient in terms of bandwidth ultilisation for transferring data therefore the system was designed using a pure polling system approach. It also enabled the communication links for remote stations to be monitored and enabled a deterministic system design approach to be implemented. Research focused on polling system optimization whereby efficient polling frequencies were calculated based on theory presented by (O. J. Boxma, Levy, & Weststrate, 1991). The aim was to efficiently allocate the limited bandwidth resource to a number of remote stations thus optimising the system performance.

The proposed theory was implemented for system optimisation. It enables efficient polling frequencies to be calculated for a polling cycle hence optimising the bandwidth utilisation and eliminating fairness problems for the medium access control. Bandwidth optimisation enables system expansion thus reducing the networks need for additional resources.

A pure polling telemetry communication system was implemented in this design using point to multipoint network topology over half duplex radio channel. Empirical data modelling enabled the design of the service duration period to allow for time sharing between the remote stations to share the bandwidth. The bandwidth was designed to share real time data and event log for SCADA systems monitoring and control. Queuing analysis was performed to establish system parameters and enable system optimisation. From the literature review the implemented design methodology uses the "mean delay approximation" method which was

used to calculate efficient visit frequencies and enabled the optimisation of the bandwidth to the remote stations based on the workload of each remote site. The software for the telemetry control system was developed and tested using ladder logic. The results prove that the bandwidth utilisation can be efficiently controlled thus optimising the telemetry control system. The implemented design improves the quality of service for the SCADA system by providing regular real time system status poll requests for control purposes and was given the highest priority for medium access. It also performs a polling of individual sites according to the “mean delay approximation method” to efficiently allocate bandwidth amongst the remote stations depending on their workload thus optimising the system. The system was designed to be responsive to high priority event log data thus enabling system flexibility.

Abbreviations

RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
PLC	Programmable Logic Controller
HMI	Human to Machine Interface
bps	bits per second
Block	Group of 64 Registers internal memory
KF S2	Kingfisher Series 2 Protocol
ACK:	Acknowledge
ASCII:	American Standard Code for Information Interchange
BCD:	Binary Coded Decimal
BIT	One or zero condition in the binary system.
CPU:	Central Processing Unit.
CRC:	Cyclic Redundancy Check
CSMA/CD:	Carrier Sense Multiple Access/Collision Detection
CSMA/CA:	Carrier Sense Multiple Access/Collision Avoidance
DCE:	Data Communications Equipment
DTE:	Data Terminal Equipment

Hex:	Hexadecimal
LAN:	Local Area Network
MAC:	Media Access Control
Modem:	Modulator - Demodulator
OSI:	Open Systems Interconnection.
TDM:	Time Division Multiplexer
IED:	Intelligent Electronic Device
DCS:	Distributed Control System

Chapter 1

Introduction of Telemetry Control Systems

1.1 Introduction

This chapter introduces telemetry control systems and communication mediums implemented for the design and development. It also outlines the motivation for the research and the main objectives for the design.

1.2 Project Background

The purpose of telemetry is to collect data at a place that is remote or inconvenient and relay the data to a point where the data may be evaluated (Carden, Jedlicka, & Henry, 2002).The telemetry control system must ensure data reliability for data acquisition and process control requirements. The telemetry communication system is required for SCADA system applications. These systems are said to be “event driven” and are required for process monitoring purposes.

1.3 Telemetry Control System

A telemetry control system requires the measurement and control of remote sites over a large distance. These systems have been applied to oil and gas, water and electrical utilities. Each remote site is responsible for the local process control using analogue and digital input signals. The communication system is used to link the central control site to all existing remote field stations which are geographically dispersed from a few meters to kilometers in distance. The central control site typically consists of a SCADA (Supervisory Control And Data Acquisition) system. Figure 1-1 shows a typical layout of a telemetry control system. The master station is the central control station and the remote sites are responsible for the local process control. Data is acquired from the remote sites over the communications link to the master station for the system control and monitoring.

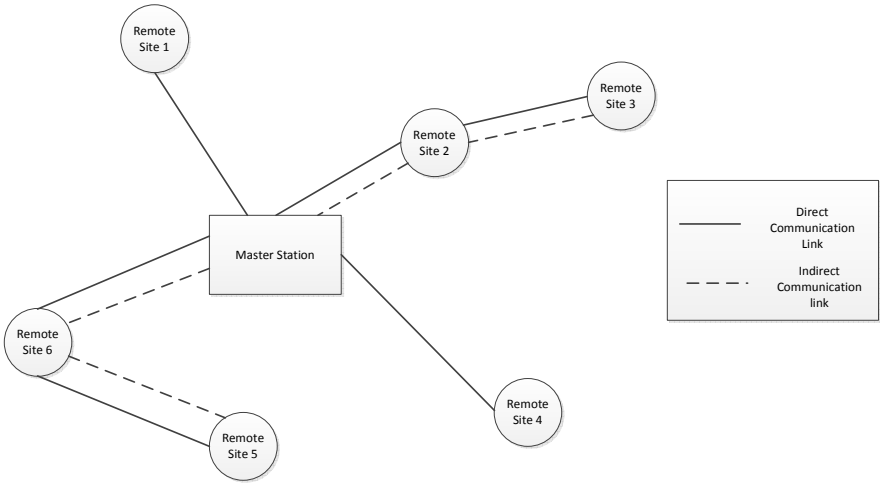


Figure 1-1 Typical telemetry control system

These systems typically use polling and exception reporting techniques to acquire data to the master station. The main problem with telemetry systems is their inefficiency to transmit data due to limited bandwidth availability over radio links which consist of protocol overheads to ensure data integrity further minimizing bandwidth. The telemetry control systems main responsibility is to reliably transmit data over the communication network to the SCADA system.

1.4 Telemetry Communication Mediums

The telemetry communication system mediums can be twisted pair cable, fibre optic, radio frequency, telephone line, microwave and satellite of application. However for a dispersed system over a few kilometres it is not practical to use a hired wired approach such as twisted pair cable and fibre optic which is not practical since they are prone to breakage, water ingress and prevent network flexibility. The practical mediums for this design are presented in this chapter.

1.4.1 Leased telephone service

Leased lines are established by a communications service provider and serve as permanent, private connections between two locations (White, 2008). Leased dedicated circuits are used for telemetry and SCADA applications for a lease or rental charge. Cable modems and digital

subscriber lines have been implemented for these systems and provide low cost efficient data transmission.

The advantage of this system is no communication expertise is required for the communication system design. The communication maintenance is supported by the telephone line provider. The major disadvantage is that circuits may not be available at some sites due to its geographical location and the factor of continual rental costs.

1.4.2 Satellites

Satellites have been investigated for their use in the utilities sector and are positioned in the circular orbit above the earths equatorial plane and offer a large coverage of the earth. The electronics in the satellite that receives the uplink singal, amplifies and possibly processes the signal and then reformats and transmits the signal back to the ground is called the transponder (Ippolito, 2008)**.** Earth stations are comprised of an antenna pointing at the satellite and radio transceiver. Two attributes deternine the specific frequency bands and other regulatory factors for a particular satellite system:

- service(s) to be provided by the satellite system/network and
- location(s) of the satellite system/network ground terminals.

Both attributes together determine the frequency band or bands where the satellite system may operate (Ippolito, 2008)**.** Satellites have a large coverage area therefore they offer ease of access to remote sites. They ensure data integrity and generally have low error rates. However they pose disadvantages of being totally dependant on a remote facility as well as transmission time delays and continual rental costs for companies.

Frequency Band Usage		Uplink Frequency (GHz)	Downlink Frequency (GHz)
L	Mobile	1.550–1.600	1.500–1.550
S	Military/Government	5.925–6.055	2.535–2.655
C	Commercial	5.2625–26.0265	3.2600–4.800
X	Military/Government	26.900–8.400	26.250–26.2650
Ku	Commercial	12.265–18.10	10.260–13.25
Ka	Military/Government	226.00–31.00	18.30–22.2

Figure 1-2 Satellite frequency bands (Reynders, Mackay, & Wright, 2005)

1.4.3 VHF radio

The VHF band operates between 30 - 300MHz (*Radio Spectrum Management*). Typically this band is utilised for mobile radio communications.

1.4.4 UHF radio systems

The UHF band extends from 300 to 3000 MHz (*Radio Spectrum Management*). The bands typically considered for UHF radio operate between 400 MHz to 900 MHz range. The upper regions of this band are utilised for microwave systems. UHF radio systems can be trunked mobile radio, microwave, spread spectrum or multiple address radio systems (MARS).

1.4.4.1 Trunked mobile radio

In a trunked radio system each user is allocated a channel on a per call basis and upon termination of the call, the previously occupied channel is immediately returned to the pool of available funds (Rappaport, 1996). This system enables a large number of users to share a limited number of channels however requires a communication controller to handle this function. This system can be used for voice communications as well as data communications and operate in the 400 MHz, 800 MHz and 900 MHz bands.

1.4.4.2 Microwave radio

Microwave radio systems operate at frequencies above 1GHz. Due to their high frequency they provide high channel capacity and high data rates. Microwave systems require clear line of sight between antennas (Horak, 2007). This is a disadvantage of this system as well being more expensive for system development.

1.4.4.3 Spread spectrum radio

Spread spectrum radio frequencies do not require a license and can operate in the 900MHz, 2.4GHz and 5.3 GHz band. A spread spectrum modulation scheme is any digital modulation that utilizes transmission bandwidth much greater than the modulating signal bandwidth, independently of the bandwidth of the modulating signal (Ziemer, 2007). The receiver correlates and demodulates the signals. Modulation techniques are Frequency-hopping spread spectrum (FHSS), direct-sequence spread spectrum (DSSS), time-hopping spread spectrum (THSS) and chirp spread spectrum (CSS). There are also hybrids of these techniques which form spread spectrum.

1.4.4.4 Multiple address radio systems (MARS)

Modern radio modems operate in the 400 and 900 MHz band. Propagation in the 400 and 900 MHz band requires a free line of sight between transmitter and receiver. These systems typically operate in a point to multi point network configuration whereby a master station communicates using omnidirectional antennae with multiple remote stations using directional antennas over a single channel radio system. Protocols implemented in these systems use simple poll - response techniques to co-ordinate the communication network. Speeds of operation are up to 9600 baud which are typically used for SCADA/Telemetry and data applications. This type of system is used by CSE-W.Arthur Fisher for implementation in their applications. Therefore this system was used in the design of the communication network.

The radios implemented for this system are low cost compared to microwave radios and transmission is possible over non line of sight. This system also provides long MTBF (Mean Time Between Failure). These systems have low channel capacity or bandwidth over serial radio communications, low data rates as well as limited transmission techniques. They typically operate at 9600 baud or slower. Due to these factors system optimisation is required.

1.5 Network Topologies

Network topology is the interconnection of multiple nodes to form a network structure via the physical medium. The radio frequency medium in this design forms a wireless network to connect multiple remote stations to a centralised station.

1.5.1 Point to point topology

A point to point topology is a direct connection of the physical medium between two nodes only. This is the most basic topology and can exist as a permanent connection or switched on demand basis.

1.5.2 Star topology

A star configuration has a central node and many peripheral nodes. All peripheral nodes connect to the central node which controls the data throughput to the network. Therefore if a peripheral node fails the network will still operate and only that node will be isolated from the rest of the network. Such centralized topology results in an efficient and reliable network service to the client devices (Reynders et al., 2005). This is referred to as point to multipoint in radio system designs.

1.5.3 Peer to peer topology

This topology is also referred to as ad-hoc mode where multiple nodes interconnect with each other without any central node to form a mesh. All nodes route traffic and communicate with one another. The nodes self-configure to form an integrated network utilizing distributed control (Reynders et al., 2005). This topology enables re-routing via multiple paths if communications to a node fails.

1.5.4 Hierarchy topology

A hierarchical topology is ideal for networks that span a range of coverage regions (Reynders et al., 2005). This network topology consists of a central node at the top of a hierarchical structure. Peripheral nodes are connected below the central node to form the secondary level. The secondary level connects to tertiary level of nodes forming a multi-level structure for the network configuration.

1.6 Motivation

The CSE–W. Arthur Fisher's telemetry control systems are used in a wide range of industries and provide communications systems for mission critical & heavy industries and utility infrastructure clients throughout New Zealand.

Utility systems depend on real time information for system operation. Telemetry systems were designed for data acquisition and alarm reporting from remote field sites. These systems have low channel capacity or bandwidth over serial radio communications, low data rates as well as limited transmission techniques. They typically operate at 9600 baud or slower and implement protocols for data integrity which further minimises the bandwidth. Due to these factors system optimisation is required.

Due to this limited bandwidth problem an efficient algorithm is required to maximise the bandwidth utilisation as well as ensuring data integrity and efficient bandwidth allocation over all the remote sites at the same time preventing fairness problems and enabling the SCADA to receive system updates.

Designing an optimised telemetry control system for CSE-W. Arthur Fisher will improve the quality of service for the Supervisory Control And Data Acquisition (SCADA) system and enable system expansion or scalability thus minimising revenue for their system design. The telemetry control system must ensure a high degree of data reliability and integrity to meet SCADA operational requirements.

This master's project was initiated with the aim to develop an optimised telemetry control system. This thesis presents the design of a SCADA system and the optimisation of the telemetry control communication network.

1.7 Objectives

The main objectives of this research may be summarised as follows

- To design and develop an optimised telemetry control system
- To develop the software for the telemetry control system using ladder logic
- To design a SCADA (Supervisory Control And Data Acquisition) system
- Improve the bandwidth utilisation and efficiency of data traffic from remote sites
- Empirical data model of the system response time for data transmission and for future design and development

Chapter 2

Survey of Literature

2.1 Introduction

This chapter introduces typical SCADA system design and compares the industrial protocols currently implemented in the utilities industry. It also introduces bandwidth allocation schemes used to control data traffic types from control systems in order to improve the bandwidth control of communication networks. A literature survey of polling systems was performed since this was the main technique used for data acquisition for the telemetry communication system design and development.

2.2 Supervisory Control And Data Acquisition (SCADA) Systems

SCADA systems are widely used in industry for Supervisory Control and Data Acquisition of industrial processes (Daneels & Salter, 1999). The distance between the control station and multiple remote field sites can range from a few meters to up to thousands of kilometres. A telemetry communication system is required to connect the control station to remote field sites to transmit commands and receive remote monitoring data. SCADA encompasses the collecting of information, transferring it back to a central site, carrying out any necessary analysis and control and then displaying the information on a number of operator screens or displays (Bailey & Wright, 2003).

2.2.1 SCADA System Integration

A SCADA system consists of a Master Station, Communication System and Field Sites (RTU's) as shown in Figure 2-1**.** SCADA systems require monitoring over a dispersed area and therefore typically require a telemetry system to connect to remote field site RTU's. The RTU's accept commands from the master station to control the process points and set points. An RTU monitors certain points and stores the information in a local addressing scheme. The addressing is designed to correlate with a database contained in the master station. This represents the predominant SCADA systems designs which implement specialised industrial protocols designed for efficient data transfer and integrity in the utility industry today.

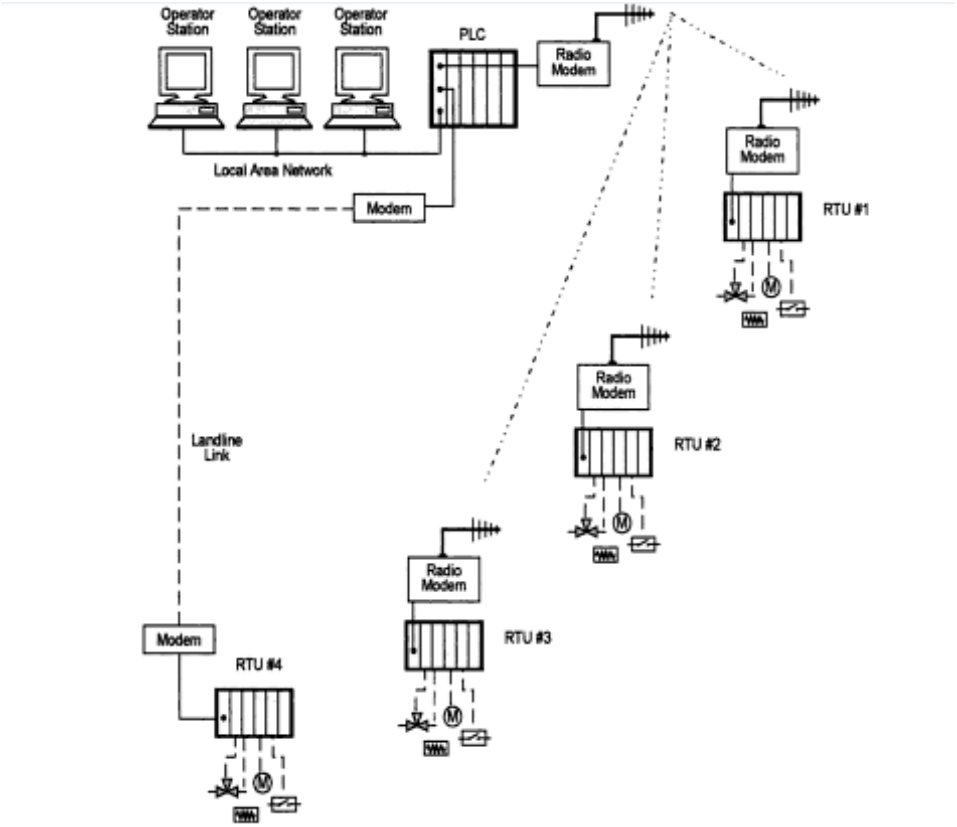


Figure 2-1 Typical SCADA system (Bailey & Wright, 2003)

SCADA systems require the integration of hardware to perform its function. Hardware must be interoperable and open for system integration. This section introduces the different hardware and manufacturers which will be used in this SCADA system application.

The proposed system is to use “Kingfisher Series 2” RTU’s which are responsible for remote process control, monitoring and communications with the central control station. In this design “Trio E-Series” serial radio modems were implemented for the communication system over UHF radio frequency. The central or master station was networked with “Wonderware Intouch” software package for Human to machine interface (HMI) over Ethernet local area network (LAN).

2.2.2 Remote Terminal Units (RTU's)

A remote terminal unit (RTU) was designed to handle communications for a SCADA system by collecting data from field devices and sending the data to the SCADA master on command(WEF, 2007). The hardware consists of a CPU and IO modules (Analogue/Digital IO's) and with communication interface as shown in Figure 2-2. They are able to communicate on a peer to peer basis with other RTU's and can also be configured to relay information to other remote stations. RTU's were designed to handle communications for SCADA systems (WEF, 2007). SCADA utilise wireless telemetry communication systems for remote process control and data acquisition which typically operate at baud rates between 300 - 115Kbps.

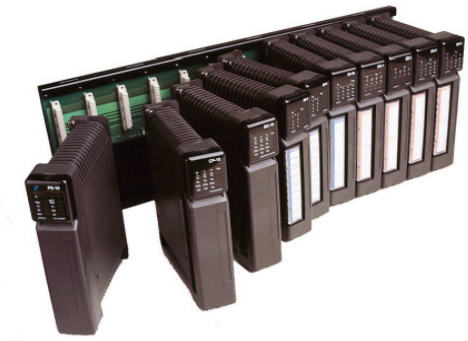


Figure 2-2 Kingfisher Series 2 Remote Terminal Unit

An RTU network is 2 or more RTUs that can communicate with each other in some way. The communication path is called a route. Usually one RTU is setup as the master RTU. The master RTU regularly polls data from all the other RTUs. Polling is a channel access method in which each RTU is asked in a sequence whether it wants to transmit. The other RTUs are referred to as remote RTUs and can report data changes as they occur sporadically called Exception Reports.

Figure 2-3 shows an RTU network which consists of RTU1 as the master and RTUs 2-4 as remote RTUs. RTU3 also stores and then forwards messages between RTU1 and RTU4. If only polling is used, it will take up to the regular polling interval before the master RTU knows about new data from remote RTUs. For example if the master polls the remote RTUs

every 2 minutes, it will take up to 2 minutes before the master receives new data from the remote RTUs. If only reporting is used, the master RTU will not know if a remote RTU has failed. If the master RTU does not receive a message from a remote RTU for a long time this could mean that either there is no new data or that the remote RTU has stopped communicating.

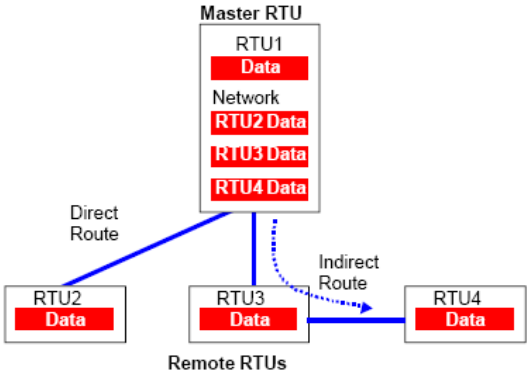


Figure 2-3 RTU network

2.2.3 PLC or Programmable Logic Controller

The PLC or Programmable Logic Controller is still one of the most widely used control systems in industry which are developed for "real time status" floor applications. PLC communication modules were developed to support Ethernet (WEF, 2007).

Equipment such as PLC's, instrumentation, sensors, PID controllers are connected to controllers via fieldbus. Communication protocols used in fieldbus are Modbus RTU, FIP (Factory Information Protocol), FF (Foundation Fieldbus), Profibus DP, DeviceNet, CANopen (Mikluszka, 2010). There are also other proprietary protocols which are implemented. These protocols are referred to as bus protocols and commonly implement Ethernet TCP/IP.

2.2.4 Distributed Control System (DCS)

Figure 2-4 shows a distributed control system which consists of a number of distributed computer systems situated near the instruments or devices being controlled. They are implemented over a high speed bus network for fast real time applications implemented typically for critical operations.

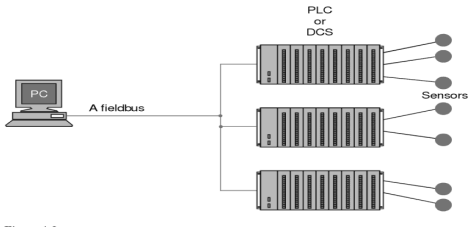


Figure 2-4 DCS system (Bailey & Wright, 2003)

2.2.5 Radio modems

Radio modems typically operate in two modes. Figure 2-5 shows Full- duplex mode which is more efficient in terms of data transmission. It also avoids any traffic collisions and typically operates over two separate channels. Full duplex systems allow messages to flow in both directions simultaneously being its major advantage however this comes at additional cost for bandwidth or frequency.

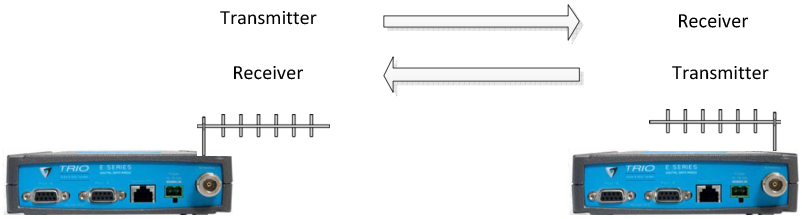


Figure 2-5 Diagram of full- duplex communication

Figure 2-6 shows half duplex mode whereby the radio modem can transmit and receive only one at a time. Typically this mode is used over a single channel.

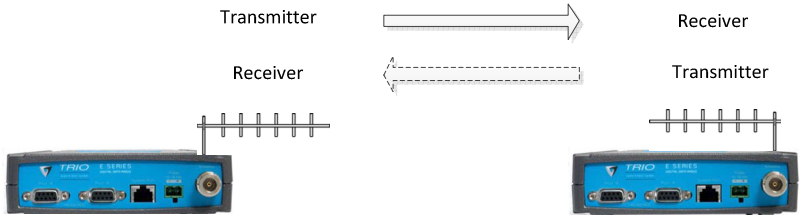


Figure 2-6 Diagram of half duplex communication

2.2.6 SCADA software

SCADA software is used to design human to machine interfaces for supervisory control and system monitoring. SCADA software can be divided into Proprietary or Open (Bailey & Wright, 2003).The ability to have interoperability of software enables system integration

from different vendors and enables communication protocol interface between RTU's and PLC's. Figure 2-7 shows a typical HMI used to provide graphic displays of process, alarms and trends.

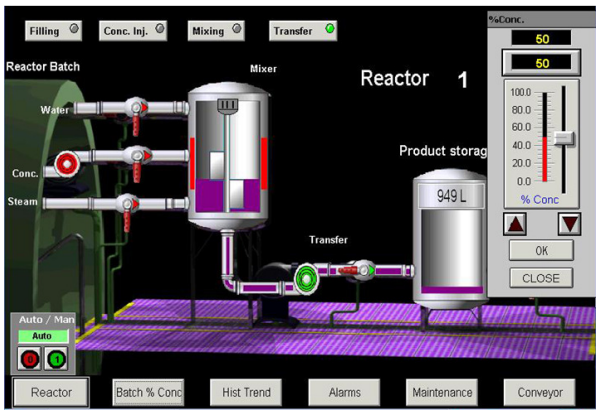


Figure 2-7 HMI (Wonderware, 2011)

2.2.7 SCADA software and hardware architecture

One distinguishes two basic layers in a SCADA system: the "client layer" which caters for the man to machine interaction and the "data server layer" which handles most of the process data control activities (Daneels & Salter, 1999). The data servers are PLC or RTU devices which act as a central data or communication devices.

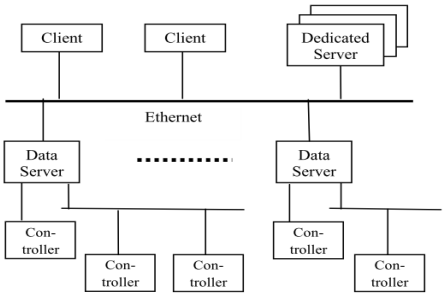


Figure 2-8 Typical Hardware Architecture (Daneels & Salter, 1999)

2.3 SCADA Industrial Protocols

2.3.1 Introduction

This chapter reviews the literature on industrial communication protocols implemented on SCADA systems. A protocol is a set of message formats and rules for exchanging messages. Selection of protocol depends mainly on system application, requirements and functions to be carried out. This chapter compares the industrial SCADA protocols for telemetry system application. Industrial protocols are commonly used in the utilities industry such as power, water, wastewater, oil and gas systems. The utility industry communication systems require high data integrity for communication and control reliability and data acquisition. The telemetry communication system periodically updates the system status. In SCADA systems the most common protocols are (DNP3, Modbus, IEC 61850 IEC and 60870-5-101) which will be presented in this chapter. SCADA systems rely on "event reporting" for system information.

2.3.2 Modbus

Modbus is a serial communication protocol published by Modicon in 1979 for use with its PLC's. It has become the industry de facto standard communications protocol in industry (Modbus organisation, 2006). The Modbus is accessed on the master/slave principle, the protocol providing for one master and up to 247 slaves (Clarke, Reynders, & Wright, 2003). Devices communicate using a master-slave technique whereby the master initiates the response and the slave devices respond with the requested data or action. Modbus devices usually include a register map to monitor, configure, and control module input and outputs. Modbus can operate in two modes ASCII Mode or RTU Mode. The following table 2-1 illustrates a typical Modbus message frame format.

Table 2-1 Modbus message frame format

Address field	Function Field	Data Field	Error Check field
1 byte	1 byte	Variable	2 bytes

The main advantage of Modbus is it offers interoperability which enables multi-vendor integration with the most purchasing options and at the lowest cost. Modbus provides simplicity, low overheads, error checking for transmission and communication errors by character framing, a parity check and by using a 16 bit Cyclic Redundancy Check. Modbus messages are transmitted as frames and are separated by silent periods. The major

disadvantage is that Modbus does not offer time stamped events which is critical in SCADA systems for monitoring and trending purposes.

2.3.3 Distributed Network Protocol (DNP3)

DNP was originally created by Westronic, Inc. (now GE Harris) in 1990 (IEEE Power & Energy Society, 2010). In 1993, the DNP 3.0 document set was released in the public domain. The DNP 3.0 is specifically developed for inter device communication involving SCADA RTU's, and provides for both IED to RTU and master-to-IED to RTU (IEEE Power & Energy Society, 2010). The protocol is designed to allow reliable communications for electrical utility.

DNP3 framing is based on the FT3 frame format as shown in Figure 2-9. DNP3 frame format is designed for high data integrity which uses cyclic redundancy checks to detect error in messages. DNP3 uses several modes Polled only, Polled (report by exception), Unsolicited (report by exception) which are spontaneous reporting based on changes in field data and uses minimal overheads however minimises data integrity. DNP3 is an open standard for implementation with RTU's, IEDs and master station vendors. The DNP3 protocol was designed for the electrical systems however it has been adopted by other utility industry applications.

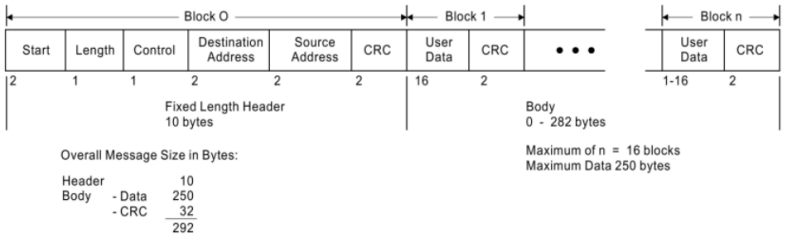


Figure 2-9 FT3 Frame Format (Clarke et al., 2003)

Its major advantage is in its bandwidth efficiency which is accomplished through class based data reporting**.** The master RTU executes a Class 0 poll to request for all static data points from an RTU. The data points generate events which are stored as Class 1, 2, 3 and raise a buffer flag which the master recognises when initiating a poll request for each class of data. This enables the system to be more efficient by preventing the repetitive polling of the all the system points. The designer controls the polling frequency and type of data depending on the application and specific requirements. The advantage of this protocol is that it enables highly

accurate time stamping which allows for accurate prediction of a field event changes. The protocol provides data frame checking for reliable data transfer. This protocol enables definition of Analogue input/outputs, Digital input/outputs as “data objects” or “points”. The disadvantage of DNP3 protocol is that it has very high overheads.

2.3.4 Kingfisher Series 2 Protocol

The Series 2 protocol frame format consists of a number of Header bytes, Argument/Data bytes and CRC bytes as per Table 2-2. The RTU transmits data as 2 byte hexadecimal numbers.

Table 2-2 Kingfisher Series 2 Frame Format

System ID	Target RTU	Number of characters	Initiating RTU	Via RTU	Message	Function code	Data Field	CRC
1 bytes	1 bytes	1 bytes	1 bytes	1 bytes	1 bytes	2 bytes	Variable	2 bytes

The header bytes consists of the following fields of one byte each

1. System ID – This is used for routing purposes
2. Target RTU – The RTU for which the message is meant for
3. Number of characters – Number of characters in the frame excluding System ID
4. Initiating RTU – The RTU transmitting the message
5. Via RTU – This can be used for indirect or relay messaging
6. Message - Message number in sequential order
7. Function code – This is the type of message refer to Appendix C
8. Data Field – This is the actual system data and consists of numerous bytes depending on the function code, refer to Appendix C.
9. CRC – Cyclic Redundancy Check which is designed for integrity check and

Data can be stored in the IO Array or local data registers. The communications system can access these data arrays using two types of message transfers – ‘data’ or ‘block’. When data transfers are used a single data value or multiple data values can be accessed from anywhere in the I/O or register array. When block transfers are used 64 data values are transferred in a message and the data transfer ends at the last non zero data where the remainder data values are all zero.

The system uses several modes for communication polling and exception reporting or spontaneous reporting. The major advantage is that it offers time stamped events which are required for trending and monitoring purposes.

The Kingfisher protocol uses two data acquisition techniques called polling and exception reporting. Its main advantage is that it supports time stamped events and time synchronisation between RTU's. It uses asynchronous communication and its frame format uses 1 start, 1 stop and no parity bits for every 8 data bits.

2.3.5 IEC 60870-5-(101 to 104)

IEC 60870-5-101 to 104 is a companion standard generated by the IEC TC57 for electric utility communication between master stations and RTUs and is implemented between substations to supervisory control centres**.** It supports unbalanced and balanced modes of data transfer. Data can be classified into groups. It uses several modes Polled only, Polled report by exception, unsolicited report by exception spontaneous reporting. Time synchronisation and time stamped events. It uses frame format IEC 101 which uses 1 start, 1 stop, 1 parity bit and 8 data bits which is suitable for asynchronous communication. This protocol has been specifically developed for electrical power systems and has therefore not been adopted by other utility industries hence it falls outside the scope of this project.

2.3.6 IEC 61850

The IEC 61850 standard defines communications between intelligent electronic devices (IEDs) in the electrical substation and the related system requirements. The objective was the development of an international standard for communication networks and systems in an automated electric substation. It is specifically designed for the sharing of high speed data protection information between protection devices and provides interconnection of substation devices on a high speed Ethernet network. (Mackiewicz, 2006) The abstract models defined divide the substation automation system functions in small entities called logical nodes. Based on their functionality, each logical node contains a list of data and data attributes.

The advantage of this protocol is fast transfers of events using (GSE) Generic Substation Event are defined for fast transfer of event data which is divided into GOOSE&GSSE. Services commonly used for communication within the whole substation are mapped to MMS (Manufacturing Message Specification). MMS is an application layer standard designed to support messaging communications between IEDs in a distributed system environment. Sampled data transfer schemes are also defined to handle transfer of sampled

values using sampled value control blocks. IEC 61850 is designed for communication between substation devices over high speed Ethernet network and is therefore only applicable to electrical power system substation applications and falls outside the scope of this project.

2.3.7 Summary

There are similarities between Modbus frame format and Kingfisher frame format however the major disadvantage is that Modbus does not allow time stamping. DNP3 is fast becoming the adopted protocol by utility industries. It provides efficient and reliable data transfer with object oriented defined variables and implements a polling structure based on static points and event class based (1-3) to enable efficiency of data traffic type however it has high protocol overheads compared to Modbus and Kingfisher S2 protocols. The Kingfisher S2 Protocol supports time stamping, polling, exception reporting techniques. Therefore the Kingfisher S2 Protocol was the selected protocol implemented in this system design since it offers time stamping over Modbus and enables high data integrity by using 16 bit CRC for error detection with less protocol overheads compared to DNP3.

2.4 Bandwidth Control

2.4.1 Introduction

A telemetry communication system is required to link the central control station to the remote field sites. The remote field sites must share the resource available to prevent fairness problems. This chapter presents a bandwidth allocation scheme for the telemetry communication system. Due to limited bandwidth capacity over the implemented single channel radio frequency an efficient control scheme is necessary for the system.

The telemetry communication system control of data traffic has similarities to many currently available fieldbus networks which adopt: 1) centralized access control of a master–slave system (e.g., FOUNDATION fieldbus, WorldFIP, MIL-STD-1553B,IEC/ISA fieldbus) and or 2) distributed access control of a token-passing system (e.g., Profibus, Arcnet, SAE HSDB) as their medium access control protocol (Hong, 2001).

There are two modes of operation for medium access control: balanced mode as shown in Figure 2-10 and unbalanced mode of operation as shown in Figure 2-11. Balanced transmission refers to the configurations where any station on a link may act as a primary which means it can initiate communication (Reynders et al., 2005). To avoid collisions (CSMA/CA) carrier sense multiple access/ collision avoidance could be utilised to coordinate

transmissions. Alternatively a full duplex system or an unbalanced mode could be implemented for the system design.

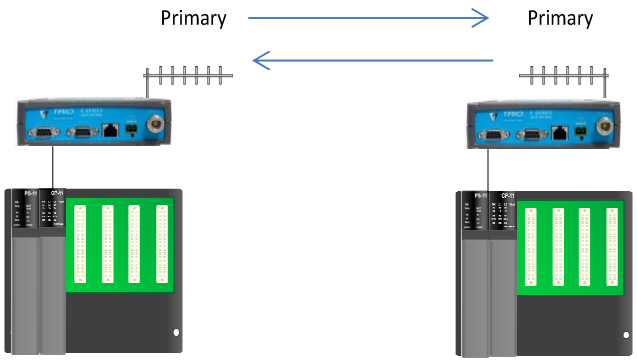


Figure 2-10 Balanced Mode Transmission

In unbalanced transmission only the master device can transmit primary frames. This technique is similar to a master/slave or token passing systems. This approach enables a more deterministic approach of designing the system and avoids the use of collision avoidance mechanisms. If there is a message fail the master will retry the transmit message called (Message Retries) within Kingfisher protocol until a valid message is received or a response time out occurs. The advantage of unbalanced communications is that there is no possibility of collision between controlled stations attempting to transmit information at the same time (Reynders et al., 2005).

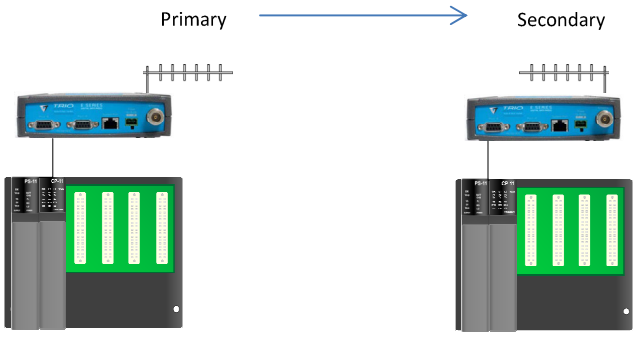


Figure 2-11 Unbalanced Mode Transmission

2.4.2 Bandwidth allocation schemes

(Hong & Kim, 2000) propose a bandwidth allocation algorithm in CAN protocol for distributed control system which give priority to real time data over event log data since DCS require fast system status and response time for plant operational requirements within a local area. Real time data is further sub divided into control data and event data. During the real time data interval control data are transmitted using the window scheduling algorithm. In the window-scheduling algorithm, real time data is divided by windows. Control data packets are transmitted through the windows. The length of a window is identical to the packet transmission time. Event data is given highest priority and transmitted during any interval.

To apply the bandwidth allocation algorithm the CAN bus, a method that divides the network bandwidth into real time and non-real-time intervals needs to be acquired. The system described here is balanced system. In the CAN protocol, this can be simply accomplished by marking an appropriate value in the "identifier field" of the CAN frame. In the case of event data, the first two bits of the 11 bit identifier field are set to '00'. The first two bits of control and non-real-time data are set to '01'and '10', respectively. '0' is (dominant bit) which is dominant to '1' (recessive bit) in the identifier field, event data have priority over control data, and control data take precedence over non-real-time data in transmitting. This means that non-real-time data cannot be transmitted until the real- time-interval is over, during the non-real-time interval, if any node generates control data, the network bandwidth is automatically resumed to the real-time interval.

This method however uses unbalanced mode of transmission whereby the outstation transmits event data with highest priority at intervals using CSMA/NBA – (carrier sense multiple access / non-destructive bitwise arbitration) mechanism.

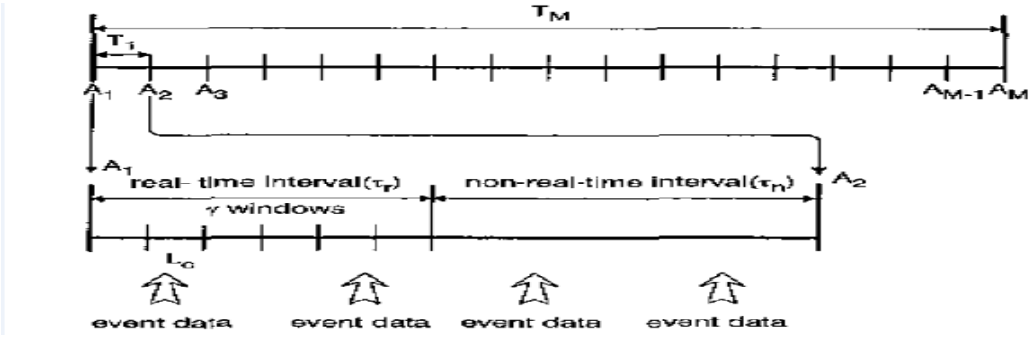


Figure 2-12 Bandwidth allocation scheme (Hong & Kim, 2000)

To satisfy the performance requirements for real-time data traffic and event log data an appropriate bandwidth allocation and traffic scheduling scheme was designed. (Hong, 2001) introduced a basic concept of bandwidth allocation scheme that can be applied to a cyclic-service fieldbus network. Figure 2-13 shows nodes connected over a bus network. Each node consists of a queue of time critical data, periodic data and time available data. This system operates in unbalanced mode whereby time critical data is transmitted during server vacation periods and the server polls the nodes for periodic and time available data.

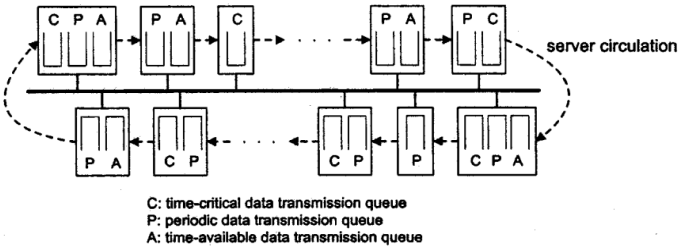


Figure 2-13 Data queue transmission (Hong, 2001)

Data generated from distributed control and automation systems are classified into real-time and non-real-time data (Hong & Kim, 2000). Therefore a similar approach was used to define the data traffic in the design of the telemetry control system. The data generated from remote field sites for this system were divided into three data traffic categories:

- critical real time and alarm data
- non critical real time data
- event log data

The bandwidth control is designed to give priority of critical real time and alarm data over the other data traffic by polling this data traffic frequently for real time system updates. Polling of real time data is a snapshot of the local process variables. The event log data has been allocated more bandwidth over real time data since SCADA systems are event driven systems requiring trending and monitoring.

This section presents a bandwidth allocation scheme for the three data traffic catergories using an unbalanced mode of operation for the telemetry control system design. This mode enables beter control of a large scale system from a central location.

2.4.3 Data addressing scheme for Kingfisher RTU's

Remote field site RTU's can be configured to "Update Register Blocks" within a local RTU memory or "Update Hardware Blocks" from analogue and digital modules. For this application it is assumed all analogue and digital values are stored in the local register addressing scheme.Therefore the Update Hardware Blocks will be configured to (NONE). The local register blocks will be updated and polled for data. A hexadecimal constant can be entered corresponding to the blocks to update. A maximum of 16 blocks each consisting of 64 registers gives a total of 1024 registers to be polled out of the total 2048 system registers which can be polled as per Table 2-3.

Table 2-3 Local Kingfisher RTU registers

Update Register Blocks Constants			
Block	**Registers**	**Ch**	**Constant**
1	#R1 to #R64	1	0001 Hex
2	#R65 to #R128	2	0002 Hex
3	#R129 to #R192	3	0004 Hex
4	#R193 to #R256	4	0008 Hex
5	#R257 to #R320	5	0010 Hex
6	#R321 to #R384	6	0020 Hex
7	#R385 to #R448	7	0040 Hex
8	#R449 to #R512	8	0080 Hex
9	#R513 to #R576	9	0100 Hex
10	#R577 to #R640	10	0200 Hex
11	#R641 to #R704	11	0400 Hex
12	#R705 to #R768	12	0800 Hex
13	#R769 to #R832	13	1000 Hex
14	#R833 to #R896	14	2000 Hex
15	#R897 to #R960	15	4000 Hex
16	#R961 to #R1024	16	8000 Hex

2.4.4 Real Time Data

2.4.4.1 Critical real time and alarm data

Critical real time and alarm data are priority for this system operation to gather system status updates. The proposed system is designed for the master station to poll this data periodically with highest priority using the (rx_data) communication block which minimises protocol overheads for a limited number of data points. Therefore all critical real time and alarm data can be allocated to (Registers #R1 - #R32) of the local registers addressing scheme per outstation RTU. This design enables the master station to poll all system remote stations with minmised system latency minimal protocol overheads thus optimising bandwidth utilisation.

2.4.4.2 Non critical real time data

The system remote field sites generate real time data which are required for process control and operational requirements within the local RTU. Therefore the proposed system allocates the data from (Blocks 1-15) of the local registers addressing scheme as per Table 2-3. This

configuration is within the outstation under Update Registers: 16#7FFF. This configuration minimises overheads during poll requests using the (rx_update) communication block.

2.4.5 Non Real Time Data

2.4.5.1 Event Log Data

Event log data is time stamped data which is required for the SCADA system for historical and trending purposes. This data is stored within the configured memory for event logs. Each event log is 12 Bytes and configuration can be calculated depending on the maximum number of event logs required for an outstation to store which is necessary to prevent old data from being over written since it has a circular buffer. Event log data can be limited for a poll request by controlling the (Maximum logs to upload) setting under the “Event Logs Controls”for the (rx_update) communication block.

The proposed system is designed for SCADA system which are event driven systems. The bandwidth allocation scheme is designed to periodically poll critical real time and alarm data. Secondly the event log data is limited to gaurantee bandwidth to non critcal real time data. The proposed system is designed to operate in the unbalanced mode whereby a primary request can only be made by the master station. Therefore this system can be considered a pure “polling system”. This scheme is designed to efficiently manage all data/traffic types by developing an efficient bandwidth allocation scheme hence optimising the system.

The communication system medium is designed to share the bandwidth resource amongst the remote field sites by dividing the service period up to the maximum limits specified to prevent fairness problems. This is not a fixed time therefore the master station can poll the next outstation when it reaches its configured polling limit. A periodic time slot is allocated to poll critical real time and alarm data from all remote stations in the system with the highest priority. After this time slots for real time data and event log data are allocated per remote site. The event log data is given priority over the non critical real time data in terms of bandwidth utilisation per remote station since SCADA systems are event driven systems.

The system also enables the SCADA to transmit commands and set points via the master station RTU to the specified remote field site therefore idle periods are forced into the system but kept to a minimal to enable the transparent write of this data over the radio link.

Figure 2-14 describes the bandwidth allocation design by over provisioning the bandwidth capacity for expected peak traffic load at a baud rate of 9600. The service period for each station is shown by the "Maximum service duration" which consists of the polling request from the master station of Blocks 1-15 for non critical real time data and 1000 event logs per remote site individually. The system was designed to poll critical real time and alarm data "periodically" from all remote stations thus enabling regular system updates. Polling of event logs have been limited per site to prevent fairness problems which is caused by serving each remote site exhaustively however the advantage of this technique is that it reduces switchover periods between remote sites. The service discpline for this system is gated, whereby only data present at the polling instant are served. The master station then polls the next site defined by the static polling table.

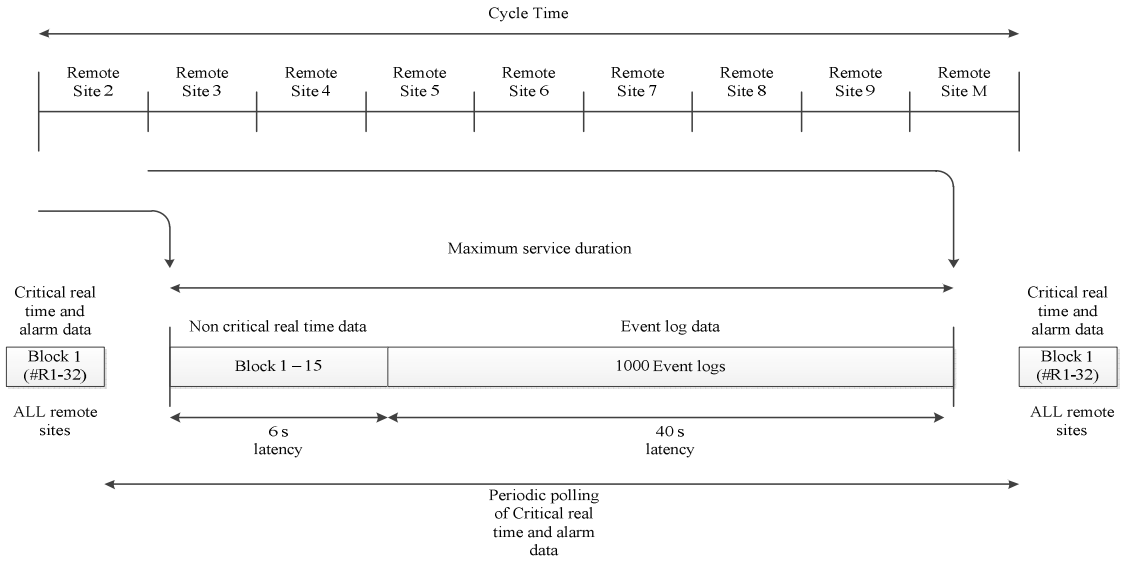


Figure 2-14 Implemented Bandwidth allocation scheme

The proposed scheme is designed to meet SCADA operational system requirements by periodically polling system status updates for real time data control purposes as well as event log data for monitoring. Since SCADA systems are event oriented systems the bandwidth slot for service periods per remote station have been designed with bias towards event log data. The hardware components used in this design were "Kingfisher Series 2 Remote Terminal Units" (Appendix A) which share a single channel radio frequency via "Trio E-Series Radios" (Appendix B) operating at generic frequency band of 450 MHz as the communication medium.

2.5 Polling Systems

2.5.1 Introduction

The term “Polling” originated with the polling data link control scheme in which the central computer interrogates each terminal on a multi drop communication line to determine whether or not it has data to transmit. The addressed terminal transmits data, and the computer then examines the next terminal (Haring, Lindemann, Reiser, & Takagi, 2000). A polling model is a system of multiple queues (1….N) attended by single or multiple servers in cyclic order (1, 2, 3……N, 1, 2, 3……N) this is also known as the basic model. Polling models and their variations appear not only in computers and communications but also in other fields of engineering such as manufacturing and transportation systems(Haring et al., 2000). Polling systems have been used to model a large variety of applications in which a single resource is shared among customers in N distinct queues (O. J. Boxma et al., 1991). In this application the telemetry communication system utilises the single resource being the half duplex radio channel which is shared amongst remote stations for transmitting data back to the control station. Two important performance measures of polling system are the mean polling cycle time that it takes the server to complete a cycle of visiting all the queues in the system, and the mean customer waiting time that it takes a customer from arrival to service start (Haring et al., 2000).

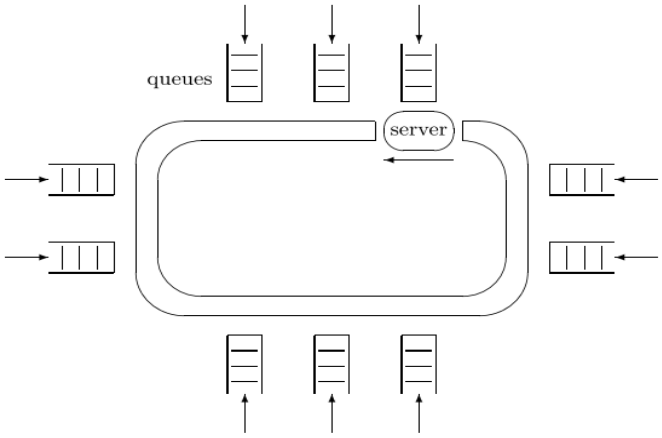


Figure 2-15 A polling system (Haring et al., 2000)

2.5.2 Polling System Surveys

There is an extensive amount of literature available on polling systems, the surveys of (Haring et al., 2000), (Levy & Sidi, 1990) and (Vishnevskii & Semenova, 2006). Most of these papers only provide approximations or focus on pseudoconservation laws. The pseudo conservation law is the weighted sum of the mean waiting times in a polling system. Such pseudoconservation laws can be used to obtain or test approximations for the individual mean waiting times and enable higher priority stations to be polled more frequently within a polling cycle.

(Levy & Sidi, 1990) describe the basic polling system shown in Figure 2-16. They summarise the analysis of polling models with variations and their applications. The customers are referred to as type i customers arriving into the queue. The arrival rate of customers to the queue is represented by the Poisson distribution with rate λ_i. The time for the data transfer is called the service time represented by β_i and represents the data transmission from the server to the terminal. The time for the server to change from serving one queue to the next is called the switch over period represented by the parameter S_i.

The polling model application in this system is with regards to the radio telemetry communication system which uses a single channel half-duplex mode of transmitting data between the master and multiple remotes sites using a time shared or bandwidth sharing method. In the designed system the server represents the master or base station which is responsible for data acquisition to central location. A queue represents a remote station responsible for local process control. The customers represent the data entry of process variables into the remote site. The time for the data transmission over the shared bandwidth from the remote site to the master station is called the service time represented by β_i. The switchover times are incurred when the server or master station moves from one queue to the next**.**

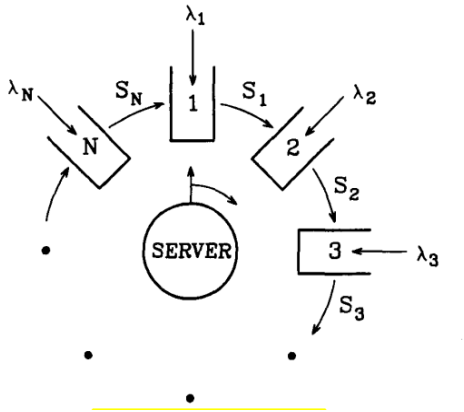


Figure 2-16 A cyclic polling system (Levy & Sidi, 1990)

Polling systems can be can be classified as discrete or continuos. Discrete polling systems are characterized by the number of queues, their capacity (the number of the waiting places), number of servers, processes of customer arrival and service, durations of server switchover between the queues, as well as the order and discipline of queue service (Vishnevskii & Semenova, 2006). The order in which a server polls the queues can be static or dynamic.

Static polling orders are classified as (cyclic, random, priority and periodic). Periodic polling is characterised by the use of a polling table which is the method implemented in telemetry systems designed by CSE - W. Arthur Fisher. Periodic polling uses time slots to share or allocate the bandwidth resource amongst multiple stations.

Dynamic order is whereby the order of service to queues changes dynamically. The advantage of dynamic order is that they can change their service requirements depending on the system state and thus can be used to improve system performance. The disadvantages of such orders is that they require information gathering during operation and that they are hard to analyse (Levy & Sidi, 1990).

With a polling system there are several design decisions that the system designer needs to make. One of these is how many customers to serve in each visit to the queue. There are four service disciplines most commonly used in polling systems. Gated service policy is whereby only those customers present at the beginning of a server visit (polling instant) at a queue are allowed to be served. New customer arrivals to the queue are deferred until the next visit. In the case of exhaustive service the server serves the queue until the queue is completely empty. This means that customers arriving at the queue while the server is at the queue are served during the current service period. For the 1-limited service policy at most 1 customer

is served in each queue for a cycle. For k–limited the number of customers served by the server is limited.

This Kingfisher S2 protocol implements the gated service discipline for the communication blocks used in the software development. The exhaustive service discipline is debated as being more efficient in terms of the data served over a period and this method also minimises switchover times however the major disadvantage with this service discipline is fairness whereby a high traffic load station could monopolise the shared bandwidth resource.

2.5.3 Mathematical Analysis of Polling Systems

This section reviews the mathematical methods used for the analysis of polling systems. A majority of the research has focussed on analysis issues, namely deriving mathematical procedures for deriving the performance measures of these models. (Baker & Rubin, 1987) and (Choudhury, 1990) derive the mean waiting times in polling systems with a polling table and exhaustive or gated service using a set of linear equations. The model under consideration consists of a single server and multiple queues to be serviced (Q1…….QN). The server visits the queues in a fixed order specified by a periodic polling table in which each queue occurs at least once.

(Baker & Rubin, 1987) derives exact results for a polling system with a polling table with exhaustive service. Stations are given higher priority by being listed more frequently in the polling table. The model describes N stations in the system with a token passing method using a polling table. They find the mean waiting times by solving a set of simultaneous equations. (Baker & Rubin, 1987) state that "partial symmetry in the polling table and the station characteristics can be used to significantly reduce the number of equations which must be solved" (p.283). They also present the reduced equation set for a two-priority class system and apply the results to a large token-passing bus network in which a few nodes account for a substantial portion of the network traffic. They show that in the overall average message waiting time can be significantly reduced by using priority polling: average waiting time at the high-priority nodes have large reductions in return for a smaller increase at low-priority nodes.

(Choudhury, 1990) derives the exact results on the mean waiting time in a non-symmetric polling system with general service order table and gated service discipline. Analysis is to provide priority service by evaluating the mean waiting time and adjust stations polling

frequency by using a general service order table. This paper also defines the pseudo cycle times and the relationship between the cycle times and the pseudo cycle times at each station. It derives the generating functions for the waiting times and the number of messages at the stations and at the pseudo stations are obtained as functions of pseudo cycle times. Equations are derived for the case of zero first two moments of the switch-over time at every station. The numerical results quantify the significant reductions in the waiting time and the queue occupancy at a station with high server utilizations by listing it more frequently in the polling table.

A pseudoconservation law for polling tables have been derived by (O.J.; Boxma, Groenendijk, & Weststrate, 1990) and derive an exact expression for polling tables with either exhaustive or gated service. Pseudoconservation laws (weighted sum of the mean waiting time in a system) at various queues. This can be used to obtain or test approximations for individual mean waiting times, and generally to provide insight into the behaviour of polling systems and can be used towards system optimisation by determining how often a station should be visited compared to other stations.

2.5.4 Efficient visit orders and optimisation of polling systems

This article states that "Polling systems have been used to model a large variety of applications in which a single resource is shared among customers accumulating in N distinct queues" (O. J. Boxma, Levy, & Weststrate, 1993) focus on efficient sever visit orders as part of developing an optimal polling table that minimises the mean total workload for a polling system. (O. J. Boxma et al., 1993) stated that "optimisation in polling systems is a subject which has so far received very little attention in queuing literature." (p.104) and thus focus on static optimisation of polling systems.

Only exhaustive and gated service disciplines were considered for this system. This article states that polling systems arise in the modelling of computer, communication and production networks where several users compete for access to a common resource. This paper proposes three step methodology to develop an optimal polling table. Step 1 presents the random polling approximation and lower bound approximations which are considered in this system to determine the queue visit frequencies. Step 2 is to determine the polling table size based on these frequencies and the number of visits to each queue. Step 3 proposes a good polling order for the queues and introduces the golden ratio policy to evenly spread the visit frequencies per queue. (O. J. Boxma et al., 1993) states that "TDM systems are very similar

to polling models" (p.110) he also refers to using weighted TDM whereby the weight refers to the frequencies with which time slots are allocated to stations.

This article proposes the concept of optimisation of a shared communication resource amongst multiple stations by polling high data traffic stations more frequently than lower data traffic stations. Thus the polling frequency is considered to be the most important step when applying this methodology. (O. J. Boxma et al., 1993) states that "the effect of the table size does not have any significant effect on the results" (p.115) and also mentions that "the crucial steps are therefore (1) and (3) to which we devote this section." (p.115) both random polling approximation and lower bound approach perform well for determining the visit frequencies. This paper states that the table size has no considerable effect on the system performance. The golden ratio was applied to the visit frequencies and this coincides with the optimal values from various table orders by taking the minimum value for the mean workload. This article proves that the golden ratio can be used to determine the visit order for the calculated frequencies based on the approximations.

2.5.5 Optimisation of polling systems

(O J; Boxma, Levy, & Westrate, 1989) describe an approximate methodology for determining an optimal polling table consisting of three steps to reduce the mean total system workload.

Step 1 Determine good visit frequencies per queue. Step 2 Determine the table size M and the number of occurrence of each queue in the table. Step 3 Determine good ordering of the queues. The random polling method versus the periodic polling method are compared in this paper and both apply the three step methodology for optimisation.

First determine relative visit frequencies f_i, using the square root assignment for polling tables. i=1,…..N for the visits of the different queues which is described as the most critical part of developing the table. Secondly determine the size of the polling table M and the number of visits m_i to be given in a cycle to Q_i, i=1,…..N. Thirdly determine the specific order by which to arrange the visits assigned in step for determining the polling table. Boxma, (Boxma, Levy, et al., 1989) specify that the most critical step in deriving efficient polling tables is determining the visit frequencies. Determining the size of the polling table does not

seem to be critical and a table consisting of up to several tens of queues will usually be a good choice.

This article introduces the Golden ratio procedure to evenly space the distance between the server visiting stations to enable a form of evenness within the system at the same time fairness of all stations allocation to bandwidth. The rounding off procedure is mentioned as well as the Golden Ratio policy (GR) to evenly spread the visits of the queues over the cycle. The mean workload was also calculated for all tables of reasonable size and table entry vector. This combined procedure can be used to "optimise" operational performance by reducing the mean workload. These articles focus on the exhaustive and gated service disciplines.

They propose minimising the mean workload in the system or weighted sum $\sum^{N}_{i=1} \rho_i EW_i$ a weighted sum of the mean waiting times were the weights are the traffic loads ρ_i of the queues. The results prove that the square root assignment performs better than the random polling system. Step 2 table size was not tested thoroughly since it was stated that this is the least important step. The Golden ratio was applied and performs satisfactorily for this system.

This article confirms that the Golden ratio procedure is proved to optimise system performance. It also presents the important point in optimisation of polling systems whereby there is a trade off between efficiency and fairness from the point of view of minimising workload in the system it may be efficient to visit heavy traffic queues frequently and for longer periods of time therefore efficiently allocating bandwidth resource however this may be unfair to the low traffic queues. Step 3 refers to sharing the communication channel amongst N transmission stations for a conflict free distributed protocol. TDM systems are similar to polling systems, however they use fixed time slots whereby each station is visited for a fixed period regardless of if it has messages present.

2.5.6 Efficient visit frequencies for polling tables: minimisation of waiting cost

(O. J. Boxma et al., 1991) stated that, "In spite of the massive research effort in this area, very little work has been devoted to the issue of how to efficiently operate these systems" (p.133). Prioritisation of remote stations affects the system performance which is controlled by the periodic polling table. (O. J. Boxma et al., 1991) stated that, "Almost no work has been done in the optimization of these models, namely on the problem of how to operate polling systems

efficiently in order to improve their performance. As a result, designers of these systems have not been equipped with effective tools and guidelines for their efficient operation." (p.134). Thus (O. J. Boxma et al., 1991) focus on efficient operation of polling systems by addressing the problem of how to construct an efficient polling table with the objective to minimise the mean waiting cost of the system. This article aims at finding a polling table that minimises the mean waiting cost for a customer using the objective function $\sum^{N}_{i=1} \lambda_i c_i EW_i$ (p.134) which is described by λ_i which is the arrival rate at queue i and c_i is the penalty on the system in terms of the cost of waiting one unit of time at queue i.

An approximate approach method based on lower bound approximation and on mean delay approximations were used to derive the number of visits frequencies to queue i. The "Golden Ratio" procedure is used to evenly space queue visits according to the derived queue visit frequencies for a cycle. This article focuses on the visiting frequency to a queue since this is the most critical aspect of polling tables and uses the cost as a performance measure to test and compare the approximations. This article states that the total number of visits in a cycle period is the sum of the visit frequencies to all the queues.

The approximations were tested by varying the following parameters: (Arrival rates, Effect of switchover parameters, effect of cost parameters, effect of service time parameters, effect of the system size and effect of mixed parameters) and comparing the 'cost' performance with the optimal pattern. Mean delay approximations perform better than lower bound approximations for the gated systems. This article extends the work conducted for service policy which is exhaustive, gated, limited-1, or a mixture of those while (O. J. Boxma et al., 1993) and (O J; Boxma et al., 1989) only exhaustive and gated service were considered. This article states that the visit frequencies are the most critical step in deriving efficient polling tables.

Therefore the "mean delay approximation" was implemented in the design of the telemetry control system to develop an efficient polling table thus optimising the bandwidth utilisation.

Chapter 3

Empirical data modelling

3.1 Introduction

This chapter presents the empirical data modeling of the system response time for data transmission over the half duplex radio frequency using Trio E-Series radio modems. The results were used to compare the efficiency of the Kingfisher S2 communication blocks within the ladder logic programming language and develop an efficient polling strategy for data traffic as well as establishing a service duration period for sharing bandwidth amongst the remote stations. The service duration is the time slot allocation of the bandwidth for a remote site to transmit data to the master station before moving to the next site.

3.2 The OSI model

The Kingfisher S2 protocol is based on the OSI model. The international standard for Open Systems Interconnection (OSI) model was developed by the International Standards Organization and provides a universal framework for devices to communicate.

The OSI reference model separates the network communication process into seven simple layers (Deal, 2008). Each layer builds upon the layer below and provides services for the layer above. This layered approach allows a structured approach for protocol development and implementation and interoperability.

Table 3-1 describes the OSI model from layer 1 through to layer 7. The first three layers occur on the network and are controlled by switches, routers, and similar devices. A network switch can only distribute packets by using MAC addresses, so it needs to only implement layers one and two. A simple router can route packets using only their IP addresses, therefore it operates on layers one through three. A web server or a laptop computer runs applications, so it must implement all seven layers.

Messages or data are transferred as bytes and pass from the application layer down to the physical layer. The data acquires additional header information at each layer and tells the next layer what to do with the data. At the receiving end the data travels from the physical layer up to the application layer and each piece of the header is removed as it progress threw

each layer. The OSI model is internationally recognized and is regarded as the network model. It provides a framework for manufacturers and network protocol implementers that can be used to build networking devices that interoperate with each other.

The Kingfisher S2 protocol is based on the OSI (Open Systems Interconnection) model and is designed to achieve communication between RTU's and computer for data transfer, configuration and diagnostic purposes.

Table 3-1 OSI Model

Layer	Name	Description
7	Application Layer	The human sits above this layer, interacting with the application. The Application Layer is the layer that file transfers and message exchanges occur. HTTP, FTP, and SMTP are all application layer protocols.
6	Presentation Layer	The Presentation Layer deals with data representation, before it reaches the application. This would include encoding, data compression, formatting checks, byte ordering, etc.
5	Session Layer	The session layer manages the communications between computers and sessions between applications. This is the layer at which the organization and synchronization of the data exchange. NetBIOS and RPC are two examples of a layer five protocol.
4	Transport Layer	The Transport Layer provides a method of reaching a particular service on a given network node. Examples of protocols that operate at this layer are TCP and UDP. Some protocols at the transport layer (such as TCP) ensure that all of the data has arrived at the destination, and is reassembled and delivered to the next layer in the proper order. UDP is a "connectionless" protocol commonly used for video and audio streaming.

3	Networking Layer	This layer controls the data flow or routing functions eg. router
2	Data Link Layer	This layer provides the controls of the physical addressing and method to transfer data as frames between nodes over the physical medium as well as error detection. Common examples of data link protocols are Ethernet, 802.11(a, b, g) and Token ring protocol. This is also known as the Medium Access Control Layer.
1	Physical Layer	This layer represents the actual physical medium over which communication occurs and for example radio frequency, fiber optic, copper CAT5 cable.

3.3 Protocol Frame Format

The OSI model provides a framework for which a specific protocol may be defined. A protocol defines the frame format which typically consists of a synch characters, source and destination bytes followed by the data bytes and ending with error checking byte.

Table 3-2 Typical frame format

Synch char	Destination address	Source address	Data to be transmitted	Error detection bytes

3.3.1 Kingfisher S2 asynchronous communication

The RTU's were configured for communication over RS232 serial link. Investigation proved that the RS232 data frame structure uses 1 start bit, 1 stop bit and no parity bits. This was found through investigation refer to Table 3-3.The first bit is the start bit then the data bits are transmitted starting from the least significant bit (LSB) first to the most significant bit (MSB). After the last data bit of data there is no parity bits and one stop bit. The investigation proves that the Kingfisher RTU uses asynchronous communication with 1 start bit, 1 stop bit for every 8 data bits and uses no parity bits. Table 3-4 describes (Rx_data) communication block using the communication analyzer via Toolbox 32 software. Figure 3-1 shows the screen shot of the data transmission viewed from the oscilloscope. The data frame structure is analyzed in Table 3-3; refer to Appendix F for the oscilloscope screenshots of the bit signal representation and Appendix O for the Kingfisher Series 2 Protocol format.

Table 3-3 (Rx_data) Communication block data frame analysis

									Logic	Voltage								LSB	Least significant bit	
									0	+V								MSB	Most significant bit	
Command #17 Send Multi Data									1	0V				**(Waveform 1)**						
	MSB							**LSB**			**LSB**								**MSB**	
Hex										Start bit										Stop bit
AE	1	0	1	0	1	1	1	0		0	0	1	1	1	0	1	0	1	1	
01	0	0	0	0	0	0	0	1		0	1	0	0	0	0	0	0	0	1	
0D	0	0	0	0	1	1	0	1		0	1	0	1	1	0	0	0	0	1	
02	0	0	0	0	0	0	1	0		0	0	1	0	0	0	0	0	0	1	
02	0	0	0	0	0	0	1	0		0	0	1	0	0	0	0	0	0	1	
E0	1	1	1	0	0	0	0	0		0	0	0	0	0	0	1	1	1	1	
11	0	0	0	1	0	0	0	1		0	1	0	0	0	1	0	0	0	1	
01	0	0	0	0	0	0	0	1		0	1	0	0	0	0	0	0	0	1	
10	0	0	0	1	0	0	0	0		0	0	0	0	0	1	0	0	0	1	
00	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	1	
00	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	1	
09	0	0	0	0	1	0	0	1		0	1	0	0	1	0	0	0	0	1	
34	0	0	1	1	0	1	0	0		0	0	0	1	0	1	1	0	0	1	
B5	1	0	1	1	0	1	0	1		0	1	0	1	0	1	1	0	1	1	
Command #16 Get Multi Data														**(Waveform 2)**						
	MSB							**LSB**			**LSB**								**MSB**	
Hex										Start bit										Stop bit
AE	1	0	1	0	1	1	1	0		0	0	1	1	1	0	1	0	1	1	
02	0	0	0	0	0	0	1	0		0	0	1	0	0	0	0	0	0	1	
0B	0	0	0	0	1	0	1	1		0	1	1	0	1	0	0	0	0	1	
01	0	0	0	0	0	0	0	1		0	1	0	0	0	0	0	0	0	1	
01	0	0	0	0	0	0	0	1		0	1	0	0	0	0	0	0	0	1	
0E	0	0	0	0	1	1	1	0		0	0	1	1	1	0	0	0	0	1	
10	0	0	0	1	0	0	0	0		0	0	0	0	0	1	0	0	0	1	
01	0	0	0	0	0	0	0	1		0	1	0	0	0	0	0	0	0	1	
10	0	0	0	1	0	0	0	0		0	0	0	0	0	1	0	0	0	1	
00	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	1	
67	0	1	1	0	0	1	1	1		0	1	1	1	0	0	1	1	0	1	
F7	1	1	1	1	0	1	1	1		0	1	1	1	0	1	1	1	1	1	

Table 3-4 (Rx_data) Communication block command structure

```
Comms Analyser : RTU 1, port 3
Rx: 11:12:51: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 17 = Send Multi Data
AE 01 0D 02 02 E0 11 01 10 00 00 09 34 B1

Tx: 11:12:51: init:  KF2 Message, Dest 2, Src 1, Via 1, Command# 16 = Get Multi Data
AE 02 0B 01 01 0E 10 01 10 00 67 F7
```

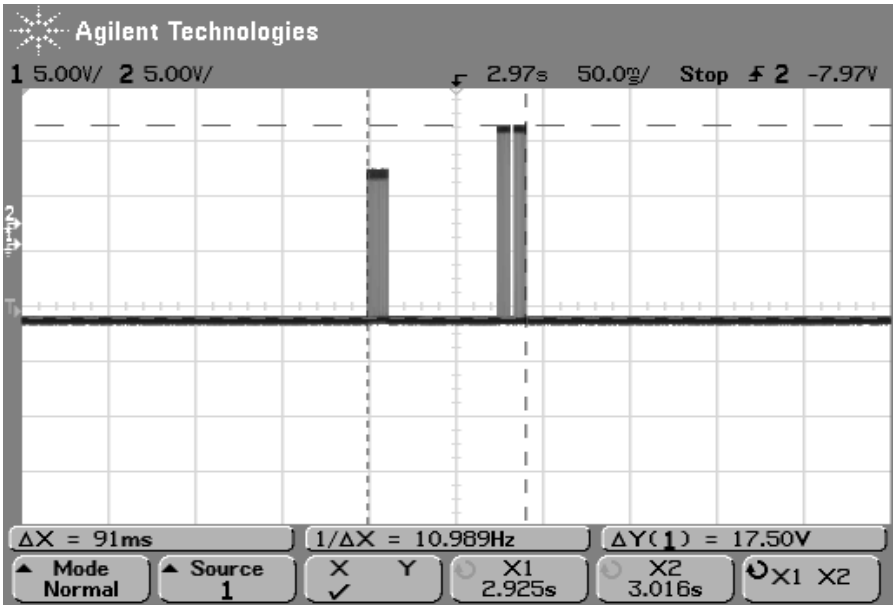


Figure 3-1 (Rx_data) communication block from oscilloscope

3.4 Kingfisher S2 communication blocks

The transmit data (Tx_data) communication block can be used to implement the exception reporting technique within the remote site to transmit data to the master station. It is designed to transmit up to 32 (16 bit) registers to a destination RTU.

