Form Number D4052
Part Number D301401X012
January 2013

BSAP Communications
Application Programmer's Reference

Remote Automation Solutions

EMERSON.
Process Management

IMPORTANT! READ INSTRUCTIONS BEFORE STARTING!

Be sure that these instructions are carefully read and understood before any operation is attempted. Improper use of this device in some applications may result in damage or injury. The user is urged to keep this book filed in a convenient location for future reference.

These instructions may not cover all details or variations in equipment or cover every possible situation to be met in connection with installation, operation or maintenance. Should problems arise that are not covered sufficiently in the text, the purchaser is advised to contact Emerson Process Management, Remote Automation Solutions for further information.

EQUIPMENT APPLICATION WARNING

The customer should note that a failure of this instrument or system, for whatever reason, may leave an operating process without protection. Depending upon the application, this could result in possible damage to property or injury to persons. It is suggested that the purchaser review the need for additional backup equipment or provide alternate means of protection such as alarm devices, output limiting, fail-safe valves, relief valves, emergency shutoffs, emergency switches, etc. If additional information is required, the purchaser is advised to contact Remote Automation Solutions.

RETURNED EQUIPMENT WARNING

When returning any equipment to Remote Automation Solutions for repairs or evaluation, please note the following: The party sending such materials is responsible to ensure that the materials returned to Remote Automation Solutions are clean to safe levels, as such levels are defined and/or determined by applicable federal, state and/or local law regulations or codes. Such party agrees to indemnify Remote Automation Solutions and save Remote Automation Solutions harmless from any liability or damage which Remote Automation Solutions may incur or suffer due to such party's failure to so act.

ELECTRICAL GROUNDING

Metal enclosures and exposed metal parts of electrical instruments must be grounded in accordance with OSHA rules and regulations pertaining to "Design Safety Standards for Electrical Systems," 29 CFR, Part 1910, Subpart S, dated: April 16, 1981 (OSHA rulings are in agreement with the National Electrical Code).

The grounding requirement is also applicable to mechanical or pneumatic instruments that include electrically operated devices such as lights, switches, relays, alarms, or chart drives.

EQUIPMENT DAMAGE FROM ELECTROSTATIC DISCHARGE VOLTAGE

This product contains sensitive electronic components that can be damaged by exposure to an electrostatic discharge (ESD) voltage. Depending on the magnitude and duration of the ESD, this can result in erratic operation or complete failure of the equipment. Read supplemental document S14006 for proper care and handling of ESD-sensitive components.

NOTE: This document describes the Bristol Synchronous / Asynchronous Protocol (BSAP). Some of the terminology and references are not relevant for current product lines such as ControlWave or future products, but remain to provide historical context for legacy products.

The basic BSAP protocol definition remains the same and applies to all BSAP-capable products.

The information of relevance to ControlWave and later products in this manual resides primarily in *Appendix B* which covers Remote Database Access (RDB) commands used in all-BSAP capable devices. Also see manuals and online help files relevant to newer products for additional information.

This page is intentionally left blank

Contents

Chapter 1 - Introduction

The objective of this manual is to provide application programmers with a comprehensive guide to Network 3000 communications. The intent is to make this manual as self-contained as possible. To this end, detailed information pertaining to the Bristol Synchronous/Asynchronous Protocol and the Remote Data Base Access message formats has been included in the appendices. This will enable users to troubleshoot Network 3000 communication systems and/or interface a foreign device to a Network 3000 controller. A detailed discussion of the network hierarchical structure, the communication protocol, and interface issues follows in subsequent chapters.

IMPORTANT

This document DOES NOT discuss the internal messages used for the Bristol Internet Protocol (IP) driver used in Open BSI 3.0 (or newer Open BSI versions). Details on IP protocol layers are available in numerous books, available at any large bookstore.

Since Internet Protocol (IP) is an industry standard protocol, third-party developers are urged to use the Open BSI Utilities provided, rather than attempting to generate customized implementations for communication with Bristol devices.

Reference Documents

- *Network Topology (NETTOP) User's Guide, D4057.*
- *DPC 3330 Instruction Manual, CI-3330.*
- *DPC 3335 Instruction Manual, CI-3335.*
- *GFC 3308-xx Instruction Manuals, CI-3308-B, CI-3308-50C*
- *RTU 3305 Instruction Manual, CI-3305.*
- *RTU 3310 Instruction Manual, CI-3310.*
- *Open BSI Utilities Manual (*Ver 2.3 or *earlier* users see document# D5076; Ver 3.x or *newer* users see document# D5081).

This page is intentionally left blank

Chapter 2 - BSAP Communications Overview

Introduction

The **Bristol Synchronous/Asynchronous Communication Protocol** (**BSAP**) is the foundation for a proprietary network that has a tree structured topology. This open-ended topology supports a variety of configurations which may include one or more nodes at each of up to six levels. Messages can be sent between nodes on the same level or on different levels. Each message is uniquely identified and has an error checking code associated with it.

BSAP operates in a polled environment. Each link in the network is capable of supporting a different poll rate. The rate selected depends on a variety of application-dependent factors and is discussed in detail in Chapter 7.

BSAP has been designed and implemented according to the functional layers of the International Standards Organization (ISO) model. Since each layer is independent of its adjacent layers, both synchronous and asynchronous transmission modes can be supported.

Network Hierarchical Structure

BSAP supports a simple tree topology. As a matter of terminology, the **network master** computer is defined as the root of the tree. Emanating from the root is the first level of **node**(s) or branch. From the first level there may be a second level; and from the second a third; and so forth up to a maximum of six levels. There is no requirement for symmetry within the tree structure.

The limit to the number of nodes on a branch is imposed by one of two limiting factors: (1) the application-dependent allowable response time for critical messages; or (2) the physical size of a node address (7 bits = 128 nodes).

As for response times, remember that BSAP operates in a polled environment. For each link in a network that a message must traverse there is a potential time lag associated with the polling cycle for that link. In addition, Alarm Report messages are given priority over other types of messages. The cumulative effect is a function of the number of levels within the network and could very easily become a determining factor (in conjunction with poll rates) in configuring your network. A complete discussion of poll rates is contained in Chapter 7.

The other practical limitation in network configuration is node addressing. Each node has a unique address which is based on the node's sequential position within its level and its level number within the network. At any given node, the allowable sequential positions are in the range of 1 to 127 and are defined as the node's **local address**. Local addresses do not have to be consecutive; you can configure your network with holes (address gaps) today and fill the holes with additional nodes in the future without having to renumber the older nodes. However, bear in mind that it is the highest number used (not the actual number of nodes) that determines the amount of

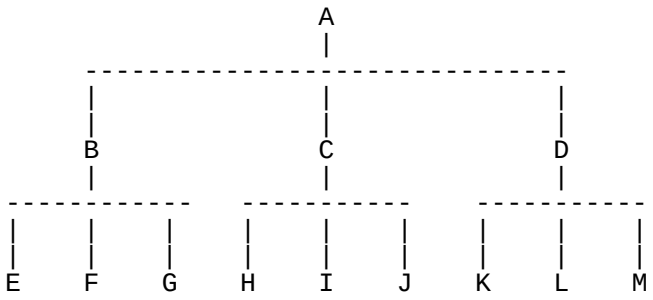
space required to define the node's address internally. Gaps will impose a burden in the form of dead node handling unless the **#NDARRAY** (node array) is used. This will be discussed in a later section.

Levels of local addresses are concatenated to yield a unique address for each node within the network. This network-unique address is known as the **global address**. The numeric value of a global address may not exceed 32,767. A detailed explanation of global address computation can be found in Chapter 4.

Relationship of Nodes on the Network

Any given node within the network (except the extremities) serves a dual purpose. It is **master** to the node(s) immediately below it and **slave** to the node immediately above it. These two relationships are defined as local because the nodes involved are adjacent to each other. Messages between a master and slave (with no intervening nodes) are defined as **local messages**. Messages which must pass through one or more intervening nodes to reach their destination are defined as **global messages**.

Figure 1 illustrates a simple network and the global and local relationships between its nodes. The configuration of the network (levels, node addresses, etc.) is specified using the Network Topology Program, also known as **NETTOP**.



GLOBAL ADDRESSES

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	-	L	L	L	G	G	G	G	G	G	G	G	G
B	L	-	G	G	L	L	L	G	G	G	G	G	G
C	L	G	-	G	G	G	G	L	L	L	G	G	G
D	L	G	G	-	G	G	G	G	G	G	L	L	L
E	G	L	G	G	-	G	G	G	G	G	G	G	G
F	G	L	G	G	G	-	G	G	G	G	G	G	G
G	G	L	G	G	G	G	-	G	G	G	G	G	G
H	G	G	L	G	G	G	G	-	G	G	G	G	G
I	G	G	L	G	G	G	G	G	-	G	G	G	G
J	G	G	L	G	G	G	G	G	G	-	G	G	G
K	G	G	G	L	G	G	G	G	G	G	-	G	G
L	G	G	G	L	G	G	G	G	G	G	G	-	G
M	G	G	G	L	G	G	G	G	G	G	G	G	-

G=Global Message L=Local Message

Figure 1

Protocol Layers

BSAP is designed and implemented in accordance with the International Standards Organization (ISO) model for Open System Interconnection. This model consists of seven layers, each of which provides a certain subset of functionality within the network. The layers may be traversed in an upward or downward fashion, depending on whether a message is being transmitted (downward) or received (upward). The bottom four layers, which Network 3000 uses, are described below. The description traces a typical message transmission through each layer.

The **Transport Layer** is responsible for accurate transmission of the message on a first-in/first-out basis at a functional level. When the transport layer determines it is ready to transmit, control is passed to the next layer.

The **Network Control Layer** is the primary transmission manipulator. It has the responsibility of determining how to route the message through the network, what addresses to use, and to establish the communication path.

The **Data Link Layer** is responsible for enhancing the message to include error checking and correction mechanisms. It also controls access to the physical channel over which the message is sent.

The bottom layer is the **Physical Link Layer**. This layer consists primarily of hardware and the software necessary to control it. This layer is totally independent of the final format of the message being transmitted.

A more detailed description of the BSAP protocol is provided in Appendix A. While BSAP was originally intended for use on asynchronous links, it has been extended to operate on synchronous links by either replacing the link level section with the appropriate synchronous link level or by including BSAP's link level within the link level information required by the synchronous level. In either case, operation of the link level for synchronous links is hidden from users of the communication facility; their access is normally via the transport layer. Interfacing to processors other than Bristol's is via asynchronous connection only, so detailed information on synchronous link level operation need not be provided in this manual.

Polling Philosophy

With the exception of the extremities, each node within the network is both a master to the nodes below it and a slave to the node above it. As a master, a node is responsible for periodically polling its slaves to determine their status and collect any available messages. As a slave, each node must respond to its master's poll. The **poll period**, or rate at which each master polls its slaves, is user-selectable and independent of all the other masters' polling rates. The polling philosophy used attempts to maximize message throughput by minimizing extraneous polls. This is accomplished by maintaining four types of polls: the main poll loop, the reactivation poll, the preferred poll loop, and the dead poll loop.

The **main poll loop** interrogates each slave which is alive to determine if it is dead or still alive and, if still alive, whether or not it has a data message to send. This poll is executed at the start of each polling cycle. A live slave which does not respond to three successive polls (in three successive polling periods) is assumed to be dead and becomes a candidate for the reactivation poll. A live slave which responds with a data message becomes a candidate for the preferred poll. A live slave with no data message is ignored until the next main polling cycle.

The **reactivation poll** is attempted only once per polling cycle. Its purpose is to determine if a known dead slave has become live. The choice of which dead slave to poll (assuming there is more than one) is made on a rotating basis from one poll cycle to the next. This ensures that every dead slave gets an equal chance to respond. The polling technique used is similar to the main poll loop. However, if the polling node has a valid Node Routing Table (described later in this chapter) then the Time Synchronization/Node Routing Table message is sent instead of the POLL message. If the polled slave responds, its status is changed to live.

The **preferred poll loop** is used to interrogate, on a round-robin basis, all of the slaves that responded to the main poll loop or reactivation poll with a data message. As long as a slave responds to the preferred poll with a data message, it remains a candidate; as soon as it responds with no data, it is removed from the preferred poll loop. The candidate slaves are polled sequentially, in local address order, until the polling period expires or all candidates have no more data messages. If there is not enough time left in the polling cycle to receive all the candidates' data, the subsequent preferred poll loop (in the next polling cycle) resumes where the previous one ended.

If there is time left after the preferred poll loop, the **dead poll loop** is used to give any remaining dead nodes an opportunity to inform the master that they are again candidates for the main poll. Unlike the preferred poll loop, the dead poll loop is executed only once during a polling cycle. There are two ways in which the master may effect the dead poll loop. If it has a valid Node Routing Table (NRT), the master will send each dead slave a Time Synchronization/NRT message. This message is described in Chapter 4 and is essentially the first step in getting a dead node back in step with the rest of the network. If a dead slave acknowledges this message, it becomes eligible for the main poll loop in the next polling cycle. If the master does not have a valid NRT, it will interrogate each of the dead slaves with a normal poll

message. If the dead slave responds, it becomes eligible for the main poll loop, and if it responds with a data message, it becomes a candidate for the preferred poll loop (in the next polling cycle) as well.

Failure Recovery

Failures within the network can occur for a variety of reasons and the recovery procedure used will vary accordingly. In all cases except discarded global messages an error code is assigned to the failed message. These error codes are listed in Chapter 8. In general there are three types of failures: routing, buffering and transmission.

Routing failures can occur in three ways: (1) a local receiving node is incapable of handling a global message which must pass through it; (2) a node incapable of originating a global message attempts to send one; and (3) a local receiving node is dead.

The ability of a particular node to handle global messages is based on the presence or absence of a **Node Routing Table (NRT)** within the node. The NRT must be present in order to process global messages. The NRT originates at the Network Master and is passed on to each succeeding level. Since each node receives its copy of the NRT from its master, the master is aware of its slaves' status regarding the NRT. Thus if a slave node has not accepted the NRT (the node is dead or the network master has not supplied the NRT yet), its master will discard any global message which must pass through that slave. Likewise, if the master receives a global message from a slave with no NRT (or if the global address is invalid) the master will discard that message as well.

It is important to note here that when a global message is discarded because of a routing error, there is no error code returned to the originator. It is assumed that the application which originated the message will use a timeout mechanism to detect that its messages are being discarded.

If a local receiving node is dead it obviously cannot accept any messages. The master is aware of its dead slaves as a result of the main poll loop. Should a message destined for a dead node arrive, the master will reject it and generate the appropriate error code.

Buffering failures occur when a node does not have sufficient space to accept a data message. The slave node informs its master of this condition via the **NAK** protocol message (described in Chapter 3). Upon receipt of a NAK for a local message, the master makes two additional attempts to transmit the message. If these attempts also fail, the master will inform the originator (via appropriate error code) of the condition and reject the message. This will cause a throttling down of messages to the clogged node which in turn will give the node an opportunity to clear itself. A global message which is NAK'ed is discarded without returning an error to the originator.

Transmission errors are usually the result of noise on the transmission line.

Should this occur, the receiving node will be incapable of decoding the message and will therefore not acknowledge it. A timeout facility is provided to allow for the unsolicited retransmission of such messages if they are local. Two such attempts are made before the master informs the originator (via an appropriate error code) of the condition. If the message is global, it is discarded.

Pseudomaster

Network 3000 also supports sub nodes which are known as **Pseudomaster** devices and are connected to Pseudo Slave ports. These devices are ancillary to the network and typically consist of a portable Personal Computer executing **ACCOL Tools** software. The Pseudomaster device is not a part of the network since it does not have a global address and as such does not appear as a node in the network as described in the NETTOP file. The main purpose of the Pseudomaster device is to access the data base in a node without affecting its normal operation. The TS/NRT message can NOT be transmitted through a Pseudo Slave Port as it will be ignored, (i.e. simply discarded) at the application level. The message will be ACKed at the protocol level. During the configuration process the ACCOL programmer may choose to configure *up to two ports on a node as a pseudo slave* port, however only one pseudo slave port can be specified to accept alarms.

The pseudo port is limited to 8 (3 in AD and earlier firmware) messages outstanding at any time. This does not affect normal operation with the ACCOL Tools software but it does limit the throughput of the pseudo port in some unusual applications.

Chapter 3 - Communication Message Structure

Protocol Messages (POLL, ACK, ACK-NODATA, NAK, UP-ACK, Node Status Byte, DIAL_UP_ACK)

The POLL message is used by a master to interrogate its slaves and determine if the slave is alive, and if so, to solicit data. The format of the poll message is:

DLE,STX,ADDR,SER,POLL,PRI,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	ADDR	=	Address of polled node
	SER	=	Serial number of message
	POLL	=	Function code 85H
	PRI	=	Priority of requested data (see Note)
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check - 2 bytes

Note: A PRI of 0 indicates that alarms or data can be accepted; A PRI of 10H indicates that alarms cannot be accepted.

The ACK (also called DOWN-ACK) message is used by a slave to acknowledge receipt of a data message (not a POLL) from its master. Its format is:

DLE,STX,ADDR,SER,DTA,SLV,NSB,DOWN,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	ADDR	=	Address of master node (always 0)
	SER	=	Serial number of message ACK'd
	DTA	=	Function code 86H
	SLV	=	Local address of slave responding
	NSB	=	Node Status Byte
	DOWN	=	Number of buffers in use
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check

The ACK-NODATA message is used by a slave to acknowledge receipt of a POLL and indicate that it has no data messages to respond with. The format is:

DLE,STX,ADDR,SER,NOD,SLV,NSB,DOWN,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	ADDR	=	Address of master node (always 0)
	SER	=	Serial number of message ACK'd
	NOD	=	Function code 87H
	SLV	=	Local address of slave responding
	NSB	=	Node Status Byte
	DOWN	=	Number of buffers in use
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check

The NAK message is used by a slave to indicate that a message other than a POLL was received but there is insufficient buffer space available. The format of the NAK message is:

DLE,STX,ADDR,SER,NAK,SLV,NSB,DOWN,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	ADDR	=	Address of master node (always 0)
	SER	=	Serial number of message being NAK'd
	NAK	=	Function code 95H
	SLV	=	Local address of slave responding
	NSB	=	Node Status Byte
	DOWN	=	Number of buffers in use
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check

The UP-ACK message is used by the master to inform the slave that it successfully received and buffered the message. Its format is:

DLE,STX,ADDR,SERM,UTA,SERS,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	ADDR	=	Local address of slave
	SERM	=	Master's message serial number
	UTA	=	Function code 8BH
	SERS	=	Serial number of message ACK'd
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check

The Node Status Byte is used to inform the Master of certain conditions existing within the slave. The bit definition of this byte may be found in Chapter 8.

Data Message (Global and Local)

Global data messages are those which must pass through at least one master before reaching their destination.[1] The general format for a global message is:

DLE,STX,LADD,SER,DADD,SADD,CTL,DFUN,SEQ,SFUN,NSB,*data*,DLE,ETX,CRC

where:	DLE	=	ASCII character 10H
	STX	=	ASCII character 02H
	-- beginning of the 12 byte Global header --		
	LADD	=	Local address + 80H
	SER	=	Message serial number
	DADD	=	Destination global address
	SADD	=	Source global address
	CTL	=	Control byte
	DFUN	=	Destination function code
	SEQ	=	Application sequence number
	SFUN	=	Source function code
	NSB	=	Node Status Byte
	--- end of the 12 byte Global header ---		
	data	=	Application-dependent data *(up to 241 bytes)*
	DLE	=	ASCII character 10H
	ETX	=	ASCII character 03H
	CRC	=	Cyclic Redundancy Check

[1] *Global data messages are those that contain the 12 byte global header to route the message to the specified destination. Generally, global messages ware used when the destination node is more than one network layer away. This message can, however, also be used to address a node in the next network layer.*

Local messages are those which do not have to pass through any nodes to reach their destination. By definition, the first node to receive a local message is the destination. The format of a local message is shown below. (The bytes from LADD through NSB form the local header):

DLE,STX,LADD,SER,DFUN,SEQ,SFUN,NSB,*data*,DLE,ETX,CRC

where:

DLE	=	ASCII character 10H
STX	=	ASCII character 02H
--- beginning of the 7 byte Local header ---		
LADD	=	Local address
SER	=	Message serial number
DFUN	=	Destination function code
SEQ	=	Application sequence number
SFUN	=	Source function code
NSB	=	Node Status Byte
--- end of the 7 byte Local header ---		
data	=	Application-dependent data (up to 246 bytes)
DLE	=	ASCII character 10H
ETX	=	ASCII character 03H
CRC	=	Cyclic Redundancy Check

The master on a communications line is always local address 0. Slave nodes have local addresses in the range 1 through 127.

DIAL_UP_ACK message

When the Slave Port successfully initiates a dial-up session, it monitors the line for receipt of any Poll message to any node. The poll message could be a standard Poll or an Expanded BSAP Poll, and can target any node. The DIAL_UP_ACK is sent in response to any poll and is matched to the Poll type as shown below:

Response to Standard Poll

DLE STX NA SN FC DSG DSA PT VNRT NVER DLE ETX CRC1 CRC2

where:

DLE	=	ASCII character 10H
STX	=	ASCII character 02H
NA	=	Node Address 0 (response to master)
SN	=	Serial number from POLL
FC	=	81h (DIAL_UP_ACK)
DSG	=	Dial-up slave group number (0)
DSA	=	Dial-up slave address (1-127)
PT	=	Port type (slave)
VNRT	=	Valid NRT Flag (1=valid; 0=invalid)
NVER	=	NRT Version (0 if VNRT=0)
DLE	=	ASCII character 10H
ETX	=	ASCII character 03H
CRC1	=	Cyclic Redundancy Check 1
CRC2	=	Cyclic Redundancy Check 2

Response to Expanded BSAP (EBSAP) Poll

DLE SOH GNO NA SN FC DSG DSA PT VNRT NVER DLE ETX CRC1 CRC2

DLE	=	ASCII character 10H
SOH	=	ASCII character 01
GNO	=	Group number from Exp. Poll
NA	=	Node Address 0 (response to master)
SN	=	Serial number from POLL
FC	=	81h (DIAL_UP_ACK)
DSG	=	Dial-up slave group number
DSA	=	Dial-up slave address (1-127)
PT	=	Port type (slave)
VNRT	=	Valid NRT Flag (1=valid; 0=invalid)
NVER	=	NRT Version (0 if VNRT=0)
DLE	=	ASCII character 10H
ETX	=	ASCII character 03H
CRC1	=	Cyclic Redundancy Check 1
CRC2	=	Cyclic Redundancy Check 2

This page is intentionally left blank

Chapter 4 - Data Transfer

This chapter discusses how data is transferred between nodes in a NETWORK 3000 configuration. Some of the data is required for the operation of the network while other data is used to support application specific programs. Each of the sub-chapters will discuss the purpose as well as the format and error codes, where appropriate, of the specific data transfer technique.

Alarm Handling

The alarm system of the Network 3000 supports two types of alarm signals: Analog and Logical Alarm Signals. Analog Alarms support the following types of conditions:

a. High Alarm - occurs when the value of the signal exceeds the application defined high limit. A Return-To-Normal occurs when the signal value falls below the high limit minus the high deadband.

b. Low Alarm - occurs when the value of the signal decreases below the application defined low limit. A Return-To-Normal occurs when the signal value falls above the low limit plus the low deadband.

c. High/High Alarm - occurs when the value of the signal exceeds the application defined high/high limit. A Return-To-Normal occurs when the signal value falls below the high/high limit minus the high deadband.

d. Low/Low Alarm - occurs when the value of the signal decreases below the application defined low/low limit. A Return-To-Normal occurs when the signal value falls above the low/low limit plus the low deadband.

Logical Alarms support the following types of conditions:

a. True State Alarm - occurs when the value of the signal is TRUE. The Return-To-Normal occurs when the signal becomes FALSE.

b. False State Alarm - occurs when the value of the signal is FALSE. The Return-To-Normal occurs when the signal becomes TRUE.

c. Change Of State Alarm - occurs when the signal value goes from TRUE to False or from FALSE to TRUE. The Return-To-Normal occurs when the alarm is acknowledged.

Both Logical and Analog Alarms support the Return-To-Normal Alarm, which is specific for each of the above alarm types.

Alarms are reported by each node to its local master node and optionally to a pseudomaster device (user selectable). Alarms traverse the network from node to node in an UPWARD direction until they arrive at the Network Master or CONSOLE. Each time a signal value is updated, the Alarm conditions are checked. If the signal is NOT alarm inhibited, and the alarm conditions are met, the signal is reported as "in alarm" and placed into a Time Stamp Buffer (i.e. a buffer containing alarm signals that are to be reported). With AIC Version 5.4, or earlier, the Time Stamp Buffer has a fixed size of 32 entries. With AIC Version 5.41, or later, the size of the Time Stamp Buffer is configurable upto 255 entries. When the alarm signal is placed into the Time Stamp Buffer it is given a five byte Date and Time stamp. The time stamp has a resolution of 20 milliseconds. Upon each poll by the local master, alarm reports (if present) in the slave are transmitted.

The local master, upon receipt of alarm report(s) from one or more slaves, places the received alarm reports in the circular Compaction List. More than 1 alarm may fit in a communication buffer. Therefore, alarms from the local master and/or one or more slaves may be placed in a single communication buffer for transmission to the next level of the network hierarchy. The compaction scheme reduces the number of communication buffers required for alarm reporting and allows communication buffers to become available for more alarm signals to be received. Alarms are typically reported as they occur, however in a node where all alarms have been reported a periodic timer is used to allow the alarm reporting task to check the Time Stamp Buffer.

The Network Master may acknowledge the alarm signals it receives. This has NO effect on the normal operation of the alarm system. When a node receives an alarm acknowledgment for an alarm report it has initiated, the alarm acknowledge state for the signal is cleared. More than one acknowledge may be transmitted in one acknowledge message. However, multiple acknowledge to one signal has no meaning. The node processing an alarm acknowledge must respond; the response has a single error code indicating whether or not the alarm acknowledge was successful.

A node may be sent the Alarm Initialization Request. The purpose is to have the node generate alarm reports for all signals which are in the Alarm Acknowledge state (i.e. those signals which have not been acknowledged yet). The alarm system will respond that it has successfully completed the Initialization process. The actual alarm reports will be transmitted upon subsequent polls.

Alarm reports have a higher priority than all other messages queued to go up; improved Alarm response is realized in this way. Alarm messages are throttled to prevent a node from becoming overwhelmed. The poll message priority byte is used to inform a slave node whether or not an alarm report message may be transmitted.

When alarms are being reported via a pseudo slave port as well as via the slave port, if either port ceases to accept alarms (due to throttling, communication failure, etc.) reporting will continue to the operational master or pseudo master ONLY.

After the failure is repaired, alarm reporting will resume with the next alarm which occurs being reported to both ports. Therefore, if alarms can be reported to either port, alarm gathering will continue. If neither port is polling for alarms, the alarm system will cease gathering alarms until alarm polling is resumed. When two ports are accepting alarms the maximum rate of alarm reporting is limited by the slowest port; this is particularly noticeable when the slave port is a 250kb data highway.

Alarm Message Formats

The Network 3000 Alarm system utilizes several types of alarm messages. These message types include Initialization Requests, Acknowledge Requests, Acknowledge Responses, and Alarm Reports. Alarm Report Messages have several different formats which the user selects via ACCOL Signals. The ACCOL signal #ALARM.FORMAT is used to select a Short/Long format for the Alarm Reports and is available in all versions of ACCOL and 33xx Firmware. ACCOL Version 5.4, and later, provides an additional signal, ALARM.FORMAT.001, which is used to select a standard or an Extended Report format. The Extended Report format also requires 33xx Firmware Version AE.00, or newer. (see the *ACCOL II Reference Manual* for details on configuring these signals.)

Extended Alarm Reports provide the status of an Analog, or Logical, Alarm signal's Inhibits bits as well as the value of the Alarm Limit exceeded for analog signals. Extended Alarm Report messages can pass through 33xx units with old firmware without a problem because the compaction algorithm used in these units works on the entire message rather than individual elements. However, older version Network Masters, which accept Alarm Reports, will be confused by the Extended reports.

It is possible for an Alarm Report message to contain both Standard and Extended report format elements, due to compaction occurring in different nodes within the network. This can occur if all nodes have not selected the Extended Report format. This presents no problem for either old or new 33xx firmware, but must be accommodated by Network Masters that accept Alarm Reports.

All alarm messages are BSAP Local messages. The format of each message follows:

ALARM INITIALIZATION REQUEST

ALARM INITIALIZATION RESPONSE

Return Error Response (RER)	1 byte RER always set to SUCCESS (0)

ALARM ACKNOWLEDGE REQUESTS

BSAP LOCAL HEADER	7 bytes
Load Version Number	2 bytes
Number of Elements (NME)	1 byte
Signal ith Address	2 bytes
Signal jth Address	2 bytes
. .	
. .	
Signal kth Address	2 bytes

ALARM ACKNOWLEDGE RESPONSES

BSAP LOCAL HEADER	7 bytes
Return Error Response (RER)	1 byte
Number of Elements (NME)	1 byte
Signal ith Address	2 bytes
Signal jth Address	2 bytes
. .	
. .	
. .	
Signal kth Address	2 bytes

where RER = 0, SUCCESS
32 (20h), LOAD VERSION MISMATCH
128 (80h), ILLEGAL SIGNAL ADDRESS

NOTE: If RER is NOT 0, then only the RER field is transmitted.

ALARM REPORT MESSAGES

BSAP LOCAL HEADER	7 bytes
Return Error Response (RER)	1 byte
Number of Elements (NME)	1 byte
Alarm element	
. .	
. .	
. .	
Alarm element	

where each alarm element contains the following fields:

Node's Global Address	2 bytes

Load Version Number	2 bytes
Signal Address (MSD)	2 bytes
Alarm Type code (AT)	2 bytes
Alarm Priority (AP)	1 byte
Time	5 bytes
Alarm data	6 or 15 bytes
Signal Name Text	21 characters maximum (NULL terminated)
Signal Descriptor Text	65 characters maximum (NULL terminated in Long Format only)

The following is a detailed description of each field contained in an alarm element:

AT - Alarm Type Code (Word). This code completely identifies the type of alarm being reported: Signal Type (Logical or Analog), Alarm Priority, Alarm Report State, Alarm Type (High, Low, True State, etc.), as well as an indication as to the validity of the Time Stamp. It has the following format:

Bits	Contents
2	Extended/Standard Format Indicator. 0 = Standard, 1 = Extended.
6	Alarm Re-Report Indicator.
10	Time Stamp Validity. 0=Valid, 1=Invalid.
14	Alarm Signal Type. 0=Analog, 1=Logical.

If (Alarm Signal Type = Analog) then,

3,7,11,15	Alarm State for Low, High, Low/Low and High/High Alarms respectively. 0=No Alarm, 1=Alarm.
0:1,4:5, 8:9, 12:13	Alarm Report State for Low, High, Low/Low and High/High respectively. 00=No Report, 01=Single, 10=Momentary, 11=Multiple.

If (Alarm Signal Type = Logical) then,

0:1	Alarm Report State (as above).
3	Logical Alarm State. 0=No Alarm, 1=Alarm.
7	Logical Signal Value.

Alarm Type Bytes - Contents

Analog Signal, bit 14 = 0

Alarm state bits 3, 7, 11, 15

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
H				L				HH	0			LL			

Time stamp validity (T) and Format (F)

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
					F				0						T

Alarm Report State

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
		H	H			L	L		0	HH	HH			LL	LL

State bits:
00 = None, 01 = single, 10 = momentary, 11 = multiple

Logical signal, bit 14 = 1

Time stamp validity (T), Format (F), Alarm Report State (RR), Alarm State (A), Signal State (S)

7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8
S				A	F	R	R		1				T		

T, F: see above
RR: 00 = None, 01 = Single, 10 = Momentary, 11 = Multiple
S: 0 = Off, 1 = On
A: 0 = No alarm, 1 = Alarm

AP - Alarm Priority (Byte). This field contains the Alarm Priority information of the Alarm Signal, and the Logical Alarm Type information (for Logical Signals).

Bits	Contents
If (Alarm Signal Type = Analog) then	
0:1, 2:3 ,4:5, 6:7	Alarm Priority for Low, High, Low/Low, and High/High Alarms respectively. 00=Event, 01=Op Guide, 10=Noncritical, 11=Critical.
If (Alarm Signal Type = Logical) then	
0:1	Logical Alarm Type. 01=False State, 10=True State, 11=Change of State.
2:3	Logical Alarm Priority (as above).

Alarm Priority Byte (AP)

Analog Signal

7	6	5	4	3	2	1	0
HH	HH	LL	LL	H	H	L	L

00 = event, 01 = op guide, 10 = non-critical, 11 = critical

Logical Signal

7	6	5	4	3	2	1	0
				P	P	L	L

PP: 00 = event, 01 = op guide, 10 = non-critical, 11 = critical
LL: 01 = False, 10 = True, 11 = Change of state

MSD - #Master Signal Directory Pointer (Word). This is the address of the Master Signal Directory entry for an alarm signal being reported or acknowledged.

TIME - Alarm Time Stamp (5 Bytes). This is the Time Stamp value for the alarm, containing the Julian Time/Date and a 20-millisecond interval count.

It has the following format:

Bytes	Contents
0:1	Julian Date: the number of days since December 31, 1976.
2:3	Julian Time: the number of 4

second intervals since 00:00:00.

5 Millisecond Count: the number of 20 msec intervals of the current 4 second interval.

DATA- Signal Data Value (Varying). This field contains the value of an alarm signal. Logical signals are represented as a 6 byte on/off text string; analog signals are represented as a 4 byte floating-point value and a 6 byte units text string.

XDATA Extended Data (Varying). This field contains additional alarm signal data. For Logical signals this field contains 1 byte for the signal's Inhibits Status. For Analog signals this field contains 4 bytes for the floating point value of the Alarm Limit exceeded, followed by 1 byte for the signal's Inhibits Status. If multiple alarm conditions are being reported, the precedence of the Alarm Limit exceeded value reported is High High, Low Low, High, and Low. If a Return to Normal is being reported, the limit value is set to 0FFFFFFFFh to indicate Not A Number (NAN). This field is only included if the Extended Format is selected.

The format of the Inhibits Status byte is as follows:

Bit	Contents
0	reserved
1	reserved
2	Questionable Data Status (Analog Only)
3	reserved
4	Manual Inhibit Status (1 = inhibited)
5	Control Inhibit Status (1 = inhibited)
6	Alarm Inhibit Status (1 = inhibited)
7	reserved

NAME Signal Name Text (Varying). This field contains the complete text of the signal name, Nul-terminated. The maximum length is 21 characters.