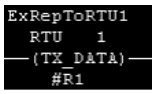


Figure 3-2 (Tx_data) communication block

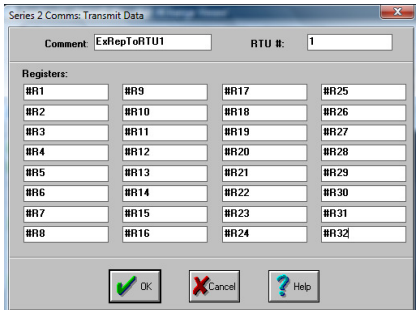


Figure 3-3 (Tx_data) communication block configuration

This communication block uses two commands. (Refer to Appendix C Function Codes)

Table 3-5 Function code for (Tx_data) block

Function code	Command	Data (Bytes)
17	Send Multi Data	(4*R) + 19
00	Simple Ack	

Note: R is the number of registers

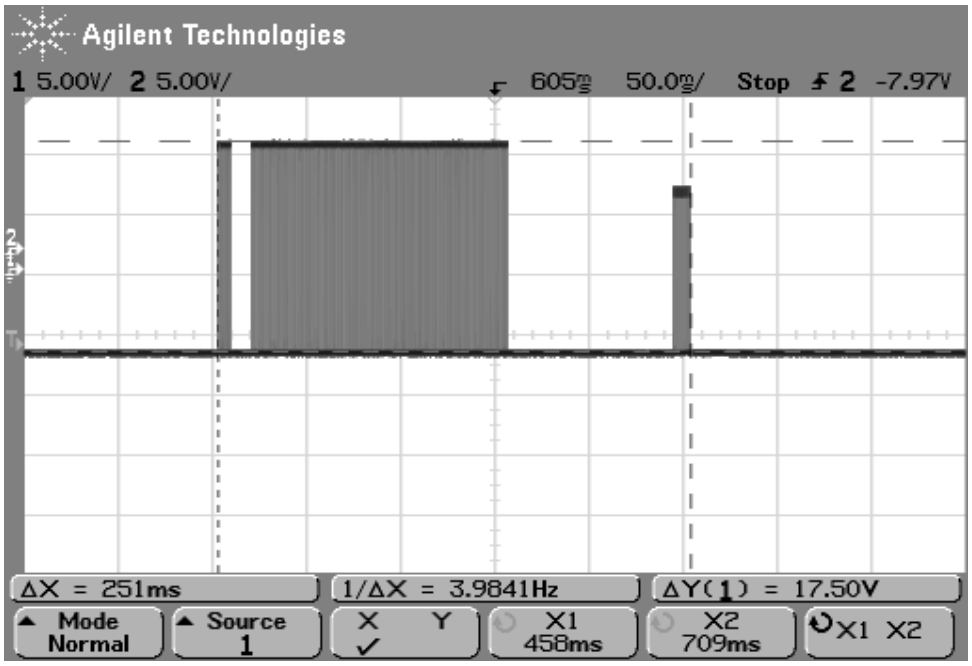


Figure 3-4 Transmission of 32 registers using (Tx_data) block

The Receive data (Rx_data) communication block is a limited poll request for real time data of up to 32 (16 bit) registers from a target RTU to an initiating RTU.

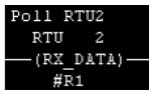


Figure 3-5 (Rx_data) communication block

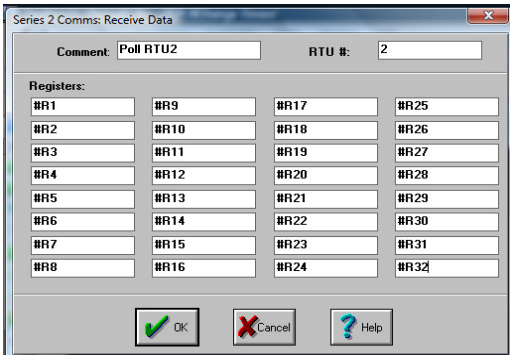


Figure 3-6 (Rx_data) communication block configuration

This communication block uses two commands. (Refer to Appendix C Function Codes)

Table 3-6 Function code for Rx_data block

Function code	Command	Data (Bytes)
16	Get Multi Data	(6*R) + 20
17	Send Multi Data	

Note: R is the number of registers

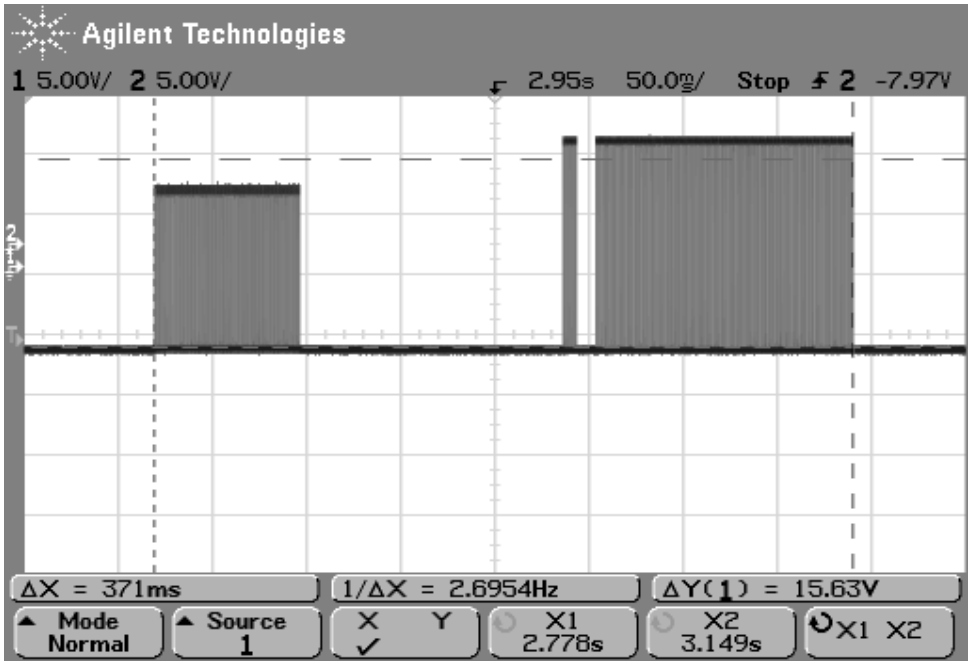


Figure 3-7 Transmission of 32 registers using (Rx_data) block

The (Rx_update) communication block is a poll request for real time data with a maximum transmission capacity of 1024 (16 bit) registers from a target RTU to an initiating RTU. Figure 3-9 “RTU’s to update” shows the configuration of polling real time data from multiple RTU’s.

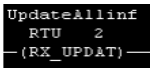


Figure 3-8 (Rx_update) communication block

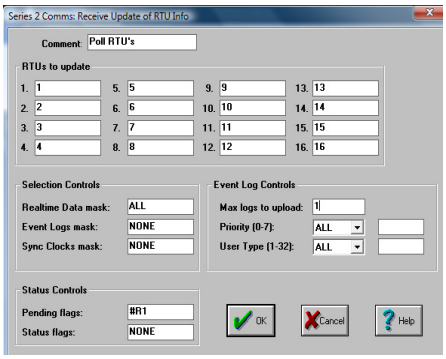


Figure 3-9 (Rx_update) block configuration

This communication block uses four commands.

Table 3-7 Function code for (Rx_update) block

Function code	Command	Configuration (ALL:16#FFFF) Data (Bytes)	Configuration(<16 Blocks) Data (Bytes)
26	Request RTU Update	148 + (24xBlocks) + (2*R)	20 + (28*Blocks) + (2*R)
15	Send RTU Update		
12	Get Data Block		
13	Send Data Block		

Note: Configuration within the outstation. ALL=16#FFFF. This configuration affects Command #15) Send RTU Update: whereby a status update of ALL 32 Blocks are transmitted. “R” is the number of registers and “Blocks” are groups of 64 Registers.

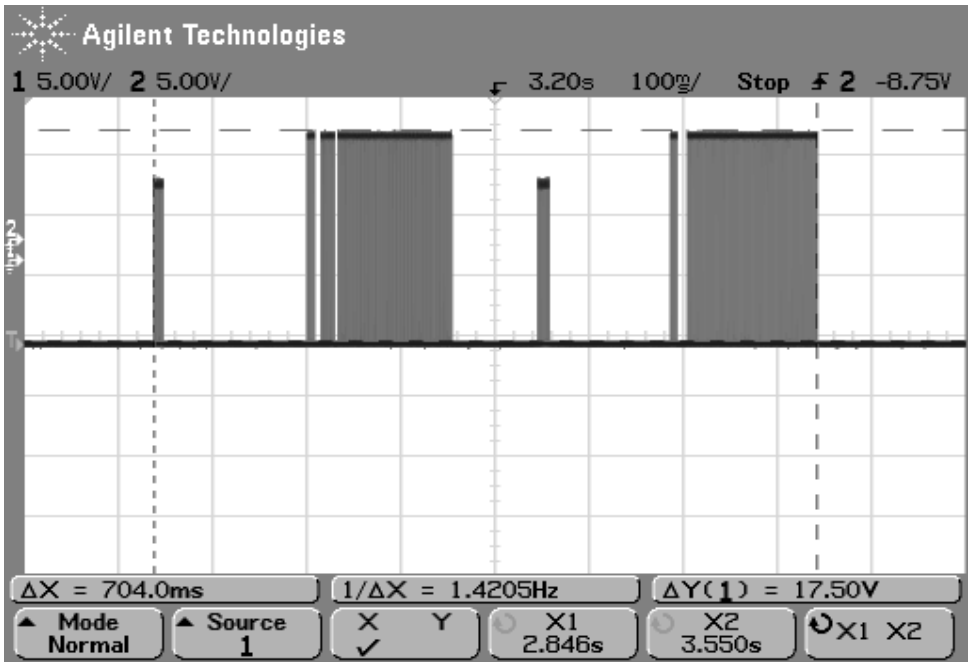


Figure 3-10 Transmission of 16 blocks using (rx_update)

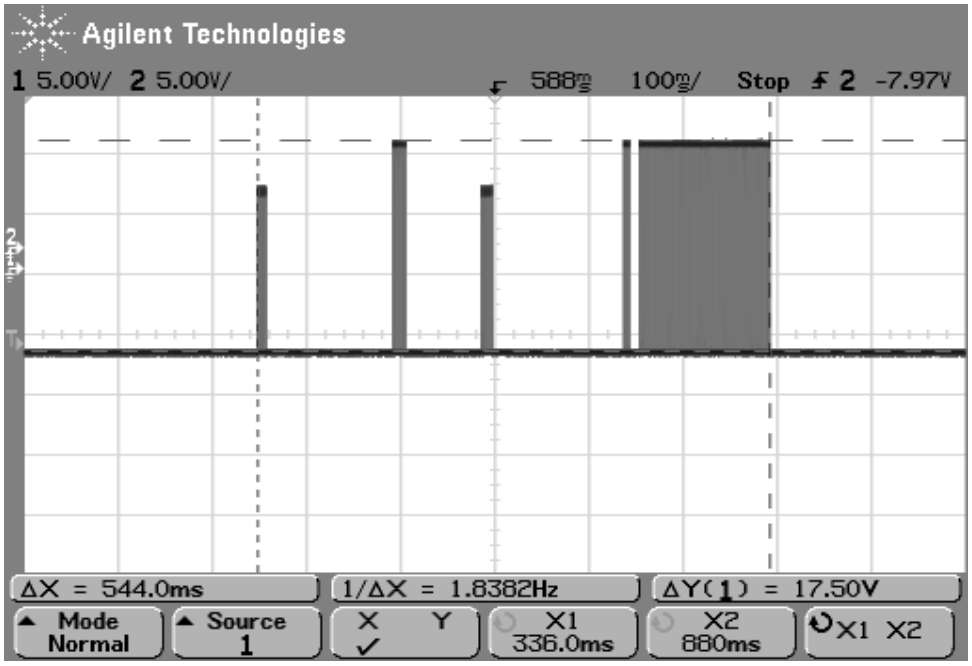


Figure 3-11 Transmission of 1 block using (rx_update)

The (Rx_update) communication block can also be used to poll non real time event log data up to the “Max logs to upload” configuration shown in figure 3-12.

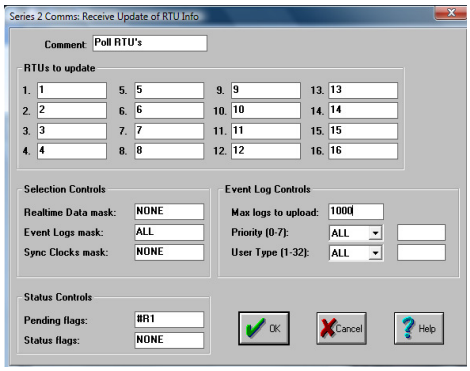


Figure 3-12 (Rx_update) driver configuration (Event log data)

Table 3-8 Function code for Rx_update (Event log data)

Function code	Command	Data(Bytes)
80	Request Event Logs (Max request of 10 logs)	30 + (12*Event logs)
81	Send Event Logs	

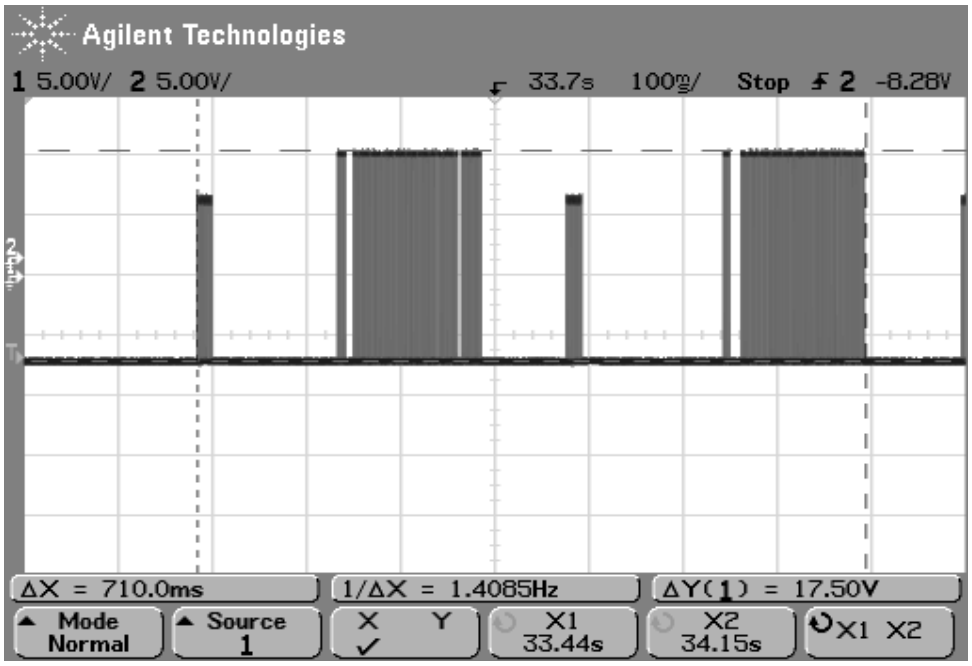


Figure 3-13 Polling 20 event logs using (rx_update) block

3.5 Empirical data modelling

The aim of the empirical data modelling was to test the derived formulas of Kingfisher S2 communication blocks to accurately predict system latency over a point to point communication link using Trio E Series serial radio modems see Appendix B and Kingfisher S2 RTU's see Appendix A. Figure 3-14 shows the break out box used to enable the oscilloscope to be connected to transmit and receive pins of the Master RTU.

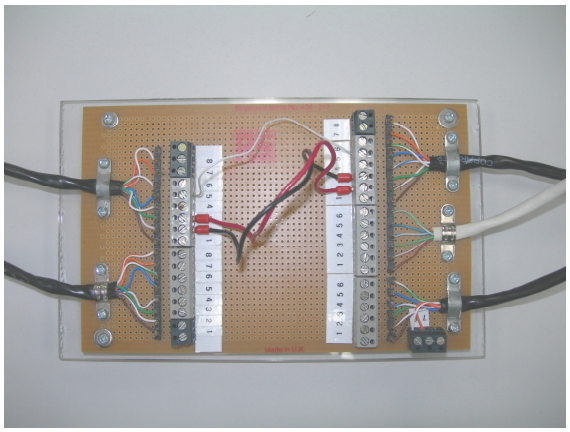


Figure 3-14 Break out box

3.5.1 Radio modem configuration – Trio E-Series

Figure 3-15 and Figure 3-16 show the radio configuration. The baud rate is set to 9600 with no parity, 8 data bits and 1 stop bit. The operating radio frequency is between 450MHz to 460MHz.

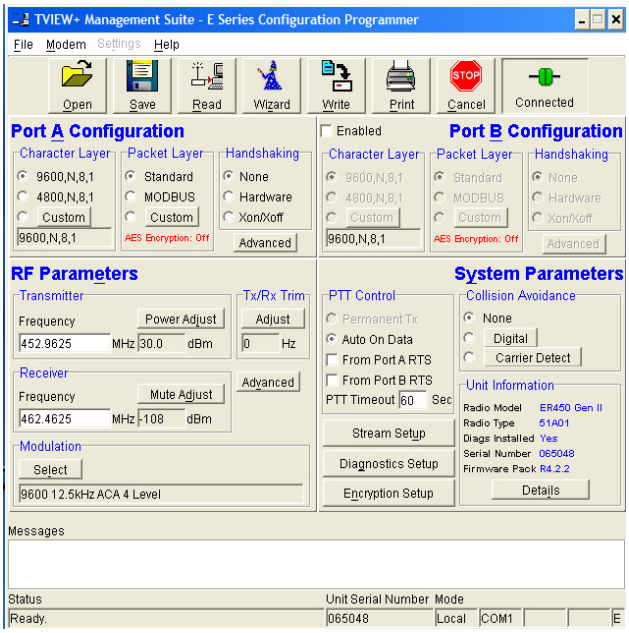


Figure 3-15 Master station radio configuration

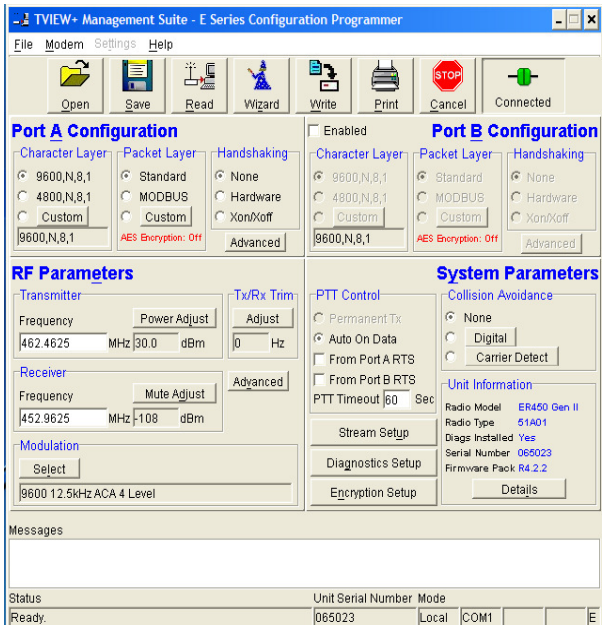


Figure 3-16 Remote station radio configuration

3.6 Empirical Data Model Analysis

The analysis compares the Kingfisher Series 2 Protocol communication blocks by dividing the data into real time data and event log data. Real time data is required for remote site status monitoring and control of process variables and is stored in the local RTU register addressing scheme which are 16 bit registers. Event Logs are required for time stamping of system events and this data is stored in the local RTU memory. The (rx_data) and (rx_update) polling techniques are poll requests transmitted from the base station to the remote stations for real time data values. The (tx_data) is typically used to transmit real time data values from the remote site to the base station. The (rx_update) communication block can also be configured to poll event logs. Event logs are "time stamped data" generated by the local RTU and stored in the CPU memory.

3.6.1 Real time data

Real time data is required for real time process control. The Kingfisher Series 2 RTU data supports the following data types, Floating point (32-bit), Long (32-bit), Signed Integer (16-bit), Unsigned Integer (16 bit) and Binary (1 bit of a 16-bit word).

3.6.2 Event log data

Event logs are time stamped data of an event and are required for system monitoring and trending. An event is something of significant change for example a change of state or threshold value which has been exceeded. When event log data is transmitted from one RTU to another the data is stored in its memory. Typically 1KB per outstation is sufficient. Event logs are generated from the RTU ladder or as System Logs. System logs are predefined logs and are generated for predefined events like Warm Start, Cold Start, and Set Real time Clock.

3.7 Serial link vs. Radio link system latency comparison at 9600 Baud

Figure 3-17 shows a comparison of serial data transmission over two physical mediums. The first configuration was conducted using serial RS232 point to point data link over CAT5 cable. The second configuration was conducted using serial radio modem interface over point to point data link. The result proves that the (rx_update) communication block has an average transmission delay of 51% over the radio link due to the system response time at 9600 Baud rate refer to Appendix T.

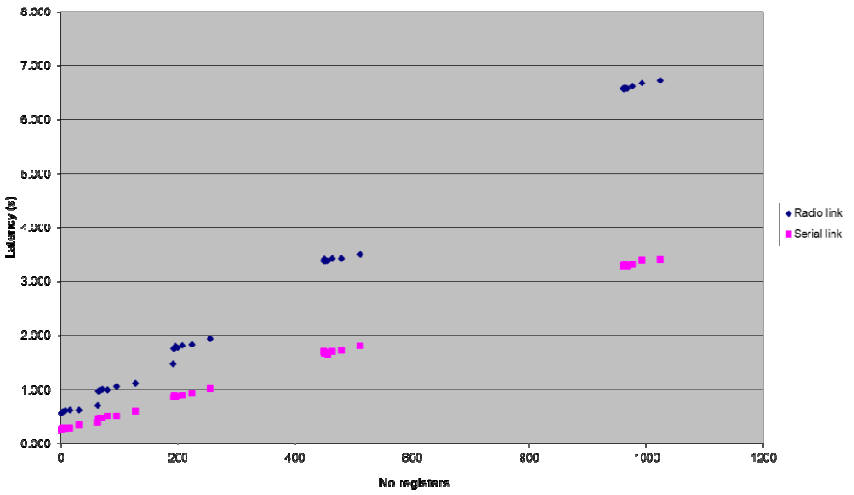


Figure 3-17 Serial link vs. Radio link latency at 9600 Baud rate

3.8 Investigation of baud rate on the system latency

The aim of the investigation was to compare all communication blocks with the configured baud rates and compare the system latency. From the results of the baud rate the system designer can make decisions for the type of radio equipment considering cost requirements for the network Table 3-9 shows a comparison of the effect of baud rate versus the system latency for transmitting 32 Registers via (rx_data) and (tx_data) communication blocks. The results show that the (tx_data) overall has lower latency at all configured baud rates due to less protocol overheads than the (rx_data) communication block.

Table 3-9 Baud rate vs. system latency for (rx-data) and (tx_data) communication blocks

Baud rate	Tx_data Latency (s)	Rx_data Latency (s)
9600	0.251	0.371
4800	0.391	0.568
2400	0.707	1.017
1200	1.33	1.93

The (rx_update) block was tested under two configurations for real time data and event log data. Table 3-10 show that the configuration (16#7FFF) reduces the system latency for real time data. This is due to reduction in data transmission from 16 Blocks (ALL) to 15 Blocks (16#7FFF), and this configuration also reduces the overheads for command 15) “Send RTU Update” whereby the configuration (ALL) transmits an update of all blocks from local register array shown in Figure 3-18 Local register array blocks (40-5F) which is a total of 32

Blocks and the configuration (16#7FFF) transmits an update of the requested blocks only (40-4E) which is a total of 15 Blocks. Table 3-11 shows the system latency to transmit 20 event logs over the configured baud rates using the (rx_update) communication blocks. The results from all communication blocks configured at different baud rates proves that the system latency is non linear and performs considerably slower at lower baud rates.

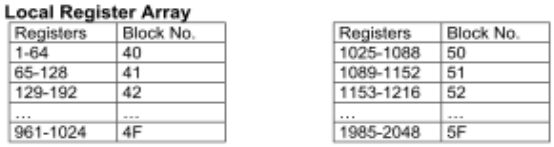


Figure 3-18 Local register array blocks

Table 3-10 Baud rate vs. system latency for (rx-update) block of real time data

Baud rate	Config (16#7FFF) Latency (s)	Config 16#(FFFF) Latency
9600	6.282	6.724
4800	8.468	9.171
2400	13.282	14.329
1200	23.56	25.675

Table 3-11 Baud rate vs. system latency for (rx_update) of event log data

Baud rate	Latency
9600	0.710
4800	0.981
2400	1.615
1200	2.910

3.9 Comparison of Communication Techniques

In this section the polling and exception reporting techniques are compared in terms of system latency versus the data throughput. This enabled an efficient bandwidth control scheme to be designed in order to optimise the telemetry control system and find out the boundaries and optimal configuration of the Kingfisher S2 communication blocks.

3.9.1 Polling vs. exception reporting

Figure 3-19 is a comparison of polling and exception reporting techniques using the (rx_data) and (tx_data) blocks at 9600. Refer to Appendix I for 4800, 2400 and 1200 Baud rate results. The results show that the (tx_data) block has a lower latency for the transmission of up to 32 Registers due to less protocol overheads. This is typically used by the remote station to transmit data based on an event and is known as the exception reporting technique. (tx_data) block consists of command 17) Send Multi Data and 00) Simple Acknowledge frames. The (rx_data) block consists of command 16) Get Multi Data and 17) Send Multi Data commands.

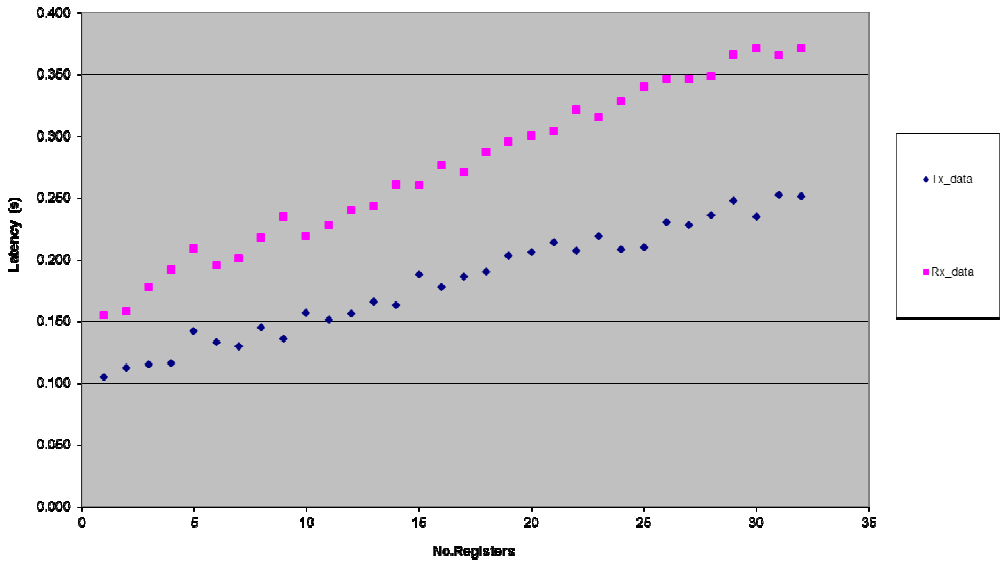


Figure 3-19 (Rx_data) vs. (Tx_data) transmission data latency@ 9600 Baud

(IEEE Power & Energy Society, 2010) DNP3 std1815 states that "a spontaneous message is an unsolicited message" which is equivalent to an exception report in the Kingfisher Series 2 Protocol whereby an outstation can report data changes as they occur. The (tx_data) communication block is used for exception reporting and has a maximum transmission capacity of 32 registers at a time. This technique requires a collision avoidance mechanism to coordinate the exception reporting technique between multiple remote stations attempting to transmit data over the communication media.

Investigation into "contention based mechanisms" for the exception report technique whereby multiple stations are able to transmit data to the master station requires a

coordination technique called Carrier Sense Multiple Access/ Collision Avoidance CSMA/CA is whereby the node senses that the medium is idle before transmitting data. The method of contention resolution occurs at the physical layer and is based on a random back off period with a maximum and minimum time. If the medium is busy the node will backoff for a random period before attempting to transmit again. This reduces the probability of collisions due to the use of a random backoff period. The Trio – E Series radios used for this system can be enabled for collision avoidance. The configuration called "RF Carrier Detect based Collision Avoidance" is used for half duplex systems and comprises of setting the random backoff time: [Backoff slot (1-16) x Time slot (1-255ms)]. When the medium is sensed idle the station will decrement its value until it reaches zero and transmits data. If the medium is still busy after this period it gets a new time slot allocation.

The advantage of this technique is reduced system latency for data transmission and remote stations may transmit when required. This technique however requires collision avoidance mechanism to coordinate messages which can be complex over a large scale network. Transmission is stochastic therefore it is difficult to analyse and develop and optimised system.

The Polling technique performs a poll request for data and is typically used by the master station to acquire data from remote stations. The remote station responds by transmitting its data back to the master station. Polling is a form of controlled access to the medium whereby exception reporting is random access. Polling is typically used in a network configuration with a designated master station which polls each remote station periodically. Each remote station has a unique address. The master station broadcasts its poll request of a specific address within the (Target field) in the header of the poll request refer to Appendix O Kingfisher S2 Protocol. The polling technique has the advantage of checking remote site link status whereby if only exception reporting was used the master station would not know if the link was off line or whether the RTU has no data to exception report.

The system was designed as a point to multi point network. This network configuration favours the polling technique whereby the master or base station requests for data over single channel frequency. This method allows the master station to have control over the entire network enabling a deterministic system approach whereby the master station can be programmed to poll in a certain order defined by a periodic polling table. This approach

allows polling according to priority which is a form of system optimisation. Therefore the system was designed to be a “pure polling system”.

3.9.2 Polling technique comparisons

Since the system was considered to be designed as a pure polling system a comparison was performed for the (rx_update) and (rx_data) blocks which was used to implement polling strategies for a deterministic system offering total network control, high data reliability as well as allowing system expansion. Table 3-12 describes the RTU configuration for polling a remote RTU. The blocks represent groups of 64 registers each as shown in Table 3-13.

The (rx_data, rx_update) communication blocks were compared in terms of data throughput vs. latency as shown in Figure 3-20. From the protocol format the (rx_data) has less protocol overheads compared to (rx_update) block.

Table 3-12 (rx_update) configuration for remote stations

Block	Local RTU configuration
1	16#0001
2	16#0003
4	16#000F
8	16#00FF
15	16#7FFF
16	16#FFFF/ ALL

Table 3-13 Local RTU addressing scheme

Update Register Blocks Constants			
Block	**Registers**	**Ch**	**Constant**
1	#R1 to #R64	1	0001 Hex
2	#R65 to #R128	2	0002 Hex
3	#R129 to #R192	3	0004 Hex
4	#R193 to #R256	4	0008 Hex
5	#R257 to #R320	5	0010 Hex
6	#R321 to #R384	6	0020 Hex
7	#R385 to #R448	7	0040 Hex
8	#R449 to #R512	8	0080 Hex
9	#R513 to #R576	9	0100 Hex
10	#R577 to #R640	10	0200 Hex
11	#R641 to #R704	11	0400 Hex
12	#R705 to #R768	12	0800 Hex
13	#R769 to #R832	13	1000 Hex
14	#R833 to #R896	14	2000 Hex
15	#R897 to #R960	15	4000 Hex
16	#R961 to #R1024	16	8000 Hex

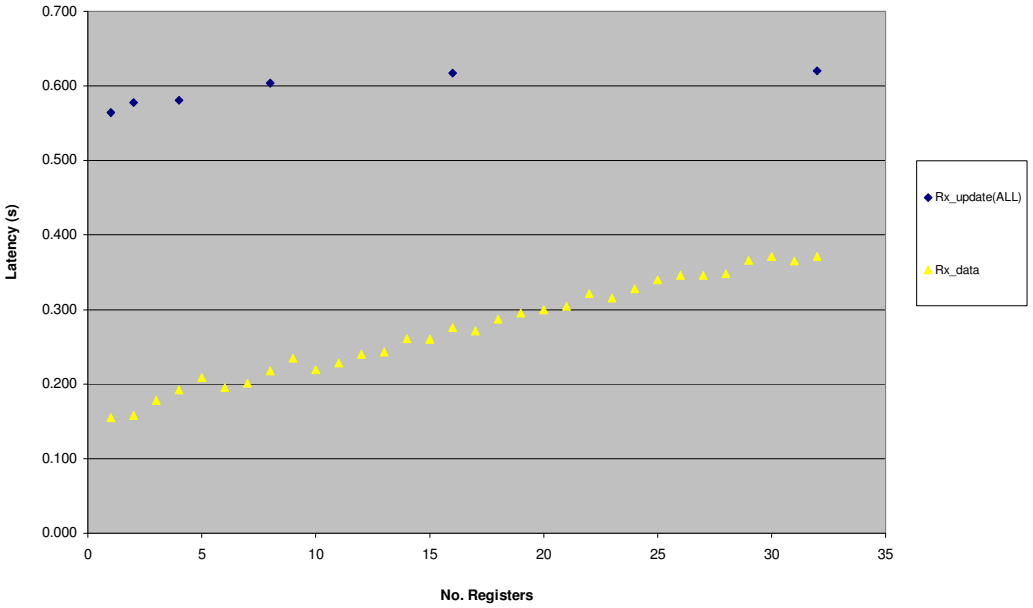


Figure 3-20 (rx_data) vs. (rx_update) at 9600 baud

The (rx_update) block consists of four commands with a maximum data transmission capacity of 1024 registers. It consists of a high number of protocol overheads but has the advantage of transmitting large amounts of data. Refer to Appendix C –Function codes for command structure. Command 26) Request RTU Update: The address of the RTU is stored in "Argument/Data Format". Command 15) Send RTU Update: Transmits the number of blocks to check for changed data. Command 12) Get Data Block: This command request data for up to 64 register values per block. Command 13) Send Data Block: Block reads identify the number of data values existing in the block requested, and are terminated at the last non-zero data value if the remaining data values are zero.

The (rx_data) block can transmit up to a maximum of 32 registers per execution. It has the advantage of being used to poll a small group of data with minimal protocol overheads and lower latency. The following two commands are used

16) Get Multi Data: Poll request up to a maximum of 32 registers.

17) Send Multi Data: Transmits requested number of registers from the target RTU to initiating RTU.

The Figure 3-20 proves that the (rx_data) block is more efficient than the (rx_update) block up to 32 registers and will be utilised in the system design to perform limited poll requests hence making the bandwidth control scheme more efficient.

3.9.3 Polling technique comparison up to 3 blocks

Figure 3-21 is a comparison of (rx_data) block and (rx_update) block transmitting up to 3 blocks of data at 9600 baud. The (rx_data) block second execution intersects with (rx_update) and incurs a slower latency after 32 registers by diverging from (rx_update) transmission latency. Therefore the results show that the (rx_update) communication block performs the best in terms of data efficiency if transmitting more than 32 registers.

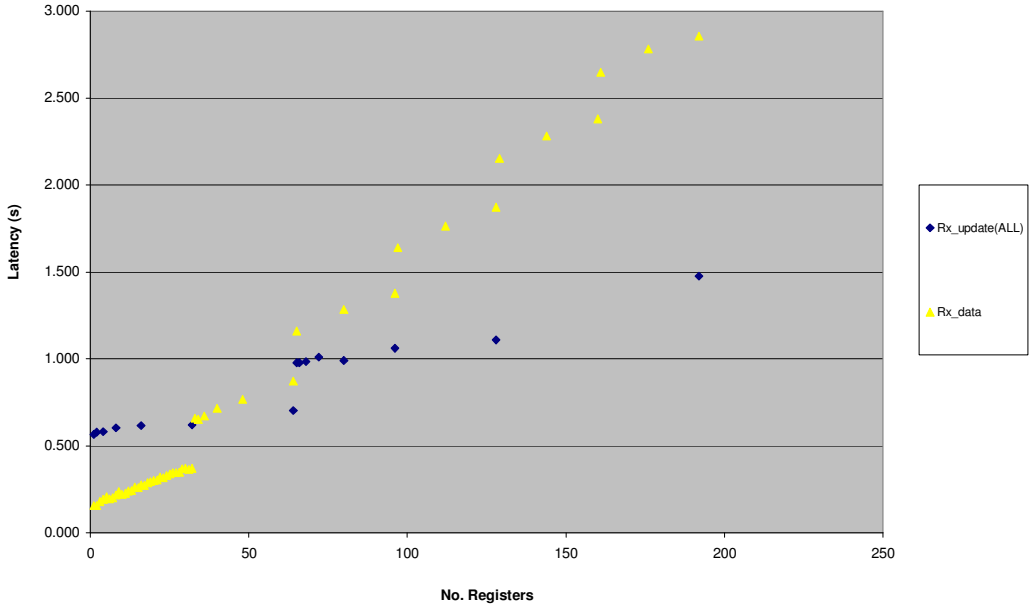


Figure 3-21 Polling technique comparison

3.9.4 Polling of event logs

An Event Log is a time stamp data based on an event. SCADA systems in particular the utility industries (Power, Oil and gas, Water/wastewater treatment) all rely on event reporting for monitoring and analysis of system events. Table 3-14 shows the event log frame format used for time stamping, prioritising and defining the type of log generated.

Table 3-14 Event Log Frame Format

Date/time	Msec	Value	Reference	RTU	Priority/user type
4 bytes	2 bytes	2 bytes	2 byte	1 byte	1 byte

Each event log is stored using 12 bytes. Event Logs are configured under Memory - Event Logs within the "Toolbox 32" RTU software. Event Logs are stored in a circular buffer whereby if the buffer is full the oldest data is written over. Three data types within those logs are 16 bit registers, 32 bit registers and bits are under the reference field.

Event logs are generated from the RTU ladder logic where it is possible to have multiple events or as System Logs. System logs are predefined logs and are generated for predefined events like Warm Start, Cold Start, and Set Real time Clock. Polling of Event Logs is configured within the (rx_update) communication block shown in Figure 3-22 below.

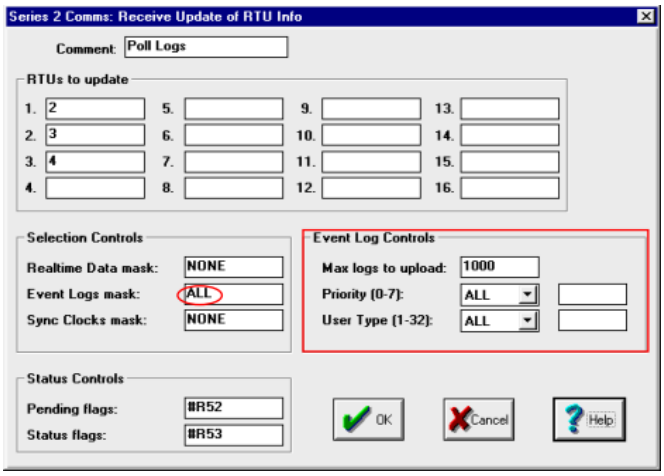


Figure 3-22 Event Log Controls

The rx_update (Event Logs) consists of 2 commands (Refer to Appendix C – Function codes)
#Command 80: Request Event Logs
The function code is set to (04 – get next) which means get from the remembered index. The RTU filter is set by the Event Log mask under Selection Controls Field. The number of logs is fixed at 10 logs for each request up to the maximum configured or until the buffer is empty. The index points to the last address that was polled. The optional Priority filter and Type filter are set under the Event Logs Controls and were both set to (ALL) therefore Priority:00, Type: 00 to poll all logs (Priority ignored)

#Command 81: Send Event Logs

The function code is set to (0E – send next) followed by the RTU filter, the number of logs transmitted is fixed at 10 logs per poll request up to the maximum configured or until the buffer is empty. The index is the address of the time stamp data as follows: date/time, msec (time stamp), value (16 bit data value), reference (system log/16bit, 32bit, bit registers), RTU number, priority and user type.

It must be noted that a maximum number of 10 event logs are transmitted per request up to the configured "Max logs to upload" value under the Event Log Controls which was limited to prevent fairness problems amongst remote stations. If the initiating RTU receives 10 event logs the next request is automatically generated. If less than 10 logs are received then the request is terminated.

3.10 Empirical Data Modelling Validation

The aim of the empirical data modelling was to validate that the derived formulas accurately predict system latency which can be emulated as bandwidth utilisation. This was used to design the service duration or time slot allocation per remote RTU and would also be used for future design and development and optimisation. The system modelling enabled the design of a deterministic system approach with better control of the telemetry network as well as allowing system analysis and optimisation to be performed. The latency of data transmission was calculated based on the baud rate for each communication block and the amount of data to be transmitted. From the system measurements the delay constant of the system response times were added to improve the accuracy of the calculations for each communication block. These calculations enable the service time duration to be designed for the network control.

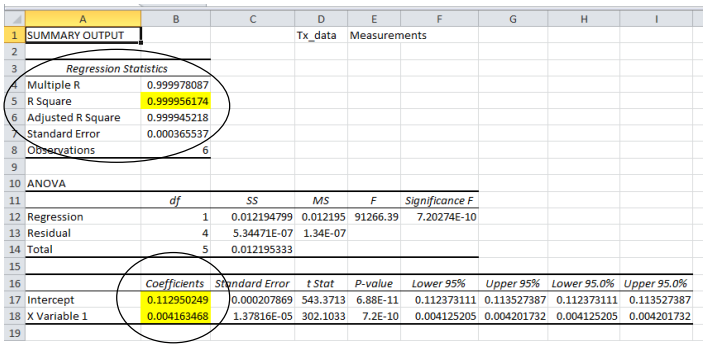
3.11 Regression Analysis

From the experimental results a number of points were measured to find the system latency depending on the communication block refer to Appendix S. To construct a function which closely fits the empirical data points a regression analysis or curve fitting was performed. The derived formula or function of the system latency can be emulated as bandwidth utilisation for bandwidth control and management purposes over the communications medium.

The empirical data proves that there is a linear relationship between number of registers transmitted and the system latency, therefore linear regression was used to find the best fit through the empirical data.

In Figure 3-23 the "Coefficients" indicate the estimated regression equation. "Intercept" refers to the y-axis intercept. "X variable1" refers to the gradient in this linear regression model.

R square value is defined as the coefficient of determination R^2. This value is a "Regression Statistic" measurement of how well the regression line approximates the real data points. This value states the correlation type is between 0 and 1. As shown in Figure 3-23 this value is 0.9999 thus it has a very strong linear correlation. The Standard Error indicates the typical error we are likely to make when we use the fitted line to estimate latency.



	A	B	C	D	E	F	G	H	I
1	SUMMARY OUTPUT			Tx_data	Measurements				
2									
3	*Regression Statistics*								
4	Multiple R	0.999978087							
5	R Square	0.999956174							
6	Adjusted R Square	0.999945218							
7	Standard Error	0.000365537							
8	Observations	6							
9									
10	ANOVA								
11		*df*	*SS*	*MS*	*F*	*Significance F*			
12	Regression	1	0.012194799	0.012195	91266.39	7.20274E-10			
13	Residual	4	5.34471E-07	1.34E-07					
14	Total	5	0.012195333						
15									
16		*Coefficients*	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
17	Intercept	0.112950249	0.000207869	543.3713	6.88E-11	0.112373111	0.113527387	0.112373111	0.113527387
18	X Variable 1	0.004163468	1.37816E-05	302.1033	7.2E-10	0.004125205	0.004201732	0.004125205	0.004201732
19									

Figure 3-23 Regression Analysis key parameters

3.12 Rx_update (ALL)

The outstation RTU was configured for all registers to be polled by the base station. Figure 3-24 shows the number of registers transmitted at 9600 Baud. Refer to Appendix J for 4800, 2400, 1200 Baud rates results. The results show similar relationship at all configured baud rates. There is a delay constant between the transmissions of each block of 64 registers. The mean calculated error was 1% difference between the modelled equation and the measured data at 9600 baud which shows that the model is accurate in predicting latency. Refer to Appendix S for regression analysis and Appendix T for empirical data.

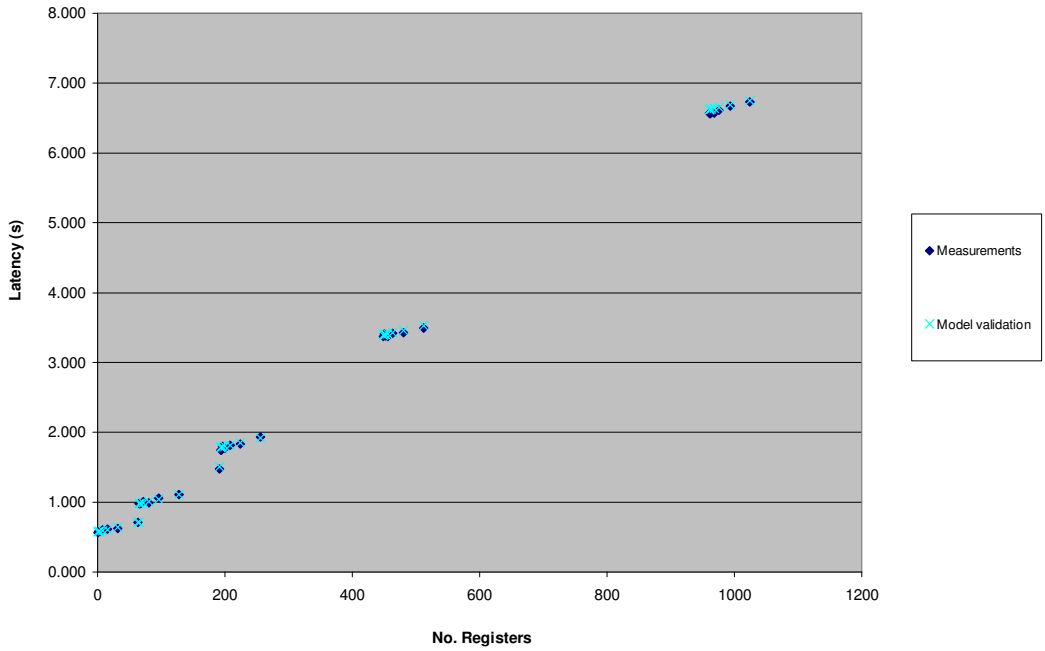


Figure 3-24 Model validation for rx_update (ALL) at 9600 baud

Latency = (0.002 x No Registers) + (0.572 + (0.270 x (No Blocks-1))) **Equation 3-1**
Note: single Block of (rx_update) = 64 registers

3.13 Rx_update (Requested Number of Blocks)

The outstation RTU was configured to poll only a requested amount of registers. Figure 3-25 shows the latency for the number of registers transmitted at 9600 Baud. Refer to Appendix K for 4800, 2400, 1200 Baud rates. The results show a similar relationship at all configured baud rates. There is incremental delay constant between the transmissions of each block containing 64 registers resulting in a delay period between transmissions. The mean calculated error was 2.3% difference between the modelled equation and the measured data at 9600 baud which shows that the model is accurate in predicting latency. Refer to Appendix S for regression analysis and Appendix T for empirical data.

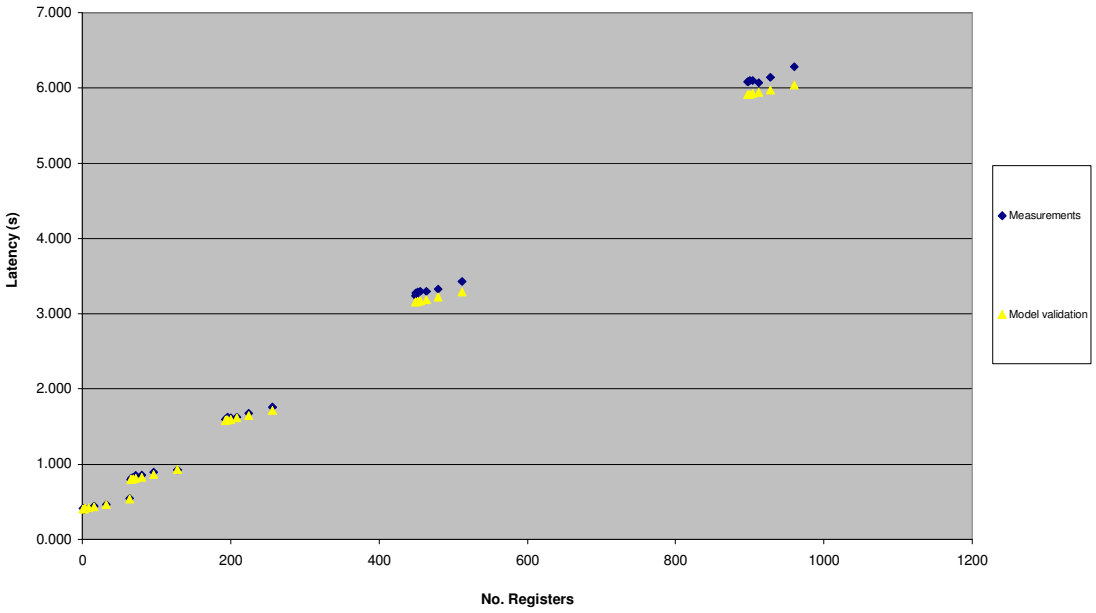


Figure 3-25 Rx_update (Req)

Latency = (0.002 x No registers) + (0.403 + (0.260 x (No Blocks-1)) **Equation 3-2**

Note: single Block of (rx_update) = 64 registers

3.14 Rx_data

The polling technique using (rx_data) block was used to poll a limited amount of registers from the system. The Figure 3-26 shows the data transmission latency and multiple executions of the (rx_data) block transmitting up to 194 registers at 9600 Baud rate. Refer to Appendix L for 4800, 2400, 1200 Baud rates. The results show a similar relationship at all configured baud rates. There is incremental delay constant between multiple transmissions of the (rx_data block). The mean calculated error was 3.5% difference between the modelled equation and the measured data at 9600 baud which shows that the model is accurate in predicting latency. Refer to Appendix S for regression analysis and Appendix T for empirical data.

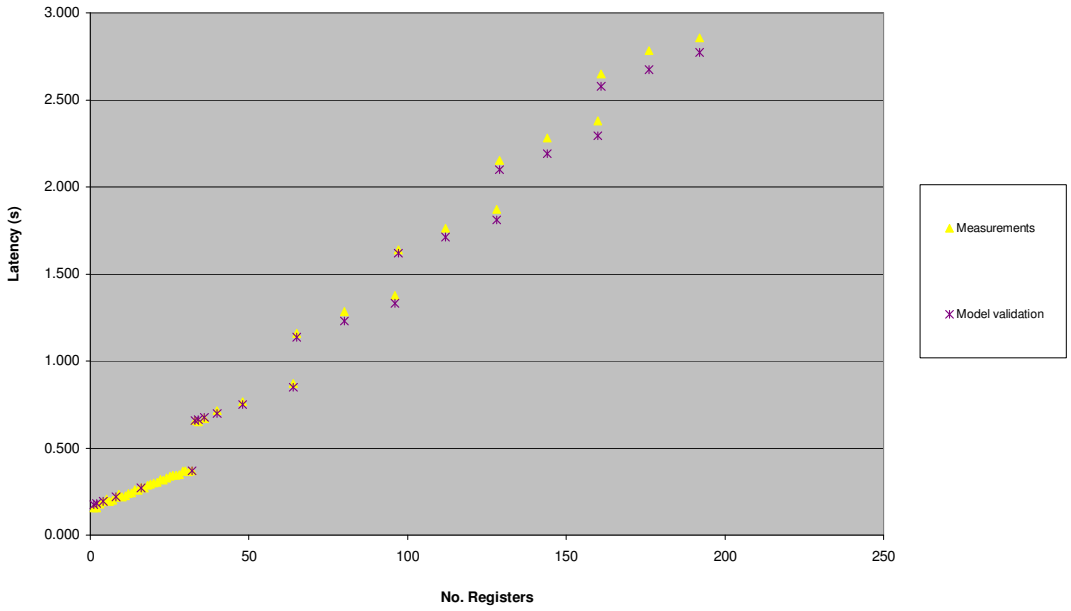


Figure 3-26 Model validation for Rx_data block at 9600 Baud

Latency = (0.006 x No registers) + (0.17 + (0.281 x (No Blocks-1))) **Equation 3-3**

Note: single Block (rx_data) = 32 registers

3.15 Tx_data

The polling technique using (tx_data) block was used to poll a limited amount of registers from the system. Figure 3-27 shows the (tx_data) block data transmission latency at 9600 Baud. Refer to Appendix M for 4800, 2400, 1200 Baud rates. The results show a similar relationship at all configured baud rates. There is delay constant between the transmissions of multiple (tx_data) blocks which were factored into the calculations. The mean calculated error was 3.2% difference between the modelled equation and the measured data at 9600 baud which shows that the model is accurate in predicting latency. Refer to Appendix S for regression analysis and Appendix T for empirical data.

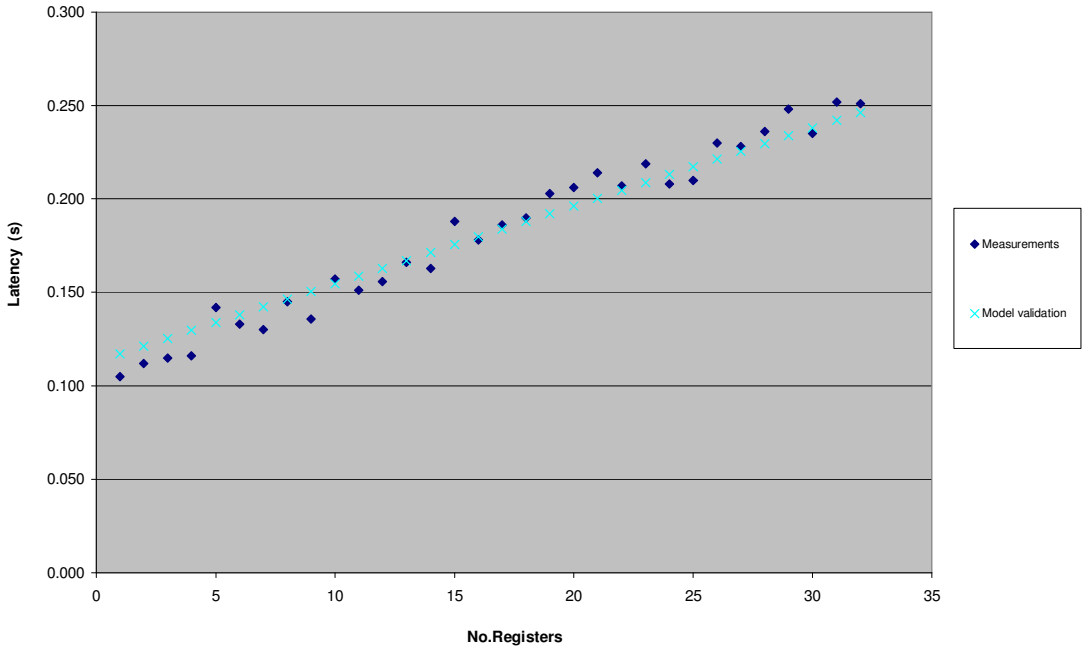


Figure 3-27 Model validation for Tx_data block at 9600 Baud

Latency = (0.0047 x No registers) + 0.1037 **Equation 3-4**

3.16 Rx_update (Event Logs)

The base station RTU was configured to poll up to a maximum of 20 event logs. The Figure 3-28 shows the polling the data latency transmission of 20 polled event logs at 9600 Baud. Refer to Appendix N for 4800, 2400 and 1200 Baud rate results. There is a delay constant after the poll response for 10 event logs since this block requests up to a maximum of 10 event logs per request up to the configure maximum number of logs or until the logs are empty. The mean calculated error was 1.8% difference between the modelled equation and the measured data at 9600 baud which shows that the model is accurate in predicting system latency**.** Refer to Appendix S for regression analysis and Appendix T for empirical data.

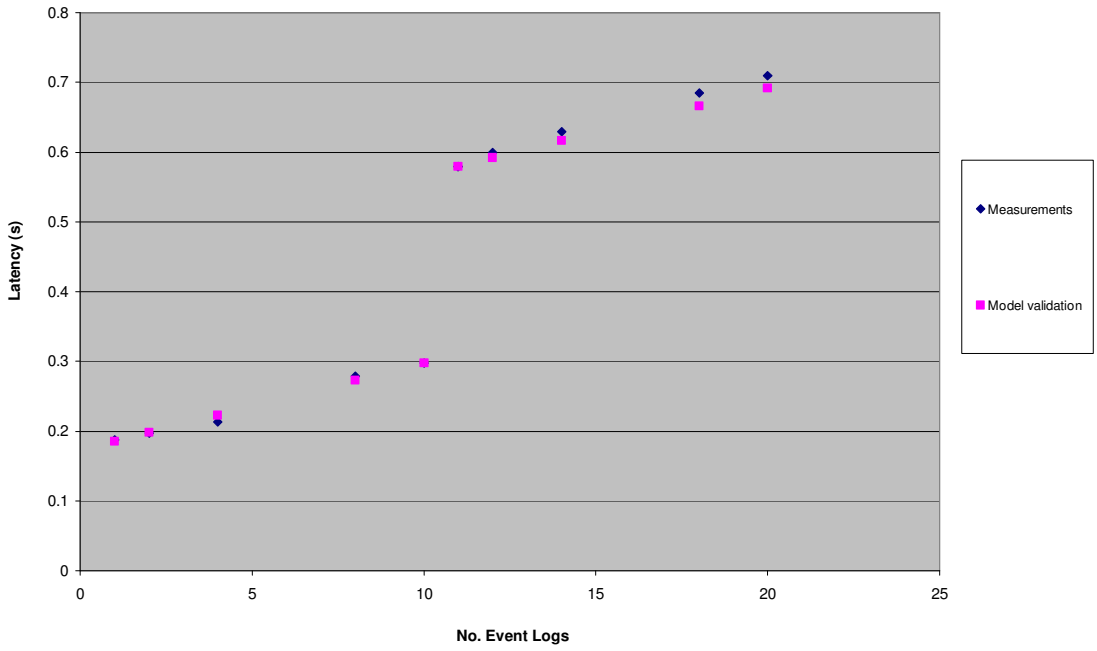


Figure 3-28 Model validation for Event log data at 9600 Baud

Latency = (0.0125 x No logs) + (0.1725 + (0.269 x (Log Block))) **Equation 3-5**

Note: single rx_update (Log Block) = 10 Event logs

Chapter 4

System Design and Development

4.1 Introduction

This section presents the SCADA and telemetry control system design. The SCADA system requires the selection and integration of a range of vendor products. The main criterion of component selection was interoperability for system flexibility. The SCADA system was designed for system expansion and data integrity.

4.2 Operational Requirements

SCADA systems require a communications system for data acquisition. A telemetry system is required for remote site process monitoring. The telemetry control system was designed to operate over radio frequency using a radio modems. These systems typically operate at 9600 baud rate therefore bandwidth is limited. The use of industrial protocols further reduces the bandwidth due to high overheads to enable data integrity. This limits the bandwidth availability for data throughput therefore the proposed system design aims at bandwidth optimisation to meet SCADA operational requirements for real time and event log data to enable process control and system monitoring.

The operational requirements require high data integrity and regular real time status updates of critical real time and alarm data which will improve the quality of service for remote station status updates enabling faster alarm response times which is critical for system monitoring and analysis. Optimisation of the system will enable system scalability, by allowing for more hardware to be added to the system over a single channel medium hence reducing the network system cost.

4.3 System Design

Figure 4-1 shows the implemented SCADA system design which uses a point to multipoint network topology. This topology was implemented since it acquired data to a central location which could then be networked over a high speed Ethernet backbone to the SCADA system. A single RTU is used as the master station and stores all remote site data in network registers configured within the master station local memory. The master station uses an omni -

directional antennae to broadcast messages to all remote field sites. The remote field sites use directional antennas aimed at the master station to transmit data when requested. The remote field sites or stations use Kingfisher RTU's for local process control and monitoring. The Kingfisher Series 2 protocol was implemented in for this telemetry system. The master station is in total control of the communication system. The system operates as a master – slave mode whereby the master station requests for data from a slave or remote site. This is known as the polling technique. This technique offers a deterministic communication control system design which is favoured to enable high data integrity. The SCADA is also responsible for transmitting set point commands and controls via the master station.

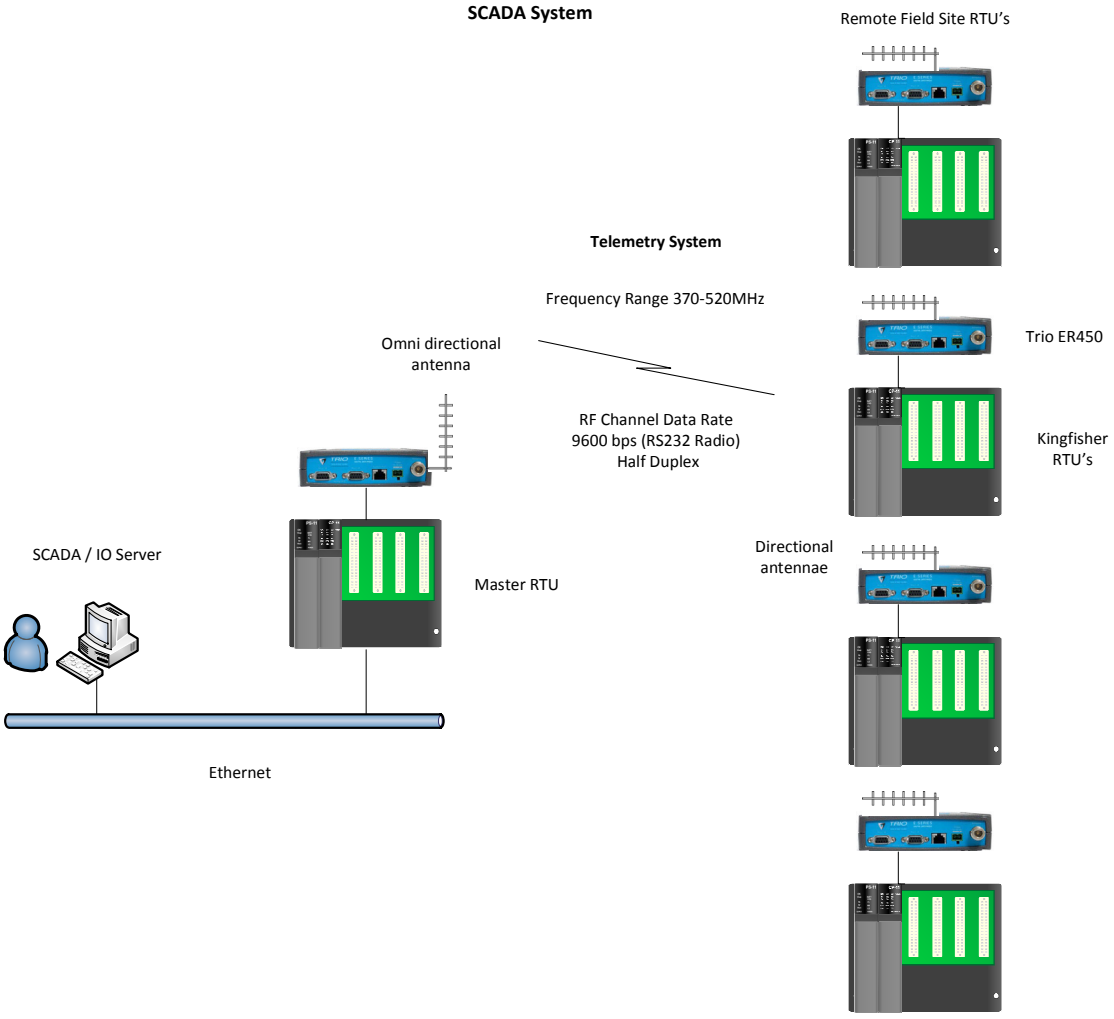


Figure 4-1 SCADA System Design

4.4 SCADA Development

4.4.1 Human to Machine Interface (HMI)

The human to machine interface was designed using “Wonderware Intouch” software. The software enables the programmer to design graphical visual representation for process control, monitoring and operational management.

The main benefits of the HMI are:

1. Real time and historical trending capabilities
2. Real time alarming capabilities
3. Device integration for wide range of systems

Figure 4-2 Designed HMI : Network Communication Statistics

The aim of the implemented HMI was to display the communication statistics to test the reliability of the telemetry system and display the link status to each remote RTU for monitoring. The system reliability is critical to ensure data acquisitions from the remote sites are transmitted successfully. The advantage of using the polling technique is that the link integrity in continually monitored.

Figure 4-2 shows the following:

Site Name: Remote site RTU address

Fail count: Number of failed transmission messages

Success count: Number of successful transmission messages and the percentage

Msg fails: Number of failed messages (increments when configured message retries are exceeded)

Link status: Digital alarm uses graphical representation of the link status between the master RTU and the remote RTU's (red – offline, green – online)

Yest.Fail: This value is the previous day message fail total (Msg fails) written over at midnight.

Figure 4-2 describes the remote sites implemented in this system to test the HMI functionality. The number of failed and successful communication messages are represented by Success and Fail counter columns. A single Message Fail column is registered after all configured "message retry" attempts do not receive a response to the initiated message within the timeout period. At the end of the day this value is written to the Yesterday fails column which represents the total number of the previous days message fails for that link.

The Intouch software is designed to operate as an (OPC client) "Object Linking and Embedding for Process Control" which performs read and write request via the IO server software (OPC Server) which regularly polls the master station who's responsibility is data acquisition from the remote sites.

4.5 Wonderware Intouch device integration

The Wonderware Intouch SCADA system enables easier device integration with the use of software products such as Factory Suite (FS) Gateway which is a communication gateway that provides a protocol translation between Suite Link/DDE, OPC and InTouch or Industrial Application Server Data Sources. The Object linking and Embedding for Process Control (OPC) is an industrial foundation for open connectivity for industrial automation devices implemented by Wonderware. FSGateway enables Wonderware products to act as OPC clients or servers enabling connectivity to any OPC software application.

In this design the Wonderware Intouch (OPC client) connects using FSGateway software which performs the protocol translation to the IO Server (OPC Server). I/O Servers are responsible for servicing the read, write requests from clients. The “IO Server software” is connected directly to the master RTU over Ethernet. An I/O Server keeps up-to-date information on all its connected I/O Devices by regularly retrieving data from the each device and storing it. Whenever a client requires data from an I/O Device, it will use the information stored in the I/O Server. Clients do not retrieve data directly from an I/O Device. In the implemented design the “IO Server software” was implemented since it supports Kingfisher S2 protocol and supports time stamped or event log data.

4.6 FSGateway Configuration

FS Gateway is application software which performs an interconnection and protocol translation between the IO Server and the Wonderware intouch software shown in Figure 4-3.

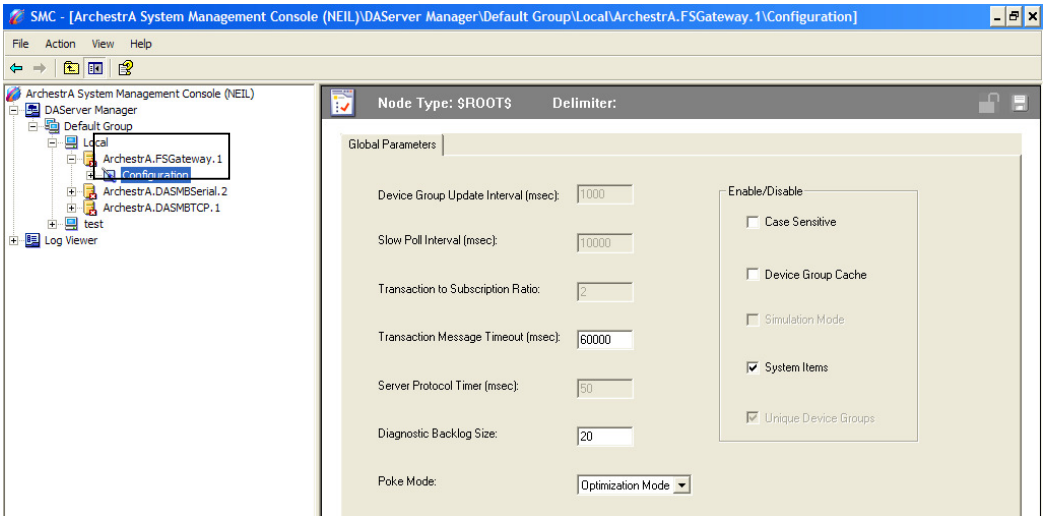


Figure 4-3

- Create server FSGateway via the System Management Console (SMC)

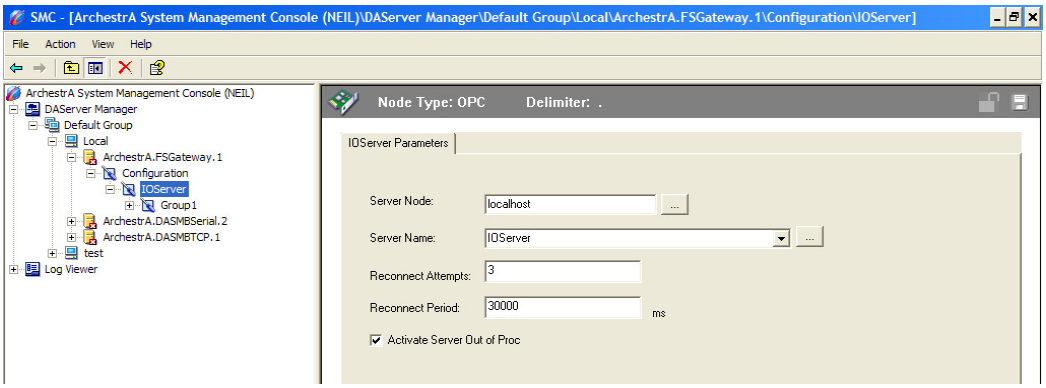


Figure 4-4

- Server Name: IOServer

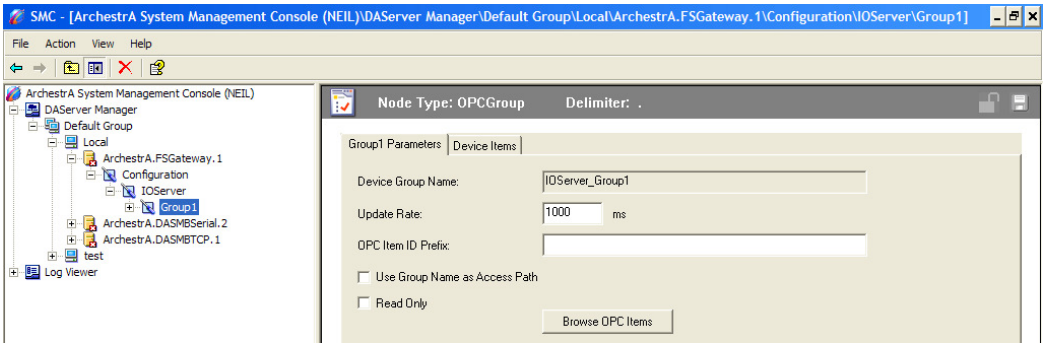


Figure 4-5

- Device Group Name: IOServer_Group1
- Update rate 1000ms

3.4.4 Intouch Object Linking

The Intouch software enables a connection to be created via the Access Names. A connection called “Device1” was used to link the intouch software to the FSGateway software as shown in Figure 4-6 and Figure 4-7. Figure 4-8 shows the Item name used to create a link to a variable tag.

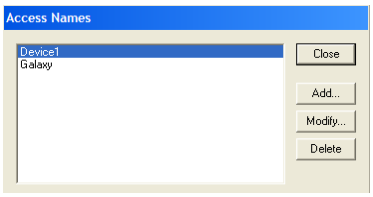


Figure 4-6

- In Intouch software we create an Access Name (name of connection) to the OPC Server : Device1

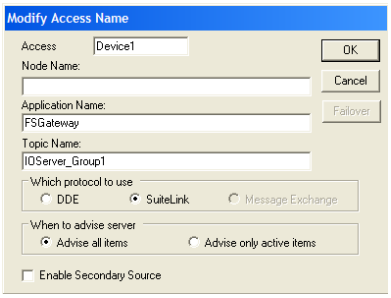


Figure 4-7

- Application Name: FSGateway
- TopicName is the OPC with name of group: IOServer_Group1 – linked to Device Group

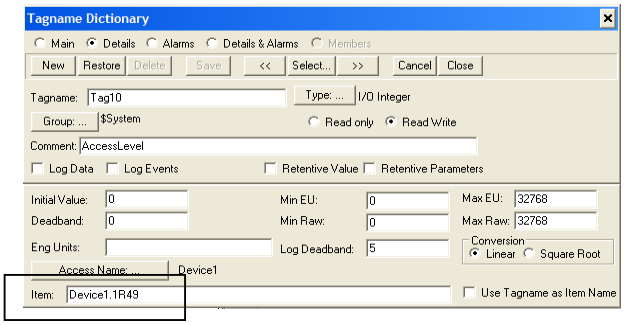


Figure 4-8

- Tagname dictionary: Tag 10: represents a process variable created within Intouch software.
- Item: Device1.1R49: Link Tag 10 to Device1, RTU1, Register 49.

4.7 IO Server Configuration

The IO Server software allows OPC server connectivity using Kingfisher Series 2 protocol to extract data from the master RTU and its main advantage is being able to extract event logged data. Only changes in values are transferred over the Ethernet LAN for the SCADA system making it more efficient.

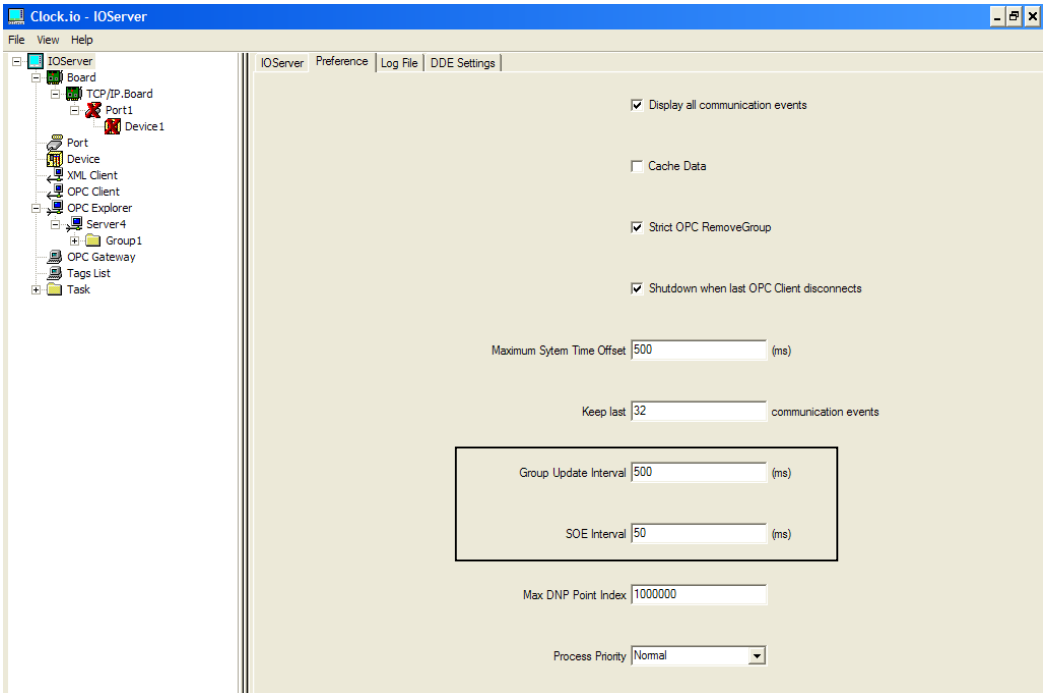


Figure 4-9

- Group Update Interval: Period to update status of the group of objects
- SOE Interval: Period to update sequence of events

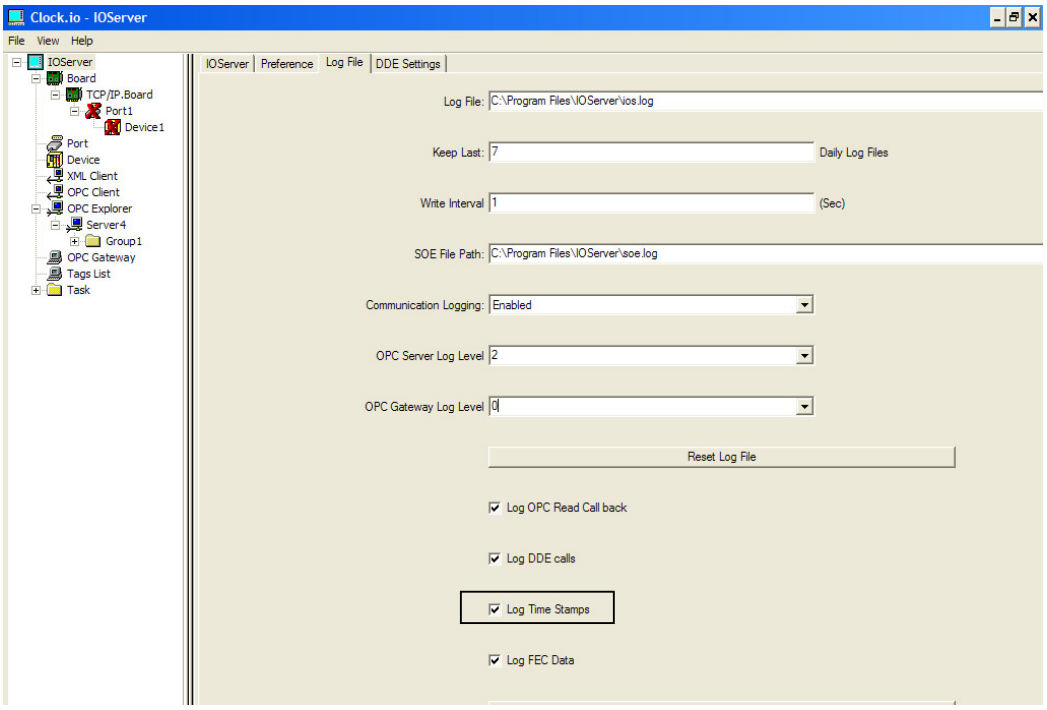


Figure 4-10

- Log Time Stamps: Enables event log data logging

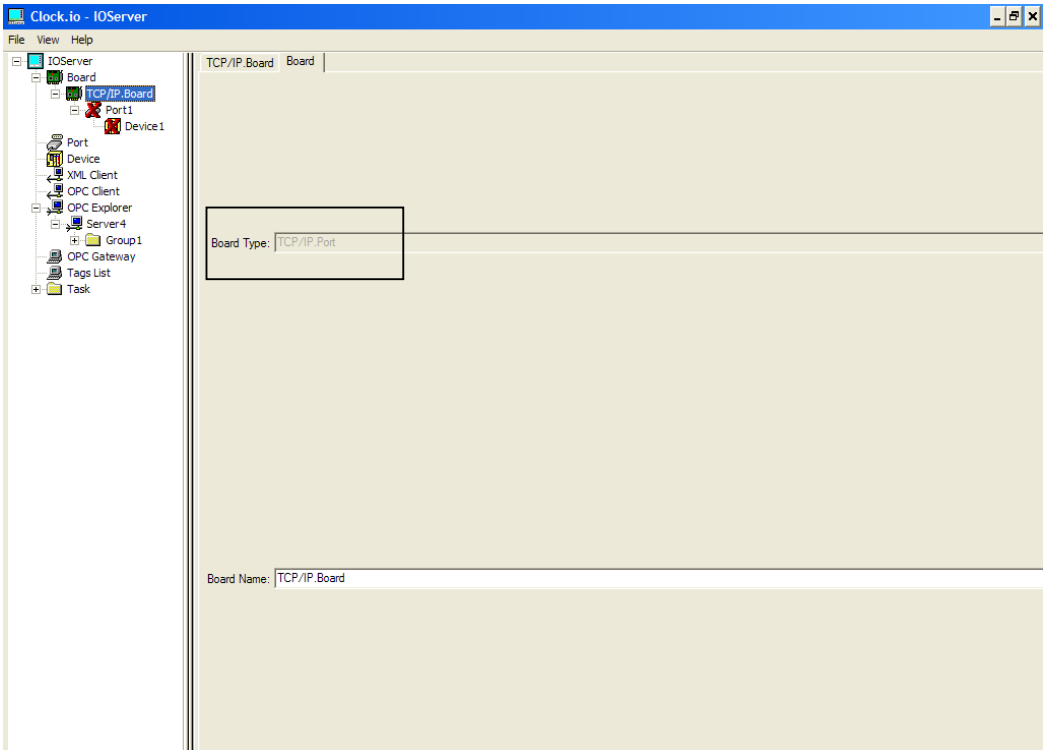


Figure 4-11

- Board Type: TCP/IP for Ethernet LAN connection

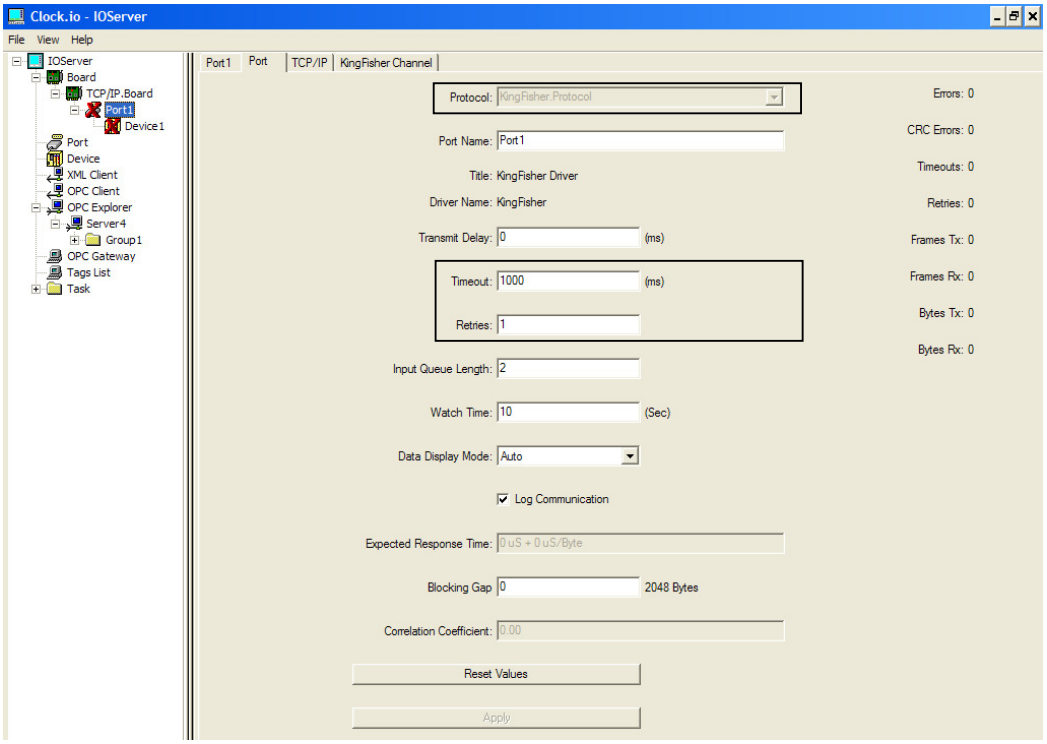


Figure 4-12

- Port Name: Port1
- Protocol: Kingfisher Protocol which supports time stamped data
- Timeout: Period to receive communication response to an initiated message
- Retries: Number of communication attempts after initial request

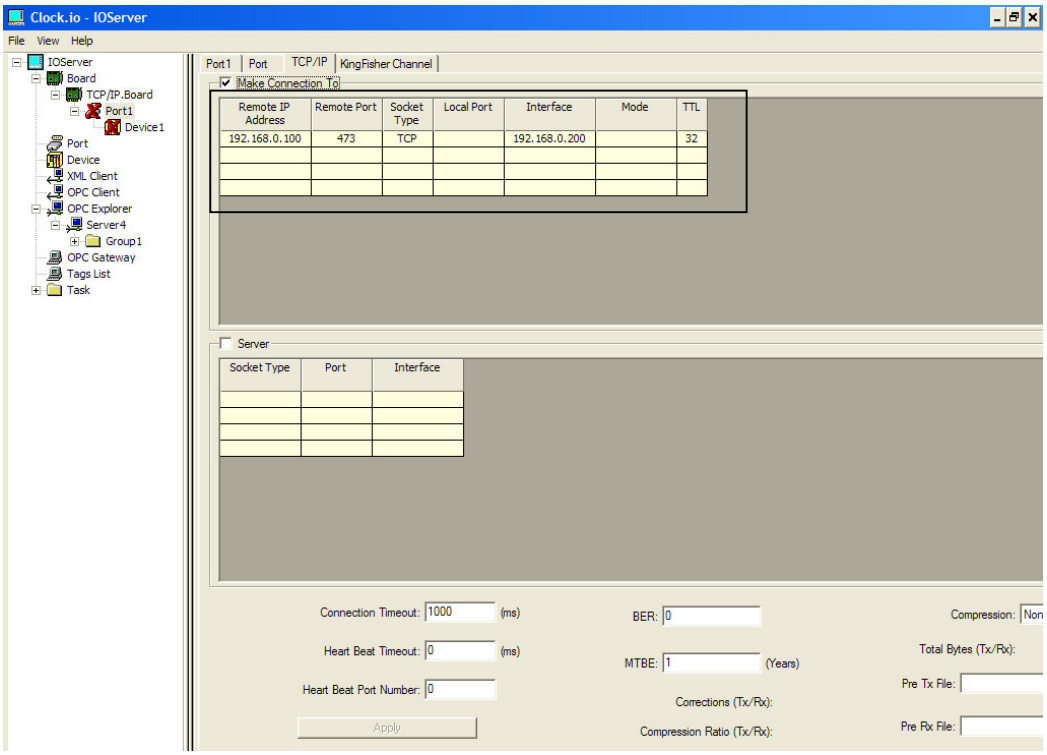


Figure 4-13

- Remote IP Address: IP address of remote server to make connection to: Master RTU
- Remote port: 473 Ethernet Port Address for Kingfisher S2 Protocol
- Interface: Network interface to use for this connection: Computer IP address

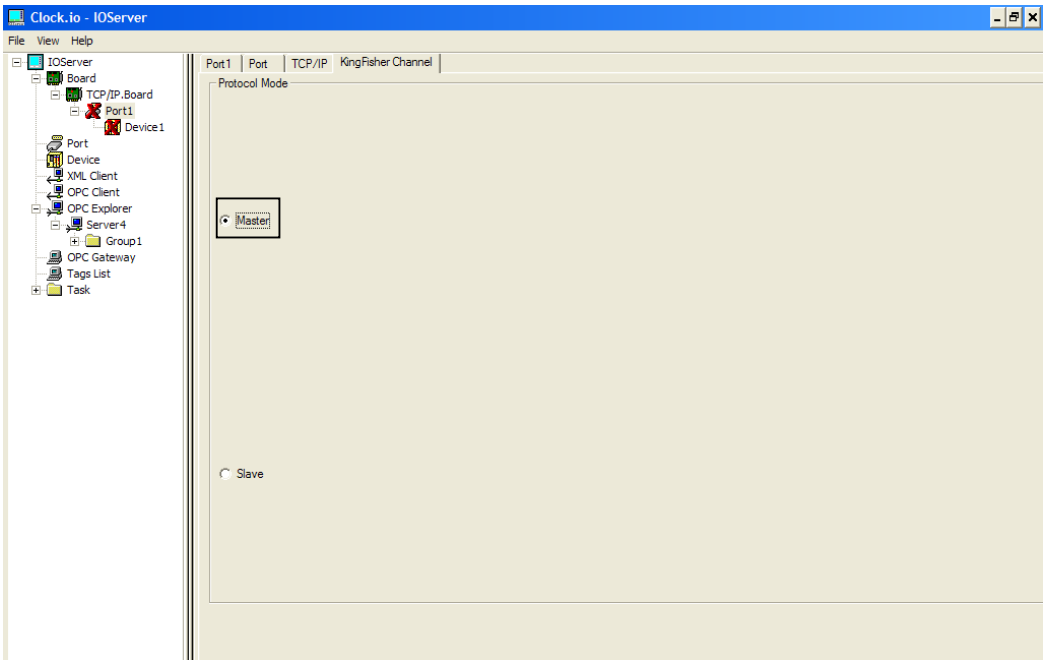


Figure 4-14

- Protocol Mode: Master: since IOServer will initiate all requests to the master RTU

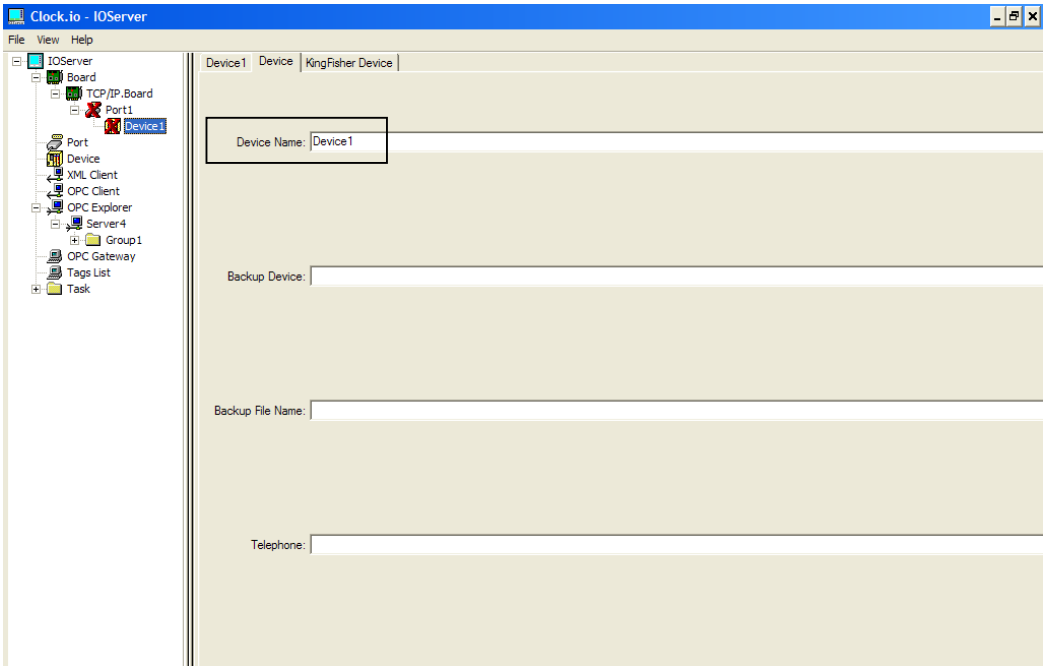


Figure 4-15

- Device Name: Device1 name association to link to the OPC client

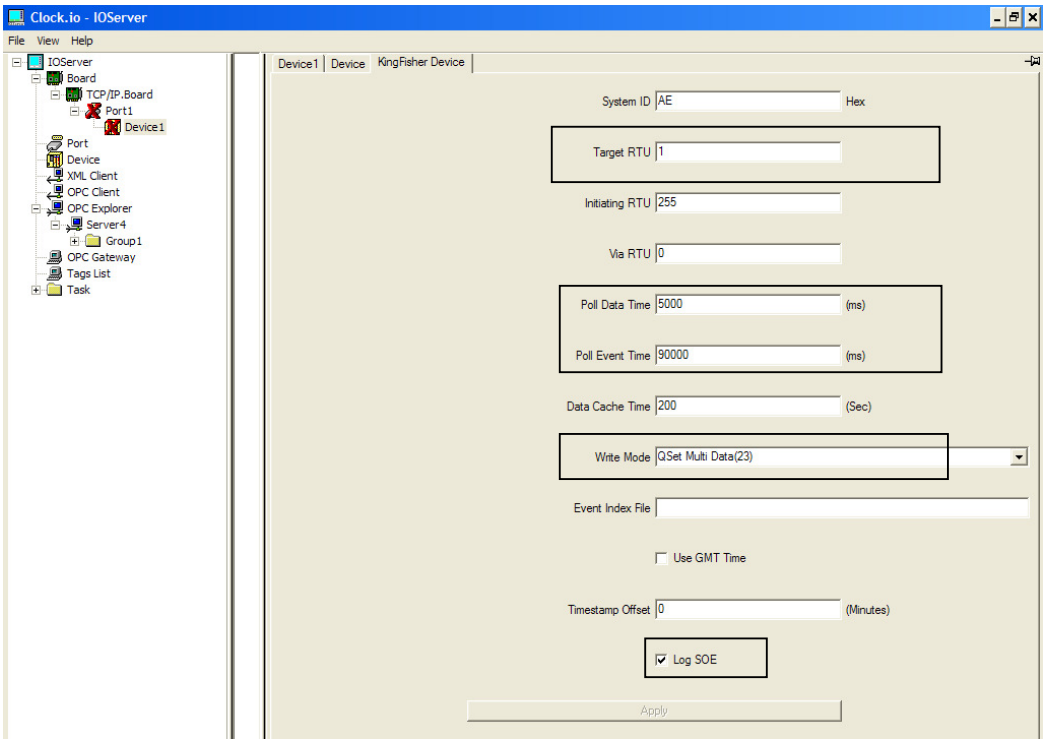


Figure 4-16

•Target RTU: The RTU address to initiate requests to: Master RTU1

•Poll Data Time: How often to poll the update counter of each remote RTU. The data from the remote RTU will be extracted if its update counter is changed or when the data is first requested. The poll time for local RTU is set by the group update interval of the OPC client.

•Poll Event time: How often to poll for time stamped data

•Write Mode: Write to remote RTU using the QSet Multi Data function code 23 which allows transparent write request to be made to the remote sites via the master RTU.

•Log SOE enabled to log Sequence of Events to ios.log File.

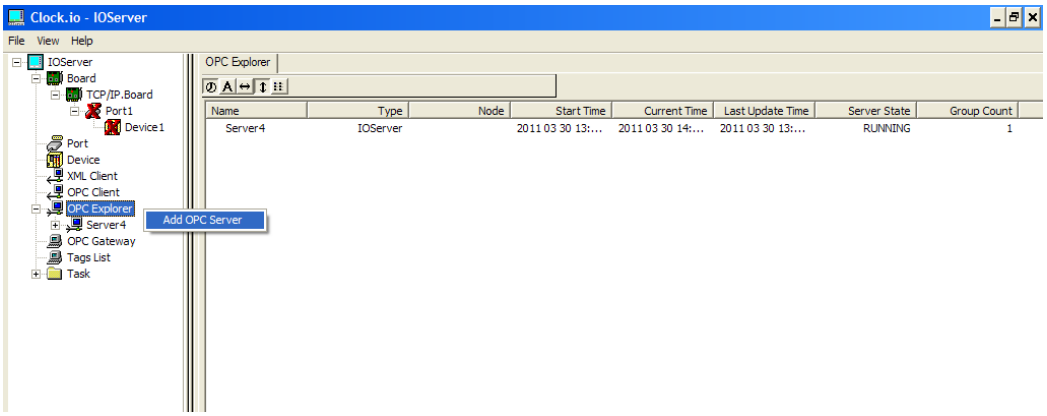


Figure 4-17

- OPC Server created for data acquisition from master RTU in the form of objects

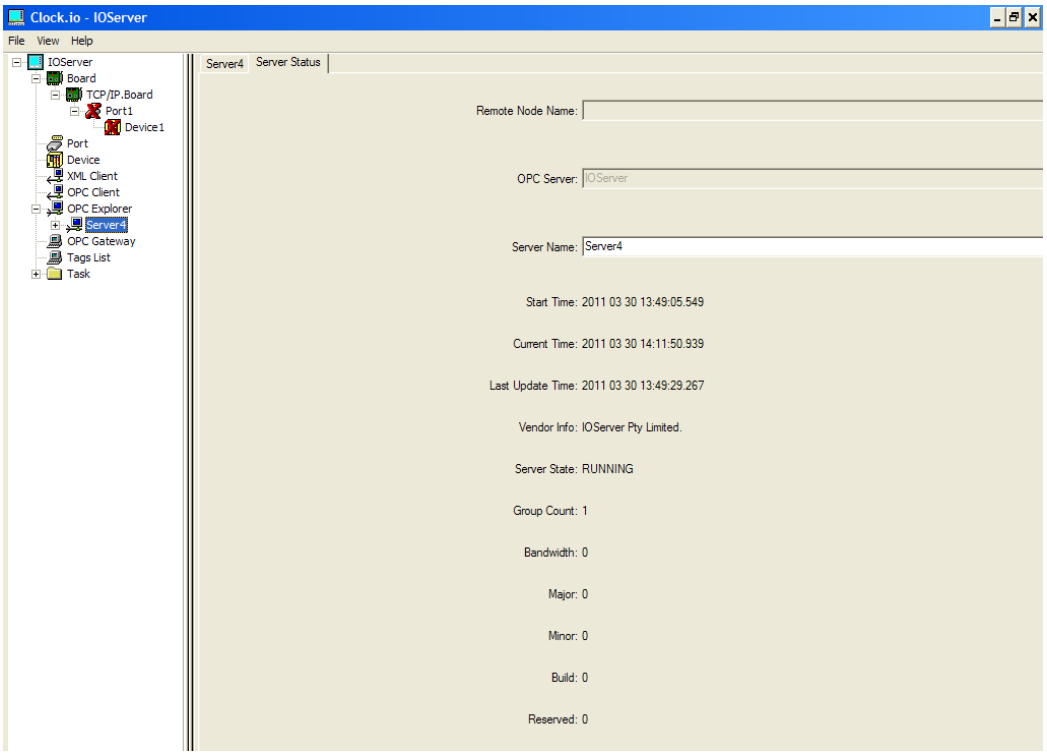


Figure 4-18

- OPC Server: IOServer
- Server Name: Server4

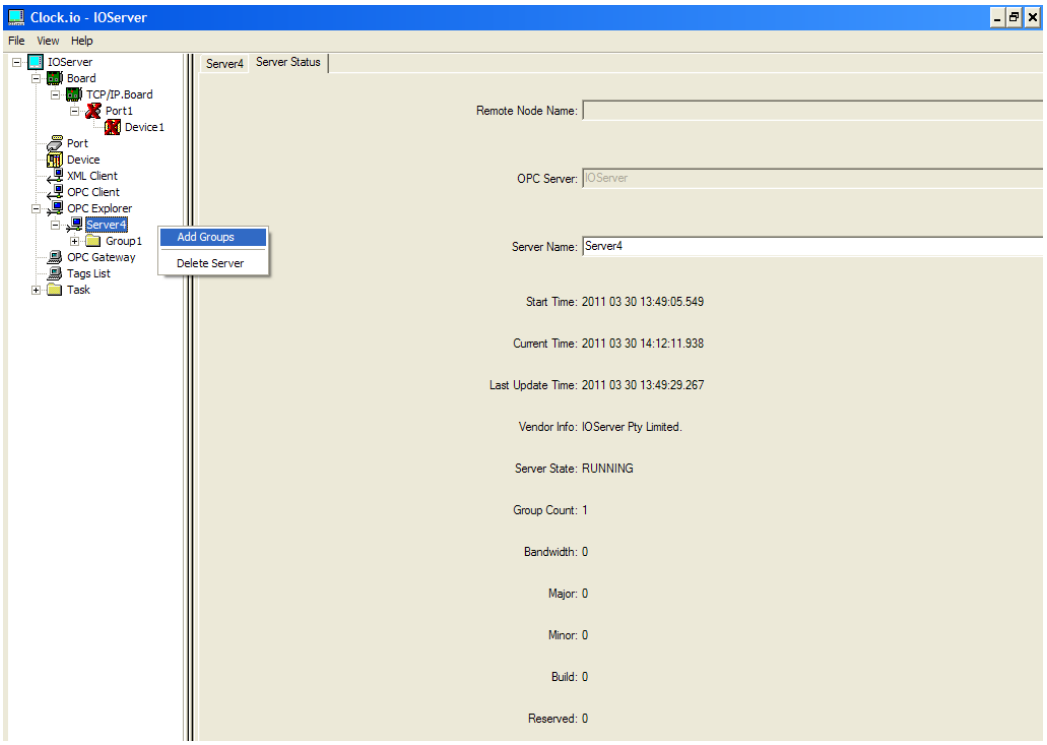


Figure 4-19

- Group created to store and update objects within the OPC Server

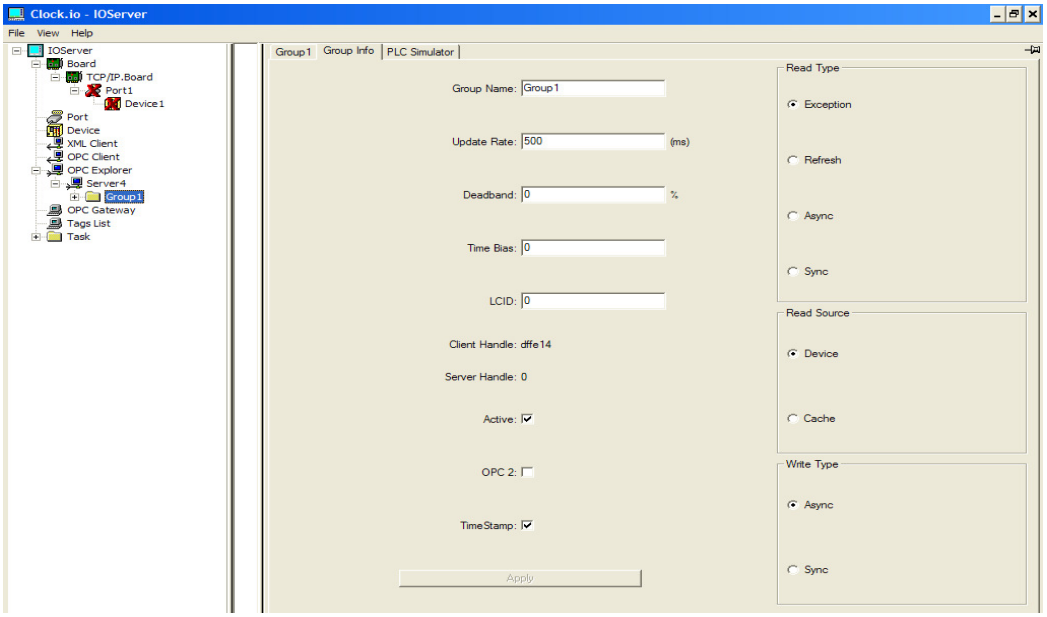


Figure 4-20

- Group name: Group1 : name of the group of objects

- Time stamp: enabled for event logging

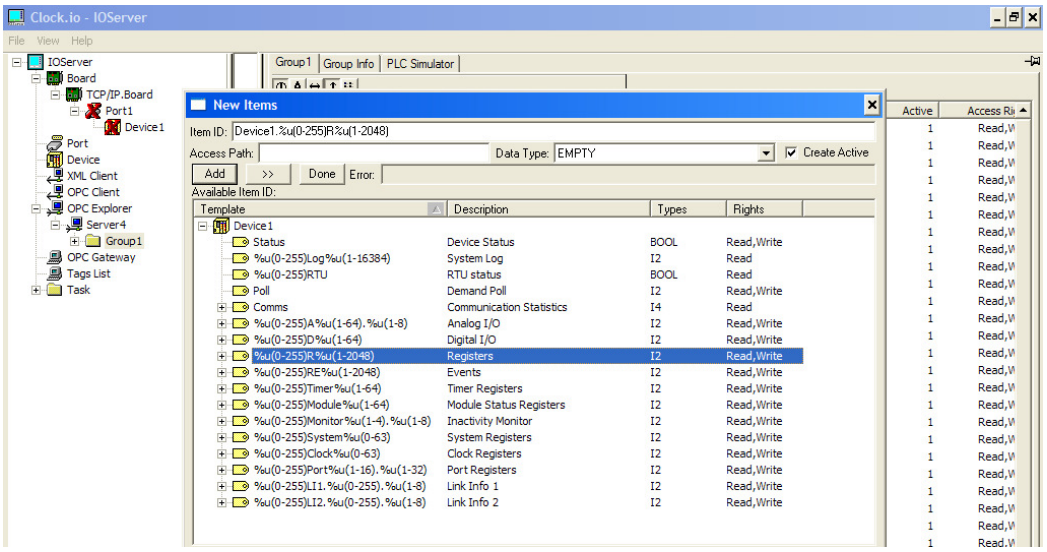


Figure 4-21

- Item ID: Device1.RTU.Registers – Name of the connection associated with Intouch(Access Name)

- Time tagged data from polled event logs are logged to the “ios.log file”

4.8 System Optimisation

The telemetry control system software was designed in the master station. The master station is responsible for controlling the communication system with the main objective of data acquisition. Ladder logic programming language was used with Kingfisher "Toolbox32" software to program the RTU's. The protocol implemented was "Kingfisher S2 Protocol". This system was designed using half duplex radio transmission at 9600 baud rate taking into consideration system scalability. The main objective for the software design was to optimise the bandwidth utilisation and efficiency via a bandwidth allocation scheme or algorithm and implement the designed "optimal static polling table".

The bandwidth allocation scheme or algorithm was designed to minimise server vacation periods and enable periodic real time system status updates of critical real time and alarm data from all remote sites. The scheme allocates service duration for non critical real time and event log data to be transmitted per remote station. The bandwidth allocation is biased towards event log data since SCADA systems rely on event monitoring.

The system was optimised by implementing an "optimal static polling table" to allocate visit frequencies amongst the remote sites thus efficiently allocating the bandwidth resource to high data traffic sites. The system uses the gated service discipline whereby only data present in the remote site at the polling instant are transmitted to the master station. The exhaustive service discipline presented fairness problems in the network design whereby a single RTU can monopolise the bandwidth usage therefore this service discipline was not implemented. Since SCADA systems have an "event" paradigm the event logs were given priority for bandwidth utilisation. However the critical real time and alarm data has the highest priority to gain access to the medium and breaks the polling cycle to gather remote site real time "status updates". Since the master station controls the entire network, this gives a more deterministic network behaviour which can be controlled, monitored and enables easier system optimisation and analysis as presented in Chapter 5 Analysis and System Optimisation.

4.9 Software Description

Figure 4-22 shows the bandwidth control algorithm for the telemetry control system. The system is designed to operate in three modes “Optimal Poll Mode” for steady state optimisation, “System Status Poll” for system critical real time status updates and “Event Poll Mode” which is executed during a significant system event for polling high priority event log data. An event is a significant change with respect to a change of state, analog threshold or dead band value which is exceeded. Refer to Appendix P for ladder logic code, Appendix Q for the variables list and Appendix R for test results.

“Optimal Poll Mode” implements the designed static polling table which defines the optimal polling frequencies per remote site, table size of the system as well as the service order for the master station to poll all remote stations within a polling cycle period. The system polls each station individually for non critical real time and event log data of (All Types) using the (rx_update) communication block. The system was designed to minimise switchover periods between stations. The polling service is designed to make efficient use of the shared bandwidth resource effectively by prioritising the service order and number of visits to each station based on the workload of the remote site. Thus stations with higher data traffic are given more priority and polled more frequently making efficient use of the bandwidth.

The “System Status Poll” is given the highest priority for medium access control over the other modes. It is designed to periodically break the optimal mode and perform a limited data poll request for each site in the system. This polling mode requests for critical real time and alarm data only, thus providing the SCADA system with regular real time status updates.

The (rx_data) communication block enables the entire system to be polled more efficiently by minimising protocol overheads and system latency. It prevents fairness problems by limiting the amount of data from each remote site to the first 32 registers for critical real time and alarm data. This polling technique improves the quality of service to the SCADA system by providing more frequent system real time status updates.

“Event Poll Mode” performs high priority (Type1) event data limited poll request for significant system event data which is triggered by a status flag from the remote stations during the “Poll Critical Data Mode”. This data is the highest priority event log data and is

grouped as Type1 in the remote stations. Hence the master station breaks from the “Optimal Poll Mode” to perform “Event Mode” poll request to a single or multiple stations up to a limited number of data before returning to the “Optimal Poll Mode”. This enables the system to be more flexible and respond to changes with regards to significant system events only. Refer to Appendix P for ladder logic code, Appendix Q for the variables list and Appendix R for test results

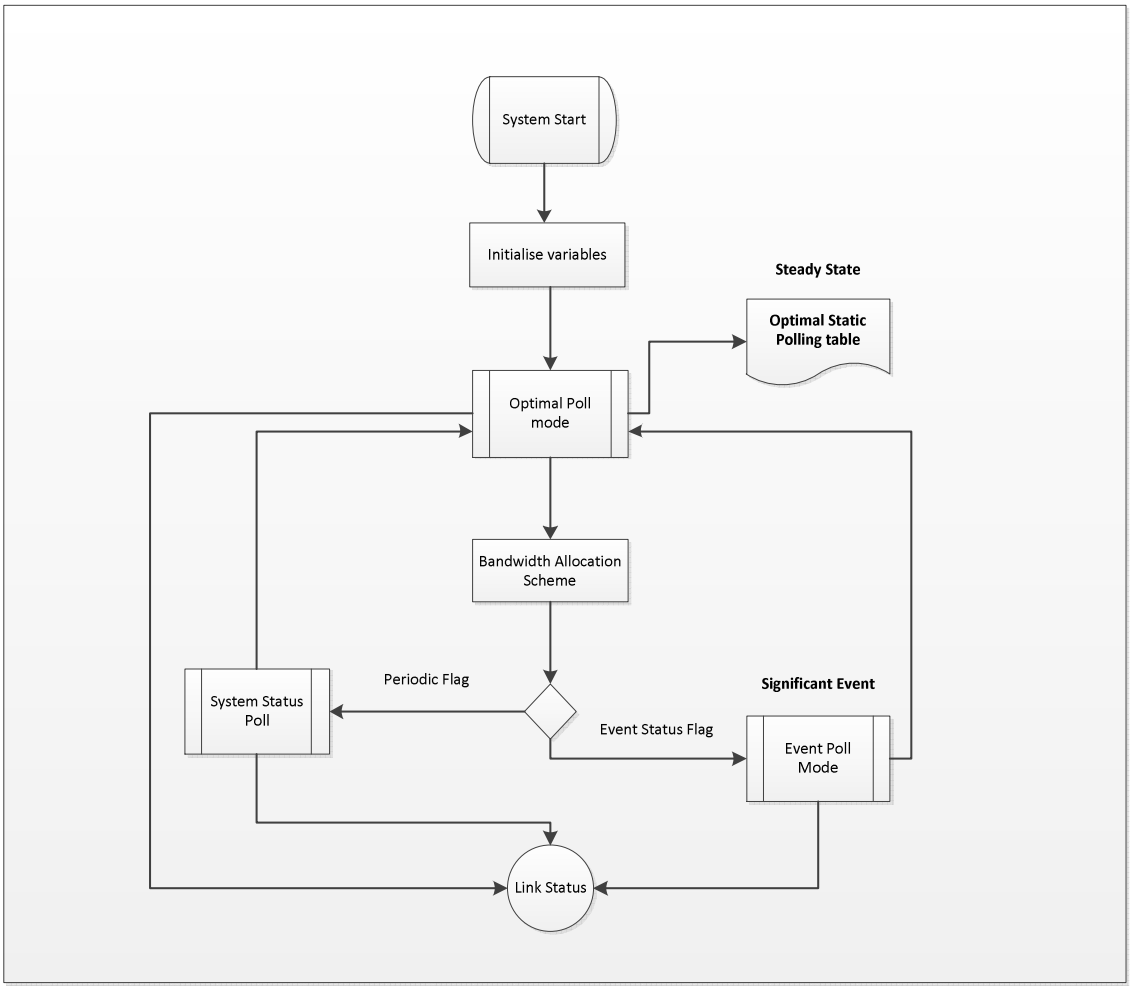


Figure 4-22 Bandwidth control algorithm for the telemetry control system

Chapter 5
Analysis and System Optimisation

5.1 Introduction

This section presents the system analysis and optimisation. From the literature review the work conducted by (O. J. Boxma et al., 1991) focussed on system optimisation by developing an "optimal polling table" which was implemented in this system. The study derives the efficient visit station frequencies which were implemented to optimise the shared resource amongst multiple queues depending on the workload. Queuing theory is the study of queues which involves mathematical analysis of several related processes. The performance measures are derived from parameters from the system which typically involve calculating the average waiting time in the queue.

5.2 System Analysis

Queuing models are represented by stochastic model. The analysis of these models involves determining the system parameters, such as the arrival rate, service time, and a diagram of the system and identifying the system states. Kendall's notation describes the nature of the arrival process to the queue, the nature of the service process (in terms of the service time), the number of servers, the maximum number of jobs that may be in the system and some basic queuing disciplines (Bose, 2001) and is written as follows : arrival process / service distribution / number of servers / no. of places in the system. These parameters are explained below:

- Arrival process typically uses the notation D (Degenerate distribution – Deterministic arrivals), M (Markovian – Poisson process), G (General), GI (General and independent), and Geom (Geometric)
- Service distribution typically uses the notation D (Degenerate distribution- Deterministic), M (Markovian – Exponential service time), G (General – Independent arrivals), GI (General and independent), and Geom (Geometric)
- Number of servers represents the total number of servers acquiring data

- Number of places in the system represents the capacity of the system and is omitted if the number is infinite.

There are a number of queuing models which are open to analysis. The simplest model is the D/D/1 type queue which has deterministic or fixed values for the inter arrival and service times. The M/M/1 queuing model represents a single server that has unlimited queue capacity, calling population and are characterised by the Poisson (random) arrival process and exponential service times. M/G/1 represents a single server that has unlimited queue capacity, infinite calling population with Poisson arrival process and the service time distribution may follow any general statistical distribution. A number of derived performance measures can be calculated based on the arrival rate and mean and variance of the service rate. Single server queues are very common queuing models and are commonly found in manufacturing, transport and communications therefore analysis of single server queues behaviour is useful in queuing models. Figure 5-1 typical queuing model can be applied to the telemetry communication system. The telemetry system design represents a single server system as the master station which periodically requests for data from the remote stations using the polling technique with a switchover period S_1 - S_N. The remote stations are represented by the queues (1-N) and the data or customers arriving at a queue represented by λ_1 - λ_N.

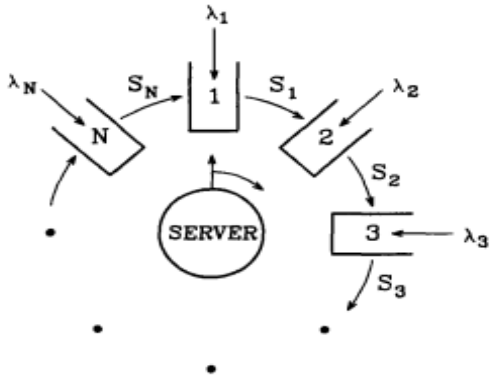


Figure 5-1 Typical queuing model

According to Kendall's notation this system can be classified as a D/D/1 system.
The network configuration parameters are described as λ_i, β_i, ρ_i, c_i, s_i.
The following parameters have been defined:

λ_i – Inter arrival rate of data

β_i – Mean service time

Workload $\rho_i = \lambda_i \beta_i$

These two parameters are the most important to the system performance since they are defined as the workload $\rho = \lambda_i \beta_i$

c_i – Cost or penalty of a customer waiting one unit of time. Can be used to add priority or weight to the system

s_i – Mean switchover time for the server to change from one station to another

- The inter arrival rate of data is generated periodically and consists of real time data and non –real time event log data. The arrival rate is a deterministic period of 100ms set by the IO Scan interval configuration of the processor. The CPU scan period is the time for the processor to scan the IO modules and the ladder logic.

- The service times have been designed as per bandwidth allocation scheme for the maximum service period duration per remote site. The system is asymmetrical since β_i is variable depending on the number of customers or data within the station. The service period is deterministic since the mean service time for each remote station was used for this system.

5.3 System Optimisation – Polling Table Design

(O. J. Boxma et al., 1993) states that "the effect of the table size does not have any significant effect on the results" (p.115) and also mentions that "the crucial steps are therefore (1) and (3)" (p.115). The aim was to apply the "Mean delay approximation" for determining the number of visits or polling frequency to be given to each queue during a cycle and apply the "Golden ratio" for the service order to evenly spread the visits in order to optimise the system performance.

To follow the design methodology proposed by (O. J. Boxma et al., 1991), the following steps were implemented in the system design.

Step1) Determination of visit numbers to a station using the mean delay approximation

$m_i = \text{sqrt}\ (c_i \lambda_i (1+ \rho_i) / s_i)$ **Equation 5-1**

This can be defined as the number of visit frequencies to a station within a cycle period for optimal system performance.

Step2) Determination of table size

Calculation $M = \sum_{i=1}^{N} m_i$ **Equation 5-2**

This is defined as the sum of the visits to all queues within a polling cycle.

Step3) Determination of visit orders

The Golden Ratio was applied to the system for the visiting frequencies in Step1 to be evenly spaced for the polling table implementation.

5.4 Switch over period

The switchover period shown in Table 5-1 is the time or system delay for the master station to change over from sampling a queue or remote station to the time it begins its next poll request to another remote station.

Table 5-1 Switchover period measurements

Sample	Measurements (ms)
1	83.2
2	89.6
3	96.4
4	88.4
5	103.3
6	89
7	107.4
8	94.1
9	87.4
10	100.1
11	93.8
12	92.9
13	90.2
14	98.4
15	96.2
16	93.8
17	95.3
18	99.2
19	88.2
20	97.4
21	82.3
22	78
23	80.1
24	92.4

25	81.4
26	86.7
27	82.5
28	80.9
29	78.2
30	78.1
31	72.4
32	81.4
33	79.1
34	83.6
35	89.8
36	87.9
37	80.5
38	83
39	79.8
40	79.4
41	87.2
42	86.9
43	82.9
44	85.4
45	78.9
46	89.4
47	79.4
48	79.6
49	89.6
50	83.4
mean	**87.3**
std dev	**7.6**

The table of results shows the switchover period between RTU's during polling which has a mean 87.3ms with a standard deviation of 7.6 ms.

5.5 Matlab System Calculations

To follow the design methodology proposed by (O. J. Boxma et al., 1991), the following 3 step design methodology was implemented for system optimisation.

```
% Step 1 Visit frequencies mi
c = 1 % cost
s = 0.087 % mean switchover duration
beta = unidrnd(46,1,4) % mean service time:uniform distribution discrete random between1 and 46 seconds
lambda = 0.1 %100ms constant: IO Scan interval configuration
rho=lambda.*beta % traffic load
```

```
f = sqrt (c*lambda.*(1+rho)/s) % Visit frequency calculations
mi=round(f) % results from rounding fi. No. visits given to Qi at least 1

%Step 2 Polling table size M
M = sum(mi) % The size of the table. Total number of visits in a cycle(period)
frequency = mi./sum(M) % visit frequencies as fractions
sum(frequency) % sum = 1

% Step 3 Visit order
phi = 0.618034 % golden ratio
circum = [1*phi 2*phi 3*phi 4*phi 5*phi 6*phi 7*phi 8*phi] % 1*phi – M*phi
circumference = mod(circum,1) % M circle of unit circumference
table = sort(circumference) %jth smallest number correspond to jth position in the table
```

5.5.1 Matlab Results

Table 5-2 Matlab Results

Workspace

Stack: Base

Name	Value
M	8
ans	1
beta	[45,23,37,7]
c	1
circum	[0.6180,1.2361,1.8541,2.4721,3.0902,3.7082,4.3262,4.9443]
circumference	[0.6180,0.2361,0.8541,0.4721,0.0902,0.7082,0.3262,0.9443]
f	[2.5143,1.9476,2.3243,1.3979]
frequency	[0.3750,0.2500,0.2500,0.1250]
lambda	0.1000
mi	[3,2,2,1]
phi	0.6180
rho	[4.5000,2.3000,3.7000,0.7000]
s	0.0870
table	[0.0902,0.2361,0.3262,0.4721, 0.6180,0.7082,0.8541,0.9443]

From the results the polling table can be derived as follows

Step 1) mi = [3, 2, 2, 1]; is the queue visit frequency per remote site (RTU 2-5).

Step2) M = 8; the size of the polling table for a single cycle.

Step3) Visit order

"Circumference" variable applies the Golden Ratio (0.6183) by multiplying each element up to the table size (M) as follows ϕ^{-1}mod1, $2\phi^{-1}$mod1…..M ϕ^{-1}mod1. This represents placing the values in a circle of unit circumference.

0.6180 0.2361 0.8541 0.4721 0.0902 0.7082
0.3262 0.9443

The queues are allocated according to Step1) visiting frequencies per queue

RTU 2, RTU 3, RTU 4, RTU 5

0.618034 0.236068 0.854102 0.472136 0.090169 0.708204
0.326238 0.944272

The visit order is determined by the table calculation whereby the j^{th} smallest number corresponds to the j^{th} position on the table

0.090169 0.236068 0.326238 0.472136 0.618034
0.708204 0.854102 0.944272

Thus the table visit order for the polling table is RTU: (3,2,4,3,2,4,2,5) for one polling cycle period.

Table 5-3 System polling cycle

RTU	Master position
3	1
2	2
4	3
3	4
2	5
4	6
2	7
5	8

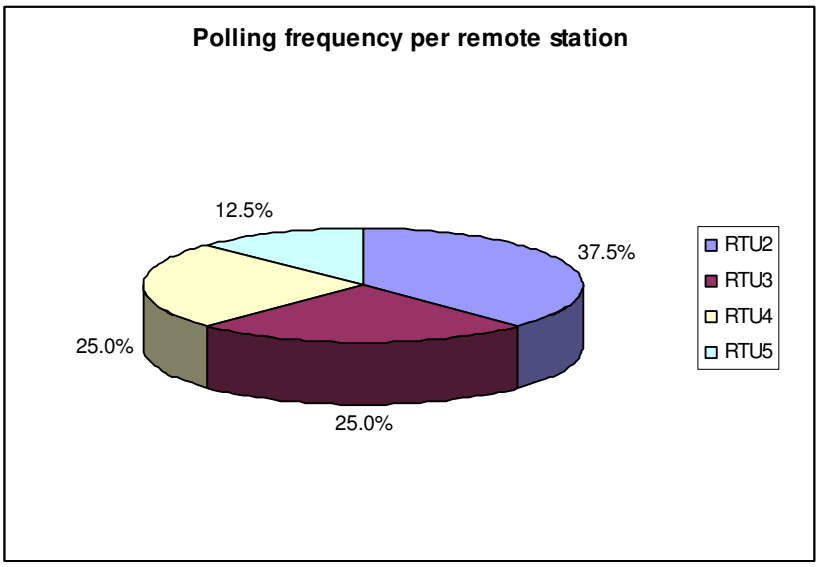


Figure 5-2 Polling frequency per remote station

5.6 Test Results

The RTU's (2 – 5) were configured as follows:

- beta = [45,23,37,7] service period (s)
- lambda = 100ms IO Scan interval configuration

Table 5-4 Polling test results

POLLING CYCLE	TIME	SERVICE ORDER	RTU5	RTU4	RTU3	RTU2
5	17:46:04.41	5	0:03:47			
5	17:45:33.38	2				0:01:20
5	17:44:57.16	4		0:01:42		
5	17:44:13.63	2				0:01:26
5	17:43:51.10	3			0:01:26	
5	17:43:14.97	4		0:02:04		
5	17:42:47.54	2				0:01:01
5	17:42:25.14	3			0:02:21	
4	17:42:17.62	5	00:03:47			
4	17:41:46.71	2				0:01:20
4	17:41:10.59	4		0:01:43		
4	17:40:26.34	2				0:01:26
4	17:40:03.85	3			0:01:26	
4	17:39:27.61	4		0:02:05		
4	17:39:00.11	2				0:01:01
4	17:38:37.60	3			0:02:11	
3	17:38:30.19	5	00:03:20			
3	17:37:58.78	2				0:01:10

3	17:37:22.76	4		0:01:15		
3	17:36:48.95	2				0:01:09
3	17:36:26.34	3			0:01:09	
3	17:36:07.44	4		0:02:05		
3	17:35:40.03	2				0:01:01
3	17:35:17.62	3			0:02:21	
2	17:35:10.11	5	00:03:47			
2	17:34:39.01	2				0:01:20
2	17:34:02.89	4		0:01:43		
2	17:33:18.77	2				0:01:26
2	17:32:56.37	3			0:01:26	
2	17:32:20.26	4		0:02:04		
2	17:31:52.85	2				0:01:01
2	17:31:30.44	3			0:02:18	
1	17:31:22.92	5				
1	17:30:52.20	2				0:01:17
1	17:30:16.07	4		0:01:21		
1	17:29:35.54	2				0:01:24
1	17:29:12.93	3			0:01:24	
1	17:28:55.42	4				
1	17:28:11.10	2				
1	17:27:48.70	3				
		Mean waiting time	**0:03:40**	**0:01:47**	**0:01:47**	**0:01:14**

The results prove that by applying the calculated polling frequencies and service order based on the system parameters the bandwidth utilisation is optimised. The remote station with the highest service duration RTU2 consumes the largest portion of bandwidth due to being polled more frequently by the master station thus resulting in the least mean waiting time. Similarly RTU3 and RTU4 have the same polling frequency and utilise the second and third largest proportion of bandwidth. RTU5 has the smallest service period thus according to calculations is the only polled once per cycle period and therefore consumes the least bandwidth. This method favours polling stations with the highest workload to efficiently allocate the bandwidth. This introduces fairness problems however the cost parameter could be used to increase a sites priority thus increasing its polling frequency. From this we can conclude that the scheme implemented in this system optimises the bandwidth utilisation of the single radio channel amongst all the remote stations. This methodology enables system scalability thus reducing network cost of implementing additional bandwidth resource.

Chapter 6

Discussion, Conclusions and Future work

6.1 Introduction

This section discusses the system design and implementation as well the results from system optimisation strategy. It also describes the future design and development and concludes this thesis.

6.2 Discussion

The design of this system involves system integration of multiple hardware components to develop the SCADA system. The half duplex radio frequency was used to develop the telemetry communication system for data acquisition. This opened the topic of how to efficiently allocate the limited bandwidth of this medium to multiple remote stations. The polling technique was used for data acquisition from remote sites to a central station. The design enables system scalability and increased hardware thus reducing the network cost for additional bandwidth. The system is also more efficient in terms of real time data system updates.

Optimisation of the telemetry control system requires the efficient utilisation of the limited bandwidth. Since protocol overheads and error checking methods further reduce the bandwidth an efficient polling strategy was used to optimise the system bandwidth utilisation. The implemented design improves the system in the following ways:

- The design is scalable and can therefore minimise the network design cost preventing the need for additional bandwidth
- Improved bandwidth utilisation by controlling the polling frequency
- Efficient real time system status updates for real time operational control
- Efficient alarm reporting to the SCADA system
- Data integrity of the network met through a pure polling system design

- Improved system flexibility to respond to real time and event log data
- Minimisation of server vacation periods
- Radio link status monitoring
- Categorisation of event log data for the system to respond to significant events

This implemented method for system optimisation favours polling stations with high data traffic to efficiently allocate the bandwidth. This introduces fairness problems to stations with low data traffic however the cost parameter could be used to increase a sites priority thus increasing its polling frequency.

6.3 Conclusion

This project involved the design of a SCADA system using Wonderware Intouch software for the human to machine interfacing with the main aim to develop an optimised telemetry control system for efficient data acquisition to the master or base station. The implemented design improves the quality of service for the SCADA system by providing regular real time system status poll requests. A pure polling telemetry communication system was implemented in this design using point to multipoint network topology over half duplex radio channel. The system design implements a deterministic service duration design which enables easier system analysis and optimisation to be performed. From the literature review the implemented design methodology using the “mean delay approximation” method was used to calculate efficient visit frequencies and enabled the optimisation of the bandwidth to the remote stations based on the workload of each remote station. The software for the telemetry control system was designed, tested and the results prove that the bandwidth utilisation can be efficiently controlled thus optimising the telemetry system.

6.4 Future Work

The telemetry control system uses Kingfisher "Toolbox 32" software to implement the ladder logic programming language which gives the design flexibility for future development.

Possible areas for future development include:

- Implementing a polled report by exception method since the current method must poll each device and look for changes in data from the remote stations.
- Improving the telemetry system software to be more adaptive to system changes and workloads.
- Further investigation into event log data categories and calculating efficient poll frequencies per category.

Appendix A - Kingfisher Series 2 Remote Terminal Units (RTU's)

KINGFISHER

Kingfisher PLUS+ RTUs are easy to install and configure, can be large or small, use interchangeable modules, have advanced communications and support powerful programming languages.

Kingfisher PLUS+ RTU features

- Four, 6 or 12 slot backplanes that can be linked in almost any combination to support up to 64 modules (per RTU). Twelve slot backplanes can be mounted in 19" racks.
- 60 W power supply (100 to 240 VAC or 20 to 60 VDC input)
- Four types of processor modules
- Up to 16 communication ports. Port options: Serial (RS232, RS422 and RS485), PSTN, GPS receiver, Private Line, Fibre Optic, Ethernet, Image Capture, Spread Spectrum Radio and Hart.
- Up to 1000 I/O points per RTU (I/O modules required)
- Eight channel analog input modules
- Four channel analog output modules
- Eight and sixteen channel digital modules
- High speed scanning of I/O signals
- Electrical isolation of I/O and communication circuits
- Ladder logic (PC-1 and CP-10/11/21 processors) or ISaGRAF (CP-30 processor) programming for control applications
- Powerful analog processing capability including PIDs
- Supports many protocols - Kingfisher, Modbus, DNP3, Allen Bradley, Omron and many more. Please see **Protocols.PDF** available from www.cse-semaphore.com/mykingfisher for a comprehensive list.
- Event logging of data (time and date stamped)
- Redundant processors and power supplies capability
- Self configuring of I/O modules at startup
- PC based configuration and diagnostic software – **Toolbox 32** or **Toolbox PLUS+**
- Local and remote configuration
- Battery backup
- Low power operation - suitable for solar installation

Kingfisher Processor Modules

Processor Module	Speed (MHz)	Processor	Memory			Comments
			FLASH	SRAM	DRAM	
CP-11	32	Intel 80C386	1024 kB	1024 kB	0 kB	Has over 15 years of software development. Very reliable.
CP-21	32	Intel 80C386	1024 kB	2048 kB	0 kB	Similar to CP-11 but has double the SRAM
CP-30	166	Cirrus ARM9 EP9301	16 MB	128 kB	32 MB	Fastest processor with greatest memory capacity
PC-1	16	80C188	128 kB	256 kB	0 kB	For small RTUs

Appendix B - Trio E-Series Radio

TRIO DATACOM

E-Series

Generation II
Remote Data Radio - ER450

The TRIO Datacom ER450 is an advanced high speed digital data radio modem designed for the most complex and demanding requirements in both Point-to-Point and Point-to-Multipoint (Multiple Address Radio) SCADA and Telemetry systems.

Normally used as a half duplex high performance Remote Data Radio for communicating with the full duplex EB450 Base/Repeater or EH450 Redundant 1+1 Base/Repeater stations, the versatile ER450 also serves as a low cost full duplex base/repeater or point-to-point-link station in both TRIO Datacom's E Series and M Series data radio systems with the addition of the ERFD450 option, and can provide network management access for remote diagnostics in either of these systems.

Features

Radio and Modem

- True 19,200 bps over-air data rates in 12.5kHz FCC channels (also 9600 bps) - Fully integrated radio, modem and data multiplexer
- 128-bit AES encryption (export restrictions may apply)
- 12.5 or 25kHz channel operation
- Fast data turnaround
- Simplex, Half Duplex and Full Duplex (Full Duplex with ERFD450 option)
- Compatible with legacy systems (Non Packet Digital and Bell 202 Modes)
- Full specification operation from -30 to + 60C
- Hazardous Environment Certification - Class I, Division II (Groups A,B,C and D)
- Compact, rugged diecast alloy housing
- Low power consumption with sleep mode operation
- Field upgradable firmware
- Multi-function LED Display
- Digital orderwire option
- DIN Rail mounting kit option
- VSWR protection

Data Ports

- Dual independent configurable data ports and separate system port
- Compatible with most industry standard data protocols. eg: MODBUS, DNP-3, IEC 870, SEL Mirrored Bits, etc.
- Selectable 300 -57.6 kbps asynchronous RS-232 interface
- Multistream™ simultaneous data streams allow for multiple vendor devices/protocols to be transported on the one radio network
- Flexible data stream routing providing optimum radio channel efficiency
- Internal repeater operation - single radio store and forward
- Channelshare™ unique integrated C/DSMA collision avoidance technology permits simultaneous polling and spontaneous alarm reporting operation in the same system

Network Management and Remote Diagnostics

(In conjunction with TVIEWs™ Software)

- Remote fully transparent Network Management and Diagnostics
- Network wide operation from any radio modem
- Full SCADA style features such as database, trending and networking
- Over-the-air modem reconfiguration
- Full graphical presentation (HMI)

Note: Not all product features are available in every mode of operation.

ER450

Radio

Frequency Range: 370-520 MHz (various sub-frequency bands available)
Frequency Splits: Various Tx/Rx frequency splits - programmable
Channel Selection: Dual synthesizer, 6.25 kHz channel step
Channel Spacing: 12.5 or 25 kHz
Frequency Accuracy: ±1ppm (-30 to +60C)(-22 to 140F) ambient
Aging: <= 1ppm/annum
Operational Modes: Simplex, Half duplexor Full duplex*
Configuration: All configuration via Windows based software
Compliances:

ETSI	EN300113, EN301489, EN60950
FCC	PART 15, PART 90
IC	RS119, ICES-001
ACA	AS4295-1995 (Data)
CSA	Class I, Division II, Groups (A,B,C,D) for Hazardous Locations ANSI/UL equivalent)

Local regulatory conditions may determine the performance and suitability of individual versions in different countries. It is the responsibility of the buyer to confirm these regulatory conditions. Performance data indicates typical values related to the described unit.

Information subject to change without notice.
© Copyright 2008 TRIO Datacom Pty Ltd.
All rights reserved. Issue 10-2008

* Export restrictions may apply

Note: Not all product features are available in every mode of operation.

Transmitter

Tx Power: 0.05 - 5W (+37 dBm) 1 dB User configurable with over-temperature and reverse power protection
Modulation: User configurable narrow band digitally filtered binary GMSK or 4 level FSK
Tx Keyup Time: <1mS
Timeout Timer: Programmable 0-255seconds
Tx Spurious: <= -37 dBm
PTT Control: Auto (Data) / RTS line (Port A or B) / System Port Override

Receiver

Sensitivity: -118 dBm for 12 dB SINAD
Selectivity: Better than 60 dB
Intermodulation: Better than 70 dB
Spurious Response: Better than 70 dB
AFC Tracking: Digital receiver frequency tracking
Mute: Programmable digital mute

Diagnostics

Network wide operation from any remote terminal.

Non intrusive protocol - runs simultaneously with the application.

Over-the-air re-configuration of user parameters.

Storage of data error and channel occupancy statistics.

In-built Error Rate testing capabilities.

Connections

User Data Ports: 2 x DB9 female ports wired as DCE (modem)
System Port: RJ45 for diagnostic, configuration and re-programming
Antenna: N female bulkhead. Separate N (Tx) and SMA (Rx) connectors for full duplex.*
Power: 2 pin locking, mating connector supplied
LED Display: Multimode Indicators for Pwr, Tx, Rx, Sync, TxD and RxD data LEDs (for both port A and B)

Modem

Data Serial Port A: RS232, DCE, 600-57,600 bps asynchronous
Data Serial Port B: RS232, DCE, 300-38,400 bps asynchronous
System Port: RS232, 19,200 bps asynchronous
Flow Control: Selectable hardware / software / 3 wire interface
RF Channel Data Rate: 4800/9600/19,200 bps Half / Full duplex*
Data Buffer: 16 kbyte of on-board RAM
Bit Error Rate:
< 1x10-6 @ -111 dBm (4800 bps)
< 1x10-6 @ -110 dBm (9600 bps)
< 1x10-6 @ -106 dBm (19,200 bps)

*Please check with your local TRIO Datacom representative.

Encryption: 128-bit AES encryption
Collision Avoidance: TRIO Datacom's unique supervisory Channelshare™ collision avoidance system
Multistream™: TRIO Datacom's unique simultaneous delivery of multiple data streams (protocols)
Data Turnaround Time: <10mS
Firmware: Field upgradable Flash memory

This device is OPEN type equipment that must be used within a suitable end-use system enclosure, the interior of which is accessible only through the use of a tool. The suitability of the enclosure is subject to investigation by the local Authority Having Jurisdiction at the time of installation.

General

Power Supply: 13.8 Vdc nominal (10-16 Vdc)
Temp Range: -30degC to +60degC
Transmit Current: 750 mA nom. @ 1 W 1600 mA nom. @5 W
Receive Current: <125 mA nom
Shutdown Mode: External control, < 10 mA
Dimensions: Rugged Diecast Enclosure
170 x 150 x 42mm
6.7 x 5.9 x 1.65 inches
With Mounting Plate
190 x 150 x 47mm
7.5 x 5.9 x 1.85 inches
Mounting: Fitted Mounting Plate
Weight: 1.27 kg (2.8lbs.)

Options

ERFD450 Full Duplex Operation with separate N (Tx) and SMA (Rx) connectors
DUPLX450BR External Duplexer, Band Reject (for single antenna operation)
NEMA 4/R Stainless Steel Enclosure (IP65, NEMA 4 rated)
TVIEW+™ Configuration, Network Management and Diagnostic Windows GUI Software
DIN Rail mounting kit
Part Number : ER-DIN-KIT*

*Not offered in all countries.

Related Products

EB450 Base Station

EH450 Hot Standby Base Station

MSR/9 Port Stream Router Multiplexer

MR450 Remote Data Radio

* With ERFD450 full duplex option plus external duplexer for single antenna operation

HEAD OFFICE (AUSTRALIA)
41 Aster Avenue, Carrum Downs, Victoria 3201.
Tel: +61 3 8773 0100 Fax: +61 3 9775 0606
Sales: sales@triodatacom.com
Support: support@triodatacom.com

NORTH AMERICA
48 Steacie Drive, Kanata, ON Canada K2K 2A9.
Tel: (613) 287 0786 Fax: (613) 591 1022
Toll Free: (866) 844 8746 (TRIO)
Sales: sales-na@triodatacom.com
Support: support-na@triodatacom.com

EUROPE (The Netherlands)
Delftse Jaagpad 1B, 2324 AA Leiden
Tel: +31 71 5320696 Fax: +31 71 5321090
Sales: sales-eu@triodatacom.com
Support: support-eu@triodatacom.com

TRIO DATACOM

Appendix C Function Codes

All RTU messages contain a function code to define the message type. The argument/data formats shown below are port of the standard message structure.