DESCR Signal Descriptor Text (Varying). This field contains the complete signal descriptor text, Nul-terminated, maximum 65 characters. This field is only included if the Long Format has been selected, otherwise only a NULL is sent.

Peer to Peer

Peer to Peer is a mechanism for data transfer between nodes on the Network 3000. Peer to Peer uses the Master and Slave modules which should not be confused with the BSAP Master/Slave communication scheme. Peer to Peer allows any node to be a Peer to Peer Master and/or Slave. The determining factor is the presence or absence of the Peer to Peer Master and Slave ACCOL module(s). A Master module is executed periodically at the request of the ACCOL task in which it is present. Slave modules operate asynchronously with respect to ACCOL tasks: when a command is received from a Master module, it is executed immediately. The Master module may initiate the following actions:

1. Send Mode - send data to a Peer to Peer Slave module.
2. Request Mode - request data from a Peer to Peer Slave module.
3. Both Mode - send data to, and request data from a Peer to Peer Slave module in one transaction.

The ACCOL programmer may specify signal lists or list elements to be transferred. The signals may be Analog, Logical or Strings. In addition, Analog or Logical Data Arrays may be transferred. There is NO implicit data type conversion by Peer to Peer. Any data transfer where a signal mismatch is detected results in an error code logged to the Master and/or Slave module terminal and the transfer is aborted.

The Peer to Peer Slave nodes are addressed by numbers with the following local addresses; -127 . . . +127. Positive numbers indicate that the node is a BSAP slave node relative to the node containing the Peer to Peer Master module. Negative numbers indicate a sibling node (i.e. at the same level on the network and having the same BSAP Local Master, where -1 corresponds to the sibling with local address 1). Remote address of zero indicates the Peer to Peer slave is the BSAP Local Master. Refer to Figure 2 for an illustration of the Peer to Peer addressing scheme.

Whenever possible the peer to peer Master should be located lower in the network hierarchy than the corresponding peer to peer Slave, since this will reduce the number of buffers required in the upper node. This is particularly important when several nodes are to move data upwards, as in data concentrator applications.

Peer to Peer messages may be Local or Global BSAP messages. Local BSAP formatted messages are used to send Peer to Peer blocks to Slave nodes. Global BSAP formatted messages are used to send Peer to Peer blocks to the Local Master node or to sibling nodes.

Should a Peer-Peer slave fail to respond, the master will wait for a default period of 2 minutes before repeating the request.

Peer to Peer Addressing Scheme

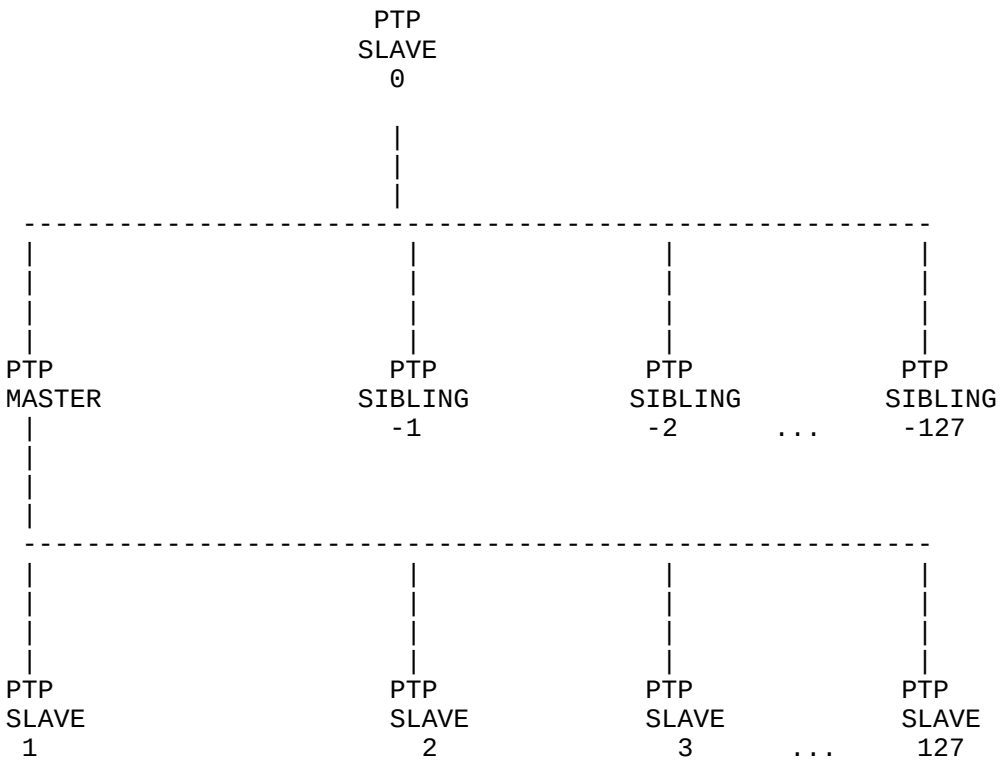


FIGURE 2

PEER TO PEER REQUEST MESSAGES

BSAP LOCAL or GLOBAL HEADER	7 or 12 bytes
Reserved	1 byte
Point	1 byte
Mode	1 byte
Input List Offset (ILO)	Mode 0-2 Mode 4-6 * 1 byte 2 bytes
Output List Offset (OLO)	Mode 0-2 Mode 4-6 * 1 byte 2 bytes
Number of Elements (NME)	Mode 0-2 Mode 4-6 * 1 byte 2 bytes
Data Packet(s)	2 to 234 bytes each, depending on data type (see packet description)

*Available with 'S' firmware. Transparent to user. Master Module Mode Terminal values are still:

0=Send
1=Poll
2=Send/Poll

New mode values and message structure are used when any array dimension, or the total number of elements in the array or list exceeds 255.

The ILO/OLO are used by the Peer-to-Peer routines to propagate POLL/SEND transactions when more than one buffer is required. Since these are low-level constructs, transparent to the ACCOL user, they are inherently zero-based offsets (i.e. similar to offsets in the 'C' Programming Language.)

For Poll Requests, the ILO is initialized by the Master and indicates to the Slave where to begin 'reading' data. A copy of the original ILO must be returned by the Slave in the response message.

For Send Requests, the OLO is initialized by the Master and indicates to the Slave where to begin 'writing' data. The Slave adjusts the OLO as data is written, and returns the OLO of the *next* element to be written, i.e. the OLO which the Master would use on the next 'Send' message to correctly propagate the 'Send' message.

PEER TO PEER RESPONSE MESSAGES

BSAP LOCAL or GLOBAL HEADER	7 or 12 bytes
Return Error Code (RER)	1 byte
Point	1 byte
Mode	1 byte
Input List Offset (ILO)	Mode 0-2: 1 byte; Mode 4-6 *: 2 bytes
Output List Offset (OLO)	Mode 0-2: 1 byte; Mode 4-6 *: 2 bytes
Number of Elements (NME)	Mode 0-2: 1 byte; Mode 4-6 *: 2 bytes
Data Packet(s)	2 to 234 bytes each, depending on data type (see packet description)

* Available in 'S' firmware.

Where:

POINT — Point number of the Slave Module that will receive the request, or that is responding to a request

MODE — Communication Type

Small Structures
0=SEND, 1=POLL, 2=BOTH

Large Structures (requires 'S' firmware)
4=SEND, 5=POLL, 6=BOTH

Small structures contain fewer than 256 total elements, or, for data arrays, have row/column dimensions less than 256. Any structures exceeding these limits are classified as large. These mode values are transparent to the user. The ACCOL Master Module MODE Terminal uses only the

values 0 through 2 to indicate communication type.

ILO — Offset into the Master's Input List (Slave's Output List) where the first data item in the transaction is to be stored (Master) or read (Slave).

NOTE: In response to a Poll message, ILO must contain a copy of the ILO from the Request message.

OLO — Offset into the Master's Output List (Slave's Input List) where the first data item in the transaction is to be accessed (Master) or stored (Slave).

NOTE: In response to a SEND message, OLO must contain the offset of the next element to be written. For example, if the Master writes 10 signals, OLO is 11 in the response.

NME — Total number of signal values contained in all data packets.

RER — Return Error Code. A detailed list of error codes is provided in Table 1.

PEER TO PEER DATA PACKET

1 byte	1 - 234 bytes (Mode = 0 - 2) 1 - 231 bytes (Mode = 4 - 6)
HEADER	DATA

Where:

HEADER - Bits 0-5 contain a count of the number of signal values contained in the data packet. The count value is actually one less than the number (0=1 value stored; 63=64 values stored(maximum).

Bits 6 and 7 indicate the type of data contained in the data packet. (0=Analog, 1=Logical, 2=String)

DATA - If data type is Analog there are 4 bytes per signal value (4 * (count+1)).

If data type is Logical there is 1 byte for each group of 8 or less logical values ((count+1)/8; 1 byte minimum). Logicals are packed from LSB (rightmostbit) to MSB (leftmost bit) in order.

If data type is String, there is a separate data packet for each string (strings have no NULL terminator). The packet header should contain the string length minus 1 count.

Examples:

A. One Logical Signal Value (minimum Data Packet)

Header: Type = 1 (Logical)
Count = 0 (1 value)
Data: 0000000X (1 byte)
NME = 1

B. One Analog Signal Value

Header: Type = 0 (Analog)
Count = 0 (1 value)
Data: 4 bytes (Floating Point value)
NME = 1

C. 1648 Logical Signal Values (26 Data Packets.) This is the maximum number of logical signals in a single communications buffer. Multiple response buffers are transmitted if more signals are to be transferred. 'S' or later firmware required.

First 25 data packets:

Header: Type = 1 (Logical)
Count = 63 (64 values)
Data: 8 bytes (8 values/byte)

Last data packet:

Header: Type = 1 (Logical)
Count = 47 (48 values)
Data: 6 bytes (8 values/byte)
NME = 1648

D. 58 Analog Signal Values (1 Data Packet

This is the maximum number of analog signals in a single communications buffer. Multiple response buffers are transmitted if more signals are to be transferred.

Header:	Type = 1 (Analog)
	Count = 57 (58 values
Data:	232 bytes
NME = 58	

NOTE: For large structures, the maximum number of analog signals in a single buffer is 57 because of the larger header area.

Peer To Peer Return Error Codes

The following status values appear at the STATUS 2 terminals of either the Master Module or Slave Module, or both. Those marked with an asterisk (*) can be reported from a Slave Module in the RER byte of the message format.

* 2	Signal was control inhibited or string was truncated
* 1	Slave has more data to send
* 0	Successful Completion
* -1	Invalid Mode (Mode Terminal must be 0, 1 or 2). If using 'S' firmware in the unit containing the Master Module, this error may indicate that you are attempting to use large structures to communicate with a Slave Module in a unit having an earlier firmware version, or the slave unit is a 3320.
-2	Invalid slave node number
-3	Invalid output list index (maximum index is 255)
-4	Invalid master input structure
-5	Invalid master output structure
* -6	Invalid slave input structure
* -7	Invalid slave output structure
* -8	Invalid slave point (no matching Slave Module)
* -9	Slave Module is not enabled
-10	Master input structure overflow
-11	Slave input structure overflow
-12	Both input and output slave errors
-13	Implicit type conversion attempted
-14	Communication send error (Master)
-15	Master 2-minute timeout waiting for slave response
-16	Communication send error (Slave)

-17	Invalid structure type (must be 0, 1, 2 or 3)
-18	Master has a zero length input/output structure
*-19	Slave has a zero length input/output structure
-20	No Peer Request Block allocated to MCB
-21	Communication buffer allocation failure
-22	Signal or array could not be updated because it was write protected, or a constant
-23	Node Routing Table not received (required for communications to master node or sibling nodes)

Remote Data Base Access

The Remote Data Base Access (RDB) feature provides a convenient and flexible way of reading the data elements of a Network 3000 node. This mechanism is extended to read all memory mapped locations including the Input ports, and may write to all memory mapped locations, excluding the Output ports.

There are five basic read operations:

1. Signal data - a signal may be read by specifying either its name or its address. Any signal present in the Master Signal Directory (MSD) is available to be read. However, each signal has a Read Security number associated with it and a RDB read access is checked for proper security level. Analog signals are transferred as 4 byte floating point numbers and Logical signals are transferred as 1 bit values.

2. Data arrays - Analog and Logical data arrays may be read by specifying the array number and row and column of the element to be read. There are no security checks associated with arrays. Analog signals are transferred as 4 byte floating point numbers and Logical signals are transferred as 1 bit values.

3. Select Signal Data - this function provides a means of selecting signals that match user specified criteria. The following criteria may be used: Control inhibit/enable; Manual inhibit/enable; Alarm inhibit/enable; Basename/index; Extension name/index; Attribute name/index; Extreme High Alarm state; Extreme Low Alarm state; High Alarm state; Low Alarm state.

4. Physical Memory - to aid in debugging a mechanism was incorporated to read arbitrary processor physical memory locations. It requires the full 24 bit address of the memory location.

5. Input ports - direct access to the Input ports are available. The user must specify the 16 bit offset address, RDB chooses the 4 bit segment address of the I/O address space.

There are three basic write operations:

1. Signal data - the signals may be selected by address, name or signal list. Signal data writes are security checked and subject to the inhibit/enable access bits (e.g. control, alarm, manual).

2. Data arrays - any element in a Read/Write data array may be written to. The user specifies the array number, row and column of the element to be updated. There is no security check for data array elements. Analog elements are 4 byte floating point numbers, and Logical elements are 1 bit (packed 8 to a byte) entities.

3. Physical Memory - Any RAM location may be written to by specifying the 24 bit physical address. It is the responsibility of the user to use this feature carefully to avoid disruption of system operation.

RDB messages may be Global or Local BSAP messages. The RDB request message format is dependent upon the type of operation requested in the FUNCTION byte. In addition the RDB response message format varies with the request message type. Descriptions of the various message formats used with/by RDB are provided in the appendix.

Immediate Response Mode

The Immediate Response Mode is a communication technique that is used to reduce the number of message transfers under certain circumstances. It is useful when a Request type message is sent to a node and a response is expected. Normally, the Request message would be followed by an ACK message, a POLL message, the Response message and ACK messages. During Immediate Response, the first ACK and subsequent POLL messages are NOT needed. The acknowledgment of the Request message is the Immediate Response message. NOTE: Not all Network 3000 devices are capable of generating an Immediate Response message. Currently, DPC 3330, DPC 3335, RTU 3310 controllers with PES03, PEX03, PLS03, PLX03, RMS04 or newer firmware, or RTU 3305 units with LS501 firmware, can generate these messages, provided they have been configured to do so, using advanced poll period parameters. Certain TeleFlow/TeleRTU units, as well as the 3320, 3740, and 3508 are also capable of generating an Immediate Response message. Although the RDC 3350 cannot generate an Immediate Response it is capable of processing one.

Time Synchronization/Node Routing Table

The Time Synchronization/Node Routing Table (TS/NRT) combined message enables each node in a Network 3000 configuration to know what the topology of the network is, the node's unique global address and the current date and time. The TS/NRT emanates from the Network Master Device and "trickles" down from

level to level until each node on the network has received it. There is NO application level acknowledgment to the TS/NRT message.

Once a node has received and validated an NRT, that node may receive and send Global messages. Without a valid NRT, the node discards all global messages it receives and it does not originate any. The NRT is created at the Network Master Device using the NETTOP utility program. The NRT may be modified at any time when a network configuration change has occurred. Once the NRT has been altered the new NRT is distributed to all nodes in the running system. Any global messages present in a node when a new NRT is being transmitted are discarded. It is assumed that the requesting unit will have an application request timer that will timeout and cause a new request to be sent to replace the discarded request.

The TS portion of the message is composed of the current date and time. The date consists of the calendar day, month and year. The time consists of hours, minutes and seconds.

The Time Synchronization/Node Routing Table message is defined as follows:

TS/NRT MESSAGE

BSAP LOCAL HEADER	7 bytes
Time Synchronization Message	12 bytes
Node Routing Table	20 bytes

where the Time Synchronization message is:

Day	1 byte
Month	1 byte
Year	2 bytes
Hour	1 byte
Minute	1 byte
Second	1 byte
Julian Day	2 bytes
Julian 4 second	2 bytes
Julian 20 millisecond	1 byte

Julian Day #1 = 1 January 1977 (Saturday)

and the Node Routing Table portion of the message is:

Version Number	1 byte
Global Address	2 bytes (calculated by receiving node)
UP/DOWN Mask	2 bytes (calculated by receiving node)
Current Level Number	1 byte (calculated by receiving node)
Level Shift Count L 0	1 byte
Level Bit Mask L 0	1 byte
Level Shift Count L 1	1 byte

Level Bit Mask	L 1	1 byte
Level Shift Count	L 2	1 byte
Level Bit Mask	L 2	1 byte
Level Shift Count	L 3	1 byte
Level Bit Mask	L 3	1 byte
Level Shift Count	L 4	1 byte
Level Bit Mask	L 4	1 byte
Level Shift Count	L 5	1 byte
Level Bit Mask	L 5	1 byte
Level Shift Count	L 6	1 byte
Level Bit Mask	L 6	1 byte

Users who are interfacing a foreign host typically access the NETFILE.DAT file to build a time sync message. For their convenience, the structure of the first few bytes of this file is included, below. Additional information on NETFILE.DAT is included in *Appendix E*.

Byte	0	Negative of max network level.
	1,2	Zero
	3-16	Mask and shift count, as found in NRT. There are 7 pairs in exactly the structure needed for the NRT. Levels 0 and 1 are always the same. The highest used level always has a shift count of zero.
	17	Current NRT version number.

In addition to receiving the TS/NRT message from the Master node, a Slave node will request a TS/NRT message following an initial download, or after a power-fail recovery. The TS/NRT Request Message is defined as follows:

TS/NRT REQUEST MESSAGE

BSAP LOCAL HEADER	7 bytes
Local Node Address	1 byte
Current NRT Version	1 byte

NOTE: The function code contained in the Local Header is 089h and the return function code is 088h.

Global Address Calculation

This section describes how a Network 3000 node determines it's global address and its UP/DOWN mask. The calculation uses the node's local address, level number, level shift count, level bit mask and the global address of the node's local master. As previously stated, the NRT trickles down from the network master to all nodes on the network. A master passes an exact copy of its NRT to each of its slaves. Each slave then uses the procedure described below to construct its NRT which it then passes to each of its slaves. Both the NRT global address field and UP/DOWN mask

starts off as zero (i.e. the network master's global address and UP/DOWN mask are by definition zero). As the NRT is distributed each node determines its global address and UP/DOWN mask and fills in the appropriate field in the NRT. Therefore the global address of a node is the concatenation of the local addresses of all previous local masters that the NRT has traveled to, from the network master to that node.

The following steps illustrate the calculation of a node's global address.

1. increment level number

2. node's local address shifted left Level_Shift_Count times

3. result from step 1 is OR'ed in with the current NRT Global address (i.e. local master's global address)

4. result from step 2 is now placed in the NRT Global address field (i.e. now it is the node's global address)

The following steps illustrate the calculation of a node's UP/DOWN mask.

1. node's Level_Bit_Mask shifted left Level_Shift_Count times

2. result from step 1 is OR'ed in with the current NRT UP/DOWN mask (i.e. local master's UP/DOWN mask).

3. result from step 2 is now placed in the NRT UP/DOWN mask field (i.e. now it is the node's UP/DOWN mask).

Download

The application level data (ACCOL load) may reside in either RAM or PROM. In FLASH-based units this data can be stored in FLASH memory from which it is copied to RAM for execution, providing some of the same 'permanent' load features as PROM. Downloading refers to transferring the user data to a RAM-based or FLASH-based system.

The user generates the application data using ACCOL Tools, e.g. the ACCOL Interactive Compiler (AIC), ACCOL Batch Compiler (ABC), or Windows-based ACCOL Workbench. The output of the compile and link process is a loadable file that can be downloaded to the RAM or FLASH memory of a 33XX node. It is also possible to process the file for transfer to a PROM programmer for units which support ACCOL PROMs.

For FLASH-based units, the download data is stored in the FLASH memory if the PROM/RAM switch on the CPU board (Switch Bank 1, Switch 6) is in the PROM position. Once the FLASH is loaded, the data is copied to the RAM for execution.

If the switch is in the PROM position when the unit is reset, it will copy and start execution of the load immediately after Selftest, without waiting for a download.

For 386EX Real Mode units, Switch Bank 1, Switch 5 is used as a LOCK/UNLOCK switch when Switch 6 is in the PROM position. By placing it in the LOCK position, the load in FLASH is protected from being overwritten by a new download until the switch is moved to the UNLOCK position.

For 386EX Protected Mode units, Switch Bank 1, Switch 4 is used as a LOCK/UNLOCK switch, if-and-only-if the configuration FLASH has been programmed, otherwise this feature is unavailable.

The Download process can be viewed from various perspectives which will be briefly described here:

Originator

Local download to node directly connected to the PC or other computer originating the download.

Global download to a node at a lower level in a network. In this case, the download messages are directed to the Command Handler in the node immediately above the target node. This intermediate node translates the messages to local Download Request Message Frames and handles communications with the target node. It also returns Command Handler response messages to the originator.

Intermediate Node

During a Global download, the master node immediately above the target node handles communications with the Originator and the Target, doing the necessary translation to local messages to the target node.

Target Node (Download Acceptor)

Download is always local at the target node. There are two environments which can exist when download is initiated, described as Cold Download, and Warm Download.

Cold Download - the node has been reset and has completed Self-test. The Watchdog LED is ON. The node monitors all ports for the start of a download sequence; baud rates and async/sync selection are based on defaults or switch settings, e.g. Ports A and C typically default to 9600 and Ports B and D are controlled by switches on the CPU Board. See the hardware manual for your unit for details. Nodes having special communications such as the CFE 3385 IEEE-488, or UCS 3380 LIU also monitor for download on those interfaces. Once the first download message (Extended Address Record) has been received on a port then

ONLY that port is available for the subsequent Download Data Records.

Warm Download - the node is executing an ACCOL load. When the Download Initiate message is received by the Command Handler, the node sends a Command Handler response and then forces a Watchdog. In this case the node does not execute Self-test. The download acceptor monitors for the start of the download sequence on the same port and at the same baud rate at which the Download Initiate message was received.

Notes:

Download messages will be NAK'd by the target node download acceptor unless the first message received is an Extended Address Record, i.e. the target node download acceptor must see the first frame of the download information.

If download transmission stops before completion, the download acceptor will timeout and begin monitoring all ports for the start of a new download. See the chart below for the timeout period for your unit type. If the sequence which timed out was a Cold Download, the same defaults and switch settings apply. If it was a Warm Download, the baud rate and configuration in use will be applied to all ports, e.g. if the Warm Download was started on an Async. port at 38.4 KB, all ports will be monitored at 38.4 KB following a timeout.

Download Timeouts

		Retry Conditions Determined By:	
CPU TYPE	TIMEOUT (approx.)	COLD DL	WARM DL
186 12mhz	12 minutes	Defaults and switches	Port on which DL Initiate msg.received
186 20mhz	6 minutes	"	"
386 RealMode 24 mhz	12 minutes	"	"
386 ProtMode 24 mhz	12 minutes	"	"

Message Formats

Download Initiate Command Message

The format of a Download Initiate message is as follows:

BSAP LOCAL OR GLOBAL HEADER	(DFUN = 70H) 7 or 12 bytes
Command Code	1 byte
Target Node Address	1 byte
Reserved	3 bytes

Command Code = 10H for local download (direct connect to Target Node)

11H for global download through Intermediate Node

Download Initiate Response Message

BSAP LOCAL OR GLOBAL HEADER	7 or 12 bytes	DFUN = SFUN from Command Message
Request Error Code (RER)	1 byte	
Delay Code	1 byte	

If RER = 1, unit has accepted the command and will Watchdog. RAM and FLASH-based units will wait for a new download; ACCOL PROM-based units will restart execution of the PROM-based load.

FLASH-based units with the PROM and LOCK switches set will return RER=0 and will NOT Watchdog.

NOTE: For local download, if the target node has already been reset, no response will be received to the Download Initiate message. The Originator should proceed to sending the initial Extended Address Record.

Delay Code: (386EX Protected Mode units with PLS03/ PLX03 / PES03/ PEX03 or newer firmware only): If present, the Downloader will wait for the amount of time specified by Delay code before proceeding with the download. This allows time for FLASH memory to be erased.

Download Data Messages

The generic format of a Download Data Message is shown below; there are 3 specific data record types. The Destination Function Code (DFUN) in the Header varies based on whether the message is global or local. For global download through an Intermediate Node, the DFUN continues to be 70H (Command Handler Request). For local download direct to the Target Node, (or for the messages from the Intermediate Node to the Target Node), the DFUN is 78H (Download Request Message Frames).

NOTE: A Global Download always requires a global header from the Originator.

BSAP LOCAL OR GLOBAL HEADER	7 or 12 bytes	Local - DFUN = 78H Global - DFUN = 70H
Command Code = FFH	1 byte	
Target Node Address	1 byte	
Data Record	n bytes (243 bytes max. msg. length)	max. n = 229

Where a Data Record may be one of the following:

Extended Address Record

Record Length = 2	1 byte	
Load Address = 0000H	2 bytes	
Record Type = 2	1 byte	
Segment Base Address	2 bytes	
Reserved	1 byte	
	(7 bytes total)	

Load Data Record

Record Length = n	1 byte	length of Data field
Load Address	2 bytes	Byte offset relative to Segment Base Address
Record Type = 0	1 byte	
Data	n bytes	max. data length = 224 bytes
Reserved	1 byte	
	(229 bytes max.)	

End of File Record

Record Length = 0	1 byte
Load Address = 0000H	2 bytes
Record Type = 1	1 byte
Reserved	1 byte
	(5 bytes total)

Download Data Response Message

If attached directly to the Target Node, the download acceptor will return a BSAP protocol ACK (DOWN-ACK) message (Function Code 86H) in response to each Download Data message.

If downloading through an Intermediate Node, that node will respond to the Originator with a global Command Handler Response as follows:

BSAP GLOBAL HEADER	12 bytes	DFUN = SFUN from the Request Message
Request Error Code (RER)	1 byte	

If RER = 0, SUCCESSFUL
1, Transmission Send Error

Diagnostics

Network 3000 uses diagnostics to assure node integrity at the system level. Diagnostics are invoked at startup during a Cold-start initialization sequence or from an external device (i.e. such as the PEI) during normal node operation.

During Cold-start the node performs a series of go/no-go tests. If the system passes each of the tests, then the download function may commence. However, if a failure results, the diagnostic test will terminate and an error code will be displayed on the CPU on-board LEDs; a download will not be accepted.

When the PEI controlled Diagnostics are run, the ACCOL program will terminate and the status of the test underway will be displayed at the external device. The external device communicates with the node under test using the BSAP protocol. Following completion of the diagnostic test(s) the node must be cold-started to resume normal operation.

The format of a Diagnostic Command is as follows:

BSAP LOCAL HEADER	7 bytes
Diagnostic Specific	variable

Diagnostic Response Message

BSAP LOCAL HEADER	7 bytes
Diagnostic Specific	variable

Chapter 5 - Transmission Modes

This chapter discusses Network 3000 physical link layer. Currently Asynchronous, Synchronous modes are supported.

Asynchronous

Asynchronous lines handle the following baud rates: 300, 1200, 2400, 4800, 9600, 19.2k, and 38.4k. The user must specify the baud rate during configuration—the software does not automatically detect and adjust to the baud rate in use. Each of the 33xx Process Controllers supports one or more of the following electrical interfaces (see specific product Technical Manuals for details). RS-423 and RS-232 electrical interfaces are supported providing point-to-point communication. In addition, RS-422 and RS-485 interfaces are supported which provides multi-drop interconnections. Only single master protocols are supported on an Asynchronous network.

Each asynchronous byte contains: one start bit, 8 data bits, and one stop bit. Parity is not used since message security is verified with a 16 bit CRC.

Bristol provides two types of Modems: a Dedicated Line Modem, and a Switched Network Modem (radio compatible) which may be connected to the public telephone system. Both modems support 300 and 1200 baud.

Synchronous

Synchronous lines provide either 187.5K baud or 1M baud communication rates. Not all 33xx products support synchronous communication lines (see ‘CI-xxxx‘ hardware manuals for details). The user must specify the baud rate during configuration -- the software does not automatically detect and adjust to the baud rate in use. Each of the 33xx Process Controllers supports one or more of the following electrical interfaces for synchronous communications. RS-422 and RS-485 electrical interfaces are supported providing multi-drop interconnections. In addition, the Redundant Asynchronous/Synchronous Communications Link (RASCL), which provides a dual redundant communications channel, is also supported. Only single master protocols are supported on an Asynchronous network.

Data Highway (for older model controllers)

The Data Highway is Bristol's Local Area Network (LAN). The transmission speed is 250 Kbaud; it may be configured for an end-to-end to span of up to 5 miles. A coax cable is used and can support multidrop capability. Although the Data Highway allows multi-master functionality it is constrained by the BSAP protocol for all upward and downward communications to be single master. However, lateral (e.g. Peer to Peer) communications allow multi-master coordination. That is, a node may directly initiate a communication transaction with another node on the same level without the local master intervening with a poll and then a forward of the message. The Data Highway provides dual redundant channels that are constantly monitored. In the event of failure an automatic switchover will occur; the communication statistics will be updated accordingly.

This page is intentionally left blank

Chapter 6 - On-Line Statistics

Network 3000 nodes retain on-line statistics reflecting the integrity of all communication transactions. An independent set of statistics is maintained for each communication line at each node. The data for a particular node and a specific line on the node may be analyzed on-line via ACCOL Tools or Open BSI (whichever is appropriate to the node type) during normal Network 3000 operation.

The type of statistics maintained for a communication line is dependent upon the modality type and whether the line is functioning as a Master or Slave. The next two sections will describe the statistics for each type of communication line. Interpretation of these statistics is covered in the ACCOL/Open BSI manual set.

Each statistic is kept as a 2 byte number. When any of the statistics reach 32000, then all of the statistics for the line are normalized. Normalization is simply a division by two of each statistic field for the line. Thus, reliable information on the operation of each line is always available without regard to the time when the counts were last reset.

The PEI supports the following system signals that provide additional information and control of the communication system. The "_#NODE.001 ... _#NODE.127" set of variables is user configurable, depending upon the number of slave nodes which are allocated for a given node. These variables are logical alarm variables, and are TRUE when the NODE is considered DEAD and FALSE when it is communicating. The "_#LINE.000 ... _#LINE.005" logical alarm variables correlate with the number of communication lines configured for the node (maximum is 6). For master lines, the "_#LINE" alarm is set TRUE if and only if all of the nodes on the master line are DEAD; it is set FALSE if at least one node on the line is communicating. For slave lines, the "_#LINE" alarm is TRUE if the corresponding port has not received a POLL or Data message from the local master within the user alterable poll period for that line (i.e. _#POLLPER.000 ... _#POLLPER.005) and FALSE otherwise. Finally, the _#NDARRAY identifies a logical array (optional) which controls the polling of slave nodes. The logical array is a bit mapped array such that when the bit is ONE the corresponding slave node may have a message sent to it, however, when it is ZERO the node will have NO message transmitted to it (i.e. POLL and Data messages will be throttled).

Asynchronous

The following table defines the statistics for an Asynchronous Slave or Pseudo Slave line.

Line Type	2 bytes
Number of Data Messages Received	2 bytes
Number of Data Messages Transmitted	2 bytes
Number of Polls Received	2 bytes
Number of Messages Flushed	2 bytes; occurs when a line is considered DEAD
Number of Not Acknowledges (NAKs)	2 bytes; occurs when no buffers are available
Number of Message Discarded ACKs	2 bytes

The number of messages transmitted should be smaller than the number of polls received by a factor of 1.5 or more. If they are approximately equal the line is overloaded -- it will work, but throughput may be slower than desired.

NAKs indicate that insufficient buffers were available to handle the messages received from the master. There are two common causes: the line is overloaded (see above) or additional buffers are needed at this node. Pseudo-slave ports will NAK if an attempt is made to have more than 3 requests (8 with AE and newer firmware) outstanding via that port.

Message discarded ACKs occur when a message is received by the slave but the master does not receive the ACK and therefore repeats the message. The slave detects the repetition, ACKs with discard, and discards the message thus avoiding duplication of the message. Usually due to a noisy line.

Messages are flushed (removed from the queue and discarded) when the master ceases polling for the specified timeout period as found in the associated _#POLLPER signal. (Alarms are never discarded.) If a NRT has been requested and not received, it will be re-requested after messages are flushed.

The following table defines the statistics for an Asynchronous Master line.

Line Type	2 bytes
Number of Data Messages Received	2 bytes
Number of Data Messages Transmitted	2 bytes
Number of Response Timeouts	2 bytes
Number of Consecutive Response Timeouts	2 bytes
Number of NAKs Received	2 bytes
Number of CRC Errors	2 bytes
Number of Message Disc. ACKs Received	2 bytes
Number of Protocol Errors	2 bytes; protocol errors include serial number, sequence number, or buffer overflow errors

Response timeouts occur when a message is damaged by noise or when a node fails (or doesn't exist). If all nodes exist and have not failed, then this is an indication of line quality. When the problem is due to noise, message discarded ACKs will typically be present on the same line.

Communications errors can occur in one direction and not the other. CRC errors, ACK discards, and protocol errors indicate errors in transmitting from slave to master. Timeouts can occur due to losses in either direction. NAKs indicate insufficient polling.

When the number of consecutive timeouts is greater than the number of slaves on the line, the line is considered dead.

Data Highway (used with older model controllers)

The following table defines the statistics for a Data Highway Slave port.

Line Type	2 bytes
Number of Data Messages Received	2 bytes
Number of Data Messages Transmitted	2 bytes
Number of Polls Received	2 bytes
Number of Messages Flushed	2 bytes; occurs when a line is considered DEAD
Number of Transmit Underruns	2 bytes
Number of No Acknowledges (NAKs)	2 bytes
Number of Transmit Link 1 Errors	2 bytes; Usually a modem or LIU problem
Number of Transmit Link2 Errors	2 bytes; Usually a modem or LIU problem
Number of Receive Frame Check Errors	2 bytes
Number of Receive Overruns	2 bytes
Number of Receive Link Aborts	2 bytes
Number of Receive Buffer Overflows	2 bytes
Number of Receive Link1 Errors	2 bytes
Number of Receive Link 2 Errors	2 bytes

Links 1 and 2 are redundant; the errors are reported separately so that repairs can be effected while communications continue on the unaffected link.