Code	Description	Argument / Data Format
00	Simple Ack	
01	Negative Ack	
02	No access	
03	Message buffer full	(possible response to QSET messages function code 20 and 24)
10	Get Data Frame	[No of IDs (64 max.)] [Start ID msb,lsb]
11	Send Data Frame	[No of IDs (64 max.)] [Start ID msb,lsb] ([Data msb,lsb])...([Data msb,lsb])
12	Get Data Blocks	[No of Blks (64 max.)] ([RTU#][Blk#])..([RTU#][Blk#])
13	Send Data Block	[RTU] [Blk#] [No of Ids(64 max.)] ([Data msb,lsb])...([Data msb,lsb])
14	Request RTU update	[No of Blks (32 max.)]([RTU][Blk#])..([RTU][Blk#])
15	Send RTU update	[No of Blks (32 max.)]([RTU][Blk#][CRC msb,lsb])..([RTU][Blk#][CRC msb,lsb])
16	Get Multi Data	[No of IDs (32 max.)] ([ID msb,lsb])...([ID msb,lsb])
17	Send Multi Data	[No of IDs (32 max.] ([ID msb,lsb][Data msb,lsb])...([ID msb,lsb][Data msb,lsb])
18	Get Multi Network Data	[No of IDs (25 max.)] ([RTU][ID msb,lsb])...([RTU][ID msb,lsb])
19	Send Multi Network Data	[No of IDs (25 max.] ([RTU][ID msb,lsb][Data msb,lsb])...([RTU][ID msb,lsb] [Data msb,lsb])
20	Set Multi Data	[No of IDs (32 max.)] ([ID msb,lsb][Data msb,lsb])...([ID msb,lsb][Data msb,lsb]) (data is stored in local registers)
21	Get Multi Data to Local Registers	[No of IDs (32 max.)] ([local ID msb,lsb][remote ID msb,lsb])…([local ID msb,lsb] [remote ID msb,lsb])
22	Set Data Block	[Block #] [NID (64 max.)]([Data msb,lsb])...([Data msb,lsb])
23	QSet Multi Data	[No of IDs (32 max.)] ([ID msb,lsb][Data msb,lsb])...([ID msb,lsb][Data msb,lsb]) (data is stored in local registers)
24	Set Digital Data	[No of IDs (32 max.)] ([ID msb,lsb][Mask msb,lsb][Data msb,lsb])...([ID msb,lsb] [Mask msb,lsb][Data msb,lsb])
26	Request RTU update	[RTU] (returns CRC's in function code15)
27	QSet Digital Data	[No of IDs (32 max.)] ([ID msb,lsb][Mask msb,lsb][Data msb,lsb])...([ID msb,lsb] [Mask msb,lsb] [Data msb,lsb])
30	Cold Start RTU	
31	Warm Start RTU	
32	Send Program Code (obsolete)	[Code Block#][No of bytes(96 max.)] [Code Bytes]…
33	Program Control	[Control Byte] [Mode]
34	Request RTU Status	
35	Send RTU Status	<V129a: [Version msb,lsb] (Time [Yr][Mth][Day][Hr][Min][Sec][100th]) [Status msb,lsb] [Netblocks msb,lsb][Comms Priority msb,lsb] [Retries msb,lsb][Timeout msb,lsb][Quiet msb,lsb] [Drivers loaded msb,lsb] V1.29a+ [Version msb,lsb] (Time [Yr][Mth][Day][Hr][Min][Sec][100th]) [Status msb,lsb] [Netblocks msb,lsb][Flags msb,lsb][Drivers loaded 1-16 msb,lsb] [Drivers loaded 17-32 msb,lsb] [Drivers loaded 33-48 msb,lsb][I/Oscan rate msb,lsb]
36	Set Real-Time Clock	Year, month, day, hours, minutes, seconds, weekday
37	Swap Master CPU	[master id] [slave id]
38	I/O module message	[slot][command]([data])..([data])
39	System output message	[messagetype][header msb,lsb][group msb,lsb][msg len]([msg data]..[msg data]) message type: 1 = pager message
40	Diagnostic Register	[Version msb,lsb][Diagnostic msb,lsb][reserved msb,lsb]

Code	Description	Argument / Data Format
70	Send File	[Segment Addr msb,lsb][Offset Addr msb,lsb][No of data]([data]..[data]) Note: the LSB of an integer comes before the MSB
71	Request File	[Segment Addr msb,lsb][Offset Addr msb,lsb][No of data] (After FC71 -Request file from RTU, reply is function code 70) Note: the LSB of an integer comes before the MSB
80	Request Event Logging	[function][RTU]([data]..[data]) [01 = get oldest][RTU Filter][No of logs] + optional ([Priority Filter][Usertype Filter][Time msb1,msb2,msb3,lsb]) [02 = get latest][RTU Filter][No of logs] + optional ([Priority Filter][Usertype Filter][Time msb1,msb2,msb3,lsb]) [03 = get previous][RTU Filter][No of logs][Index msb,lsb] + optional ([Priority Filter][Usertype Filter][Time msb1,msb2,msb3,lsb]) [04 = get next][RTU Filter][No of logs][Index msb,lsb] + optional ([Priority Filter][Usertype Filter][Time msb1,msb2,msb3,lsb]) [05 = get number of logs][RTU Filter][Index msb,lsb] + optional ([Priority Filter][Usertype Filter][Time msb1,msb2,msb3,lsb]) [06 = reset logs][RTU]
81	Send Event Logging	[function][RTU]([data]..[data]) <V130a: [01 = send oldest][RTU][No of logs][Index msb,lsb] ([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved] [Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved]) [02 = send latest][RTU][No of logs][Index msb,lsb] ([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved] [Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved]) [03 = send previous][RTU][No of logs][Index msb,lsb] [Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved] [Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved]) [04 = send next][RTU][No of logs][Index msb,lsb] ([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved] [Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref][Reserved]) Maximum No of log entries per message is 12; Ref: = Index in log definition table. V130a+: [11 = send oldest][RTU][No of logs][Index msb,lsb] ([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype])([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype]) [12 = send latest][RTU][No of logs][Index msb,lsb] ([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype])([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype]) [13 = send previous][RTU][No of logs][Index msb,lsb]([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype])([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype]) [14 = send next][RTU][No of logs][Index msb,lsb]([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype])([Time msb1,msb2,msb3,lsb][msec msb,lsb][Value msb,lsb][Ref msb,lsb][RTU][Usertype]) Maximum No of log entries per message is 10; Ref: = Number of system log or Data ID. (all firmware versions) [05 = send number of logs][RTU][No of logs msb,lsb] [06 = ack reset logs][RTU]

Appendix D – Spreadsheet columns

The spreadsheet shows the following parameters:

Blocks: Groups of 64 Registers

Registers: Internal 16 bit memory allocations

Baud: Rate of change of signal

Data (Bytes): Refer to Calculations and KF S2

Data overhead (Bytes): For 8 Data bits there is 1 stop and 1 start bit (refer to doc)

Data overhead (Bytes): Sum of Data and Data overheads

Calculated data + overheads (s): Transmission time of data (without idle periods)

Predicted Transmission Time: Calculated transmission time for x Registers

% Error: Percentage of value deviation from the measured value

Regression Calculation: Statistical Regression Analysis of Predicted Transmission Time (more detailed explanation refers to key coefficients much)

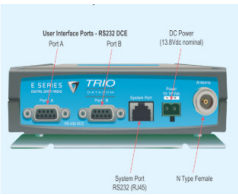
Total message length: Measured values from the system

Regression measurement: Statistical Regression Analysis of Predicted Transmission Time

Calculated message delay: Idle periods between messages

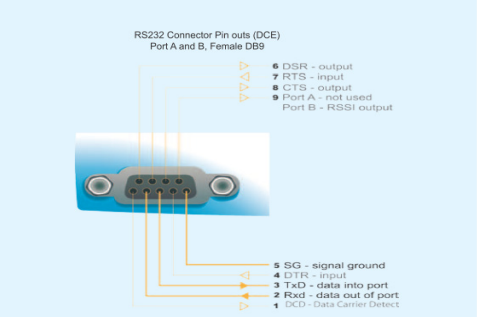
Radio delay: Radio link value – Serial link value

Appendix E – Trio radio interface



Wiring diagram (DB9 – RJ45)

DB9 connection for radio modem communications port



RJ45 connection to RTU Port

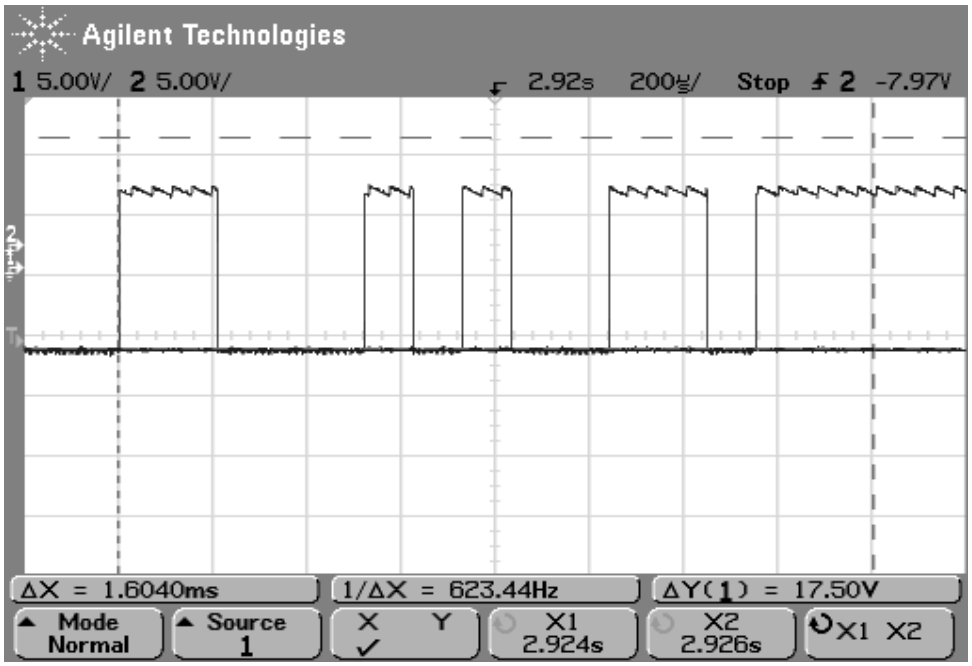
KINGFISHER RTU
RADIO PORT (RJ45)

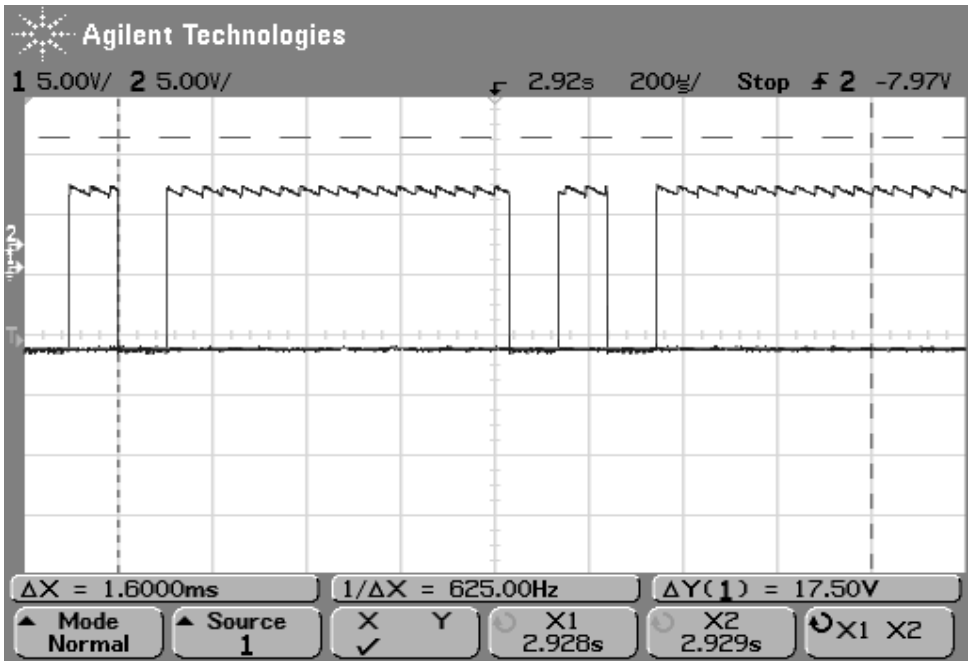
1	BROWN	TXD
2	BROWN/WHI	RXD
3	ORANGE	CTS
6	ORANG/WHI	RTS
4	BLUE/WHI	GND
8	GREEN/WHI	DTR
5	BLUE	DCD
7 *	GREEN	+12V

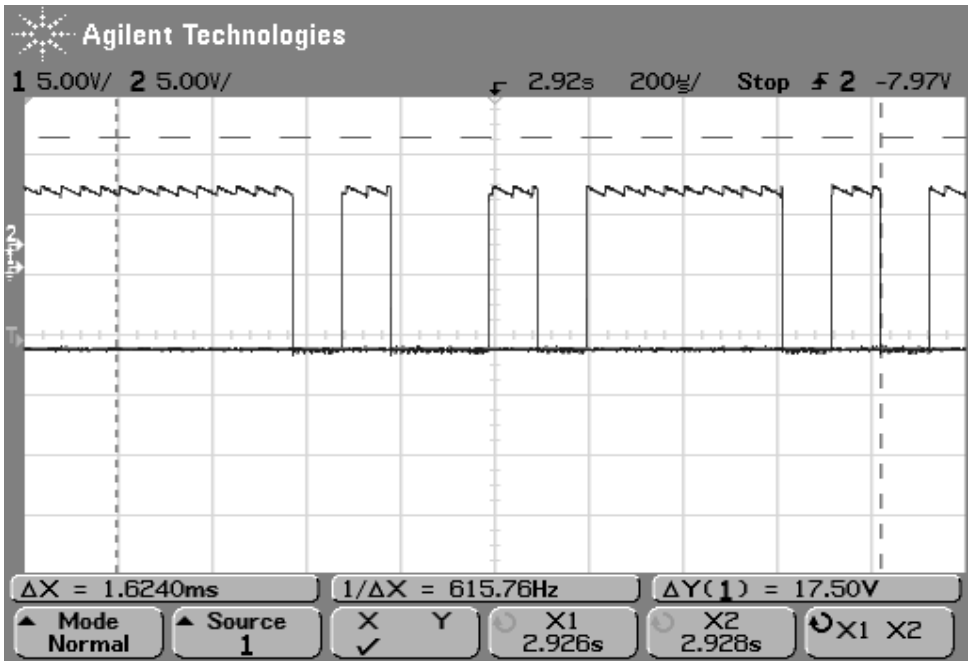
Appendix F – Data Structure KF S2

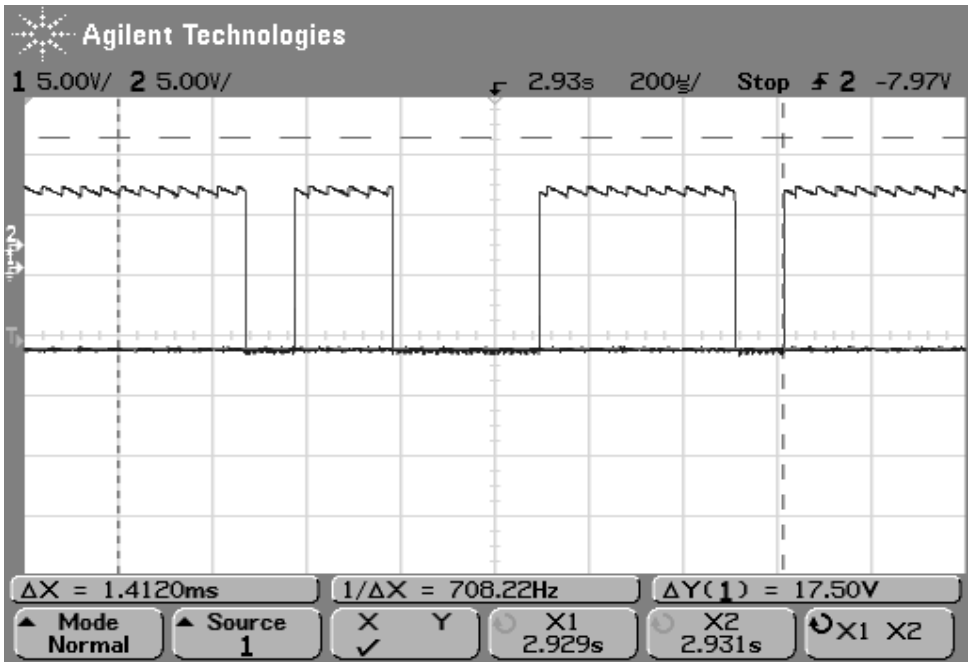
This Appendix shows the bit signal representation of the (rx_data) communication block message commands upon execution in the ladder logic software.

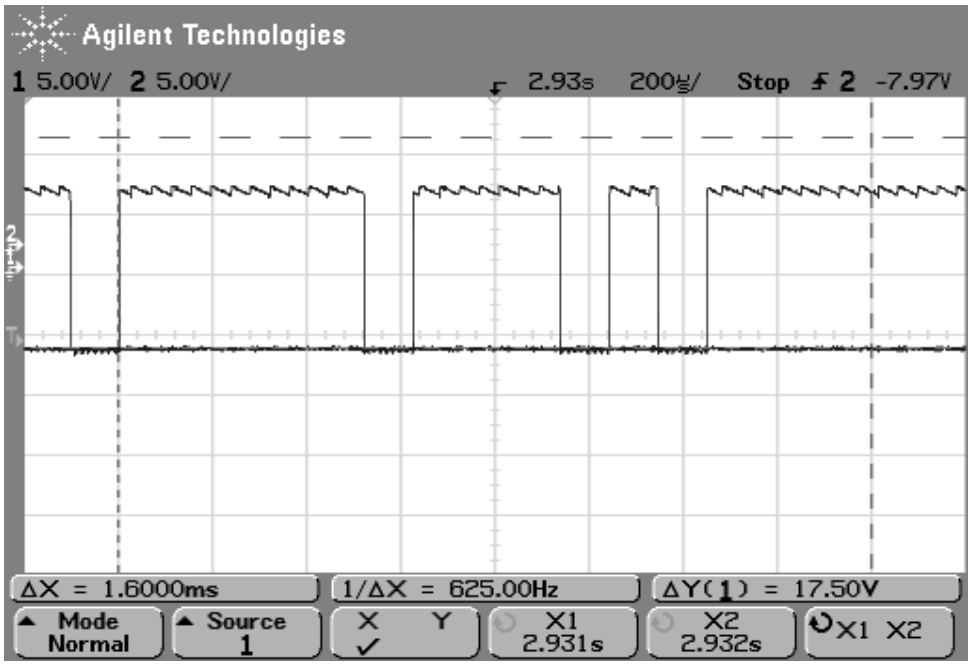
Waveform 1: Command 16 Get Multi Data

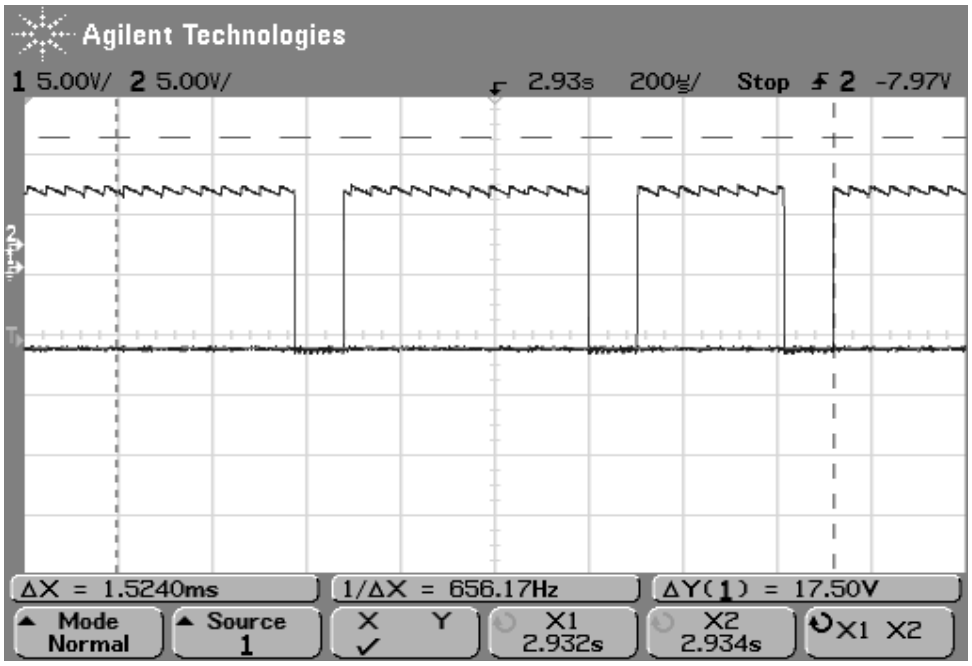


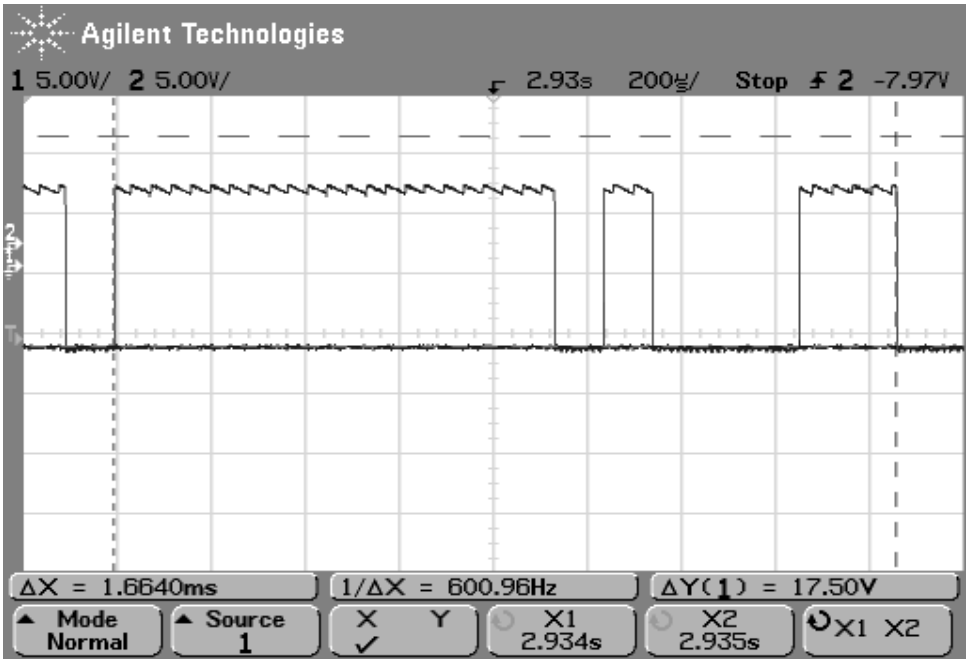


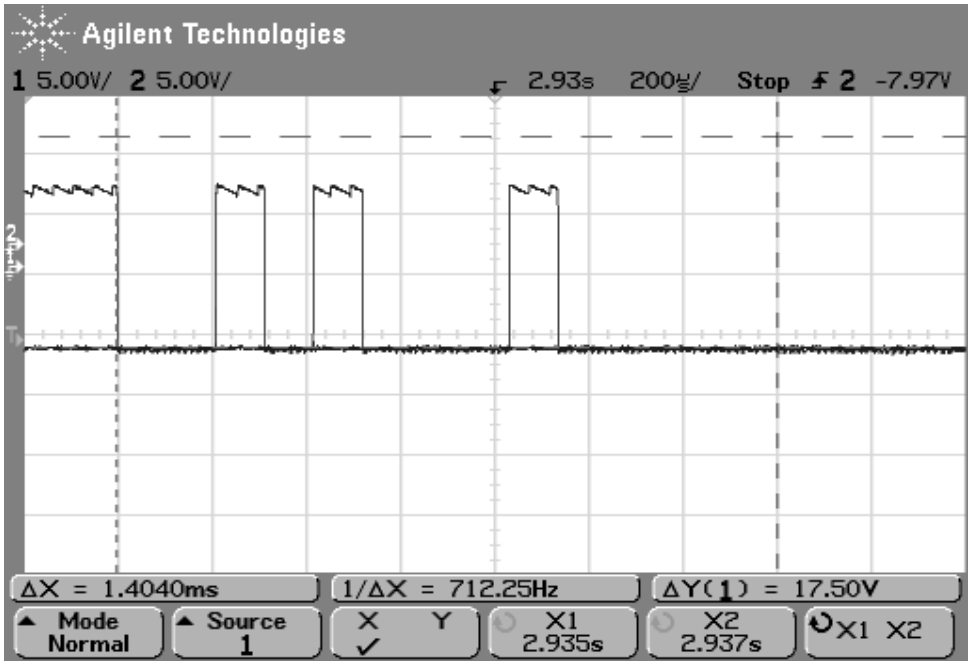

Agilent Technologies
1 5.00V/ 2 5.00V/
2.92s 200µ/ Stop 2 -7.97V
ΔX = 1.6240ms
1/ΔX = 615.76Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.926s
X2 2.928s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
2.93s 200µ/ Stop 2 -7.97V
ΔX = 1.4120ms
1/ΔX = 708.22Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.929s
X2 2.931s
X1 X2

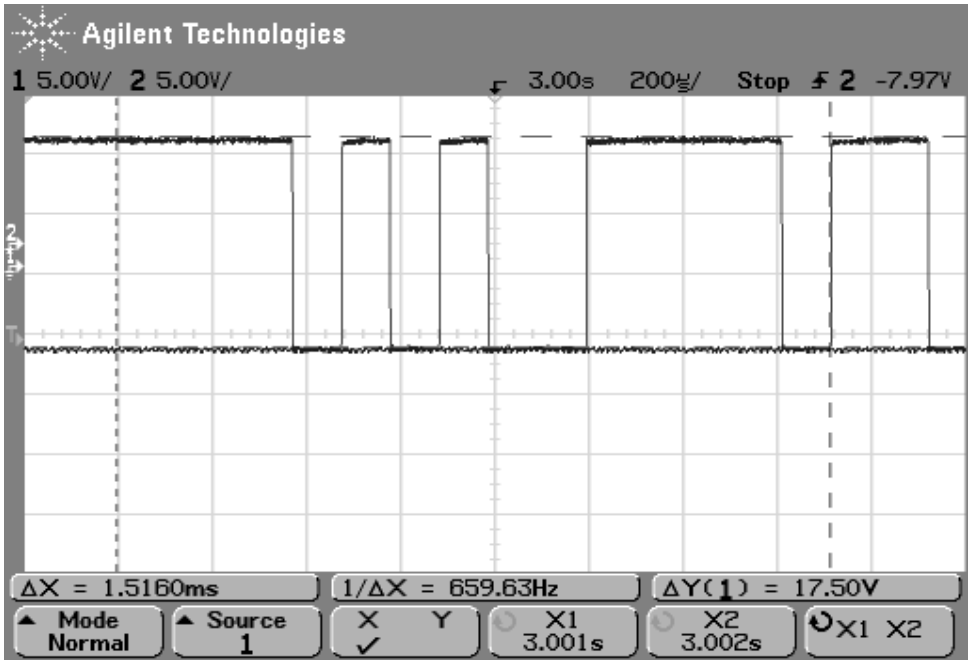

Agilent Technologies
1 5.00V/ 2 5.00V/
2.93s 200ns/ Stop 2 -7.97V
ΔX = 1.6000ms
1/ΔX = 625.00Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.931s
X2 2.932s
X1 X2

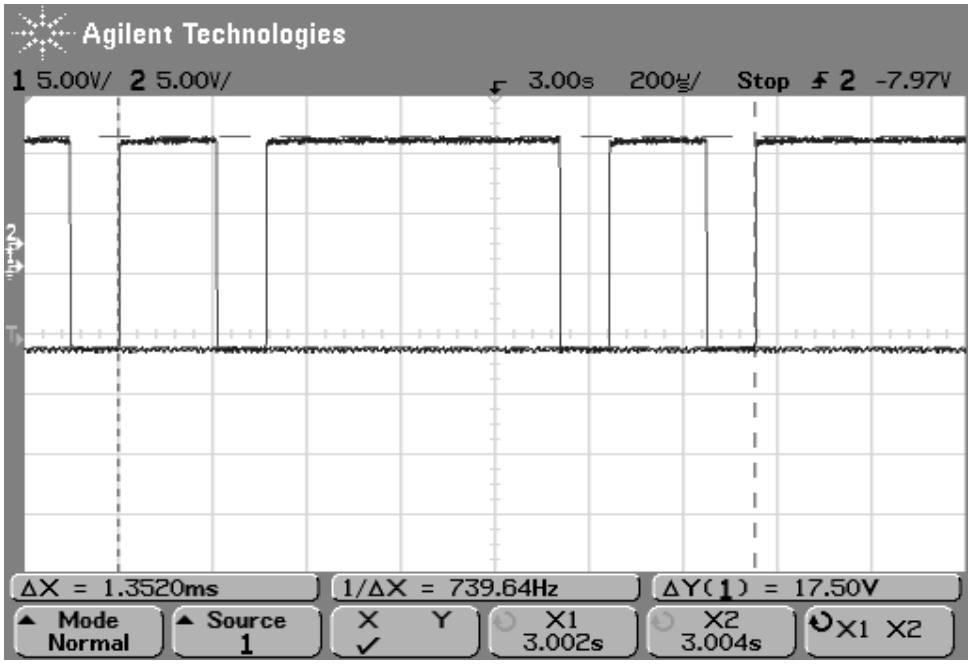

Agilent Technologies
1 5.00V/ 2 5.00V/
2.93s 200ns/ Stop 2 -7.97V
ΔX = 1.5240ms
1/ΔX = 656.17Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.932s
X2 2.934s
X1 X2

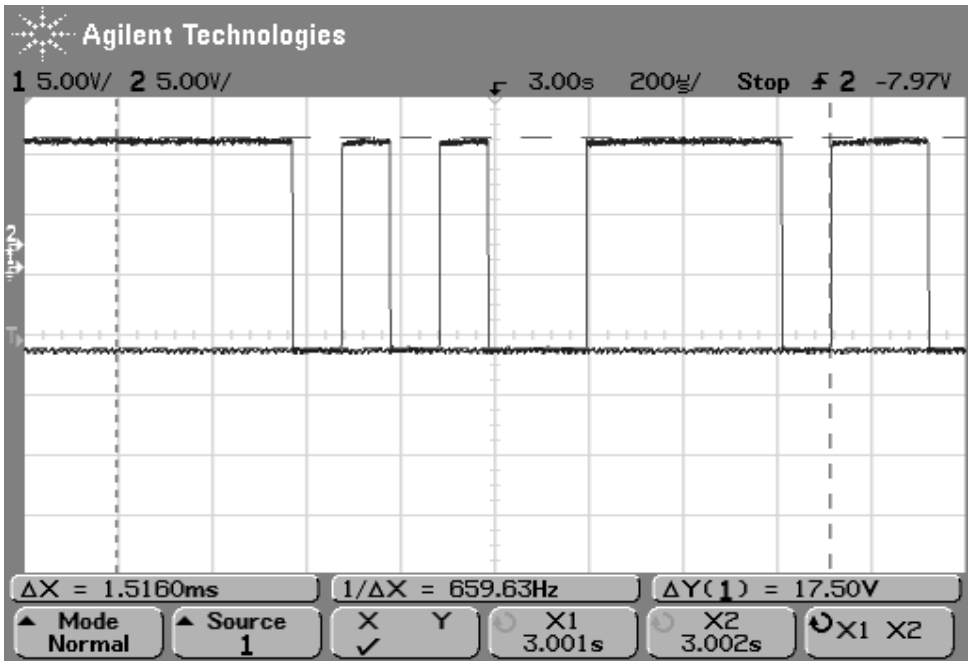

Agilent Technologies
1 5.00V/ 2 5.00V/
2.93s 200µ/ Stop 2 -7.97V
ΔX = 1.6640ms
1/ΔX = 600.96Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.934s
X2 2.935s
X1 X2

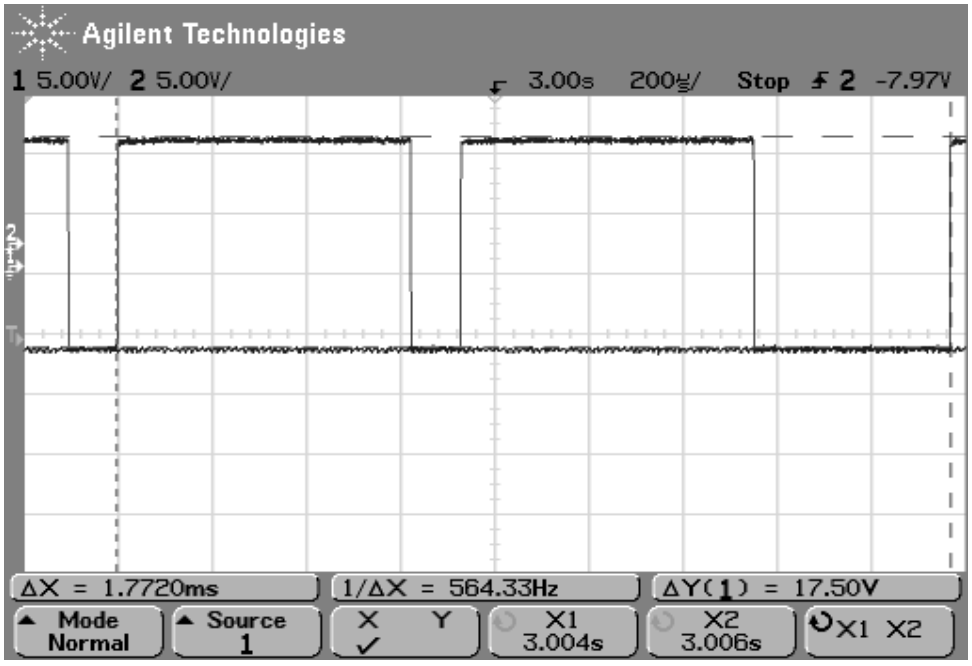

Agilent Technologies
1 5.00V/ 2 5.00V/
2.93s 200µ/ Stop 2 -7.97V
ΔX = 1.4040ms
1/ΔX = 712.25Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 2.935s
X2 2.937s
X1 X2

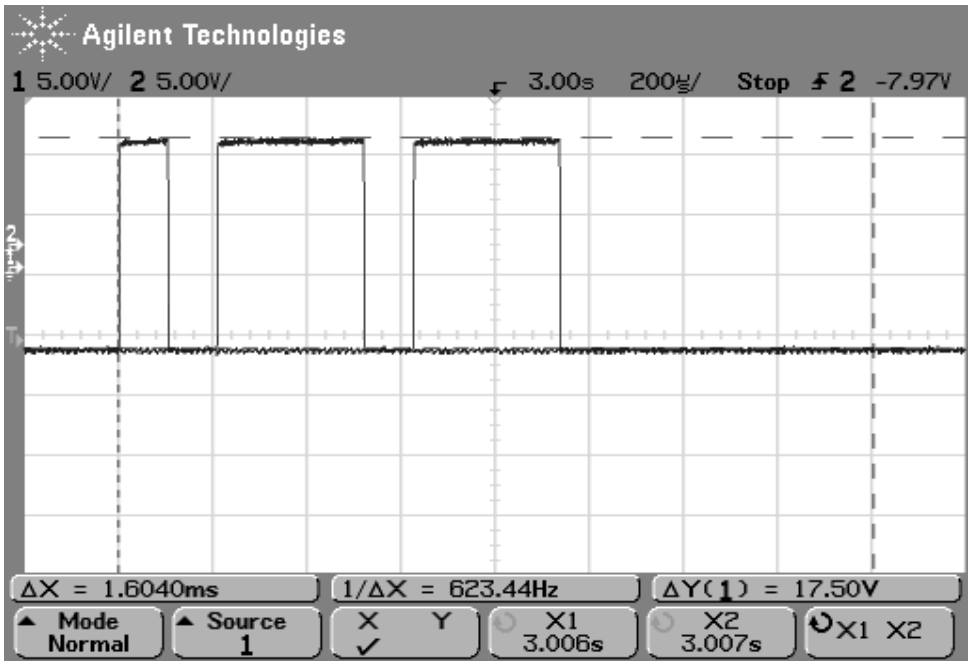
Waveform 2.0: Command #17 Send Multi Data

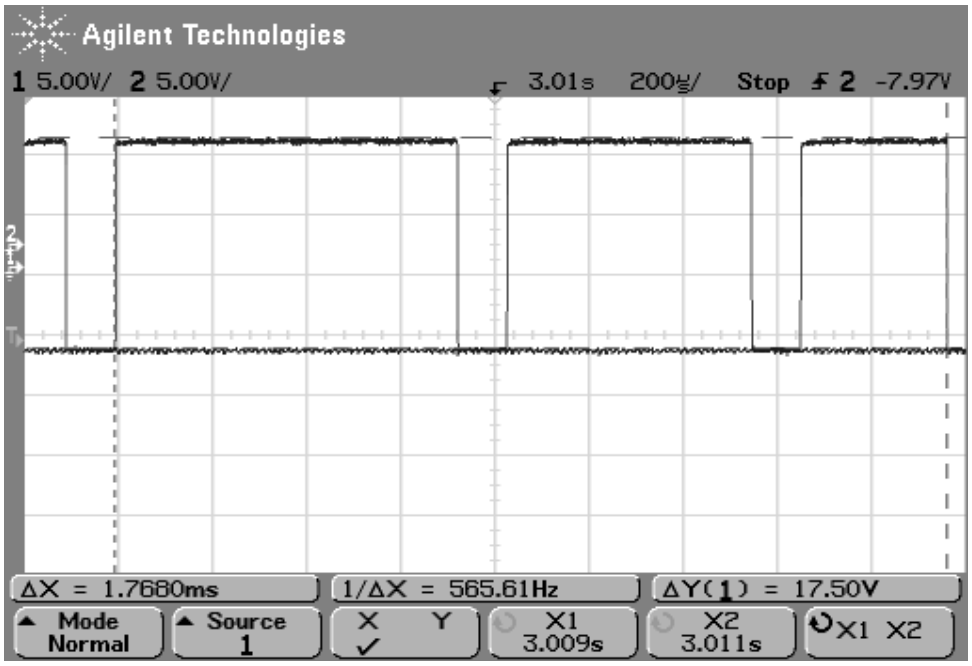


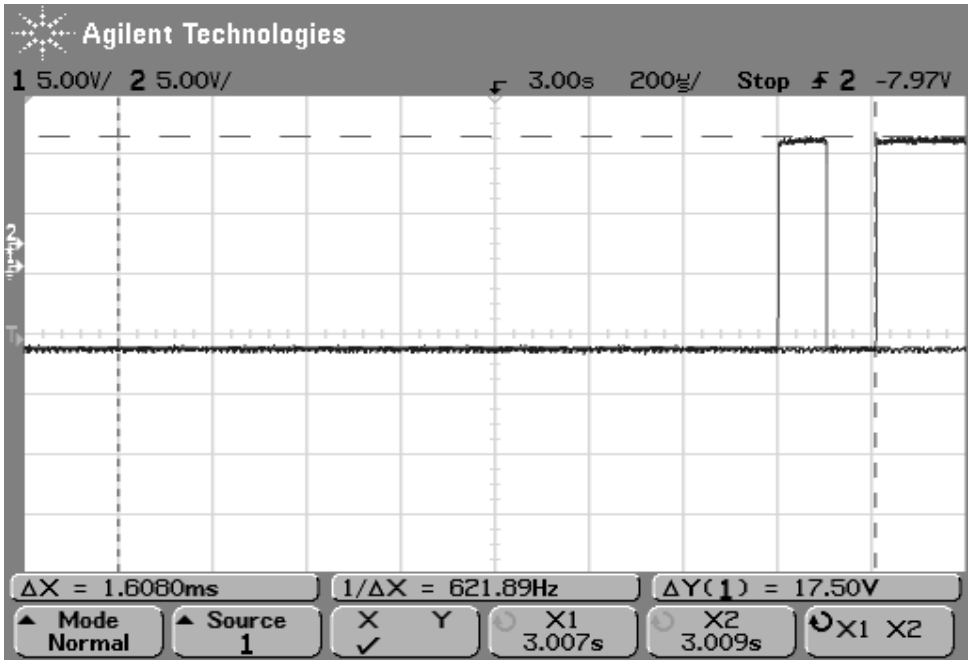


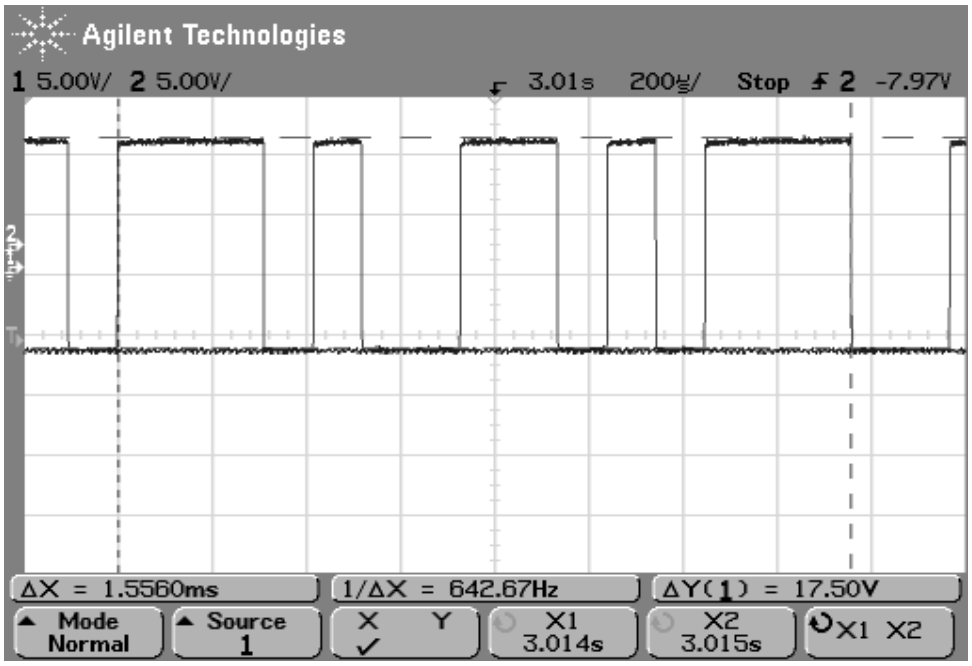

Agilent Technologies
1 5.00V/ 2 5.00V/
3.00s 200µ/ Stop 2 -7.97V
ΔX = 1.5160ms
1/ΔX = 659.63Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.001s
X2 3.002s
X1 X2

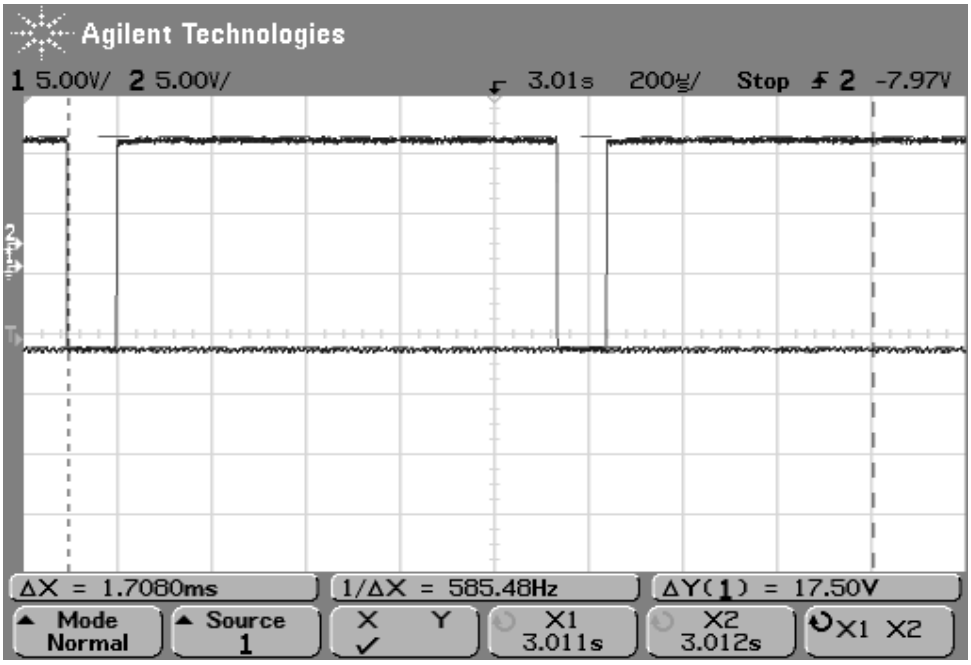

Agilent Technologies
1 5.00V/ 2 5.00V/
3.00s 200µ/ Stop 2 -7.97V
ΔX = 1.7720ms
1/ΔX = 564.33Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.004s
X2 3.006s
X1 X2

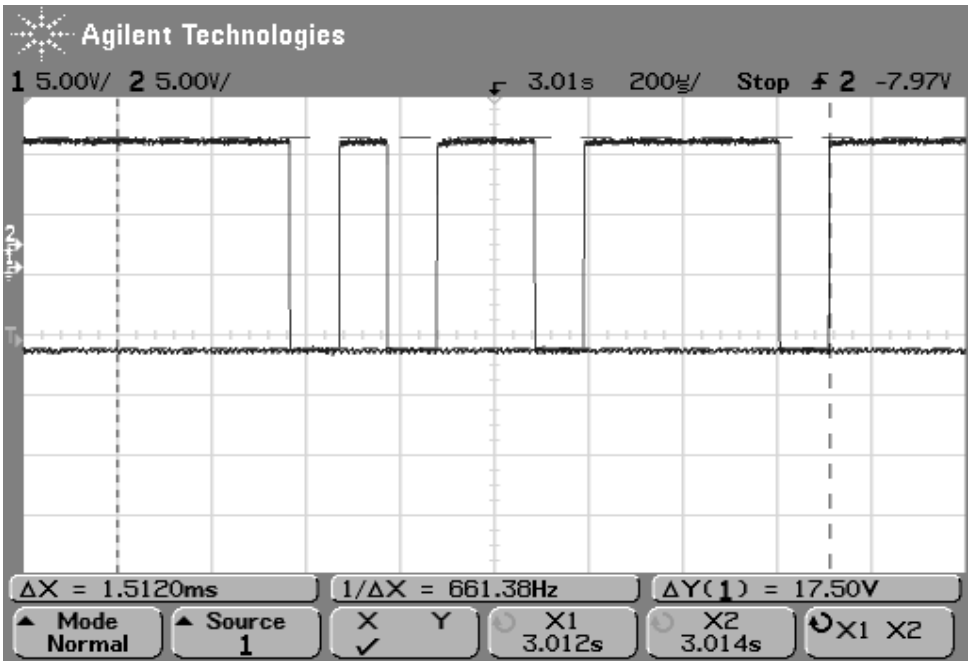

Agilent Technologies
1 5.00V/ 2 5.00V/
3.00s 200ks/ Stop 2 -7.97V
ΔX = 1.6040ms
1/ΔX = 623.44Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.006s
X2 3.007s
X1 X2

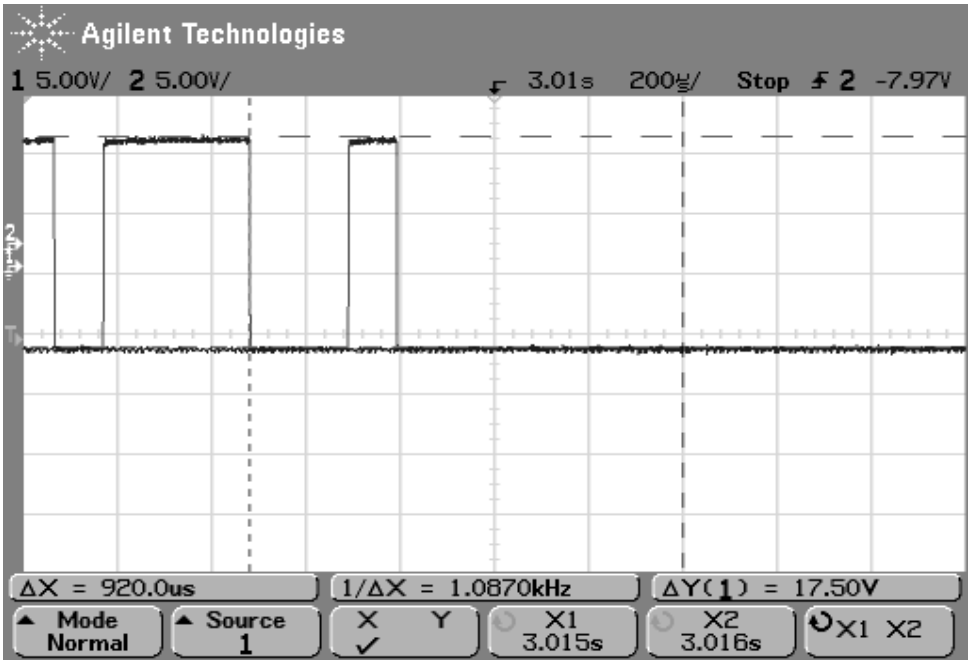

Agilent Technologies
1 5.00V/ 2 5.00V/
3.01s 200ks/ Stop 2 -7.97V
ΔX = 1.7680ms
1/ΔX = 565.61Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.009s
X2 3.011s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
3.00s 200ks/ Stop 2 -7.97V
ΔX = 1.6080ms
1/ΔX = 621.89Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.007s
X2 3.009s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
3.01s 200ks/ Stop 2 -7.97V
ΔX = 1.5560ms
1/ΔX = 642.67Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.014s
X2 3.015s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
3.01s 200s/ Stop 2 -7.97V
ΔX = 1.7080ms
1/ΔX = 585.48Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.011s
X2 3.012s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
3.01s 200s/ Stop 2 -7.97V
ΔX = 1.5120ms
1/ΔX = 661.38Hz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.012s
X2 3.014s
X1 X2


Agilent Technologies
1 5.00V/ 2 5.00V/
3.01s 200s/ Stop 2 -7.97V
ΔX = 920.0us
1/ΔX = 1.0870kHz
ΔY(1) = 17.50V
Mode Normal
Source 1
X Y
X1 3.015s
X2 3.016s
X1 X2

Appendix G – Event Log Service Period

Shows the polling of event log data from period (11:51:28 to 11:52:08) = 40 s service duration with a maximum polling configuration of 1000 logs.

Tx: 11:51:28: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 09 22 00 00 88 72

Rx: 11:51:28: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 09 2C 3A 6D 00 61 01 9F 20 DE 54 05 02 01 3A 6D 00 61 03 92 20 DF 54 01 02 01 3A 6D 00 61 03 92 20 DF 54 02 02 01 3A 6D 00 61 03 92 20 DF 54 03 02 01 3A 6D 00 61 03 92 20 DF 54 04 02 01 3A 6D 00 61 03 93 20 DF 54 05 02 01 3A 6D 00 62 01 A3 20 E0 54 01 02 01 3A 6D 00 62 01 A4 20 E0 54 02 02 01 3A 6D 00 62 01 A4 20 E0 54 03 02 01 3A 6D 00 62 01 A4 20 E0 54 04 02 01 04 F8

Tx: 11:51:28: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 09 2C 00 00 D1 DD

Rx: 11:51:29: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 09 36 3A 6D 00 62 01 A4 20 E0 54 05 02 01 3A 6D 00 62 03 93 20 E1 54 01 02 01 3A 6D 00 62 03 93 20 E1 54 02 02 01 3A 6D 00 62 03 94 20 E1 54 03 02 01 3A 6D 00 62 03 94 20 E1 54 04 02 01 3A 6D 00 62 03 94 20 E1 54 05 02 01 3A 6D 00 63 02 09 20 E2 54 01 02 01 3A 6D 00 63 02 0A 20 E2 54 02 02 01 3A 6D 00 63 02 0B 20 E2 54 03 02 01 3A 6D 00 63 02 0B 20 E2 54 04 02 01 3B B4

Tx: 11:51:29: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 09 36 00 00 BA 24

Rx: 11:51:29: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 09 40 3A 6D 00 63 02 0B 20 E2 54 05 02 01 3A 6D 00 64 00 19 20 E3 54 01 02 01 3A 6D 00 64 00 19 20 E3 54 02 02 01 3A 6D 00 64 00 1A 20 E3 54 03 02 01 3A 6D 00 64 00 1A 20 E3 54 04 02 01 3A 6D 00 64 00 1A 20 E3 54 05 02 01 3A 6D 00 64 02 6A 20 E4 54 01 02 01 3A 6D 00 64 02 6A 20 E4 54 02 02 01 3A 6D 00 64 02 6A 20 E4 54 03 02 01 3A 6D 00 64 02 6A 20 E4 54 04 02 01 63 C1

Tx: 11:51:29: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 09 40 00 00 1C 14

Rx: 11:51:29: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 09 4A 3A 6D 00 64 02 6B 20 E4 54 05 02 01 3A 6D 00 65 00 D2 20 E5 54 01 02 01 3A 6D 00 65 00 D2 20 E5 54 02 02 01 3A 6D 00 65 00 D2 20 E5 54 03 02 01 3A 6D 00 65 00 D3 20 E5 54 04 02 01 3A 6D 00 65 00 D3 20 E5 54 05 02 01 3A 6D 00 65 02 C8 20 E6 54 01 02 01 3A 6D 00 65 02 C9 20 E6 54 02 02 01 3A 6D 00 65 02 C9 20 E6 54 03 02 01 3A 6D 00 65 02 C9 20 E6 54 04 02 01 36 E8

Tx: 11:51:29: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 09 4A 00 00 8F D6

Rx: 11:51:30: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 09 54 3A 6D 00 65 02 C9 20 E6 54 05 02 01 3A 6D 00 66 00 DA 20 E7 54 01 02 01 3A 6D 00 66 00 DA 20 E7 54 02 02 01 3A 6D 00 66 00 DA 20 E7 54 03 02 01 3A 6D 00 66 00 DA 20 E7 54 04 02 01 3A 6D 00 66 00 DB 20 E7 54 05 02 01 3A 6D 00 66 02 CA 20 E8 54 01 02 01 3A 6D 00 66 02 CA 20 E8 54 02 02 01 3A 6D 00 66 02 CA 20 E8 54 03 02 01 3A 6D 00 66 02 CA 20 E8 54 04 02 01 FD E0

Tx: 11:51:30: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 09 54 00 00 A4 AB

Rx: 11:51:30: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 09 5E 3A 6D 00 66 02 CB 20 E8 54 05 02 01 3A 6D 00 67 00 DA 20 E9 54 01 02 01 3A 6D 00 67 00 DB 20 E9 54 02 02 01 3A 6D 00 67 00 DB 20 E9 54 03 02 01 3A 6D 00 67 00 DB 20 E9 54 04 02 01 3A 6D 00 67 00 DB 20 E9 54 05 02 01 3A 6D 00 67 03 2D 20 EA 54 01 02 01 3A 6D 00 67 03 2E 20 EA 54 02 02 01 3A 6D 00 67 03 2E 20 EA 54 03 02 01 3A 6D 00 67 03 2E 20 EA 54 04 02 01 D4 C1

Tx: 11:51:30: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 09 5E 00 00 BD 80

Rx: 11:51:31: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 09 68 3A 6D 00 67 03 2E 20 EA 54 05 02 01 3A 6D 00 68 01 42 20 EB 54 01 02 01 3A 6D 00 68 01 42 20 EB 54 02 02 01 3A 6D 00 68 01 42 20 EB 54 03 02 01 3A 6D 00 68 01 42 20 EB 54 04 02 01 3A 6D 00 68 01 43 20 EB 54 05 02 01 3A 6D 00 68 03 36 20 EC 54 01 02 01 3A 6D 00 68 03 36 20 EC 54 02 02 01 3A 6D 00 68 03 36 20 EC 54 03 02 01 3A 6D 00 68 03 36 20 EC 54 04 02 01 21 E4

Tx: 11:51:31: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 09 68 00 00 F2 51

Rx: 11:51:31: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 09 72 3A 6D 00 68 03 37 20 EC 54 05 02 01 3A 6D 00 69 01 3D 20 ED 54 01 02 01 3A 6D 00 69 01 3E 20 ED 54 02 02 01 3A 6D 00 69 01 3E 20 ED 54 03 02 01 3A 6D 00 69 01 3E 20 ED 54 04 02 01 3A 6D 00 69 01 3F 20 ED 54 05 02 01 3A 6D 00 69 03 33 20 EE 54 01 02 01 3A 6D 00 69 03 34 20 EE 54 02 02 01 3A 6D 00 69 03 34 20 EE 54 03 02 01 3A 6D 00 69 03 34 20 EE 54 04 02 01 D6 5B

Tx: 11:51:31: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 09 72 00 00 F9 4B

Rx: 11:51:31: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 09 7C 3A 6D 00 69 03 34 20 EE 54 05 02 01 3A 6D 00 6A 01 41 20 EF 54 01 02 01 3A 6D 00 6A 01 41 20 EF 54 02 02 01 3A 6D 00 6A 01 42 20 EF 54 03 02 01 3A 6D 00 6A 01 42 20 EF 54 04 02 01 3A 6D 00 6A 01 42 20 EF 54 05 02 01 3A 6D 00 6A 03 35 20 F0 54 01 02 01 3A 6D 00 6A 03 35 20 F0 54 02 02 01 3A 6D 00 6A 03 36 20 F0 54 03 02 01 3A 6D 00 6A 03 36 20 F0 54 04 02 01 9C 90

Tx: 11:51:31: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 09 7C 00 00 C0 07

Rx: 11:51:32: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 09 86 3A 6D 00 6A 03 36 20 F0 54 05 02 01 3A 6D 00 6B 01 42 20 F1 54 01 02 01 3A 6D 00 6B 01 42 20 F1 54 02 02 01 3A 6D 00 6B 01 43 20 F1 54 03 02 01 3A 6D 00 6B 01 43 20 F1 54 04 02 01 3A 6D 00 6B 01 43 20 F1 54 05 02 01 3A 6D 00 6B 03 2F 20 F2 54 01 02 01 3A 6D 00 6B 03 2F 20 F2 54 02 02 01 3A 6D 00 6B 03 2F 20 F2 54 03 02 01 3A 6D 00 6B 03 2F 20 F2 54 04 02 01 8E 70

Tx: 11:51:32: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 09 86 00 00 36 33

Rx: 11:51:32: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 09 90 3A 6D 00 6B 03 30 20 F2 54 05 02 01 3A 6D 00 6C 01 42 20 F3 54 01 02 01 3A 6D 00 6C 01 42 20 F3 54 02 02 01 3A 6D 00 6C 01 42 20 F3 54 03 02 01 3A 6D 00 6C 01 43 20 F3 54 04 02 01 3A 6D 00 6C 01 43 20 F3 54 05 02 01 3A 6D 00 6C 03 94 20 F4 54 01 02 01 3A 6D 00 6C 03 94 20 F4 54 02 02 01 3A 6D 00 6C 03 95 20 F4 54 03 02 01 3A 6D 00 6C 03 95 20 F4 54 04 02 01 2D 1E

Tx: 11:51:32: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 09 90 00 00 CE 2D

Rx: 11:51:32: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 09 9A 3A 6D 00 6C 03 95 20 F4 54 05 02 01 3A 6D 00 6D 01 A4 20 F5 54 01 02 01 3A 6D 00 6D 01 A4 20 F5 54 02 02 01 3A 6D 00 6D 01 A4 20 F5 54 03 02 01 3A 6D 00 6D 01 A5 20 F5 54 04 02 01 3A 6D 00 6D 01 A5 20 F5 54 05 02 01 3A 6D 00 6D 03 92 20 F6 54 01 02 01 3A 6D 00 6D 03 93 20 F6 54 02 02 01 3A 6D 00 6D 03 93 20 F6 54 03 02 01 3A 6D 00 6D 03 93 20 F6 54 04 02 01 63 8E

Tx: 11:51:33: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 09 9A 00 00 D7 06

Rx: 11:51:33: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 09 A4 3A 6D 00 6D 03 94 20 F6 54 05 02 01 3A 6D 00 6E 01 A0 20 F7 54 01 02 01 3A 6D 00 6E 01 A0 20 F7 54 02 02 01 3A 6D 00 6E 01 A0 20 F7 54 03 02 01 3A 6D 00 6E 01 A1 20 F7 54 04 02 01 3A 6D 00 6E 01 A1 20 F7 54 05 02 01 3A 6D 00 6E 03 92 20 F8 54 01 02 01 3A 6D 00 6E 03 92 20 F8 54 02 02 01 3A 6D 00 6E 03 93 20 F8 54 03 02 01 3A 6D 00 6E 03 93 20 F8 54 04 02 01 B7 00

Tx: 11:51:33: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 09 A4 00 00 D8 19

Rx: 11:51:33: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 09 AE 3A 6D 00 6E 03 93 20 F8 54 05 02 01 3A 6D 00 6F 01 A6 20 F9 54 01 02 01 3A 6D 00 6F 01 A6 20 F9 54 02 02 01 3A 6D 00 6F 01 A6 20 F9 54 03 02 01 3A 6D 00 6F 01 A6 20 F9 54 04 02 01 3A 6D 00 6F 01 A7 20 F9 54 05 02 01 3A 6D 00 6F 03 99 20 FA 54 01 02 01 3A 6D 00 6F 03 9A 20 FA 54 02 02 01 3A 6D 00 6F 03 9A 20 FA 54 03 02 01 3A 6D 00 6F 03 9A 20 FA 54 04 02 01 AE FF

Tx: 11:51:33: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 09 AE 00 00 C1 32

Rx: 11:51:34: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 09 B8 3A 6D 00 6F 03 9A 20 FA 54 05 02 01 3A 6D 00 70 01 A5 20 FB 54 01 02 01 3A 6D 00 70 01 A6 20 FB 54 02 02 01 3A 6D 00 70 01 A6 20 FB 54 03 02 01 3A 6D 00 70 01 A6 20 FB 54 04 02 01 3A 6D 00 70 01 A6 20 FB 54 05 02 01 3A 6D 00 71 00 17 20 FC 54 01 02 01 3A 6D 00 71 00 17 20 FC 54 02 02 01 3A 6D 00 71 00 18 20 FC 54 03 02 01 3A 6D 00 71 00 18 20 FC 54 04 02 01 D0 DA

Tx: 11:51:34: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 09 B8 00 00 AA 81

Rx: 11:51:34: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 09 C2 3A 6D 00 71 00 19 20 FC 54 05 02 01 3A 6D 00 71 02 67 20 FD 54 01 02 01 3A 6D 00 71 02 67 20 FD 54 02 02 01 3A 6D 00 71 02 67 20 FD 54 03 02 01 3A 6D 00 71 02 67 20 FD 54 04 02 01 3A 6D 00 71 02 68 20 FD 54 05 02 01 3A 6D 00 72 00 74 20 FE 54 01 02 01 3A 6D 00 72 00 74 20 FE 54 02 02 01 3A 6D 00 72 00 75 20 FE 54 03 02 01 3A 6D 00 72 00 75 20 FE 54 04 02 01 B6 78

Tx: 11:51:34: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 09 C2 00 00 CD 3D

Rx: 11:51:34: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 09 CC 3A 6D 00 72 00 75 20 FE 54 05 02 01 3A 6D 00 72 02 CA 20 FF 54 01 02 01 3A 6D 00 72 02 CA 20 FF 54 02 02 01 3A 6D 00 72 02 CB 20 FF 54 03 02 01 3A 6D 00 72 02 CB 20 FF 54 04 02 01 3A 6D 00 72 02 CC 20 FF 54 05 02 01 3A 6D 00 73 00 DE 21 00 54 01 02 01 3A 6D 00 73 00 DE 21 00 54 02 02 01 3A 6D 00 73 00 DE 21 00 54 03 02 01 3A 6D 00 73 00 DF 21 00 54 04 02 01 69 F7

Tx: 11:51:35: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 09 CC 00 00 F4 71

Rx: 11:51:35: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 09 D6 3A 6D 00 73 00 DF 21 00 54 05 02 01 3A 6D 00 73 02 D0 21 01 54 01 02 01 3A 6D 00 73 02 D1 21 01 54 02 02 01 3A 6D 00 73 02 D1 21 01 54 03 02 01 3A 6D 00 73 02 D1 21 01 54 04 02 01 3A 6D 00 73 02 D1 21 01 54 05 02 01 3A 6D 00 74 00 DC 21 02 54 01 02 01 3A 6D 00 74 00 DD 21 02 54 02 02 01 3A 6D 00 74 00 DD 21 02 54 03 02 01 3A 6D 00 74 00 DD 21 02 54 04 02 01 99 8F

Tx: 11:51:35: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 09 D6 00 00 FF 6B

Rx: 11:51:35: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 09 E0 3A 6D 00 74 00 DD 21 02 54 05 02 01 3A 6D 00 74 02 D0 21 03 54 01 02 01 3A 6D 00 74 02 D1 21 03 54 02 02 01 3A 6D 00 74 02 D1 21 03 54 03 02 01 3A 6D 00 74 02 D1 21 03 54 04 02 01 3A 6D 00 74 02 D1 21 03 54 05 02 01 3A 6D 00 75 00 E6 21 04 54 01 02 01 3A 6D 00 75 00 E6 21 04 54 02 02 01 3A 6D 00 75 00 E7 21 04 54 03 02 01 3A 6D 00 75 00 E7 21 04 54 04 02 01 2C F2

Tx: 11:51:35: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 09 E0 00 00 9B 76

Rx: 11:51:36: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 09 EA 3A 6D 00 75 00 E7 21 04 54 05 02 01 3A 6D 00 75 02 D2 21 05 54 01 02 01 3A 6D 00 75 02 D2 21 05 54 02 02 01 3A 6D 00 75 02 D2 21 05 54 03 02 01 3A 6D 00 75 02 D2 21 05 54 04 02 01 3A 6D 00 75 02 D3 21 05 54 05 02 01 3A 6D 00 76 00 E1 21 06 54 01 02 01 3A 6D 00 76 00 E2 21 06 54 02 02 01 3A 6D 00 76 00 E2 21 06 54 03 02 01 3A 6D 00 76 00 E2 21 06 54 04 02 01 85 53

Tx: 11:51:36: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 09 EA 00 00 E2 BE

Rx: 11:51:36: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 09 F4 3A 6D 00 76 00 E2 21 06 54 05 02 01 3A 6D 00 76 02 D1 21 07 54 01 02 01 3A 6D 00 76 02 D2 21 07 54 02 02 01 3A 6D 00 76 02 D2 21 07 54 03 02 01 3A 6D 00 76 02 D2 21 07 54 04 02 01 3A 6D 00 76 02 D2 21 07 54 05 02 01 3A 6D 00 77 00 DE 21 08 54 01 02 01 3A 6D 00 77 00 DF 21 08 54 02 02 01 3A 6D 00 77 00 DF 21 08 54 03 02 01 3A 6D 00 77 00 DF 21 08 54 04 02 01 D4 02

Tx: 11:51:36: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 09 F4 00 00 A9 20

Rx: 11:51:36: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 09 FE 3A 6D 00 77 00 DF 21 08 54 05 02 01 3A 6D 00 77 02 D6 21 09 54 01 02 01 3A 6D 00 77 02 D6 21 09 54 02 02 01 3A 6D 00 77 02 D7 21 09 54 03 02 01 3A 6D 00 77 02 D7 21 09 54 04 02 01 3A 6D 00 77 02 D7 21 09 54 05 02 01 3A 6D 00 78 01 40 21 0A 54 01 02 01 3A 6D 00 78 01 40 21 0A 54 02 02 01 3A 6D 00 78 01 41 21 0A 54 03 02 01 3A 6D 00 78 01 42 21 0A 54 04 02 01 79 A1

Tx: 11:51:37: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 09 FE 00 00 11 2E

Rx: 11:51:37: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 0A 08 3A 6D 00 78 01 42 21 0A 54 05 02 01 3A 6D 00 78 03 36 21 0B 54 01 02 01 3A 6D 00 78 03 37 21 0B 54 02 02 01 3A 6D 00 78 03 37 21 0B 54 03 02 01 3A 6D 00 78 03 37 21 0B 54 04 02 01 3A 6D 00 78 03 38 21 0B 54 05 02 01 3A 6D 00 79 01 46 21 0C 54 01 02 01 3A 6D 00 79 01 46 21 0C 54 02 02 01 3A 6D 00 79 01 47 21 0C 54 03 02 01 3A 6D 00 79 01 47 21 0C 54 04 02 01 53 1E

Tx: 11:51:37: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 0A 08 00 00 73 C5

Rx: 11:51:37: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 0A 12 3A 6D 00 79 01 47 21 0C 54 05 02 01 3A 6D 00 79 03 3C 21 0D 54 01 02 01 3A 6D 00 79 03 3C 21 0D 54 02 02 01 3A 6D 00 79 03 3C 21 0D 54 03 02 01 3A 6D 00 79 03 3D 21 0D 54 04 02 01 3A 6D 00 79 03 3D 21 0D 54 05 02 01 3A 6D 00 7A 01 44 21 0E 54 01 02 01 3A 6D 00 7A 01 44 21 0E 54 02 02 01 3A 6D 00 7A 01 44 21 0E 54 03 02 01 3A 6D 00 7A 01 45 21 0E 54 04 02 01 E8 20

Tx: 11:51:37: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 0A 12 00 00 18 3C

Rx: 11:51:38: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 0A 1C 3A 6D 00 7A 01 45 21 0E 54 05 02 01 3A 6D 00 7A 03 38 21 0F 54 01 02 01 3A 6D 00 7A 03 38 21 0F 54 02 02 01 3A 6D 00 7A 03 38 21 0F 54 03 02 01 3A 6D 00 7A 03 38 21 0F 54 04 02 01 3A 6D 00 7A 03 39 21 0F 54 05 02 01 3A 6D 00 7B 01 42 21 10 54 01 02 01 3A 6D 00 7B 01 42 21 10 54 02 02 01 3A 6D 00 7B 01 43 21 10 54 03 02 01 3A 6D 00 7B 01 43 21 10 54 04 02 01 70 04

Tx: 11:51:38: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 0A 1C 00 00 41 93

Rx: 11:51:38: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 0A 26 3A 6D 00 7B 01 43 21 10 54 05 02 01 3A 6D 00 7B 03 3B 21 11 54 01 02 01 3A 6D 00 7B 03 3C 21 11 54 02 02 01 3A 6D 00 7B 03 3C 21 11 54 03 02 01 3A 6D 00 7B 03 3C 21 11 54 04 02 01 3A 6D 00 7B 03 3C 21 11 54 05 02 01 3A 6D 00 7C 01 A8 21 12 54 01 02 01 3A 6D 00 7C 01 A8 21 12 54 02 02 01 3A 6D 00 7C 01 A8 21 12 54 03 02 01 3A 6D 00 7C 01 A9 21 12 54 04 02 01 2D 8C

Tx: 11:51:38: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 0A 26 00 00 5C 63

Rx: 11:51:38: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 0A 30 3A 6D 00 7C 01 A9 21 12 54 05 02 01 3A 6D 00 7C 03 A4 21 13 54 01 02 01 3A 6D 00 7C 03 A4 21 13 54 02 02 01 3A 6D 00 7C 03 A4 21 13 54 03 02 01 3A 6D 00 7C 03 A4 21 13 54 04 02 01 3A 6D 00 7C 03 A5 21 13 54 05 02 01 3A 6D 00 7D 01 B0 21 14 54 01 02 01 3A 6D 00 7D 01 B0 21 14 54 02 02 01 3A 6D 00 7D 01 B0 21 14 54 03 02 01 3A 6D 00 7D 01 B0 21 14 54 04 02 01 D8 2F

Tx: 11:51:39: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 0A 30 00 00 96 F5

Rx: 11:51:39: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 0A 3A 3A 6D 00 7D 01 B1 21 14 54 05 02 01 3A 6D 00 7E 00 26 21 15 54 01 02 01 3A 6D 00 7E 00 26 21 15 54 02 02 01 3A 6D 00 7E 00 27 21 15 54 03 02 01 3A 6D 00 7E 00 27 21 15 54 04 02 01 3A 6D 00 7E 00 27 21 15 54 05 02 01 3A 6D 00 7E 02 10 21 16 54 01 02 01 3A 6D 00 7E 02 10 21 16 54 02 02 01 3A 6D 00 7E 02 11 21 16 54 03 02 01 3A 6D 00 7E 02 11 21 16 54 04 02 01 4C 5B

Tx: 11:51:39: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 0A 3A 00 00 EF 3D

Rx: 11:51:39: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 0A 44 3A 6D 00 7E 02 11 21 16 54 05 02 01 3A 6D 00 7F 00 1F 21 17 54 01 02 01 3A 6D 00 7F 00 1F 21 17 54 02 02 01 3A 6D 00 7F 00 20 21 17 54 03 02 01 3A 6D 00 7F 00 20 21 17 54 04 02 01 3A 6D 00 7F 00 20 21 17 54 05 02 01 3A 6D 00 7F 02 70 21 18 54 01 02 01 3A 6D 00 7F 02 70 21 18 54 02 02 01 3A 6D 00 7F 02 70 21 18 54 03 02 01 3A 6D 00 7F 02 71 21 18 54 04 02 01 FD 09

Tx: 11:51:39: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 0A 44 00 00 C8 05

Rx: 11:51:40: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 0A 4E 3A 6D 00 7F 02 71 21 18 54 05 02 01 3A 6D 00 80 00 86 21 19 54 01 02 01 3A 6D 00 80 00 86 21 19 54 02 02 01 3A 6D 00 80 00 86 21 19 54 03 02 01 3A 6D 00 80 00 87 21 19 54 04 02 01 3A 6D 00 80 00 87 21 19 54 05 02 01 3A 6D 00 80 02 7A 21 1A 54 01 02 01 3A 6D 00 80 02 7A 21 1A 54 02 02 01 3A 6D 00 80 02 7A 21 1A 54 03 02 01 3A 6D 00 80 02 7A 21 1A 54 04 02 01 AB 9E

Tx: 11:51:40: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 0A 4E 00 00 70 0B

Rx: 11:51:40: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 0A 58 3A 6D 00 80 02 7B 21 1A 54 05 02 01 3A 6D 00 81 00 8C 21 1B 54 01 02 01 3A 6D 00 81 00 8C 21 1B 54 02 02 01 3A 6D 00 81 00 8C 21 1B 54 03 02 01 3A 6D 00 81 00 8C 21 1B 54 04 02 01 3A 6D 00 81 00 8D 21 1B 54 05 02 01 3A 6D 00 81 02 81 21 1C 54 01 02 01 3A 6D 00 81 02 82 21 1C 54 02 02 01 3A 6D 00 81 02 82 21 1C 54 03 02 01 3A 6D 00 81 02 82 21 1C 54 04 02 01 A2 C7

Tx: 11:51:40: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 0A 58 00 00 BA 9D

Rx: 11:51:40: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 0A 62 3A 6D 00 81 02 83 21 1C 54 05 02 01 3A 6D 00 82 00 8B 21 1D 54 01 02 01 3A 6D 00 82 00 8B 21 1D 54 02 02 01 3A 6D 00 82 00 8B 21 1D 54 03 02 01 3A 6D 00 82 00 8C 21 1D 54 04 02 01 3A 6D 00 82 00 8C 21 1D 54 05 02 01 3A 6D 00 82 02 82 21 1E 54 01 02 01 3A 6D 00 82 02 83 21 1E 54 02 02 01 3A 6D 00 82 02 83 21 1E 54 03 02 01 3A 6D 00 82 02 83 21 1E 54 04 02 01 00 3C

Tx: 11:51:41: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 0A 62 00 00 F5 06

Rx: 11:51:41: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 0A 6C 3A 6D 00 82 02 83 21 1E 54 05 02 01 3A 6D 00 83 00 8E 21 1F 54 01 02 01 3A 6D 00 83 00 8E 21 1F 54 02 02 01 3A 6D 00 83 00 8F 21 1F 54 03 02 01 3A 6D 00 83 00 8F 21 1F 54 04 02 01 3A 6D 00 83 00 8F 21 1F 54 05 02 01 3A 6D 00 83 02 86 21 20 54 01 02 01 3A 6D 00 83 02 86 21 20 54 02 02 01 3A 6D 00 83 02 87 21 20 54 03 02 01 3A 6D 00 83 02 87 21 20 54 04 02 01 7B 0C

Tx: 11:51:41: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 0A 6C 00 00 AC A9

Rx: 11:51:41: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 0A 76 3A 6D 00 83 02 87 21 20 54 05 02 01 3A 6D 00 84 00 90 21 21 54 01 02 01 3A 6D 00 84 00 90 21 21 54 02 02 01 3A 6D 00 84 00 90 21 21 54 03 02 01 3A 6D 00 84 00 91 21 21 54 04 02 01 3A 6D 00 84 00 91 21 21 54 05 02 01 3A 6D 00 84 02 82 21 22 54 01 02 01 3A 6D 00 84 02 82 21 22 54 02 02 01 3A 6D 00 84 02 82 21 22 54 03 02 01 3A 6D 00 84 02 83 21 22 54 04 02 01 B1 E0

Tx: 11:51:41: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 0A 76 00 00 2D 5A

Rx: 11:51:42: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 0A 80 3A 6D 00 84 02 83 21 22 54 05 02 01 3A 6D 00 85 00 92 21 23 54 01 02 01 3A 6D 00 85 00 92 21 23 54 02 02 01 3A 6D 00 85 00 92 21 23 54 03 02 01 3A 6D 00 85 00 93 21 23 54 04 02 01 3A 6D 00 85 00 93 21 23 54 05 02 01 3A 6D 00 85 02 86 21 24 54 01 02 01 3A 6D 00 85 02 86 21 24 54 02 02 01 3A 6D 00 85 02 86 21 24 54 03 02 01 3A 6D 00 85 02 86 21 24 54 04 02 01 65 FE

Tx: 11:51:42: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 0A 80 00 00 7A 01

Rx: 11:51:42: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 0A 8A 3A 6D 00 85 02 87 21 24 54 05 02 01 3A 6D 00 86 00 95 21 25 54 01 02 01 3A 6D 00 86 00 95 21 25 54 02 02 01 3A 6D 00 86 00 96 21 25 54 03 02 01 3A 6D 00 86 00 96 21 25 54 04 02 01 3A 6D 00 86 00 96 21 25 54 05 02 01 3A 6D 00 86 02 E7 21 26 54 01 02 01 3A 6D 00 86 02 E8 21 26 54 02 02 01 3A 6D 00 86 02 E8 21 26 54 03 02 01 3A 6D 00 86 02 E8 21 26 54 04 02 01 DF 70

Tx: 11:51:42: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 0A 8A 00 00 63 2A

Rx: 11:51:42: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 0A 94 3A 6D 00 86 02 E8 21 26 54 05 02 01 3A 6D 00 87 00 FC 21 27 54 01 02 01 3A 6D 00 87 00 FC 21 27 54 02 02 01 3A 6D 00 87 00 FC 21 27 54 03 02 01 3A 6D 00 87 00 FD 21 27 54 04 02 01 3A 6D 00 87 00 FD 21 27 54 05 02 01 3A 6D 00 87 03 5A 21 28 54 01 02 01 3A 6D 00 87 03 5A 21 28 54 02 02 01 3A 6D 00 87 03 5B 21 28 54 03 02 01 3A 6D 00 87 03 5B 21 28 54 04 02 01 89 10

Tx: 11:51:43: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 0A 94 00 00 89 91

Rx: 11:51:43: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 0A 9E 3A 6D 00 87 03 5C 21 28 54 05 02 01 3A 6D 00 88 01 5E 21 29 54 01 02 01 3A 6D 00 88 01 5F 21 29 54 02 02 01 3A 6D 00 88 01 5F 21 29 54 03 02 01 3A 6D 00 88 01 5F 21 29 54 04 02 01 3A 6D 00 88 01 5F 21 29 54 05 02 01 3A 6D 00 88 03 5A 21 2A 54 01 02 01 3A 6D 00 88 03 5A 21 2A 54 02 02 01 3A 6D 00 88 03 5A 21 2A 54 03 02 01 3A 6D 00 88 03 5B 21 2A 54 04 02 01 18 6A

Tx: 11:51:43: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 0A 9E 00 00 90 BA

Rx: 11:51:43: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 0A A8 3A 6D 00 88 03 5C 21 2A 54 05 02 01 3A 6D 00 89 01 5E 21 2B 54 01 02 01 3A 6D 00 89 01 5E 21 2B 54 02 02 01 3A 6D 00 89 01 5F 21 2B 54 03 02 01 3A 6D 00 89 01 5F 21 2B 54 04 02 01 3A 6D 00 89 01 5F 21 2B 54 05 02 01 3A 6D 00 89 03 58 21 2C 54 01 02 01 3A 6D 00 89 03 58 21 2C 54 02 02 01 3A 6D 00 89 03 58 21 2C 54 03 02 01 3A 6D 00 89 03 58 21 2C 54 04 02 01 7C 49

Tx: 11:51:43: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 0A A8 00 00 1E AD

Rx: 11:51:44: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 00 08 3A 6D 00 89 03 59 21 2C 54 05 02 01 3A 6D 00 8A 01 5E 21 2D 54 01 02 01 3A 6D 00 8A 01 5E 21 2D 54 02 02 01 3A 6D 00 8A 01 5E 21 2D 54 03 02 01 3A 6D 00 8A 01 5F 21 2D 54 04 02 01 3A 6D 00 8A 01 5F 21 2D 54 05 02 01 3A 6D 00 8A 03 52 21 2E 54 01 02 01 3A 6D 00 8A 03 52 21 2E 54 02 02 01 3A 6D 00 8A 03 52 21 2E 54 03 02 01 3A 6D 00 8A 03 53 21 2E 54 04 02 01 51 8B

Tx: 11:51:44: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 00 08 00 00 FC ED

Rx: 11:51:44: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 00 12 3A 6D 00 8A 03 53 21 2E 54 05 02 01 3A 6D 00 8B 01 63 21 2F 54 01 02 01 3A 6D 00 8B 01 63 21 2F 54 02 02 01 3A 6D 00 8B 01 64 21 2F 54 03 02 01 3A 6D 00 8B 01 64 21 2F 54 04 02 01 3A 6D 00 8B 01 64 21 2F 54 05 02 01 3A 6D 00 8B 03 52 21 30 54 01 02 01 3A 6D 00 8B 03 52 21 30 54 02 02 01 3A 6D 00 8B 03 52 21 30 54 03 02 01 3A 6D 00 8B 03 53 21 30 54 04 02 01 E5 91

Tx: 11:51:44: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 00 12 00 00 C5 7F

Rx: 11:51:44: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 00 1C 3A 6D 00 8B 03 53 21 30 54 05 02 01 3A 6D 00 8C 01 5F 21 31 54 01 02 01 3A 6D 00 8C 01 5F 21 31 54 02 02 01 3A 6D 00 8C 01 5F 21 31 54 03 02 01 3A 6D 00 8C 01 5F 21 31 54 04 02 01 3A 6D 00 8C 01 60 21 31 54 05 02 01 3A 6D 00 8C 03 56 21 32 54 01 02 01 3A 6D 00 8C 03 56 21 32 54 02 02 01 3A 6D 00 8C 03 56 21 32 54 03 02 01 3A 6D 00 8C 03 57 21 32 54 04 02 01 E5 98

Tx: 11:51:45: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 00 1C 00 00 9C D0

Rx: 11:51:45: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 00 26 3A 6D 00 8C 03 57 21 32 54 05 02 01 3A 6D 00 8D 01 62 21 33 54 01 02 01 3A 6D 00 8D 01 62 21 33 54 02 02 01 3A 6D 00 8D 01 62 21 33 54 03 02 01 3A 6D 00 8D 01 63 21 33 54 04 02 01 3A 6D 00 8D 01 63 21 33 54 05 02 01 3A 6D 00 8D 03 56 21 34 54 01 02 01 3A 6D 00 8D 03 56 21 34 54 02 02 01 3A 6D 00 8D 03 56 21 34 54 03 02 01 3A 6D 00 8D 03 57 21 34 54 04 02 01 99 C4

Tx: 11:51:45: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 00 26 00 00 D3 4B

Rx: 11:51:45: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 00 30 3A 6D 00 8D 03 57 21 34 54 05 02 01 3A 6D 00 8E 01 60 21 35 54 01 02 01 3A 6D 00 8E 01 60 21 35 54 02 02 01 3A 6D 00 8E 01 60 21 35 54 03 02 01 3A 6D 00 8E 01 60 21 35 54 04 02 01 3A 6D 00 8E 01 61 21 35 54 05 02 01 3A 6D 00 8E 03 58 21 36 54 01 02 01 3A 6D 00 8E 03 58 21 36 54 02 02 01 3A 6D 00 8E 03 58 21 36 54 03 02 01 3A 6D 00 8E 03 58 21 36 54 04 02 01 FD 2A

Tx: 11:51:45: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 00 30 00 00 19 DD

Rx: 11:51:46: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 00 3A 3A 6D 00 8E 03 59 21 36 54 05 02 01 3A 6D 00 8F 01 C0 21 37 54 01 02 01 3A 6D 00 8F 01 C0 21 37 54 02 02 01 3A 6D 00 8F 01 C0 21 37 54 03 02 01 3A 6D 00 8F 01 C1 21 37 54 04 02 01 3A 6D 00 8F 01 C1 21 37 54 05 02 01 3A 6D 00 8F 03 B4 21 38 54 01 02 01 3A 6D 00 8F 03 B4 21 38 54 02 02 01 3A 6D 00 8F 03 B4 21 38 54 03 02 01 3A 6D 00 8F 03 B4 21 38 54 04 02 01 FC 84

Tx: 11:51:46: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 00 3A 00 00 A1 D3

Rx: 11:51:46: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 00 44 3A 6D 00 8F 03 B5 21 38 54 05 02 01 3A 6D 00 90 01 C1 21 39 54 01 02 01 3A 6D 00 90 01 C1 21 39 54 02 02 01 3A 6D 00 90 01 C1 21 39 54 03 02 01 3A 6D 00 90 01 C1 21 39 54 04 02 01 3A 6D 00 90 01 C2 21 39 54 05 02 01 3A 6D 00 90 03 BC 21 3A 54 01 02 01 3A 6D 00 90 03 BD 21 3A 54 02 02 01 3A 6D 00 90 03 BD 21 3A 54 03 02 01 3A 6D 00 90 03 BD 21 3A 54 04 02 01 AD BC

Tx: 11:51:46: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 00 44 00 00 86 EB

Rx: 11:51:46: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 00 4E 3A 6D 00 90 03 BD 21 3A 54 05 02 01 3A 6D 00 91 01 CA 21 3B 54 01 02 01 3A 6D 00 91 01 CB 21 3B 54 02 02 01 3A 6D 00 91 01 CB 21 3B 54 03 02 01 3A 6D 00 91 01 CB 21 3B 54 04 02 01 3A 6D 00 91 01 CB 21 3B 54 05 02 01 3A 6D 00 91 03 BE 21 3C 54 01 02 01 3A 6D 00 91 03 BE 21 3C 54 02 02 01 3A 6D 00 91 03 BF 21 3C 54 03 02 01 3A 6D 00 91 03 BF 21 3C 54 04 02 01 35 18

Tx: 11:51:47: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 00 4E 00 00 FF 23

Rx: 11:51:47: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 00 58 3A 6D 00 91 03 BF 21 3C 54 05 02 01 3A 6D 00 92 01 C4 21 3D 54 01 02 01 3A 6D 00 92 01 C4 21 3D 54 02 02 01 3A 6D 00 92 01 C4 21 3D 54 03 02 01 3A 6D 00 92 01 C4 21 3D 54 04 02 01 3A 6D 00 92 01 C5 21 3D 54 05 02 01 3A 6D 00 93 00 35 21 3E 54 01 02 01 3A 6D 00 93 00 36 21 3E 54 02 02 01 3A 6D 00 93 00 36 21 3E 54 03 02 01 3A 6D 00 93 00 36 21 3E 54 04 02 01 D3 33

Tx: 11:51:47: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 00 58 00 00 35 B5

Rx: 11:51:47: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 00 62 3A 6D 00 93 00 37 21 3E 54 05 02 01 3A 6D 00 93 02 32 21 3F 54 01 02 01 3A 6D 00 93 02 32 21 3F 54 02 02 01 3A 6D 00 93 02 32 21 3F 54 03 02 01 3A 6D 00 93 02 33 21 3F 54 04 02 01 3A 6D 00 93 02 33 21 3F 54 05 02 01 3A 6D 00 94 00 9A 21 40 54 01 02 01 3A 6D 00 94 00 9A 21 40 54 02 02 01 3A 6D 00 94 00 9A 21 40 54 03 02 01 3A 6D 00 94 00 9A 21 40 54 04 02 01 48 C7

Tx: 11:51:47: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 00 62 00 00 90 24

Rx: 11:51:48: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 00 6C 3A 6D 00 94 00 9B 21 40 54 05 02 01 3A 6D 00 94 02 93 21 41 54 01 02 01 3A 6D 00 94 02 94 21 41 54 02 02 01 3A 6D 00 94 02 94 21 41 54 03 02 01 3A 6D 00 94 02 94 21 41 54 04 02 01 3A 6D 00 94 02 94 21 41 54 05 02 01 3A 6D 00 95 00 98 21 42 54 01 02 01 3A 6D 00 95 00 98 21 42 54 02 02 01 3A 6D 00 95 00 98 21 42 54 03 02 01 3A 6D 00 95 00 99 21 42 54 04 02 01 20 9C

Tx: 11:51:48: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 00 6C 00 00 A9 68

Rx: 11:51:48: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 00 76 3A 6D 00 95 00 99 21 42 54 05 02 01 3A 6D 00 95 02 8F 21 43 54 01 02 01 3A 6D 00 95 02 8F 21 43 54 02 02 01 3A 6D 00 95 02 90 21 43 54 03 02 01 3A 6D 00 95 02 90 21 43 54 04 02 01 3A 6D 00 95 02 90 21 43 54 05 02 01 3A 6D 00 96 00 98 21 44 54 01 02 01 3A 6D 00 96 00 98 21 44 54 02 02 01 3A 6D 00 96 00 98 21 44 54 03 02 01 3A 6D 00 96 00 99 21 44 54 04 02 01 9A 3D

Tx: 11:51:48: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 00 76 00 00 A2 72

Rx: 11:51:48: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 00 80 3A 6D 00 96 00 99 21 44 54 05 02 01 3A 6D 00 96 02 8C 21 45 54 01 02 01 3A 6D 00 96 02 8C 21 45 54 02 02 01 3A 6D 00 96 02 8C 21 45 54 03 02 01 3A 6D 00 96 02 8C 21 45 54 04 02 01 3A 6D 00 96 02 8D 21 45 54 05 02 01 3A 6D 00 97 00 A4 21 46 54 01 02 01 3A 6D 00 97 00 A4 21 46 54 02 02 01 3A 6D 00 97 00 A5 21 46 54 03 02 01 3A 6D 00 97 00 A5 21 46 54 04 02 01 DD C1

Tx: 11:51:49: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 00 80 00 00 34 EF

Rx: 11:51:49: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 00 8A 3A 6D 00 97 00 A5 21 46 54 05 02 01 3A 6D 00 97 02 98 21 47 54 01 02 01 3A 6D 00 97 02 98 21 47 54 02 02 01 3A 6D 00 97 02 98 21 47 54 03 02 01 3A 6D 00 97 02 99 21 47 54 04 02 01 3A 6D 00 97 02 99 21 47 54 05 02 01 3A 6D 00 98 00 9E 21 48 54 01 02 01 3A 6D 00 98 00 9E 21 48 54 02 02 01 3A 6D 00 98 00 9F 21 48 54 03 02 01 3A 6D 00 98 00 9F 21 48 54 04 02 01 98 6F

Tx: 11:51:49: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 00 8A 00 00 2D C4

Rx: 11:51:49: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 00 94 3A 6D 00 98 00 9F 21 48 54 05 02 01 3A 6D 00 98 02 98 21 49 54 01 02 01 3A 6D 00 98 02 98 21 49 54 02 02 01 3A 6D 00 98 02 98 21 49 54 03 02 01 3A 6D 00 98 02 98 21 49 54 04 02 01 3A 6D 00 98 02 99 21 49 54 05 02 01 3A 6D 00 99 00 AA 21 4A 54 01 02 01 3A 6D 00 99 00 AA 21 4A 54 02 02 01 3A 6D 00 99 00 AA 21 4A 54 03 02 01 3A 6D 00 99 00 AA 21 4A 54 04 02 01 B3 88

Tx: 11:51:49: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 00 94 00 00 06 B9

Rx: 11:51:50: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 00 9E 3A 6D 00 99 00 AB 21 4A 54 05 02 01 3A 6D 00 99 02 9B 21 4B 54 01 02 01 3A 6D 00 99 02 9C 21 4B 54 02 02 01 3A 6D 00 99 02 9C 21 4B 54 03 02 01 3A 6D 00 99 02 9C 21 4B 54 04 02 01 3A 6D 00 99 02 9C 21 4B 54 05 02 01 3A 6D 00 9A 00 A7 21 4C 54 01 02 01 3A 6D 00 9A 00 A8 21 4C 54 02 02 01 3A 6D 00 9A 00 A8 21 4C 54 03 02 01 3A 6D 00 9A 00 A8 21 4C 54 04 02 01 AD 2D

Tx: 11:51:50: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 00 9E 00 00 1F 92

Rx: 11:51:50: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 00 A8 3A 6D 00 9A 00 A8 21 4C 54 05 02 01 3A 6D 00 9A 02 A2 21 4D 54 01 02 01 3A 6D 00 9A 02 A2 21 4D 54 02 02 01 3A 6D 00 9A 02 A3 21 4D 54 03 02 01 3A 6D 00 9A 02 A3 21 4D 54 04 02 01 3A 6D 00 9A 02 A3 21 4D 54 05 02 01 3A 6D 00 9B 00 AD 21 4E 54 01 02 01 3A 6D 00 9B 00 AD 21 4E 54 02 02 01 3A 6D 00 9B 00 AD 21 4E 54 03 02 01 3A 6D 00 9B 00 AE 21 4E 54 04 02 01 C2 CF

Tx: 11:51:50: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 00 A8 00 00 C3 EE

Rx: 11:51:50: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 00 B2 3A 6D 00 9B 00 AE 21 4E 54 05 02 01 3A 6D 00 9B 03 02 21 4F 54 01 02 01 3A 6D 00 9B 03 03 21 4F 54 02 02 01 3A 6D 00 9B 03 03 21 4F 54 03 02 01 3A 6D 00 9B 03 03 21 4F 54 04 02 01 3A 6D 00 9B 03 03 21 4F 54 05 02 01 3A 6D 00 9C 01 07 21 50 54 01 02 01 3A 6D 00 9C 01 07 21 50 54 02 02 01 3A 6D 00 9C 01 07 21 50 54 03 02 01 3A 6D 00 9C 01 08 21 50 54 04 02 01 53 4B

Tx: 11:51:51: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 00 B2 00 00 C8 F4

Rx: 11:51:51: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 00 BC 3A 6D 00 9C 01 08 21 50 54 05 02 01 3A 6D 00 9C 02 FB 21 51 54 01 02 01 3A 6D 00 9C 02 FB 21 51 54 02 02 01 3A 6D 00 9C 02 FB 21 51 54 03 02 01 3A 6D 00 9C 02 FB 21 51 54 04 02 01 3A 6D 00 9C 02 FC 21 51 54 05 02 01 3A 6D 00 9D 01 08 21 52 54 01 02 01 3A 6D 00 9D 01 08 21 52 54 02 02 01 3A 6D 00 9D 01 08 21 52 54 03 02 01 3A 6D 00 9D 01 08 21 52 54 04 02 01 A8 C9

Tx: 11:51:51: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 00 BC 00 00 F1 B8

Rx: 11:51:51: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 00 C6 3A 6D 00 9D 01 09 21 52 54 05 02 01 3A 6D 00 9D 02 FF 21 53 54 01 02 01 3A 6D 00 9D 02 FF 21 53 54 02 02 01 3A 6D 00 9D 02 FF 21 53 54 03 02 01 3A 6D 00 9D 03 00 21 53 54 04 02 01 3A 6D 00 9D 03 00 21 53 54 05 02 01 3A 6D 00 9E 01 07 21 54 54 01 02 01 3A 6D 00 9E 01 08 21 54 54 02 02 01 3A 6D 00 9E 01 08 21 54 54 03 02 01 3A 6D 00 9E 01 08 21 54 54 04 02 01 66 A7

Tx: 11:51:51: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 00 C6 00 00 96 04

Rx: 11:51:52: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 00 D0 3A 6D 00 9E 01 08 21 54 54 05 02 01 3A 6D 00 9E 03 63 21 55 54 01 02 01 3A 6D 00 9E 03 63 21 55 54 02 02 01 3A 6D 00 9E 03 64 21 55 54 03 02 01 3A 6D 00 9E 03 64 21 55 54 04 02 01 3A 6D 00 9E 03 64 21 55 54 05 02 01 3A 6D 00 9F 01 6C 21 56 54 01 02 01 3A 6D 00 9F 01 6C 21 56 54 02 02 01 3A 6D 00 9F 01 6C 21 56 54 03 02 01 3A 6D 00 9F 01 6C 21 56 54 04 02 01 71 0A

Tx: 11:51:52: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 00 D0 00 00 FD B7

Rx: 11:51:52: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 00 DA 3A 6D 00 9F 01 6D 21 56 54 05 02 01 3A 6D 00 9F 03 CB 21 57 54 01 02 01 3A 6D 00 9F 03 CC 21 57 54 02 02 01 3A 6D 00 9F 03 CC 21 57 54 03 02 01 3A 6D 00 9F 03 CC 21 57 54 04 02 01 3A 6D 00 9F 03 CC 21 57 54 05 02 01 3A 6D 00 A0 02 42 21 58 54 01 02 01 3A 6D 00 A0 02 42 21 58 54 02 02 01 3A 6D 00 A0 02 42 21 58 54 03 02 01 3A 6D 00 A0 02 42 21 58 54 04 02 01 9B ED

Tx: 11:51:52: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 00 DA 00 00 E4 9C

Rx: 11:51:52: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 00 E4 3A 6D 00 A0 02 43 21 58 54 05 02 01 3A 6D 00 A1 00 A4 21 59 54 01 02 01 3A 6D 00 A1 00 A5 21 59 54 02 02 01 3A 6D 00 A1 00 A5 21 59 54 03 02 01 3A 6D 00 A1 00 A5 21 59 54 04 02 01 3A 6D 00 A1 00 A5 21 59 54 05 02 01 3A 6D 00 A1 02 98 21 5A 54 01 02 01 3A 6D 00 A1 02 98 21 5A 54 02 02 01 3A 6D 00 A1 02 99 21 5A 54 03 02 01 3A 6D 00 A1 02 99 21 5A 54 04 02 01 65 CF

Tx: 11:51:53: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 00 E4 00 00 EB 83

Rx: 11:51:53: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 00 EE 3A 6D 00 A1 02 99 21 5A 54 05 02 01 3A 6D 00 A2 01 09 21 5B 54 01 02 01 3A 6D 00 A2 01 0A 21 5B 54 02 02 01 3A 6D 00 A2 01 0A 21 5B 54 03 02 01 3A 6D 00 A2 01 0A 21 5B 54 04 02 01 3A 6D 00 A2 01 0A 21 5B 54 05 02 01 3A 6D 00 A2 03 5D 21 5C 54 01 02 01 3A 6D 00 A2 03 5D 21 5C 54 02 02 01 3A 6D 00 A2 03 5E 21 5C 54 03 02 01 3A 6D 00 A2 03 5E 21 5C 54 04 02 01 18 48

Tx: 11:51:53: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 00 EE 00 00 F2 A8

Rx: 11:51:53: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 00 F8 3A 6D 00 A2 03 5E 21 5C 54 05 02 01 3A 6D 00 A3 01 6D 21 5D 54 01 02 01 3A 6D 00 A3 01 6E 21 5D 54 02 02 01 3A 6D 00 A3 01 6E 21 5D 54 03 02 01 3A 6D 00 A3 01 6E 21 5D 54 04 02 01 3A 6D 00 A3 01 6E 21 5D 54 05 02 01 3A 6D 00 A3 03 CF 21 5E 54 01 02 01 3A 6D 00 A3 03 D0 21 5E 54 02 02 01 3A 6D 00 A3 03 D0 21 5E 54 03 02 01 3A 6D 00 A3 03 D0 21 5E 54 04 02 01 BF F0

Tx: 11:51:53: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 00 F8 00 00 B2 D7

Rx: 11:51:54: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 01 02 3A 6D 00 A3 03 D0 21 5E 54 05 02 01 3A 6D 00 A4 01 DA 21 5F 54 01 02 01 3A 6D 00 A4 01 DB 21 5F 54 02 02 01 3A 6D 00 A4 01 DB 21 5F 54 03 02 01 3A 6D 00 A4 01 DB 21 5F 54 04 02 01 3A 6D 00 A4 01 DB 21 5F 54 05 02 01 3A 6D 00 A5 00 48 21 60 54 01 02 01 3A 6D 00 A5 00 48 21 60 54 02 02 01 3A 6D 00 A5 00 49 21 60 54 03 02 01 3A 6D 00 A5 00 49 21 60 54 04 02 01 09 55

Tx: 11:51:54: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 01 02 00 00 17 31

Rx: 11:51:54: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 01 0C 3A 6D 00 A5 00 49 21 60 54 05 02 01 3A 6D 00 A5 02 A1 21 61 54 01 02 01 3A 6D 00 A5 02 A1 21 61 54 02 02 01 3A 6D 00 A5 02 A1 21 61 54 03 02 01 3A 6D 00 A5 02 A2 21 61 54 04 02 01 3A 6D 00 A5 02 A3 21 61 54 05 02 01 3A 6D 00 A6 00 AD 21 62 54 01 02 01 3A 6D 00 A6 00 AD 21 62 54 02 02 01 3A 6D 00 A6 00 AD 21 62 54 03 02 01 3A 6D 00 A6 00 AE 21 62 54 04 02 01 89 B8

Tx: 11:51:54: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 01 0C 00 00 4E 9E

Rx: 11:51:54: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 01 16 3A 6D 00 A6 00 AE 21 62 54 05 02 01 3A 6D 00 A6 02 A3 21 63 54 01 02 01 3A 6D 00 A6 02 A3 21 63 54 02 02 01 3A 6D 00 A6 02 A4 21 63 54 03 02 01 3A 6D 00 A6 02 A4 21 63 54 04 02 01 3A 6D 00 A6 02 A4 21 63 54 05 02 01 3A 6D 00 A7 00 B8 21 64 54 01 02 01 3A 6D 00 A7 00 B8 21 64 54 02 02 01 3A 6D 00 A7 00 B8 21 64 54 03 02 01 3A 6D 00 A7 00 B8 21 64 54 04 02 01 3A 21

Tx: 11:51:55: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 01 16 00 00 E4 A1

Rx: 11:51:55: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 01 20 3A 6D 00 A7 00 B9 21 64 54 05 02 01 3A 6D 00 A7 02 A7 21 65 54 01 02 01 3A 6D 00 A7 02 A8 21 65 54 02 02 01 3A 6D 00 A7 02 A8 21 65 54 03 02 01 3A 6D 00 A7 02 A8 21 65 54 04 02 01 3A 6D 00 A7 02 A9 21 65 54 05 02 01 3A 6D 00 A8 01 1C 21 66 54 01 02 01 3A 6D 00 A8 01 1C 21 66 54 02 02 01 3A 6D 00 A8 01 1C 21 66 54 03 02 01 3A 6D 00 A8 01 1D 21 66 54 04 02 01 45 77

Tx: 11:51:55: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 01 20 00 00 0A 55

Rx: 11:51:55: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 01 2A 3A 6D 00 A8 01 1D 21 66 54 05 02 01 3A 6D 00 A8 03 6A 21 67 54 01 02 01 3A 6D 00 A8 03 6A 21 67 54 02 02 01 3A 6D 00 A8 03 6B 21 67 54 03 02 01 3A 6D 00 A8 03 6B 21 67 54 04 02 01 3A 6D 00 A8 03 6C 21 67 54 05 02 01 3A 6D 00 A9 01 78 21 68 54 01 02 01 3A 6D 00 A9 01 78 21 68 54 02 02 01 3A 6D 00 A9 01 79 21 68 54 03 02 01 3A 6D 00 A9 01 79 21 68 54 04 02 01 27 D8

Tx: 11:51:55: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 01 2A 00 00 73 9D

Rx: 11:51:56: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 01 34 3A 6D 00 A9 01 79 21 68 54 05 02 01 3A 6D 00 A9 03 CE 21 69 54 01 02 01 3A 6D 00 A9 03 CE 21 69 54 02 02 01 3A 6D 00 A9 03 CF 21 69 54 03 02 01 3A 6D 00 A9 03 D0 21 69 54 04 02 01 3A 6D 00 A9 03 D0 21 69 54 05 02 01 3A 6D 00 AA 01 DA 21 6A 54 01 02 01 3A 6D 00 AA 01 DA 21 6A 54 02 02 01 3A 6D 00 AA 01 DB 21 6A 54 03 02 01 3A 6D 00 AA 01 DB 21 6A 54 04 02 01 0C 92

Tx: 11:51:56: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 01 34 00 00 38 03

Rx: 11:51:56: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 01 3E 3A 6D 00 AA 01 DC 21 6A 54 05 02 01 3A 6D 00 AA 03 D8 21 6B 54 01 02 01 3A 6D 00 AA 03 D8 21 6B 54 02 02 01 3A 6D 00 AA 03 D8 21 6B 54 03 02 01 3A 6D 00 AA 03 D9 21 6B 54 04 02 01 3A 6D 00 AA 03 DA 21 6B 54 05 02 01 3A 6D 00 AB 01 DE 21 6C 54 01 02 01 3A 6D 00 AB 01 DE 21 6C 54 02 02 01 3A 6D 00 AB 01 DE 21 6C 54 03 02 01 3A 6D 00 AB 01 DF 21 6C 54 04 02 01 D6 D2

Tx: 11:51:56: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 01 3E 00 00 13 A0

Rx: 11:51:56: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 01 48 3A 6D 00 AB 01 DF 21 6C 54 05 02 01 3A 6D 00 AB 03 DC 21 6D 54 01 02 01 3A 6D 00 AB 03 DC 21 6D 54 02 02 01 3A 6D 00 AB 03 DC 21 6D 54 03 02 01 3A 6D 00 AB 03 DD 21 6D 54 04 02 01 3A 6D 00 AB 03 DD 21 6D 54 05 02 01 3A 6D 00 AC 02 46 21 6E 54 01 02 01 3A 6D 00 AC 02 46 21 6E 54 02 02 01 3A 6D 00 AC 02 47 21 6E 54 03 02 01 3A 6D 00 AC 02 47 21 6E 54 04 02 01 B5 DF

Tx: 11:51:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 01 48 00 00 B5 90

Rx: 11:51:57: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 01 52 3A 6D 00 AC 02 47 21 6E 54 05 02 01 3A 6D 00 AD 00 58 21 6F 54 01 02 01 3A 6D 00 AD 00 59 21 6F 54 02 02 01 3A 6D 00 AD 00 59 21 6F 54 03 02 01 3A 6D 00 AD 00 59 21 6F 54 04 02 01 3A 6D 00 AD 00 5A 21 6F 54 05 02 01 3A 6D 00 AD 02 45 21 70 54 01 02 01 3A 6D 00 AD 02 46 21 70 54 02 02 01 3A 6D 00 AD 02 46 21 70 54 03 02 01 3A 6D 00 AD 02 46 21 70 54 04 02 01 33 5F

Tx: 11:51:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 01 52 00 00 DE 69

Rx: 11:51:57: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 01 5C 3A 6D 00 AD 02 46 21 70 54 05 02 01 3A 6D 00 AE 00 5A 21 71 54 01 02 01 3A 6D 00 AE 00 5A 21 71 54 02 02 01 3A 6D 00 AE 00 5B 21 71 54 03 02 01 3A 6D 00 AE 00 5C 21 71 54 04 02 01 3A 6D 00 AE 00 5C 21 71 54 05 02 01 3A 6D 00 AE 02 4A 21 72 54 01 02 01 3A 6D 00 AE 02 4A 21 72 54 02 02 01 3A 6D 00 AE 02 4A 21 72 54 03 02 01 3A 6D 00 AE 02 4A 21 72 54 04 02 01 9E 1E

Tx: 11:51:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 01 5C 00 00 87 C6

Rx: 11:51:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 01 66 3A 6D 00 AE 02 4B 21 72 54 05 02 01 3A 6D 00 AF 00 54 21 73 54 01 02 01 3A 6D 00 AF 00 54 21 73 54 02 02 01 3A 6D 00 AF 00 54 21 73 54 03 02 01 3A 6D 00 AF 00 54 21 73 54 04 02 01 3A 6D 00 AF 00 55 21 73 54 05 02 01 3A 6D 00 AF 02 46 21 74 54 01 02 01 3A 6D 00 AF 02 46 21 74 54 02 02 01 3A 6D 00 AF 02 46 21 74 54 03 02 01 3A 6D 00 AF 02 47 21 74 54 04 02 01 95 C9

Tx: 11:51:58: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 01 66 00 00 09 9B

Rx: 11:51:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 01 70 3A 6D 00 AF 02 47 21 74 54 05 02 01 3A 6D 00 B0 00 B8 21 75 54 01 02 01 3A 6D 00 B0 00 B8 21 75 54 02 02 01 3A 6D 00 B0 00 B8 21 75 54 03 02 01 3A 6D 00 B0 00 B8 21 75 54 04 02 01 3A 6D 00 B0 00 B9 21 75 54 05 02 01 3A 6D 00 B0 02 AE 21 76 54 01 02 01 3A 6D 00 B0 02 AE 21 76 54 02 02 01 3A 6D 00 B0 02 AE 21 76 54 03 02 01 3A 6D 00 B0 02 AE 21 76 54 04 02 01 27 F0

Tx: 11:51:58: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 01 70 00 00 C3 0D

Rx: 11:51:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 01 7A 3A 6D 00 B0 02 AF 21 76 54 05 02 01 3A 6D 00 B1 01 20 21 77 54 01 02 01 3A 6D 00 B1 01 20 21 77 54 02 02 01 3A 6D 00 B1 01 20 21 77 54 03 02 01 3A 6D 00 B1 01 21 21 77 54 04 02 01 3A 6D 00 B1 01 21 21 77 54 05 02 01 3A 6D 00 B1 03 18 21 78 54 01 02 01 3A 6D 00 B1 03 18 21 78 54 02 02 01 3A 6D 00 B1 03 18 21 78 54 03 02 01 3A 6D 00 B1 03 19 21 78 54 04 02 01 FD AC

Tx: 11:51:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 01 7A 00 00 BA C5

Rx: 11:51:59: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 01 84 3A 6D 00 B1 03 19 21 78 54 05 02 01 3A 6D 00 B2 01 27 21 79 54 01 02 01 3A 6D 00 B2 01 27 21 79 54 02 02 01 3A 6D 00 B2 01 27 21 79 54 03 02 01 3A 6D 00 B2 01 27 21 79 54 04 02 01 3A 6D 00 B2 01 28 21 79 54 05 02 01 3A 6D 00 B2 03 19 21 7A 54 01 02 01 3A 6D 00 B2 03 19 21 7A 54 02 02 01 3A 6D 00 B2 03 19 21 7A 54 03 02 01 3A 6D 00 B2 03 19 21 7A 54 04 02 01 77 19

Tx: 11:51:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 01 84 00 00 0C 75

Rx: 11:51:59: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 01 8E 3A 6D 00 B2 03 1A 21 7A 54 05 02 01 3A 6D 00 B3 01 21 21 7B 54 01 02 01 3A 6D 00 B3 01 21 21 7B 54 02 02 01 3A 6D 00 B3 01 21 21 7B 54 03 02 01 3A 6D 00 B3 01 21 21 7B 54 04 02 01 3A 6D 00 B3 01 22 21 7B 54 05 02 01 3A 6D 00 B3 03 7B 21 7C 54 01 02 01 3A 6D 00 B3 03 7B 21 7C 54 02 02 01 3A 6D 00 B3 03 7B 21 7C 54 03 02 01 3A 6D 00 B3 03 7B 21 7C 54 04 02 01 EA C6

Tx: 11:51:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 01 8E 00 00 9F B7

Rx: 11:52:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 01 98 3A 6D 00 B3 03 7C 21 7C 54 05 02 01 3A 6D 00 B4 01 87 21 7D 54 01 02 01 3A 6D 00 B4 01 87 21 7D 54 02 02 01 3A 6D 00 B4 01 87 21 7D 54 03 02 01 3A 6D 00 B4 01 87 21 7D 54 04 02 01 3A 6D 00 B4 01 88 21 7D 54 05 02 01 3A 6D 00 B4 03 7C 21 7E 54 01 02 01 3A 6D 00 B4 03 7D 21 7E 54 02 02 01 3A 6D 00 B4 03 7D 21 7E 54 03 02 01 3A 6D 00 B4 03 7D 21 7E 54 04 02 01 16 23

Tx: 11:52:00: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 01 98 00 00 35 C2

Rx: 11:52:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 01 A2 3A 6D 00 B4 03 7D 21 7E 54 05 02 01 3A 6D 00 B5 01 8F 21 7F 54 01 02 01 3A 6D 00 B5 01 8F 21 7F 54 02 02 01 3A 6D 00 B5 01 8F 21 7F 54 03 02 01 3A 6D 00 B5 01 8F 21 7F 54 04 02 01 3A 6D 00 B5 01 90 21 7F 54 05 02 01 3A 6D 00 B5 03 DB 21 80 54 01 02 01 3A 6D 00 B5 03 DC 21 80 54 02 02 01 3A 6D 00 B5 03 DC 21 80 54 03 02 01 3A 6D 00 B5 03 DC 21 80 54 04 02 01 99 07

Tx: 11:52:00: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 01 A2 00 00 1A BA

Rx: 11:52:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 01 AC 3A 6D 00 B5 03 DC 21 80 54 05 02 01 3A 6D 00 B6 01 ED 21 81 54 01 02 01 3A 6D 00 B6 01 ED 21 81 54 02 02 01 3A 6D 00 B6 01 ED 21 81 54 03 02 01 3A 6D 00 B6 01 ED 21 81 54 04 02 01 3A 6D 00 B6 01 EE 21 81 54 05 02 01 3A 6D 00 B7 00 5C 21 82 54 01 02 01 3A 6D 00 B7 00 5D 21 82 54 02 02 01 3A 6D 00 B7 00 5D 21 82 54 03 02 01 3A 6D 00 B7 00 5D 21 82 54 04 02 01 12 6D

Tx: 11:52:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 01 AC 00 00 E2 30

Rx: 11:52:01: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 01 B6 3A 6D 00 B7 00 5D 21 82 54 05 02 01 3A 6D 00 B7 02 53 21 83 54 01 02 01 3A 6D 00 B7 02 53 21 83 54 02 02 01 3A 6D 00 B7 02 53 21 83 54 03 02 01 3A 6D 00 B7 02 53 21 83 54 04 02 01 3A 6D 00 B7 02 54 21 83 54 05 02 01 3A 6D 00 B8 00 61 21 84 54 01 02 01 3A 6D 00 B8 00 61 21 84 54 02 02 01 3A 6D 00 B8 00 62 21 84 54 03 02 01 3A 6D 00 B8 00 62 21 84 54 04 02 01 1F 78

Tx: 11:52:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 01 B6 00 00 E9 2A

Rx: 11:52:01: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 01 C0 3A 6D 00 B8 00 63 21 84 54 05 02 01 3A 6D 00 B8 02 55 21 85 54 01 02 01 3A 6D 00 B8 02 55 21 85 54 02 02 01 3A 6D 00 B8 02 55 21 85 54 03 02 01 3A 6D 00 B8 02 55 21 85 54 04 02 01 3A 6D 00 B8 02 56 21 85 54 05 02 01 3A 6D 00 B9 00 BB 21 86 54 01 02 01 3A 6D 00 B9 00 BB 21 86 54 02 02 01 3A 6D 00 B9 00 BB 21 86 54 03 02 01 3A 6D 00 B9 00 BC 21 86 54 04 02 01 7F 91

Tx: 11:52:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 01 C0 00 00 2F F9

Rx: 11:52:02: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 01 CA 3A 6D 00 B9 00 BC 21 86 54 05 02 01 3A 6D 00 B9 02 B7 21 87 54 01 02 01 3A 6D 00 B9 02 B7 21 87 54 02 02 01 3A 6D 00 B9 02 B7 21 87 54 03 02 01 3A 6D 00 B9 02 B7 21 87 54 04 02 01 3A 6D 00 B9 02 B8 21 87 54 05 02 01 3A 6D 00 BA 00 BD 21 88 54 01 02 01 3A 6D 00 BA 00 BD 21 88 54 02 02 01 3A 6D 00 BA 00 BE 21 88 54 03 02 01 3A 6D 00 BA 00 BE 21 88 54 04 02 01 98 7C

Tx: 11:52:02: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 01 CA 00 00 36 D2

Rx: 11:52:02: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 01 D4 3A 6D 00 BA 00 BE 21 88 54 05 02 01 3A 6D 00 BA 02 B6 21 89 54 01 02 01 3A 6D 00 BA 02 B7 21 89 54 02 02 01 3A 6D 00 BA 02 B7 21 89 54 03 02 01 3A 6D 00 BA 02 B7 21 89 54 04 02 01 3A 6D 00 BA 02 B7 21 89 54 05 02 01 3A 6D 00 BB 00 C2 21 8A 54 01 02 01 3A 6D 00 BB 00 C3 21 8A 54 02 02 01 3A 6D 00 BB 00 C3 21 8A 54 03 02 01 3A 6D 00 BB 00 C3 21 8A 54 04 02 01 E5 AB

Tx: 11:52:02: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 01 D4 00 00 4F C4

Rx: 11:52:02: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 01 DE 3A 6D 00 BB 00 C3 21 8A 54 05 02 01 3A 6D 00 BB 03 1D 21 8B 54 01 02 01 3A 6D 00 BB 03 1D 21 8B 54 02 02 01 3A 6D 00 BB 03 1D 21 8B 54 03 02 01 3A 6D 00 BB 03 1D 21 8B 54 04 02 01 3A 6D 00 BB 03 1E 21 8B 54 05 02 01 3A 6D 00 BC 01 28 21 8C 54 01 02 01 3A 6D 00 BC 01 29 21 8C 54 02 02 01 3A 6D 00 BC 01 29 21 8C 54 03 02 01 3A 6D 00 BC 01 29 21 8C 54 04 02 01 83 68

Tx: 11:52:03: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 01 DE 00 00 56 EF

Rx: 11:52:03: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 01 E8 3A 6D 00 BC 01 2A 21 8C 54 05 02 01 3A 6D 00 BC 03 18 21 8D 54 01 02 01 3A 6D 00 BC 03 19 21 8D 54 02 02 01 3A 6D 00 BC 03 19 21 8D 54 03 02 01 3A 6D 00 BC 03 19 21 8D 54 04 02 01 3A 6D 00 BC 03 19 21 8D 54 05 02 01 3A 6D 00 BD 01 22 21 8E 54 01 02 01 3A 6D 00 BD 01 22 21 8E 54 02 02 01 3A 6D 00 BD 01 22 21 8E 54 03 02 01 3A 6D 00 BD 01 23 21 8E 54 04 02 01 1F 8E

Tx: 11:52:03: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 01 E8 00 00 D8 F8

Rx: 11:52:03: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 01 F2 3A 6D 00 BD 01 23 21 8E 54 05 02 01 3A 6D 00 BD 03 16 21 8F 54 01 02 01 3A 6D 00 BD 03 16 21 8F 54 02 02 01 3A 6D 00 BD 03 16 21 8F 54 03 02 01 3A 6D 00 BD 03 17 21 8F 54 04 02 01 3A 6D 00 BD 03 17 21 8F 54 05 02 01 3A 6D 00 BE 01 8D 21 90 54 01 02 01 3A 6D 00 BE 01 8E 21 90 54 02 02 01 3A 6D 00 BE 01 8E 21 90 54 03 02 01 3A 6D 00 BE 01 8E 21 90 54 04 02 01 A2 4F

Tx: 11:52:03: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 01 F2 00 00 D3 E2

Rx: 11:52:04: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 01 FC 3A 6D 00 BE 01 8E 21 90 54 05 02 01 3A 6D 00 BE 03 7A 21 91 54 01 02 01 3A 6D 00 BE 03 7A 21 91 54 02 02 01 3A 6D 00 BE 03 7A 21 91 54 03 02 01 3A 6D 00 BE 03 7B 21 91 54 04 02 01 3A 6D 00 BE 03 7B 21 91 54 05 02 01 3A 6D 00 BF 01 86 21 92 54 01 02 01 3A 6D 00 BF 01 86 21 92 54 02 02 01 3A 6D 00 BF 01 86 21 92 54 03 02 01 3A 6D 00 BF 01 87 21 92 54 04 02 01 3D AE

Tx: 11:52:04: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 01 FC 00 00 2B 68

Rx: 11:52:04: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 02 06 3A 6D 00 BF 01 87 21 92 54 05 02 01 3A 6D 00 BF 03 E0 21 93 54 01 02 01 3A 6D 00 BF 03 E0 21 93 54 02 02 01 3A 6D 00 BF 03 E0 21 93 54 03 02 01 3A 6D 00 BF 03 E0 21 93 54 04 02 01 3A 6D 00 BF 03 E1 21 93 54 05 02 01 3A 6D 00 C0 01 F1 21 94 54 01 02 01 3A 6D 00 C0 01 F2 21 94 54 02 02 01 3A 6D 00 C0 01 F2 21 94 54 03 02 01 3A 6D 00 C0 01 F2 21 94 54 04 02 01 19 74

Tx: 11:52:04: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 02 06 00 00 88 0F

Rx: 11:52:04: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 02 10 3A 6D 00 C0 01 F2 21 94 54 05 02 01 3A 6D 00 C0 03 DE 21 95 54 01 02 01 3A 6D 00 C0 03 DE 21 95 54 02 02 01 3A 6D 00 C0 03 DE 21 95 54 03 02 01 3A 6D 00 C0 03 DF 21 95 54 04 02 01 3A 6D 00 C0 03 DF 21 95 54 05 02 01 3A 6D 00 C1 02 54 21 96 54 01 02 01 3A 6D 00 C1 02 54 21 96 54 02 02 01 3A 6D 00 C1 02 54 21 96 54 03 02 01 3A 6D 00 C1 02 54 21 96 54 04 02 01 E8 84

Tx: 11:52:05: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 02 10 00 00 22 7A

Rx: 11:52:05: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 02 1A 3A 6D 00 C1 02 55 21 96 54 05 02 01 3A 6D 00 C2 00 BD 21 97 54 01 02 01 3A 6D 00 C2 00 BD 21 97 54 02 02 01 3A 6D 00 C2 00 BD 21 97 54 03 02 01 3A 6D 00 C2 00 BE 21 97 54 04 02 01 3A 6D 00 C2 00 BE 21 97 54 05 02 01 3A 6D 00 C2 02 B6 21 98 54 01 02 01 3A 6D 00 C2 02 B6 21 98 54 02 02 01 3A 6D 00 C2 02 B6 21 98 54 03 02 01 3A 6D 00 C2 02 B7 21 98 54 04 02 01 2E 18

Tx: 11:52:05: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 02 1A 00 00 3B 51

Rx: 11:52:05: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 02 24 3A 6D 00 C2 02 B7 21 98 54 05 02 01 3A 6D 00 C3 00 C0 21 99 54 01 02 01 3A 6D 00 C3 00 C0 21 99 54 02 02 01 3A 6D 00 C3 00 C0 21 99 54 03 02 01 3A 6D 00 C3 00 C1 21 99 54 04 02 01 3A 6D 00 C3 00 C1 21 99 54 05 02 01 3A 6D 00 C3 03 1E 21 9A 54 01 02 01 3A 6D 00 C3 03 1E 21 9A 54 02 02 01 3A 6D 00 C3 03 1E 21 9A 54 03 02 01 3A 6D 00 C3 03 1F 21 9A 54 04 02 01 B6 BB

Tx: 11:52:05: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 02 24 00 00 DE 44

Rx: 11:52:06: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 02 2E 3A 6D 00 C3 03 1F 21 9A 54 05 02 01 3A 6D 00 C4 01 24 21 9B 54 01 02 01 3A 6D 00 C4 01 24 21 9B 54 02 02 01 3A 6D 00 C4 01 24 21 9B 54 03 02 01 3A 6D 00 C4 01 25 21 9B 54 04 02 01 3A 6D 00 C4 01 25 21 9B 54 05 02 01 3A 6D 00 C4 03 1E 21 9C 54 01 02 01 3A 6D 00 C4 03 1E 21 9C 54 02 02 01 3A 6D 00 C4 03 1E 21 9C 54 03 02 01 3A 6D 00 C4 03 1E 21 9C 54 04 02 01 70 97

Tx: 11:52:06: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 02 2E 00 00 A7 8C

Rx: 11:52:06: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 02 38 3A 6D 00 C4 03 1F 21 9C 54 05 02 01 3A 6D 00 C5 01 8C 21 9D 54 01 02 01 3A 6D 00 C5 01 8C 21 9D 54 02 02 01 3A 6D 00 C5 01 8C 21 9D 54 03 02 01 3A 6D 00 C5 01 8D 21 9D 54 04 02 01 3A 6D 00 C5 01 8D 21 9D 54 05 02 01 3A 6D 00 C5 03 82 21 9E 54 01 02 01 3A 6D 00 C5 03 82 21 9E 54 02 02 01 3A 6D 00 C5 03 82 21 9E 54 03 02 01 3A 6D 00 C5 03 83 21 9E 54 04 02 01 2D AD

Tx: 11:52:06: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 02 38 00 00 6D 1A

Rx: 11:52:06: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 02 42 3A 6D 00 C5 03 83 21 9E 54 05 02 01 3A 6D 00 C6 01 8C 21 9F 54 01 02 01 3A 6D 00 C6 01 8C 21 9F 54 02 02 01 3A 6D 00 C6 01 8C 21 9F 54 03 02 01 3A 6D 00 C6 01 8C 21 9F 54 04 02 01 3A 6D 00 C6 01 8D 21 9F 54 05 02 01 3A 6D 00 C6 03 DF 21 A0 54 01 02 01 3A 6D 00 C6 03 DF 21 A0 54 02 02 01 3A 6D 00 C6 03 DF 21 A0 54 03 02 01 3A 6D 00 C6 03 DF 21 A0 54 04 02 01 FB 7F

Tx: 11:52:07: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 02 42 00 00 AB 83

Rx: 11:52:07: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 02 4C 3A 6D 00 C6 03 E0 21 A0 54 05 02 01 3A 6D 00 C7 01 EC 21 A1 54 01 02 01 3A 6D 00 C7 01 EC 21 A1 54 02 02 01 3A 6D 00 C7 01 EC 21 A1 54 03 02 01 3A 6D 00 C7 01 ED 21 A1 54 04 02 01 3A 6D 00 C7 01 ED 21 A1 54 05 02 01 3A 6D 00 C8 00 06 21 A2 54 01 02 01 3A 6D 00 C8 00 06 21 A2 54 02 02 01 3A 6D 00 C8 00 06 21 A2 54 03 02 01 3A 6D 00 C8 00 07 21 A2 54 04 02 01 AB 74

Tx: 11:52:07: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 02 4C 00 00 F2 2C

Rx: 11:52:07: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 02 56 3A 6D 00 C8 00 07 21 A2 54 05 02 01 3A 6D 00 C8 02 55 21 A3 54 01 02 01 3A 6D 00 C8 02 55 21 A3 54 02 02 01 3A 6D 00 C8 02 56 21 A3 54 03 02 01 3A 6D 00 C8 02 56 21 A3 54 04 02 01 3A 6D 00 C8 02 56 21 A3 54 05 02 01 3A 6D 00 C9 00 65 21 A4 54 01 02 01 3A 6D 00 C9 00 66 21 A4 54 02 02 01 3A 6D 00 C9 00 66 21 A4 54 03 02 01 3A 6D 00 C9 00 66 21 A4 54 04 02 01 C5 D1

Tx: 11:52:07: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 02 56 00 00 99 D5

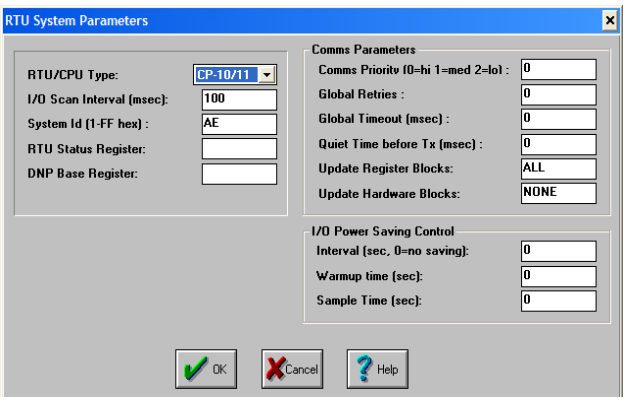
Rx: 11:52:08: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 02 60 3A 6D 00 C9 00 66 21 A4 54 05 02 01 3A 6D 00 C9 02 57 21 A5 54 01 02 01 3A 6D 00 C9 02 57 21 A5 54 02 02 01 3A 6D 00 C9 02 57 21 A5 54 03 02 01 3A 6D 00 C9 02 58 21 A5 54 04 02 01 3A 6D 00 C9 02 58 21 A5 54 05 02 01 3A 6D 00 CA 00 60 21 A6 54 01 02 01 3A 6D 00 CA 00 61 21 A6 54 02 02 01 3A 6D 00 CA 00 61 21 A6 54 03 02 01 3A 6D 00 CA 00 61 21 A6 54 04 02 01 8F D8

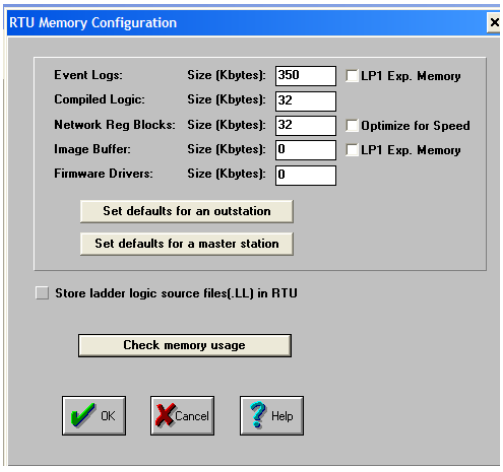
Appendix H – RTU Configuration

The RTU configuration and software were programmed using “Kingfisher Toolbox32” software.