The following table defines the statistics for a Data Highway Master port.

Line Type	2 bytes
Number of Data Messages Received	2 bytes
Number of Data Messages Transmitted	2 bytes
Number of Response Timeouts	2 bytes
Number of Consecutive Response Timeouts	2 bytes
Number of NAKs Received	2 bytes
Number of CRC Errors	2 bytes
Number of Transmit Underruns	2 bytes
Number of Transmit Link 1 Errors	2 bytes; Usually a modem or LIU problem
Number of Transmit Link 2 Errors	2 bytes; Usually a modem or LIU problem
Number of Receive Overruns	2 bytes
Number of Receive Link Aborts	2 bytes
Number of Receive Buffer Overflows	2 bytes
Number of Receive Link 1 Errors	2 bytes
Number of Receive Link2 Errors	2 bytes

The Line (Port) Type is defined as follows:

0 - Line unused or statistics not kept
1 - Asynchronous BSAP Slave
2 - Asynchronous BSAP Master
3 - Asynchronous Pseudo Slave
4 - Data Highway Slave
5 - Data Highway Master
6 - IEEE-488 (reserved for CFE)

Chapter 7 - User Notes

This chapter provides information that may aid the user in configuring his network. Specifically, the selection of the poll period and the Peer-to-Peer rate will be discussed. For those users who have a PROM based load, the default Node Routing Table will be described. Also, the mechanism for interfacing to other devices that do not support the standard BSAP protocol will be discussed.

Selecting the Poll Period

Each communication line within a Network 3000 node has a user defined poll period (adjustable via #POLLPER.*nnn* system signals). For lines which are configured as a BSAP Master, this signal contains the period in seconds at which the poll cycle will be restarted. BSAP Slave lines or Pseudo Slave lines use the poll period signal to determine the time it will take before the communication line is considered Dead. This is the maximum amount of time since the last message, data or protocol, has been received. ACCOL uses twenty (20) seconds as the poll period default for any type of communication line. The user should define the poll periods first for all Master lines in the Network and then for all Slave and Pseudo Slave lines.

The poll period for the Slave node is simply set to a value 1.5 to 3.1 times longer than the poll period for the corresponding Master node. This will eliminate any possibility of a false indication of a dead communication line. There is one special case: AUX1 in the CFE (the IEEE-488 VAX link) should be set to 20 seconds.

IMPORTANT
Failure to set the master poll period to slave poll period ratio appropriately as described above is the single most common source of difficulty encountered with Bristol's communication system.

The choice of the poll rate for Master lines depends upon the following considerations:

1. BAUD rate and line turnaround time
2. Number and type of nodes on the line
3. Responsiveness to a request message
4. Maximum allowable time for a Peer to Peer Request Message, (2 minutes)
5. Node online array (#NDARRAY)

A poll period of zero for a master port is a special case which indicates polling has been turned off on that line. The poll period should never be set to zero for a slave port.

The BAUD rate for the line dictates the character by character throughput. For the Data Highway this is not a limitation due to the high speed of the line, (i.e. 250 KHZ). The master's poll period is typically set to 1 second for highways with up to about 12 nodes, 2 seconds for highways with 12-30 or so, etc.

However, for Asynchronous lines the BAUD rate becomes an important factor. For example, a maximum size message of 261 bytes, (i.e. 255 bytes plus the protocol characters DLE, STX, ETX, and CRCs) transferred at 9600 BAUD would take about 261 milliseconds to be transmitted plus approximately 12 milliseconds for the DOWN-ACK message. However, at 300 BAUD (lowest speed supported on an Asynchronous line) the same message transfer would take approximately 8.6 seconds for the message transmission and 400 milliseconds for the DOWN-ACK message. Most messages on a Network will be much smaller than the maximum size data message, however that depends on the application.

The next consideration is the number of nodes the Master line is configured for. This may be greater than or equal to the actual number of nodes installed on the line. In the case where there are Dead nodes on the line and the NRT has been received and validated, then the combined Time Synchronization/NRT message (55 bytes) will be sent to each Dead node rather than the smaller Poll message (10 bytes). Again, depending upon the BAUD rate selected this time could become a significant factor in choosing the line's poll rate.

In systems where node numbers have been allocated for nodes which will be added in the future, the _#NDARRAY should be used to eliminate polling of these non-existent nodes.

The desired responsiveness expected is an application dependent factor which is outside the scope of this document. However, it is important that this factor not be the primary driving force. The actual throughput could be less than anticipated if the other considerations mentioned are not analyzed.

When a node is master on a line the following formula allows the worst case poll period to be calculated:

(7650 / BAUD) * (_# of nodes on the line) = Poll Period in seconds

Example: For 9600 BAUD and 4 nodes ==> 3.18 seconds

Bristol provides a menu driven program, POLLPER, which estimates the poll period much more accurately than the above formula by taking message size, RTS/CTS delay, etc. into account.
If line turnaround is appreciable (due to modems and/or radios) this should be taken into account in calculating the poll period. For example, if a modem/radio combination

requires .2 seconds after RTS assertion before CTS is asserted, then .8 * (_# of nodes on the line) should be added to the poll period calculated above (4 * RTS/CTS delay is needed because there are four messages per transaction worst case). See the following section on Comm Hardware Timing Considerations for more info.

The poll period calculated above is worst case. System response can sometimes be improved by shortening the poll period, however shortening the poll period too much will actually degrade performance. To judge whether the poll period is set properly, the system must be operating normally with all nodes operational. Observe the receive and transmit LEDs for the line in question at the master node connected to that line: they should both be off simultaneously 30% to 50% of the time. If they are off more than this, you may shorten the poll period. If they are on almost continuously then lengthen the poll period. The dead time on the line allows handling of alarms, downloads, etc. with little impact on performance other than temporary reduction of dead time.

CAUTION:
This tuning technique won't work if some nodes aren't communicating --the timeouts which occur create periods of no activity on the line, leading to erroneous adjustments.

Lines whose slaves are exclusively 3320's require special consideration. Those models of 3320 which do not generate alarms need not be polled frequently since the 3320 uses immediate response mode. Polling is used to determine when a node has recovered from a communications failure rather than to move data, so a period of one to 5 minutes is often satisfactory.

Lines with a mixture of 3320's and 3350's also require special attention. In the typical case, peer-to-peer is used to communicate with each of the slave nodes. The rate or rates used for the master modules must be chosen such that each node receives a fair share of the time available on the line. To do this, keep the poll period and the master's rate approximately equal. Adjust them so that some dead time (judged by the LEDs, see section on tuning above) occurs during each poll period, typically 10% is appropriate rather than 30% when only 3350's are on the line.

The Peer to Peer application has a built in timer of two minutes. This is the amount of time allowed for the Request/Response message transaction to complete, as discussed in greater detail in the next chapter. Therefore, any Master line that supports Peer to Peer transactions must have a Poll period of less than two minutes.

Finally, a user definable logical data array, (see _#NDARRAY) is available to the ACCOL programmer. This bit mapped array allows the ACCOL program to selectively turn ON/OFF communication with any slave node during normal operation. This can alter the effective Poll period and hence message throughput.

Care must be exercised in using this facility to synchronize properly with application messages and polling.

Selecting Peer-to-Peer Rate

This section discusses the areas to consider when choosing the execution rate for the ACCOL Task that contains one or more Master Module(s). The ACCOL programmer needs to be aware of the application specifics and the poll rates for each of the communication line(s) that the Peer to Peer messages flow over. The application details are beyond the scope of this document and as such will not be discussed further.

Every time an ACCOL Task invokes a Master module, the module checks to see if it is busy (i.e. a Peer to Peer request is still in progress). When busy, the Task simply bypasses the module and goes on to the next one. The ACCOL programmer can cause a problem by invoking multiple Master modules more quickly than the resultant response messages can be acquired. Each Peer to Peer message requires certain system resources (e.g. communication buffers) to complete successfully. Unless logic is used to ensure that all master modules are finished before starting the next round, specific master modules may be randomly ignored for long periods of time.

The Peer to Peer system software associates a timer with each master request-response transaction. A value of two minutes was chosen as the maximum time allowed from the time a Peer to Peer request message has been sent till a Peer to Peer response message is received. In the case where multiple transactions are required to transfer a large signal list or data array, the two minute timer is restarted for each partial request message.

For Peer to Peer transactions between BSAP Local Master and Slave (i.e. remote node number is positive or zero) there is only one poll involved in a request-response transaction. However, Peer to Peer among siblings involves two polls. This should be taken into account when selecting the rate.

The following method will aid the ACCOL programmer in determining the appropriate Task rate. This selection process assumes that the Rate Task has NO other real time requirements except the execution efficiency of the Master Module. The programmer should first choose the poll period for the path(s) that the Peer to Peer request-response messages will traverse. Once the poll period(s) have been selected then the ACCOL Task rate should be set to execute two to three times more frequently than the poll period but not more frequently than twice per second. This scheme will minimize the time required between completion of one request-response transaction and the issuance of the next request. Placing the Master modules in a separate task or tasks should be considered whenever possible since it facilitates adjusting the timing.

Whenever possible the peer to peer Master Module should be located lower in the network hierarchy than the corresponding peer to peer Slave Module. This is particularly important when several nodes are to move data upwards, as in data

concentrator applications, because the number of buffers available limits the number of outstanding peer to peer requests from a node. Since the timing of messages returned is random with respect to rate execution, the several Master modules in a node may transmit unequal numbers of requests - in some cases particular Master modules may be excluded for periods of over a minute. By locating the Master modules lower in the network than the corresponding Slave modules there are fewer Master modules per node, reducing or eliminating the problem. Note that multiple Masters in a node is not a problem when communicating exclusively with 3320's and 3740's because they use immediate response. Hint: when several master modules are located in the same node, verify each master's status before proceeding to the next master module.

It should be noted that all Master Module requests are handled by a single system level Peer-to-Peer Master Task. Thus all Master Module requests are processed serially and are affected by any delays associated with the Master Task processing. (i.e. waiting for communications buffer, retry processing...) This is true even if multiple BSAP Master ports are utilized by the Master Modules.

When communicating with 3320's and 3740's only one peer-peer request will be outstanding at a time since the Peer-to-Peer Master Task must wait for each response before it can send the next. This causes an apparent interaction between ports when only peer-peer requests are being handled on those ports. Using two or more Master modules with 3320's will produce a smaller increase in throughput than with 3350's.

Communications Hardware Timing Considerations

The communications software provides delays to accommodate various configurations of communication hardware, including modems and radios. The delays which may be of interest when configuring a system are:

(1) RTS/CTS timeout = 2.5 seconds. When RTS is asserted, CTS must be asserted within this time or a hardware fault is assumed and RTS is dropped.

(2) Response timeout = 0.5 second plus the larger of: 50ms or 5 character times. This is the maximum time allowed between a transmission and the beginning of the subsequent response. This is adjustable from within the ACCOL Tools to provide more flexibility in configuring systems. The setting is in tenths of a second from 1 to 125. i.e. 12.5 seconds max.

(3) Inter-character timeout = 50ms. This was eliminated in the AE release to simplify support for satellite (VSAT) communications.

If modems are used with the delay jumper set for more than 200ms then the response timeout must be set appropriately or communications may not work at all. This is a common source of difficulty in systems which use modems.

When radios are used, the squelch should be set to suppress noise in order to avoid

delays incurred in determining whether the characters being received are a legitimate message or noise. Unsquelched noise will typically cause a pause of two seconds per unanswered master transmission while the noise/data determination is made.

Default Network Configuration Files

A default network configuration has been established to provide a basic Node Routing Table for executing the UCS3380/CFE3385 High Speed Data Highway diagnostic, and to provide a basic global message capability before a full network design is implemented. The configuration files used to create this NRT are provided as part of the ACCOL software release and reside in the ACCOL directory. In order to activate this NRT you must copy (or rename) the following network configuration files as indicated:

- NETFILE.REL to NETFILE.DAT
- RTUXREF.REL to RTUXREF.DAT

The default network configuration provides for a four level network with a Network Monitor as the network master (Level 0). A PEI or other unit type can be used without changing the Network Master type description. There can be up to 127 slave nodes on level 1. For each Master node on level 1 there can be up to 31 slave nodes on level 2. For each Master node on level 2 there can be 8 Slave nodes on level 3. Levels 4 through 6 do not exist when using the default network configuration. This network is sometimes referred to as 7-5-3, per the number of bits for each level.

The network configuration files define Node Names for various nodes in the configuration. The Node Name for the Network Master is L001. The Node Names L101 through L163 have been assigned for the first 63 nodes (node names for nodes 64 through 127 have not been assigned). The Node Names L201 through L231 have been assigned for the slave nodes below Node Name L101 (node names have not been assigned for nodes below L102 through L163). Node Names L301 through L307 have been assigned for the first 7 nodes below Node Name L201 (node names have not been assigned for nodes below L202 through L231).

If no network configuration files are found by the on-line program, then a default NRT is generated so that a response to NRT requests can be issued. This internally generated, default NRT provides a 7-7-1 network configuration.

This 7-7-1 default node routing table is also forced into use by any of the on-line ACCOL Tools when the '3308 FLAG' is set on the communications page and the address of the local node is 127. This case is intended to handle the situation where a user is attempting to communicate to an GFC 3308 whose address is set to the factory default of 127. Since the user's network configuration may not accommodate 127 nodes at this level, this facility allows the user to change the local address of the GFC 3308 to a valid address for the user's network configuration. The '3308 FLAG' should never be used if the GFC 3308 is a master to other nodes since the 7-7-1 NRT would

be sent to the lower nodes.

The NRT created from the default network configuration files is as follows:

Version Number	1-255 (assigned by NETTOP)
Global Address	Filled in by each node *
UP/DOWN Mask	Filled in by each node *
Current Level Number	Filled in by each node *
Level Shift Count L 0	= 8
Level Bit Mask L 0	= 07FH
Level Shift Count L 1	= 8
Level Bit Mask L 1	= 07FH
Level Shift Count L 2	= 3
Level Bit Mask L 2	= 01FH
Level Shift Count L 3	= 0
Level Bit Mask L 3	= 07H
Level Shift Count L 4	= 0
Level Bit Mask L 4	= 0
Level Shift Count L 5	= 0
Level Bit Mask L 5	= 0
Level Shift Count L 6	= 0
Level Bit Mask L 6	= 0

* See Chapter 4 for details on calculation of these fields.

The resultant global addresses that may be formed from the default NRT may be illustrated in the following way:

Interfacing to Foreign Hosts

This section discusses the requirements imposed upon a customer supplied Network Master. The most important thing to remember is that the Host must poll in order for data to be transferred to it. The polling may be selective, that is, it may allow alarm and data messages to be received or it may only allow data messages to be received. In the latter case, alarms will be throttled but the important thing to note is that the alarm system of the Network 3000 will still function even if the Host does not poll for them. In order to successfully transfer data the BSAP protocol must be adhered to (refer to Appendix A for a complete definition of the allowed BSAP protocol state

transitions).

The most common error is insufficient polling. Foreign hosts often attempt to send a request, then issue a corresponding poll to retrieve the result. This fails when noise damages the ACK to the request because the host thinks the message was lost, therefore not polling, while the 33XX honors the request. The host then issues a different request and polls, eliciting the response from the first request; checking the sequence number will reveal the problem, assuming the host is using sequence numbers. The only way to recover is to issue additional polls to ensure that all prior responses have been flushed. Immediate re-transmission of a request which is not ACKed (using the same serial number) will minimize the frequency of this problem - the 33XX will ACK and discard repetitions - but will not eliminate it. A secondary symptom of insufficient polling is slow response to requests on the pseudo-slave port of a node which is being polled infrequently on the slave port or vice versa.

One other point relative to polling: if a NAK is received in response to a transmission, poll the remote before re-transmitting. NAKs occur when no buffers are available for incoming messages, often due to insufficient polling. You should also ensure that the offending remote has sufficient buffers allocated (see following section for information on this point).

The Host is also responsible for supplying the combined Time Synchronization and Node Routing Table message. The Time Synchronization information is only required if alarms are polled for by the network host. However, for global messages to flow through the network a valid NRT must be supplied to all nodes. The NRT may be generated in one of three ways:

(1) Run NETTOP and transfer the table produced to the customer's host computer. See Chapter 4 for information on NETFILE.DAT.

(2) The customer writes a software program similar to NETTOP which can produce a NRT.

(3) Use the default NRT.

The host, as master on a BSAP link, is responsible for generating the link level serial number in a manner consistent with other nodes in the system. Serial number zero must be avoided; it is used for routing internal to the node. In all cases the serial number is used for a single message from master to slave and the associated response message. In particular, note that the UP-ACK uses a different serial number from the original poll message. See the example in the appendix. If a slave fails to respond to a message the master may re-transmit the message using the same serial number. If a slave fails to respond to a poll, the serial number must NOT be re-used.

The sequence number is part of the transport level protocol and may be handled as you see fit. It is normally used to detect whether the response received is from the last request sent or from a prior request which may have been delayed and should be

discarded because it may contain stale data. It is VERY useful in debugging problems with new systems via a comm analyzer.

Stand-alone Systems

In those particular applications that do not require a Network Master, the user may configure a Stand-alone system using a standard node at Level 0 in the network. In this case, the Level 0 node has one or more Master ports, but has no Slave port on which to receive the normal Node Routing Table/Time Sync message. When an ACCOL load without a BSAP Slave port is linked, the linker includes an NRT and the Master port(s) High Slave Address information in the load, using NETFILE.DAT in the ACCOL directory. This may be the default NRT if the user has activated it, or an NRT defined by the user using NETTOP.

The NRT will be propagated downwards in the network from the node at Level 0 immediately following start-up, and after a power fail recovery. The NRT is also used for polling a node which is on the Dead List (see Chapter 2 under 'Polling Philosophy'), and is sent to a node which requests it. A node requests an NRT immediately following start-up (Download or reset of a PROM-based unit), or following a power fail recovery.

The Time Sync. portion of the message will be derived from the time in the Level 0 node. If not set from outside the unit, this will be based on a default starting date and time of January 1, 1987, 00:00:00. The date and time can be set in the Level 0 node using the Keypad/Display device, an input device on a Logger port, or the ENCODE Module (see *ACCOL II Reference Manual*). The ENCODE Module date and time inputs can be set using a Pseudo-slave port. The date and time can also be set from a PEI using the Real Time Clock Offline Diagnostic Menu. After setting the correct date and time in the Level 0 node, turn its power off, then on. This will initiate transfer of the TS/NRT information down the network.

If the Level 0 load is placed in PROM, the user must establish the date and time on the initial start-up using one of the methods mentioned above. No default date and time are established by the firmware when a PROM-based unit is reset. This allows a PROM-based unit to retain a valid date and time, once established, as long as the battery backup of the RTC is maintained.

A Pseudo-slave port can also be used to monitor network activities, both locally and globally, from the Level 0 node. The same version of the NRT used to build the load must reside in the unit attached to the Pseudo-slave port. The Pseudo-slave port can be configured to report alarm information, but requires that the receiving device poll for alarms and have the capability to process them. The PEI does not poll for or process alarm reports.

ACCOL PROM-based Units

When the ACCOL load is placed in PROM, there are no special communications requirements. The load functions exactly as if it were downloaded into RAM with the following differences:

If the unit is reset, it begins load execution immediately following successful completion of Self-test diagnostics, instead of waiting for a download.

The Self-test diagnostics do not write the default date and time of January 1, 1987, 00:00:00, to the Real Time Clock chip in a PROM-based unit. If the load has a BSAP Slave port, the RTC will be set when a NRT/TS message is received from the node's master. If there is no BSAP Slave port (Level 0 node) the user must set the date and time on the initial startup.

If the unit receives a Download Initiate while it is running, it will reset and immediately restart load execution without executing Self-test diagnostics. The RAM is cleared before load execution begins.

Additional Buffers

The AIC compiler allocates communication buffers based on the configuration specified by the user. The algorithm used is based upon the number of communication ports and other factors that influence the use of communication buffers in a given node. The user may request additional buffers during the ACCOL program generation phase. The ACCOL Tools (Workbench or AIC) will attempt to provide as many of the additional buffers as it can, depending upon the amount of Random Access Memory (RAM) available (i.e. the size of the load file). Nodes which make heavy use of Peer-to-Peer communications will benefit from extra buffers -- one per Master module will help peer-to-peer considerably. Data Concentrator nodes typically benefit from additional buffers since they can then handle their store-and-forward functions more effectively. The CFE can use as many buffers as can be provided; use the memory size information provided by the ACCOL Tools to tune the load for maximum buffers. Overall, the system will operate reasonably well with the default number of buffers in almost every case, however, adding buffers won't hurt and it will always help the communications system at times of peak load.

Common Trouble Shooting Techniques

The following discussion is intended to assist the reader if a communication problem is suspected. Certain problems appear to be communication problems but are only symptoms and not necessarily the cause. For instance, if a node does not appear to be polled, or no communication is occurring on a particular line a simple cause may be the _#NDARRAY bit for the node in the local master may have the node deselected (i.e. bit set to zero) or, the max slave number may be set incorrectly.

As mentioned above, comprehensive statistics are kept for both Asynchronous and

Data Highway communication lines. A very good starting point in trouble shooting is to access the statistics for affected lines via the Open BSI Monitor. This implies that at least one slave or pseudo-slave port within the node under question is operational so that the PC may be connected. If this is not possible and it is a slave line that is in question then the local master's statistics may be analyzed to provide insight into the problem. Note that if a port has no communications hardware (i.e, Comm board) installed or if the installed board has been misconfigured (i.e. switches and/or jumpers are set wrong) the Open BSI Monitor will show an error condition for that port and individual statistics will be unavailable. The hardware MUST be installed and configured correctly. This is particularly important in the case of a BSAP Master or BSAP Expanded Master port where missing, misconfigured or failed hardware can affect the overall operation of an ACCOL load. A high number of Slave nodes and /or a short Poll-period can result in high CPU utilization by the Master port handlers causing lower priority tasks to run infrequently or not at all.

The onboard communication LEDs may be observed (both transmit and receive indicators) to see if any communication activity is occurring. If not, the communication cables and connectors should be checked.

On an asynchronous line a commercially available Data Line Monitor may be placed in series with the regular communication cables and a history of the communication transactions may be analyzed. One may see, for example, RDB request messages being transmitted but no response message returned. This may lead one to conclude that the RDB response node is receiving the request but having difficulty responding. A possible cause could be lack of communication buffers within the response node.

In general, the approach is to isolate the problem in the following order:

(1) Check the communication line, usually by observing the LEDs.

(2) Check the interface to the communication line, observe the modem's LEDS.

(3) Check and analyze statistics for the affected line.

(4) Check the software configuration within the node and the affected master or slave node, particularly max slave address and #NDARRAY.

(5) Check the Nettop file to ensure that it accurately describes the network configuration. Pay particular attention to the maximum number of slaves declared for nodes on the branch which is having difficulty.

In all cases the on-line communication statistics are most helpful in analyzing the problem.

Standards for Drawing Network Diagrams

Network diagrams are more quickly and easily understood if they are drawn according to the rules below:

(1) Master lines depart from the bottom of the box representing a node.

(2) Slave lines enter at the top of the box.

(3) Pseudo-slave lines enter on the side of the box, as do Loggers.

(4) The box representing a slave appears closer to the bottom of the drawing than the associated master.

(5) Nodes at the same level in a network are shown at the same level in the drawing.

These rules should cause a topology drawing to appear as the hierarchy that it is.

Notes on Modems and Radios

When using radios the modem timing must frequently be set longer than the default 8/25ms; 200ms is frequently required although some models will work with a 50ms delay. The response timeout may also require adjustment, see section 7.3.

In some cases activating the amplitude equalizer in the private line modem (S1-3) improves the reliability of 1200 baud communications considerably. Try it both ways and see which works best. The amplitude equalizer is most often helpful when using voice grade radios which have been adapted for use with data.

An aside on voice grade radios: the poll rate must sometimes be reduced in order to avoid over-heating the radio at the master -- usually not a problem at the slaves due to the lower duty cycle.

The amplitude equalizer is occasionally useful on dedicated lines - again, try it both ways if you are getting any appreciable number of errors on a 1200 baud line.

On radios with squelch, it should be set to suppress noise in order to avoid delays incurred in determining whether the characters being received are a legitimate message or noise. Unsquelched noise will typically cause a pause of two seconds per unanswered master transmission while the noise/data determination is made.

Port Interaction

Port interaction can occur due to RDB being single threaded. For example, if the Network Master is connected via the slave port and *another PC* is connected via the pseudo-slave port (Pseudo Master), then if the Pseudo Master is disconnected while it

has a request outstanding, requests to RDB via the slave port will not elict a response until the pseudo-slave port times out. This occurs because RDB's output buffer is tied up until it is released by being transmitted or it is timed out. This effect is most noticeable when long poll periods are specified.

Another interaction occurs on pseudo ports with alarms. Here, the rate of alarms returned is limited by the slowest port because the alarm buffer is sequentially queued to each port through which alarms must be reported.

This page is intentionally left blank

Chapter 8 - Communication Code Definitions

Communication Function Codes

The following is a list of function codes supported by Network 3000 nodes. These codes constitute the transport level identification for end-to-end message routing. One may think of them as the number of the mailbox for messages to be deposited in once they are received at the desired node.

Application Function Codes (HEX)	Description
00	ILLEGAL
03	On-line PEI messages for PC
40	On-line PEI messages for RTU
48	Diagnostics
50	3508 Transmitter request
51	3508 Transmitter response
60	CBO
68	Flash download
6A	Flash Configuration
70	Command Handler Requests
71	Command Handler Responses
72	Reserved (used internally only for EMaster)
78	Download Request Message Frames
88	TS/NRT Message (for SLAVE ports only)
89	Request for TS/NRT Message
98	Pseudo Slave passthru for Task # 1 (33xx only)
99	Pseudo Slave passthru for Task # 2 (33xx only)
9A	Pseudo Slave passthru for Task #3 (33xx only)
9B	Pseudo Slave passthru for Task #4 (33xx only)
9C	Pseudo Slave passthru for Task #5 (33xx only)
9D	Pseudo Slave passthru for Task #6 (33xx only)
9E	Pseudo Slave passthru for Task #7 (33xx only)
9F	Pseudo Slave passthru for Task #8 (33xx only)
A0	Remote Database Access (RDB)
A1	RDB Extensions
A2	Report By Exception (RBE) Firmware Task
A3	Report By Exception (RBE) Manager
A8	Alarm Acknowledge
A9	Alarm Initialization
AA	Alarm Reports
AB	SPARE (reserved for the Alarm system)
B0	Peer to Peer Master Task
B1	Peer to Peer Slave Task
BF	Tunnel requests (CW Designer through BSAP)
C0	IP Processing Task – routes BSAP frames
C1	Pseudo Slave Pass-thru message for COM1 (CWave)
C2	Pseudo Slave Pass-thru message for COM2 (CWave)
C2	CFE Template Manager Request messages (33xx)
C3	Pseudo Slave Pass-thru message for COM3 (CWave)
C3	CFE Template Manager Multiple messages (33xx)
C4	Pseudo Slave Pass-thru message for COM4 (CWave)
C5	Pseudo Slave Pass-thru message for COM5 (CWave)
C6	Pseudo Slave Pass-thru message for COM6 (CWave)

C7	Pseudo Slave Pass-thru message for COM7 (CWave)
C8	Pseudo Slave Pass-thru message for COM8 (CWave)
C9	Pseudo Slave Pass-thru message for COM9 (CWave)
CA	Pseudo Slave Pass-thru message for COM10 (CWave)
CB	Pseudo Slave Pass-thru message for COM11 (CWave)
D0-EF	Reserved for Users

The following function codes are reserved for protocol messages:

Protocol Function Codes (HEX)	Description
81h	DIAL_UP_ACK response to first poll message received following a slave initiated dial-up sequence. (For use with Open BSI)
83	ACK, Message Discarded
85	Poll message
86	Down Transmit ACK, Slave ACKing message
87	ACK No Data, in response to a Poll message
8A	UP-ACK with poll (recognized by VSAT Slave)
8B	UP-ACK, Master ACKing message
95	NAK, No buffer available for received data

Communication Error Codes

Error codes are placed within the NON-TRANSMITTED portion of the message by the Network 3000 software. The following are the supported internal communication error codes:

Error Codes (HEX)	Description
2	Transmission is complete and successful
1	Transmission is in progress
-1	Bad software calling parameters
-2	Master receive buffer overflows
-3	Master receive timeout
-4	Master received Nak message
-5	NRT version mismatch
-6	Serial number mismatch detected by Master
-7	Invalid protocol message received by Master
-8	CRC error detected by Master
-9	LIU error detected

All other error codes are application in nature and are reported in the response message or request message. These error codes have been listed in this document under the applicable chapters.

Sample Communication Transactions

To illustrate the BSAP protocol in operation a Data Line Monitor was used on an Asynchronous line to capture a series of communication transactions. The excerpt that is reproduced here illustrates an RDB global request by name message, protocol messages, the RDB response message, RDB global request by address of the same signal, and the protocol message. The actual serial numbers, sequence numbers and checksums are provided along with the NULL, line initiation byte. All numbers are shown in Hexidecimal.

1. RDB REQUEST - READ BY NAME from Network Master to node 0420H

Characteristics of message:

local address field = 81; 80=global message, 1=node address
serial number = 24
global dest address = 0420
source dest address = 0000
control byte = 0
destination function code = A0
RDB sequence number = CF55
source function code = 3, PEI
node status byte (NSB) = 0
RDB function code = 4

```
0,10,2,81,24,20,4,0,0,0,A0,55,CF,3,0,4,3,F3,81,F,1,43,53,31,53,44,48,
|---------BSAP----------------------|-READ BY NAME-|C S  1  S   D  H

56,2E,43,4C,4F,53,45,44,2E,0,0,31,0,0,0,10,3,C0,CD
 V .   C  L O  S  E  D  .                  |-CRC-|
```

2. PROTOCOL DOWN TRANSMIT ACK

Characteristics of message:

same serial number (24), as request message
local address = 0; headed for local master

```
0,10,2,0,24,86,1,4,5,10,3,77,64
```

There are intervening poll messages to other nodes and then the poll message to this node Local address = 1

3. PROTOCOL POLL MESSAGE

Characteristics of message:

serial number = 30 ; ie 12 messages have been sent
local address = 1; headed for slave number 1

```
0,10,2,1,30,85,0,10,3,F4,70
```

4. PROTOCOL ACK/NODATA MESSAGE

Characteristics of message:

local address = 0; headed for local master
serial number = 30, ACKing the poll message just received

```
0,10,2,0,30,87,1,0,4,10,3,96,47
```

5. PROTOCOL POLL MESSAGE

Characteristics of message:

local address = 1; headed for slave number 1
serial number = 31

```
0,10,2,1,31,85,0,10,3,4F,6C
```

6. RDB RESPONSE MESSAGE

Characteristics of message:

local address = 80, global message to go UP
serial number = 31, same as poll message just received
dest global address = 0000
source global address = 0420
sequence number = CF55, from RDB Request message
rer = 0, success
nme = 1, one signal requested
type = 4, logical alarm
value = 0, signal is OFF

MSD signal address = 2EEA

```
0,10,2,80,31,0,0,20,4,40,3,55,CF,A0,0,0,1,4,0,20,20,20,20,20,20,
|----- BSAP ------------------------|----------RDB----------

EA,2E,43,4F,4D,50,20,31,20,53,2E,20,44,53,43,48,20,56,4C,56,0,43
|ADR|  C  O  M  P    1     S  .      D S  C  H     V  L  V    C

4C,4F,53,45,44,20,20,20,20,20,20,71,B3,37,10,3,5E,8B
V  O  S  E  D
```

7. PROTOCOL UP/ACK MESSAGE

Characteristic of message:

message has new serial number = 32
within message is serial number of message being ACKed = 31

```
0,10,2,1,32,8B,31,10,3,E3,F6
```

8. PROTOCOL DOWN TRANSMIT ACK

Characteristics of message:

local address = 0; headed for local master
same serial number (32), as UP/ACK message

```
0,10,2,0,32,86,1,0,4,10,3,84,44
```

9. PROTOCOL POLL MESSAGE

Characteristics of message:

local address = 1; headed for slave number 1
serial number = 33

```
0,10,2,1,33,85,0,10,3,39,55
```

10. PROTOCOL ACK/NODATA MESSAGE

Characteristics of message:

serial number = 33, ACKing the poll message just received

```
0,10,2,0,33,87,1,0,4,10,3,EB,4B
```

11. RDB REQUEST -READ BY ADDRESS from Network Master to node 0420H

Characteristics of message:

local address field = 81; 80=global message, 1=node address
serial number = 34
global dest address = 0420
source dest address = 0000
destination function code = A0; RDB
sequence number = CF62
source function code = 3, PEI
node status byte = 0
RDB function code = 0

```
0,10,2,81,34,20,4,0,0,0,A0,62,CF,3,0,0,1,C0,B3,37,F,1,EA,2E,10,3,B6,A6
|---------BSAP----------------------|                                |MSD|
                                ADR
```

12. PROTOCOL DOWN TRANSMIT ACK

Characteristics of message:

local address = 0; headed for local master
same serial number (34), as request message

```
0,10,2,0,34,86,1,4,4,10,3,1F,3F
```

Node Status Byte

The Node Status Byte appears in several of the messages described throughout this document. It is used to inform the master of certain conditions which exist within the slave. The state of unused bits are indeterminate. Network 3000 controllers use all of the bits shown, ControlWave-series controllers only make use of bit 0 and bit 5. These conditions, and their corresponding bit positions are:

0 - Unreported alarms exist
1 - Discrete Input (DI) change exists. Set the very first time a DI change occurs. Never reset. (unused).
2 - Slave has more data to send (Up transmits pending). This essentially means that additional messages are ready to be polled by the host. The host can look at this bit to determine whether it should poll at a faster rate to collect the pending data messages quicker.
3 - Power failure recovery in effect (unused)
4 - Download reception in progress
5 - Message from a VSAT slave port
6 - Destination message exchange not found (unused)
7 - Communications failure (unused)

This page is intentionally left blank

Appendix A
Bristol Synchronous/ Asynchronous Protocol (BSAP)

1. Introduction

This document defines the network addressing as well as the asynchronous communication protocol used in a variety of Bristol products.

2. Overview

A layered communication protocol is presented which is applicable to asynchronous lines. The protocol presented is compatible with the ISO 1745/2111/2629 standards.

The addressing technique described is suitable for hierarchical networks, i.e. those networks with progressively fewer nodes at higher levels of the network. The technique presented is applicable to networks having six or fewer levels.