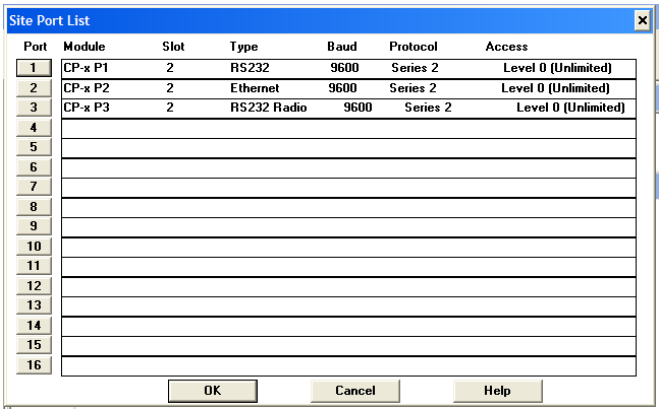
Master/ Base Station configuration



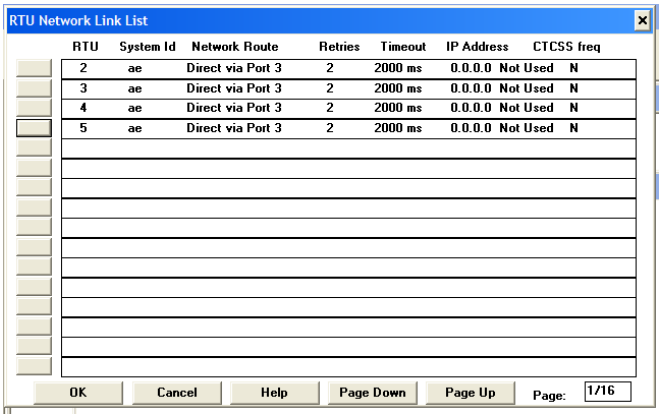
- IO Scan Interval = 100 ms (Typical) Time for the central processor to scan the IO modules, ladder logic and respond to communication messages.
- System ID: AE (default)



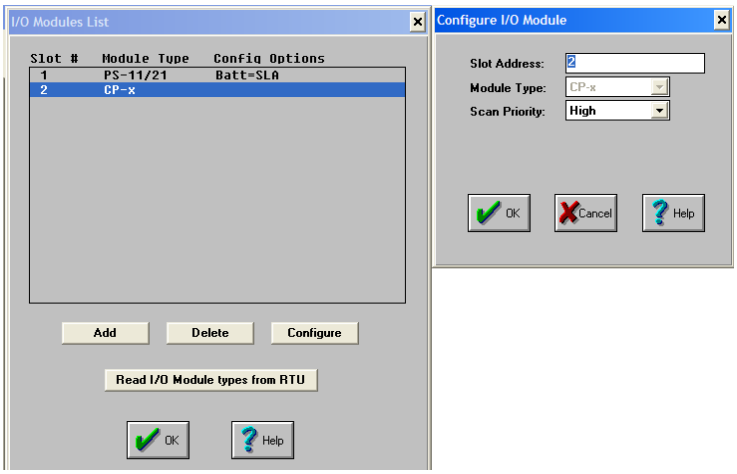
- Network Register Blocks: Memory allocation to store data from remote RTU’s typically 1KB per station
- Event logs: 350kb = 29866 Event Logs



- Port 1: Connection to PC
- Port 2: Ethernet for SCADA LAN
- Port 3: RS232 Radio for radio modem

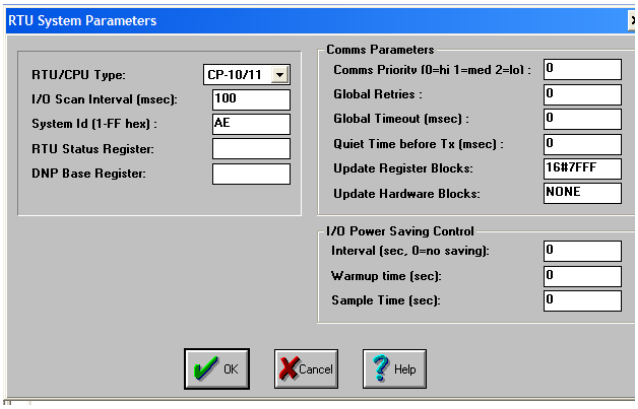


- Network Link List of RTU's



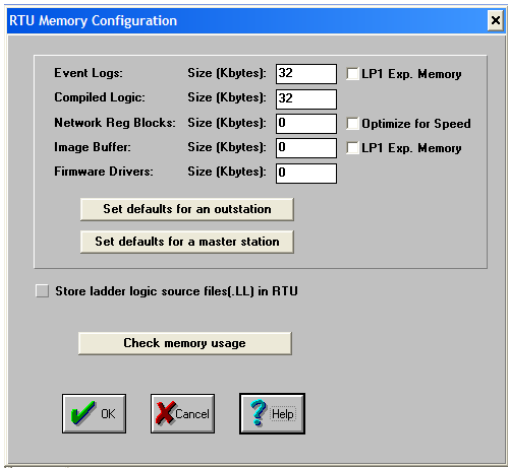
- IO modules list

Remote Field Sites Configuration



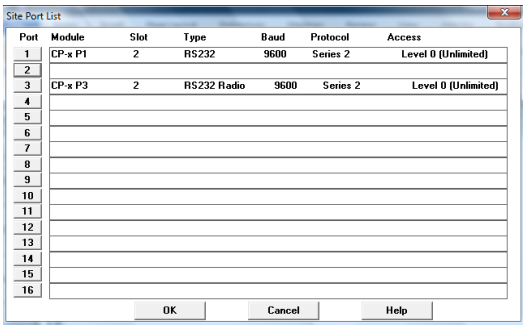
- IO Scan Interval = 100 ms (Typical)

- Update register Blocks: 16#7FFF

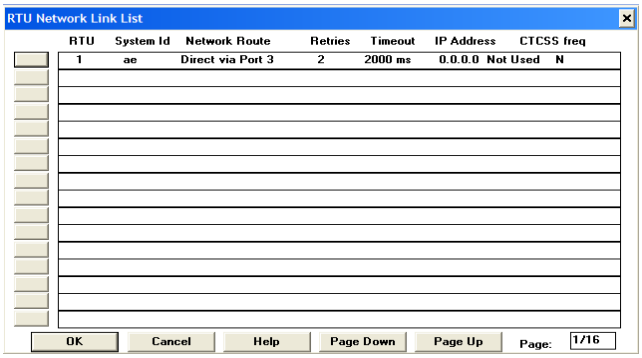


- Network Registers set to zero since the remote sites do not store any other sites data

- Event logs: 32Kb = 2730 Event log buffer size

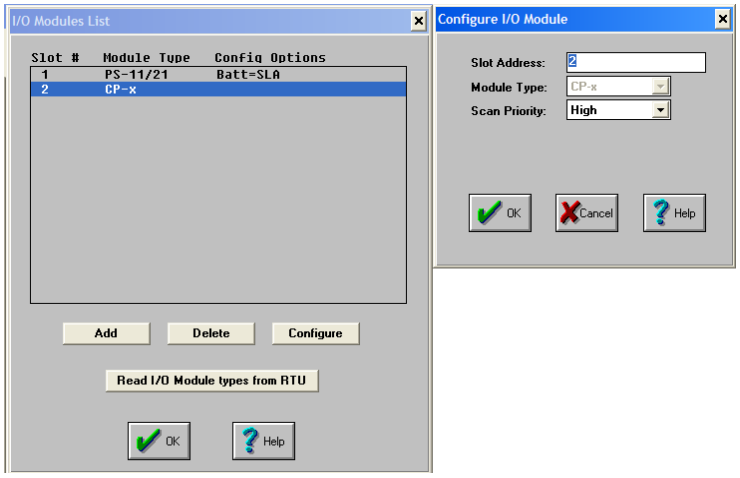


- Port 3: RS232 Radio using radio modems



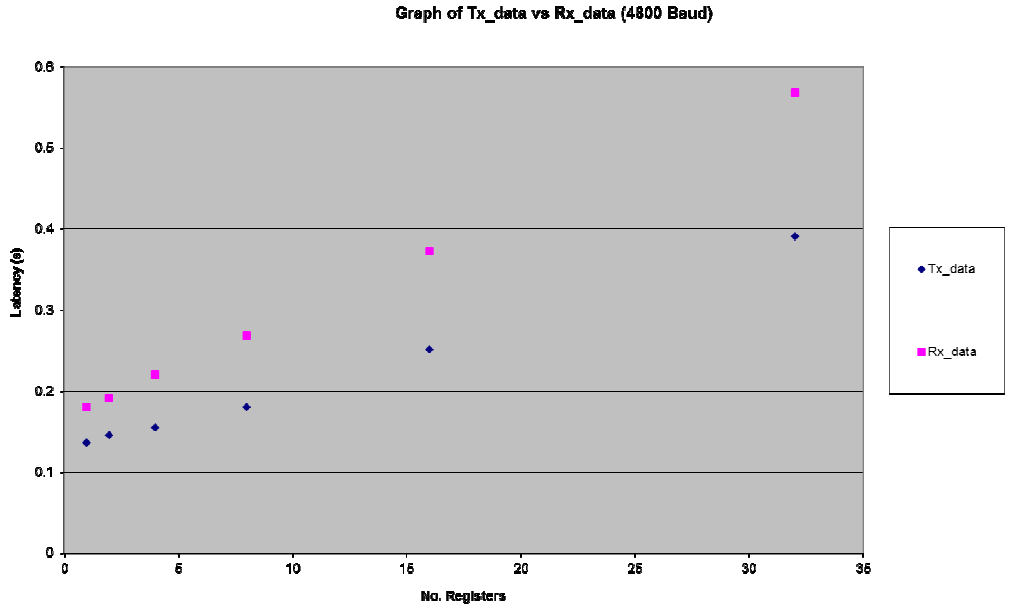
- Retries = 2 Message retries if the intial message fails

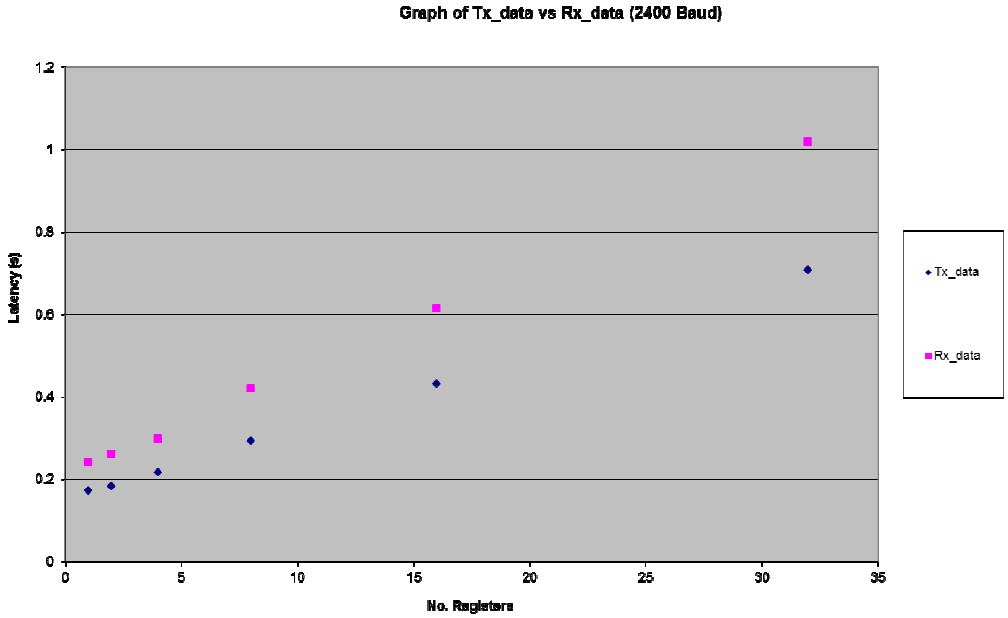
- Timeout = 2000ms response time before a failed message

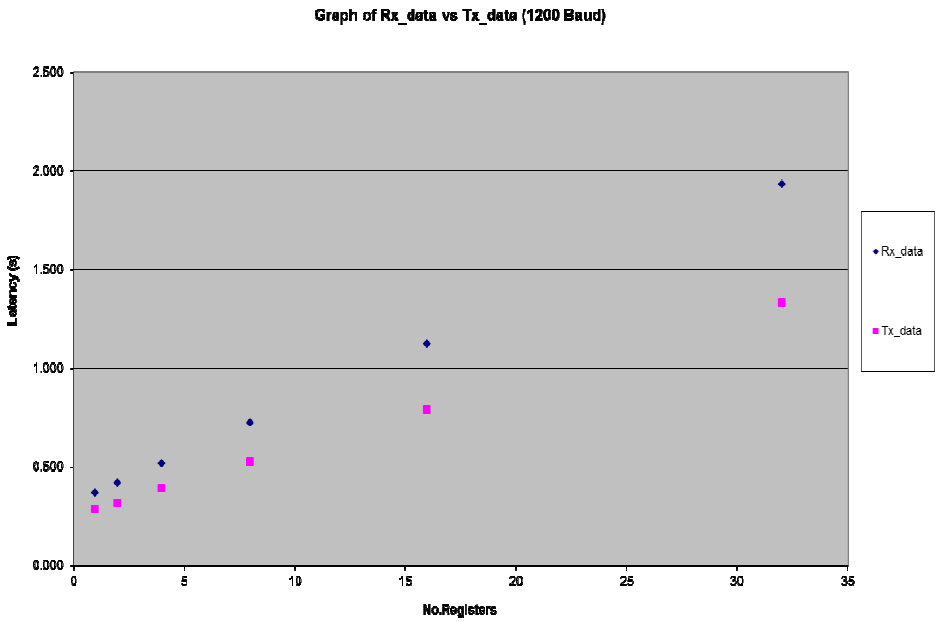


- IO modules list

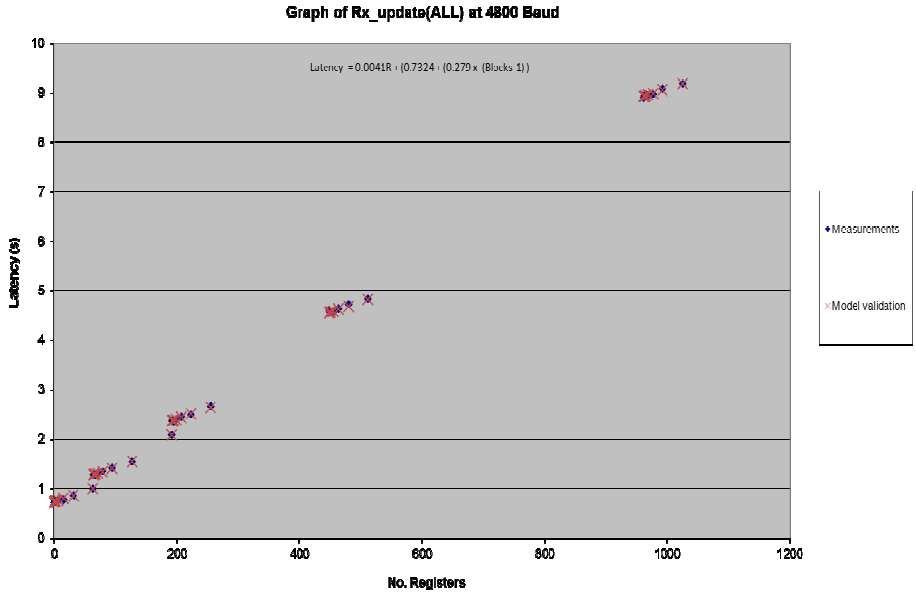
Appendix I – (Tx_data) vs (Rx_data) latency vs data throughput

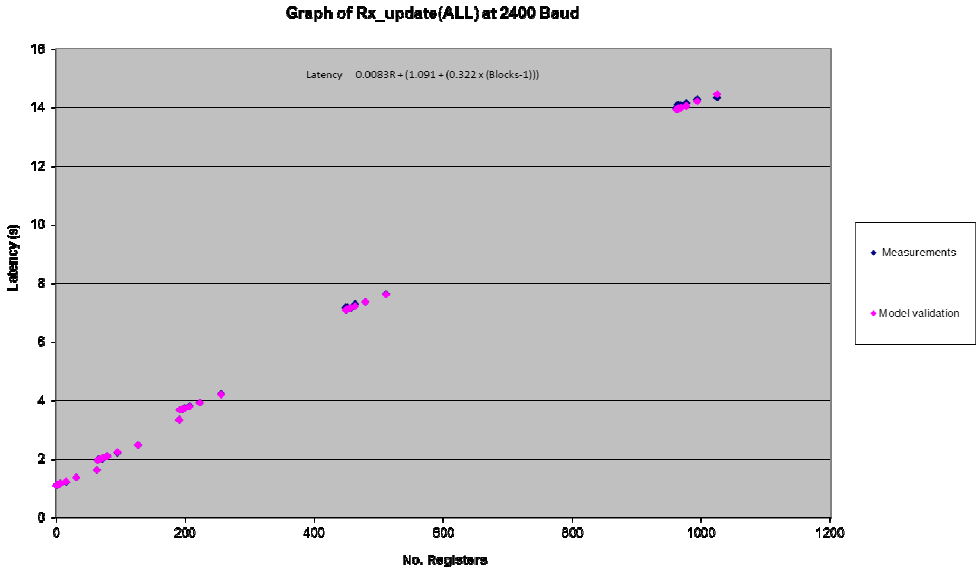


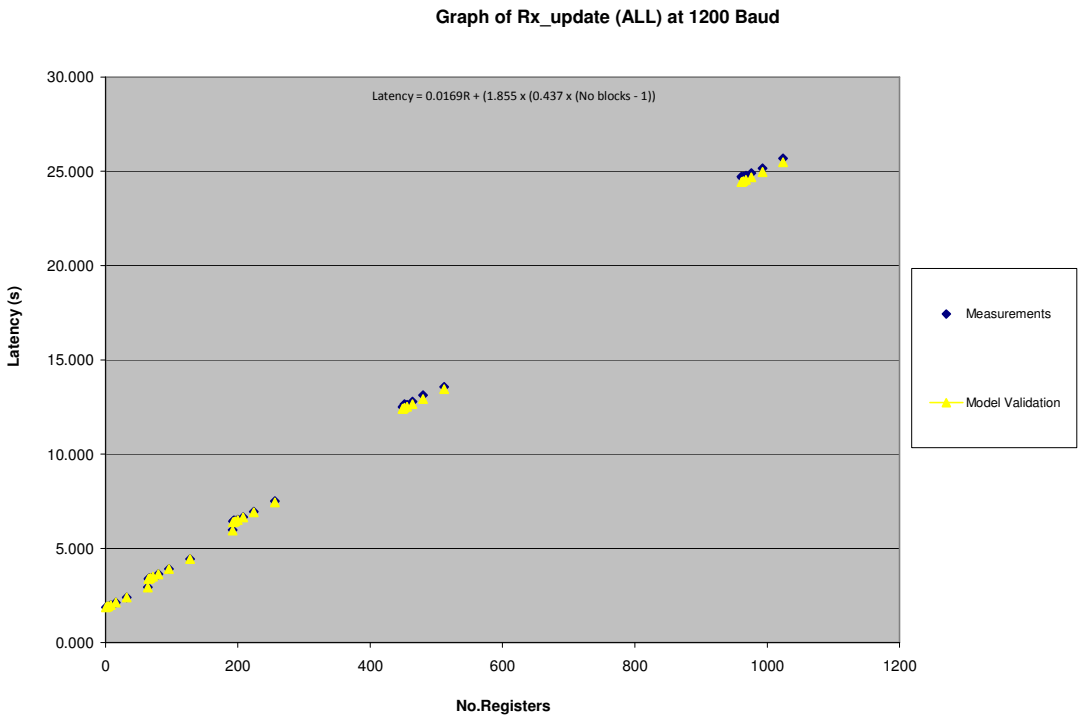



Graph of Rx_data vs Tx_data (1200 Baud)
Latency (s)
2.500
2.000
1.500
1.000
0.500
0.000
0
5
10
15
20
25
30
35
No.Registers
Rx_data
Tx_data

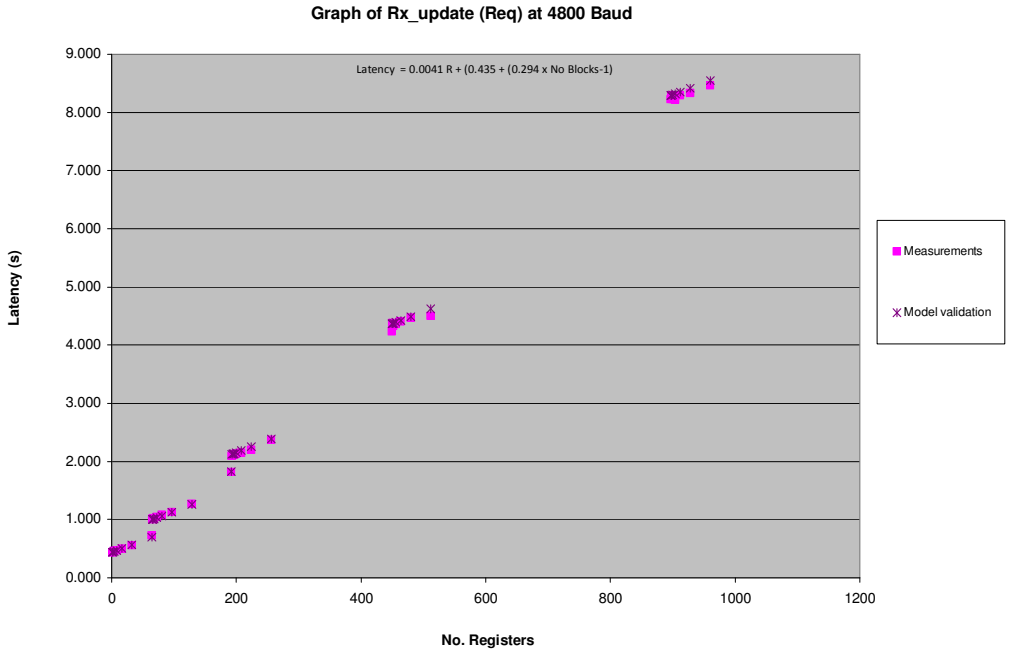
Appendix J – Rx_update (ALL) latency vs data throughput

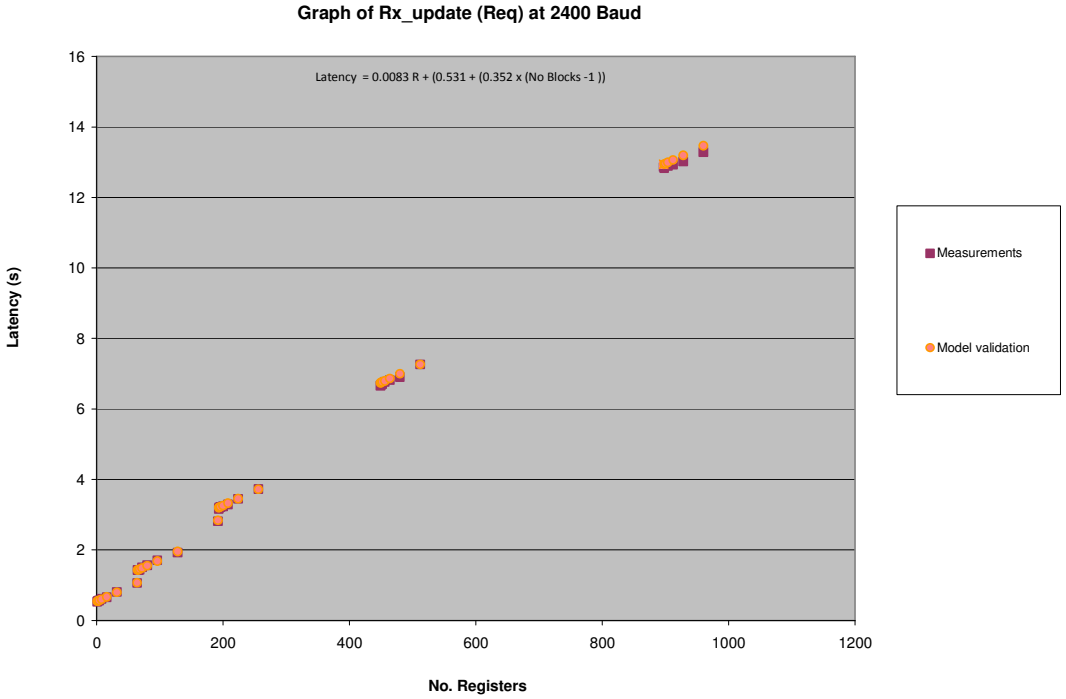


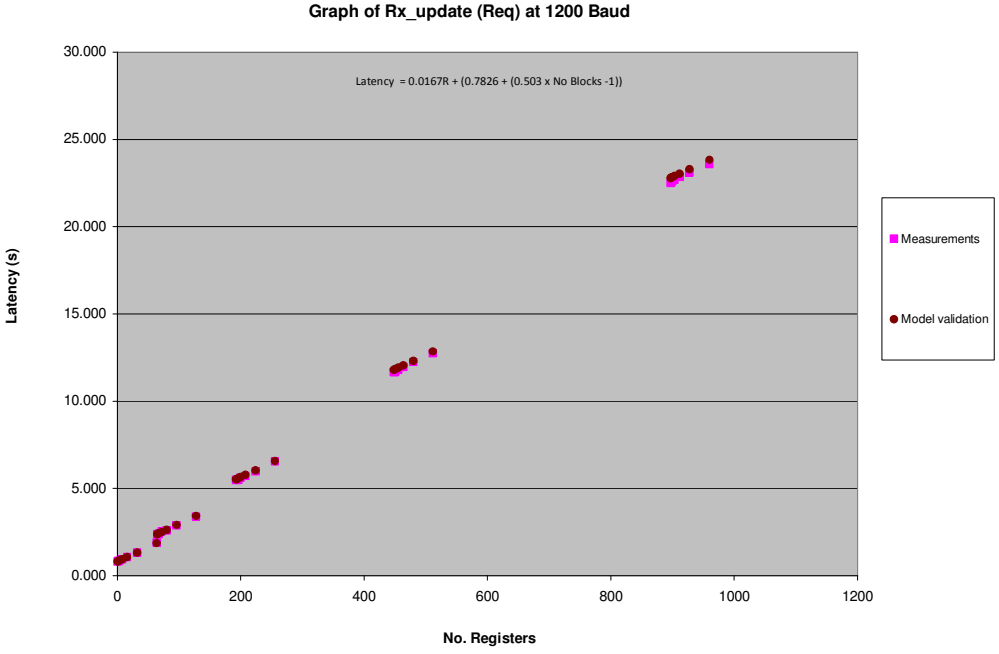



Graph of Rx_update (ALL) at 1200 Baud
Latency = 0.0169R + (1.855 x (0.437 x (No blocks - 1))
30.000
25.000
20.000
15.000
10.000
5.000
0.000
Latency (s)
0
200
400
600
800
1000
1200
No.Registers
Measurements
Model Validation

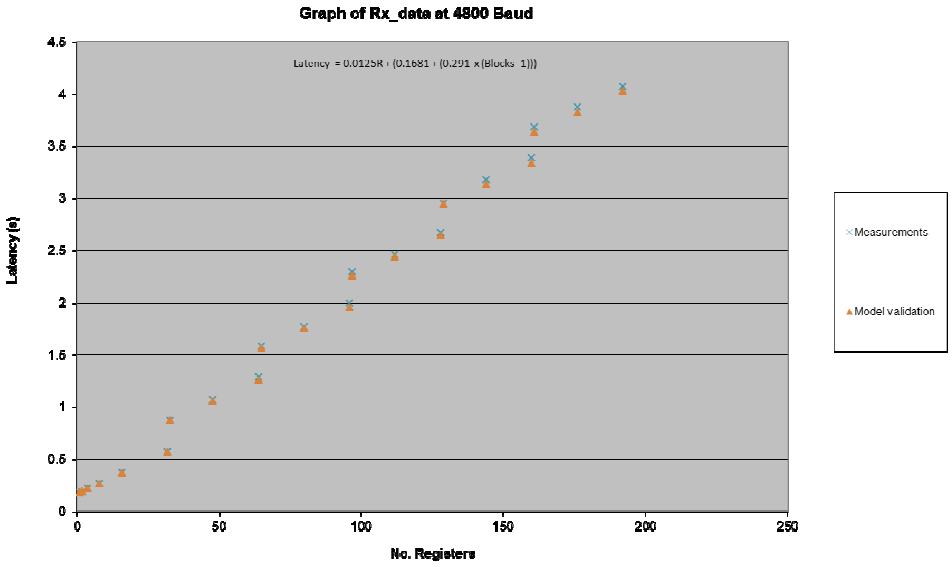
Appendix K – Rx_update (Requested number of blocks) latency vs data throughput

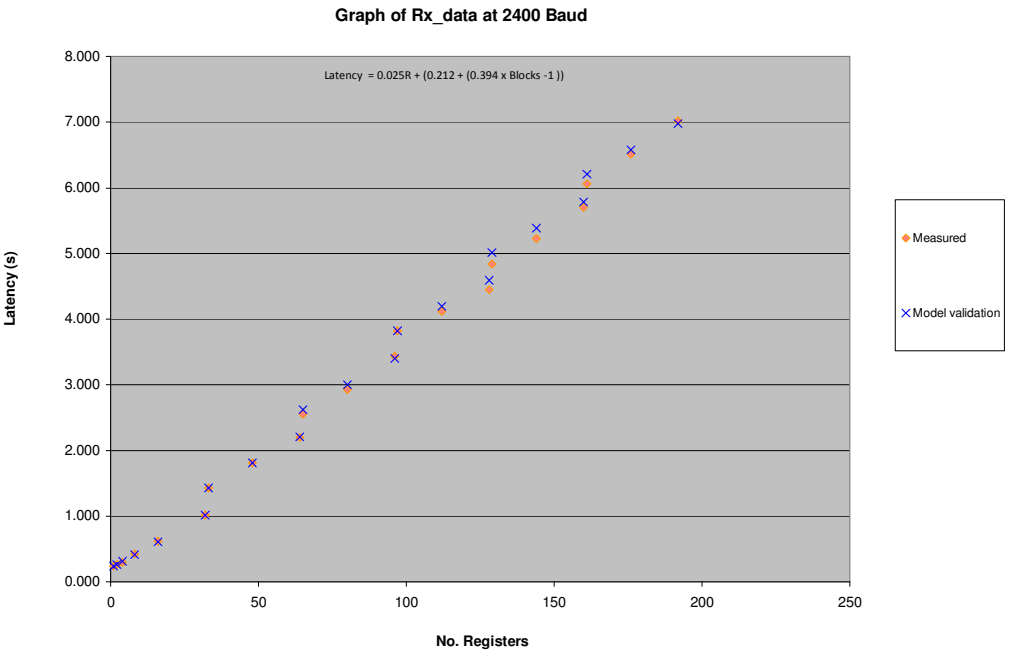


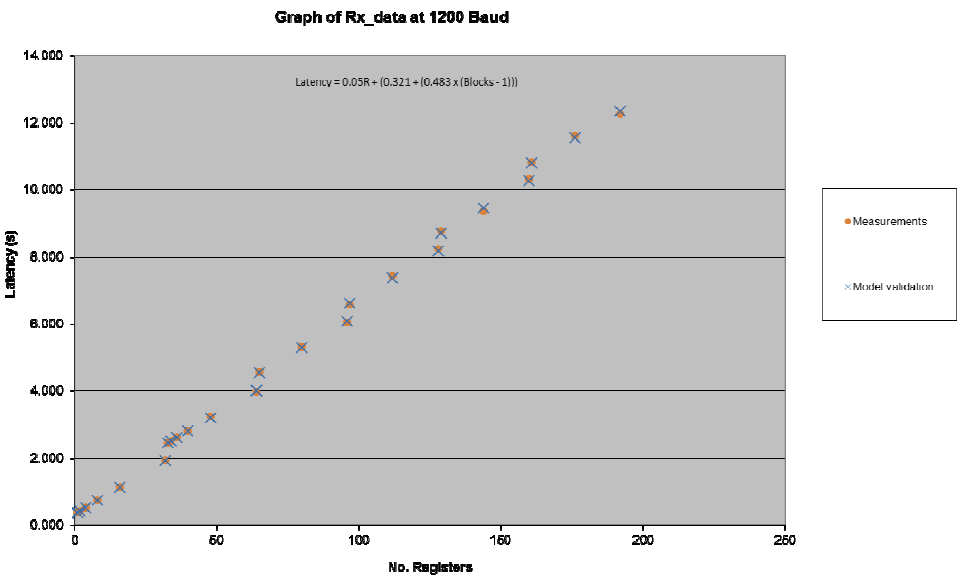

Graph of Rx_update (Req) at 2400 Baud
Latency = 0.0083 R + (0.531 + (0.352 x (No Blocks -1))
Latency (s)
No. Registers
Measurements
Model validation
0
2
4
6
8
10
12
14
16
0
200
400
600
800
1000
1200


Graph of Rx_update (Req) at 1200 Baud
Latency = 0.0167R + (0.7826 + (0.503 x No Blocks -1))
Latency (s)
No. Registers
Measurements
Model validation
0.000
5.000
10.000
15.000
20.000
25.000
30.000
0
200
400
600
800
1000
1200

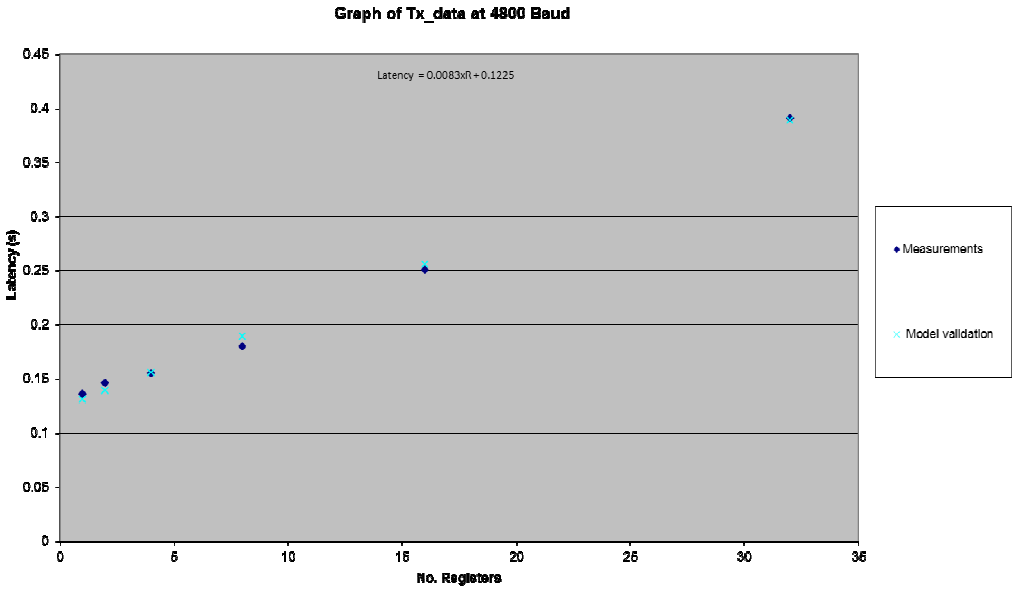
Appendix L – Rx_data

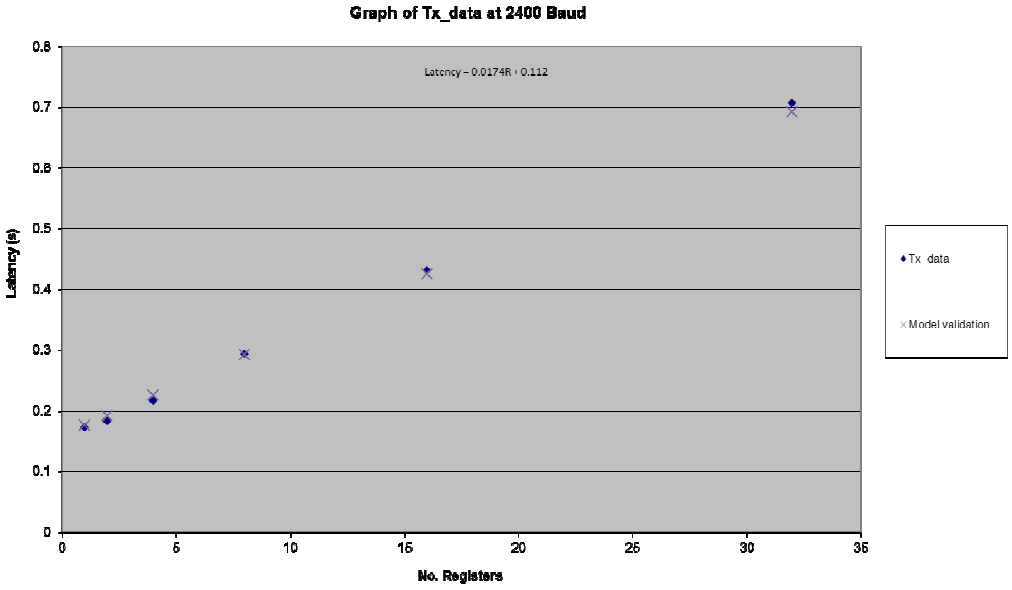


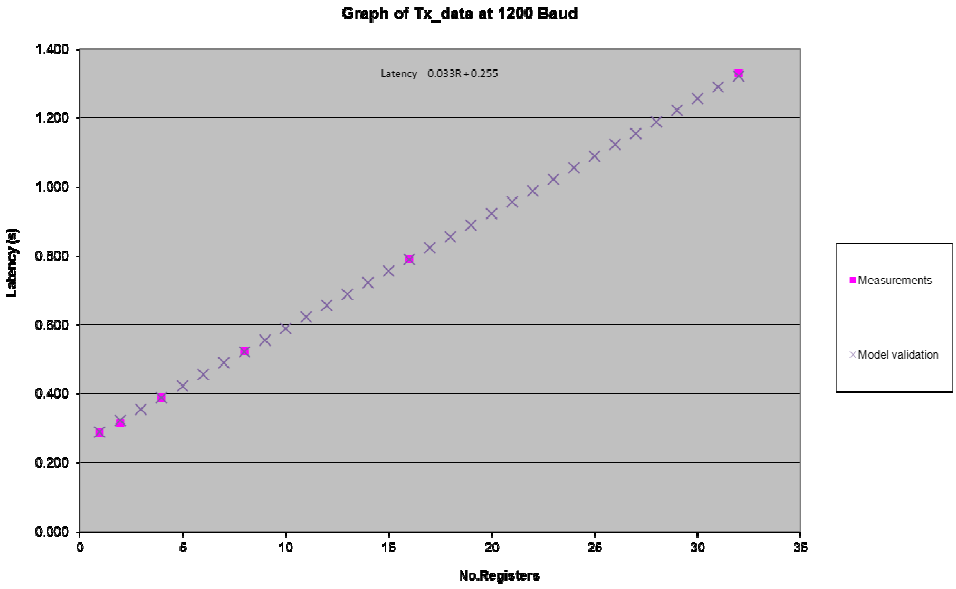



Graph of Rx_data at 1200 Baud
Latency = 0.05R + (0.321 + (0.483 x (Blocks - 1)))
Latency (s)
No. Registers
Measurements
Model validation
14.000
12.000
10.000
8.000
6.000
4.000
2.000
0.000
0
50
100
150
200
250

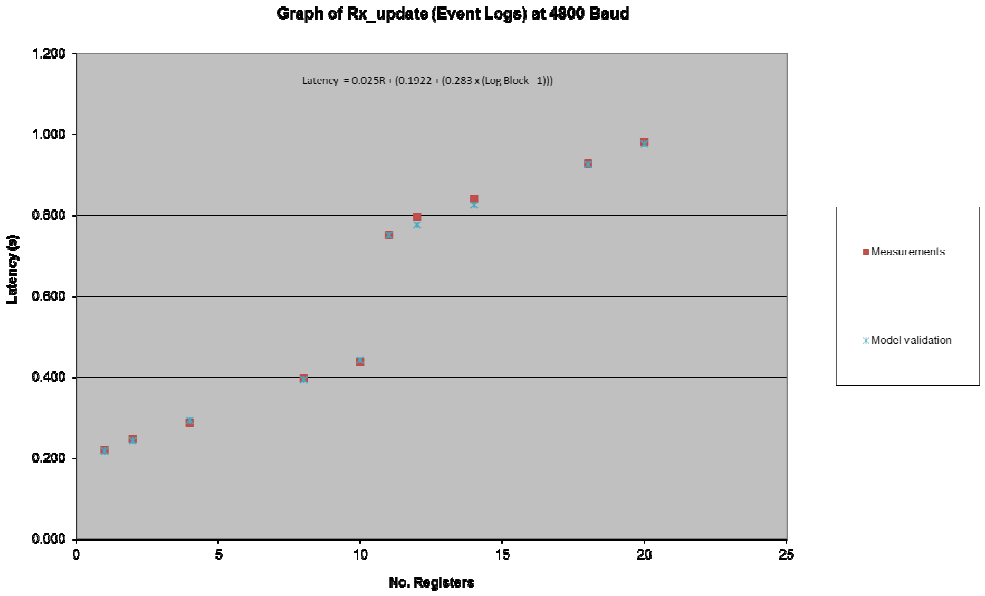
Appendix M – Tx_data

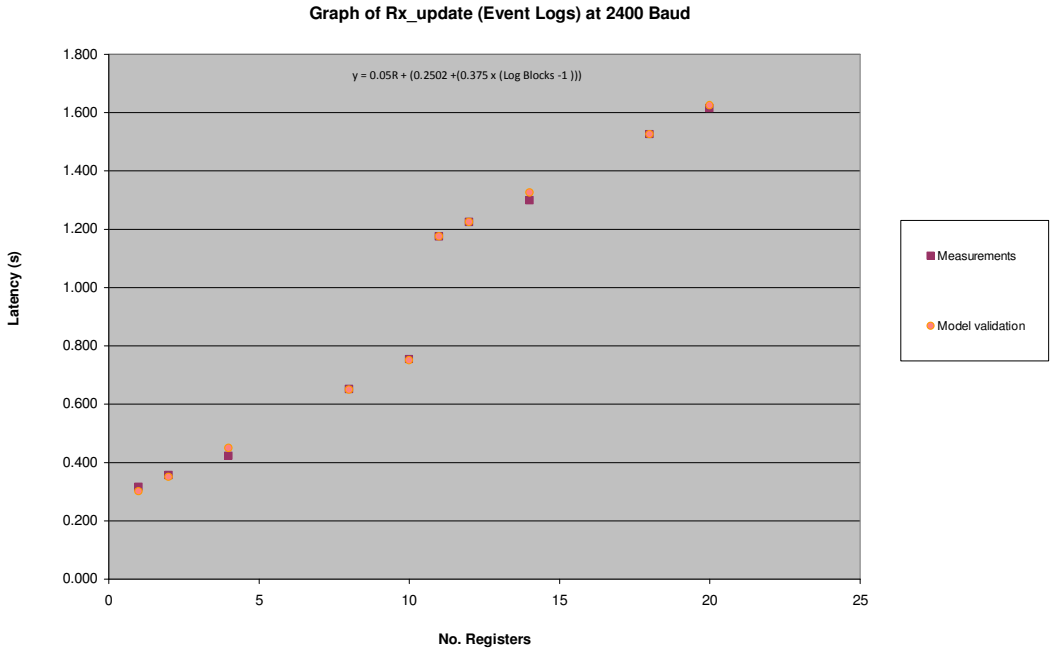


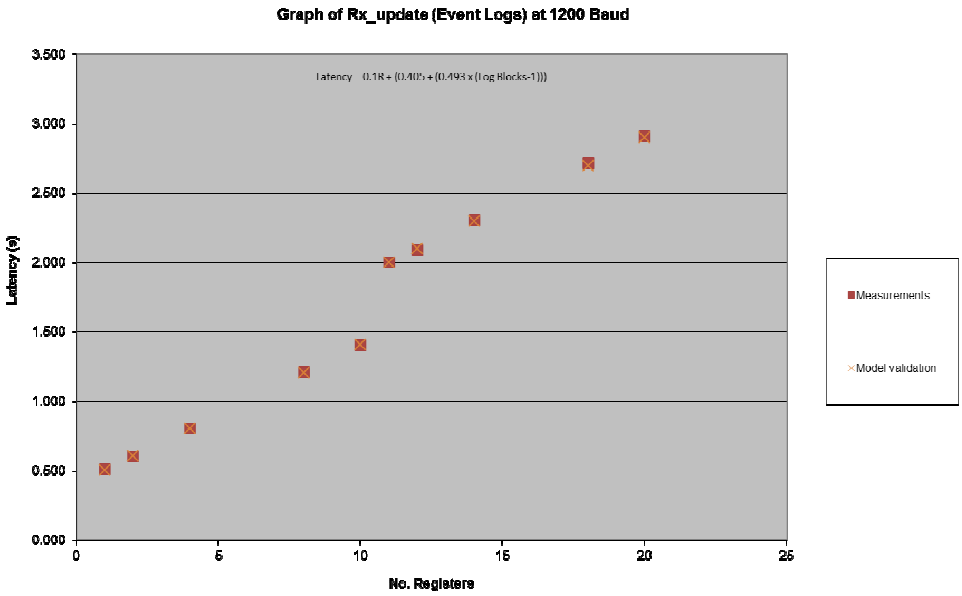



Graph of Tx_data at 1200 Baud
Latency 0.033R + 0.255
1.400
1.200
1.000
0.800
0.600
0.400
0.200
0.000
Latency (s)
0
5
10
15
20
25
30
35
No.Registers
Measurements
Model validation

Appendix N – Rx_update (Event Logs)






Graph of Rx_update (Event Logs) at 1200 Baud
Latency = 0.18 + (0.405 + (0.493 x (Log Blocks-1)))
Latency (s)
3.500
3.000
2.500
2.000
1.500
1.000
0.500
0.000
0
5
10
15
20
25
No. Registers
Measurements
Model validation

Appendix O – Kingfisher S2 Protocol Frame Format

The SERIES II protocol is based on the OSI (Open Systems Interconnection) model and is designed to achieve the following functions:

- data transfer between RTUs
- data transfer between a master RTU and a host computer
- upload and download of RTU configuration details
- diagnostic interrogation

The protocol includes a number of header bytes in each message which defines the message destination, message length, message number, message type, followed by the message argument, then a 16-bit CRC to ensure data integrity.

The message destination information includes up to 4 addresses, the SYSTEM ID (Sync character), TARGET RTU, INITIATING RTU, and VIA RTU. This allows messages to be passed up and down through the RTU network hierarchy. The two modes of RTU communications used are defined as DIRECT and INDIRECT. In DIRECT mode the VIA RTU address is set to the INIT address indicating that this message will not be relayed. In INDIRECT mode the VIA RTU address will contain the address of an RTU within one 'level' of the hierarchy which 'knows of the routing requirement' to relay the message to the TARGET RTU (ie. the VIA RTU will have a DIRECT or INDIRECT path to the TARGET RTU).

All messages are of the following form:

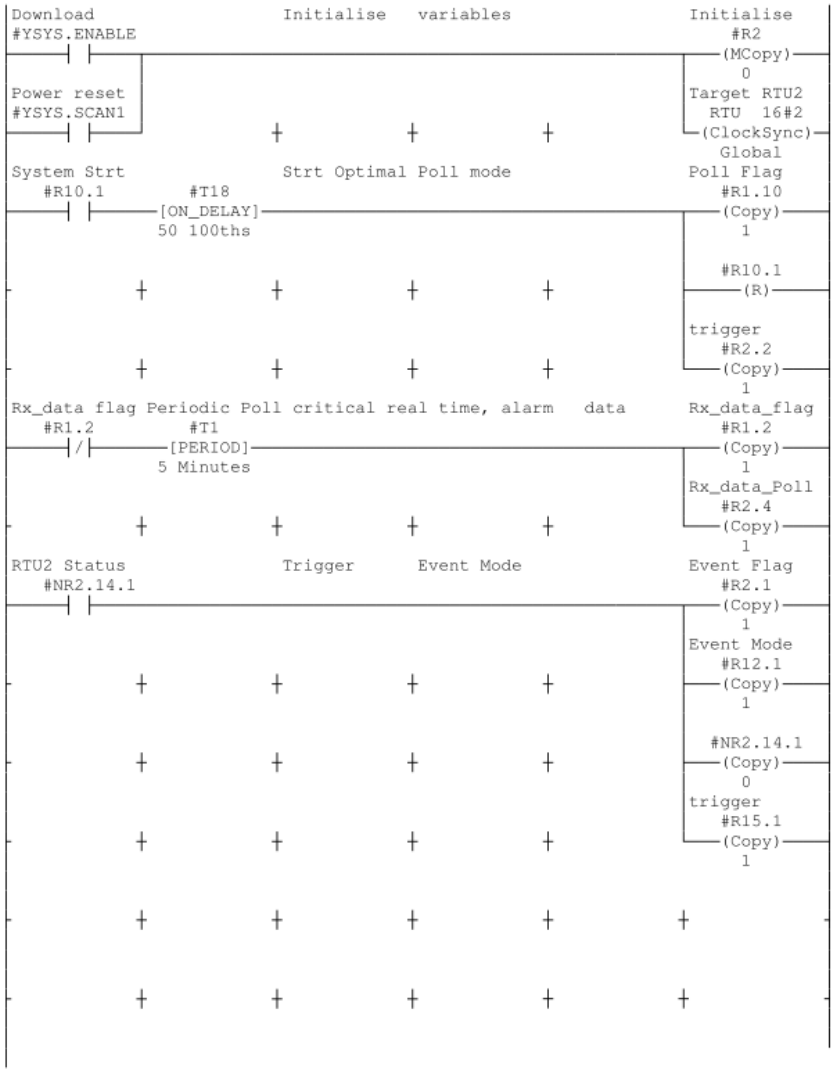
[SYSTEM ID]	(AE hex as default]
[TARGET RTU]	RTU for which this message is meant for
[No. of CHARS]	Characters in message (excluding SYSTEM ID)
[INITIATING RTU]	RTU sending message (and expecting reply)
[VIA RTU]	RTU to relay the message on to the target RTU
[MESSAGE NUMBER]	A sequential message number low 4 bits = message number (1-15, 0 message does not expect a reply) high 4 bits = reply number (1-15, 0 message is no reply)
[FUNCTION CODE]	The message type
.....	
[ARGUMENT / DATA]	The message contents
.....	
[CRC-MSB]	16 bit CCITT-CRC integrity check
[CRC-LSB]	(on all characters except the System ID)

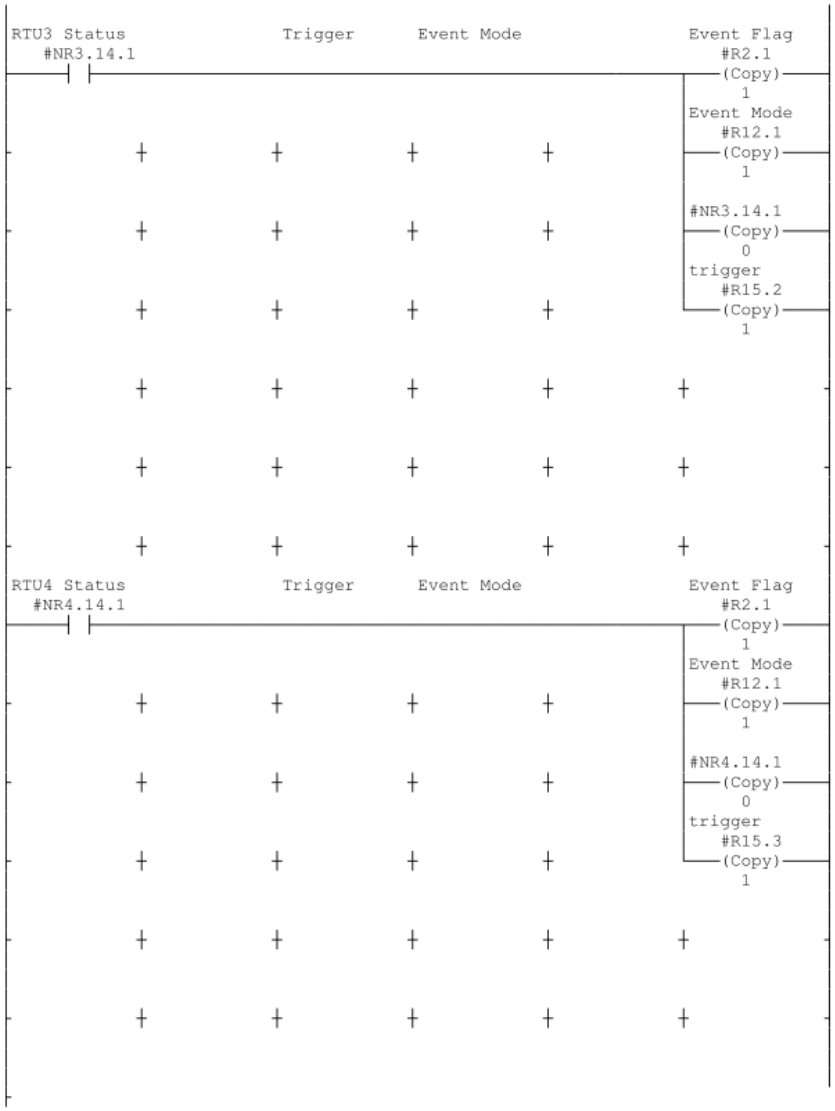
Each RTU in the network contains configuration details for each port and for each communications link that it uses. Some comms parameters are global and apply to every message - eg System ID. Other parameters can be defined differently for each port – eg Baud Rate and Protocol.

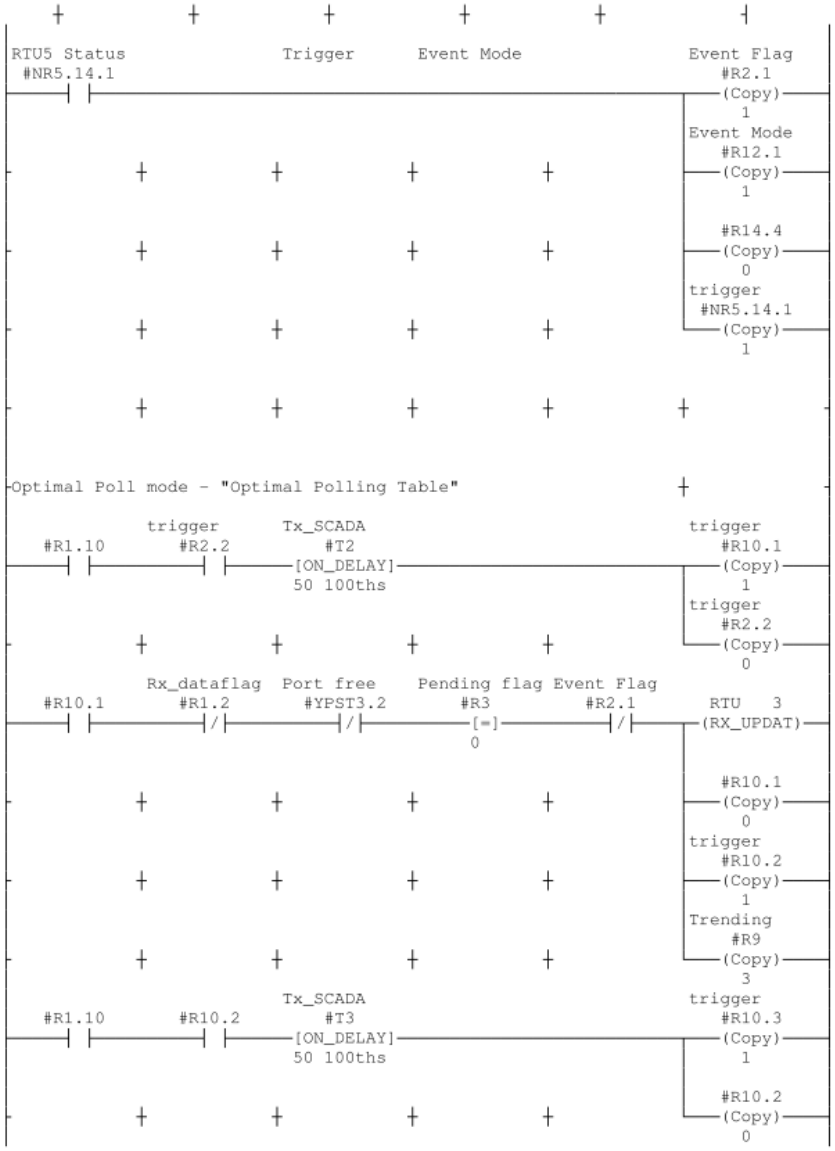
A Network List can be configured in each RTU and describes how the local RTU communicates with other RTUs in the network. The network list tells the RTU which port to use, how many retries to have, how long to wait for the reply and whether it can talk directly or indirectly to each RTU. All other RTUs that the local RTU communicates with should be entered in the network list. Note: the RTU ladder logic is able to modify any of the parameters in the network link list to allow the RTU to use another communication path.

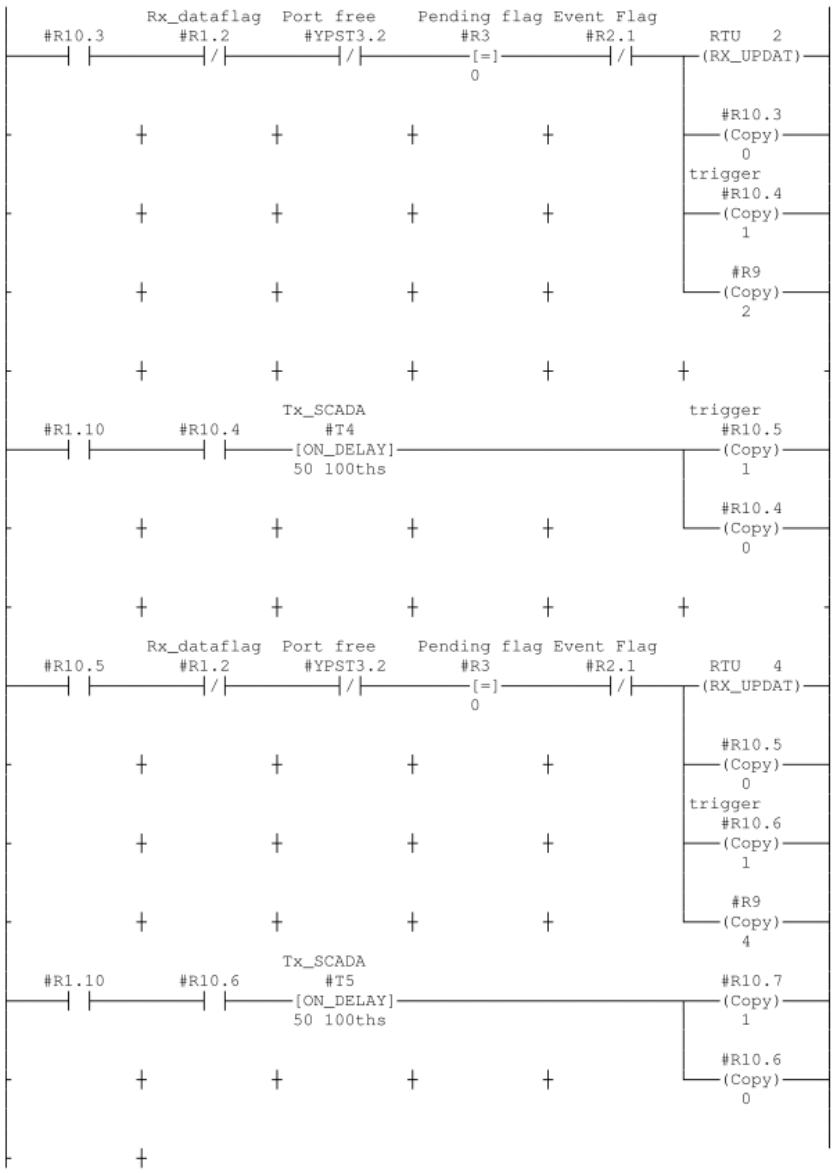
An RTU is said to talk directly to another RTU if it has a communications link to that RTU. An RTU is said to have an indirect link to an RTU if it talks via one or more 'Store and Forward' RTUs.

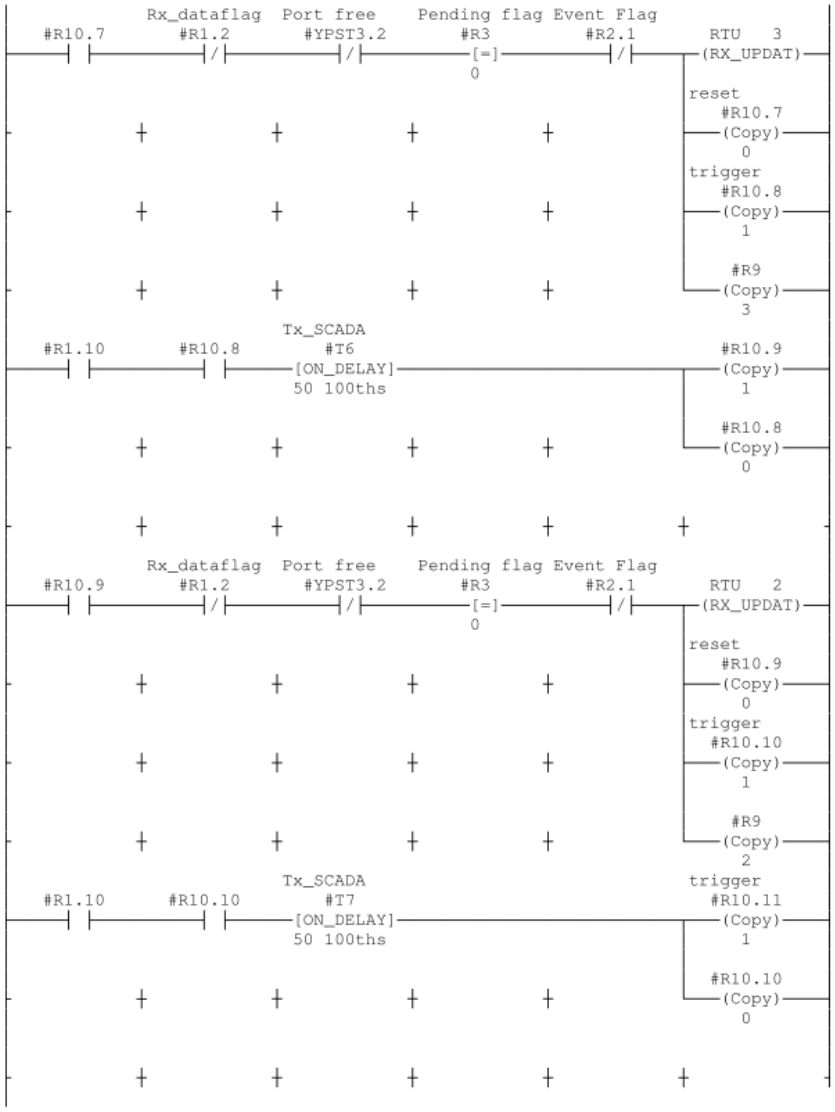
Appendix P System Code Ladder Logic

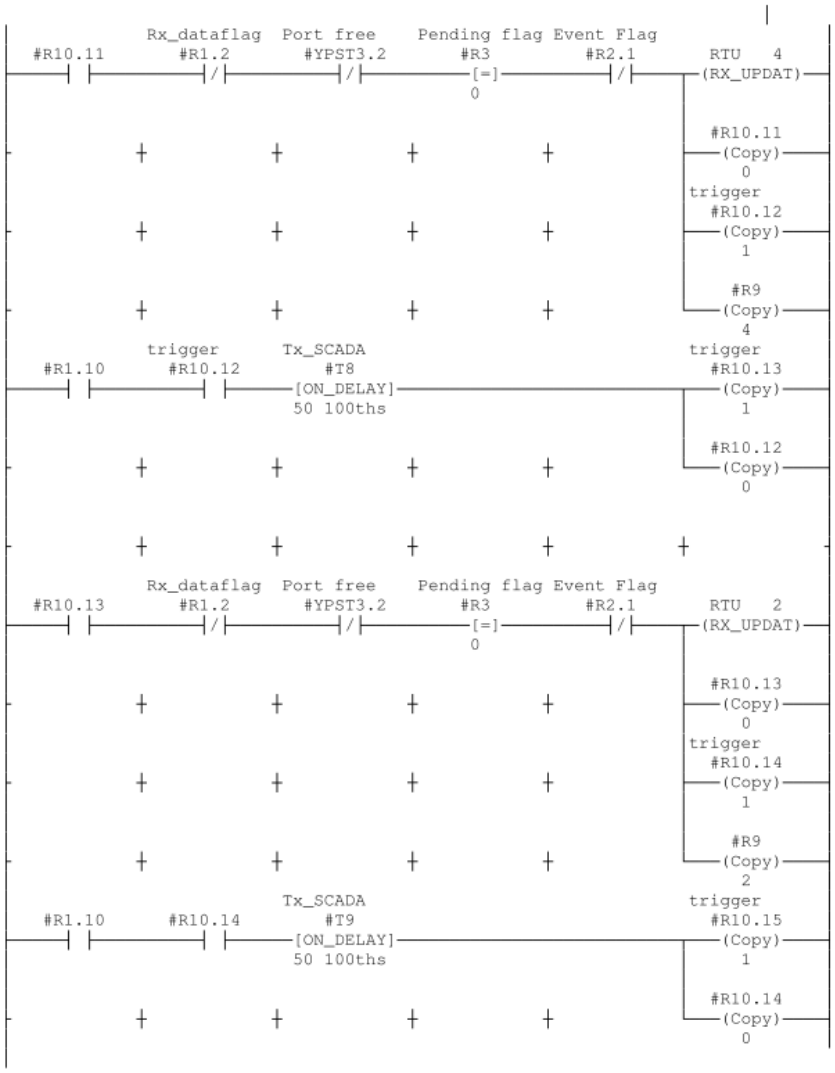


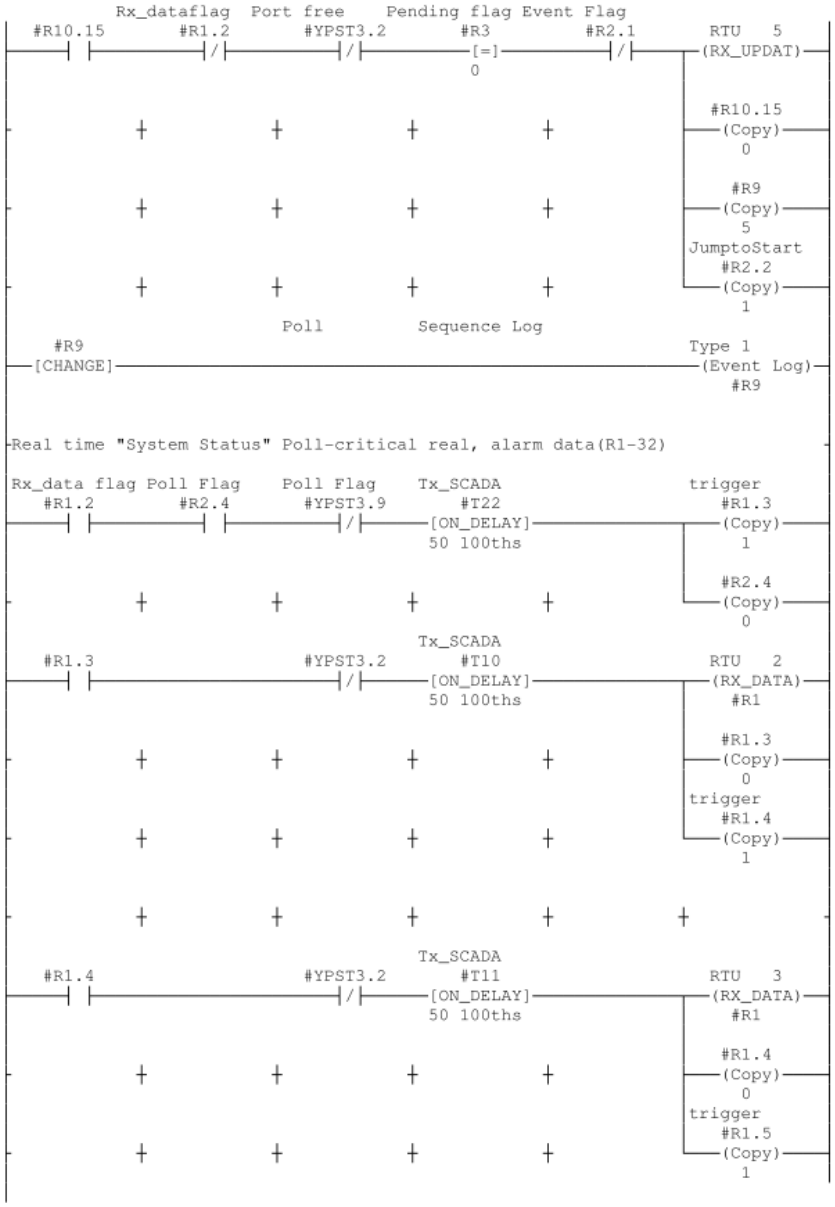

RTU3 Status
#NR3.14.1
Trigger
Event Mode
Event Flag
#R2.1
(Copy)
1
Event Mode
#R12.1
(Copy)
1
#NR3.14.1
(Copy)
0
trigger
#R15.2
(Copy)
1
RTU4 Status
#NR4.14.1
Trigger
Event Mode
Event Flag
#R2.1
(Copy)
1
Event Mode
#R12.1
(Copy)
1
#NR4.14.1
(Copy)
0
trigger
#R15.3
(Copy)
1

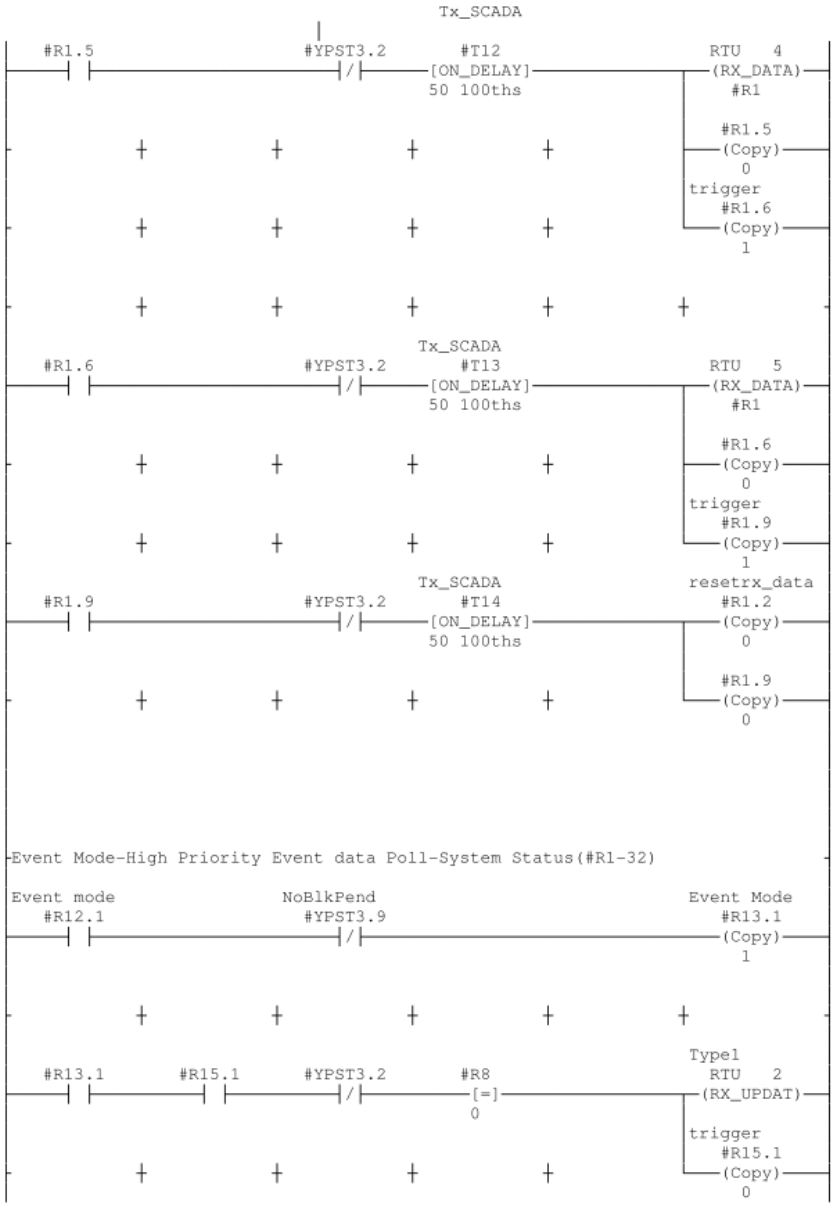

RTU5 Status
#NR5.14.1
Trigger
Event Mode
Event Flag
#R2.1
(Copy)
1
Event Mode
#R12.1
(Copy)
1
#R14.4
(Copy)
0
trigger
#NR5.14.1
(Copy)
1
Optimal Poll mode - "Optimal Polling Table"
#R1.10
trigger
#R2.2
Tx_SCADA
#T2
[ON_DELAY]
50 100ths
trigger
#R10.1
(Copy)
1
trigger
#R2.2
(Copy)
0
#R10.1
Rx_dataflag
#R1.2
Port free
#YPST3.2
Pending flag
#R3
[=]
0
Event Flag
#R2.1
RTU 3
(RX_UPDAT)
#R10.1
(Copy)
0
trigger
#R10.2
(Copy)
1
Trending
#R9
(Copy)
3
#R1.10
#R10.2
Tx_SCADA
#T3
[ON_DELAY]
50 100ths
trigger
#R10.3
(Copy)
1
#R10.2
(Copy)
0

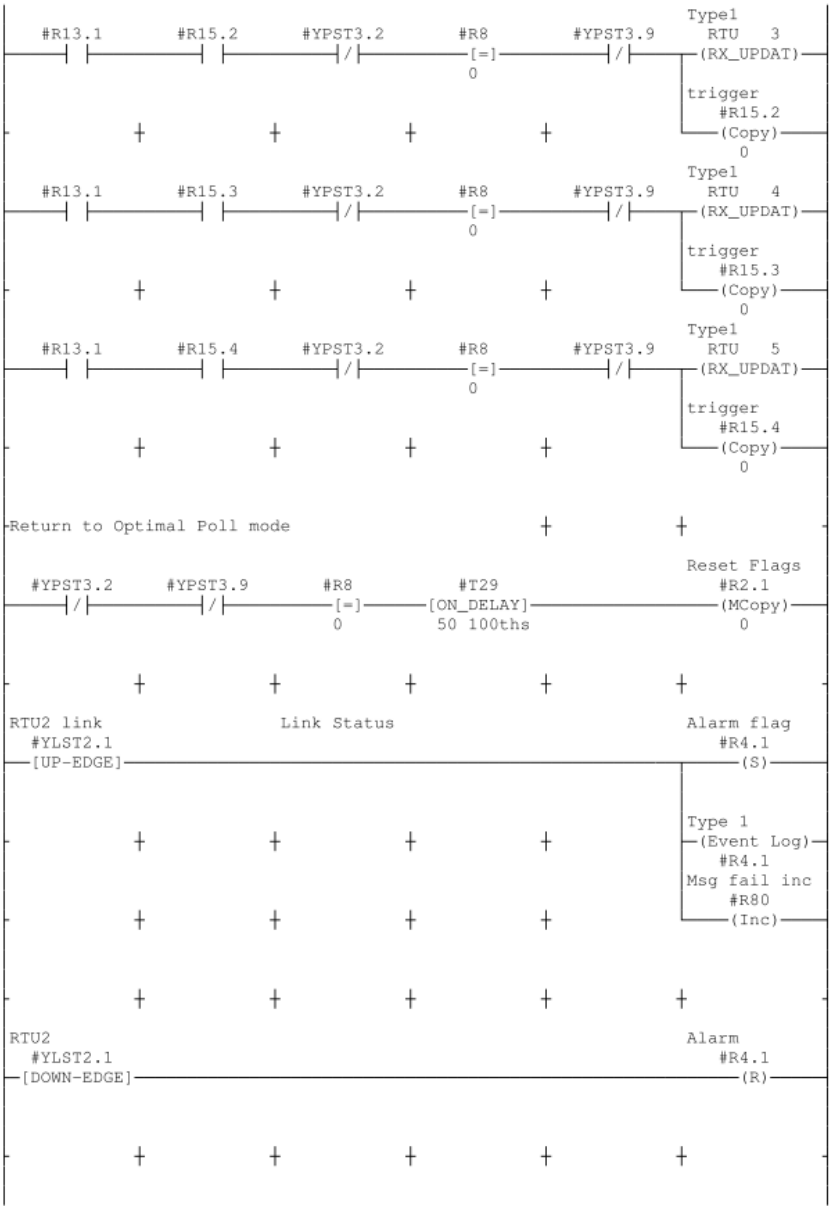

Rx_dataflag
Port free
Pending flag
Event Flag
#R10.3
#R1.2
#YPST3.2
#R3
#R2.1
RTU 2
(RX_UPDAT)
[=]
0
#R10.3
(Copy)
0
trigger
#R10.4
(Copy)
1
#R9
(Copy)
2
Tx_SCADA
#R1.10
#R10.4
#T4
[ON_DELAY]
50 100ths
trigger
#R10.5
(Copy)
1
#R10.4
(Copy)
0
Rx_dataflag
Port free
Pending flag
Event Flag
#R10.5
#R1.2
#YPST3.2
#R3
#R2.1
RTU 4
(RX_UPDAT)
[=]
0
#R10.5
(Copy)
0
trigger
#R10.6
(Copy)
1
#R9
(Copy)
4
Tx_SCADA
#R1.10
#R10.6
#T5
[ON_DELAY]
50 100ths
#R10.7
(Copy)
1
#R10.6
(Copy)
0

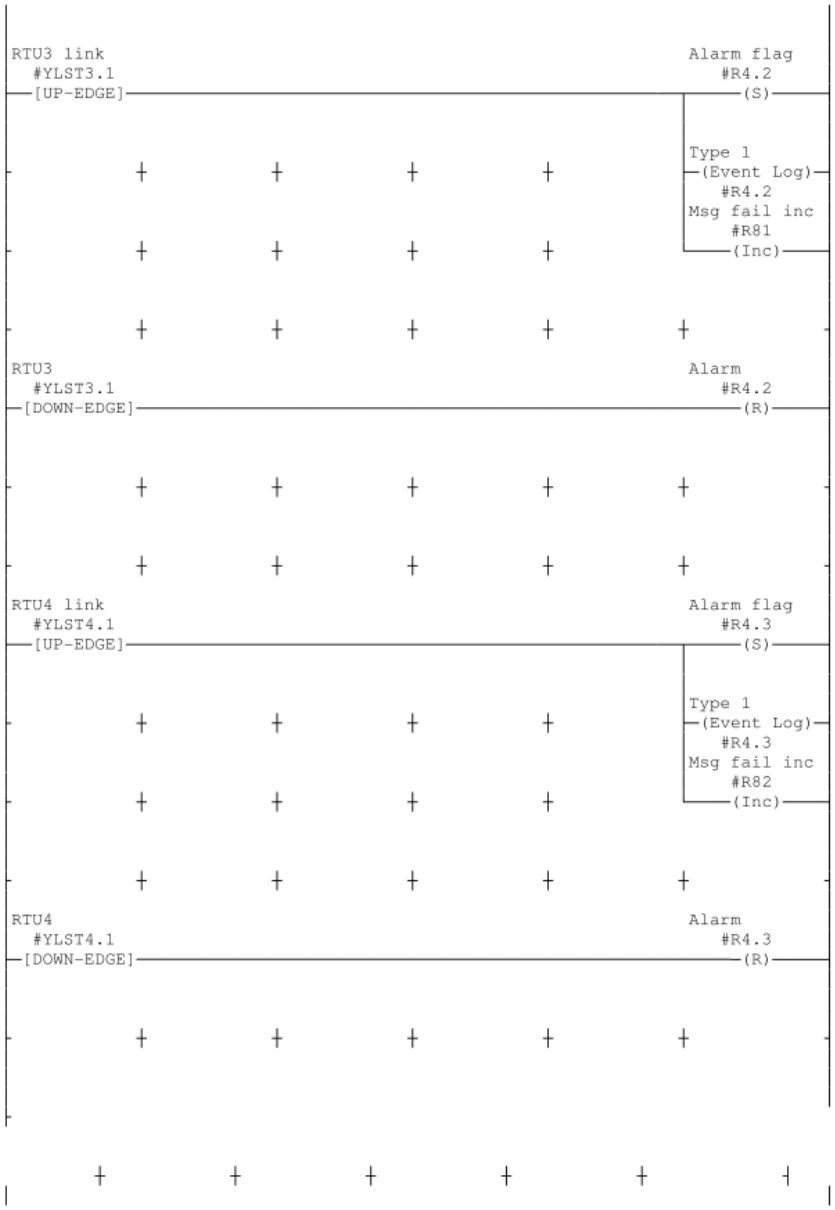

Rx_dataflag
#R10.7
#R1.2
Port free
#YPST3.2
Pending flag
#R3
[=]
0
Event Flag
#R2.1
RTU 3
(RX_UPDAT)
reset
#R10.7
(Copy)
0
trigger
#R10.8
(Copy)
1
#R9
(Copy)
3
Tx_SCADA
#R1.10
#R10.8
#T6
[ON_DELAY]
50 100ths
#R10.9
(Copy)
1
#R10.8
(Copy)
0
Rx_dataflag
#R10.9
#R1.2
Port free
#YPST3.2
Pending flag
#R3
[=]
0
Event Flag
#R2.1
RTU 2
(RX_UPDAT)
reset
#R10.9
(Copy)
0
trigger
#R10.10
(Copy)
1
#R9
(Copy)
2
Tx_SCADA
#R1.10
#R10.10
#T7
[ON_DELAY]
50 100ths
trigger
#R10.11
(Copy)
1
#R10.10
(Copy)
0