A message switching network results when the hierarchical addressing scheme is used with the proposed protocol; packet switching, in which a message is broken into two or more pieces for transmission, is not covered or anticipated.

A hierarchical network interconnected with low speed (1200 baud) lines is anticipated as a target application. To achieve reasonable communication throughput, many communication lines must be simultaneously active. This implies decentralized control in which nodes autonomously collect and concentrate information from their slaves. In this type of operation only local communication is needed between directly connected master/slave nodes.

The bulk of communications are expected to occur as outlined above; however, a relatively small but very important fraction of communications are associated with command and control functions.

Command and control messages typically flow between the highest node in the network and any other node. The message switching capability provided is expected to be used primarily for command and control functions.

3. Async Communication Protocol

The asynchronous protocol presented here is appropriate for transparent message switching applications utilizing hard wire, telephone, and radio communication links. Message security is provided by a 16 bit CRC code.

The technique chosen meets the ISO 1745 standard (async character oriented) as extended by ISO 2111 (transparent messages) and ISO 2629 (conversational mode). The ANSI X3.28-1976 standard, which is effectively a subset of the ISO standards,

does not cover the proposed technique.

3.1.Protocol Layering Overview

The technique chosen accommodates the bottom four levels of the ANSI/ISO model:

Physical control
Link control
Network control
Transport end-to-end

Physical level will be RS232, RS422, RS423, or RS485 and is outside the scope of this specification.

Link level contains the control functions needed to reliably transfer data across a single communication link. Link level uses codes and methods which fall within the ISO standards listed earlier.

Network level contains the control functions required for network addressing and routing. The addressing scheme described in a following section is used here.

Transport level contains the functions necessary to provide a network independent interface to users of the transport service. This is application dependent; in our system it includes message exchange ID, message sequence number, etc. The specification of the complete transport level protocol is outside the scope of this specification.

The goal of a layered protocol system is to provide a modular communications mechanism in which each level need only know how to deal with adjacent levels.

3.2. Link Level Protocol

There are two major sections to the link level protocol:

Data structures transmitted
Line handling procedures

3.2.1. Link Level Data Structures

The structures used for message blocking are those provided in ISO 2111, which are identical to those provided in ANSI X3.28 transparent mode.

In transparent mode each control character must be immediately preceded by a DLE (Data Link Escape). The control characters which must be protected in this manner are:

STX - start of text
ETX - end of text
SOH - start of header

In addition, if a DLE appears within the message a DLE must be inserted adjacent to it during transmission. On reception, if two consecutive DLE's occur the first is discarded.

Each BSAP message will begin with the following sequence of four bytes:

DLE
STX
Local address
Serial number

Each Expanded BSAP message has the following starting sequence:

DLE
SOH
Group number
Local address
Serial number

The group number is used at the link level only; it is not propagated with the message to subsequent levels.

The DLE STX or DLE SOH sequence is prescribed by the ISO standard.

The high order bit of the local address is a flag indicating (if on) that the message includes a network level protocol structure (see Figure 1, Global Addressing Mode). The low order 7 bits are the address of the destination node on this link.

The serial number is an 8 bit number assigned by the master which must be returned in the response from the slave. It is used to avoid confusion when a message or acknowledgement is lost due to noise, etc. Serial number zero is not used in transmission; it is reserved for internal use at the node.

Each message terminates with the following four byte sequence:

DLE
ETX
CRC - high order byte
CRC - low order byte

This complete sequence is prescribed by the ISO standard.

3.2.2. CRC Calculation

The CRC under the ISO standard is CRC-CCITT which uses a generator polynomial with the (16, 12, 5, 1) terms.

CRC calculation will begin following the STX in the initiating sequence. The first

DLE in each two character DLE sequence (DLE DLE, DLE ETX) will not be included in the CRC calculation. The CRC calculation will terminate following the ETX in the terminating sequence.

3.2.3.Link Level Line Handling Procedures

Since the use of the network is master query and slave response the line handling procedure is simplified for our conversational protocol. The line establishment procedure is a combination of establishment and message text which can be used to interchange master/slave information messages in a fast conversational manner, where the positive acknowledgment of a master's request is the slave response.

If the master does not receive a response to its request or the message is NAK'd, the master will retransmit the message two times.

Message Flow:

The master initiates a request to a slave node. The message flow procedures are the following (refer to Figures 1,2,3)

Global Addressing:

If Global addressing and the message must pass through other nodes on the way to its destination - The request is routed to its proper intermediate node Fig. 3 (1).

The pass-thru node will send a 'DOWN TRANSMIT ACK' to the master with the same message serial number as in the request and local address of 0 which indicates local master Fig. 3 (1),(2). The message is then routed to the next level down via new local address, derived from the routing address, with a new serial number generated by this node.

If the slave response is not immediate, the local master must poll the slave for a response Fig. 3 (4), If 'NO DATA TO TRANSMIT' slave response then this ends the poll sequence Fig. 3 (4),(8). Upon a slave response to a poll Fig. 3 (4),(5) the master will send the slave an 'UP TRANSMIT ACK' to acknowledge reception of the slave response and no further polling therefore ending the sequence Fig. 3 (6),(2) or respond with 'UP TRANSMIT ACK' with poll for more information Fig. 3 (7).

The slave will send either a 'NO DATA TO TRANSMIT' to end the sequence Fig. 3 (7),(8) or another poll response Fig. 3 (7),(5) and continue master polling and slave responses until a final 'NO DATA TO TRANSMIT' is sent to the local master Fig. 3 (7),(8). The slave poll response could cause the local master to send an 'UP TRANSMIT ACK' with no poll Fig. 3 (5),(6) and the sequence is ended with a slave 'DOWN TRANSMIT ACK' response Fig. 3 (6),(2).

After processing at the global destination node is complete a response must be sent to the message originator. To do this the destination and source message addresses are interchanged, causing the response message to be returned to the originator. The affirmative acknowledgment to the request is the match of the transport level sequence number of the original request and the final response.

If the slave response to the master's request is immediate then an 'IMMEDIATE RESPONSE' is sent with the same transport level sequence number as the master's original request transport level sequence number Fig. 3 (1),(3).

Local Addressing:

If Local addressing - the request is sent to its local slave node Fig. 3 (1).

The slave responds with either a 'DOWN TRANSMIT ACK' as described above in global addressing item number 2 Fig. 3 (1),(2) or an immediate response with the same transport level sequence number as the request message Fig. 3 (1),(3).

If a "DOWN TRANSMIT ACK' is sent the master continues polling as described in global addressing Fig. 3 (4),(8) or 3 (4),(5).

3.3.Network Control Protocol

The network control protocol segment of a message is optional. When present it contains network routing and message status information.

Omission of the network level protocol segment is detected on a per message basis by examining the high order bit of the local address byte contained in the link level protocol. The link level handler then transfers the message to the network level or transport level handler as appropriate.

3.3.1.Network Control Data Structures Transmitted

The network control section of a message, if present, contains the following:

Destination address - 2 bytes
Source address - 2 bytes
Control info - 1 byte

The destination and source address structures are described in a following section.

The control byte may contain the following:

Message type (request or response)
Return status - network level number if error response

3.3.2.Network Control Procedures

Network control procedures using the message Destination address are described in the 'Algorithm for Routing' pseudo-code. To route the response to the master node the source and destination addresses are interchanged thus allowing the same algorithm

to be used to route up the network as is used to route down.

In the case of more than one slave line the RTU 3350 must select the proper slave line to transmit messages to the slave node. This is done via the NODE ROUTING TABLE, described below. Once the line is selected, the message link level local address must be adjusted to allow further downward transmission. In global addressing mode, as a message decends to the next level the next level down address, which is contained in the destination address, is extracted and put into the link level local address and the global mode flag set. Since the message has now been altered the CRC must be recalculated to ensure message security.

3.4. Transport Control End-to-End

The first byte of the transport end-to-end message will contain the code for the message exchange.

Bytes two and three contain the message sequence number which is required to avoid accidentally duplicating messages under noisy conditions.

Byte four contains the message exchange for the response.

Byte five contains the response status; if a communication failure occurs, the high bit will be set and the remaining bits will indicate the type of failure.

If a length byte is required for a particular message type, it should immediately follow the response status byte.

Other transport control structures and procedures are based on prior agreement between the nodes and are beyond the scope of this specification.

3.5. Error Recovery Procedures

Error recovery procedures are not covered by the ISO standard; errors are typically handled using techniques which are mutually understood by all equipment connected to a line. The ISO standard does not prescribe error handling techniques because the variety of problems which can arise is so large and the circumstances so diverse that a general recovery method would be unwieldy.

The Bristol Standard Asynchronous Protocol is designed to minimize the need for timing constraints during error recovery. This tends to simplify error recovery and also reduces the number of points of mutual agreement required among processors for error recovery.

The remainder of this section presents some suggested techniques for recovery from common types of errors. These are only suggestions and are not part of the standard protocol.

3.5.1. Re-transmission

All re-transmissions by a master (due to timeout) which are associated with a message should be accomplished without intervening transmissions of other messages on that line by the master.

The master may transmit to other slaves before re-transmitting a message to a slave which NAK'ed due to lack of a buffer.

3.5.2. Response Timeout

When a message is damaged in transit (e.g. due to noise) it will be ignored by the intended destination node. This particular area requires a message timeout facility for recovery.

When the lost message is from master to slave, the master will retransmit after the agreed upon time interval expires; the serial number will not change.

When the lost message is from slave to master the result is the same as above. The slave will retransmit the same response (after receiving a retransmitted request from the master) if the response was a data message rather than a Down Xmit ACK. If the lost response was a Down Xmit ACK, then the new message from the master will be discarded based on duplication of serial number and a Down Xmit ACK with an indication that the repeated message was discarded will be sent to the master. The master will accept this if the previous master to slave transmission was a retransmission.

Otherwise, the master will insert a new serial number into the message and send it to the slave again (this is not a retransmission since the serial number differs).

This technique avoids problems which might occur due to modulo 255 serial numbers while avoiding timing of serial number usage.

3.6. Protocol Design Notes

Some of the rationale used in constructing the Bristol Standard Asynchronous Protocol is presented here so that readers may understand the motives behind certain choices.

Transparent mode data transmission was chosen for efficiency of line use and for simplicity - encoding/decoding to/from hex ASCII or similar schemes is avoided at the expense of DLE insertion.

Conversational mode was chosen because it halves the number of transmissions in certain common cases.

ISO 1745/2111/2629 was chosen over ANSI X3.28 because transparency and conversational modes are mutually exclusive under X3.28. The ISO standard is a

superset of X3.28 so the DLE STX sequences, etc. are identical.

Serial number insertion is required each time a message is transmitted to prevent message loss or duplication when line faults occur.

CRC recalculation at each store and forward node is required due to serial number insertion. CRC calculation takes 56ms in a Z80; expect about 8ms in the 80186.

Local addressing was included to reduce overhead. It is particularly suited to single level networks where global addressing would be quite inefficient. Local addressing also simplifies the communication techniques imposed on low level remotes.

Up Xmit ACK is included so that each global data transmission receives an immediate acknowledgement. This also shortens the time a slave must retain a buffer for possible retransmission.

4. Network Addressing

Process control applications frequently use data collection and compression at each level in a network, that is, data concentration. When data concentration is used extensively in a network application the resulting network structure is typically hierarchical. This structure is appropriate where central control and/or monitoring of a widely distributed process is required. The interconnection of nodes and the number of levels in the hierarchy is based primarily on the interactions required between nodes, their physical proximity, and the cost to install and maintain the communications mechanism.

Since hierarchical network structures are well adapted to the process control applications to be considered, hierarchical addressing was selected as being most appropriate for this network structure.

4.1. Hierarchical Addressing Overview

In a hierarchically addressed network the required routing is inherently specified by the address. The node address is effectively a list of all the nodes a message would pass through to travel from the top of the network to the node. There are two major ramifications to having the address imply the routing: each node address is dictated by the point at which it is attached to the network, and the network is singly connected, i.e. there are no redundant connections.

The routing tables required for hierarchical addressing are minimal and the routing control algorithm is straightforward, compact, and executes quickly.

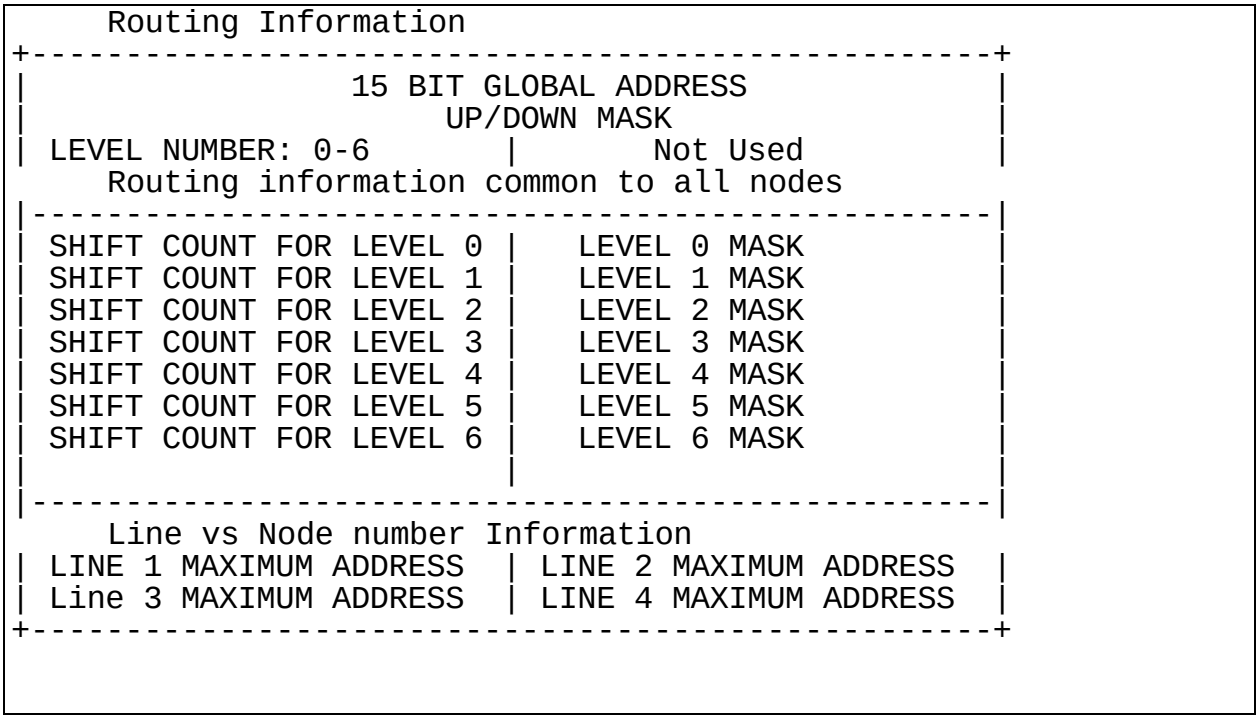
4.2. Hierarchical Addressing Design

This section provides the details of the network addressing structure as well as

describing the routing algorithm in detail.

The length of the address field is a tradeoff between addressing flexibility and transmission overhead since the address must be included in each message. A 16 bit address field was selected as a reasonable balance; an alternative 8 bit addressing mode is provided to allow compact addressing of local nodes.

4.3.Node Routing Table



4.3.1. Contents of the Routing Table

There are three sections to the node routing table:

Node specific routing information.
Routing information common to all nodes.
Line vs node number information.

Each of these parts of the routing table will be described in a following section.

Node Specific Routing Information

Local Address -

The local address is read from switches in the 3350. It is the node's slave address for use in receiving local transmissions from its master.

Version Number-

The routing table version number is revised by the program which prepares the routing table each time the table is changed in a way which modifies the number of levels, or maximum node address at a level. It is used to ensure that all nodes are using consistent routing tables; it is checked each time that a time sync occurs.

15 Bit Global Address

The global address is the complete address required to route a message to the node from anywhere in the network. It consists of a list of the nodes which a message would pass through in moving from the network master to the target node. This list is compressed and packed into 15 bits; it may be unpacked using the shift counts and level masks provided in the network routing table. The global address is 15 bits rather than 16 because 1 bit is required for internal use by the RTU routing routines.

During system generation, the user indicates the number of levels in the network and the maximum number of slaves connected to any node for each level in the network. The global address packing required may be calculated from this information; the shift count and level mask information is a direct consequence of this packing. The lowest level in the network is packed into the rightmost (least significant) bits, therefore the shift count for the lowest level in the network is zero.

Level Number

The level number indicates the network level on which this node is a slave. The network master is a slave on (hypothetical) level 0 by definition. The level number is used to select the appropriate shift count and mask from the routing table to accomplish routing.

Up/Down Mask

The up/down mask consists of all ones for the current level and all higher levels as packed in the global address. The up/down mask is the same for level 1 as for the level 0 (network master) node, by definition.

The up/down mask, when ANDed with the global address, determines how the message should be routed. A complete description of the routing process is provided below.

Routing Information Common To All Nodes

The routing information common to all nodes in a network describes the topology or structure of the network. It describes the structure as it would be if all possible branches existed; the maximum line addresses which follow in the routing table constrain the branching to maximums selected by the configuring engineer.

Shift Count

The shift count indicates the number of positions to shift the global address right in order to right justify the address field for the selected level of the network. The shift count for the lowest level of the network is zero by definition.

Level Mask

The level mask is used to extract the address field for the selected level of the network from the global address. The level mask is right justified so the global address must be shifted before applying the level mask.

Line Vs. Node Number Information

This section of the routing table is specific to each node and indicates the highest node number for each line; unused lines are indicated by a zero. Nodes are always assigned numbers so that the groups do not overlap. Thus, the line to be used to talk to a slave node is uniquely determined by searching this table from lowest line number to highest line number, transmitting on the first line whose highest node number equals or exceeds the target node number.

4.3.2. Use of the Routing Table

This section describes the manner in which it was anticipated that the routing table would be used. The basic steps in routing a global message are:

Use the up/down mask to decide whether the message is going up, down, or is staying in this node.

If the message is going up, send it to the master (local addr=0). If it's going down, determine the slave address to send it to by using the level mask and shift count for the next level down and determine which line to send it to using the line max address section of the routing table. If the message is for this node, hook it to the appropriate exchange.

The detailed steps required to use the routing table follow:

```
(First, check for local addressing)

IF local address mode and it matches local address in
routing table
    THEN accept message (it is for this node)

(Handle global mode messages)

ELSE IF global address AND up/down mask for this
       level = global address in routing table
    THEN (msg is directed to or through this node)
       IF global address AND NOT up/down mask = 0
       THEN accept message (it's for us)
       ELSE shift global address right per next level...
```

... down shift count,
AND with next level down mask, select line, ...
...send down
ELSE IF from a slave
THEN route msg up (use local addr = 0)
ELSE ignore message (implies routing table error
or routing table being replaced)

Appendix B
Remote Data Base (RDB) Access

1. Introduction

This document describes the design of the Remote Data Base (RDB) access functions used in the Bristol 33xx and Teleflow products. The basic functions of RDB are presented first, followed by a description of the design criteria used. Structures of the individual bytes which make up a message are presented next, followed by examples of message formats.

The message structures shown are in the transport level of the BSAP protocol; the link level and network level protocols are described in the Bristol Standard Asynchronous Protocol (BSAP) document, which may be found in Appendix A.

2. 33xx Remote Data Base Functions

The basic function of RDB is to provide a method to read/write data from/to the data base (memory) of a 33xx. For the purposes of this document the data base of the 33xx may be considered to be the entire memory of the 33xx. The intent is to provide a uniform method of reading or writing any data in the 33xx.

2.1. Data Read From 33xx

2.1.1. Signal Data

The data associated with signals created with the AIC or ABC resides in the Master Signal Directory (MSD) in a 33xx. Signals may be requested by signal name, MSD address or signal list.

Whenever signal data from the 33xx is requested via address or list, the MSD Version number of the 33xx load file must also be supplied. This ensures that the address or list request is consistent with the load executing in the 33xx device.

Any data existing within the MSD data base is available to be read. Since each signal contains a read and a write security level, each read or write request for signal data requires that a security byte be sent to the 33xx. A data packing option is available if reading ONLY LOGICAL SIGNAL VALUES where the signal values are bit encoded in the response.

2.1.2. Select Signal Data

Signals may be selected by matching against the following data fields:

Control inhibit/enable
Manual inhibit/enable
Alarm inhibit/enable
Base name/index
Extension name/index
Attribute name/index
Extreme High Alarm state
Extreme Low Alarm state
High Alarm state
Low Alarm state
Questionable data/O.K. data (analog signals only)

The user may select the data fields to be returned for each signal that meet the match criteria. For example the user may gather the specified data from only those signals that are in alarm inhibit state, only those signals with a base name of "XYZ.", or those signals that are in alarm inhibit state, are in control enable state, and are in an extreme alarm state.

2.1.3. Data Array Elements

Data arrays may contain either floating point or logical data, with each element of a floating point array being 4 bytes in length, while each element of a logical data array is one bit in length. Data arrays have no individual security checks associated with them.

Data arrays are addressed logically (i.e. array number, row, column).

There are two independent data array request formats available, the short form and the general form. The short form uses byte fields to specifiy the row and column numbers for the referenced data array. This request format is supported by all 33xx firmware versions, but is only suitable for arrays with row and column dimensions less than or equal to 255. The general form uses word fields to specify the row and column numbers and thus may be used to access any data array in a 33xx.

The general data array request format is only supported in 33xx firmware Versions AF.00 or higher. The associated message structures for both data array request formats are defined later in this document.

2.1.4. Memory Data

This function provides a way to read any memory location in the 33xx.

The user provides the full address, i.e. segment and offset, allowing any memory address to be accessed. Each memory data read request must also supply a Load Version number for the 33xx file being accessed.

There are two modes of access for memory data reads. Using the first mode, the requester can read data beginning at the specified memory address.

The second mode allows reads from multiple blocks of memory to be specified in a single request.

2.2. Data Writes to the 33xx

2.2.1. Signal Data

Signal data writes are restricted to alarm, control, and manual inhibit/enable bits, questionable data flag, read/write security,and signal value.

Signal(s) may be accessed for write via MSD address, signal name, or a signal list.

All signal data writes carry the user security code and will be disallowed if the user security is too low. Signal data writes by MSD address or list must also include the MSD version number. Write requests by MSD address or list will be disallowed if the MSD version does not match.

2.2.2. Data Array Elements

Elements can be written to a read/write analog or logical array. A data array is accessed logically via array number, row, and column. Analog and logical arrays are numbered independently. As described previously, both the short and general form for data array write requests are available.

However, the general form is only supported in 33xx firmware Version AF.00 or higher.

It must be remembered that the element length of an analog array is 4 bytes, while the element length of a logical array is one bit. Any attempt to write to a read only data array will result in an error.

2.2.3. Memory Data

Any dynamic memory in the 33xx, with the exception of the I/O memory, may be changed via this command. All Write Memory Data requests must include the 33xx Load version number. Write Memory Data requests will be rejected if the Load version does not match.

The user is responsible for performing any interlocks necessary to insure the integrity of a running system. Any attempt to write to PROM or I/O memory will be reported as an error.

2.3. RDB Extension Functions

The RDB extension mechanism allows an easy way to add new functions to the RDB access interface. Beginning in 1995 with the 386 Protected-mode family of products a number of new 'extension' messages were created to replace the 'memory read/write' previously used. These devices are called 'XRDB' devices. The following functions are available under RDB Extension:

2.3.1. Audit File Interface

This interface allows to accessed and manage the Audit File:

Read 'n' records starting from given index from the oldest record in the Audit File.
Delete oldest 'n' records (where 'n' is the number of records that must have been read with a previous read request.
Delete oldest 'n' records and Read oldest 'm' records.

2.3.2. Security Code Validation

This interface allows to verify whether a given security code is defined in the running ACCOL load or not and if defined then determine the assigned security level.

2.3.3. Manage Expanded BSAP Group Number

This allows easy management of the "soft" programmable Expanded BSAP Group Number as follows:

Define/Change the EBSAP Group Number.
Undefine the EBSAP Group Number.
Read the EBSAP Group Number.

2.3.4. Read/Reset Communication Statistics

This allows read and reset of the Communications statistics in a XRDB device.

2.3.5. Read Array Elements

This allows array access by column and by individual array elements in XRDB devices.

2.3.6. Read Special Information

This allows access to Version, Features, and PROM link date information in XRDB devices.

3. Design Philosophy

The following is a brief description of some of the design criteria used in this specification.

3.1. Asynchronous Communications Considerations

RDB must work on a relatively slow, hierarchical network as well as a highway; a major factor in its design is that its primary use will be on hierarchical networks. Therefore an attempt has been made to limit the number of bytes being transmitted for each request/response.

The second point to be considered in slower networks is resource allocation. Any node in the network should be able to perform critical control data base operations, such as set a signal to start a shutdown procedure, without having to wait for a data base search operation to complete. While several methods of implementing such a scheme have been discussed it has been decided that the simple approach of one request/one response is the best fit given the storage requirements and complexity of the 33xx.

3.2. Data Base

The complexity of the data base in the 33xx has been addressed by bit encoding many of the parameters and making the parameters themselves variable in length. The "field select selector" byte is used to shorten many requests.

The message structure that RDB works with is the basic building block. RDB itself has no knowledge of where the message came from; this is a function of the inter-task communication system. The inter-task communication system is responsible for sending the response message back to the requesting node over the same path in the reverse direction.

4. Remote Data Base Access Message Structures

NOTATION

M(d) Destination Message Exchange
Coded byte to direct request message to the Remote Data Base Access task.

M(s) Source Message Exchange
Coded byte to direct response message to a handler task.

SQ Sequence Number
16 bit number that may be used to match responses with requests. (See intertask communications document)

ST Current node status
8 bit data describing the overall system status
The state of unused bits are indeterminate.

bit 7 currently unused
bit 6 currently unused
bit 5 Message from a VSAT slave port
bit 4 Download reception in progress
bit 3 currently unused
bit 2 Slave has more data to send (Up transmits pending)
bit 1 currently unused
bit 0 Unreported alarm

IOA Two byte memory mapped I/O address.

MA 32 bit memory address. Formatted as SEGMENT followed by OFFSET. This allows any memory location to be specified.

SEC Current security level of requestor.
Legal values are: 0, 1, 3, 7, F. Larger number denotes higher security level.

VER MSD Version number.

16 bit number used to identify the data base version. This is used to verify all Master Signal Directory requests that are made via physical address or list. (This system field will be updated by the AIC whenever changes have been made to the system that could effect any direct addressing of signals. This field is updated in the 33xx directly if changes are made via the on-line AIC. Changes made via the off-line AIC or ABC update this field in the 33xx load file only.)

MSD 16 bit Master Signal Directory address

LST List indicator
8 bit number indicating list to use in processing request.

NME Number of message elements
8 bit number indicating how many elements are in the request or response.

RER Overall Request error/status code

80h bit 7 is set ON if the remote is reporting an error. If no other bits are ON then the message contains data elements and one of them is in error; each element will be preceeded by an EER byte for that element. These EER bytes must be examined to find the problem data element.

01h bit 0 is set ON if more data is available, 0 if all data fit in one message.

When BIT7 is ON other bits in the RER may also be ON when the RDB request Function Code is for Signal, Direct I/O, or Memory Data (see Function Code definitions below). If any of these bits are set ON in the RER, no message elements will be returned in the response:

40h bit 6 set if invalid data base function requested
20h bit 5 set if version number does not match
10h bit 4 set if no match found
08h bit 3 set if an analog value was found in packed logical processing
04h bit 2 set if no match on name found, or Memory Read request exceeds maximum allowed message buffer byte count

When BIT7 is ON other bits in the RER code may also be ON when the RDB request Function Code is for Data Array Data (see Function Code definitions below). If bit 6, 3, or 2 is set in the RER, no message elements will be returned in the response:

40h bit 6 set if invalid data base function requested
08h bit 3 set if attempt to write to a read-only array
04h bit 2 set if invalid array reference
02h bit 1 set if array bounds were exceeded

EER - Element error/status code

EER bytes are appended to the front of each returned data element if one of them is in error, and BIT7 of the RER will be set ON. Elements without error will have zero (0) in their EER bytes, the erroneous element could have one of the following codes.

02h Attempt to write to a read only area
04h Invalid identifier, for example, name, list, or invalid MSD address
05h Security violation

06h Attempt to write to manual inhibited signal
07h Invalid write attempt, for example set an analog signal ON.
08h Premature End_of_Data encountered
09h Incomplete Remote Data Base request
10h No match for indicated Signal Name

NOTE: THE ELEMENT ERROR CODE IS ONLY RETURNED FOR SIGNAL DATA REQUESTS (EXCLUDING PACKED LOGICAL DATA). ALSO, THE ELEMENT ERROR CODE WILL ONLY BE PRESENT IN THOSE RESPONSES THAT CONTAIN ERRORS (i.e. BIT 7 OF RER = 1). THIS ENABLES THE USER TO DETERMINE WHICH INDIVIDUAL ELEMENTS ARE IN ERROR. IF NO ERROR IS PRESENT IN THE RESPONSE THEN THE EER BYTE WILL NOT PRECEDE EACH ELEMENT.

FUN Function code
bit 7 function bit
0 read function
1 write function
bits 6-4 = data type code
000 signal data
001 data array data
010 direct I/O data
100 memory data
111 Extended RDB function
bits 3-2 = select code
00 select via address or data array number, control
01 select via name or data array number, short form
10 select via match or data array number, general form
11 select via list
bits 1-0 = signal READ return code, WRITE Memory code
For READ:
00 return first match
01 return next match/continue list
10 return packed logical data (logical value data only)
For WRITE Memory,
00 Write Data to memory
01 OR Data with memory
10 AND Data with memory

NOTE:
PACKED LOGICAL DATA MAY ONLY BE USED IN READING LOGICAL SIGNAL VALUES. THE SIGNALS MAY BE SELECTED VIA ADDRESS, NAME, OR LIST.

Another expression of the above FUN code bits.

READ

- 0000 Signal
  - 0000 First match
  - 0001 Next match
  - 0010 Packed logical
- 0001 Data array
  - 0000 Select control
  - 0100 Select short form
  - 1000 Select general form
  - 1100 by list
- 0010 Direct I/O
- 0100 Memory
  - 0000 Write
  - 0001 OR data
  - 0010 AND data
- 0111 Extended RDB function

WRITE

- 1000 Signal
  - 00XX By Name
  - 01XX By MSD address
- 1001 Data array
  - 00XX
  - 01XX Short form
  - 10XX General form
- 1010 Direct I/O
  - XXXX
- 1100 Memory
  - XX00 Write element
  - XX01 OR element
  - XX10 AND element
- 0111 Extended

WFD Write Field for signal data only

01h = set signal to alarm inhibited
02h = set signal to alarm enabled
03h = set signal to control inhibited
04h = set signal to control enabled
05h = set signal to manual inhibited
06h = set signal to manual enabled
07h = set questionable data bit
08h = clear questionable data bit

09h = set logical value "on"
0Ah = set logical value "off"
0Bh = set analog value
0Dh = set string value
0Eh = set read security
0Fh = set write security
10h = reserved
11h = reserved
12h = reserved
13h = reserved
14h = reserved
15h = reserved
16h = reserved

NOTE:FIELD SELECTOR BYTES ARE ONLY USED IN ACCESSING SIGNAL DATA WHEN PACKED LOGICAL DATA IS NOT USED.

FSS Field select selector defines which field select bytes are included in the request:
BITS 7-4 not used at this time (reserved for priority)
BIT 3 Field select byte 4 is used
BIT 2 Field select byte 3 is used
BIT 1 Field select byte 2 is used
BIT 0 Field select byte 1 is used

FS1 Field selector byte one (most common options):
BIT 7 Type and Inhibits byte
BIT 6 Value
BIT 5 Units/Logical text
BIT 4 Msd address
BIT 3 Name Text (includes Base.Extension.Attribute)
BIT 2 Alarm status
BIT 1 Descriptor text
BIT 0 Logical ON/OFF text

FS2 Field selector byte two:
BIT 7 Protection byte
BIT 6 Alarm priority
BIT 5 Base Name
BIT 4 Extension
BIT 3 Attribute
BIT 2 Low alarm deadband MSD address
BIT 1 High alarm deadband MSD address
BIT 0 MSD Version number

FS3 Field selector byte three:
BIT 7 Low alarm limit MSD address
BIT 6 High alarm limit MSD address

BIT 5 Extreme low alarm limit MSD address
BIT 4 Extreme high alarm limit MSD address
BIT 3 Reserved for Report by exception
BIT 2 Reserved (old Lock bit)
BIT 1 Signal value; Analogs return no Q-bit in FP mantissa.
BIT 0 When set, allows long signal names to be returned, instead of standard length names. (Requires ControlWave firmware version 2.2 or newer)

FS4 Field selector byte four:
This byte is reserved for future use.

XFC Extended Function Code
See Section 7 of this appendix for details.

XSFC Extended Sub-function
See Section 7 of this appendix for details.

PES Primary Error Status
See Section 7 of this appendix for details.

SES Secondary Error Status
See Section 7 of this appendix for details.

5. Read Message Formats

5.1. Signal Data

The signal data read request type and returned information is entirely user specified. In this manner the user controls how much data is transmitted over the network. Only the data that is actually required need be requested. For example, once a signal's MSD address is known, there is no need to ask for it again. A Read Signal by Name request may be used to return the MSD address and the MSD version number for a group of signals. Subsequent Read (or Write) requests for those signals may be performed using Read (or Write) via MSD address. This helps to minimize the number of bytes transmitted over the network since this message format is more compact than Read Signal by Name.

Each signal is considered as an individual element of the response. If an error has occurred in the processing of any element of the request, the RER byte will have bit 7 set and each element of the response will be preceded by its own error code. The error code for error free elements of that request will be equal to zero. If an error has occurred on any signal element, no data will be returned for that signal, only the error code.

If no errors have occurred in the processing of the response, bit 7 of the RER will be equal to 0 and NO ELEMENT ERROR CODES will precede each signal element.

The selected data fields are returned, assuming no other errors, in the order specified in the field select bytes, where the field select bytes are processed high order to low order. In other words, if the type byte is selected it will be the first field in any data element after the element error code. The length of, and sometimes the existence of, the returned data field for each signal is dependent on the data requested and the signal type.

The format of each field is as follows:

Type and Inhibits one byte bit encoded
- bit 0-1 00 = logical
  10 = analog
  11 = string
- bit 2 0 = not alarm signal
  1 = alarm signal
- bit 3 0 = dynamic signal
  1 = constant signal
- bit 4 0 = manual enabled
  1 = manual inhibited
- bit 5 0 = control enabled
  1 = control inhibited
- bit 6 0 = alarm enabled

1 = alarm inhibited

bit 7 0 = O.K. data

1 = questionable data (analog only)

Value (type dependent)

If Analog

floating point value in IEEE single precision
standard format - 4 bytes

If Logical

1 byte where OFF = 00h and ON = 01h

If String

Null terminated ASCII string maximum characters =65

If packed logical value

bit encoded logical values packed 8 signals per byte where the first signal selected is bit 7, the next bit 6 etc.

Units/Logical text

If string signal

no data is returned

If analog signal

6 ASCII characters of units text

If logical signal

6 ASCII characters current logical text

Msd address

16 bit MSD address

Name text

Variable length null terminated ASCII string in format of BASE.EXT.ATT Two periods are required, whether attribute is present or not. Maximum length = 21 bytes

Alarm status (type dependent)

if not an alarm signal

no data is returned

if logical alarm

1 byte

bits	description
0,1	Alarm Report State

00 = Nothing to Report

01 = Normal
10 = Momentary
11 = Multiple

2 Alarm Ack State (0 = ACK, 1 = Not ACK)
3 Alarm State (0 = Off, 1 = On)
4-6 Unused
7 Alarm Time Stamp Buffer bit

if analog alarm
2 bytes

byte 1

bits	description
0,1	Low Alarm Report State 00 = Nothing to Report 01 = Normal 10 = Momentary 11 = Multiple
2	Low Alarm Ack State (0 = ACK, 1 = Not ACK)
3	Low alarm state (0= OFF; 1 = ON)
4,5	High alarm state 00 = Nothing to Report 01 = Normal 10 = Momentary 11 = Multiple
6	High Alarm Ack State (0 = ACK, 1 = Not ACK)
7	High alarm state (0= OFF; 1 = ON)

byte 2

bits	description
0,1	Extreme Low Alarm Report State 00 = Nothing to Report 01 = Normal 10 = Momentary 11 = Multiple
2	Extreme Low Alarm Ack State (0 = ACK, 1 = Not ACK)
3	Extreme Low alarm state (0= OFF; 1 = ON)
4,5	Extreme High alarm state 00 = Nothing to Report 01 = Normal 10 = Momentary 11 = Multiple
6	Extreme High Alarm Ack State (0 = ACK, 1 = Not ACK)
7	Extreme High alarm state (0= OFF; 1 = ON)

Descriptor text

Variable length null terminated ASCII character string 65 bytes maximum

Logical ON/OFF text type dependent

if logical signal

12 bytes of ASCII character data formatted as
6 bytes of on text followed by 6 bytes of off text

if string or analog signal

nothing is returned

Protection byte (Each nibble has the same structure as SEC byte)

bits 0-3 read protection
bits 4-7 write protection

Alarm priority type dependent

if not an alarm type of signal

no data is returned

if logical alarm

one byte

bits 0-1 = alarm type

01 false alarm
10 true alarm
11 change of state alarm

bits 2-3 = alarm priority

00 event
01 operator guide
10 non critical
11 critical

if analog alarm

one byte

bits 0-1 = low alarm priority

00 event
01 operator guide
10 non critical
11 critical

bits 2-3 = high alarm priority

00 event
01 operator guide
10 non critical
11 critical

bits 4-5 = extreme low alarm priority

00 event
01 operator guide
10 non critical
11 critical

bits 6-7 = extreme high alarm priority

00 event
01 operator guide
10 non critical
11 critical

Base name
Variable length null terminated ASCII string maximum length of 9

Extension
Variable length null terminated ASCII string maximum length of 7

Attribute
Variable length null terminated ASCII string maximum length of 5

Version number
16 bit PEI/ACCOL Load Version number

Low alarm deadband MSD address
if analog alarm
16 bit MSD address
else no data

High alarm deadband MSD address
if analog alarm
16 bit MSD address
else no data

Low alarm limit MSD address
if analog alarm
16 bit MSD address
else no data

High alarm limit MSD address
if analog alarm
16 bit MSD address
else no data

Extreme low alarm limit MSD address
if analog alarm
16 bit MSD address
else no data

Extreme high alarm limit MSD address
if analog alarm
16 bit MSD address
else no data

5.1.1. Signal Read By Address

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 0H (read signal via MSD address)
6	FSS Field select selector
7	FS1 Field select byte 1
8	FS2 Field select byte 2 (if needed)
9	FS3 Field select byte 3 (if needed)
10	FS4 Field select byte 4 (if needed)
8,9...	VER Version number
10...	SEC Security level
11...	NME Number of Message Elements
12,13...	MSD Address 1
14,15...	MSD Address 2
.	
.	

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7..	Data Element 1
	EER error Code only if errors
	field 1
	field .
	field .
	field n
	Data Element 2
	EER error Code only if errors
	field 1
	field .
	field .
	field n

5.1.2. Read Packed Logical By Address

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 02H (read PACKED signal via MSD)
6,7	VER Version number
8	SEC Security level
9	NME Number of Message Elements
10,11	MSD Address 1
12,13	MSD Address 2
.	
.	
.	

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	Data Element 1
	Eight logical signal values
.	Data Element 2
	Eight logical signal values

.
.

5.1.3. Read Signal By Name

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 04H (read signal via name)
6	FSS Field select selector
7	FS1 Field select byte 1
8	FS2 Field select byte 2 (if needed)
9	FS3 Field select byte 3 (if needed)
10	FS4 Field select byte 4 (if needed)
11	SEC Security level
12..	NME Number of Message Elements
13..	Name 1*
14..	Name 2*
.	
.	

* Name is a variable length, null terminated ASCII string in the following format:

basename.extension.attribute

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7..	Data Element 1
	EER error Code only if errors
	field 1
	field .
	field .
	field n
	Data Element 2
	EER error Code only if errors
	field 1
	field .
	field .
	field n

5.1.4. Read Packed Logical By Name

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 06H (read packed logical via name)
6	SEC Security level
7	NME Number of Message Elements
8..	Name 1*
*..	Name 2*
.	
.	
.	

* Name is a variable length, null terminated ASCII string in the following format:

basename.extension.attribute

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	Data Element 1 - Eight logical signal values
.	Data Element 2 - Eight logical signal values
.	
.	

5.1.5. Read Signal By List

The read by list functions comes in two "flavors", one to start the read of a list, and another to continue the list. This is necessary because the length of the list and the data requested on a per signal basis may result in more data being returned than can fit into one buffer. The next list element number will be returned along with an indication that more data in the list remains. The user may then issue the continue list request with the next element number to continue processing the list. Note that you may calculate the elements that will be returned in one buffer and issue the next request before the first response is received.

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code, 0CH = (read signal via list start) -or- 0DH = (read signal via list continue)
6	FSS Field select selector
7	FS1 Field select byte 1
8	FS2 Field select byte 2 (if needed)
9	FS3 Field select byte 3 (if needed)
10	FS4 Field select byte 4 (if needed)
11,12	VER Version number
13	SEC Security level
14	LST List number
15,16	List entry number (if list continue function)

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	List number
8,9	Next list entry number (0 if end of list)
10..	Data Element EER error Code only if errors field 1 field 2

5.1.6. Read Packed Logical By List

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code, 0EH = (read logical via list start) *-or-* 0FH = (read logical via list continue)
6,7	VER Version number
8	SEC Security level
9	LST List number
10,11	List entry number (if list continue function)

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	List number
8,9	Next list entry number (0 if end of list)
10	Data Element 1 - Eight logical signal values
.	Data Element 2 - Eight logical signal values
.	
.	

5.1.7. Read of Selected Signal Data

Signals may be selected by comparing the current signal data with a set of user provided masks which select both the data fields to be compared and the required data for that field. The selection of signals may be based on the current signal data in the following fields:

control inhibit/enable
manual inhibit/enable
alarm inhibit/enable
high alarm state
low alarm state
extreme high alarm state
extreme low alarm state
base name
extension name
attribute name

The general format of the match parameters is as follows:

Byte	Content
1	States used for signal selection
2	Value required for selected states for signal selection
3	Ascii fields used signal selection
4...	null terminated strings for each Ascii field

The user selects the states to be used in the signal selection via the first match parameter. This parameter byte must be present in the request. The user sets the corresponding bit to indicate the states to be used in the selection.

Bit	Content
7	set if alarm inhibit/enable used
6	set if control inhibit/enable used
5	set if manual inhibit/enable used
4	set if high alarm state or logical alarm state used
3	set if low alarm state used
2	set if extreme high alarm state used
1	set if extreme low alarm state used
0	set if questionable data bit used (analog signals only)

If any bits are set on in the first match byte the next parameter will be the match conditions for the selected fields. The match conditions will take the same format as the first match parameter in that the match condition of the control inhibit/enable field will be specified as bit 7 of the second match byte and bit 3 would be used to match the low alarm state. The respective bit settings are as follows:

Bit	Content
7	set if alarm inhibited
6	set if control inhibited
5	set if manual inhibited
4	set if in high alarm state
3	set if in low alarm state
2	set if in extreme high alarm state
1	set if in extreme low alarm state
0	set if questionable data (analog signals only)

For example to select all alarm inhibited signal that are in only in extreme high alarm the two bytes would be:

Byte	Content	
0	84 hex	use data of alarm inhibit and alarm states
1	84 hex	value required for alarm inhibit and alarm states

The third byte of the selection determines the ASCII fields to be used in the signal selection. The user may search the MSD via the Base name, Extension name or Attribute name. The format of the third byte is as follows:

Bit	Content
7	set if Base name match
6	set if Extension name match
5	set if Attribute name match
4	reserved
3	reserved
2	reserved
1	reserved
0	reserved

The user may either supply the Base, extension, or attribute name if this is the first match , as a null terminated string, for each name to be matched. If continuing a search, the search will be performed using the base, extension, or attribute of the named signal.

The data returned for each signal selected is user selected by the use of the FSS and FS1 through FS4 bytes. As many signals as possible will be packed into each response buffer but partial signal data will not be packed into a buffer. The response will indicate through the RER byte whether or not there are more signals left to search. The user has the option of starting data gathering at the specified signal address, or of getting the next signal address. To do an initial search to find the first signal in the MSD the user would start at MSD address 0000.

String signals are not included in any searching.

REQUEST MESSAGE FORMAT

Byte	Content
0	M(s) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(d) Message Source Function Code
4	ST Node Status
5	FUN Function code = 08H (Signal match first)
6	FSS Field select selector
7	FS1 Field select byte 1
8..	FS2 Field select byte 2 (if needed)
9..	FS3 Field select byte 3 (if needed)
10..	FS4 Field select byte 4 (if needed)
11..	SEC Security level
12,13..	MSD Start MSD address (use 0000 to start)
14..	BIT FIELD SELECT BYTE
15..	BIT FIELD MATCH BYTE
16..	ASCII FIELD SELECT BYTE
17..	Null terminated string for Base if used
....	Null terminated string for Extension if used
....	Null terminated string for Attribute if used

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7,8	Last Msd address in response
9..	Data Element
	EER error Code only if errors
	field 0
	field 1
	.
	.
	.

5.2. Data Array Elements or Array Sizes

5.2.1. Read Array Size Information

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 010H (read array size info.)
6	Data array type 0 = analog 1 = logical
7	Data array number

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements (always 1)
7	Data array number
8,9	Rows in this array
10,11	Columns in this array
12	Array type: 0 = read-only, 1 = read/write

5.2.2. Read by Data Array Number, Short Form

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 014H (read data array: Short form)
6	Data array type 0 = analog 1 = logical
7	Data array number
8	Starting row
9	Starting column
10	Number of elements requested

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	Data array number
8	Next row not in this response
9	Next column not in this response
10	element 1
.	element 2
.	
.	

NOTE:

The size of each element depends on the type of array being read. Analog array elements are 4 byte single precision IEEE standard formatted floating point values. Logical array elements are one bit packed eight elements per byte, where each bit represents a logical signal value. The packing of logical array elements proceeds from bit 7 to bit 0. The first element is at bit 7, the second at bit 6 and so on.

5.2.3. Read By Data Array Number, General Form

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 018H (read data array: General form)
6	Data array type 0 = analog 1 = logical
7	Data array number
8,9	Starting row
10,11	Starting column
12	Number of elements requested

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	Data array number
8,9	Next row not in this response
10,11	Next column not in this response
12	element 1
.	element 2
.	
.	

NOTE:

The size of each element depends on the type of array being read. Analog array elements are 4 byte single precision IEEE standard formatted floating point values. Logical array elements are one bit packed eight elements per byte, where each bit represents a logical signal value. The packing of logical array elements proceeds from bit 7 to bit 0. The first element is at bit 7, the second at bit 6 and so on.

5.3. "Raw" I/O Data

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 020H (read direct I/O data)
6,7	VER version number
8	Number of Message Elements
9,10	16-bit I/O Address
11,12	16-bit I/O Address
.	
.	
.	

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
9	element 1
.	element 2
.	
.	

WARNING:

Not all I/O memory offsets are valid for reading.If the offset is associated with a BYTE of data, the byte will be repeated in both bytes of the returned word. If the offset is NOT associated with any valid hardware, unpredictable data is returned.

5.4. Read Memory Element (Single Memory Address)

REQUEST MESSAGE FORMAT

Byte	Content
0	M(s) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(d) Message Source Function Code
4	ST Node Status (return parameter)
5	FUN Function code = 40 hex (Read Memory)
6,7	VER Version number needed because of addressing
8	Number of bytes to read (max = 230; 0E6H)
9,10	Starting 16-bit Segment address
11,12	Starting 16-bit Memory address

NOTE: Each element is 32 bits.

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	Total number of data bytes in the response - n
7	8 bits memory data byte 1
8	8-bits memory data byte 2
9	8-bits memory data byte 3
.	
.	
.	

5.5. Read Memory Elements (Multiple Memory Addresses)

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status (return parameter)
5	FUN Function code = 40 hex (Read Memory)
6,7	VER Version number needed because of addressing
8	Number of bytes to read = 0FF (Designates this as a Multiple memory address read request)
9	Number of Groups: Number of memory address groups that follows:
xx,xx	Address 1 Specification:
	1 Number of bytes to read
	2,3 Starting 16-bit Segment address
	4,5 Starting 16-bit Memory address
xx,xx	Address 2 Specification
.	
.	
xx,xx	Address n Specification
.	
.	

NOTE: Total number of bytes read for ALL blocks cannot exceed 228 bytes (0E4H).

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(d) Message Source Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	Number of bytes = 0FF (Designates this as a multiple memory address read request)
7	Number of Groups: Number of memory address groups that were requested
8	Total number of data bytes in the response - n
9	8 bits memory data byte 1
10	8 bits memory data byte 2
11	8 bits memory data byte 3
:	
n+8	8 bits memory data byte n

6. Write Message Formats

6.1. Signal Data Formats

Individual signals may be specified by logical name or physical address within the 33xx. Alternatively a list of signals, the list existing within the 33xx, may be provided. Since each signal contains a read and write security level, each read or write request for signal data requires that a security byte be sent to the 33xx.

While the Master Signal Directory structure gives the user many fields to read, the structure is not conducive to writing in the same manner. For example, the base name of a signal is kept in the MSD as an index into a base name list. Therefore, only a limited set of data fields are available to be dynamically changed via the signal data write function. If the user requires dynamic changes to fields that are not available in the signal data write function, they may be accomplished by use of the memory write function.

The following fields may be dynamically changed with the signal data write function:

Alarm Inhibit/Enable
Control Inhibit/Enable
Manual Inhibit/Enable
"questionable data" data bit
Signal value (logical, analog, or string)
Write security
Read security

One field may be changed for each signal element in the message, with that field being defined by the Write Field Descriptor.

The format of the Write Field Descriptor is as follows:

WFD Write Field Descriptor for signal data only

1 = set signal to alarm inhibited
2 = set signal to alarm enabled
3 = set signal to control inhibited
4 = set signal to control enabled
5 = set signal to manual inhibited
6 = set signal to manual enabled
7 = set questionable data bit
8 = clear questionable data bit
9 = set logical value "on"
10 = set logical value "off"
11 = set analog value
13 = set string value
14 = set read security

15 = set write security
16 = reserved
17 = reserved
18 = reserved
19 = reserved
20 = reserved
21 = reserved
22 = reserved

Please note that with functions codes 1 through 10 there is no other additional data required.

The data required for the remaining functions is:

Analog value	4 bytes single precision IEEE floating point
String value	Null terminated ASCII string 65 characters maximum
Read security	1 byte low order 4 bits = read security
Write security	1 byte low order 4 bits = write security

6.1.1. Write Signal By Address

Requests to write by address will be rejected if the address supplied is not a valid signal address in the current load.

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 080H (write signal via MSD address)
6,7	VER Version number.
8	SEC Security level
9	NME Number of Message Elements
10,11	MSD Address 1
12	WFD Write Field descriptor 1
13...	Write data 1 (if needed, see description above for length)
...	MSD Address 2
...	WFD Write Field descriptor 2
...	Write data 2 (if needed, see description above for length)
.	
.	
.	

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	EER Element error code 1
8	EER Element error code 2
.	
.	
.	
...	EER Element error code n

6.1.2. Write Signal By Name

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 084H (write signal via name)
6	SEC Security level
7	NME Number of Message Elements
8..	Name 1*
...	WFD Write Field descriptor 1
...	Write data 1 (if needed)
...	Name 2*
...	WFD Write Field descriptor 2
...	Write data 2 (if needed)
.	
.	
.	

* Name is a variable length, null terminated ASCII string in the following format:

basename.extension.attribute

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	EER Element error code 1
8	EER Element error code 2
.	
.	
...	EER Element error code n

6.1.3. Write Signal By List

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 08CH (write signal via list) *-or-* = 08DH (write signal via list continue)
6,7	VER Version number
8	SEC Security level
9	LST List number
10,11	List entry number (if list continue function)
12	WFD Write Field descriptor 1
...	Write data 1 (if needed, see description above for length)
...	WFD Write Field descriptor 2
...	Write data 2 (if needed, see description above for length)
...	WFD Write Field descriptor 3
...	Write data 3 (if needed, see description above for length)

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code
6	NME Number of Message Elements
7	EER Element error code 1
8	EER Element error code 2
.	
.	
.	
...	EER Element error code n

6.2. Data Array Elements

6.2.1. Write By Data Array Number, Short Form

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 094H (write data array: Short form)
6	Data array type 0 = analog 1 = logical
7	Data array number
8	Starting row
9	Starting column
10	NME number of message elements
11..	Data for element number 1

NOTE:

The size of each element depends on the type of array being read. Analog array elements are 4 byte single precision IEEE standard formatted floating point values.

A logical array element will be set 'OFF' if the data element is 0,
otherwise it will be set 'ON'.

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code

6.2.2. Write By Data Array Number, General Form

REQUEST MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status
5	FUN function code = 098H (write data array: General form)
6	Data array type 0 = analog 1 = logical
7	Data array number
8,9	Starting row
10,11	Starting column
12	NME number of message elements
13..	Data for element number 1

NOTE:

The size of each element depends on the type of array being read. Analog array elements are 4 byte single precision IEEE standard formatted floating point values.

A logical array element will be set 'OFF' if the data element is 0, otherwise it will be set 'ON'.

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(s) Message Source Function Code
1,2	SEQ Sequence Number
3	M(d) Message Destination Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code

6.3. Write of Memory Data

REQUEST MESSAGE FORMAT

Byte	Content
0	M(s) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(d) Message Source Function Code
4	ST Node Status (return parameter)
5	FUN Function code: C0 hex (Write Memory) C1 hex (OR Memory) C2 hex (AND Memory)
6,7	VER Version number needed because of addressing
8	Number of bytes to write (max = 238 for local request, max = 233 for Global request)
9,10	Starting 16-bit Segment address
11,12	Starting 16-bit Memory address
13	8-bits memory Data
14	8-bits memory Data

RESPONSE MESSAGE FORMAT

Byte	Content
0	M(d) Message Destination Function Code
1,2	SEQ Sequence Number
3	M(s) Message Source Function Code
4	ST Node Status (return parameter)
5	RER Request Error Code

7. RDB Extension Function Request/Response Message Structures:

These functions were created to provide a means of creating additional RDB messages to meet the needs of future products. An RDB message with a Function code (FUN) of 70 hex indicates to the receiver that the fields following the FUN will contain Extended Function and Sub-function codes. The interpretation of the response message from the RTU is based on the received PES and SES values; content and size of a response message depends upon success or error status reported via the PES and SES. Some codes have been assigned, new ones will be added as needs arise. The active Extended Function Codes (XFC) are:

03	Audit Trail Functions
04	Unused
05	Security Code Functions
06	Expanded BSAP Functions
07	Archive File Functions
08	Read Communication Statistics
09	Read array elements
0A	Read Special Information
0B	Read and write memory

Within each XFC there are assigned Sub-function Codes (XSFC) that dictate the action to be done within the category; these are explained in the detailed format descriptions in section 8.0, but are listed here for reference.

XFC	XSFC	ACTION
03	01	Read Audit file - MSD address, time, value
	02	Read Audit file - Name and Units text
	03	Read Audit file - ASCII report
	04	same as 01 but 6 bytes longer
	05	same as 02 but 6 bytes longer
	06	same as 03 but 6 bytes longer
	07	Write ASCII text to Audit file
	08	Set Audit pointer
05	01	Validate security code (Simple Mode)
	02	Validate security code (Secure Mode - GET KEY)
	03	Validate security code (Secure Mode - LOGIN REQUEST)
06	01	Change Expanded BSAP Group number
	02	Clear Expanded BSAP Group number to zero
	03	Read the Expanded BSAP Group number
08	00	Read port communication statistics
	01	Reset port communication statistics

	02	Read buffer usage
	03	Reset buffer usage
	04	Read crash blocks
	05	Reset (clear) crash blocks
	06	Read port summary
	10	Read Global IP Statistics
	11	Read Global ICMP Statistics
	12	Read Global UDP Statistics
	13	Read Global IBP Statistics
	1F	Reset all IP Global Statistics.
09	00	Read array elements by column
	01	Read individual elements
0A	00	Read Version, Features, Prom link date
	01	Read Custom Prom information
	02	Read NRT information
	03	Read on-board serial EEPROM information (ControlWave/ControlWaveLP ONLY).
0B	00	Read from a memory address
	01	Write to memory
	02	OR to memory
	03	AND to memory

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code xx
7	XSFC- Extended Request Sub-function Code xx
8	DATA- Request specific data values xx,xx,...,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx

6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code xx
11	XSFC- Extended Request Sub-function Code xx
12...	DATA- response specific data values xx,xx,...,xx

7.1. Extended RDB Error Codes:

7.1.1. Primary Error Status (PES)

The PES (this byte is same as RER byte in standard RDB response messages) has following values (values are in decimal which are represented as hex values in the response messages):

-127	Unsupported RDB Extension Function Code
-126	Unsupported RDB Extension Sub Function Code
-001	Reserved for GFC 3308
-002	Reserved for GFC 3308
-003	Error with Audit File Interface request
-004	Reserved for GFC 3308
-005	Error with Security Code Validation request
-006	Error with Expanded BSAP Group Number request
-007	Error in Communication statistics request
-008	Error in Extended array request
-009	Error in Special request
-010	?? Error in Audit request ??

7.1.2. Secondary Error Status (SES)

The SES must be interpreted differently for each PES. The following SES error codes are defined:

PES	SES	
-127	XXX	Unsupported function code
-126	XXX	Unsupported sub-function code
-001	XXX	Error in system reset request
-002	XXX	Error in date/time request
-XXX	-101	Request format is incorrect.
-003	-001	EAUDIT Module is not defined.
-003	-002	Audit File is not defined.
-003	-003	Warning: Number of records in response is less than requested.
-003	-004	Security Code did not match. Delete operation not performed.
-003	-005	Delete record count is greater then the previous read count or Audit file has wrapped around. Delete operation is not performed.
-003	-006	Combination of SESs -003 and -004.
-003	-007	Combination of SESs -003 and -005.
-003	-008	Start Index is specified when Delete count is non zero.
-005	-001	Security Code is not defined in the running ACCOL load.
-006	-001	Expanded BSAP Group Number is not defined.
-007	-001	Port number invalid
	-002	Port is unused
	-003	Port error
-008	-001	Not all elements fit in response buffer (more to go)
	-002	Invalid array type (not 0 or 1)
	-003	Error in a request group (test the EER byte in the response for more error info)
	-004	Not all elements fit in the response AND there is an error in a group (see EER byte)
	-005 *	Invalid array number (not in load)
	-006 *	Invalid row or column
-009	-001	Invalid Custom Interface Table entry number specified in request

***** *These are also in the EER bye of the bad element.*

8. RDB Extension Request / Response Definitions:

When examples are presented the bytes 0-4 are not shown as they are common to all requests. Always use the response message length (number of response bytes received by your communication system) as the final authority to determine the actual number of data bytes/values that are present in a response message.

8.1. Audit File Interface

8.1.1. Audit Trail File - Read Request (MSD address, time):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 01
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 01
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message xx
17…	DATA - NUM_REC Audit File Records xx,xx,...,xx

REQUEST DESCRIPTION:

Delete previously read, NUM_REC, records from the Audit Trail File and then send oldest READ_COUNT number of records or nothing is to be deleted but send READ_COUNT number of records starting with the record number specified in the START_INDEX. Send response record in row data form and include the MSD address of the corresponding signal.

Where:

SEC_CODE	Two byte Unsigned Integer value. Security code for the delete operation. Same as the SEC_CODE value from the last response message.
DEL_COUNT	One byte unsigned integer value. Number of records to be deleted. Less then or equal to the NUM_REC value from the last response message. This value must be zero when the START_INDEX is non-zero. Otherwise an error will be returned.
READ_COUNT	One byte unsigned integer value. Number of records to read. Value range: 1-14.
START_INDEX	Two byte unsigned integer value. It specifies the first record to read from the Audit File. This number must be 0 when the DEL_COUNT is specified. Otherwise an error will be returned. For example, when this value is 0 or 1 the oldest record is read first. When this value is 5 the fifth record (oldest record is the first record) is read first.
REC_COUNT	Two byte unsigned integer value. In all 33XX platform RTUs this count is the number of records remaining in the Audit Trail File including the records present in the current response message. In all CW platform RTUs this count is the number of records remaining in the Audit Trail File after the last record reported in the current response message.
NUM_REC	One byte unsigned integer value. Number of records included in this response message. Value will range from 1-14.

PES	SES	
-003	-001	EAudit module is not defined.
-003	-002	Audit File is not defined.
-003	-003	Warning: Number of records in response is less than requested.
-003	-004	Security Code did not match. Delete operation not performed.

-003	-005	Delete record count is greater than the previous read count or Audit file has wrapped around. Delete operation is not performed.
-003	-006	Combination of SESs -003 and -004.
-003	-007	Combination of SESs -003 and -005.
-003	-008	Start Index is specified when Delete count is non zero.

REQUEST PROCESSING:

First the request format is validated. After that it is verified that the DEL_COUNT and START_INDEX both are not greater than 0. Any error will result in request being rejected with appropriate error status. Now the security code is matched against the one that was sent in the last response message. If it matches then the delete count is compared against the number of records of the last response message. If the delete count is less than or equal to NUM_REC count from the last read or NUM_REC count from the first read of the multiple reads with START_INDEX then the specified number of records are deleted from the Audit Trail File. The security code will match only it the Audit Trail File has not wrapped around since the last read request (see the ACCOL II Reference Manual for the recording mode options). If the delete operation fails then the response message that follows reestablishes the security code. The delete count can be 0. In this case the validity check is not performed. Once the delete operation is performed the read operation is started.

If the START_INDEX is not zero then it is used as the index for the first record to read on the way to reporting READ_COUNT number of records. A value of 0 or 1 means the oldest record in the file is read first and a value of 'n' means the 'nth' record (relative to the oldest record) is read first. Specified number of records, READ_COUNT, are read from the Audit Trail File and included in the response message. If the read count is invalid (> 14) then the response message includes maximum number of records that will fit in the response buffer. In the event that the Audit File has less then specified number of records all records are reported. If the read count is set to 0 then response does not include any record data and short response is generated.

For example, the following request simply returns the number of records in the Audit Trail File.

Request: 70 03 01 0000 00 00 00 00

Response: 00 00 VER 70 03 01 REC_COUNT

FORMAT OF EACH RESPONSE RECORD:

ADDRESS TYPE TIME NEW_VALUE OLD_VALUE

Where:

ADDRESS - Is a two byte MSD address of the signal, or if TYPE is 80, a system event:

0 SYSTEM TIME (System Time has been changed)
1 COLD START (System Cold Start has occurred)
2 WARM START (System Warm Start has occurred)
3 EPROM CHECKSUM (New system firmware downloaded)

TYPE - Is a one byte Reason For Recording code that can take one of two formats. The choice of format is determined based on the values of bits 3, 4, and 5. (NOTE: Values are shown in decimal but are represented as hex values in the actual response messages)

Format1: (bits 3, 4, and 5 are all 0)

*pp*000*eee*:

where *pp* = Alarm Priority

0 = Event
1 = Operator Guide
2 = Non Critical
3 = Critical

eee = Alarm Change Events

0 = Signal Change
1 = Logical Alarm
2 = Logical Return to Normal
3 = Analog Low Alarm
4 = Analog High Alarm
5 = Analog LowLow Alarm
6 = Analog HighHigh Alarm
7 = Analog Return to Normal

Format2: (At least one of bits 3, 4, and 5 is non-zero)

eeeeeeee

16 = Override Enable (signal has been control inhibited)
17 = Override Disable (signal has been control enabled)
18 = Operator Lockout (signal has been manually inhibited)
19 = Operator Enable (signal has been alarm enabled)
20 = Alarms Disabled (signal has been alarm inhibited)
21 = Alarms Enabled (signal has been alarm enabled)
32 = Frozen (input has been frozen for calibration)
33 = Unfrozen (input has been unfrozen)
34 = DP Verification

35 = SP Verification
36 = T Verification
37 = Unused
38 = Unused
39 = T Zero Adjustment
48 = DP Zero Calibration
49 = DP Span Calibration
50 = SP Zero Calibration
51 = SP Span Calibration
52 = T Zero Calibration
53 = T Span Calibration
80 = System Event has occurred (*When set, ADDRESS field indicates type of system event.)*
81 = User Audit Note created
82 = User Audit Node start
83 = User Audit Note continuation
84 = User Audit Note end
112 = Archive Collection occurring
113 = Archive Definition created/changed

TIME — Is a 5 byte field as follows:

0-1 = Number of days since 12/31/1976
2-3 = Number of 4 second intervals
4 = Number of 20 millisecond intervals

NEW_VALUE - Is a 4 byte floating point number:
For analog signal: signal value at the time of recording.
For logical signal:1.0/0.0 for ON/OFF state at the time of recording.
For system event (eee=80 in TYPE when ADDRESS=3, new firmware checksum)

OLD_VALUE - Is a 4 byte floating point number:

<u>For analog Alarm:</u>

Event Type	Value
Low Alarm	Low Alarm Limit
High Alarm	High Alarm Limit
LowLow Alarm	LowLow Alarm Limit
HighHigh Alarm	HighHigh Alarm Limit

For Logical Alarm: 0.0

For analog signal: signal value before the recording.

For logical signal: 1.0/0.0 for ON/OFF state before the recording.

For system event (eee=80 in TYPE when ADDRESS=3, old

firmware checksum)

8.1.2. Audit Trail File - Read Request (report Signal Name/Unit texts):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 02
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx,xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 02
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message
17...	DATA - NUM_REC Audit File Records xx,xx,...,xx

REQUEST DESCRIPTION:

Delete previously read, NUM_REC, records from the Audit Trail File and then sendnext READ_COUNT number of records or nothing is to be deleted but send READ_COUNT number of records starting with the record number specified in the START_INDEX. Send response record in row data form and include the name text of the corresponding signal - do NOT include the MSD address.

See the READ REQUEST (report MSD address) above for the individual request / response field description.

REQUEST PROCESSING:

This request is processed similar to the previous request (70 03 01 ....) except that the signal's name text is returned instead of it's MSD address. If the units text of the signal is required, then use the request with SFC=05.
Maximum number of records returned is strictly dependent on the length of the involved signal name text strings. Total number of data bytes available is 224.

FORMAT OF EACH RESPONSE RECORD:

TYPE Name_Text TIME NEW_VALUE OLD_VALUE oooooo nnnnnn

Where:

TIME, NEW_VALUE, OLD_VALUE are described under READ REQUEST (report MSD address).

TYPE: Is a one byte Reason For Recording code that can take one of two formats. The choice of format is determined based on the values of bits 4 and 5. (NOTE: Values are shown in decimal but are represented as hex values in the actual response messages.)

<u>Format 1: (bits 4,5 are both 0)</u>

ppttteee:

eee = Same as described earlier.
ttt = 000 for Logical Signal.
001 for Analog Signal.
pp = Same as described earlier.

<u>Format 2: (bits 4 or 5 are non-zero)</u>

eeeeteee

t = 0 for Logical Signal
1 for Analog Signal

eeee …eee = various events. Mask off bit 3 (value=8), and choose meaning from the following list:

16 = Override Enable (signal has been control inhibited)
17 = Override Disable (signal has been control enabled)
18 = Operator Lockout (signal has been manually inhibited)
19 = Operator Enable (signal has been alarm enabled)
20 = Alarms Disabled (signal has been alarm inhibited)
21 = Alarms Enabled (signal has been alarm enabled)
32 = Frozen (input has been frozen for calibration)
33 = Unfrozen (input has been unfrozen)
34 = DP Verification
35 = SP Verification
36 = T Verification
37 = Unused
38 = Unused
39 = T Zero Adjustment
48 = DP Zero Calibration
49 = DP Span Calibration
50 = SP Zero Calibration
51 = SP Span Calibration
52 = T Zero Calibration
53 = T Span Calibration
80 = System Event has occurred (*When set, ADDRESS field indicates type of system event.)*
81 = User Audit Note created
82 = User Audit Node start
83 = User Audit Note continuation
84 = User Audit Note end
112 = Archive Collection occurring
113 = Archive Definition created/changed

For example, if TYPE = 19_{hex} you get a decimal value of 25, but you must ignore or mask out bit 3 (value of 8) so you get a decimal value of 17 (which corresponds to Override Disable, i.e. signal has been control inhibited).

bit 7 e	bit 6 e	bit 5 e	bit 4 e	bit 3 t *Don't include this bit in your decimal value* *0=logical; 1= analog*	bit 2 e	bit 1 e	bit 0 e
0	0	0	1	1	0	0	1

Name_Text: Base_name.Extension.AttributeNULL Terminator

Where each of the name is a variable length ASCII string with NULL termination:

Base_Name = Maximum 8 Bytes + '.'
Extension = Maximum 6 bytes + '.'
Attribute = Maximum 4 bytes + Null terminator

example: #NDARRAY..\0
MAINT.MODE.\0
ENABLE.FIRST.RUN\0

Note: There is no space between the fields of a data record.