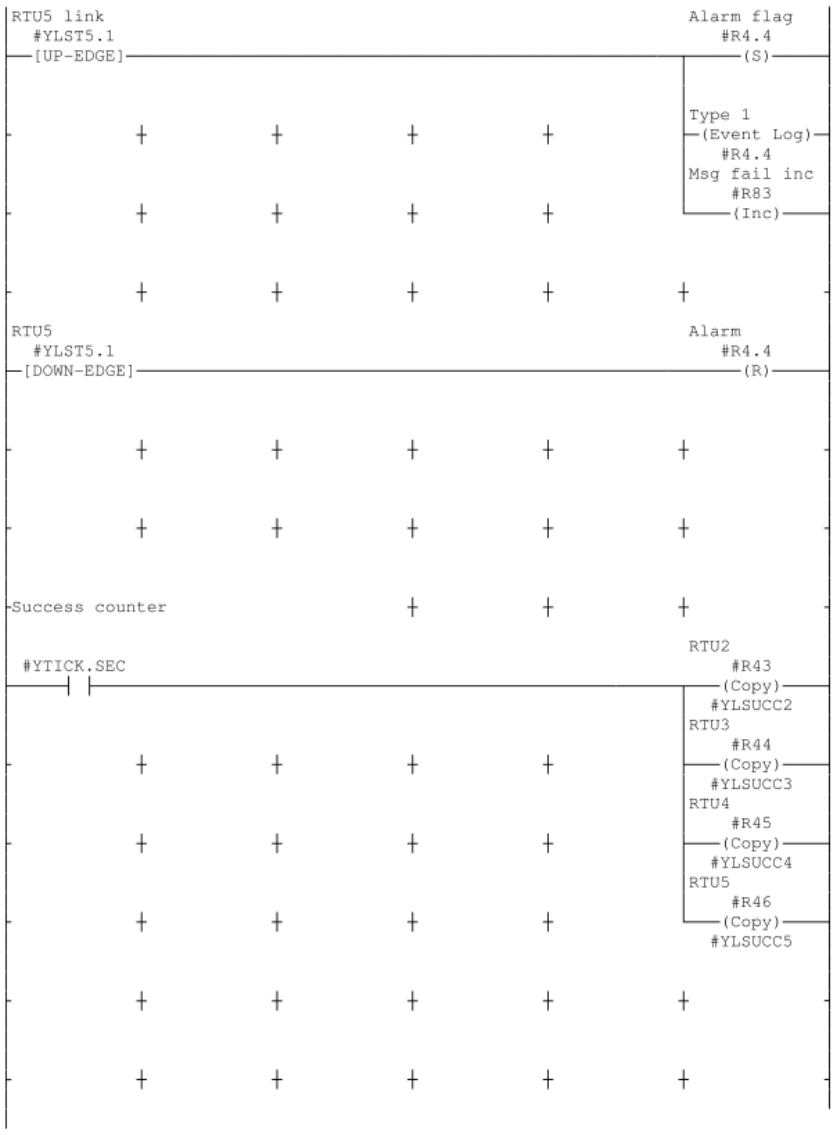

Rx_dataflag
Port free
Pending flag
Event Flag
#R10.11
#R1.2
#YPST3.2
#R3
#R2.1
RTU 4
(RX_UPDAT)
[=]
0
#R10.11
(Copy)
0
trigger
#R10.12
(Copy)
1
#R9
(Copy)
4
trigger
Tx_SCADA
#R1.10
#R10.12
#T8
[ON_DELAY]
50 100ths
trigger
#R10.13
(Copy)
1
#R10.12
(Copy)
0
Rx_dataflag
Port free
Pending flag
Event Flag
#R10.13
#R1.2
#YPST3.2
#R3
#R2.1
RTU 2
(RX_UPDAT)
[=]
0
#R10.13
(Copy)
0
trigger
#R10.14
(Copy)
1
#R9
(Copy)
2
Tx_SCADA
trigger
#R1.10
#R10.14
#T9
[ON_DELAY]
50 100ths
#R10.15
(Copy)
1
#R10.14
(Copy)
0

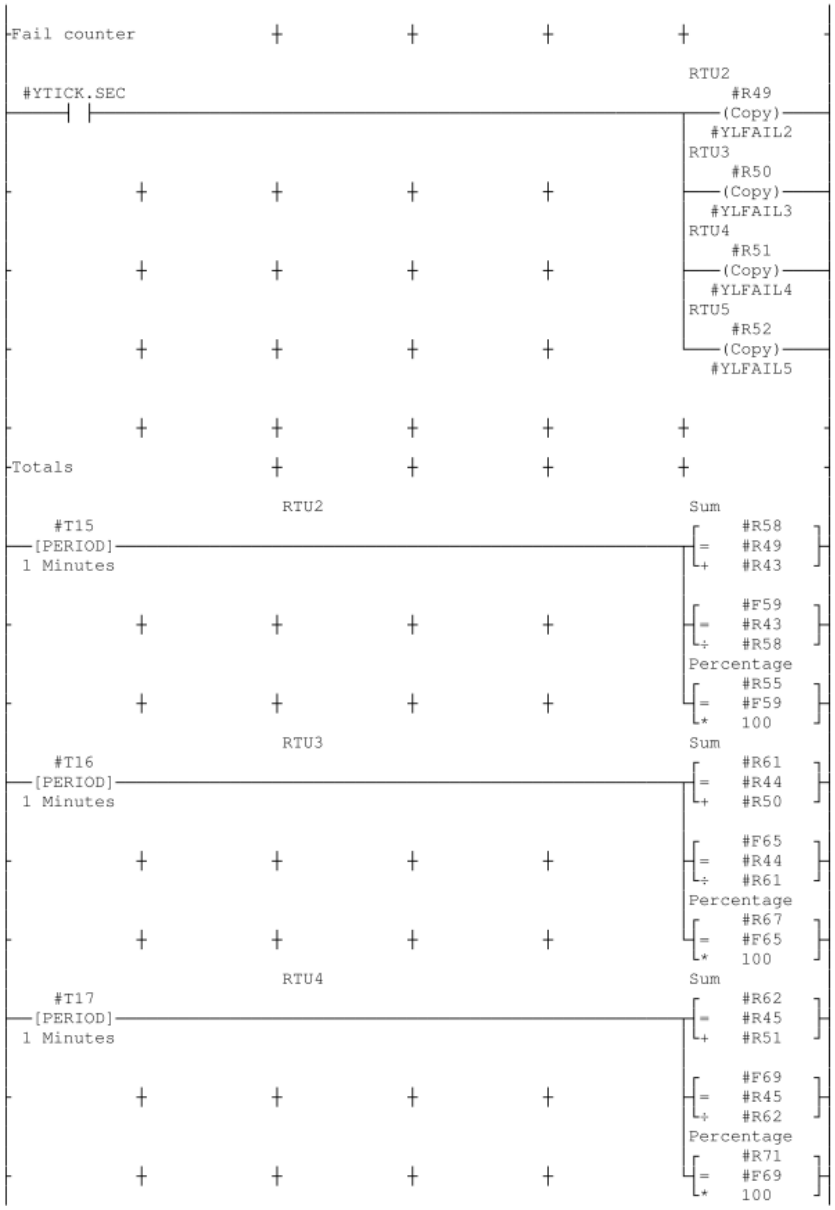

Rx_dataflag
#R10.15
Port free
#R1.2
#YPST3.2
Pending flag
#R3
[=]
0
Event Flag
#R2.1
RTU 5
(RX_UPDAT)
#R10.15
(Copy)
0
#R9
(Copy)
5
JumptoStart
#R2.2
(Copy)
1
Poll Sequence Log
#R9
[CHANGE]
Type 1
(Event Log)
#R9
Real time "System Status" Poll-critical real, alarm data(R1-32)
Rx_data flag
#R1.2
Poll Flag
#R2.4
Poll Flag
#YPST3.9
Tx_SCADA
#T22
[ON_DELAY]
50 100ths
trigger
#R1.3
(Copy)
1
#R2.4
(Copy)
0
Tx_SCADA
#R1.3
#YPST3.2
#T10
[ON_DELAY]
50 100ths
RTU 2
(RX_DATA)
#R1
#R1.3
(Copy)
0
trigger
#R1.4
(Copy)
1
Tx_SCADA
#R1.4
#YPST3.2
#T11
[ON_DELAY]
50 100ths
RTU 3
(RX_DATA)
#R1
#R1.4
(Copy)
0
trigger
#R1.5
(Copy)
1

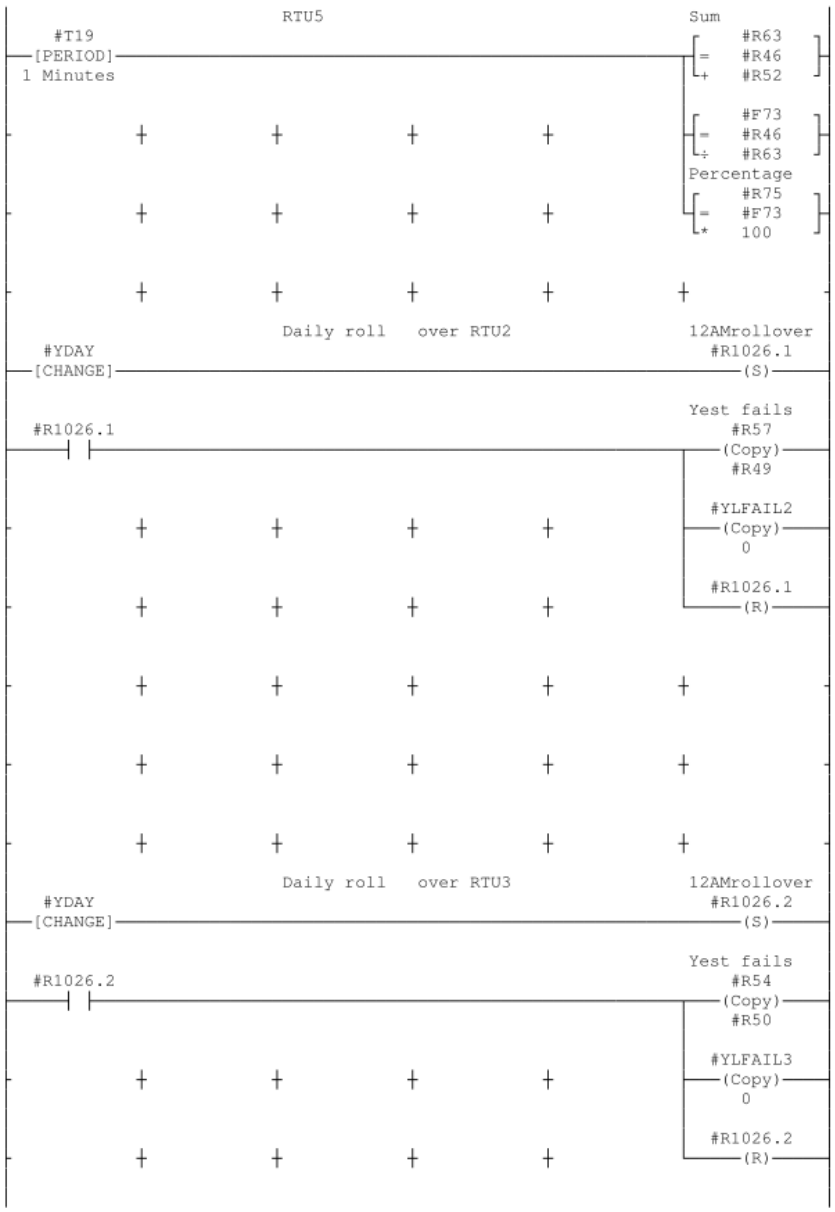

Tx_SCADA
#R1.5
#YPST3.2
#T12
[ON_DELAY]
50 100ths
RTU 4
(RX_DATA)
#R1
#R1.5
(Copy)
0
trigger
#R1.6
(Copy)
1
Tx_SCADA
#R1.6
#YPST3.2
#T13
[ON_DELAY]
50 100ths
RTU 5
(RX_DATA)
#R1
#R1.6
(Copy)
0
trigger
#R1.9
(Copy)
1
Tx_SCADA
resetrx_data
#R1.9
#YPST3.2
#T14
[ON_DELAY]
50 100ths
#R1.2
(Copy)
0
#R1.9
(Copy)
0
Event Mode-High Priority Event data Poll-System Status(#R1-32)
Event mode
#R12.1
NoBlkPend
#YPST3.9
Event Mode
#R13.1
(Copy)
1
Type1
#R13.1
#R15.1
#YPST3.2
#R8
[=]
0
RTU 2
(RX_UPDAT)
trigger
#R15.1
(Copy)
0

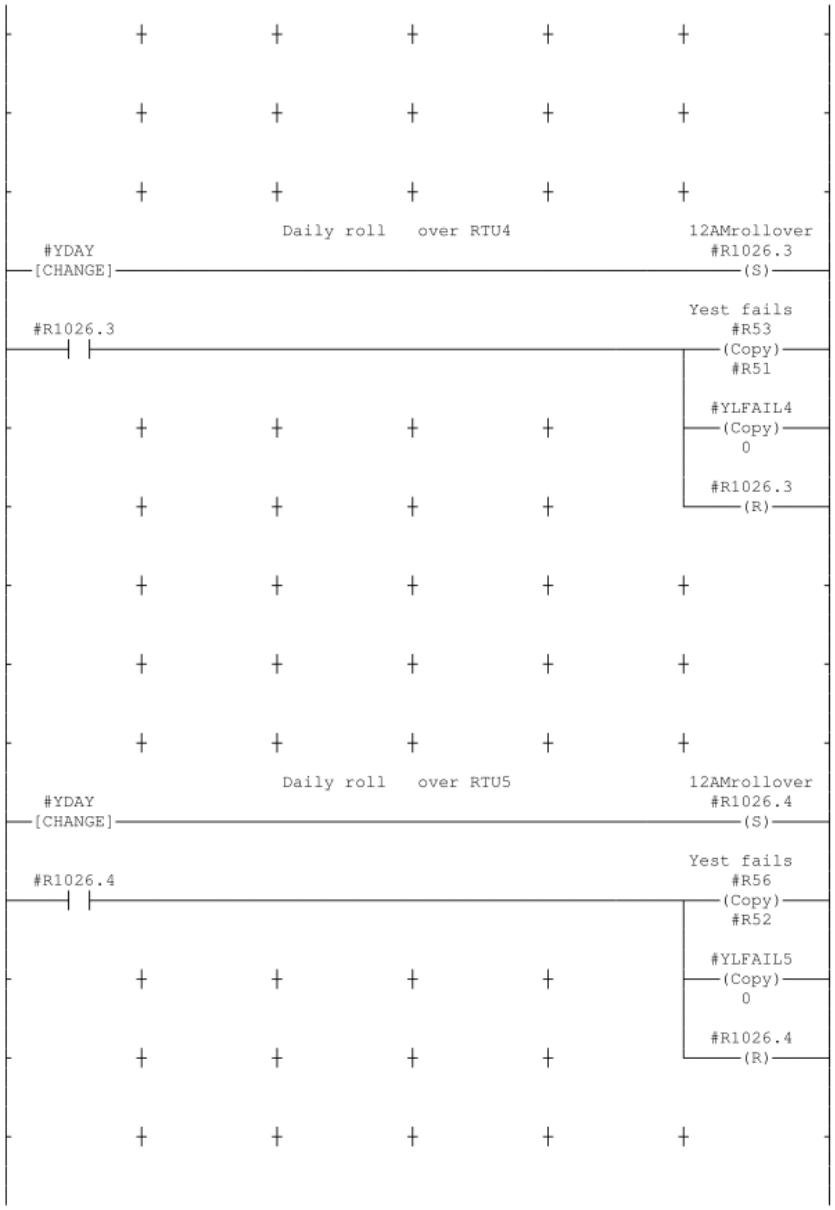

#R13.1
#R15.2
#YPST3.2
#R8
[=]
0
#YPST3.9
Type1
RTU 3
(RX_UPDAT)
trigger
#R15.2
(Copy)
0
#R13.1
#R15.3
#YPST3.2
#R8
[=]
0
#YPST3.9
Type1
RTU 4
(RX_UPDAT)
trigger
#R15.3
(Copy)
0
#R13.1
#R15.4
#YPST3.2
#R8
[=]
0
#YPST3.9
Type1
RTU 5
(RX_UPDAT)
trigger
#R15.4
(Copy)
0
Return to Optimal Poll mode
#YPST3.2
#YPST3.9
#R8
[=]
0
#T29
[ON_DELAY]
50 100ths
Reset Flags
#R2.1
(MCopy)
0
RTU2 link
#YLST2.1
[UP-EDGE]
Link Status
Alarm flag
#R4.1
(S)
Type 1
(Event Log)
#R4.1
Msg fail inc
#R80
(Inc)
RTU2
#YLST2.1
[DOWN-EDGE]
Alarm
#R4.1
(R)


RTU3 link
#YLST3.1
[UP-EDGE]
Alarm flag
#R4.2
(S)
Type 1
(Event Log)
#R4.2
Msg fail inc
#R81
(Inc)
RTU3
#YLST3.1
[DOWN-EDGE]
Alarm
#R4.2
(R)
RTU4 link
#YLST4.1
[UP-EDGE]
Alarm flag
#R4.3
(S)
Type 1
(Event Log)
#R4.3
Msg fail inc
#R82
(Inc)
RTU4
#YLST4.1
[DOWN-EDGE]
Alarm
#R4.3
(R)


RTU5 link
#YLST5.1
[UP-EDGE]
Alarm flag
#R4.4
(S)
Type 1
(Event Log)
#R4.4
Msg fail inc
#R83
(Inc)
RTU5
#YLST5.1
[DOWN-EDGE]
Alarm
#R4.4
(R)
Success counter
#YTICK.SEC
RTU2
#R43
(Copy)
#YLSUCC2
RTU3
#R44
(Copy)
#YLSUCC3
RTU4
#R45
(Copy)
#YLSUCC4
RTU5
#R46
(Copy)
#YLSUCC5


Fail counter
#YTICK.SEC
RTU2
#R49
(Copy)
#YLFAIL2
RTU3
#R50
(Copy)
#YLFAIL3
RTU4
#R51
(Copy)
#YLFAIL4
RTU5
#R52
(Copy)
#YLFAIL5
Totals
RTU2
Sum
#T15
[PERIOD]
1 Minutes
= #R58
+ #R49
#R43
= #F59
÷ #R43
#R58
Percentage
= #R55
* #F59
100
RTU3
Sum
#T16
[PERIOD]
1 Minutes
= #R61
+ #R44
#R50
= #F65
÷ #R44
#R61
Percentage
= #R67
* #F65
100
RTU4
Sum
#T17
[PERIOD]
1 Minutes
= #R62
+ #R45
#R51
= #F69
÷ #R45
#R62
Percentage
= #R71
* #F69
100


RTU5
Sum
#T19
[PERIOD]
1 Minutes
#R63
= #R46
+ #R52
#F73
= #R46
÷ #R63
Percentage
#R75
= #F73
* 100
Daily roll over RTU2
12AMrollover
#YDAY
[CHANGE]
#R1026.1
(S)
Yest fails
#R1026.1
#R57
(Copy)
#R49
#YLFAIL2
(Copy)
0
#R1026.1
(R)
Daily roll over RTU3
12AMrollover
#YDAY
[CHANGE]
#R1026.2
(S)
Yest fails
#R1026.2
#R54
(Copy)
#R50
#YLFAIL3
(Copy)
0
#R1026.2
(R)


Daily roll over RTU4
#YDAY
[CHANGE]
12AMrollover
#R1026.3
(S)
#R1026.3
Yest fails
#R53
(Copy)
#R51
#YLFAIL4
(Copy)
0
#R1026.3
(R)
Daily roll over RTU5
#YDAY
[CHANGE]
12AMrollover
#R1026.4
(S)
#R1026.4
Yest fails
#R56
(Copy)
#R52
#YLFAIL5
(Copy)
0
#R1026.4
(R)

Appendix Q - Variables list

#R-local registers	
#R1.2	Rx_data flag
#R1.10	Poll Flag
#R2.1	Event flag
#R2.2	system trigger
#R2.4	Rx_data Poll
#R3	Pending flag
#R4.1	RTU 2 alarm flag
#R4.2	RTU 3 alarm flag
#R4.3	RTU 4 alarm flag
#R4.4	RTU 5 alarm flag
#R8	Pending flag
#R9	Trending Poll sequence
#R43	RTU2 Success counter
#R44	RTU3 Success counter
#R45	RTU4 Success counter
#R46	RTU5 Success counter
#R49	RTU2 Fail counter
#R50	RTU3 Fail counter
#R51	RTU4 Fail counter
#R52	RTU5 Fail counter
#R58	RTU2 total messages
#R61	RTU3 total messages
#R62	RTU4 total messages
#R63	RTU5 total messages
#R55	RTU2 Succes messages%
#R67	RTU2 Succes messages%
#R71	RTU2 Succes messages%
#R75	RTU2 Succes messages%
#R83	Message fail increment
#R1026.1	RTU2 daily roll over flag
#R1026.2	RTU3 daily roll over flag
#R1026.3	RTU4 daily roll over flag
#R1026.4	RTU5 daily roll over flag
#YL: Network Link registers	
YLST2.1	RTU2 link status
YLST3.1	RTU3 link status
YLST4.1	RTU4 link status
YLST5.1	RTU5 link status

Appendix R - Test Results

The results are obtained using a "Communications Analyser" within Toolbox 32 software. The "Communication Analyser" shows the network traffic on the master RTU communication port.

- **Bandwidth allocation scheme polling of non critical real time data (Blocks 1-15) per remote site proposed in Chapter 2**
- **(Latency 11:17:56 to 11:18:02 = 6 seconds)**
- **Remote Site Configuration (16#7FFF)**

Tx: 11:17:56: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 26 = Request RTU update
AE 02 09 01 01 05 1A 02 F9 52

Rx: 11:17:56: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 15 = Send RTU update
AE 01 45 02 02 50 0F 0F 02 40 E1 5C 02 41 E1 5C 02 42 E1 5C 02 43 E1 5C 02 44 E1 5C 02 45 E1 5C 02 46 E1 5C 02 47 E1 5C 02 48 E1 5C 02 49 E1 5C 02 4A E1 5C 02 4B E1 5C 02 4C E1 5C 02 4D E1 5C 02 4E E1 5C 6C 5C

Tx: 11:17:56: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 06 0C 01 02 40 9A 08

Rx: 11:17:56: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 60 0D 02 40 40 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C A9 26

Tx: 11:17:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 07 0C 01 02 41 AD 39

Rx: 11:17:57: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 70 0D 02 41 40 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C

03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C
03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C
03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C 03 1C DE 2C

Tx: 11:17:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 08 0C 01 02 42 81 0B

Rx: 11:17:57: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 80 0D 02 42 40 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D D3 35

Tx: 11:17:57: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 09 0C 01 02 43 B6 3A

Rx: 11:17:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 90 0D 02 43 40 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D A4 3F

Tx: 11:17:58: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 0A 0C 01 02 44 EF 6D

Rx: 11:17:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 A0 0D 02 44 40 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03
1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D
03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 03 1D 57 37

Tx: 11:17:58: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 0B 0C 01 02 45 D8 5C

Rx: 11:17:58: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block

AE 01 8B 02 02 B0 0D 02 45 40 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 92 34

Tx: 11:17:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 0C 0C 01 02 46 5D CF

Rx: 11:17:59: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block

AE 01 8B 02 02 C0 0D 02 46 40 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E AD 35

Tx: 11:17:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 0D 0C 01 02 47 6A FE

Rx: 11:17:59: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block

AE 01 8B 02 02 D0 0D 02 47 40 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E 03 1E DA 3F

Tx: 11:17:59: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 0E 0C 01 02 48 33 A1

Rx: 11:18:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block

AE 01 8B 02 02 E0 0D 02 48 40 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F
03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F
1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F
03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03
1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 93 1C

Tx: 11:18:00: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 0F 0C 01 02 49 04 90

Rx: 11:18:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 F0 0D 02 49 40 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F
03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F
1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03
1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03
1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F 03 1F E4 16

Tx: 11:18:00: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 01 0C 01 02 4A 1F 92

Rx: 11:18:00: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 10 0D 02 4A 40 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 A9 03

Tx: 11:18:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 02 0C 01 02 4B 46 C3

Rx: 11:18:01: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 20 0D 02 4B 40 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20

03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20
03 20 03 20 03 20 03 20 03 20 03 20 03 20 03 20 05 16

Tx: 11:18:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 03 0C 01 02 4C 71 F4

Rx: 11:18:01: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 30 0D 02 4C 40 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 43 06

Tx: 11:18:01: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 04 0C 01 02 4D F4 65

Rx: 11:18:02: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 40 0D 02 4D 40 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 49 0C

Tx: 11:18:02: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks
AE 02 0B 01 01 05 0C 01 02 4E C3 56

Rx: 11:18:02: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 13 = Send Data Block
AE 01 8B 02 02 50 0D 02 4E 40 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21
03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 03 21 0B 0D

- **Bandwidth allocation scheme polling of event log data per remote site proposed in Chapter 2**
- **(Latency 11:51:28 to 11:52:08 = 40s)**
- **Polling Configuration: Maximum logs: 1000**

Tx: 11:51:28: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 09 22 00 00 88 72

Rx: 11:51:28: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 09 2C 3A 6D 00 61 01 9F 20 DE 54 05 02 01 3A 6D 00 61 03 92 20 DF 54 01 02 01 3A 6D 00 61 03 92 20 DF 54 02 02 01 3A 6D 00 61 03 92 20 DF 54 03 02 01 3A 6D 00 61 03 92 20 DF 54 04 02 01 3A 6D 00 61 03 93 20 DF 54 05 02 01 3A 6D 00 62 01 A3 20 E0 54 01 02 01 3A 6D 00 62 01 A4 20 E0 54 02 02 01 3A 6D 00 62 01 A4 20 E0 54 03 02 01 3A 6D 00 62 01 A4 20 E0 54 04 02 01 04 F8

To:

Tx: 11:52:07: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 02 56 00 00 99 D5

Rx: 11:52:08: reply: KF2 Message, Dest 1, Src 2, Via 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 02 60 3A 6D 00 C9 00 66 21 A4 54 05 02 01 3A 6D 00 C9 02 57 21 A5 54 01 02 01 3A 6D 00 C9 02 57 21 A5 54 02 02 01 3A 6D 00 C9 02 57 21 A5 54 03 02 01 3A 6D 00 C9 02 58 21 A5 54 04 02 01 3A 6D 00 C9 02 58 21 A5 54 05 02 01 3A 6D 00 CA 00 60 21 A6 54 01 02 01 3A 6D 00 CA 00 61 21 A6 54 02 02 01 3A 6D 00 CA 00 61 21 A6 54 03 02 01 3A 6D 00 CA 00 61 21 A6 54 04 02 01 8F D8

- **“System Status poll” as proposed in Chapter 4**

- **Periodic poll for critcal real time and alarm data**
- **Polling flag set every 5 minutes for system status update**

Tx: 14:20:03: init: KF2 Message, Dest 3, Src 1, Command# 26 = Request RTU update

AE 03 09 01 02 04 1A 03 16 70

- **Poll flag set after 5 minutes**

Rx: 14:25:25: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 A0 51 0E 00 0A 04 7C 4D EC DC DD 03 52 78 14 54 01 04 01 4D EC DC DD 03 52 78 14 54 02 04 01 4D EC DC DD 03 52 78 14 54 03 04 01 4D EC DC DD 03 52 78 14 54 04 04 01 4D EC DC DD 03 52 78 14 54 05 04 01 4D EC DC DE 01 72 78 15 54 01 04 01 4D EC DC DE 01 72 78 15 54 02 04 01 4D EC DC DE 01 72 78 15 54 03 04 01 4D EC DC DE 01 72 78 15 54 04 04 01 4D EC DC DE 01 72 78 15 54 05 04 01 39 6B

- **System waits for block transmission to complete (shown above #Command 81) before intiating poll request using the (rx_data) communication block and polls remotes sites 2 – 5**

Tx: 14:25:26: init: KF2 Message, Dest 2, Src 1, Command# 16 = Get Multi Data

AE 02 49 01 01 0D 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F 4D D5

Rx: 14:25:26: reply: KF2 Message, Dest 1, Src 2, Command# 17 = Send Multi Data

AE 01 89 02 02 D0 11 20 10 00 2F 0D 10 01 2F 0D 10 02 2F 0D 10 03 2F 0D 10 04 2F 0D 10 05 2F 0D 10 06 2F 0D 10 07 2F 0D 10 08 2F 0D 10 09 2F 0D 10 0A 2F 0D 10 0B 2F 0D 10 0C 2F 0D 10 0D 2F 0D 10 0E 2F 0D 10 0F 2F 0D 10 10 2F 0D 10 11 2F 0D 10 12 2F 0D 10 13 2F 0D 10 14 2F 0D 10 15 2F 0D 10 16 2F 0D 10 17 2F 0D 10 18 2F 0D 10 19 2F 0D 10 1A 2F 0D 10 1B 2F 0D 10 1C 2F 0D 10 1D 2F 0D 10 1E 2F 0D 10 1F 2F 0D 41 E1

Tx: 14:25:27: init: KF2 Message, Dest 3, Src 1, Command# 16 = Get Multi Data

AE 03 49 01 02 01 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F FF 7B

Rx: 14:25:28: reply: KF2 Message, Dest 1, Src 3, Command# 17 = Send Multi Data

AE 01 89 03 02 10 11 20 10 00 40 D5 10 01 40 D5 10 02 40 D5 10 03 40 D5 10 04 40 D5 10 05 40 D5 10 06 40 D5 10 07 40 D5 10 08 40 D5 10 09 40 D5 10 0A 40 D5 10 0B 40 D5 10 0C 40 D5 10 0D 40 D5 10 0E 40 D5 10 0F 40 D5 10 10 40 D5 10 11 40 D5 10 12 40 D5 10 13 40 D5 10 14 40 D5 10 15 40 D5 10 16 40 D5 10 17 40 D5 10 18 40 D5 10 19 40 D5 10 1A 40 D5 10 1B 40 D5 10 1C 40 D5 10 1D 40 D5 10 1E 40 D5 10 1F 40 D5 76 33

Tx: 14:25:28: init: KF2 Message, Dest 4, Src 1, Command# 16 = Get Multi Data

AE 04 49 01 02 0B 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F CB 88

Rx: 14:25:29: reply: KF2 Message, Dest 1, Src 4, Command# 17 = Send Multi Data

AE 01 89 04 02 B0 11 20 10 00 19 F4 10 01 19 F4 10 02 19 F4 10 03 19 F4 10 04 19 F4 10 05 19 F4 10 06 19 F4 10 07 19 F4 10 08 19 F4 10 09 19 F4 10 0A 19 F4 10 0B 19 F4 10 0C 19 F4 10 0D 19 F4 10 0E 19 F4 10 0F 19 F4 10 10 19 F4 10 11 19 F4 10 12 19 F4 10 13 19 F4 10 14 19 F4 10 15 19 F4 10 16 19 F4 10 17 19 F4 10 18 19 F4 10 19 19 F4 10 1A 19 F4 10 1B 19 F4 10 1C 19 F4 10 1D 19 F4 10 1E 19 F4 10 1F 19 F4 1D F5

Tx: 14:25:30: init: KF2 Message, Dest 5, Src 1, Command# 16 = Get Multi Data

AE 05 49 01 02 01 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F AF 0C

Rx: 14:25:31: reply: KF2 Message, Dest 1, Src 5, Command# 17 = Send Multi Data

AE 01 89 05 02 10 11 20 10 00 2D 31 10 01 2D 31 10 02 2D 31 10 03 2D 31 10 04 2D 31 10 05 2D 31 10 06 2D 31 10 07 2D 31 10 08 2D 31 10 09 2D 31 10 0A 2D 31 10 0B 2D 31 10 0C 2D 31 10 0D 2D 31 10 0E 2D 31 10 0F 2D 31 10 10 2D 31 10 11 2D 31 10 12 2D 31 10 13 2D 31 10 14 2D 31 10 15 2D 31 10 16 2D 31 10 17 2D 31 10 18 2D 31 10 19 2D 31 10 1A 2D 31 10 1B 2D 31 10 1C 2D 31 10 1D 2D 31 10 1E 2D 31 10 1F 2D 31 15 0F

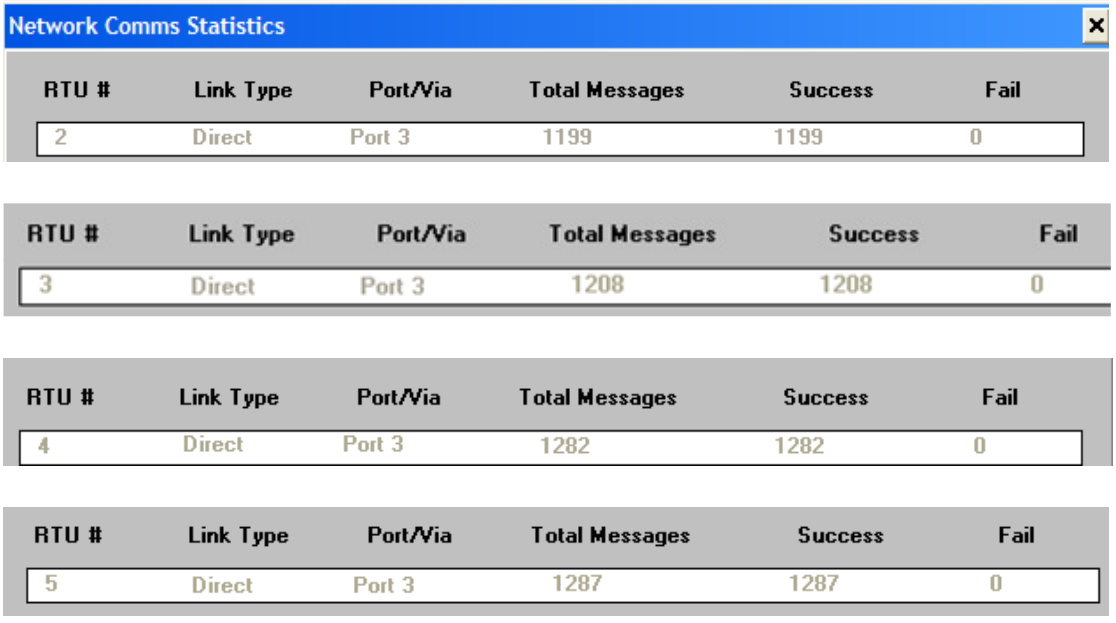
- **The end of the poll request for critical real time and alarm data**
- **System continues polling for non critical real time and event data (shown below)**

Tx: 14:25:31: init: KF2 Message, Dest 3, Src 1, Via 2, Command# 26 = Request RTU update

AE 03 09 01 02 02 1A 03 76 B6

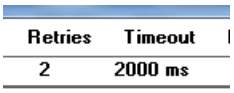
System Reliability – Network Communication Statistics

The network communication statistics within "Toolbox 32" software shows that the communication network is stable and extremely reliable which meets one of the objective criteria for the design.



Message Retries configuration

2 Message retries were configured under Network Link with timeout 2s



Retries	Timeout
2	2000 ms

Tx: 12:10:**08**: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 02 0C 01 02 4C 46 C4

Tx: 12:10**:10**: init: KF2 Message, Dest 2, Src 1, Via 1, Command# 12 = Get Data Blocks

AE 02 0B 01 01 02 0C 01 02 4C 46 C4

- **"Event Poll Mode" as proposed in Chapter 4**
- **Polls RTU 5 for (Type 1) event logs up to 200 logs**

Rx: 17:37:46: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 01 06 4D FE 32 DC 00 71 21 72 54 01 02 01 4D FE 32 DC 00 71 21 72 54 02 02 01 4D FE 32 DC 00 72 21 72 54 03 02 01 4D FE 32 DC 00 72 21 72 54 04 02 01 4D FE 32 DC 00 72 21 72 54 05 02 01 4D FE 32 DC 02 6B 21 73 54 01 02 01 4D FE 32 DC 02 6B 21 73 54 02 02 01 4D FE 32 DC 02 6C 21 73 54 03 02 01 4D FE 32 DC 02 6C 21 73 54 04 02 01 4D FE 32 DC 02 6C 21 73 54 05 02 01 BA A4

Tx: 17:37:47: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 07 50 04 00 0A 05 55 00 00 69 88

Rx: 17:37:47: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 70 51 0E 00 0A 05 5F 2D D6 F7 0C 02 EE 1C CC 54 01 05 01 2D D6 F7 0C 02 EE 1C CC 54 02 05 01 2D D6 F7 0C 02 EE 1C CC 54 03 05 01 2D D6 F7 0C 02 EE 1C CC 54 04 05 01 2D D6 F7 0C 02 F8 1C CC 54 05 05 01 2D D6 F7 0D 00 FA 1C CD 54 01 05 01 2D D6 F7 0D 00 FA 1C CD 54 02 05 01 2D D6 F7 0D 00 FA 1C CD 54 03 05 01 2D D6 F7 0D 00 FA 1C CD 54 04 05 01 2D D6 F7 0D 00 FA 1C CD 54 05 05 01 0D 31

Tx: 17:37:47: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 08 50 04 00 0A 05 5F 00 00 42 2B

Rx: 17:37:48: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 80 51 0E 00 0A 05 69 2D D6 F7 0D 02 EE 1C CE 54 01 05 01 2D D6 F7 0D 02 EE 1C CE 54 02 05 01 2D D6 F7 0D 02 EE 1C CE 54 03 05 01 2D D6 F7 0D 02 EE 1C CE 54 04 05 01 2D D6 F7 0D 02 F8 1C CE 54 05 05 01 2D D6 F7 0E 00 FA 1C CF 54 01 05 01 2D D6 F7 0E 00 FA 1C CF 54 02 05 01 2D D6 F7 0E 00 FA 1C CF 54 03 05 01 2D D6 F7 0E 00 FA 1C CF 54 04 05 01 2D D6 F7 0E 00 FA 1C CF 54 05 05 01 EE 01

Tx: 17:37:48: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 09 50 04 00 0A 05 69 00 00 AC DF

Rx: 17:37:48: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 90 51 0E 00 0A 05 73 2D D6 F7 0E 02 EE 1C D0 54 01 05 01 2D D6 F7 0E 02 EE 1C D0 54 02 05 01 2D D6 F7 0E 02 EE 1C D0 54 03 05 01 2D D6 F7 0E 02 EE 1C D0 54 04 05 01 2D D6 F7 0E 02 EE 1C D0 54 05 05 01 2D D6 F7 0F 00 FA 1C D1 54 01 05 01 2D D6 F7 0F 00 FA 1C D1 54 02 05 01 2D D6 F7 0F 00 FA 1C D1 54 03 05 01 2D D6 F7 0F 00 FA 1C D1 54 04 05 01 2D D6 F7 0F 00 FA 1C D1 54 05 05 01 3A 3E

Tx: 17:37:49: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0A 50 04 00 0A 05 73 00 00 C7 26

Rx: 17:37:49: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 A0 51 0E 00 0A 05 7D 2D D6 F7 0F 02 EE 1C D2 54 01 05 01 2D D6 F7 0F 02 EE 1C D2 54 02 05 01 2D D6 F7 0F 02 EE 1C D2 54 03 05 01 2D D6 F7 0F 02 EE 1C D2 54 04 05 01 2D D6 F7 0F 02 EE 1C D2 54 05 05 01 2D D6 F7 10 01 04 1C D3 54 01 05 01 2D D6 F7 10 01 04 1C D3 54 02 05 01 2D D6 F7 10 01 04 1C D3 54 03 05 01 2D D6 F7 10 01 04 1C D3 54 04 05 01 2D D6 F7 10 01 04 1C D3 54 05 05 01 8E 7E

Tx: 17:37:49: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0B 50 04 00 0A 05 7D 00 00 9E 89

Rx: 17:37:50: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 B0 51 0E 00 0A 05 87 2D D6 F7 10 02 F8 1C D4 54 01 05 01 2D D6 F7 10 02 F8 1C D4 54 02 05 01 2D D6 F7 10 02 F8 1C D4 54 03 05 01 2D D6 F7 10 02 F8 1C D4 54 04 05 01 2D D6 F7 10 02 F8 1C D4 54 05 05 01 2D D6 F7 11 01 5E 1C D5 54 01 05 01

2D D6 F7 11 01 5E 1C D5 54 02 05 01 2D D6 F7 11 01 5E 1C D5 54 03 05 01 2D D6 F7 11 01 5E 1C D5 54 04 05 01 2D D6 F7 11 01 5E 1C D5 54 05 05 01 2C F8

Tx: 17:37:50: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0C 50 04 00 0A 05 87 00 00 C9 98

Rx: 17:37:50: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 C0 51 0E 00 0A 05 91 2D D6 F7 11 03 52 1C D6 54 01 05 01 2D D6 F7 11 03 52 1C D6 54 02 05 01 2D D6 F7 11 03 52 1C D6 54 03 05 01 2D D6 F7 11 03 52 1C D6 54 04 05 01 2D D6 F7 11 03 52 1C D6 54 05 05 01 2D D6 F7 12 01 72 1C D7 54 01 05 01 2D D6 F7 12 01 72 1C D7 54 02 05 01 2D D6 F7 12 01 72 1C D7 54 03 05 01 2D D6 F7 12 01 72 1C D7 54 04 05 01 2D D6 F7 12 01 72 1C D7 54 05 05 01 0C BC

Tx: 17:37:51: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0D 50 04 00 0A 05 91 00 00 03 0E

Rx: 17:37:51: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 D0 51 0E 00 0A 05 9B 2D D6 F7 12 03 B6 1C D8 54 01 05 01 2D D6 F7 12 03 B6 1C D8 54 02 05 01 2D D6 F7 12 03 B6 1C D8 54 03 05 01 2D D6 F7 12 03 B6 1C D8 54 04 05 01 2D D6 F7 12 03 B6 1C D8 54 05 05 01 2D D6 F7 13 01 C2 1C D9 54 01 05 01 2D D6 F7 13 01 C2 1C D9 54 02 05 01 2D D6 F7 13 01 C2 1C D9 54 03 05 01 2D D6 F7 13 01 C2 1C D9 54 04 05 01 2D D6 F7 13 01 C2 1C D9 54 05 05 01 E7 2B

Tx: 17:37:51: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0E 50 04 00 0A 05 9B 00 00 7A C6

Rx: 17:37:52: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 E0 51 0E 00 0A 05 A5 2D D6 F7 13 03 B6 1C DA 54 01 05 01 2D D6 F7 13 03 B6 1C DA 54 02 05 01 2D D6 F7 13 03 C0 1C DA 54 03 05 01 2D D6 F7 13 03 C0 1C DA 54 04 05 01 2D D6 F7 13 03 C0 1C DA 54 05 05 01 2D D6 F7 14 01 C2 1C DB 54 01 05 01 2D D6 F7 14 01 C2 1C DB 54 02 05 01 2D D6 F7 14 01 C2 1C DB 54 03 05 01 2D D6 F7 14 01 C2 1C DB 54 04 05 01 2D D6 F7 14 01 C2 1C DB 54 05 05 01 4F 8A

Tx: 17:37:52: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0F 50 04 00 0A 05 A5 00 00 15 3A

Rx: 17:37:52: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 F0 51 0E 00 0A 05 AF 2D D6 F7 14 03 B6 1C DC 54 01 05 01 2D D6 F7 14 03 B6 1C DC 54 02 05 01 2D D6 F7 14 03 B6 1C DC 54 03 05 01 2D D6 F7 14 03 B6 1C DC 54 04 05 01 2D D6 F7 14 03 B6 1C DC 54 05 05 01 2D D6 F7 15 01 C2 1C DD 54 01 05 01 2D D6 F7 15 01 C2 1C DD 54 02 05 01 2D D6 F7 15 01 C2 1C DD 54 03 05 01 2D D6 F7 15 01 C2 1C DD 54 04 05 01 2D D6 F7 15 01 C2 1C DD 54 05 05 01 07 8A

Tx: 17:37:52: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 01 50 04 00 0A 05 AF 00 00 86 F8

Rx: 17:37:53: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 10 51 0E 00 0A 05 B9 2D D6 F7 15 03 B6 1C DE 54 01 05 01 2D D6 F7 15 03 C0 1C DE 54 02 05 01 2D D6 F7 15 03 C0 1C DE 54 03 05 01 2D D6 F7 15 03 C0 1C DE 54 04 05 01 2D D6 F7 15 03 C0 1C DE 54 05 05 01 2D D6 F7 16 01 CC 1C DF 54 01 05 01 2D D6 F7 16 01 CC 1C DF 54 02 05 01 2D D6 F7 16 01 CC 1C DF 54 03 05 01 2D D6 F7 16 01 CC 1C DF 54 04 05 01 2D D6 F7 16 01 CC 1C DF 54 05 05 01 FC C6

Tx: 17:37:53: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 02 50 04 00 0A 05 B9 00 00 2C 8D

Rx: 17:37:54: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 20 51 0E 00 0A 05 C3 2D D6 F7 16 03 C0 1C E0 54 01 05 01 2D D6 F7 16 03 C0 1C E0 54 02 05 01 2D D6 F7 16 03 C0 1C E0 54 03 05 01 2D D6 F7 16 03 C0 1C E0 54 04 05 01 2D D6 F7 16 03 C0 1C E0 54 05 05 01 2D D6 F7 17 02 26 1C E1 54 01 05 01 2D D6 F7 17 02 26 1C E1 54 02 05 01 2D D6 F7 17 02 26 1C E1 54 03 05 01 2D D6 F7 17 02 26 1C E1 54 04 05 01 2D D6 F7 17 02 26 1C E1 54 05 05 01 4A 86

Tx: 17:37:54: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 03 50 04 00 0A 05 C3 00 00 4B 31

Rx: 17:37:54: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 30 51 0E 00 0A 05 CD 2D D6 F7 18 00 3C 1C E2 54 01 05 01 2D D6 F7 18 00 3C 1C E2 54 02 05 01 2D D6 F7 18 00 46 1C E2 54 03 05 01 2D D6 F7 18 00 46 1C E2 54 04 05 01 2D D6 F7 18 00 46 1C E2 54 05 05 01 2D D6 F7 18 02 94 1C E3 54 01 05 01 2D D6 F7 18 02 94 1C E3 54 02 05 01 2D D6 F7 18 02 94 1C E3 54 03 05 01 2D D6 F7 18 02 94 1C E3 54 04 05 01 2D D6 F7 18 02 94 1C E3 54 05 05 01 15 6B

Tx: 17:37:54: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 04 50 04 00 0A 05 CD 00 00 B3 BB

Rx: 17:37:55: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 40 51 0E 00 0A 05 D7 2D D6 F7 19 01 0E 1C E4 54 01 05 01 2D D6 F7 19 01 0E 1C E4 54 02 05 01 2D D6 F7 19 01 0E 1C E4 54 03 05 01 2D D6 F7 19 01 0E 1C E4 54 04 05 01 2D D6 F7 19 01 0E 1C E4 54 05 05 01 2D D6 F7 19 03 66 1C E5 54 01 05 01 2D D6 F7 19 03 66 1C E5 54 02 05 01 2D D6 F7 19 03 66 1C E5 54 03 05 01 2D D6 F7 19 03 66 1C E5 54 04 05 01 2D D6 F7 19 03 66 1C E5 54 05 05 01 79 BE

Tx: 17:37:55: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 05 50 04 00 0A 05 D7 00 00 B8 A1

Rx: 17:37:55: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 50 51 0E 00 0A 05 E1 2D D6 F7 1A 01 CC 1C E6 54 01 05 01 2D D6 F7 1A 01 CC 1C E6 54 02 05 01 2D D6 F7 1A 01 D6 1C E6 54 03 05 01 2D D6 F7 1A 01 D6 1C E6 54 04 05 01 2D D6 F7 1A 01 D6 1C E6 54 05 05 01 2D D6 F7 1A 03 CA 1C E7 54 01 05 01 2D D6 F7 1A 03 CA 1C E7 54 02 05 01 2D D6 F7 1A 03 CA 1C E7 54 03 05 01 2D D6 F7 1A 03 CA 1C E7 54 04 05 01 2D D6 F7 1A 03 CA 1C E7 54 05 05 01 78 5F

Tx: 17:37:56: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 06 50 04 00 0A 05 E1 00 00 36 B6

Rx: 17:37:56: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 60 51 0E 00 0A 05 EB 2D D6 F7 1B 02 30 1C E8 54 01 05 01 2D D6 F7 1B 02 30 1C E8 54 02 05 01 2D D6 F7 1B 02 3A 1C E8 54 03 05 01 2D D6 F7 1B 02 3A 1C E8 54 04 05 01 2D D6 F7 1B 02 3A 1C E8 54 05 05 01 2D D6 F7 1C 00 3C 1C E9 54 01 05 01

2D D6 F7 1C 00 3C 1C E9 54 02 05 01 2D D6 F7 1C 00 46 1C E9 54 03 05 01 2D D6 F7 1C 00 46 1C E9 54 04 05 01 2D D6 F7 1C 00 46 1C E9 54 05 05 01 F5 64

Tx: 17:37:56: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 07 50 04 00 0A 05 EB 00 00 2F 9D

Rx: 17:37:57: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 70 51 0E 00 0A 05 F5 2D D6 F7 1C 02 30 1C EA 54 01 05 01 2D D6 F7 1C 02 30 1C EA 54 02 05 01 2D D6 F7 1C 02 30 1C EA 54 03 05 01 2D D6 F7 1C 02 30 1C EA 54 04 05 01 2D D6 F7 1C 02 30 1C EA 54 05 05 01 2D D6 F7 1D 00 46 1C EB 54 01 05 01 2D D6 F7 1D 00 46 1C EB 54 02 05 01 2D D6 F7 1D 00 46 1C EB 54 03 05 01 2D D6 F7 1D 00 46 1C EB 54 04 05 01 2D D6 F7 1D 00 46 1C EB 54 05 05 01 6D 0D

Tx: 17:37:57: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 08 50 04 00 0A 05 F5 00 00 56 8B

Rx: 17:37:57: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 80 51 0E 00 0A 05 FF 2D D6 F7 1D 02 3A 1C EC 54 01 05 01 2D D6 F7 1D 02 3A 1C EC 54 02 05 01 2D D6 F7 1D 02 3A 1C EC 54 03 05 01 2D D6 F7 1D 02 3A 1C EC 54 04 05 01 2D D6 F7 1D 02 3A 1C EC 54 05 05 01 2D D6 F7 1E 00 A0 1C ED 54 01 05 01 2D D6 F7 1E 00 A0 1C ED 54 02 05 01 2D D6 F7 1E 00 A0 1C ED 54 03 05 01 2D D6 F7 1E 00 A0 1C ED 54 04 05 01 2D D6 F7 1E 00 A0 1C ED 54 05 05 01 0B 5B

Tx: 17:37:58: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 09 50 04 00 0A 05 FF 00 00 4F A0

Rx: 17:37:58: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 90 51 0E 00 0A 06 09 2D D6 F7 1E 02 94 1C EE 54 01 05 01 2D D6 F7 1E 02 94 1C EE 54 02 05 01 2D D6 F7 1E 02 94 1C EE 54 03 05 01 2D D6 F7 1E 02 94 1C EE 54 04 05 01 2D D6 F7 1E 02 9E 1C EE 54 05 05 01 2D D6 F7 1F 00 A0 1C EF 54 01 05 01 2D D6 F7 1F 00 A0 1C EF 54 02 05 01 2D D6 F7 1F 00 A0 1C EF 54 03 05 01 2D D6 F7 1F 00 A0 1C EF 54 04 05 01 2D D6 F7 1F 00 A0 1C EF 54 05 05 01 80 93

Tx: 17:37:58: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0A 50 04 00 0A 06 09 00 00 4D A8

Rx: 17:37:59: reply: KF2 Message, Dest 1, Src , Command# 81 = Send Event Logs
AE 01 85 05 02 A0 51 0E 00 0A 06 13 2D D6 F7 1F 02 94 1C F0 54 01 05 01 2D D6 F7 1F 02 9E 1C F0 54 02 05 01 2D D6 F7 1F 02 9E 1C F0 54 03 05 01 2D D6 F7 1F 02 9E 1C F0 54 04 05 01 2D D6 F7 1F 02 9E 1C F0 54 05 05 01 2D D6 F7 20 00 A0 1C F1 54 01 05 01 2D D6 F7 20 00 AA 1C F1 54 02 05 01 2D D6 F7 20 00 AA 1C F1 54 03 05 01 2D D6 F7 20 00 AA 1C F1 54 04 05 01 2D D6 F7 20 00 AA 1C F1 54 05 05 01 25 0F

Tx: 17:37:59: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs
AE 05 0F 01 02 0B 50 04 00 0A 06 13 00 00 46 B2

Rx: 17:37:59: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs
AE 01 85 05 02 B0 51 0E 00 0A 06 1D 2D D6 F7 20 02 94 1C F2 54 01 05 01 2D D6 F7 20 02 94 1C F2 54 02 05 01 2D D6 F7 20 02 94 1C F2 54 03 05 01 2D D6 F7 20 02 94 1C F2 54 04 05 01 2D D6 F7 20 02 94 1C F2 54 05 05 01 2D D6 F7 21 00 AA 1C F3 54 01 05 01 2D D6 F7 21 00 AA 1C F3 54 02 05 01 2D D6 F7 21 00 AA 1C F3 54 03 05 01 2D D6 F7 21 00 AA 1C F3 54 04 05 01 2D D6 F7 21 00 AA 1C F3 54 05 05 01 3C 33

Tx: 17:38:01: init: KF2 Message, Dest 2, Src 1, Command# 16 = Get Multi Data
AE 02 49 01 01 0F 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F BC B3

- **"Event Poll Mode" as proposed in Chapter 4**
- **Poll RTU's 2-5 for (Type 1) event log data up to 200 logs each**

Tx: 17:32:11: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 08 86 00 00 F6 4C

Rx: 17:32:11: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 08 90 4D FE 31 67 00 AF 1E B0 54 01 02 01 4D FE 31 67 00 B0 1E B0 54 02 02 01 4D FE 31 67 00 B0 1E B0 54 03 02 01 4D FE 31 67 00 B0 1E B0 54 04 02 01 4D FE 31 67 00 B0 1E B0 54 05 02 01 4D FE 31 67 02 A9 1E B1 54 01 02 01 4D FE 31 67 02 AA 1E B1 54 02 02 01 4D FE 31 67 02 AA 1E B1 54 03 02 01 4D FE 31 67 02 AA 1E B1 54 04 02 01 4D FE 31 67 02 AA 1E B1 54 05 02 01 EA 2A

Tx: 17:32:11: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 08 90 00 00 5C 39

Rx: 17:32:11: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 08 9A 4D FE 31 68 00 BC 1E B2 54 01 02 01 4D FE 31 68 00 BC 1E B2 54 02 02 01 4D FE 31 68 00 BC 1E B2 54 03 02 01 4D FE 31 68 00 BD 1E B2 54 04 02 01 4D FE 31 68 00 BD 1E B2 54 05 02 01 4D FE 31 68 03 0D 1E B3 54 01 02 01 4D FE 31 68 03 0E 1E B3 54 02 02 01 4D FE 31 68 03 0E 1E B3 54 03 02 01 4D FE 31 68 03 0E 1E B3 54 04 02 01 4D FE 31 68 03 0F 1E B3 54 05 02 01 83 C2

Tx: 17:32:12: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 08 9A 00 00 45 12

Rx: 17:32:12: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 08 A4 4D FE 31 69 01 1D 1E B4 54 01 02 01 4D FE 31 69 01 1E 1E B4 54 02 02 01 4D FE 31 69 01 1E 1E B4 54 03 02 01 4D FE 31 69 01 1E 1E B4 54 04 02 01 4D FE 31 69 01 1E 1E B4 54 05 02 01 4D FE 31 69 03 0A 1E B5 54 01 02 01 4D FE 31 69 03 0A 1E B5 54 02 02 01 4D FE 31 69 03 0A 1E B5 54 03 02 01 4D FE 31 69 03 0B 1E B5 54 04 02 01 4D FE 31 69 03 0B 1E B5 54 05 02 01 B1 41

Tx: 17:32:12: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 08 A4 00 00 A0 07

Rx: 17:32:12: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 08 AE 4D FE 31 6A 01 1B 1E B6 54 01 02 01 4D FE 31 6A 01 1B 1E B6 54 02 02 01 4D FE 31 6A 01 1C 1E B6 54 03 02 01 4D FE 31 6A 01 1C 1E B6 54 04 02 01 4D FE 31 6A 01 1C 1E B6 54 05 02 01 4D FE 31 6A 03 0F 1E B7 54 01 02 01 4D FE 31 6A 03 0F 1E B7 54 02 02 01 4D FE 31 6A 03 10 1E B7 54 03 02 01 4D FE 31 6A 03 10 1E B7 54 04 02 01 4D FE 31 6A 03 10 1E B7 54 05 02 01 E7 9B

Tx: 17:32:12: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 08 AE 00 00 D9 CF

Rx: 17:32:13: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 08 B8 4D FE 31 6B 01 1E 1E B8 54 01 02 01 4D FE 31 6B 01 1E 1E B8 54 02 02 01 4D FE 31 6B 01 1E 1E B8 54 03 02 01 4D FE 31 6B 01 1F 1E B8 54 04 02 01 4D FE 31 6B 01 1F 1E B8 54 05 02 01 4D FE 31 6B 03 1F 1E B9 54 01 02 01 4D FE 31 6B 03 1F 1E B9 54 02 02 01 4D FE 31 6B 03 1F 1E B9 54 03 02 01 4D FE 31 6B 03 1F 1E B9 54 04 02 01 4D FE 31 6B 03 20 1E B9 54 05 02 01 9E 0C

Tx: 17:32:13: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 03 50 04 00 0A 08 B8 00 00 13 59

Rx: 17:32:13: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 30 51 0E 00 0A 08 C2 4D FE 31 6C 01 7C 1E BA 54 01 02 01 4D FE 31 6C 01 7C 1E BA 54 02 02 01 4D FE 31 6C 01 7D 1E BA 54 03 02 01 4D FE 31 6C 01 7D 1E BA 54 04 02 01 4D FE 31 6C 01 7D 1E BA 54 05 02 01 4D FE 31 6C 03 76 1E BB 54 01 02 01 4D FE 31 6C 03 76 1E BB 54 02 02 01

4D FE 31 6C 03 77 1E BB 54 03 02 01 4D FE 31 6C 03 77 1E BB 54 04 02 01 4D FE 31 6C 03 77 1E BB 54 05 02 01 82 22

Tx: 17:32:13: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 04 50 04 00 0A 08 C2 00 00 D5 C0

Rx: 17:32:13: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 40 51 0E 00 0A 08 CC 4D FE 31 6D 01 7A 1E BC 54 01 02 01 4D FE 31 6D 01 7B 1E BC 54 02 02 01 4D FE 31 6D 01 7B 1E BC 54 03 02 01 4D FE 31 6D 01 7B 1E BC 54 04 02 01 4D FE 31 6D 01 7B 1E BC 54 05 02 01 4D FE 31 6D 03 74 1E BD 54 01 02 01 4D FE 31 6D 03 74 1E BD 54 02 02 01 4D FE 31 6D 03 74 1E BD 54 03 02 01 4D FE 31 6D 03 74 1E BD 54 04 02 01 4D FE 31 6D 03 75 1E BD 54 05 02 01 E3 7C

Tx: 17:32:14: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 05 50 04 00 0A 08 CC 00 00 8C 6F

Rx: 17:32:14: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 50 51 0E 00 0A 08 D6 4D FE 31 6E 01 82 1E BE 54 01 02 01 4D FE 31 6E 01 82 1E BE 54 02 02 01 4D FE 31 6E 01 82 1E BE 54 03 02 01 4D FE 31 6E 01 83 1E BE 54 04 02 01 4D FE 31 6E 01 83 1E BE 54 05 02 01 4D FE 31 6E 03 74 1E BF 54 01 02 01 4D FE 31 6E 03 74 1E BF 54 02 02 01 4D FE 31 6E 03 74 1E BF 54 03 02 01 4D FE 31 6E 03 74 1E BF 54 04 02 01 4D FE 31 6E 03 75 1E BF 54 05 02 01 60 7C

Tx: 17:32:14: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 06 50 04 00 0A 08 D6 00 00 E7 96

Rx: 17:32:14: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 60 51 0E 00 0A 08 E0 4D FE 31 6F 01 E6 1E C0 54 01 02 01 4D FE 31 6F 01 E6 1E C0 54 02 02 01 4D FE 31 6F 01 E6 1E C0 54 03 02 01 4D FE 31 6F 01 E7 1E C0 54 04 02 01 4D FE 31 6F 01 E7 1E C0 54 05 02 01 4D FE 31 70 00 5C 1E C1 54 01 02 01 4D FE 31 70 00 5C 1E C1 54 02 02 01 4D FE 31 70 00 5D 1E C1 54 03 02 01 4D FE 31 70 00 5D 1E C1 54 04 02 01 4D FE 31 70 00 5D 1E C1 54 05 02 01 A9 98

Tx: 17:32:14: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 07 50 04 00 0A 08 E0 00 00 09 62

Rx: 17:32:15: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 70 51 0E 00 0A 08 EA 4D FE 31 70 02 AC 1E C2 54 01 02 01 4D FE 31 70 02 AD 1E C2 54 02 02 01 4D FE 31 70 02 AD 1E C2 54 03 02 01 4D FE 31 70 02 AD 1E C2 54 04 02 01 4D FE 31 70 02 AD 1E C2 54 05 02 01 4D FE 31 71 01 15 1E C3 54 01 02 01 4D FE 31 71 01 15 1E C3 54 02 02 01 4D FE 31 71 01 15 1E C3 54 03 02 01 4D FE 31 71 01 15 1E C3 54 04 02 01 4D FE 31 71 01 16 1E C3 54 05 02 01 F3 50

Tx: 17:32:15: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 08 50 04 00 0A 08 EA 00 00 22 C1

Rx: 17:32:15: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 80 51 0E 00 0A 08 F4 4D FE 31 71 03 0D 1E C4 54 01 02 01 4D FE 31 71 03 0D 1E C4 54 02 02 01 4D FE 31 71 03 0D 1E C4 54 03 02 01 4D FE 31 71 03 0D 1E C4 54 04 02 01 4D FE 31 71 03 0E 1E C4 54 05 02 01 4D FE 31 72 01 19 1E C5 54 01 02 01 4D FE 31 72 01 19 1E C5 54 02 02 01 4D FE 31 72 01 19 1E C5 54 03 02 01 4D FE 31 72 01 19 1E C5 54 04 02 01 4D FE 31 72 01 1A 1E C5 54 05 02 01 17 0E

Tx: 17:32:15: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 09 50 04 00 0A 08 F4 00 00 69 5F

Rx: 17:32:15: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 90 51 0E 00 0A 08 FE 4D FE 31 72 03 12 1E C6 54 01 02 01 4D FE 31 72 03 13 1E C6 54 02 02 01 4D FE 31 72 03 13 1E C6 54 03 02 01 4D FE 31 72 03 13 1E C6 54 04 02 01 4D FE 31 72 03 13 1E C6 54 05 02 01 4D FE 31 73 01 79 1E C7 54 01 02 01 4D FE 31 73 01 79 1E C7 54 02 02 01 4D FE 31 73 01 79 1E C7 54 03 02 01 4D FE 31 73 01 79 1E C7 54 04 02 01 4D FE 31 73 01 7A 1E C7 54 05 02 01 72 EE

Tx: 17:32:16: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0A 50 04 00 0A 08 FE 00 00 10 97

Rx: 17:32:16: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 A0 51 0E 00 0A 09 08 4D FE 31 73 03 72 1E C8 54 01 02 01 4D FE 31 73 03 73 1E C8 54 02 02 01 4D FE 31 73 03 73 1E C8 54 03 02 01 4D FE 31 73 03 73 1E C8 54 04 02 01 4D FE 31 73 03 74 1E C8 54 05 02 01 4D FE 31 74 01 82 1E C9 54 01 02 01 4D FE 31 74 01 83 1E C9 54 02 02 01 4D FE 31 74 01 83 1E C9 54 03 02 01 4D FE 31 74 01 83 1E C9 54 04 02 01 4D FE 31 74 01 83 1E C9 54 05 02 01 31 6C

Tx: 17:32:16: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0B 50 04 00 0A 09 08 00 00 14 1E

Rx: 17:32:16: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 B0 51 0E 00 0A 09 12 4D FE 31 74 03 76 1E CA 54 01 02 01 4D FE 31 74 03 77 1E CA 54 02 02 01 4D FE 31 74 03 77 1E CA 54 03 02 01 4D FE 31 74 03 77 1E CA 54 04 02 01 4D FE 31 74 03 77 1E CA 54 05 02 01 4D FE 31 75 01 83 1E CB 54 01 02 01 4D FE 31 75 01 83 1E CB 54 02 02 01 4D FE

31 75 01 84 1E CB 54 03 02 01 4D FE 31 75 01 84 1E CB 54 04 02 01 4D FE 31 75 01 84 1E CB 54 05 02 01 FE F6

Tx: 17:32:16: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0C 50 04 00 0A 09 12 00 00 BE 21

Rx: 17:32:17: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 C0 51 0E 00 0A 09 1C 4D FE 31 75 03 79 1E CC 54 01 02 01 4D FE 31 75 03 79 1E CC 54 02 02 01 4D FE 31 75 03 7A 1E CC 54 03 02 01 4D FE 31 75 03 7A 1E CC 54 04 02 01 4D FE 31 75 03 7B 1E CC 54 05 02 01 4D FE 31 76 01 81 1E CD 54 01 02 01 4D FE 31 76 01 81 1E CD 54 02 02 01 4D FE 31 76 01 82 1E CD 54 03 02 01 4D FE 31 76 01 82 1E CD 54 04 02 01 4D FE 31 76 01 82 1E CD 54 05 02 01 FF EF

Tx: 17:32:17: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0D 50 04 00 0A 09 1C 00 00 E7 8E

Rx: 17:32:17: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 D0 51 0E 00 0A 09 26 4D FE 31 76 03 D3 1E CE 54 01 02 01 4D FE 31 76 03 D3 1E CE 54 02 02 01 4D FE 31 76 03 D4 1E CE 54 03 02 01 4D FE 31 76 03 D5 1E CE 54 04 02 01 4D FE 31 76 03 D5 1E CE 54 05 02 01 4D FE 31 77 01 DD 1E CF 54 01 02 01 4D FE 31 77 01 DD 1E CF 54 02 02 01 4D FE 31 77 01 DE 1E CF 54 03 02 01 4D FE 31 77 01 DE 1E CF 54 04 02 01 4D FE 31 77 01 DE 1E CF 54 05 02 01 10 27

Tx: 17:32:17: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0E 50 04 00 0A 09 26 00 00 A8 15

Rx: 17:32:17: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 E0 51 0E 00 0A 09 30 4D FE 31 77 03 D3 1E D0 54 01 02 01 4D FE 31 77 03 D3 1E D0 54 02 02 01 4D FE 31 77 03 D3 1E D0 54 03 02 01 4D FE 31 77 03 D4 1E D0 54 04 02 01 4D FE 31 77 03 D5 1E D0 54 05 02 01 4D FE 31 78 01 DF 1E D1 54 01 02 01 4D FE 31 78 01 DF 1E D1 54 02 02 01 4D FE 31 78 01 DF 1E D1 54 03 02 01 4D FE 31 78 01 E0 1E D1 54 04 02 01 4D FE 31 78 01 E1 1E D1 54 05 02 01 23 95

Tx: 17:32:18: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 0F 50 04 00 0A 09 30 00 00 62 83

Rx: 17:32:18: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 F0 51 0E 00 0A 09 3A 4D FE 31 78 03 D9 1E D2 54 01 02 01 4D FE 31 78 03 D9 1E D2 54 02 02 01 4D FE 31 78 03 D9 1E D2 54 03 02 01 4D FE 31 78 03 DA 1E D2 54 04 02 01 4D FE 31 78 03 DA 1E D2 54 05 02 01 4D FE 31 79 01 E9 1E D3 54 01 02 01 4D FE 31 79 01 E9 1E D3 54 02 02 01 4D FE 31 79 01 E9 1E D3 54 03 02 01 4D FE 31 79 01 E9 1E D3 54 04 02 01 4D FE 31 79 01 EA 1E D3 54 05 02 01 FF EC

Tx: 17:32:18: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 01 50 04 00 0A 09 3A 00 00 F1 41

Rx: 17:32:18: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 10 51 0E 00 0A 09 44 4D FE 31 7A 00 59 1E D4 54 01 02 01 4D FE 31 7A 00 5A 1E D4 54 02 02 01 4D FE 31 7A 00 5A 1E D4 54 03 02 01 4D FE 31 7A 00 5B 1E D4 54 04 02 01 4D FE 31 7A 00 5B 1E D4 54 05 02 01 4D FE 31 7A 02 46 1E D5 54 01 02 01 4D FE 31 7A 02 46 1E D5 54 02 02 01 4D FE 31 7A 02 46 1E D5 54 03 02 01 4D FE 31 7A 02 47 1E D5 54 04 02 01 4D FE 31 7A 02 47 1E D5 54 05 02 01 EF 80

Tx: 17:32:18: init: KF2 Message, Dest 2, Src 1, Command# 80 = Request Event Logs

AE 02 0F 01 01 02 50 04 00 0A 09 44 00 00 B6 9A

Rx: 17:32:19: reply: KF2 Message, Dest 1, Src 2, Command# 81 = Send Event Logs

AE 01 85 02 02 20 51 0E 00 0A 09 4E 4D FE 31 7B 00 54 1E D6 54 01 02 01 4D FE 31 7B 00 54 1E D6 54 02 02 01 4D FE 31 7B 00 55 1E D6 54 03 02 01 4D FE 31 7B 00 55 1E D6 54 04 02 01 4D FE 31 7B 00 55 1E D6 54 05 02 01 4D FE 31 7B 02 45 1E D7 54 01 02 01 4D FE 31 7B 02 46 1E D7 54 02 02 01 4D FE 31 7B 02 46 1E D7 54 03 02 01 4D FE 31 7B 02 46 1E D7 54 04 02 01 4D FE 31 7B 02 46 1E D7 54 05 02 01 DE 78

Tx: 17:32:19: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0F 50 04 00 0A 02 9C 00 00 DA 23

Rx: 17:32:19: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 F0 51 0E 00 0A 02 A6 4D FE 31 74 01 45 5D BE 54 01 03 01 4D FE 31 74 01 45 5D BE 54 02 03 01 4D FE 31 74 01 46 5D BE 54 03 03 01 4D FE 31 74 01 46 5D BE 54 04 03 01 4D FE 31 74 01 46 5D BE 54 05 03 01 4D FE 31 74 01 47 5D BF 54 01 03 01 4D FE 31 74 01 47 5D BF 54 02 03 01 4D FE 31 74 01 47 5D BF 54 03 03 01 4D FE 31 74 01 47 5D BF 54 04 03 01 4D FE 31 74 01 48 5D BF 54 05 03 01 B4 2E

Tx: 17:32:19: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 01 50 04 00 0A 02 A6 00 00 7F B2

Rx: 17:32:20: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 10 51 0E 00 0A 02 B0 4D FE 31 74 03 3D 5D C0 54 01 03 01 4D FE 31 74 03 3D 5D C0 54 02 03 01 4D FE 31 74 03 3E 5D C0 54 03 03 01 4D FE 31 74 03 3E 5D C0 54 04 03 01 4D FE 31 74 03 3E 5D C0 54 05 03 01 4D FE 31 74 03 3E 5D C1 54 01 03 01 4D FE 31 74 03 3F 5D C1 54 02 03 01

4D FE 31 74 03 3F 5D C1 54 03 03 01 4D FE 31 74 03 40 5D C1 54 04 03 01 4D FE 31 74 03 40 5D C1 54 05 03 01 E1 83

Tx: 17:32:20: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 02 50 04 00 0A 02 B0 00 00 D5 C7

Rx: 17:32:20: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 20 51 0E 00 0A 02 BA 4D FE 31 75 01 49 5D C2 54 01 03 01 4D FE 31 75 01 49 5D C2 54 02 03 01 4D FE 31 75 01 4A 5D C2 54 03 03 01 4D FE 31 75 01 4A 5D C2 54 04 03 01 4D FE 31 75 01 4A 5D C2 54 05 03 01 4D FE 31 75 01 4A 5D C3 54 01 03 01 4D FE 31 75 01 4B 5D C3 54 02 03 01 4D FE 31 75 01 4B 5D C3 54 03 03 01 4D FE 31 75 01 4C 5D C3 54 04 03 01 4D FE 31 75 01 4C 5D C3 54 05 03 01 02 D7

Tx: 17:32:21: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 03 50 04 00 0A 02 BA 00 00 CC EC

Rx: 17:32:21: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 30 51 0E 00 0A 02 C4 4D FE 31 75 03 3B 5D C4 54 01 03 01 4D FE 31 75 03 3B 5D C4 54 02 03 01 4D FE 31 75 03 3B 5D C4 54 03 03 01 4D FE 31 75 03 3C 5D C4 54 04 03 01 4D FE 31 75 03 3C 5D C4 54 05 03 01 4D FE 31 75 03 3C 5D C5 54 01 03 01 4D FE 31 75 03 3C 5D C5 54 02 03 01 4D FE 31 75 03 3D 5D C5 54 03 03 01 4D FE 31 75 03 3D 5D C5 54 04 03 01 4D FE 31 75 03 3D 5D C5 54 05 03 01 D3 0B

Tx: 17:32:21: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 04 50 04 00 0A 02 C4 00 00 4A F1

Rx: 17:32:22: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 40 51 0E 00 0A 02 CE 4D FE 31 76 01 47 5D C6 54 01 03 01 4D FE 31 76 01 47 5D C6 54 02 03 01 4D FE 31 76 01 47 5D C6 54 03 03 01 4D FE 31 76 01 48 5D C6 54 04 03 01 4D FE 31 76 01 48 5D C6 54 05 03 01 4D FE 31 76 01 48 5D C7 54 01 03 01 4D FE 31 76 01 48 5D C7 54 02 03 01 4D FE 31 76 01 49 5D C7 54 03 03 01 4D FE 31 76 01 49 5D C7 54 04 03 01 4D FE 31 76 01 49 5D C7 54 05 03 01 B1 ED

Tx: 17:32:22: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 05 50 04 00 0A 02 CE 00 00 53 DA

Rx: 17:32:22: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 50 51 0E 00 0A 02 D8 4D FE 31 76 03 3B 5D C8 54 01 03 01 4D FE 31 76 03 3B 5D C8 54 02 03 01 4D FE 31 76 03 3B 5D C8 54 03 03 01 4D FE 31 76 03 3B 5D C8 54 04 03 01 4D FE 31 76 03 3C 5D C8 54 05 03 01 4D FE 31 76 03 3C 5D C9 54 01 03 01 4D FE 31 76 03 3C 5D C9 54 02 03 01 4D FE 31 76 03 3D 5D C9 54 03 03 01 4D FE 31 76 03 3D 5D C9 54 04 03 01 4D FE 31 76 03 3D 5D C9 54 05 03 01 D2 05

Tx: 17:32:22: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 06 50 04 00 0A 02 D8 00 00 F9 AF

Rx: 17:32:23: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 60 51 0E 00 0A 02 E2 4D FE 31 77 01 44 5D CA 54 01 03 01 4D FE 31 77 01 44 5D CA 54 02 03 01 4D FE 31 77 01 44 5D CA 54 03 03 01 4D FE 31 77 01 44 5D CA 54 04 03 01 4D FE 31 77 01 45 5D CA 54 05 03 01 4D FE 31 77 01 45 5D CB 54 01 03 01 4D FE 31 77 01 45 5D CB 54 02 03 01 4D FE 31 77 01 45 5D CB 54 03 03 01 4D FE 31 77 01 46 5D CB 54 04 03 01 4D FE 31 77 01 46 5D CB 54 05 03 01 FD 8E

Tx: 17:32:23: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 07 50 04 00 0A 02 E2 00 00 D6 D7

Rx: 17:32:23: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 70 51 0E 00 0A 02 EC 4D FE 31 77 03 3B 5D CC 54 01 03 01 4D FE 31 77 03 3C 5D CC 54 02 03 01 4D FE 31 77 03 3C 5D CC 54 03 03 01 4D FE 31 77 03 3C 5D CC 54 04 03 01 4D FE 31 77 03 3C 5D CC 54 05 03 01 4D FE 31 77 03 3D 5D CD 54 01 03 01 4D FE 31 77 03 3D 5D CD 54 02 03 01 4D FE 31 77 03 3D 5D CD 54 03 03 01 4D FE 31 77 03 3D 5D CD 54 04 03 01 4D FE 31 77 03 3E 5D CD 54 05 03 01 9B 4B

Tx: 17:32:24: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 08 50 04 00 0A 02 EC 00 00 BD F0

Rx: 17:32:24: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 80 51 0E 00 0A 02 F6 4D FE 31 78 01 47 5D CE 54 01 03 01 4D FE 31 78 01 48 5D CE 54 02 03 01 4D FE 31 78 01 48 5D CE 54 03 03 01 4D FE 31 78 01 48 5D CE 54 04 03 01 4D FE 31 78 01 48 5D CE 54 05 03 01 4D FE 31 78 01 49 5D CF 54 01 03 01 4D FE 31 78 01 49 5D CF 54 02 03 01 4D FE 31 78 01 49 5D CF 54 03 03 01 4D FE 31 78 01 49 5D CF 54 04 03 01 4D FE 31 78 01 4A 5D CF 54 05 03 01 21 FD

Tx: 17:32:24: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 09 50 04 00 0A 02 F6 00 00 B6 EA

Rx: 17:32:25: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 90 51 0E 00 0A 03 00 4D FE 31 78 03 39 5D D0 54 01 03 01 4D FE 31 78 03 39 5D D0 54 02 03 01 4D FE 31 78 03 3A 5D D0 54 03 03 01 4D FE 31 78 03 3A 5D D0 54 04 03 01 4D FE 31 78 03 3A 5D D0 54 05 03 01 4D FE 31 78 03 3B 5D D1 54 01 03 01 4D FE 31 78 03 3B 5D D1 54 02 03 01

4D FE 31 78 03 3B 5D D1 54 03 03 01 4D FE 31 78 03 3B 5D D1 54 04 03 01 4D FE 31 78 03 3C 5D D1 54 05 03 01 98 19

Tx: 17:32:25: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0A 50 04 00 0A 03 00 00 00 D2 80

Rx: 17:32:25: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 A0 51 0E 00 0A 03 0A 4D FE 31 79 01 48 5D D2 54 01 03 01 4D FE 31 79 01 48 5D D2 54 02 03 01 4D FE 31 79 01 49 5D D2 54 03 03 01 4D FE 31 79 01 49 5D D2 54 04 03 01 4D FE 31 79 01 49 5D D2 54 05 03 01 4D FE 31 79 01 49 5D D3 54 01 03 01 4D FE 31 79 01 4A 5D D3 54 02 03 01 4D FE 31 79 01 4A 5D D3 54 03 03 01 4D FE 31 79 01 4A 5D D3 54 04 03 01 4D FE 31 79 01 4A 5D D3 54 05 03 01 A0 41

Tx: 17:32:25: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0B 50 04 00 0A 03 0A 00 00 CB AB

Rx: 17:32:26: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 B0 51 0E 00 0A 03 14 4D FE 31 79 03 41 5D D4 54 01 03 01 4D FE 31 79 03 41 5D D4 54 02 03 01 4D FE 31 79 03 41 5D D4 54 03 03 01 4D FE 31 79 03 41 5D D4 54 04 03 01 4D FE 31 79 03 42 5D D4 54 05 03 01 4D FE 31 79 03 42 5D D5 54 01 03 01 4D FE 31 79 03 42 5D D5 54 02 03 01 4D FE 31 79 03 43 5D D5 54 03 03 01 4D FE 31 79 03 43 5D D5 54 04 03 01 4D FE 31 79 03 43 5D D5 54 05 03 01 75 BF

Tx: 17:32:26: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0C 50 04 00 0A 03 14 00 00 21 10

Rx: 17:32:26: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 C0 51 0E 00 0A 03 1E 4D FE 31 7A 01 AC 5D D6 54 01 03 01 4D FE 31 7A 01 AC 5D D6 54 02 03 01 4D FE 31 7A 01 AD 5D D6 54 03 03 01 4D FE 31 7A 01 AD 5D D6 54 04 03 01 4D FE 31 7A 01 AD 5D D6 54 05 03 01 4D FE 31 7A 01 AD 5D D7 54 01 03 01 4D FE 31 7A 01 AE 5D D7 54 02 03 01 4D FE 31 7A 01 AE 5D D7 54 03 03 01 4D FE 31 7A 01 AE 5D D7 54 04 03 01 4D FE 31 7A 01 AE 5D D7 54 05 03 01 55 8B

Tx: 17:32:26: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0D 50 04 00 0A 03 1E 00 00 38 3B

Rx: 17:32:27: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 D0 51 0E 00 0A 03 28 4D FE 31 7A 03 A5 5D D8 54 01 03 01 4D FE 31 7A 03 A5 5D D8 54 02 03 01 4D FE 31 7A 03 A5 5D D8 54 03 03 01 4D FE 31 7A 03 A5 5D D8 54 04 03 01 4D FE 31 7A 03 A6 5D D8 54 05 03 01 4D FE 31 7A 03 A6 5D D9 54 01 03 01 4D FE 31 7A 03 A6 5D D9 54 02 03 01 4D FE 31 7A 03 A7 5D D9 54 03 03 01 4D FE 31 7A 03 A7 5D D9 54 04 03 01 4D FE 31 7A 03 A7 5D D9 54 05 03 01 89 1F

Tx: 17:32:27: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0E 50 04 00 0A 03 28 00 00 B6 2C

Rx: 17:32:28: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 E0 51 0E 00 0A 03 32 4D FE 31 7B 02 16 5D DA 54 01 03 01 4D FE 31 7B 02 17 5D DA 54 02 03 01 4D FE 31 7B 02 17 5D DA 54 03 03 01 4D FE 31 7B 02 17 5D DA 54 04 03 01 4D FE 31 7B 02 17 5D DA 54 05 03 01 4D FE 31 7B 02 18 5D DB 54 01 03 01 4D FE 31 7B 02 18 5D DB 54 02 03 01 4D FE 31 7B 02 18 5D DB 54 03 03 01 4D FE 31 7B 02 18 5D DB 54 04 03 01 4D FE 31 7B 02 19 5D DB 54 05 03 01 E4 D7

Tx: 17:32:28: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 0F 50 04 00 0A 03 32 00 00 BD 36

Rx: 17:32:28: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 F0 51 0E 00 0A 03 3C 4D FE 31 7C 00 81 5D DC 54 01 03 01 4D FE 31 7C 00 81 5D DC 54 02 03 01 4D FE 31 7C 00 81 5D DC 54 03 03 01 4D FE 31 7C 00 81 5D DC 54 04 03 01 4D FE 31 7C 00 82 5D DC 54 05 03 01 4D FE 31 7C 00 82 5D DD 54 01 03 01 4D FE 31 7C 00 82 5D DD 54 02 03 01 4D FE 31 7C 00 82 5D DD 54 03 03 01 4D FE 31 7C 00 83 5D DD 54 04 03 01 4D FE 31 7C 00 83 5D DD 54 05 03 01 E4 F6

Tx: 17:32:28: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 01 50 04 00 0A 03 3C 00 00 6E 70

Rx: 17:32:29: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 10 51 0E 00 0A 03 46 4D FE 31 7C 02 75 5D DE 54 01 03 01 4D FE 31 7C 02 75 5D DE 54 02 03 01 4D FE 31 7C 02 75 5D DE 54 03 03 01 4D FE 31 7C 02 75 5D DE 54 04 03 01 4D FE 31 7C 02 76 5D DE 54 05 03 01 4D FE 31 7C 02 76 5D DF 54 01 03 01 4D FE 31 7C 02 76 5D DF 54 02 03 01 4D FE 31 7C 02 76 5D DF 54 03 03 01 4D FE 31 7C 02 77 5D DF 54 04 03 01 4D FE 31 7C 02 77 5D DF 54 05 03 01 71 17

Tx: 17:32:29: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 02 50 04 00 0A 03 46 00 00 69 2F

Rx: 17:32:29: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 20 51 0E 00 0A 03 50 4D FE 31 7D 00 7E 5D E0 54 01 03 01 4D FE 31 7D 00 7F 5D E0 54 02 03 01 4D FE 31 7D 00 7F 5D E0 54 03 03 01 4D FE 31 7D 00 7F 5D E0 54 04 03 01 4D FE 31 7D 00 7F 5D E0 54 05 03 01 4D FE 31 7D 00 80 5D E1 54 01 03 01 4D FE 31 7D 00 80 5D E1 54 02 03 01

4D FE 31 7D 00 80 5D E1 54 03 03 01 4D FE 31 7D 00 80 5D E1 54 04 03 01 4D FE 31 7D 00 81 5D E1 54 05 03 01 55 46

Tx: 17:32:29: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 03 50 04 00 0A 03 50 00 00 A3 B9

Rx: 17:32:30: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 30 51 0E 00 0A 03 5A 4D FE 31 7D 02 72 5D E2 54 01 03 01 4D FE 31 7D 02 73 5D E2 54 02 03 01 4D FE 31 7D 02 73 5D E2 54 03 03 01 4D FE 31 7D 02 73 5D E2 54 04 03 01 4D FE 31 7D 02 73 5D E2 54 05 03 01 4D FE 31 7D 02 74 5D E3 54 01 03 01 4D FE 31 7D 02 74 5D E3 54 02 03 01 4D FE 31 7D 02 74 5D E3 54 03 03 01 4D FE 31 7D 02 74 5D E3 54 04 03 01 4D FE 31 7D 02 75 5D E3 54 05 03 01 B4 8C

Tx: 17:32:30: init: KF2 Message, Dest 3, Src 1, Command# 80 = Request Event Logs

AE 03 0F 01 02 04 50 04 00 0A 03 5A 00 00 1B B7

Rx: 17:32:30: reply: KF2 Message, Dest 1, Src 3, Command# 81 = Send Event Logs

AE 01 85 03 02 40 51 0E 00 0A 03 64 4D FE 31 7E 00 86 5D E4 54 01 03 01 4D FE 31 7E 00 86 5D E4 54 02 03 01 4D FE 31 7E 00 87 5D E4 54 03 03 01 4D FE 31 7E 00 87 5D E4 54 04 03 01 4D FE 31 7E 00 87 5D E4 54 05 03 01 4D FE 31 7E 00 87 5D E5 54 01 03 01 4D FE 31 7E 00 88 5D E5 54 02 03 01 4D FE 31 7E 00 88 5D E5 54 03 03 01 4D FE 31 7E 00 88 5D E5 54 04 03 01 4D FE 31 7E 00 88 5D E5 54 05 03 01 24 12

Tx: 17:32:31: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0B 50 04 00 0A 05 35 00 00 8F 0C

Rx: 17:32:31: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 B0 51 0E 00 0A 05 3F 4D FE 2C 80 02 D0 F5 E1 54 01 04 01 4D FE 2C 80 02 D0 F5 E1 54 02 04 01 4D FE 2C 80 02 D0 F5 E1 54 03 04 01 4D FE 2C 80 02 D0 F5 E1 54 04 04 01 4D FE 2C 80 02 D0 F5 E1 54 05 04 01 4D FE 2C 81 00 DC F5 E2 54 01 04 01 4D FE 2C 81 00 DC F5 E2 54 02 04 01 4D FE 2C 81 00 DC F5 E2 54 03 04 01 4D FE 2C 81 00 DC F5 E2 54 04 04 01 4D FE 2C 81 00 DC F5 E2 54 05 04 01 A2 D1

Tx: 17:32:31: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0C 50 04 00 0A 05 3F 00 00 37 02

Rx: 17:32:32: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 C0 51 0E 00 0A 05 49 4D FE 2C 81 02 D0 F5 E3 54 01 04 01 4D FE 2C 81 02 D0 F5 E3 54 02 04 01 4D FE 2C 81 02 D0 F5 E3 54 03 04 01 4D FE 2C 81 02 D0 F5 E3 54 04 04 01 4D FE 2C 81 02 DA F5 E3 54 05 04 01 4D FE 2C 82 00 DC F5 E4 54 01 04 01 4D FE 2C 82 00 DC F5 E4 54 02 04 01 4D FE 2C 82 00 E6 F5 E4 54 03 04 01 4D FE 2C 82 00 E6 F5 E4 54 04 04 01 4D FE 2C 82 00 E6 F5 E4 54 05 04 01 D1 26

Tx: 17:32:32: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0D 50 04 00 0A 05 49 00 00 91 32

Rx: 17:32:32: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 D0 51 0E 00 0A 05 53 4D FE 2C 82 02 D0 F5 E5 54 01 04 01 4D FE 2C 82 02 D0 F5 E5 54 02 04 01 4D FE 2C 82 02 D0 F5 E5 54 03 04 01 4D FE 2C 82 02 D0 F5 E5 54 04 04 01 4D FE 2C 82 02 D0 F5 E5 54 05 04 01 4D FE 2C 83 00 F0 F5 E6 54 01 04 01 4D FE 2C 83 00 F0 F5 E6 54 02 04 01 4D FE

2C 83 00 F0 F5 E6 54 03 04 01 4D FE 2C 83 00 F0 F5 E6 54 04 04 01 4D FE 2C 83 00 F0 F5 E6 54 05 04 01 CC D2

Tx: 17:32:33: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0E 50 04 00 0A 05 53 00 00 FA CB

Rx: 17:32:33: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 E0 51 0E 00 0A 05 5D 4D FE 2C 83 03 3E F5 E7 54 01 04 01 4D FE 2C 83 03 3E F5 E7 54 02 04 01 4D FE 2C 83 03 3E F5 E7 54 03 04 01 4D FE 2C 83 03 3E F5 E7 54 04 04 01 4D FE 2C 83 03 3E F5 E7 54 05 04 01 4D FE 2C 84 01 5E F5 E8 54 01 04 01 4D FE 2C 84 01 5E F5 E8 54 02 04 01 4D FE 2C 84 01 5E F5 E8 54 03 04 01 4D FE 2C 84 01 5E F5 E8 54 04 04 01 4D FE 2C 84 01 5E F5 E8 54 05 04 01 53 D7

Tx: 17:32:33: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0F 50 04 00 0A 05 5D 00 00 A3 64

Rx: 17:32:34: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 F0 51 0E 00 0A 05 67 4D FE 2C 84 03 52 F5 E9 54 01 04 01 4D FE 2C 84 03 52 F5 E9 54 02 04 01 4D FE 2C 84 03 52 F5 E9 54 03 04 01 4D FE 2C 84 03 52 F5 E9 54 04 04 01 4D FE 2C 84 03 52 F5 E9 54 05 04 01 4D FE 2C 85 01 72 F5 EA 54 01 04 01 4D FE 2C 85 01 72 F5 EA 54 02 04 01 4D FE 2C 85 01 72 F5 EA 54 03 04 01 4D FE 2C 85 01 72 F5 EA 54 04 04 01 4D FE 2C 85 01 72 F5 EA 54 05 04 01 0D F4

Tx: 17:32:34: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 01 50 04 00 0A 05 67 00 00 06 F5

Rx: 17:32:34: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 10 51 0E 00 0A 05 71 4D FE 2C 85 03 C0 F5 EB 54 01 04 01 4D FE 2C 85 03 C0 F5 EB 54 02 04 01 4D FE 2C 85 03 C0 F5 EB 54 03 04 01 4D FE 2C 85 03 C0 F5 EB 54 04 04 01 4D FE 2C 85 03 C0 F5 EB 54 05 04 01 4D FE 2C 86 01 CC F5 EC 54 01 04 01 4D FE 2C 86 01 D6 F5 EC 54 02 04 01 4D FE 2C 86 01 D6 F5 EC 54 03 04 01 4D FE 2C 86 01 D6 F5 EC 54 04 04 01 4D FE 2C 86 01 D6 F5 EC 54 05 04 01 40 05

Tx: 17:32:35: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 02 50 04 00 0A 05 71 00 00 AC 80

Rx: 17:32:35: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 20 51 0E 00 0A 05 7B 4D FE 2C 86 03 CA F5 ED 54 01 04 01 4D FE 2C 86 03 CA F5 ED 54 02 04 01 4D FE 2C 86 03 CA F5 ED 54 03 04 01 4D FE 2C 86 03 CA F5 ED 54 04 04 01 4D FE 2C 86 03 CA F5 ED 54 05 04 01 4D FE 2C 87 01 E0 F5 EE 54 01 04 01 4D FE 2C 87 01 E0 F5 EE 54 02 04 01 4D FE 2C 87 01 E0 F5 EE 54 03 04 01 4D FE 2C 87 01 E0 F5 EE 54 04 04 01 4D FE 2C 87 01 E0 F5 EE 54 05 04 01 76 E5

Tx: 17:32:35: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 03 50 04 00 0A 05 7B 00 00 B5 AB

Rx: 17:32:36: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 30 51 0E 00 0A 05 85 4D FE 2C 88 00 5A F5 EF 54 01 04 01 4D FE 2C 88 00 5A F5 EF 54 02 04 01 4D FE 2C 88 00 5A F5 EF 54 03 04 01 4D FE 2C 88 00 64 F5 EF 54 04 04 01 4D FE 2C 88 00 64 F5 EF 54 05 04 01 4D FE 2C 88 02 4E F5 F0 54 01 04 01 4D FE 2C 88 02 4E F5 F0 54 02 04 01 4D FE 2C 88 02 4E F5 F0 54 03 04 01 4D FE 2C 88 02 4E F5 F0 54 04 04 01 4D FE 2C 88 02 4E F5 F0 54 05 04 01 76 42

Tx: 17:32:36: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 04 50 04 00 0A 05 85 00 00 A2 3E

Rx: 17:32:36: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 40 51 0E 00 0A 05 8F 4D FE 2C 89 00 BE F5 F1 54 01 04 01 4D FE 2C 89 00 BE F5 F1 54 02 04 01 4D FE 2C 89 00 BE F5 F1 54 03 04 01 4D FE 2C 89 00 BE F5 F1 54 04 04 01 4D FE 2C 89 00 BE F5 F1 54 05 04 01 4D FE 2C 89 02 B2 F5 F2 54 01 04 01 4D FE 2C 89 02 B2 F5 F2 54 02 04 01 4D FE 2C 89 02 B2 F5 F2 54 03 04 01 4D FE 2C 89 02 B2 F5 F2 54 04 04 01 4D FE 2C 89 02 BC F5 F2 54 05 04 01 FE 13

Tx: 17:32:37: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 05 50 04 00 0A 05 8F 00 00 BB 15

Rx: 17:32:37: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 50 51 0E 00 0A 05 99 4D FE 2C 8A 00 BE F5 F3 54 01 04 01 4D FE 2C 8A 00 BE F5 F3 54 02 04 01 4D FE 2C 8A 00 BE F5 F3 54 03 04 01 4D FE 2C 8A 00 BE F5 F3 54 04 04 01 4D FE 2C 8A 00 BE F5 F3 54 05 04 01 4D FE 2C 8A 02 B2 F5 F4 54 01 04 01 4D FE 2C 8A 02 B2 F5 F4 54 02 04 01 4D FE 2C 8A 02 B2 F5 F4 54 03 04 01 4D FE 2C 8A 02 B2 F5 F4 54 04 04 01 4D FE 2C 8A 02 B2 F5 F4 54 05 04 01 CB A1

Tx: 17:32:37: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 06 50 04 00 0A 05 99 00 00 11 60

Rx: 17:32:38: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 60 51 0E 00 0A 05 A3 4D FE 2C 8B 00 C8 F5 F5 54 01 04 01 4D FE 2C 8B 00 C8 F5 F5 54 02 04 01 4D FE 2C 8B 00 C8 F5 F5 54 03 04 01 4D FE 2C 8B 00 C8 F5 F5 54 04 04 01 4D FE 2C 8B 00 D2 F5 F5 54 05 04 01 4D FE 2C 8B 02 BC F5 F6 54 01 04 01 4D FE 2C 8B 02 C6 F5 F6 54 02 04 01 4D FE 2C

8B 02 C6 F5 F6 54 03 04 01 4D FE 2C 8B 02 C6 F5 F6 54 04 04 01 4D FE 2C 8B 02 C6 F5 F6 54 05 04 01 42 72

Tx: 17:32:38: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 07 50 04 00 0A 05 A3 00 00 3E 18

Rx: 17:32:38: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 70 51 0E 00 0A 05 AD 4D FE 2C 8C 00 D2 F5 F7 54 01 04 01 4D FE 2C 8C 00 D2 F5 F7 54 02 04 01 4D FE 2C 8C 00 DC F5 F7 54 03 04 01 4D FE 2C 8C 00 DC F5 F7 54 04 04 01 4D FE 2C 8C 00 DC F5 F7 54 05 04 01 4D FE 2C 8C 02 D0 F5 F8 54 01 04 01 4D FE 2C 8C 02 D0 F5 F8 54 02 04 01 4D FE 2C 8C 02 D0 F5 F8 54 03 04 01 4D FE 2C 8C 02 D0 F5 F8 54 04 04 01 4D FE 2C 8C 02 D0 F5 F8 54 05 04 01 C8 2D

Tx: 17:32:39: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 08 50 04 00 0A 05 AD 00 00 55 3F

Rx: 17:32:39: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 80 51 0E 00 0A 05 B7 4D FE 2C 8D 00 DC F5 F9 54 01 04 01 4D FE 2C 8D 00 DC F5 F9 54 02 04 01 4D FE 2C 8D 00 DC F5 F9 54 03 04 01 4D FE 2C 8D 00 DC F5 F9 54 04 04 01 4D FE 2C 8D 00 DC F5 F9 54 05 04 01 4D FE 2C 8D 03 34 F5 FA 54 01 04 01 4D FE 2C 8D 03 34 F5 FA 54 02 04 01 4D FE 2C 8D 03 34 F5 FA 54 03 04 01 4D FE 2C 8D 03 34 F5 FA 54 04 04 01 4D FE 2C 8D 03 34 F5 FA 54 05 04 01 7C B8

Tx: 17:32:39: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 09 50 04 00 0A 05 B7 00 00 5E 25

Rx: 17:32:40: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 90 51 0E 00 0A 05 C1 4D FE 2C 8E 01 A4 F5 FB 54 01 04 01 4D FE 2C 8E 01 A4 F5 FB 54 02 04 01 4D FE 2C 8E 01 A4 F5 FB 54 03 04 01 4D FE 2C 8E 01 A4 F5 FB 54 04 04 01 4D FE 2C 8E 01 A4 F5 FB 54 05 04 01 4D FE 2C 8E 03 98 F5 FC 54 01 04 01 4D FE 2C 8E 03 98 F5 FC 54 02 04 01 4D FE 2C 8E 03 98 F5 FC 54 03 04 01 4D FE 2C 8E 03 98 F5 FC 54 04 04 01 4D FE 2C 8E 03 98 F5 FC 54 05 04 01 69 DA

Tx: 17:32:40: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0A 50 04 00 0A 05 C1 00 00 98 F6

Rx: 17:32:40: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 A0 51 0E 00 0A 05 CB 4D FE 2C 8F 02 12 F5 FD 54 01 04 01 4D FE 2C 8F 02 12 F5 FD 54 02 04 01 4D FE 2C 8F 02 12 F5 FD 54 03 04 01 4D FE 2C 8F 02 12 F5 FD 54 04 04 01 4D FE 2C 8F 02 12 F5 FD 54 05 04 01 4D FE 2C 90 00 32 F5 FE 54 01 04 01 4D FE 2C 90 00 32 F5 FE 54 02 04 01 4D FE 2C 90 00 32 F5 FE 54 03 04 01 4D FE 2C 90 00 3C F5 FE 54 04 04 01 4D FE 2C 90 00 3C F5 FE 54 05 04 01 B3 94

Tx: 17:32:40: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0B 50 04 00 0A 05 CB 00 00 81 DD

Rx: 17:32:41: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 B0 51 0E 00 0A 05 D5 4D FE 2C 90 02 80 F5 FF 54 01 04 01 4D FE 2C 90 02 80 F5 FF 54 02 04 01 4D FE 2C 90 02 80 F5 FF 54 03 04 01 4D FE 2C 90 02 80 F5 FF 54 04 04 01 4D FE 2C 90 02 80 F5 FF 54 05 04 01 4D FE 2C 91 00 96 F6 00 54 01 04 01 4D FE 2C 91 00 96 F6 00 54 02 04 01 4D FE 2C 91 00 96 F6 00 54 03 04 01 4D FE 2C 91 00 96 F6 00 54 04 04 01 4D FE 2C 91 00 96 F6 00 54 05 04 01 E4 9D

Tx: 17:32:41: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0C 50 04 00 0A 05 D5 00 00 6B 66

Rx: 17:32:42: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 C0 51 0E 00 0A 05 DF 4D FE 2C 91 02 8A F6 01 54 01 04 01 4D FE 2C 91 02 8A F6 01 54 02 04 01 4D FE 2C 91 02 94 F6 01 54 03 04 01 4D FE 2C 91 02 94 F6 01 54 04 04 01 4D FE 2C 91 02 94 F6 01 54 05 04 01 4D FE 2C 92 00 AA F6 02 54 01 04 01 4D FE 2C 92 00 AA F6 02 54 02 04 01 4D FE 2C 92 00 AA F6 02 54 03 04 01 4D FE 2C 92 00 AA F6 02 54 04 04 01 4D FE 2C 92 00 AA F6 02 54 05 04 01 06 B6

Tx: 17:32:42: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0D 50 04 00 0A 05 DF 00 00 72 4D

Rx: 17:32:42: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 D0 51 0E 00 0A 05 E9 4D FE 2C 92 03 02 F6 03 54 01 04 01 4D FE 2C 92 03 02 F6 03 54 02 04 01 4D FE 2C 92 03 02 F6 03 54 03 04 01 4D FE 2C 92 03 02 F6 03 54 04 04 01 4D FE 2C 92 03 02 F6 03 54 05 04 01 4D FE 2C 93 01 68 F6 04 54 01 04 01 4D FE 2C 93 01 68 F6 04 54 02 04 01 4D FE 2C 93 01 68 F6 04 54 03 04 01 4D FE 2C 93 01 68 F6 04 54 04 04 01 4D FE 2C 93 01 68 F6 04 54 05 04 01 4D 45

Tx: 17:32:42: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0E 50 04 00 0A 05 E9 00 00 FC 5A

Rx: 17:32:43: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 E0 51 0E 00 0A 05 F3 4D FE 2C 93 03 66 F6 05 54 01 04 01 4D FE 2C 93 03 66 F6 05 54 02 04 01 4D FE 2C 93 03 66 F6 05 54 03 04 01 4D FE 2C 93 03 66 F6 05 54 04 04 01 4D FE 2C 93 03 66 F6 05 54 05 04 01 4D FE 2C 94 01 72 F6 06 54 01 04 01 4D FE 2C 94 01 72 F6 06 54 02 04 01 4D FE 2C

94 01 72 F6 06 54 03 04 01 4D FE 2C 94 01 72 F6 06 54 04 04 01 4D FE 2C 94 01 72 F6 06 54 05 04 01 B9 BB

Tx: 17:32:43: init: KF2 Message, Dest 4, Src 1, Command# 80 = Request Event Logs

AE 04 0F 01 02 0F 50 04 00 0A 05 F3 00 00 F7 40

Rx: 17:32:44: reply: KF2 Message, Dest 1, Src 4, Command# 81 = Send Event Logs

AE 01 85 04 02 F0 51 0E 00 0A 05 FD 4D FE 2C 94 03 C0 F6 07 54 01 04 01 4D FE 2C 94 03 C0 F6 07 54 02 04 01 4D FE 2C 94 03 C0 F6 07 54 03 04 01 4D FE 2C 94 03 C0 F6 07 54 04 04 01 4D FE 2C 94 03 C0 F6 07 54 05 04 01 4D FE 2C 95 01 D6 F6 08 54 01 04 01 4D FE 2C 95 01 D6 F6 08 54 02 04 01 4D FE 2C 95 01 D6 F6 08 54 03 04 01 4D FE 2C 95 01 D6 F6 08 54 04 04 01 4D FE 2C 95 01 D6 F6 08 54 05 04 01 CB FD

Tx: 17:32:44: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 09 50 04 00 0A 03 BB 00 00 ED 46

Rx: 17:32:44: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 90 51 0E 00 0A 03 C5 2D D6 F5 C3 03 8E 1A 58 54 01 05 01 2D D6 F5 C3 03 8E 1A 58 54 02 05 01 2D D6 F5 C3 03 8E 1A 58 54 03 05 01 2D D6 F5 C3 03 8E 1A 58 54 04 05 01 2D D6 F5 C3 03 8E 1A 58 54 05 05 01 2D D6 F5 C4 01 A4 1A 59 54 01 05 01 2D D6 F5 C4 01 A4 1A 59 54 02 05 01 2D D6 F5 C4 01 A4 1A 59 54 03 05 01 2D D6 F5 C4 01 A4 1A 59 54 04 05 01 2D D6 F5 C4 01 A4 1A 59 54 05 05 01 DB 9E

Tx: 17:32:44: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0A 50 04 00 0A 03 C5 00 00 AA 9D

Rx: 17:32:45: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 A0 51 0E 00 0A 03 CF 2D D6 F5 C4 03 98 1A 5A 54 01 05 01 2D D6 F5 C4 03 98 1A 5A 54 02 05 01 2D D6 F5 C4 03 98 1A 5A 54 03 05 01 2D D6 F5 C4 03 98 1A 5A 54 04 05 01 2D D6 F5 C4 03 98 1A 5A 54 05 05 01 2D D6 F5 C5 01 A4 1A 5B 54 01 05 01 2D D6 F5 C5 01 A4 1A 5B 54 02 05 01 2D D6 F5 C5 01 A4 1A 5B 54 03 05 01 2D D6 F5 C5 01 A4 1A 5B 54 04 05 01 2D D6 F5 C5 01 A4 1A 5B 54 05 05 01 DA B6

Tx: 17:32:45: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0B 50 04 00 0A 03 CF 00 00 B3 B6

Rx: 17:32:46: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 B0 51 0E 00 0A 03 D9 2D D6 F5 C6 00 14 1A 5C 54 01 05 01 2D D6 F5 C6 00 14 1A 5C 54 02 05 01 2D D6 F5 C6 00 14 1A 5C 54 03 05 01 2D D6 F5 C6 00 14 1A 5C 54 04 05 01 2D D6 F5 C6 00 14 1A 5C 54 05 05 01 2D D6 F5 C6 02 62 1A 5D 54 01 05 01 2D D6 F5 C6 02 6C 1A 5D 54 02 05 01 2D D6 F5 C6 02 6C 1A 5D 54 03 05 01 2D D6 F5 C6 02 6C 1A 5D 54 04 05 01 2D D6 F5 C6 02 6C 1A 5D 54 05 05 01 C5 72

Tx: 17:32:46: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0C 50 04 00 0A 03 D9 00 00 D8 05

Rx: 17:32:46: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 C0 51 0E 00 0A 03 E3 2D D6 F5 C7 00 6E 1A 5E 54 01 05 01 2D D6 F5 C7 00 6E 1A 5E 54 02 05 01 2D D6 F5 C7 00 6E 1A 5E 54 03 05 01 2D D6 F5 C7 00 6E 1A 5E 54 04 05 01 2D D6 F5 C7 00 78 1A 5E 54 05 05 01 2D D6 F5 C7 02 62 1A 5F 54 01 05 01 2D D6 F5 C7 02 62 1A 5F 54 02 05 01 2D D6 F5 C7 02 62 1A 5F 54 03 05 01 2D D6 F5 C7 02 62 1A 5F 54 04 05 01 2D D6 F5 C7 02 62 1A 5F 54 05 05 01 30 91

Tx: 17:32:46: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0D 50 04 00 0A 03 E3 00 00 F7 7D

Rx: 17:32:47: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 D0 51 0E 00 0A 03 ED 2D D6 F5 C8 00 6E 1A 60 54 01 05 01 2D D6 F5 C8 00 6E 1A 60 54 02 05 01 2D D6 F5 C8 00 6E 1A 60 54 03 05 01 2D D6 F5 C8 00 6E 1A 60 54 04 05 01 2D D6 F5 C8 00 78 1A 60 54 05 05 01 2D D6 F5 C8 02 6C 1A 61 54 01 05 01 2D D6 F5 C8 02 6C 1A 61 54 02 05 01 2D D6 F5 C8 02 6C 1A 61 54 03 05 01 2D D6 F5 C8 02 6C 1A 61 54 04 05 01 2D D6 F5 C8 02 6C 1A 61 54 05 05 01 AB 8C

Tx: 17:32:47: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0E 50 04 00 0A 03 ED 00 00 CE 31

Rx: 17:32:48: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 E0 51 0E 00 0A 03 F7 2D D6 F5 C9 00 D2 1A 62 54 01 05 01 2D D6 F5 C9 00 DC 1A 62 54 02 05 01 2D D6 F5 C9 00 DC 1A 62 54 03 05 01 2D D6 F5 C9 00 DC 1A 62 54 04 05 01 2D D6 F5 C9 00 DC 1A 62 54 05 05 01 2D D6 F5 C9 02 C6 1A 63 54 01 05 01 2D D6 F5 C9 02 C6 1A 63 54 02 05 01 2D D6 F5 C9 02 C6 1A 63 54 03 05 01 2D D6 F5 C9 02 C6 1A 63 54 04 05 01 2D D6 F5 C9 02 C6 1A 63 54 05 05 01 A5 5B

Tx: 17:32:48: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0F 50 04 00 0A 03 F7 00 00 C5 2B

Rx: 17:32:48: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 F0 51 0E 00 0A 04 01 2D D6 F5 CA 00 DC 1A 64 54 01 05 01 2D D6 F5 CA 00 DC 1A 64 54 02 05 01 2D D6 F5 CA 00 DC 1A 64 54 03 05 01 2D D6 F5 CA 00 DC 1A 64 54 04 05 01 2D D6 F5 CA 00 DC 1A 64 54 05 05 01 2D D6 F5 CA 02 D0 1A 65 54 01 05 01 2D D6 F5 CA 02 D0 1A 65 54 02 05 01

2D D6 F5 CA 02 D0 1A 65 54 03 05 01 2D D6 F5 CA 02 D0 1A 65 54 04 05 01 2D D6 F5 CA 02 D0 1A 65 54 05 05 01 C5 31

Tx: 17:32:48: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 01 50 04 00 0A 04 01 00 00 E1 ED

Rx: 17:32:49: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 10 51 0E 00 0A 04 0B 2D D6 F5 CB 01 36 1A 66 54 01 05 01 2D D6 F5 CB 01 36 1A 66 54 02 05 01 2D D6 F5 CB 01 36 1A 66 54 03 05 01 2D D6 F5 CB 01 36 1A 66 54 04 05 01 2D D6 F5 CB 01 40 1A 66 54 05 05 01 2D D6 F5 CB 03 2A 1A 67 54 01 05 01 2D D6 F5 CB 03 2A 1A 67 54 02 05 01 2D D6 F5 CB 03 2A 1A 67 54 03 05 01 2D D6 F5 CB 03 2A 1A 67 54 04 05 01 2D D6 F5 CB 03 2A 1A 67 54 05 05 01 7C 58

Tx: 17:32:49: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 02 50 04 00 0A 04 0B 00 00 98 25

Rx: 17:32:49: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 20 51 0E 00 0A 04 15 2D D6 F5 CC 01 4A 1A 68 54 01 05 01 2D D6 F5 CC 01 4A 1A 68 54 02 05 01 2D D6 F5 CC 01 4A 1A 68 54 03 05 01 2D D6 F5 CC 01 4A 1A 68 54 04 05 01 2D D6 F5 CC 01 4A 1A 68 54 05 05 01 2D D6 F5 CC 03 98 1A 69 54 01 05 01 2D D6 F5 CC 03 98 1A 69 54 02 05 01 2D D6 F5 CC 03 98 1A 69 54 03 05 01 2D D6 F5 CC 03 98 1A 69 54 04 05 01 2D D6 F5 CC 03 98 1A 69 54 05 05 01 92 40

Tx: 17:32:50: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 03 50 04 00 0A 04 15 00 00 D3 BB

Rx: 17:32:50: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 30 51 0E 00 0A 04 1F 2D D6 F5 CD 01 A4 1A 6A 54 01 05 01 2D D6 F5 CD 01 A4 1A 6A 54 02 05 01 2D D6 F5 CD 01 A4 1A 6A 54 03 05 01 2D D6 F5 CD 01 A4 1A 6A 54 04 05 01 2D D6 F5 CD 01 A4 1A 6A 54 05 05 01 2D D6 F5 CD 03 98 1A 6B 54 01 05 01 2D D6 F5 CD 03 98 1A 6B 54 02 05 01 2D D6 F5 CD 03 98 1A 6B 54 03 05 01 2D D6 F5 CD 03 98 1A 6B 54 04 05 01 2D D6 F5 CD 03 98 1A 6B 54 05 05 01 65 C2

Tx: 17:32:50: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 04 50 04 00 0A 04 1F 00 00 6B B5

Rx: 17:32:51: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 40 51 0E 00 0A 04 29 2D D6 F5 CE 01 A4 1A 6C 54 01 05 01 2D D6 F5 CE 01 A4 1A 6C 54 02 05 01 2D D6 F5 CE 01 A4 1A 6C 54 03 05 01 2D D6 F5 CE 01 A4 1A 6C 54 04 05 01 2D D6 F5 CE 01 A4 1A 6C 54 05 05 01 2D D6 F5 CE 03 A2 1A 6D 54 01 05 01 2D D6 F5 CE 03 A2 1A 6D 54 02 05 01 2D D6 F5 CE 03 A2 1A 6D 54 03 05 01 2D D6 F5 CE 03 A2 1A 6D 54 04 05 01 2D D6 F5 CE 03 A2 1A 6D 54 05 05 01 E4 1E

Tx: 17:32:51: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 05 50 04 00 0A 04 29 00 00 85 41

Rx: 17:32:51: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 50 51 0E 00 0A 04 33 2D D6 F5 CF 01 AE 1A 6E 54 01 05 01 2D D6 F5 CF 01 AE 1A 6E 54 02 05 01 2D D6 F5 CF 01 AE 1A 6E 54 03 05 01 2D D6 F5 CF 01 AE 1A 6E 54 04 05 01 2D D6 F5 CF 01 AE 1A 6E 54 05 05 01 2D D6 F5 D0 00 1E 1A 6F 54 01 05 01 2D D6 F5 D0 00 1E 1A 6F 54 02 05 01 2D D6 F5 D0 00 1E 1A 6F 54 03 05 01 2D D6 F5 D0 00 1E 1A 6F 54 04 05 01 2D D6 F5 D0 00 1E 1A 6F 54 05 05 01 24 93

Tx: 17:32:52: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 06 50 04 00 0A 04 33 00 00 EE B8

Rx: 17:32:52: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 60 51 0E 00 0A 04 3D 2D D6 F5 D0 02 12 1A 70 54 01 05 01 2D D6 F5 D0 02 12 1A 70 54 02 05 01 2D D6 F5 D0 02 12 1A 70 54 03 05 01 2D D6 F5 D0 02 12 1A 70 54 04 05 01 2D D6 F5 D0 02 12 1A 70 54 05 05 01 2D D6 F5 D1 00 82 1A 71 54 01 05 01 2D D6 F5 D1 00 82 1A 71 54 02 05 01 2D D6 F5 D1 00 82 1A 71 54 03 05 01 2D D6 F5 D1 00 82 1A 71 54 04 05 01 2D D6 F5 D1 00 82 1A 71 54 05 05 01 D4 0A

Tx: 17:32:52: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 07 50 04 00 0A 04 3D 00 00 B7 17

Rx: 17:32:53: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 70 51 0E 00 0A 04 47 2D D6 F5 D1 02 D0 1A 72 54 01 05 01 2D D6 F5 D1 02 D0 1A 72 54 02 05 01 2D D6 F5 D1 02 D0 1A 72 54 03 05 01 2D D6 F5 D1 02 D0 1A 72 54 04 05 01 2D D6 F5 D1 02 D0 1A 72 54 05 05 01 2D D6 F5 D2 00 DC 1A 73 54 01 05 01 2D D6 F5 D2 00 DC 1A 73 54 02 05 01 2D D6 F5 D2 00 E6 1A 73 54 03 05 01 2D D6 F5 D2 00 E6 1A 73 54 04 05 01 2D D6 F5 D2 00 E6 1A 73 54 05 05 01 F8 F3

Tx: 17:32:53: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 08 50 04 00 0A 04 47 00 00 E2 23

Rx: 17:32:53: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 80 51 0E 00 0A 04 51 2D D6 F5 D2 02 D0 1A 74 54 01 05 01 2D D6 F5 D2 02 D0 1A 74 54 02 05 01 2D D6 F5 D2 02 D0 1A 74 54 03 05 01 2D D6 F5 D2 02 D0 1A 74 54 04 05 01 2D D6 F5 D2 02 D0 1A 74 54 05 05 01 2D D6 F5 D3 00 DC 1A 75 54 01 05 01 2D D6 F5 D3 00 DC 1A 75 54 02 05 01

2D D6 F5 D3 00 E6 1A 75 54 03 05 01 2D D6 F5 D3 00 E6 1A 75 54 04 05 01 2D D6 F5 D3 00 E6 1A 75 54 05 05 01 20 1B

Tx: 17:32:54: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 09 50 04 00 0A 04 51 00 00 28 B5

Rx: 17:32:54: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 90 51 0E 00 0A 04 5B 2D D6 F5 D3 02 D0 1A 76 54 01 05 01 2D D6 F5 D3 02 D0 1A 76 54 02 05 01 2D D6 F5 D3 02 DA 1A 76 54 03 05 01 2D D6 F5 D3 02 DA 1A 76 54 04 05 01 2D D6 F5 D3 02 DA 1A 76 54 05 05 01 2D D6 F5 D4 00 DC 1A 77 54 01 05 01 2D D6 F5 D4 00 DC 1A 77 54 02 05 01 2D D6 F5 D4 00 DC 1A 77 54 03 05 01 2D D6 F5 D4 00 DC 1A 77 54 04 05 01 2D D6 F5 D4 00 DC 1A 77 54 05 05 01 24 18

Tx: 17:32:54: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0A 50 04 00 0A 04 5B 00 00 51 7D

Rx: 17:32:55: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 A0 51 0E 00 0A 04 65 2D D6 F5 D4 02 D0 1A 78 54 01 05 01 2D D6 F5 D4 02 DA 1A 78 54 02 05 01 2D D6 F5 D4 02 DA 1A 78 54 03 05 01 2D D6 F5 D4 02 DA 1A 78 54 04 05 01 2D D6 F5 D4 02 DA 1A 78 54 05 05 01 2D D6 F5 D5 00 DC 1A 79 54 01 05 01 2D D6 F5 D5 00 DC 1A 79 54 02 05 01 2D D6 F5 D5 00 E6 1A 79 54 03 05 01 2D D6 F5 D5 00 E6 1A 79 54 04 05 01 2D D6 F5 D5 00 E6 1A 79 54 05 05 01 71 2C

Tx: 17:32:55: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0B 50 04 00 0A 04 65 00 00 3E 81

Rx: 17:32:55: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 B0 51 0E 00 0A 04 6F 2D D6 F5 D5 02 DA 1A 7A 54 01 05 01 2D D6 F5 D5 02 DA 1A 7A 54 02 05 01 2D D6 F5 D5 02 DA 1A 7A 54 03 05 01 2D D6 F5 D5 02 DA 1A 7A 54 04 05 01 2D D6 F5 D5 02 DA 1A 7A 54 05 05 01 2D D6 F5 D6 01 40 1A 7B 54 01 05 01 2D D6 F5 D6 01 40 1A 7B 54 02 05 01 2D D6 F5 D6 01 40 1A 7B 54 03 05 01 2D D6 F5 D6 01 40 1A 7B 54 04 05 01 2D D6 F5 D6 01 40 1A 7B 54 05 05 01 00 CF

Tx: 17:32:56: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0C 50 04 00 0A 04 6F 00 00 86 8F

Rx: 17:32:56: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 C0 51 0E 00 0A 04 79 2D D6 F5 D6 03 3E 1A 7C 54 01 05 01 2D D6 F5 D6 03 3E 1A 7C 54 02 05 01 2D D6 F5 D6 03 3E 1A 7C 54 03 05 01 2D D6 F5 D6 03 3E 1A 7C 54 04 05 01 2D D6 F5 D6 03 3E 1A 7C 54 05 05 01 2D D6 F5 D7 01 A4 1A 7D 54 01 05 01 2D D6 F5 D7 01 A4 1A 7D 54 02 05 01 2D D6 F5 D7 01 A4 1A 7D 54 03 05 01 2D D6 F5 D7 01 A4 1A 7D 54 04 05 01 2D D6 F5 D7 01 A4 1A 7D 54 05 05 01 8F 57

Tx: 17:32:56: init: KF2 Message, Dest 5, Src 1, Command# 80 = Request Event Logs

AE 05 0F 01 02 0D 50 04 00 0A 04 79 00 00 4C 19

Rx: 17:32:57: reply: KF2 Message, Dest 1, Src 5, Command# 81 = Send Event Logs

AE 01 85 05 02 D0 51 0E 00 0A 04 83 2D D6 F5 D7 03 98 1A 7E 54 01 05 01 2D D6 F5 D7 03 A2 1A 7E 54 02 05 01 2D D6 F5 D7 03 A2 1A 7E 54 03 05 01 2D D6 F5 D7 03 A2 1A 7E 54 04 05 01 2D D6 F5 D7 03 A2 1A 7E 54 05 05 01 2D D6 F5 D8 01 A4 1A 7F 54 01 05 01 2D D6 F5 D8 01 A4 1A 7F 54 02 05 01 2D D6 F5 D8 01 AE 1A 7F 54 03 05 01 2D D6 F5 D8 01 AE 1A 7F 54 04 05 01 2D D6 F5 D8 01 AE 1A 7F 54 05 05 01 5E EC

Tx: 17:32:58: init: KF2 Message, Dest 2, Src 1, Command# 16 = Get Multi Data

AE 02 49 01 01 03 10 20 10 00 10 01 10 02 10 03 10 04 10 05 10 06 10 07 10 08 10 09 10 0A 10 0B 10 0C 10 0D 10 0E 10 0F 10 10 10 11 10 12 10 13 10 14 10 15 10 16 10 17 10 18 10 19 10 1A 10 1B 10 1C 10 1D 10 1E 10 1F BB A5

Rx: 17:32:58: reply: KF2 Message, Dest 1, Src 2, Command# 17 = Send Multi Data

AE 01 89 02 02 30 11 20 10 00 68 99 10 01 68 99 10 02 68 99 10 03 68 99 10 04 68 99 10 05 68 99 10 06 68 99 10 07 68 99 10 08 68 99 10 09 68 99 10 0A 68 99 10 0B 68 99 10 0C 68 99 10 0D 68 99 10 0E 68 99 10 0F 68 99 10 10 68 99 10 11 68 99 10 12 68 99 10 13 68 99 10 14 68 99 10 15 68 99 10 16 68 99 10 17 68 99 10 18 68 99 10 19 68 99 10 1A 68 99 10 1B 68 99 10 1C 68 99 10 1D 68 99 10 1E 68 99 10 1F 68 99 23 C6

Appendix S – Regression Analysis

Rx_update (ALL)

SUMMARY OUTPUT

Regression Statistics	
Multiple R	1
R Square	1
Adjusted R Square	1
Standard Error	3.6E-17
Observations	7

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.013701637	0.013701637	1.07933E+31	4.95929E-77
Residual	5	6.34729E-33	1.26946E-33		
Total	6	0.013701637			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.57192	1.77121E-17	3.22897E+16	5.40735E-82	0.571916667	0.571916667	0.571916667	0.571916667
X Variable 1	0.00208	6.34135E-19	3.28531E+15	4.95929E-77	0.002083333	0.002083333	0.002083333	0.002083333

Rx_update (Requested Number of Blocks)

SUMMARY OUTPUT

Regression Statistics	
Multiple R	1
R Square	1
Adjusted R Square	1
Standard Error	1.48952E-16
Observations	7

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.013701637	0.013701637	6.1756E+29	6.33294E-74
Residual	5	1.10934E-31	2.21867E-32		
Total	6	0.013701637			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.402630952	7.40468E-17	5.43752E+15	3.99299E-78	0.402630952	0.402630952	0.402630952	0.402630952
X Variable 1	0.002083333	2.65106E-18	7.8585E+14	6.33294E-74	0.002083333	0.002083333	0.002083333	0.002083333

Rx_data

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.99999459
R Square	0.99998918
Adjusted R Square	0.999986475
Standard Error	0.000272528
Observations	6

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.027457036	0.027457036	369684.689	4.39016E-11
Residual	4	2.97086E-07	7.42715E-08		
Total	5	0.027457333			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.170069652	0.000154978	1097.38009	4.13733E-12	0.169639364	0.170499939	0.169639364	0.170499939
X Variable 1	0.006247335	1.02749E-05	608.0170137	4.39016E-11	0.006218807	0.006275863	0.006218807	0.006275863

Tx_data

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.987925738
R Square	0.975997263
Adjusted R Square	0.975197172
Standard Error	0.007073168
Observations	32

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.061029109	0.061029109	1219.857477	7.3895E-26
Residual	30	0.001500891	5.00297E-05		
Total	31	0.06253			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.103707661	0.00256053	40.50242556	9.50121E-28	0.098478362	0.10893696	0.098478362	0.10893696
X Variable 1	0.004729839	0.000135423	34.92645812	7.3895E-26	0.004453268	0.005006409	0.004453268	0.005006409

Event Logs

SUMMARY OUTPUT

Regression Statistics	
Multiple R	1
R Square	1
Adjusted R Square	1
Standard Error	1.6342E-16
Observations	5

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	1	0.009375	0.009375	3.51042E+29	1.06031E-44
Residual	3	8.01187E-32	2.67062E-32		
Total	4	0.009375			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.1725	1.28331E-16	1.34418E+15	9.08025E-46	0.1725	0.1725	0.1725	0.1725
X Variable 1	0.0125	2.10975E-17	5.92488E+14	1.06031E-44	0.0125	0.0125	0.0125	0.0125

Appendix T – Empirical data (9600 Baud)

Rx_update (ALL)		Radio link					
Registers	Baud	% Error	Regression:Calculation	Total message length(s)	Calculated message delay	Radio delay (Radio - Serial link)	%Delay over Radio link
1	9600	1.8	0.574	0.564	0.383	0.312	55
2		-0.3	0.576	0.578	0.395	0.308	53
3							
4		-0.1	0.580	0.581	0.394	0.303	52
5							
6							
7							
8		-2.6	0.589	0.604	0.408	0.343	57
9							
10							
11							
12							
13							
14							
15							
16		-1.9	0.605	0.617	0.405	0.338	55
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32		3.0	0.639	0.620	0.374	0.285	46
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44							
45							
46							
47							
48							
49							
50							
51							
52							
53							
54							
55							
56							
57							
58							
59							
60							
61							
62							
63							
64		0.2	0.705	0.704	0.392	0.322	46
65			0.707	0.977			
			0.270 delay constant				
65		0.0	0.977	0.977			
66		0.1	0.979	0.978		0.522	53
67							
68		-0.1	0.983	0.984		0.516	52
69							
70							

	-1.6	0.992	1.008	0.542	54
	1.6	1.008	0.992	0.490	49
	-1.7	1.042	1.060	0.560	53
	-0.2	1.108	1.110	0.515	46

192	2.3	1.511	1.477		
193	1.9	1.783	1.750	0.890	51
194	2.0	1.785	1.750	0.856	49
196	-0.6	1.789	1.800	0.942	52
200	1.2	1.798	1.776	0.898	51
208	-0.1	1.814	1.816	0.928	51

224	0.8	1.848	1.833	0.917	50
256	-1.2	1.914	1.937	0.925	48

449	0.7	3.395	3.372	1.676	50
450	-0.2	3.397	3.404	1.740	51
452	0.7	3.401	3.379	1.707	51
456	1.1	3.410	3.372	1.732	51
464	0.2	3.426	3.419	1.719	50
480	1.1	3.460	3.422	1.702	50
512	0.9	3.526	3.495	1.691	48

959					
960					
961	0.7	6.619	6.570	3.290	50
962	0.9	6.621	6.560	3.270	50
963					
964	0.7	6.625	6.580	3.270	50
965					
966					
967					
968	1.0	6.634	6.566	3.286	50
969					
970					
971					
972					
973					
974					
975					
976	0.5	6.650	6.614	3.303	50
977					
978					
979					
980					
981					
982					
983					
984					
985					
986					
987					
988					
989					
990					
991					
992					
993	0.2	6.686	6.671	3.281	49
994					
995					
996					
997					
998					
999					
1000					
1001					
1002					
1003					
1004					
1005					
1006					
1007					
1008					
1009					
1010					
1011					
1012					
1013					
1014					
1015					
1016					
1017					
1018					
1019					
1020					
1021					
1022					
1023					
1024	0.3	6.750	6.727	3.327	49
					51 **Average**
					2 **Standar dev**

Rx_update (Req No.Blocks)				Radio link		
Registers	Baud	Regression: Calculated	% Error	Total message length(s)	Calculated message delay	
	1	9600	0.405	-1.5	0.411	0.359
	2		0.407	1.7	0.400	0.346
	3					
	4		0.411	1.5	0.405	0.347
	5					
	6					
	7					
	8		0.419	1.0	0.415	0.348
	9					
	10					
	11					
	12					
	13					
	14					
	15					
	16		0.436	-1.4	0.442	0.359
	17					
	18					
	19					
	20					
	21					
	22					
	23					
	24					
	25					
	26					
	27					
	28					
	29					
	30					
	31					
	32		0.469	0.9	0.465	0.348
	33					
	34					
	35					
	36					
	37					
	38					
	39					
	40					
	41					
	42					
	43					
	44					
	45					
	46					
	47					
	48					
	49					
	50					
	51					
	52					
	53					
	54					
	55					
	56					
	57					
	58					
	59					
	60					
	61					
	62					
	63					
	64		0.536	-1.7	0.545	0.362
	65		0.538		0.798	
			0.260	**delay constant**		
	65		0.798	0.0	0.798	
	66		0.800	-1.2	0.810	

67			
68	0.804	-2.9	0.828
69			
70			
71			
72	0.813	-4.4	0.850
73			
74			
75			
76			
77			
78			
79			
80	0.829	-3.4	0.858
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			
92			
93			
94			
95			
96	0.863	-3.2	0.891
97			
98			
99			
100			
101			
102			
103			
104			
105			
106			
107			
108			
109			
110			
111			
112			
113			
114			
115			
116			
117			
118			
119			
120			
121			
122			
123			
124			
125			
126			
127			
128	0.929	0.6	0.924
129			
130			
131			
132			
133			
134			
135			
136			
137			
138			

1.648	-1.7	1.678
1.718	-2.5	1.757

	3.158	-2.5	3.239
	3.160	-3.5	3.276
	3.164	-3.7	3.286
	3.172	-3.7	3.295
	3.189	-3.4	3.300
	3.222	-3.1	3.325

512	3.289	-4.1	3.429

897	5.911	-2.8	6.080
898	5.913	-2.8	6.080
900	5.917	-3.0	6.100
904	5.925	-2.9	6.100
912	5.942	-2.1	6.070
928	5.975	-2.7	6.140
960	6.042	-3.8	6.282

Rx_data	Radio link			
Registers	Baud	Error %	Regression:Calculation	**Total message length(s)**
1	9600	13.8	0.176	0.155
2		15.5	0.183	0.158
3				0.178
4		1.6	0.195	0.192
5				0.209
6				0.195
7				0.201
8		0.9	0.220	0.218
9				0.235
10				0.219
11				0.228
12				0.240
13				0.243
14				0.261
15				0.260
16		-2.2	0.270	0.276
17				0.271
18				0.287
19				0.295
20				0.300
21				0.304
22				0.321
23				0.315
24				0.328
25				0.340
26				0.346
27				0.346
28				0.348
29				0.366
30				0.371
31				0.365
32		-0.3	0.370	0.371
33			0.376	0.657
	delay constant		**0.281**	
33			0.657	0.657
34		1.7	0.663	0.652
35				
36		0.6	0.676	0.672
37				
38				
39				
40		-2.3	0.701	0.717
41				
42				
43				
44				
45				
46				
47				
48		-2.1	0.751	0.767
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				

63			
64	-2.4	0.851	0.872
65	-1.9	1.138	1.160
66			
67			
68			
69			
70			
71			
72			
73			
74			
75			
76			
77			
78			
79			
80	-4.0	1.231	1.283
81			
82			
83			
84			
85			
86			
87			
88			
89			
90			
91			
92			
93			
94			
95			
96	-3.4	1.331	1.378
97	-1.2	1.618	1.638
98			
99			
100			
101			
102			
103			
104			
105			
106			
107			
108			
109			
110			
111			
112	-3.0	1.712	1.765
113			
114			
115			
116			
117			
118			
119			
120			
121			
122			
123			
124			
125			
126			
127			
128	-3.3	1.812	1.873

129	-2.5	2.099	2.153
144	-3.9	2.193	2.281
160	-3.7	2.293	2.381
161	-2.7	2.580	2.650
176	-3.9	2.673	2.782
192	-2.9	2.773	2.857

Tx_data	Radio link				
	Registers	Baud	Error %	Regression	Total message length(s)
	1	9600	3	0.108	0.105
	2		1	0.113	0.112
	3		3	0.118	0.115
	4		6	0.123	0.116
	5		-10	0.127	0.142
	6		-1	0.132	0.133
	7		5	0.137	0.130
	8		-2	0.142	0.145
	9		8	0.146	0.136
	10		-4	0.151	0.157
	11		3	0.156	0.151
	12		3	0.160	0.156
	13		0	0.165	0.166
	14		4	0.170	0.163
	15		-7	0.175	0.188
	16		1	0.179	0.178
	17		-1	0.184	0.186
	18		-1	0.189	0.190
	19		-5	0.194	0.203
	20		-4	0.198	0.206
	21		-5	0.203	0.214
	22		0	0.208	0.207
	23		-3	0.212	0.219
	24		4	0.217	0.208
	25		6	0.222	0.210
	26		-1	0.227	0.230
	27		1	0.231	0.228
	28		0	0.236	0.236
	29		-3	0.241	0.248
	30		5	0.246	0.235
	31		-1	0.250	0.252
	32		2	0.255	0.251

Event Logs						
No Logs	Baud	Predicted Transmision time(s)	Error %	Regression	Total message length(s)	
1	9600	0.185	-1.6	0.185		0.188
2		0.198	0.3	0.198		0.197
4		0.223	4.5	0.223		0.213
8		0.273	-2.3	0.273		0.279
10		0.298	-0.2	0.298		0.298
11				0.310		0.579
				0.269	**delay constant**	
11			0.0	0.579		0.579
12			-1.4	0.592		0.600
14			-2.0	0.617		0.629
18			-2.7	0.667		0.685
20			-2.6	0.692		0.710

References

Bailey, D., & Wright, E. (2003). *Practical SCADA for industry*

Baker, J., & Rubin, I. (1987). Polling with a General-Service Order Table. *Communications, IEEE Transactions on, 35*(3), 283-288.

Bose, S. K. (2001). *An introduction to queueing systems*

Boxma, O. J., Groenendijk, W. P., & Weststrate, J. A. (1990). "A pseudoconservation law for service systems with a polling table ,". *Communications, IEEE Transactions vol.38,* (no.10,), pp.1865-1870,. doi:doi: 10.1109/26.61458

Boxma, O. J., Levy, H., & Westrate, J. A. (1989. *Optimization of polling systems*. presented at the meeting of the Center for Mathematics and Computer Science, Amsterdam (Netherlands). Dept. of Operations Research, Statistics, and System Theory., NETHERLANDS.

Boxma, O. J., Levy, H., & Weststrate, J. A. (1991). Efficient visit frequencies for polling tables: minimization of waiting cost. *Queueing Systems, 9*(1), 133-162. doi:10.1007/bf01158795

Boxma, O. J., Levy, H., & Weststrate, J. A. (1993). Efficient visit orders for polling systems. *Performance Evaluation, Volume 18* (2), Pages 103-123. doi:10.1016/0166-5316(93)90031-O

Carden, F., Jedlicka, R. P., & Henry, R. (2002). *Telemetry systems engineering*: Artech House - Technology & Engineering.

Choudhury, G. L. (1990, 3-7 Jun 1990). Polling with a general service order table: gated service Symposium conducted at the meeting of the INFOCOM '90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. 'The Multiple Facets of Integration'. Proceedings., IEEE

Clarke, G., Reynders, D., & Wright, E. (2003). *Practical modern SCADA protocols: DNP3,IEC 60870.5 and related systems*: Newnes. Retrieved from http://books.google.co.nz/books?id=ta5dSentTzoC

Daneels, A., & Salter, W. (1999. *WHAT IS SCADA?* presented at the meeting of the International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy.

Deal, R. (2008). *Cisco Certified Network Associate study guide*

Haring, G., Lindemann, C., Reiser, M., & Takagi, H. (2000). Analysis and Application of Polling Models. In *Performance Evaluation: Origins and Directions* (Vol. 1769, pp.

423-442): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/3-540-46506-5_18. doi:10.1007/3-540-46506-5_18

Hong, S. H. (2001). Bandwidth allocation scheme for cyclic-service fieldbus networks. *Mechatronics, IEEE/ASME Transactions on, 6*(2), 197-204.

Hong, S. H., & Kim, W. H. (2000). Bandwidth allocation scheme in CAN protocol. *Control Theory and Applications, IEE Proceedings -, 147*(1), 37-44.

Horak, R. (2007). *Telecommunications and data communications handbook*

IEEE Power & Energy Society. (2010). *IEEE Standard for Electric Power Systems Communications - Distributed Network Protocol (DNP3)* (IEEE Std 1815-2010).

Ippolito, L. J. (2008). *Satellite communications systems engineering: atmospheric effects, satellite link design and system performance*

Levy, H., & Sidi, M. (1990). Polling systems: applications, modeling, and optimization. *Communications, IEEE Transactions on, 38*(10), 1750-1760.

Mackiewicz, R. E. (2006, 0-0 0). Overview of IEC 61850 and benefits Symposium conducted at the meeting of the Power Engineering Society General Meeting, 2006. IEEE

Mikluszka, W. (2010, 13-15 May 2010). Safety conditions verification of communication in distributed control system Symposium conducted at the meeting of the Human System Interactions (HSI), 2010 3rd Conference on

Modbus organisation. (2006). *MODBUS over serial line specification and implementation guide* (V1.02).

Radio Spectrum Management. Retrieved from http://www.rsm.govt.nz/cms/tools-and-services/publications/public-information-brochures-pibs/pib-21-table-of-radio-spectrum-usage-in-new-zealand/2-3-new-zealand-table-of-allocation-1/2-3-4-vhf-band-30-300mhz

Rappaport, T. S. (1996). *Wireless Communications: Principles and Practice*: Prentic Hall PTR.

Reynders, D., Mackay, S., & Wright, E. (2005). *Practical industrial data communications: best practice techniques*

Vishnevskii, V., & Semenova, O. (2006). Mathematical methods to study the polling systems. *Automation and Remote Control, 67*(2), 173-220. doi:10.1134/s0005117906020019

WEF, W. E. F. (2007). *Automation of wastewater treatment facilities*: McGraw-Hill Professional.

White, C. M. (2008). *Data Communications and Computer Networks: A Business User's Approach* Cengage Learning.

Wonderware. (2011). *HMI graphic visualization*. Retrieved from http://global.wonderware.com/SiteCollectionImages/Products/Product/InTouch/intouch-ce_thumb01_lg.jpg

Ziemer, R. E. (2007). *Fundamentals of spread spectrum modulation* Morgan & Claypool Publishers.