For Logical signals only:

In the EGM 3530 TeleFlow, RTU 3530 TeleRTU:
oooooo = Signal's old state before change as ON/OFF text.
nnnnnn = Signal's new state after change as ON/OFF text.

In the RTU 3305, GFC 3308, and all versions of the DPC 3330, DPC 3335 and RTU 3310:
oooooo = Always the signal's ON text.
nnnnnn = Always the signal's OFF text.

If it is desired to represent the ON/OFF text as "old state - new state" then the signal' new value or old values can be used to rearrange the text, e.g. if the new value is ON (1.0) then swap the text "nnnnnn (OFF Text) - oooooo (ON Text)" otherwise leave 'as is'.

For analog signals with SFC=05 only:
oooooo nnnnnn represent the units text.

8.1.3. Audit Trail File - Read Request (report in ASCII format):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 03
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx,xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 02
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message xx
17…	DATA - NUM_REC Audit File Records xx,xx,...,xx

REQUEST DESCRIPTION:

Delete previously read, NUM_REC, records from the Audit Trail File and then send next READ_COUNT number of records or nothing is to be deleted but send READ_COUNT number of records starting with the record number specified in the START_INDEX. Send response record in formatted ASCII form.

See the READ REQUEST (report MSD address) above for the individual request / response field description.

REQUEST PROCESSING:

This request is processed similar to the previous request (70 03 01 ....) except that the response records are fully formatted. Only 2 records can be reported in a single response.

FORMAT OF EACH RESPONSE RECORD:

When the response record type is Analog Value Change:

hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa sooooo.oo TO snnnnn.nn uuuuuu VALUE CHNG

hh:mm:ss	= time: hours:minutes:seconds
mm/dd/yy	= date: month:day:year
bbbbbbbb	= signal's base name
eeeeee.	= signal's extension name
aaaa.	= signal's attribute name
sooooo.oo	= Value before change
snnnnn.nn	= value after change
uuuuuu	= signal's Units Text

When the response record type is a Logical Value Change:

hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa oooooo TO nnnnnn STATUS CHANGE

oooooo = State before change as signal's ON/OFF text.
nnnnnn = State after change as signal's ON/OFF text.

When the response record type is an Analog Alarm:

hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa snnnnn.nn uuuuuu RETURN TO NORMAL

-or-

hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa snnnnn.nn uuuuuu xxxx
y-ALARM (sllllll.ll)

xxxx = Alarm Type: HIGH, HIHI, LOW, or LOLO
y = Alarm priority:
C = Critical
N = N on-Critical
O = Operator Guide
E = Event

sllllll.ll = Corresponding Alarm Limit

When the response record type is a Logical Alarm:

hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa nnnnnn y-ALARM
-or-
hh:mm:ss mm/dd/yy bbbbbbbb.eeeeee.aaaa nnnnnn
y-RETURN TO NORMAL

Note:
1. There is always a blank character present between successive data fields of a data records.
2. A data record is terminated with a NULL terminator.

8.1.4. Audit Trail File - Read Request (MSD, time, SEQ#):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 04
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 04
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message xx
17	DATA - NUM_REC Audit File Records xx,xx,...,xx

Note: All records will have a Local Seq, Global Seq, Operator# words (2 bytes each) at the end.

8.1.5. Audit Trail File - Read Request (Name,Units,Seq#):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 05 (Analog Signal's units text)
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 05
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message xx
17	DATA - NUM_REC Audit File Records xx,xx,...,xx

Note: All records wil have a Local Seq, Global Seq, Operator# words (2 bytes each) at the end.

8.1.6. Audit Trail File - Read Request (ASCII, SEQ#):

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 06
8,9	SEC_CODE - Delete Security Code xx,xx
10	DEL_COUNT - Number of Records to Delete xx
11	READ_COUNT - Number of Records to Read xx
12,13	START_INDEX - First (from oldest) record to read xx,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 06
12,13	REC_COUNT - Number of records in the Audit File xx,xx
14,15	SEC_CODE - Delete Security Code xx,xx
16	NUM_REC - Num of records in the Response Message xx
17	DATA - NUM_REC Audit File Records xx,xx,...,xx

Note: All records wiil have a Local Seq, Global Seq, Operator# words (2 bytes each) at the end.

8.1.7. Audit Trail File - Write Request (ASCII note)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 07
8	REC_COUNT – number of 16 byte records in the message
First record begins:	
9,10	MSD – record type
	00 = start note
	01 = continue note
	02 = end note
11	NOTE_TYPE - 51h
12-16	JTIME – Julian time (usually all zero)
17-24	DATA – 8 ASCII data bytes
End of first record	
25-…	*Additional records* (up to 6 total records)

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 07

8.1.8. Audit Trail File - Set Pointer

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M (s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 08
8,9	SEQ_NUM - Local sequence number

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 03
11	XSFC- Extended Request Sub-function Code 08

8.1.9. Read Audit Logs – Initial Selective Request

(ControlWave Firmware 04.62 and *newer*)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M (s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 09
8	Format Code 00 (include variable name, units text, and sequence numbers). All other possible code values reserved for future.
9	Type is one of the following: 00 Return both alarm and event logs 01 Return event logs *only* 02 Return alarm logs *only* 03 to 255: Assume default of 00; return both.
10	ORDER of records: 00 Return oldest records first 01 Return newest records first
11	Count - Number of records requested
12-21	ASCII Date (MM-DD-YYYY) or NULL String. If string not NULL, but conversion invalid, months and days are defaulted to 01; there is no default for the year. If date is NULL, records are returned based on ORDER (oldest or newest). If date string is valid but no record from that date exists, records returned are also based on ORDER: If ORDER is '00' (oldest records first), but no records from that date, the closest record with a *newer* date is reported first, then reporting continues from oldest to newest. If ORDER is '01' (newest records first), but no records from that date, the closest record with an *older* date is reported first, then reporting continues from newest to oldest.

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx

PES SES

-003 -002 Audit log (Alarm, Event, or both) not defined
-003 -003 Not Supported
-003 -009 Records for the specified date/indices not found

7,8 VER - ACCOL load version number xx,xx

9 FUN - Function Code: 70H

10 XFC - Extended Request Function Code 03

11 XSFC- Extended Request Sub-function Code 09

12 Format Code 00 (include variable name, units text, and sequence numbers). All other possible code values reserved for future.

13 Type is one of the following:
00 Return both alarm and event logs
01 Return event logs *only*
02 Return alarm logs *only*
03 to 255: Assume default of 00; return both.

14 ORDER of records:
00 Return oldest records first
01 Return newest records first

15 Total Number of audit events present

16 Total number of audit alarms present

17 Number of records remaining in the Audit Event log after the last event record in this response.

18 Number of records remaining in the Audit Alarm log after the last alarm record in this response.

19 Event Log Index – Returns the record position in the Event log past the last record in this response message. The requester should use this index when issuing a subsequent Continuation Selective Request. NOTE: The RTU does not guard against a wrap-around.

20 Alarm Log Index – Returns the record position in the Alarm log past the last record in this response message. The requester should use this index when issuing a subsequent Continuation Selective Request. NOTE: The RTU does not guard against a wrap-around.

21 Number of records (*n*) in this response. This number may be less than the requested record count. NOTE: Unlike the older interface, the SES error is not used to report a mismatch in number requested vs. number returned. Use this value to detect any mismatches.

22+ Records 1 to (*n*) from alarm / event logs are returned here, in sequential order. Alarm and event records may be mixed based on sequence number. The following information is returned (identical to XSFUN = 0x05): Record Type, Variable Name, Time Stamp, New Value, Old Value, ON/OFF text or Units Text, Audit Sequence Number, Global Sequence Number, and Operator Word (always 0).

8.1.10. Read Audit Logs – Continuation Selective Request

(ControlWave Firmware 04.62 and *newer*)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M (s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code 0A
8	Format Code 00 (include variable name, units text, and sequence numbers). All other possible code values reserved for future.
9	Type is one of the following: 00 Return both alarm and event logs 01 Return event logs *only* 02 Return alarm logs *only* 03 to 255: Assume default of 00; return both.
10	ORDER of records: 00 Return oldest records first 01 Return newest records first
11	Count - Number of records requested
12	Event Log Index (comes from Initial Selective Request Response XSFUN=0x09), or possibly through some other source via HMI. NOTE: The RTU does not guard against a wrap-around.
13	Alarm Log Index (comes from Initial Selective Request Response XSFUN=0x09) or possibly through some other source via HMI. NOTE: The RTU does not guard against a wrap-around.

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx PES SES -003 -002 Audit log (Alarm, Event, or both) not defined -003 -003 Not Supported -003 -009 Records for the specified date/indices not found
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H

10 XFC - Extended Request Function Code 03

11 XSFC- Extended Request Sub-function Code 09

12 Format Code 00 (include variable name, units text, and sequence numbers). All other possible code values reserved for future.

13 Type is one of the following:
- 00 Return both alarm and event logs
- 01 Return event logs *only*
- 02 Return alarm logs *only*

03 to 255: Assume default of 00; return both.

14 ORDER of records:
- 00 Return oldest records first
- 01 Return newest records first

15 Total Number of audit events present

16 Total number of audit alarms present

17 Number of records remaining in the Audit Event log after the last event record in this response.

18 Number of records remaining in the Audit Alarm log after the last alarm record in this response.

19 Event Log Index – Returns the record position in the Event log past the last record in this response message. The requester should use this index when issuing a subsequent Continuation Selective Request. NOTE: The RTU does not guard against a wrap-around.

20 Alarm Log Index – Returns the record position in the Alarm log past the last record in this response message. The requester should use this index when issuing a subsequent Continuation Selective Request. NOTE: The RTU does not guard against a wrap-around.

21 Number of records (*n*) in this response. This number may be less than the requested record count. NOTE: Unlike the older interface, the SES error is not used to report a mismatch in number requested vs. number returned. Use this value to detect any mismatches.

22+ Records 1 to (*n*) from alarm / event logs are returned here, in sequential order. Alarm and event records may be mixed based on sequence number. The following information is returned (identical to XSFUN = 0x05): Record Type, Variable Name, Time Stamp, New Value, Old Value, ON/OFF text or Units Text, Audit Sequence Number, Global Sequence Number, and Operator Word (always 0).

8.1.11. Collect Records from Audit Logs Starting From a Specified Point (ControlWave Firmware 05.20 and newer)

This request has following multiple uses:
1. Start collecting records from the oldest record from the events log only, the alarms log only, or both logs.
2. Continue collection. Use the specified Index / Sequence number and search the log. Start reporting from the next (older/newer) record in the log.

Users must process the Alarm/Event Log Status flags as well as Primary and Secondary Error Status' (PES/SES) to detect wraparounds and other possible errors.

The response includes the index in the log and sequence number of the last record in the response. If the response does not have a record from a given log then the index/sequence numbers are copied from the request. If Index / Sequence number were received but the specified record was not found then the response will use the Index / Sequence Numbers to 65535.

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M (s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 03
7	XSFC- Extended Request Sub-function Code: 0x0B = Initial Request 0x0C = Continuation Request
8	Format Code 00 Include variable name, units text, and sequence numbers. 01 Row record data (PDD index, Type, Time, New Value, and Old Value, Local Sequence, Global Sequence, Operator# [unused]; See Read Request with Sub-Function code, XSFC, = 04)
9	Request Type is one of the following: 00 Return both alarm and event logs 01 Return event logs *only* 02 Return alarm logs *only* 03 to 255: Assume default of 00; return both.
10	Collection order of records: 00 Return oldest records first 01 Return newest records first 02 Reserved for future; defaults to 00
11	Count - Number of records requested
12-13	Relative Index to the oldest/newest Audit Events Log record. Not used

when only alarms log is processed. Set this to 0 for the 0x0B request otherwise set it to the index received in the last response.

14-15 Sequence Number of the oldest/newest Audit Events Log record. Not used when only alarms log is processed. Set this to 0 for the 0x0B request otherwise set it to the sequence number received in the last response.

16-17 Relative Index to the oldest/newest Audit Alarms Log record. Not used when only alarms log is processed. Not used when only events log is processed. Set this to 0 for the 0x0B request otherwise set it to the index received in the last response.

18-19 Relative Sequence Number of the oldest/newest Audit Alarms Log record. Not used when only events log is processed. Set this to 0 for the 0x0B request otherwise set it to the index received in the last response.

Processing of Index / Sequence Number:

1. When this is an Initial Request (XSFC = 0x0B) the reporting begins with the selected log(s) and from the oldest / newest record depending on the order.
2. When this is a Continuous Request (XSFC =0x0C) the reporting is as follows:
   a. If the Event record at the specified index has the same sequence number as specified in the request then processing begins with the next event record. Otherwise, the event log is reset to the oldest event log record and an error is reported (SES = -16).
   b. If the Alarm record at the specified index has the same sequence number as specified in the request then processing begins with the next alarm record. Otherwise, the alarm log is reset to the oldest alarm log record and an error is reported (SES = -17).
   c. If both logs were reset then the SES error is set to = -14.
3. Use the count Records in Response to determine how many records are present in the response. The SES field is not set to -3 when the response does not include the requested number of records.

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx

PES	SES	
-003	-002	Audit log (Alarm, Event, or both) not defined
-003	-003	Not Supported

-003 -004 Security Mismatch
-003 -010 The response does not include any records as there were none found that match the requested criteria.
-003 -101 Request message size is invalid

7,8 VER - ACCOL load version number xx,xx

9 FUN - Function Code: 70H

10 XFC - Extended Request Function Code 03

11 XSFC- Extended Request Sub-function Code 0

12 Format Code (include variable name, units text, and sequence numbers). All other possible code values reserved for future.

13 Type is one of the following:
00 Return both alarm and event logs
01 Return event logs *only*
02 Return alarm logs *only*
03 to 255: Assume default of 00; return both.

14 ORDER of records:
00 Return oldest records first
01 Return newest records first

15-16 Total Number of audit events present

17-18 Total Number of audit alarms present

19-20 Remaining Events = Number of records remaining in the Audit Event Log after the last event record in this response.

21-22 Remaining Alarms = Number of records remaining in the Audit Alarm Log after the last alarm record in this response.

23-24 Event Log Index = Index in the log of the last Event Log record in the response or the index to the oldest record in the events log. This is the Index in the Event Log to the last event record in the response. If the response does not have any event log record then copy the index from the request.

25-26 Event Log Seq = Sequence Number of the last Event Log record in the response or the sequence number of the oldest record in the events log or zero if no records in the events log. This is the Sequence number of the last event record in the response. If the response does not have any event log record then copy the sequence number from the request.

27-28 Alarm Log Index = Index in the log of the last Alarm Log record in the response or the index to the oldest record in the Alarm log. This is the Index in the Alarm Log to the last alarm record in the response. If the response does not have any alarm log record then copy the index from the request.

29-30 Alarm Log Seq = Sequence Number of the last Alarm Log record in the response or the sequence number of the oldest record in the Alarm log or zero if no records in the Alarm log. This is the Sequence number of the last Alarm record in the response. If the response does not have any Alarm log record then copy the sequence number from the request.

31 Records in Resp = Number of records in this response. This number may be less than the requested record count. Unlike the older

interface the SES error is not used to report this mismatch. Instead, use the value here to detect any mismatches.

32-n Audit Log Records from the log(s) are in the sequential order. If both logs are selected then the records may be mixed from both logs. Information included depends on the selected format.

8.2. Security Code Validation:

8.2.1. Standard Security Code Validation (Simple Mode)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 05
7	XSFC- Extended Request Sub-function Code 01
8-14	SEC_CODE - Security Code to Validate xx,xx,xx,xx,xx,xx, 0 NOTE: This field is variable (from 1 to 6 characters for the password; terminated by a NULL string, e.g. a 1 character password will be two bytes - 1 for the password, and 1 for the NULL, a total of 2 bytes.)

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 05
11	XSFC- Extended Request Sub-function Code 01
12	SEC_LEVEL - Assigned Security Level xx

REQUEST DESCRIPTION:

Validate the specified security code against the security code defined in the running ACCOL load and if it is a valid code then report the assigned security level.

REQUEST PROCESSING:

Specified security code is matched against the six security code defined in the running ACCOL load. If the code matches with one of the security code then the assigned security level is returned. If there is no match then an error is reported and the security level is set to 0.

Assigned security level is the position number of the matching security code within the security code table in the ACCOL load. Thus, the first code has security level 1 and the last code in the table has security level 6.

Example (assumes the security code SUNRIS is 4th in the security code table):

Request: 70 05 01 53 55 4E 52 49 53 00

Response: 00 00 VER 70 05 01 04

PES	SES	
000	000	Request processed successfully.
-005	-001	Security Code is not defined in the running ACCOL load.

8.2.2. Enhanced Security Code Validation (Secure Mode - GET KEY)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 05
7	XSFC- Extended Request Sub-function Code 02
8-11	Lifetime - Wait up to this number of milliseconds for Login request

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx, xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 05
11	XSFC- Extended Request Sub-function Code 02
12	KEY - Encryption key

REQUEST DESCRIPTION:

In this request, the PEI is requesting access to a 33xx controller which has an ACCOL load using 'secured mode' passwords. The user enters a password at the PEI, and this message requests a 'key' from the controller which will be used to encrypt the user's password. This request must be followed up (before expiration of 'lifetime') by a LOGIN REQUEST message (see next sub-section).

NOTE: If 'Secure Mode' is not being used at the controller, i.e. password is stored as plain text, or 'Secure Mode' is not supported by the firmware, the response message will return an error code of '-5'.

8.2.3. Enhanced Security Code Validation (Secure Mode - LOGIN REQUEST)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 05
7	XSFC- Extended Request Sub-function Code 03
8-13	Encrypted password (generated from user's entry and KEY)

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx, xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 05
11	XSFC- Extended Request Sub-function Code 03
12	SEC - Authorized Security Access Level

REQUEST DESCRIPTION:

In this request, the PEI sends an encrypted password to the controller. The encrypted password was generated by applying the encryption key (obtained using the GET KEY message) to the password entered by the user at the password prompt.

The response from the controller is either an authorized security access level, or an error indicating that the password sent was invalid.

8.3. Manage Expanded BSAP Group Number

The Expanded BSAP Group Number (EBSAP GN) is "soft' programmable entity. It resides in the RTC memory where it is battery backed up and survives the power failure. It eliminates the need to have a specially programmed custom prom or in case of a GFC 3308 it eliminates the use of the CPU Configuration switch to designate the node as the Group 1 Expanded BSAP node. There are three requests available to manage the GN.

8.3.1. Define/Change the Expanded BSAP Group Number

This request assigns the EBSAP GN to any 33XX node or changes it. To understand how the EBSAP GN is interpreted see the ACCOL II Reference Manual.

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 06
7	XSFC- Extended Request Sub-function Code 01
8	EBSAP GN - New Group Number xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 06
11	XSFC- Extended Request Sub-function Code 01
12	EBSAP GN - Assigned Group Number xx

REQUEST DESCRIPTION:

Define the specified group number in the node or change the existing group number to the new one given in the request.

REQUEST PROCESSING:

The Group Number in the RTC memory is set to the New Group Number specified in the request. A check is not performed to see if a group number is already defined. The new group number is simply assigned as the EBSAP GN. Also internal security codes are set that designates to the internal firmware that the EBSAP GN is now defined.

WARNING: THE RTU MUST BE POWERED OFF AND THEN POWERED BACK ON OR RESET OR RELOADED IN ORDER FOR THE COMM SYSTEM TO RECOGNIZE THE NEW GROUP NUMBER.

Example:

Request: 70 06 01 02

Response: 0000 VER 70 06 01 02

8.3.2. Undefine (Clear) the Expanded BSAP Group Number

This request undefines the EBSAP GN, i.e., sets it to zero. The node now becomes a standard BSAP node. To understand how the EBSAP GN is interpreted see the ACCOL II Reference Manual.

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 06
7	XSFC- Extended Request Sub-function Code 02

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 06
11	XSFC- Extended Request Sub-function Code 02
12	EBSAP GN - Undefined Group Number xx

REQUEST DESCRIPTION:

Undefine the specified group number from the node.

REQUEST PROCESSING:

The Group Number in the RTC memory is set to 00. Also internal security codes are reset to designates to the internal firmware that the EBSAP GN is NOT defined.

!!! WARNING: IF THE NODE WAS A EBSAP NODE RUNNING WITH A KNOWN GROUP NUMBER THEN THE RTU MUST BE POWERED OFF AND THEN POWERED BACK ON OR RESET OR RELOADED IN ORDER FOR THE COMM SYSTEM TO RECOGNIZE THIS CHANGE (TO STANDARD BSAP NODE).

Example:

Request: 70 06 02

Response: 00 00 VER 70 06 02 02

8.3.3. Read the Expanded BSAP Group Number

This request reads the EBSAP GN. To understand how the EBSAP GN is interpreted see the ACCOL II Reference Manual.

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 06
7	XSFC- Extended Request Sub-function Code 03

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 06
11	XSFC- Extended Request Sub-function Code 03
12	EBSAP GN - Defined Group Number xx

REQUEST DESCRIPTION:

Read the specified group number from the node.

REQUEST PROCESSING:

The Group Number defined in the RTC memory is reported. If the EBSAP GN is not defined then an error is reported.

PES	SES	
000	000	Request processed successfully.
-006	-001	Expanded BSAP Group Number is not defined.

8.4. Manage Archive Files

8.4.1. Read File Attribute Table

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 01
8,9	FILE_NO xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 01
12,13	FILE_CNT - xx
14	NME - Definitions in the response
15	DATA

PES SES -007 -001 Invalid file number in request

Response DATA*: Attributes of up to 10 files starting with the given FILE_NO:*

9 Bytes:	File Name.
2 Bytes:	File Number
1 Byte:	File Type: 0x00 – Periodic. 0x20 = Other.
2 Bytes:	Number of records
2 Bytes:	Record Size: (this value is 1 less than the actual internal record size as the Julian time stamp is stored as 5 byte value and reported as 4 byte value.
1 Byte:	Interval 00 = None (write on demand). 01 = One minute interval. 02 = Five minute interval. 03 = Fifteen minute interval. 04 = Hourly (one Hour interval). 05 = Daily (24 hour interval).
1 Byte	Archive Time Stamp Mode (Firmware 04.61 and *newer*) 0 = Start of Period 1 = At the time of storage
1 Byte	Archive Location (Firmware 04.61 and *newer*) 0 = Archive is in FLASH 1 = Archive is in SRAM
2 Bytes	Number of columns in the archive file (Firmware 04.61 and *newer*)

8.4.2. Read Record Format Definition

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 02
8,9	FILE_NO- Archive file number
10,11	FLD_NO- starting field number
12	ADFSS- Archive definition field selector
	xxxx xxx1 include header information
	xxxx xx1x include file definition

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 02
12,13	FILE_NO- file number
14,15	FIELD_CNT- total fields in record
16	NME- number of field definitions in response
17	DATA- content depends on ADFSS above

PES	SES	
-007	-001	Invalid file number
-007	-002	Invalid field number

Response DATA*: Attributes of the specified file (FILE_NO):*

9 Bytes:	File Name.
2 Bytes:	File Number
1 Byte:	File Type: 0x00 – Periodic. 0x20 = Other.
2 Bytes:	Number of records
2 Bytes:	Record Size: (this value is 1 less than the actual internal record size as the Julian time stamp is stored as 5 byte value and reported as 4 byte value.
1 Byte:	Interval 00 = None (write on demand). 01 = One minute interval. 02 = Five minute interval. 03 = Fifteen minute interval. 04 = Hourly (one Hour interval). 05 = Daily (24 hour interval).
1 Byte	Archive Time Stamp Mode (CW Firmware 04.61 and *newer*) 0 = Start of Period 1 = At the time of storage
1 Byte	Archive Location (Firmware 04.61 and *newer*) 0 = Archive is in FLASH 1 = Archive is in SRAM
2 Bytes	Number of columns in the archive file (Firmware 04.61 and *newer*)
10 Bytes…	Each Field / Column definitions – Up to 22 entries:

Field /Column definitions - Starting at nth position:

17 Bytes:	Name/Title (Depending on the request. This field may or may not be present).
2 Bytes:	Signal MSD Address.
1 Byte:	Signal Type: 00 Analog 01 Logical
1 Byte:	Characteristics: 00 = Average for time when weight factor 2 != 0. 01 = Arithmetic mean over weight factor 1. 02 = Average of square-root (variable) for time when weight factor 2 != 0. 03 = Square of (average of square-root (variable)). 04 = Instantaneous - place value in log. 05 = Minimum observed value for period.

	06 = Maximum observed value for period.
	07 = Place value in log, and zero signal.
	08 = Integration over weight factor 2.
1 Byte:	Data Size
	01 Logical
	04 Analog
1 Byte:	Display Precision
4 Bytes	Reserved

8.4.3. Read Record Format Definition By Name

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 03
8-16	FILE_NAME- 8 character name, NUL terminated
17,18	FLD_NO- start Field number
19	ADFSS- archive definition field selector

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 03
12	See sub-function 2

PES SES

-007 -001	Invalid file name.
-007 -002	Invalid field number.

Response DATA: *Attributes of the specified file (FILE_NO):*

9 Bytes: File Name.
2 Bytes: File Number
1 Byte: File Type:
0x20 other (non periodic)

2 Bytes: Number of records
2 Bytes: Record Size: (this value is 1 less than the actual internal record size as the Julian time stamp is stored as 5 byte value and reported as 4 byte value.
1 Byte: Interval
00 = None (write on demand).
01 = One minute interval.
02 = Five minute interval.
03 = Fifteen minute interval.
04 = Hourly (one Hour interval).
05 = Daily (24 hour interval).
1 Byte Archive Time Stamp Mode (CW Firmware 04.61 and *newer*)
0 = Start of Period
1 = At the time of storage
1 Byte Archive Location (Firmware 04.61 and *newer*)
0 = Archive is in FLASH
1 = Archive is in SRAM
2 Bytes Number of columns in the archive file
(Firmware 04.61 and *newer*)

27 Bytes… Each Field / Column definitions – Up to 8 entries:

Field /Column definitions - Starting at nth position:

17 Bytes: Name/Title (Depending on the request. This field may or may not be present).
2 Bytes: Signal MSD Address.
1 Byte: Signal Type:
00 Analog
01 Logical
1 Byte: Characteristics:
00 = Average for time when weight factor 2 != 0.
01 = Arithmetic mean over weight factor 1.
02 = Average of square-root (variable) for time when weight factor 2 != 0.
03 = Square of (average of square-root (variable)).
04 = Instantaneous - place value in log.
05 = Minimum observed value for period.

06 = Maximum observed value for period.
07 = Place value in log, and zero signal.
08 = Integration over weight factor 2.

1 Byte: Data Size
01 Logical
04 Analog

1 Byte: Display Precision

4 Bytes: Reserved.

8.4.4. Read File Pointers

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 04
8,9	FILE_NO- file number

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 04
12,13	FILE_NO- file number
14,15	GSEQ- current Global sequence number
16,17	ASEQ- current Archive sequence number
18,19	OSEQ- oldest record's sequence number
20,21	RSEQ- archive sequence of last reported record

PES	SES	
-007	-001	Invalid file number.

8.4.5. Read File Pointers By Name

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 05
8-16	FILE_NAME- 8 character name, NUL terminated

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 05
12	Same as sub-function 4 above

PES	SES	
-007	-001	Invalid file name.

8.4.6. Read Records By Start Date

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 06
8,9	FILE_NO- Archive file number
10,13	JDATE- Julian date

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 06
12	MORE- More data indicator, 0=none, 1=more to go
13	NME- Number of records in response
14,15	NEXT- Next sequence number not in this response
16	DATA- Archive records

PES	SES	
-007	-001	Invalid File number requested.
-007	-003	Date request for non-historical archive.

Specified Julian time stamp is used to find the first record with that or newer time stamp. Selected and newer records are included in the response. Number of records included in the response depends on the size of the record. For each record the following information is included:

4 Bytes	Julian Time Stamp
2 Bytes	Archive Sequence number
2 Bytes	Global sequence number
n Bytes	data values for all columns of a record. 1 Byte for Logical Fields / Columns; 4 Bytes for Analog Fields / Columns

8.4.7. Read Records By Sequence Number

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 07
8,9	FILE_NO - Archive file number
10,11	ASEQ - Archive sequence number of record
12	NME - number of records requested

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 07
12	Same as sub-function 06 above

PES	SES	
-007	-001	Invalid file number.

8.4.8. Write File Definition

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 07
7	XSFC- Extended Request Sub-function Code 08
8,9	FILE_NO- file number
10,11	FIELD_CNT- total fields in a record
12,13	FLD_NO- starting field number
14	NME- number of definitions in this message
15	ADFSS- archive definition field selector
16	DATA- file and field definitions

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 07
11	XSFC- Extended Request Sub-function Code 08

PES	SES	
-007	-004	Memory allocation error.
-007	-005	Missing file definition.
-007	-006	Error in definition, cannot use it.

8.5. Read/Reset Communications Statistics

8.5.1. Read Port Communication Statistics

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 00
8	PNUM- port number, 1=A, 2=B, 3=C, etc.

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 00
12	PNUM- port number
13	PTYP - port type xx
14	CTIDX - Index to custom interface table
15-46	Comm statistics
47,48	Custom protocol mode
49-62	Custom protocol name
63	NUMBYTF- Number of bytes to follow
64-83	Port characteristics bytes

Note: Bytes 47 to 62 are only present if this is a Custom port.

Byte 63 is the number of port characteristics that follow; each port characteristic is 2 bytes long. The relation NUMBYTF/2 gives the number of characteristics for this port. The number of characteristics bytes and their content depends on the port type.

The following are some port characteristics for each port type.

BSAP Slave, Serial CFE, Pseudo-slave, Pseudo-slave with Alarms

Number of bytes: 2
Number of characteristics: 1
Bytes 64,65: Baud Rate Index

BSAP Master

Number of bytes: 6
Number of characteristics: 3
Bytes 64,65: Baud Rate Index
Bytes 66,67: High Slave address
Bytes 68,69: Timeout

Expanded BSAP Master

Number of bytes: 8
Number of characteristics: 4
Bytes 64,65: Baud Rate Index
Bytes 66,67: High Slave address
Bytes 68,69: High EASlave address
Bytes 70,71: Timeout

VSAT Slave

Number of bytes: 6
Number of characteristics: 3
Bytes 64,65: Baud Rate Index
Bytes 66,67: Minimum response time
Bytes 68,69: Maximum response time

BSAP Master

Number of bytes: 4
Number of characteristics: 2
Bytes 64,65: Baud Rate Index
Bytes 66,67: Max RIOR Slave node

LOGGER

Number of bytes: 12
Number of characteristics: 6
Bytes 64,65: Baud Rate Index
Bytes 66,67: Character length
Bytes 68,69: Stop bits
Bytes 70,71: Parity
Bytes 72,73: Duplex
Bytes 74,75: Handshake

CUSTOM

Number of bytes: 12
Number of characteristics: 6
Bytes 64,65: Baud Rate Index
Bytes 66,67: Character length
Bytes 68,69: Stop bits
Bytes 70,71: Parity
Bytes 72,73: Duplex
Bytes 74,75: Handshake

8.5.2. Reset Port Communication Statistics

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 01
8	PNUM- port number

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 01
12	PNUM- port number

8.5.3. Read Buffer Usage

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 02

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 02
12,13	BCNT- total buffers allocated
14	Task waiting for buffer
15	Down buffers used
16	Up buffers used
17	Generic buffers used
18,19	Total buffers used
20	Minimum buffer usage
21	Maximum buffer usage

8.5.4. Reset Buffer Usage Counts

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 03

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 03

8.5.5. Read Crash Blocks

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 04

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 04
12	NUM- number of crash blocks. Note: If upper bit of this is set, indicates destination RTU is ARM-based. See Read Hardware Registers message for more information.
13	Number of bytes per crash block
	--- start of crash block groups ---
14,15	Julian date
16,17	Julian time
18-21	EIP
22,23	Code segment
24,25	Crash code
26-29	Active TCB
30	other crash block groups
30+n	crash registers (60 bytes, 4 bytes per register)

Note: register save order is:
EFLAGS, CS, EIP, EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP, DS, ES, FS, GS

8.5.6. Reset Crash Blocks

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 05

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 05

8.5.7. Read Port Summary

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 06

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 06
12	NUM- number of ports in response
13	Port type for port A
14	Custom type for port A
15	Custom protocol for port A
16,17	Characters received port A
18,19	Characters transmitted port A
20	Port type for port B
21	Custom type for port B
22	Custom protocol port B
23,24	Characters received
25,26	Characters transmitted
	: other port groups
	:
133	Port P
134	Custom type for port P
135	Custom protocol for P
136,137	Characters received P
138,139	Characters transmitted P

Note: if a port is not Custom it's Custom protocol byte is 00. Unused ports contain 00 data bytes.

8.5.8. Read Global IP Statistics

IP stands for Internet Protocol; it is a specification which defines the most basic packets of information transported in a TCP/IP network. IP provides an addressing and packet routing mechanisms.

The statistics maintained are as follows:

Request Message Format:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 10H

Response Message Format:

Byte	Content
0	M(s) Message Source Function Code (from request): xx
1,2	SEQ Sequence Number (from request) xx,xx
3	M(d) Message Source Function Code A0H
4	ST Node Status xx
5	PES - Primary Error Status: xx
6	SES - Secondary Error Status: xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function code: 70H
10	XFC - Extended Request Function Code 08H
11	XSFC - Extended Request Sub-function Code 10H
12	Number of statistics returned. Each returned as an unsigned 4 byte value.
13-16	**Packets Received** - Number of data packets received from the Data Link. Invalid packets and packets destined for "pass-thru" are included in this count. Not counted are packets discarded by the data-link due to checksums or length checks performed at that layer.
17-20	**Received with Header Error** - Discards due to header errors: these include invalid IP data length, invalid IP version, and IP header checksum errors.
21-24	**Received with invalid IP Address** - Number of times which a packet was received which is not for the current RTU and the current RTU does not know how to route the packet to the contained address
25-28	**Packets Forwarded** - Number of receive packets not for the current RTU, which have been forwarded to another machine for

processing.

29-32 **Received with invalid protocol** - Discards due to an unrecognized protocol code in the header.

33-36 Unused

37-40 **Packets delivered to stack** - Number of packets properly received, and send on to be processed by a protocol handler.

41-44 **Packet send attempts** - Number of packets which the IP layer has been asked to send. This includes discards. Note: This count does not include send attempts which are discarded by UDP or other higher-level layers.

45-48 **Send Packets discarded** - Packets discarded due to badly formed packets (length errors, bad destination, etc.).

49-52 **Send Packets (No Route)** - Packets discarded because there is no known route to the destination address. Also, increments the 'discarded' statistic.

53-56 **Packet Fragments received** - A data-link does not support sending of an entire large packet in one section; therefore, the source machine has broken it into fragments. This is the total number of these fragments received.

57-60 **Packets assembled from fragments** - The number of packets which have been put together from fragments.

61-64 **Reassembly of packet failed** - The number of times a packet has been discarded due to not receiving all of its fragments within the allotted time.

65-68 **Send Packet fragmented OK** - A data link on the current RTU cannot support sending of full-size packets. This is the number of packets which have been split into fragments for sending.

69-72 **Failed to get packet for fragment** - The number of times which a packet has been discarded due to the IP layer not being able to allocate a send packet for the fragment.

73-76 **Number of send fragments** - The number of packet fragments which have been sent out a data-link.

77-80 Unused

81-84 **Default time to live** - The number of "hops" (sends over data-links) a packet can have before it is discarded. This is not a statistic; but, a display of the default value used by this RTU.

85-88 **Timeout for packet reassembly** - The time between the arrival of the first fragment of a packet to when all fragments must arrive. If the fragments all do not arrive, the fragments are discarded. This is not a statistic; but, a display of the default value used by this RTU.

8.5.9. Read Global ICMP Statistics

ICMP is a low-level IP protocol which performs notification of communications events. Request

Message Format:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 11H

Response Message Format:

Byte	Content
0	M(s) Message Source Function Code (from request): xx
1,2	SEQ Sequence Number (from request) xx,xx
3	M(d) Message Source Function Code A0H
4	ST Node Status xx
5	PES - Primary Error Status: xx
6	SES - Secondary Error Status: xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function code: 70H
10	XFC - Extended Request Function Code 08H
11	XSFC - Extended Request Sub-function Code 11H
12	Number of statistics returned. Each returned as an unsigned 4 byte value.
13-16	**Packets received** - Protocol packets received. This includes discards.
17-20	**Receive packets discarded** - Packets discarded due to length error, ICMP checksum, or invalid request type.
21-24	**Destination unreachable packet received** - Notifications of the following: A packet was sent from this RTU to a destination (either IP address or Protocol Port) which could not be reached.
25-28	**Time to Live Exceeded packet received** - Notifications of the following: A packet was sent from this RTU which was transmitted over too many data-links on the way to its destination (and was discarded).
29-32	Unused
33-36	Unused
37-40	**Redirect packets received** - Requests received to modify internal routing information based on a routing machine determining a better path for a previously sent message.
41-44	**Echo Request packets received** - PING requests received.

PING is a program which sends a simple ECHO packet to another IP machine to determine if communications is possible.

45-48	**Echo Reply packets received** - Number of replies received to 'PING' requests made by this RTU.
49-52	**Time stamp request packet received** - Requests for time-stamp received. Note: this is not the RTU time-synch request.
53-56	Unused.
57-60	Unused
61-64	Unused
65-68	**Packets Sent** - Total ICMP packets sent. Does not include discards.
69-72	**Out packets discarded** -Message discarded due to not being able to allocate send packet.
73-76	**Destination unreachable packets sent** - A packet could not be delivered or forwarded. A notification was sent back to the originator of the packet.
77-80	**Time to live exceeded packets sent** - When forwarding a packet, its time-to-live count was exhausted; a notification was sent back to the originator of the packet.
81-84	Unused
85-88	Unused
89-92	**Redirect packets sent** - When forwarding a packet out the same line it was received over, a notification is sent back to the originating node that a better path is available.
93-96	**Echo request packets sent** - PING requests sent by this RTU.
97-100	**Echo reply packets sent** - PING responses sent by this RTU.

8.5.10. Read Global UDP Statistics

UDP stands for User Datagram Protocol - A method of transmitting user data from one Protocol Port on a computer to another (either on the same or another computer). UDP provides a checksum on the data sent; but, does not guarantee delivery. UDP is connectionless, there is no need to establish a connection before sending data.

Request Message Format:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 12H

Response Message Format:

Byte	Content
0	M(s) Message Source Function Code (from request): xx
1,2	SEQ Sequence Number (from request) xx,xx
3	M(d) Message Source Function Code A0H
4	ST Node Status xx
5	PES - Primary Error Status: xx
6	SES - Secondary Error Status: xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function code: 70H
10	XFC - Extended Request Function Code 08H
11	XSFC - Extended Request Sub-function Code 12H
12	Number of statistics returned. Each returned as an unsigned 4 byte value.
13-16	**Packets received** - Packets received and processed. Does not include discards.
17-20	**Port not present** - A packet was discarded because it was destined for an undefined Port. A ICMP error packet is returned.
21-24	**Receive packet discarded** - Packets discarded due to header or checksum errors.
25-28	**Packets sent** - Packets sent to IP layer for processing.

8.5.11. Read Global IBP Statistics

IBP stands for Internet Bristol Protocol - It is the protocol used inside UDP packets to perform reliable data communication between PCs and IP RTUs. This communication method allows both detection and retry of missed packets, and proper ordering of requests. In addition multiple request (sub-packet) can be combined into a single network packet.

Request Message Format:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 13H

Response Message Format:

Byte	Content
0	M(s) Message Source Function Code (from request): xx
1,2	SEQ Sequence Number (from request) xx,xx
3	M(d) Message Source Function Code A0H
4	ST Node Status xx
5	PES - Primary Error Status: xx
6	SES - Secondary Error Status: xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function code: 70H
10	XFC - Extended Request Function Code 08H
11	XSFC - Extended Request Sub-function Code 10H
12	Number of statistics returned. Each returned as an unsigned 4 byte value.
13-16	**No connection available**. - Number of packets dropped due to not being able to find an inactive connection.
17-20	**Total discards based on mult ACK tmo**. - The number of packets discarded due to exceeding the ACK limit.
21-24	**ACK timeouts** - The number of times that an ACK for a packet was not received within the time-out
25-28	**Discarded by purge operations**. - Packets discarded due to connection inactivity.
29-32	**Discarded due to quota.** - Packets discarded at the RTU due to a shortage of available network packets.
33-36	**Discarded due to sequence #.** - Packets discarded due to sequence #s which were not in the proper range. Note: this can include packets which were resent due to time-out, but already received.

37-40 **Invalid form for packet**. - Packets received with an invalid header length.

41-44 **Invalid identifier for sub-packet.** - Number of sub-packets detected with invalid type code.

45-48 **Packets received out-of-order** - The number of packets which were received out of sequence (and thus loaded onto the out-of-order list). Items are removed from the list when the preceding packets are received.

49-52 **Packets accepted.** - The number of IBP packets accepted for processing

53-56 **Packets sent** - The number of IBP packets given to the IP stack for send.

57-60 **Restart connection.** - Number of times an IBP packet was received which indicated that the local sequence number should be reset.

61-64 **Errors attempting to send packet.** - The number of times the IP stack issued an error while trying to send a packet

65-68 **Sub-packets received.** - Number of IBP sub-packets received from IP stack

69-72 **Sub-packets sent.** - Number of IBP sub-packets given to IP stack for sending.

8.5.12. Reset Global IP Statistics

This request resets the following statistics at the RTU: IP, ICMP, UDP, and IBP.

Request Message Format:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 1FH

Response Message Format:

Byte	Content
0	M(s) Message Source Function Code (from request): xx
1,2	SEQ Sequence Number (from request) xx,xx
3	M(d) Message Source Function Code A0H
4	ST Node Status xx
5	PES - Primary Error Status: xx
6	SES - Secondary Error Status: xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function code: 70H
10	XFC - Extended Request Function Code 08H
11	XSFC - Extended Request Sub-function Code 1FH

8.5.13. Read Hardware Registers (ARM-based – ControlWave Micro, EFM, GFC, GFC-CL, XFC, Express, CW_10, CW_30)

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 08
7	XSFC- Extended Request Sub-function Code 08

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 08
11	XSFC- Extended Request Sub-function Code 08
12	Hardware Type (Firmware Version) xx
13	Number of hardware registers (4 bytes/register) xx
14-17	Register 1 xxxxxxxx
:	
n-n+3	Register m xxxxxxxx

8.6. Read Array Elements

8.6.1. Read Array Elements By Column

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 09
7	XSFC- Extended Request Sub-function Code 00
8	TYPE- Array type 0= Analog, 1= Logical
9	ANUM- array number
10,11	Start row
12,13	Start column
14	NME- number of array elements to read

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 09
11	XSFC- Extended Request Sub-function Code 00
12	NME- number of message elements
13	ANUM- array number
14,15	Next row not in this response
16	DATA- array element data

8.6.2. Read Individual Array Elements

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 09
7	XSFC- Extended Request Sub-function Code 01
8	ATYP- array type 0= Analog, 1= Logical
9	NUM- number of groups in request
10	ANUM- first array number
11,12	Row number
13,14	Column number
15	second array number
16,17	Row number
18,19	Column number
20	other groups

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 09
11	XSFC- Extended Request Sub-function Code 01
12	NGRP- number of groups
13	EER- first EER code
14	first array elements
	:
n	EER- last EER code
n+1...	last array elements

Note: Array element size depends on array type. Analog array elements are 4 bytes long, Logical array elements are 1 byte with the Least Significant Bit state showing the state of the element, 0= OFF, 1= ON.

8.7. Read Special Information

8.7.1. Read Version, Features, PROM Link Date

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 0A
7	XSFC- Extended Request Sub-function Code 00

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 0A
11	XSFC- Extended Request Sub-function Code 00
12,13	MSD Version number (pg0)
14,15	PEI version number (pg0)
16	Runtime system version number 1 (06,106,150)
17	Runtime system version number 2
18	List version number
19	AIC version number
20	Features identifier
21-36	up to 16 bytes, PROM link date
37-52	up to 16 bytes, BOOT PROM ID
53	NPX (Math co-processor) present

Note: PROM & BOOT PROM link date fields:

1	Day	1 byte binary
2	Month	1 byte binary
3-5	Product	3 byte ASCII
6	Major	1 byte binary
7	Update	1 byte binary
8	Beta rev	1 byte binary
9-16	Unused	

8.7.2. Read Custom PROM Information

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 0A
7	XSFC- Extended Request Sub-function Code 01
8	Entry number to start from
9	FSS- selector, 0= no headers, 1= headers

Note: Byte 9 should be 1 if Custom Headers are to be in the response.

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 0A
11	XSFC- Extended Request Sub-function Code 01
12	NUM- number of entries in the PROM
13	Version
14-25	Prom date and identification
26,27	Prom checksum
28	Next entry number not in response
29	Number of entries in response
30,31	Mode value for first entry
32-45	Protocol mode for first entry
	:
n,n+1	Protocol mode for nth entry
n+2,n+15	Protocol mode for nth entry

Note: Byte 13 has the format:

3 = Compatible with pre-AE.00 Firmware
5 = Requires AE.00 or later firmware
7 = Converted to Microsoft Tools/PACLIB

Bytes 14 to 25 PROM link date (as described below):

Byte	Content	
14	Day	1 byte binary
15	Month	1 byte binary
16-18	Product	3 byte ASCII
19	Major	1 byte binary
20	Update	1 byte binary
21	Beta rev	1 byte binary
22-25	Unused	

Bytes 12 to 27 (Custom header) are included only if the FSS byte is 1.

Byte 28 is zero if there are no more entries in the Custom PROM.

8.7.3. Read NRT Information

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 0A
7	XSFC- Extended Request Sub-function Code 02

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 0A
11	XSFC- Extended Request Sub-function Code 02
12	NRT Version
13,14	Global Address
15,16	Up/Down Mask
17	Current Level Number
18-31	Level Shift Count, Masks (7 pairs) - *SEE CHAPTER 4 (D4052) FOR DETAILS*
32-47	16 bytes of next level down
48,63	Driver types
64,65	NRT Status Word

8.7.4. Read On Board Serial EEPROM Information (ControlWave/ControlWaveLP ONLY)

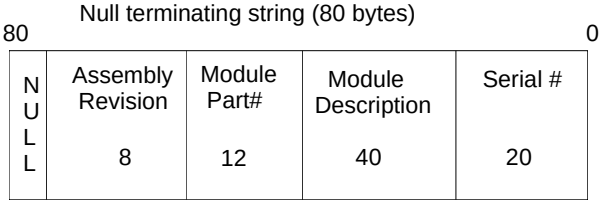
REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 0A
7	XSFC- Extended Request Sub-function Code 03
8	Slot number start (1 based)
9	Number of slots to request (maximum of 2)

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A0H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - firmware version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 0A
11	XSFC- Extended Request Sub-function Code 03
12	Total number of slots in unit
13	NME (number of slots returned)
14	Slot number for string #1
15	Null terminating string #1
16	Slot number for string #2
17	Null terminating string #2

Note: Null terminating strings can be up to 80 characters:



If an error occurs, a null terminating string containing just the error message e.g. 'Board not present', 'Invalid slot #', 'Information not available' will be returned, instead of the string described in the figure above.

8.7.5. Read RTU Hardware/Software Items

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A0H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	FUN Function Code: 70H
6	XFC - Extended Request Function Code 0A
7	XSFC- Extended Request Sub-function Code 04
8-11	4-byte mask indicating which items will be fetched. Although up to 32 (numbered 0 to 31) items can be fetched in one message, the actual number of items in the response is limited by the size of a BSAP buffer.

Item Number (bit)	Description	Length of response (bytes)
0	Read MAC Address	6
1	Flash file size available. This byte contains two LONG values: Offset 0 – Flash space currently in use by flash files. Offset 4 – Space available for additional flash files. Note: this number has file header overhead subtracted already.	8
2	Application Run State. Can be one of the following: 1 System is running 4 Application is loaded 0x10 System is running in DIAG mode.	1
3	System Flash Download Locks This byte is a bit field, and can be one of the following: 1 Switch indicates flash cannot be programmed. 2 Industry Canada application lock set. 4 Switch to enable programming on restart is not set.	1
4	Debug Allowed Switch – reads the current state of the RUN / REMOTE / LOCAL switch:	1

Item Number (bit)	Description	Length of response (bytes)
	0 RUN – Debug not allowed. 1 REMOTE – Local debug access not allowed. 2 LOCAL – Debug access allowed for both local and remote.	

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(s)- Message Source Function Code (from request) xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(d)- Message Source Function Code: A0
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7,8	VER - ACCOL load version number xx,xx
9	FUN - Function Code: 70H
10	XFC - Extended Request Function Code 0A
11	XSFC- Extended Request Sub-function Code 04
12-15	4-byte mask indicating which items could not be fetched because data did not fit in the BSAP buffer xxxxxxxx
16	Number of items in the response xx
17	Item Number (0-31) xx
18	Item's data size in bytes xx
19	Item's data
n	Item Number (0-31).. xx
n+1	Item's data size in bytes xx
n+2	Item's data
:	

This page is intentionally left blank

Appendix C

RDB Extensions For GFC 3308 (DEST EXCH = 0A1)

1. SCOPE

This appendix describes the RDB Extension Facilities created for the 3308 Accurate Gas Flow Computer (GFC). These extensions are supported only by the GFC 3308 firmware under message exchange 0A1 hex.

2. EXTENSION FUNCTIONS

To facilitate special requirements of the GFC 3308 new RDB level functions are added to the GFC 3308 firmware; these functions support a message destination exchange of 0A1 hex. Implementation of these functions does not have any impact on the rest of the RDB functions. The extension supports three special functions which are described later.

GENERIC REQUEST/RESPONSE MESSAGE STRUCTURES:

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A1H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	XFC - Extended Request Function Code xx
6	XSFC- Extended Request Subfunction Code xx
7...	DATA- Request specific data values xx,xx,...,xx

RESPONSE MESSAGE FORMAT:

Byte	Content
0	M(d)- Message Destination Function Code xx
1,2	SEQ - Sequence Number (from request) xx,xx
3	M(s)- Message Source Function Code: A1H
4	ST - Node Status xx
5	PES - Primary Error Status xx
6	SES - Secondary Error Status xx
7	XFC - Extended Request Function Code xx
8	XSFC- Extended Request Subfunction Code xx
9...	DATA- response specific data values xx,xx,...,xx

ERROR CODES:

The PES (this byte is in the same position as the RER byte in standard RDB response messages) has following values (values shown are decimal equivalents of hex values in the response messages):

-127	Unsupported RDB Extension Function Code.
-001	Error with System Reset request.

-002 Error with Date and Time request.
-004 Error with Change Node Address.
-005 Error with Security Code Validation request.
-006 Error with Expanded BSAP Group Number request.

The SES must be interpreted in conjunction with the PES. Following SES error codes are defined:

PES	SES	
000	000	Request processed successfully.
-001	-001	Sub Function Code is not 0.
-001	-002	Bad Confirmation Code or Confirmation received after a timeout.
-002	-001	Bad Value in Date data field.
-002	-002	Bad Value in Time data fields.
-002	-003	Bad Value in Date and/or Time data field.
-004	-001	GFC node address is either 0 or >126 (GFC 3308 only).
-004	-002	Bad subfunction code (GFC 3308 only).

3. RESET SYSTEM REQUESTS

There are two types of reset requests available:

3.1. RESET SYSTEM - NORMAL RESET

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	01
6	XSFC- Extended Request Subfunction Code	01
7	CC - Confirmation Code	xx

RESPONSE MESSAGE FORMAT:

Byte	Content	
0	M(d)- Message Destination Function Code	xx
1,2	SEQ - Sequence Number (from request)	xx,xx
3	M(s)- Message Source Function Code:	A1H
4	ST - Node Status	xx
5	PES - Primary Error Status	xx
6	SES - Secondary Error Status	xx
7	XFC - Extended Request Function Code	01
8	XSFC- Extended Request Subfunction Code	01
9	CC - Confirmation Code	XX

REQUEST DESCRIPTION:

This request forces a reset of the controller. Effect of this request is same as if the system reset switch was pressed. Following the reset and subsequent reset diagnostics the controller waits for the download of an ACCOL load. If the controller already has its ACCOL load in EPROM then the load is initialized and the controller begins normal operation.

REQUEST PROCESSING:

To reset a controller two requests (request and confirmation) are necessary. When the first reset request is received it is validated and if it is a valid request a response is generated. The first request must specify 00 as the confirmation code. Response to the first request includes a confirmation code (CC). The requester must then confirm the reset request within 30 seconds for this reply otherwise the reset request is aborted. The confirmation request must include the CC from the response to the first request.

PES	SES	
-01	-01	Sub function code is not 0
-01	-02	Bad confirmation code or confirmation received after 1 second timeout

Example:

Request: 01 01 00

Response: 00 00 01 01 xx

Confirmation Request: 01 01 xx

Response to confirmation: None. The controller is reset.

3.2. RESET SYSTEM - DIAGNOSTICS RESET

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	01
6	XSFC- Extended Request Subfunction Code	02
7	CC - Confirmation Code	xx

RESPONSE MESSAGE FORMAT:

Byte Content

```
  0     M(d)- Message Destination Function Code  xx
1,2     SEQ - Sequence Number (from request)       xx,xx
  3     M(s)- Message Source Function Code:       A1H
  4     ST  - Node Status                    xx
  5     PES - Primary Error Status       xx
  6     SES - Secondary Error Status       xx
  7     XFC - Extended Request Function Code       01
  8     XSFC- Extended Request Subfunction Code       02
  9     CC -  Confirmation Code                 XX
```

REQUEST DESCRIPTION:

This request forces a reset of the controller and prepares it to run diagnostics. This mode is applicable only when the controller has it's ACCOL load in the EPROM. Effect of this request is same as if the system reset switch was pressed and a diagnostics load was downloaded. Following the reset and subsequent reset diagnostics the selected modules from the ACCOL EPROM load are initialized and other ACCOL functions are not initialized to avoid interference with the diagnostics. The diagnostics can now be run.

REQUEST PROCESSING:

Processing of this request is same as the NORMAL RESET REQUEST described above.

Example:

Request: 01 02 00

Response: 00 00 01 02 xx

Confirmation Request: 01 02 xx

Response to confirmation: None. The controller is reset and selective functions in the EPROM load are initialized.

4. DATE AND/OR TIME REQUESTS

There are six requests available to read or write system date and/or time. They are:

4.1. READ SYSTEM DATE AND TIME:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	02
6	XSFC- Extended Request Subfunction Code	01

RESPONSE MESSAGE FORMAT:

Byte	Content	
0	M(d)- Message Destination Function Code	xx
1,2	SEQ - Sequence Number (from request)	xx,xx
3	M(s)- Message Source Function Code:	A1H
4	ST - Node Status	xx
5	PES - Primary Error Status	xx
6	SES - Secondary Error Status	xx
7	XFC - Extended Request Function Code	02
8	XSFC- Extended Request Subfunction Code	01
9-20	Date and Time: DD MM yy YY HH MN SS jdJD j4sJ4S J20MS	

Where:

HH = hour of the day 1 byte, binary
MN = minutes 1 byte, binary
SS = seconds 1 byte, binary
DD = day of the month 1 byte, binary
MM = month of the year 1 byte, binary
yyYY = year 2 bytes, LSB binary
jdJD = Julian day 2 bytes, LSB binary
j4sJ4S = Julian 4 second intervals 2 bytes, LSB binary
J20MS = Julian 20 milliseconds intervals, 1 byte, binary

NOTE: See the description of time in Chapter 4 for additional information on Julian date and time.

REQUEST DESCRIPTION:

This request reports the current system date and time including the Julian time from the controller.

REQUEST PROCESSING:

Example:

Request: 02 01

Response: 00 00 02 01 DD MM YY YY HH MM SS JD J4S J20MS

4.2. READ SYSTEM DATE:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	02
6	XSFC- Extended Request Subfunction Code	02

RESPONSE MESSAGE FORMAT:

Byte	Content	
0	M(d)- Message Destination Function Code	xx
1,2	SEQ - Sequence Number (from request)	xx,xx
3	M(s)- Message Source Function Code:	A1H
4	ST - Node Status	xx
5	PES - Primary Error Status	xx
6	SES - Secondary Error Status	xx
7	XFC - Extended Request Function Code	02
8	XSFC- Extended Request Subfunction Code	02
9-20	Date and Time: DD MM YY YY HH MN SS JD J4S J20MS XX,...,XX	

REQUEST DESCRIPTION:

This request reports the current system date and time including the julian time from the controller.

4.3. READ SYSTEM TIME:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	02
6	XSFC- Extended Request Subfunction Code	03

RESPONSE MESSAGE FORMAT:

Byte	Content	
0	M(d)- Message Destination Function Code	xx
1,2	SEQ - Sequence Number (from request)	xx,xx
3	M(s)- Message Source Function Code:	A1H
4	ST - Node Status	xx
5	PES - Primary Error Status	xx
6	SES - Secondary Error Status	xx
7	XFC - Extended Request Function Code	02
8	XSFC- Extended Request Subfunction Code	03
9-20	Date and Time: DD MM YY YY HH MN SS JD J4S J20MS XX,...,XX	

REQUEST DESCRIPTION:

This request reports the current system date and time including the julian time from the controller.

4.4. WRITE SYSTEM DATE AND TIME:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	02
6	XSFC- Extended Request Subfunction Code	81
7-13	Date and Time: DD MM YY YY HH MN SS	XX,...,XX

RESPONSE MESSAGE FORMAT:

There is no response to this request.

REQUEST DESCRIPTION:

Sets system date and time as requested. There is no reply to

this request (as this could be a broadcast function). If there is an error in the request then it is ignored and the date and the time are not modified.

Example:

Request: 02 81 DD MM YY YY HH MN SS

Response: None.

4.5. WRITE SYSTEM DATE:

REQUEST MESSAGE FORMAT:

Byte	Content
0	M(d) Message Destination Function Code: A1H
1,2	SEQ Sequence Number xx,xx
3	M(s) Message Source Function Code xx
4	ST Node Status xx
5	XFC - Extended Request Function Code 02
6	XSFC- Extended Request Subfunction Code 82
7-10	Date and Time: DD MM YY YY XX,...,XX

RESPONSE MESSAGE FORMAT:

There is no response to this request.

REQUEST DESCRIPTION:

Sets system date as requested. There is no reply to this request (as this could be a broadcast function). If there is an error in the request then it is ignored and the date is not modified.

Example:

Request: 02 82 DD MM YY YY

Response: None.

4.6. WRITE SYSTEM TIME:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	02
6	XSFC- Extended Request Subfunction Code	83
7-9	Date and Time: HH MN SS	XX,...,XX

RESPONSE MESSAGE FORMAT:

There is no response to this request.

REQUEST DESCRIPTION:

Sets system time as requested. There is no reply to this request (as this could be a broadcast function). If there is an error in the request then it is ignored and the time is not modified.

Example:

Request: 02 83 DD MM YY YY HH MN SS

Response: None.

5. CHANGE LOCAL NODE ADDRESS:

REQUEST MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	A1H
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	xx
4	ST Node Status	xx
5	XFC - Extended Request Function Code	04
6	XSFC- Extended Request Subfunction Code	00
7	New Local Address	na

RESPONSE MESSAGE FORMAT:

Byte	Content	
0	M(d) Message Destination Function Code:	xx
1,2	SEQ Sequence Number	xx,xx
3	M(s) Message Source Function Code	A1H
4	ST Node Status	xx

5	XFC - Extended Request Function Code	04
6	XSFC- Extended Request Subfunction Code	00
7	Old Local Address	oa
8	New Local Address	na

REQUEST DESCRIPTION:

Changes the local node address of the controller to new address.

REQUEST PROCESSING:

Once the local address is changed it becomes effective immediately after the response is sent. The communication will stop until the user switches to using the new address.

PES	SES	
000	000	Request processed successfully.
-004	-001	Specified new address is wither 0 or is greater than 126.
-004	-002	Subfunction code is invalid for this request.

Example:

Request: 04 00 na

Response: 04 00 oa na

This page is intentionally left blank

Appendix D

Report By Exception (RBE)

ABOUT THIS APPENDIX

This appendix describes the functions and top level design of the Report By Exception (RBE) system in the Network 3000 platform. It includes definition of the RBE system parameters and the message structures for RBE interface messages. The message structures show the RBE application data only.

NOTE: For information on RBE in the ControlWave platform, see the online help in ControlWave Designer.

The link, network, and transport level protocols are described elsewhere in this manual. Several common interface scenarios are included.

In addition, a discussion on the configuration and performance of the RBE system is included. Information presented in this document should be sufficient enough to understand how RBE works and to be able to design the RBE Manager with complete interface to the RBE Task.

Note: All references to the signals in this document are to the RBE signals unless explicitly stated otherwise.

OVERVIEW

Traditional data collection systems require the centralized database manager, such as Template Collection function of the Enterprise Server, to periodically poll for all the relevant signals whether they are changed or not. In contrast the RBE system reports only the changes that result in exceptions. These exceptions are reported at predetermined scanning frequencies. This strategy is expected to provide following advantages/disadvantages:

- Reduce processing overhead at the central system as it now processes only the exceptions rather than all signals.
- Reduce network traffic as relatively few signals are transmitted over the network.
- Improve network throughput as network traffic for the data collection is reduced.

In addition, the RBE system can be used to effectively reclassify some of the present day alarm system signals as the RBE signals.

RBE system in each RTU can be configured independently of other RTUs in the network to optimize RBE performance and reduce the effect of RBE activity on other functions of the RTU. This configuration may be done offline at the ACCOL load generation time or online by the RBE Manager.

RBE system includes the following components.

a. The RBE Module: An ACCOL module that does not execute. It provides terminals where the RBE system parameters are defined and status and statistics information is displayed.

b. The RBE Database (DB). It includes:

1. Pointers to the signals declared as the RBE signals.

2. Pointers to the deadband signals or constants for the analog signals.

3. Static and dynamic RBE structures to hold the Last Reported Values (LRV) and other control information.

c. The RBE Task: This is a system task that performs RBE functions. These functions include exception detection, exception report generation, and processing of the interface request/response messages.

d. The RBE Manager: It is a function of the central system, e.g. a sub system of the Enterprise Server. It exercises broad control over RBE activities of each RBE node in the network. The RBE Manager of the Enterprise server supports following major functions:

1. Initializes RBE nodes as it receives WAITING_INIT messages from the.

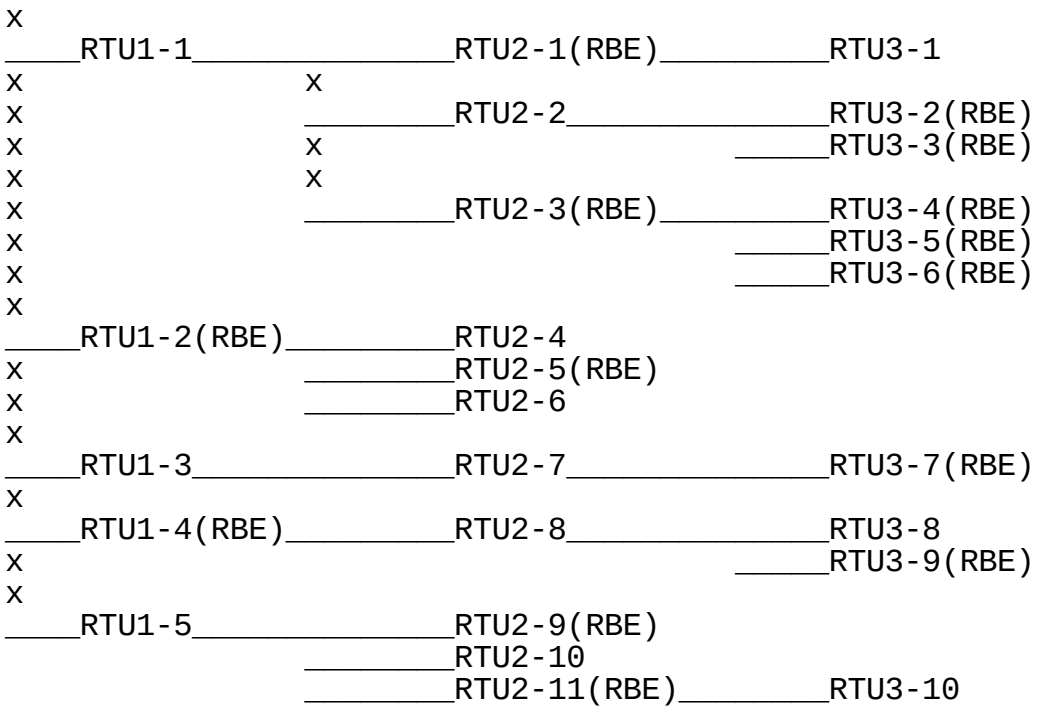
2. Stores RBE data in the Enterprise realtime database.

3. Allows user to configure the RBE Task and monitor RBE activities of RBE nodes.

4. Throttles RBE message flow from each RBE node.

One example of the Bristol Network with RBE system is presented below. Note that it is possible to introduce an RBE node in an existing Bristol Network without any functional impact on any other node of the network. The following figure describes a four layer network. The RBE Manager is at the network central and the three layers below include a total of 14 nodes that are capable of generating Report By Exceptions.

RBE MANAGER
X
X



RBE

A node is defined as an RBE node by including the RBE Module anddeclaring one or more ACCOL signals as the RBE signals. At the RTU initialization time, the RBE Task initializes the RBE system parameters and the RBE DB. Depending on the the value of the MODE terminal, the RBE Task either directly enters the active state or enters the wait state. From the wait state the RBE Task enters the active state after it receives a valid initialization request only.

The RBE Task performs its primary function of generating Exception Report Messages (ERM) while in the active state. In this state theRBE Task periodically scans all signals starting from the first analog signal to the last analog signal followed by the first logical signal to the last logical signal. It selects the signals that are changed since last scan. Selected signal's changes are analyzed to see if an exception has occurred or not. If an exception has not occurred then scan process continues with the next signal. An exception has occurred if any one of the following four conditions is satisfied:

1. One of the current Enable/Inhibit flag is different then the corresponding flag from the last report.

2. The signal is a logical signal and it's ON/OFF state has changed since the last report.

3. The signal is an analog signal and its current value exceeds the Last Reported Value (LRV) + absolute deadband value.

4. The signal is an analog signal, it's deadband signal is not assigned (is unwired), and signal was written to.

When a signal is in exception an Exception Report Element (ERE) is prepared for it. As EREs are prepared they are logged in the ERM buffer and when a buffer is full it is sent to the RBE Manager.

Note: It is important to note that with this design approach it is possible to occasionally miss exceptions if multiple changes occur to the same signal between two successive periodic scans. For example suppose a logical signal was reported being in exception when it was in ON state. This signal's state changes to OFF and then back to ON. If subsequently it is scanned when it is in OFF state then the exception will be reported but if it is not scanned until after it is back to ON state then the exception will not be reported because the current state is same as the LRV thus missing an exception report.

The RBE Manager may acknowledge received ERMs. Occasionally the RBE Manager may send requests for status, demand reports, activate- deactivate signal(s), or reinitialization with different set of parameters. The RBE Task processes these messages and sends appropriate response messages.

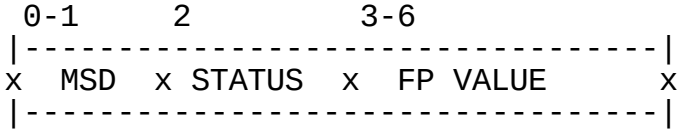
BASIC ELEMENTS OF RBE

First let us look at the ERE, ERM, and other interface request and response messages.

Exception Report Element

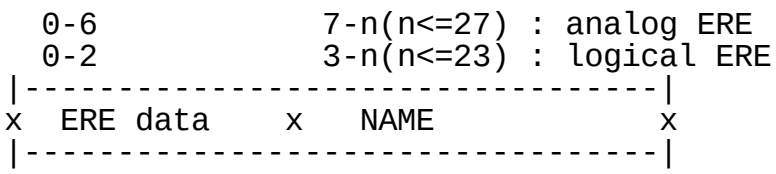
All exceptions are reported as ERE. Two formats are available to report the exceptions: Short Format and Long Format. Selection for the type of format to be used is done either offline at the RBE Module terminal or online with the INIT_REQ by the RBE Manager.

a. Short Format ERE: This format provides the economical way to convey the necessary change information to the RBE Manager. It is 3 bytes in length for a logical signal and 7 bytes in length for an analog signal. A report message has room for 32 analog, 75 logical, or a combination of the two, signals. A short format report includes following information:



```
MSD=     The  Master  Signal  Directory  pointer  of  the
              signal. (2 bytes).
STATUS  =   Signal Status. (1 byte), which includes:
            Bit(s): (bit 0 is low order bit)
                0-1 = Signal type: logical or analog
                        00 = logical
                        10 = analog
                2-3 = Unused
                  4 = Manual Enable/Inhibit state
                  5 = Control Enable/Inhibit state
                  6 = Alarm Enable/Inhibit State
                  7 = OK/Questionable data status
                        for analog signal -OR-
                      logical signal's ON/OFF state
FP VALUE =   Signal value  (4  bytes);  present  only  for
             the analog signals.
```

b. Long Format ERE: This report is composed of the Short Format ERE information plus the signal name text. The signal name text consists of the Base, Extension, and Attribute text with separating periods and a null terminator (Max. 21 bytes). As can be seen this format reduces the throughput and degrades the performance. The Long Format ERE is defined as:




```
ERE Data=   Short format ERE data (3 or 7 bytes)
NAME    =   Signal Name Text (max 21 Bytes)
```

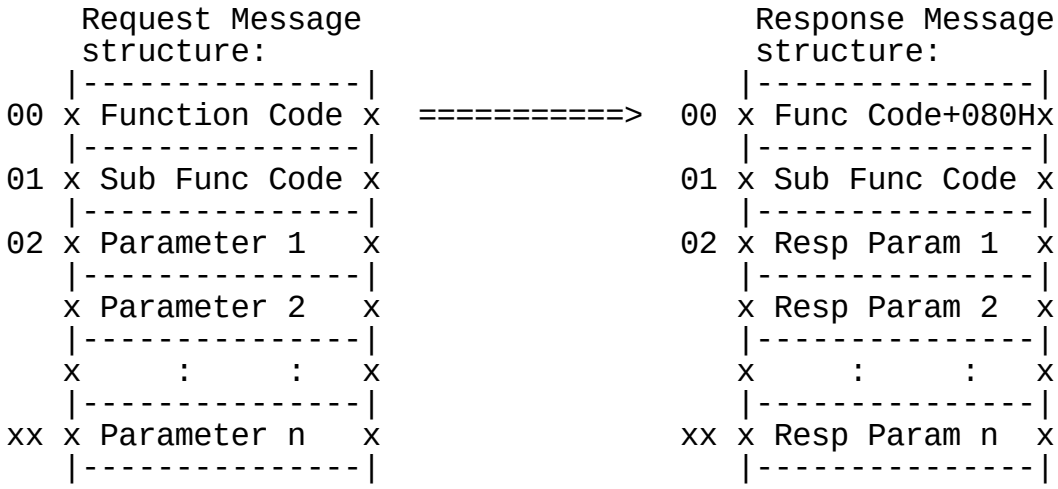
Report Messages

There are four type of Report Messages. They all share same format. However they are generated for different reasons. They are generated in response to INIT_REQ, DEMAND_REQ, or ACTDACT_REQ messages or to report the exceptions. The Report Messages due to request messages include control information and data for one or more signals and are known as Init, Demand, and Active Report Message respectively. The Report Messages generated to report exceptions are known as ERM and include one or more EREs. All ERE included in any given ERM are either in short format or long format but never mixed. Format of the Report Message is described in the next section.

Interface Messages

The interface messages are messages between the RBE Manager and the RBE Task. Each message is assigned a unique function code and can have several sub function codes to identify

different types of requests. All messages confirm to following basic structures:

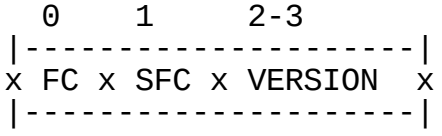


Following Request and Response messages are supported:

1. WAITING_INIT: Waiting for Initialization. This message is sent by the RBE Task to notify the RBE Manager that the RBE Task is waiting for the INIT_REQ message. It is generated when:

- The initialization mode is wait and an RBE node is being initialized after a cold start following a down load
- A switchover has occured in a redundnat pair system
- The RBE Task ennters the reinitialization state because of an error in the previous INIT_REQ or
- A STATUS_REQ is received while waiting for the INIT_REQ.

This message is always sent to the RBE Manager. The ACCOL load version number is the only parameter. It is included in the message to properly identify the ACCOL load.



Where:

FC = 81
SFC = 01 - Waiting For INIT_REQ after Cold Start
= 02 - Waiting for INIT_REQ after switchover
= 03 - Previous INIT_REQ was in error, waiting for an error free INIT_REQ
VERSION = ACCOL load version

2. INIT_REQ: Initialization Request. This message is sent by the RBE Manager to re/initialize the parameters and the RBE DB and activate RBE functions in an RBE node. Normally this message is sent in response to the WAITING_INIT message. It may be sent at any time to change the current parameters or to recover from catastrophic failures.

There are three types of INIT_REQs (distinguished using the Sub Function code):

a. Initialize the RBE system parameters along with the RBE DB and generate Init Report Messages for all RBE signals.

b. Initialize the RBE system parameters along with the the RBE DB but do not generate the Init Report Messages.

c. Initialize the RBE system Parameters only. Do not initialize the RBE DB and do not generate the Init Report Messages.

This message has following parameters. A detailed discussion of relevant parameters is given in a later sections.

- SCANRATE: The minimum delay between the successive starts of the RBE DB scans. It is in 10th of seconds, e.g. 300 = 30 seconds. It can range from 01 (0.1 second) to 65535 (6553.5 seconds). A value of 0 is invalid and results in an error message. A valid value must be specified.

- SCANSLICE: This value is used to divide the work done during a scan session in equal number of blocks known as slices so that the lower priority tasks are able to gain some processing time between each slice. It can range from 1 to SCANRATE. If it is set to 0 or exceeds the SCANRATE then default of 1 (no slicing) is used. Normally it is set to 1 unless RBE is causing performance related problems. When this value is 1 the slicing is inactive.

- FORMAT: This parameter determines the type of format used for the EREs. It can be Short Format (=01) or Long Format (=02). Any other value is invalid and results in an error message.

- STOPXMIT: This parameter is used by the RBE Task as a limit to temporarily stop sending the ERMs. When the difference between the current RSN and the RSN in the last REPORT_ACK message equals this limit the RBE Task suspends the scan process and waits for a valid REPORT_ACK or an INIT_REQ message from the RBE Manager. If this parameter is set to 00 the RBE Task never waits for a REPORT_ACK message for the ERMs. A value greater then 127 is invalid and results in an error message.

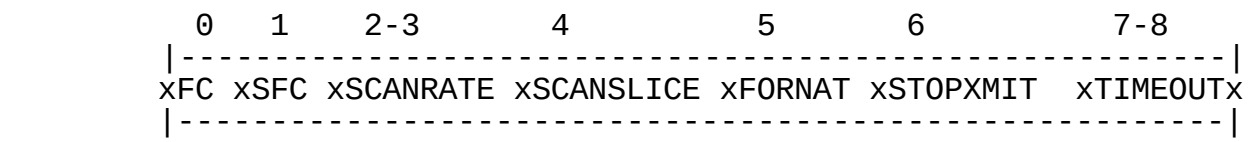
  The RBE Manager can send a REPORT_ACK before the RBE Task
  reaches this limit. For example:

  STOPXMIT limit = 10
  RSN in the last REPORT_ACK = 120
  RSN of the latest ERM reported = 01 (i.e. ERMs with RSNs 121-127,00,01 are sent and pending acknowledgement).
  RSN in the current REPORT_ACK = 125 (i.e. ERMs with RSN 121-125 are acknowledged and ERMs with RSN 126,127,0,1 are to be acknowledged).

  Then the RBE Task will send 6 more ERMs, i.e. until ERM with RSN=7 is sent out. Then it will temporarily stop sending ERMs until a REPORT_ACK (with one of the RSN from 126,127,00-07) is received.

- TIMEOUT: This parameter specifies the period of time to wait for the INIT_REQ before the WAITING_INIT messages is
  repeated to the RBE Manager. It is in 10th of seconds, e.g. 600 = 60 seconds. It can range from 300 (30 seconds) to 65535 (6553.5 seconds). If it is less then 30 seconds then it is set to 30 seconds. If it is 0 then the waiting period remains unchanged from its previous value which could have been set from the RBE Module terminal or by the previous INIT_REQ.

Message Format:



Where:

```
FC          = 03
SFC         = 01 - Do not generate any reports
              = 02 - Generate init reports
              = 03 - Initialize the parameters only
SCANRATE    = Scan period in tenth of seconds
SCANSLICE   = Number of slices (1 <= X <= SCANRATE)
FORMAT      = ERE format type:
                  01 = Short Format
                  02 = Long Format
STOPXMIT    = Limit for required REPORT_ACK
                  00 -    Never wait for a REPORT_ACK
             nn -    Stop    sending    ERMs    after nn
                             reports are   transmitted and
                             wait for a REPORT_ACK (1 <= nn
                             <= 127)
TIMEOUT = Time to wait  between successive WAITING_INIT
                messages.
```

All INIT_REQs result in STATUS RESP (with a FC = 83H) message from the RBE Task. In addition Init Report Messages may be generated if the SFC (=02) was set to generate reports.

3. DEMAND_REQ: A request for the demand reports. This message is sent by the RBE Manager to retrieve all EREs generated since the specified RSN. This message is normally used to recover ERMs lost due to transitory failures, e.g. comm link failure.

The RBE Task replies with the requested ERE in the Demand Report Messages and requires REPORT_ACK from the RBE Manager for each Demand Report Message sent.

```
  0    1     2
|----------------|
x FC x SFC x RSN x
|----------------|
```

```
Where:  FC  = 05
        SFC = 01
        RSN =   Starting ERM Sequence Number
```

4. STATUS_REQ: A request for the current RBE system status. This message is sent by the RBE Manager to collect the status and current statistics. The RBE Task replies with the STATUS RESP message.

```
  0    1
|----------|
x FC x SFC x
|----------|
```

```
Where:  FC = 06
        SFC= 01
```

5. ACTDACT_REQ: A request to activate or deactivate signal(s). This message is sent by the RBE Manager. (Note: The RBE Task performs exception analysis only for the active signals). This message is normally used to turn off signals that may be behaving erratically. The RBE Task replies with the ACTDACT_RESP and if applicable sends the Active Report Messages.

```
  0   1    2          3   4-5  6   7-8  ...
|-------------------------------------------------|
xFC xSFC xNUM ENTRY  xENTRY 1 xENTRY 2 x...xENTRY n  x
x   x    x          xOP xMSD xOP xMSD x   xOP xMSD  x
|-------------------------------------------------|
```

Where:

FC	= 07	
SFC	= 01 - All signals	
	= 02 - All analog signals	
	= 03 - All logical signals	
	= 04 - Selected signals	
NUM ENTRY	= xx - Unused when the SFC = 01-03	
	nn - Number of signals specified in the request (if SFC = 04)	
OP	= Action to take:	
	00 - Activate	
	01 - Deactivate	
MSD	= MSD address of selected signal (present only for SFC=04 requests)	

6. Report Message: This message is from the RBE Task to the RBE Manager. It is used to report the exceptions, current values in response to the INIT_REQ and ACTDACT_REQ, or previously reported values in response to the DEMAND_REQ. The ERMs are always sent to the RBE Manager where as all other Report Messages are sent to the requester.

```
     0     1     2-3   4-5        6-10  11     12          13       14
...............
   |-------------------------------------------------------------
|
   xFC    xSFCx    NA    xVERSION    xTS    xRSN    xNUM    EREXFLAG
xERE1xERE2x..xERENx
   |-------------------------------------------------------------
|
```

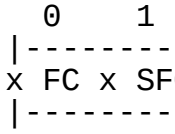
Where:	FC	= 85	
	SFC	= 00 -	Normal ERM; Result of Periodic Scan and Analysis.
		= 03 -	Result of INIT REQ.
		= 05 -	Result of DEMAND REQ.
		= 07 -	Result of ACTDACT_REQ.
	NA	=	BASAP Network Global Address of an RBE node.

VERSION = ACCOL load version.
TS = Julian Date/Time:
Byte 6-7: Julian Date; Number of days since December 31, 1976
Byte 8-9: Julian Time; Number of 4 second intervals since 00:00:00
Byte 10: Num of 20 msec intervals since the last 4 second interval
RSN = Report Sequence Number.
NUM ERE = No. of ERE in this ERM
FLAG = Continuation Flag (bit 7) + Format Type:
01H = Short format + This is last Report
02H = Long Format + This is last Report
81H = Short format + More Report Follows
82H = Long Format + More Report Follows
(Bit 7 is always set to 0 for the ERMs; i.e. FLAG=01/02)

EREx = One or more ERE entries. See ERE section for the formats.

7. REPORT_ACK: Acknowledgement to the Report Message(s). This message is sent by the RBE Manager or any other requester to the RBE Task. It is an acknowledgement for one or more Report Messages. Reports generated for the INIT_REQ, DEMAND_REQ, and ACTDACT REQ require this acknowledgement for each Report Message.

The requirement for the REPORT_ACK for the ERMs depends on the STOPXMIT limit.



Where: FC = 08
SFC = 01
RSN = Report Sequence Number of the Report Message being acknowledged. All previous Report Messages are considered acknowledged.

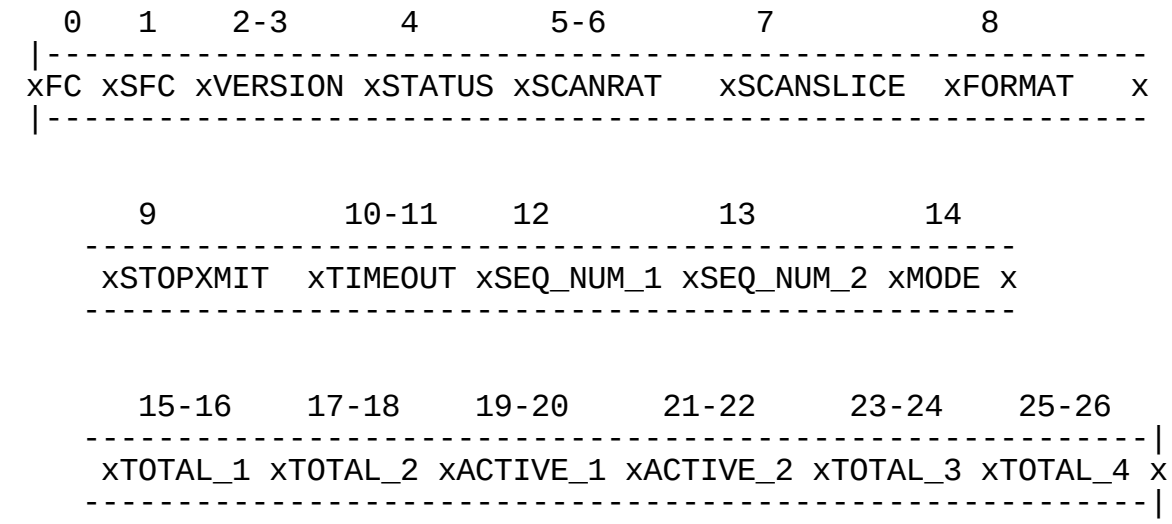
8. STATUS_RESP: A response message. This message is sent by the RBE Task:

a. In response to the STATUS_REQ. Sent to the requester.

b. To report the completion of the initialization whether it is because of Going Active or as a result of the INIT_REQ. If it is to report the Going Active then it is sent

to the RBE Manager otherwise it is sent to the requester.

c. When an unwanted message is received while waiting for a REPORT_ACK or an ACTDACT_REQ. Sent to the requester.



Where:

FC = 83 - Result of the INIT_REQ or Going Active
= 86 - Result of the STATUS_REQ
SFC = 01 - Init Completed - No Reports
= 02 - Init Completed - Report Follows
= 03 - Init Completed - Going Active
= 04 - Init Completed - Only Parameters
= 05 - Response to the STATUS_REQ
VERSION = ACCOL load version
STATUS = See section on "STATUS AND ERROR CODES".
SCANRATE = Scan Period being used
SCANSLICE = Scan Period Slices count
FORMAT = ERE Format being used
STOPXMIT = STOPXMIT limit being used
TIMEOUT = Wait Period between WAITING messages
SEQ_NUM_1 = Last RSN number sent by the RBE Task
SEQ_NUM_2 = RSN from the last REPORT_ACK message
MODE = Initialization Mode
TOTAL_1 = Total analog RBE signals
TOTAL_2 = Total logical RBE signals
ACTIVE_1 = Active analog RBE signals
ACTIVE_2 = Active logical RBE signals
TOTAL_3 = Analog EREs generated (0-65535)
TOTAL_4 = Logical EREs generated (0-65535)

9. ACTDACT_RESP: The response message to an ACTDACT_REQ request. This message is sent by the RBE Task. It includes the completion code and if one or more request entries were in error then the MSD addresses of the erroneous entries.

```
  0    1    2       3      4   5-6  7   9-10 ...........
|-------------------------------------------------------|
xFC  xSFC xSTATUS xNUM   xENTRY 1 xENTRY 2 x... xENTRY n x
x    x    x       xENTRY xOP xMSD xOP xMSD x    xOPx MSD x
|-------------------------------------------------------|
```

Where:

FC = 87
SFC = (Same as ACTDACT_REQ message)
STATUS = 00 - Successful processing
= XX - Error Code: See ACCOL Reference Manual.
NUM ENTRY = Number of error entries included in the response (present only if STATUS != 0).
OP = Operation code for the error entry
MSD = MSD address of the error entry

10. ERROR_RESP: General purpose error response to any erroneous request message. The message includes the Function Code, specific error as Sub Function code, and copy of the received message. Following error responses are possible:

a. ACTDACT_REQ = Bad SubFunction Code
= Bad Op-Code in a group requests (SFC=01-03)
= Bad Number of Entry when the the request is for selected signals
= Bad Entry (Op_Code or MSD error)

b. DEMAND_REQ = Bad SubFunction Code
= Bad RSN (> 127)

c. INIT_REQ = Bad SubFunction Code
= Bad Scan Period (== 0)
= Bad Report Format (not 01 or 02)
= Bad STOPXMIT limit (> 127)

d. REPORT_ACK = Bad SubFunction Code
= Bad RSN (not between Last ACKed RSN and Current RSN)

e. STATUS_REQ = Bad SubFunction Code

f. Unknown = Bad Function Code

g. Any message = Message received over the Pseudo Slave port

```
  0    1     2
|--------------------------|
x FC x SFC x MSG Text      x
|--------------------------|
```

Where: FC = FF
SFC = XX - Error Code: See Status and Error Codes section.
MSG = Copy of the received message starting from "FC, SFC, ..."

RBE Database

The RBE DB is always defined in the Expansion RAM. Each analog signal requires fourteen bytes and each logical signal requires six bytes. The RBE DB is made up of static and dynamic information for each RBE signal as described below.

a. Static Information (prommable):

1. MSD pointers to the signal's static and dynamic areas.

2. For analog signals: MSD pointer to the deadband signal's static and dynamic areas.

b. Dynamic Information (non prommable):

1. Last Reported Value (LRV):

A. Status byte to hold last reported status information:

1. Signal type indicator: logical (00) or analog (10).
2. Enable/Inhibit status bits for manual, control, and alarms.
3. If analog signal then OK/Questionable data status, if logical then signal's ON/OFF status.

B. If analog signal then the last reported floating point value.

2. Control byte:

A. Signal's RBE active/inactive state. All signals are set to active state at the cold start after a download. Subsequently this active/inactive state is controlled by the RBE Manager only. This state can be manipulated using the ACTDACT_REQ message. The state of signal does not change following a switchover, i.e. if a signal was inactive when a switchover occurs then it will remain inactive in the new active controller.

B. The RSN of the ERM in which the signal was reported last.

FUNCTIONAL DESCRIPTION

Initialization

Exception report generation begins only after RBE system is initialized and enters the Active State. At the RTU initialization time if the MODE is selected to Go Active then the RBE Task first validate the RBE Module terminals that are present. It assumes default for the terminals that are unwired or has invalid values and then initializes the RBE structures. Next it checks for the presence of a valid Node Routing Table (NRT). The NRT is required as all message communication is done in the global mode. If the RTU has not received a valid NRT yet, it waits in a permanent loop periodically (every 5 seconds) checking to see if a valid NRT is received. This loop is exited only when a valid NRT is detected. The initialization is then continued in one of two ways:

a. The Mode terminal of the RBE Module indicates (=01) that RBE system is to be initialized and become active immediately AND it is not a switchover. This is the only time when the RBE Module's input terminals (SCANRATE, SCANSLICE, STOPXMIT, and FORMAT) are used as input. The TIMEOUT terminal is used initially and then it may be overwritten by the INIT_REQ. The RBE database is initialized using the current value and status' for each signal.

The initialization completed (STATUS_RESP - FC=83 and SFC=03) message is sent to the RBE Task, a periodic timer with SCANRATE timeout value is started, and the task state is changed to Active Idle State.

b. The Mode terminal of the RBE Module indicates (=00) that RBE system is to wait for initialization

- OR -

A switchover has occurred.

In this case all the input parameters, with the exception of the TIMEOUT, defined at the RBE Module are ignored and the RBE Task sends the WAITING_INIT message to the RBE Manager and starts a periodic timer (using the value specified at the TIMEOUT terminal or a default time of 1 Hour). The WAITING_INIT message is repeated if the periodic timer expires and a valid INIT_REQ is not received yet.

If a message other then the INIT_REQ is received the message is ignored, the WAITING_INIT is immediately sent in reply, and the periodic timer is restarted.

If an INIT_REQ message is received the message and parameters are validated. If there is an error the ERROR_RESP message is sent, followed by the WAITING_INIT message, and the periodic timer is restarted.

When an error free INIT_REQ message is received a STATUS_RESP message is sent to the requester and the request is processed based on the Sub Function Code:

SFC = 01 - The RBE DB is initialized. The current value and status' for each signal are stored in the

RBE DB. A periodic timer with SCANRATE timeout value is started, and the task state is changed to Active Idle State.

SFC = 02 - The RBE DB is initialized. The current value and status' for each signal are stored in the RBE DB and at the same time the Init Report Messages are prepared in parallel. The report elements are in the latest format specified by the INIT_REQ. When the Init Report Message is full it is sent to the requester (generally the RBE Manager). If there are more report elements to send then the continuation flag of the Init Report Message is set to indicate this.

A REPORT_ACK is required for each Init Report Message transmitted. When the REPORT_ACK is received the initialization and report generation continues. When the initialization is complete a periodic timer with SCANRATE timeout value is started, and the task state is changed to Active Idle state.

When the RBE Task is waiting for the REPORT_ACK message it handles all interface messages as follows:

1. If the REPORT_ACK is received it is processed and the initialization is continued.

2. If an INIT_REQ message is received restart the initialization process.

3. For any other request message send the STATUS_RESP message. The status code in this message will be set to one of the initialization state values. The RSN in the response message indicates for which the RBE Task is waiting for the REPORT_ACK message.

3.2.2. Active State

The RBE Task enters this state from either the Initialization state or from the Inactive state. Once in this state it performs the following basic functions of RBE system:

a. SCANRATE Timer: When this periodic timer expires -

1. Reschedule the periodic scan session timer with a timeout value of SCANRATE.

2. If the SCANSLICE terminal is not 1 then activates the slicing mechanism. A periodic timer with a timeout value of "slice period" is started. (Slicing is described in detail in the SCANSLICE section).

3. Perform the scan session: examine the RBE DB and generate ERMs as appropriate.

b. Scan Session: In a given scan session first analog signals are processed followed by the logical signals. First a signal's active/inactive state is checked in the RBE DB. If a signal is marked as inactive then it is skipped. Next it is chacked to see if an active signal's change flag is set. The RTU DB system sets an ACCOL signal's change flag whenever it is written into whether it is for a data value or for any enable/inhibit flag. In this sense all changes are detected in realtime.

If a signal is active and it is identified as changed then it is a candidate for the report. First it's change flag is reset so that new changes can be processed during the subsequent scan session. Now changes are analyzed to see if an exception has occurred or not. If an exception has not occurred then scan process continues with the next signal. When a signal is in exception an ERE is prepared for it. Its current value/state and status information are saved as LRV in the RBE DB. The LRV is then used for change analysis in the subsequent scan sessions.

A determination is made to see if there is a room for the prepared ERE in the ERM buffer. If there is enough space then the ERE is logged in the ERM buffer. The RSN of this ERM is assigned to the signal in the RBE DB and the scan session is continued.

If the prepared ERE can not fit in the ERM buffer then the ERM is transmitted as described below. It is the message transmission phase that handles the STOPXMIT limit and scan slicing. The continuation flag for the ERM is always set to 0. The pending ERE is logged in the new ERM and the scan session is continued with the next signal.

When all signals in the RBE DB are analyzed and the last ERM is transmitted the scan session is terminated. The task state is changed to Active Idle State where it waits for the SCANRATE periodic timeout or an interface message from the RBE Manager.

c. ERM Transmission: ERMs are always sent to the RBE Manager at the Network Master (BSAP Network global address of 0).

Prepared ERM is transmitted. If the STOPXMIT limit is non zero and the number of the ERMs transmitted since the last acknowledged ERM equals the STOPXMIT limit, the RBE Task enters a wait state where it requires a REPORT_ACK message. The task will remain in this state and process the following events:

1. A valid REPORT_ACK message is received. The number of ERMs transmitted count is adjusted based on the acknowledged RSN.

2. An INIT_REQ is received. Reinitializes RBE system as described earlier and enter the Active State.

3. A DEMAND_REQ is received. The scan session is aborted and demand request processing is started.

4. ACTDACT_REQ or STATUS_REQ is received: The STATUS_RESP message with the RSN of the last ERM is sent to the requester and remains in the wait state.

If the scan slicing is active (refer to section on SCANSLICE) and the number of ERMs transmitted equals the limit "messages per slice" then the work of the current scan slice is finished and the RBE Task waits for the periodic "slice period" timer to expire. During this state interface messages are NOT processed.
When the slice timer expires it is rescheduled and the scan process is resumed.

d. Control Message interface: While waiting for the SCANRATE timeout the RBE Task may receive interface messages which are processed as follows:

1. INIT_REQ: Reinitializes RBE system as described earlier and enters the Active State.

2. DEMAND_REQ: When this request is received the RBE Task scans the RBE DB for the signals whose assigned RSN is the same or newer then the requested RSN. The RBE DB is scanned in sequential order from the first analog signal to the last logical signal. When such a signal is found a report element is prepared. As a result the order of demand report elements most likely will not be the same as the order of EREs in the original ERMs. As report elements are prepared they are logged in the Demand Report Message buffer and as long

as the buffer is not full the processing continues. If the buffer is full and the prepared report element can not fit in this buffer, the continuation flag is set and the Demand Report Message is sent to the requester. The RSN assigned to the Demand Report Messages are random and explained later in the RSN section. The RBE Task then waits for the required REPORT_ACK message and processes the following events:

A. A valid REPORT_ACK message is received. Log the pending demand report element in the next Demand Report Message buffer and continue with the demand request processing.

B. An INIT_REQ is received. The RBE system is reinitialized.

C. Other Messages: The STATUS_RESP message is sent to the RBE Manager and waits for one of these three events. The RSN in the STATUS_RESP is the RSN of the last Demand Report Message for which the RBE Task is waiting for the REPORT_ACK message.

After the REPORT_ACK for the last Demand Report Message is received the state is changed back to the Active State where the RBE Task waits for the scan session timer to expire or an interface message to arrive.

Note: After processing a DEMAND_REQ the RSN of the next Scan Report Message will be the RSN of the last Demand Report Message + 1. In the event this results in 128 the RSN of 0 will be used.

3. STATUS_REQ: When this message is received the RBE Task sends the STATUS_RESP in reply. It is sent to the requester.

4. ACTDACT_REQ: Activates or deactivates one or more signals as requested. Process following different type of requests:

A. Activate all signals. Sets status of all signals to active.

B. Activate all analog signals. Sets status of all analog signals to active.

C. Activate all logical signals. Sets status of all logical signals to active.

D. Deactivate all signals. Sets status of all signals to inactive.

E. Deactivate all analog signals. Sets status of all analog signals to inactive.

F. Deactivate all logical signals. Sets status of all logical signals to inactive.

G. Activate/Deactivate selected signal(s). Sets status of the specified signals as requested.

If an op code is not activate / deactivate or the MSD address is not found in the RBE DB then it is returned in the ACTDACT_RESP to indicate an error with that entry.

When the request processing is complete the ACTDACT_RESP is sent to the requester. If one or more signals were set to active by the request then Active Report Messages are generated. This report generation process is similar to the process of the Init Report Message generation. The RSN assigned to the Active Report Messages are sequential and follow the same rule as the ERMs.

After the REPORT_ACK for the last Active Report Message is received the state is changed to Active State where the RBE Task waits for the scan session timer to expire or an interface message to arrive.

If as a result of the request processing all signals were found to be inactive then the RBE Task enters Inactive State.

Inactive State

The RBE Task enters and remains in this state when all signals in this node are set to inactive. While in this state the RBE Task responds only to the following interface messages:

a. ACTDACT_REQ. Process this request message and if one or more signals become active then start the periodic timer with SCANRATE and change the state to Active Idle state.

b. INIT_REQ. Reinitialize RBE System and reenter the Inactive State as no signals are active.

c. Other Interface Messages: Generate the STATUS_RESP message indicating the Inactive State.

DEFINITIONS

RSN

The RSN is a sequence number used to identify a given Report Message. It ranges from 0 to 127. RBE maintains a

variable called current RSN that is set to the RSN of the last Report Message generated. The current RSN is always displayed at the SEQ_NUM_1 module terminal. Following rules govern the RSN:

a. Following an initialization the current RSN is set to 0. If the INIT_REQ is for "parameter initialization only" then the current RSN is NOT reset and remains unchanged.

b. RSN of the Active, Init, and Exception Report Messages is assigned as follows. The current RSN is incremented by 1. If it equals to 128 it is reset to 0. Thus valid RSNs are 0-127 only.
The current RSN is then assigned to the new Report Message. It is also assigned to all signals, that are logged in this Report message, and along with its LRVs it is stored in the RBE DB.

c. The RSNs of Demand Report Messages, except for the last one, come from the RBE DB. When the last signal is stored in a Demand Report Message its associated RSN from the RBE DB is assigned to the Demand Report Message. RSNs in the RBE DB are not changed.
The RSN of the last Demand Report Message is same as the RSN of the last Report Message that was sent prior to receiving the DEMAND_REQ. The REPORT_ACK must use the RSN of the received Demand Report Message. Interim to the demand request processing the current RSN is set to the RSN of the latest Demand Report Message sent and it is displayed at the SEQ_NUM_1 module terminal.

d. The RSN included in a STATUS_RESP is the value of the current RSN, i.e. the RSN of the most recent Report Message except right after initialization when it is set to 0 as Report Messages are not yet generated.

e. A REPORT_ACK message for ERMs may include an RSN that is equal to the current RSN or is a valid RSN between the current RSN and the RSN used in the previous REPORT_ACK message.

f. A REPORT_ACK for the Init, Demand, or Active Report Messages must use the RSN of the received Report Message.

Examples:

ERMs are being sent from a scan session:

```
STOPXMIT limit      =  32
Last REPORT_ACK RSN = 120
Current RSN         = 015

Valid RSN for the REPORT_ACK are 121-127,00-15.
```

Report Messages are being sent for INIT_REQ, DEMAND_REQ, or ACTDACT_REQ:

```
STOPXMIT limit      = XX  (not applicable)
Last Report_ACK RSN = 33
Current RSN         = 34

Valid RSN for the REPORT_ACK is 34 only.
```

SCANRATE

The SCANRATE is a timeout value. It is a periodic timeout value used as a delay between the start of two successive scan sessions. This is a critical parameter. Following table describes the relationship between the value of this parameter and RBE behavior.

Effect on:	Value of the SCANRATE Increases	Value of the SCANRATE Decreases
Rate at which Report Messages are generated	Decreases	Increases
Number of Exception Report Messages per scan session	Increases	Decreases
Num of ERE per Report Messages	Increases	Decreases
Efficiency of Exception Detection	Decreases	Increases

When the slicing is not active (see SCANSLICE section) the scan session is started following the periodic scan session (SCANRATE) timeout and finished in a single session. Only the STOPXMIT limit is considered.

When the MODE terminal is set (=1) to Go_Active then the value specified at the SCANRATE terminal of the RBE module is used otherwise the value of the SCANRATE is set by the INIT_REQ only. The value of the SCANRATE is interpreted in 10th of seconds. For example to specify a delay of 60 seconds it must be set to 600.

If this period is set to less then or equal to the time required to perform a single scan session then as soon as one session is finished the next one is started leading to continuous RBE actions.
When this happens it most likely will degrade the performance of the lower priority tasks in the RTU. The lower priority ACCOL tasks may start to slip. On the other hand too large a

value for this parameter will result in missed exceptions and slower rate at which the RBE DB is examined and reports are generated.

Thus it is critical to decide on the value for this parameter judiciously. It is not possible to give exact recommended value for this parameter, however, following suggestion will allow for a proper value selection.

Activate all the tasks and other functions in an RTU leading close to the maximum load on the processor. Now experiment with different values for the SCANRATE parameter. Lowest SCANRATE value when there is no slippage of ACCOL tasks is the appropriate value.

SCANSLICE

There may be situations when the RBE Task requires a large amount of processing time to scan the RBE DB and generate necessary ERMs. This can happen when the RBE DB is very large or lots of signals are in exception. If this time was used by the RBE Task in one continuous chunk then it is more then likely that some lower priority tasks will be deprived of the timely schedulting and/or required processor time to complete their work on time. The ACCOL tasks reflect this via slippage and the system task reflect this via degraded performance. Slicing may be the solution to resolve this problem.

This SCANSLICE parameter allows the RBE Task to break up it's work load for each scan session into SCANSLICE number of slices. Each slice is governed by two calculated parameters: slice period and number of ERMs per slice. Simply stated each slice generates exactly "ERMs per slice" number of ERMs and starts at the regular interval of "slice period". Breaking down a single scan sessions into smaller slices allows the lower priority tasks to gain processing time more frequently between each slice and thus improving their performance.

Disadvantage of using this mechanism is that an artificial delay is introduced between the ERMs of a single scan session. However, overall RTU performance may necessitate use of this mechanism.

The SCANSLICE parameter may be set at the RBE Module terminal. It can not be changed at the RBE Module terminal. It can be set/reset via the INIT_REQ. The slicing is active only when the value of the SCANSLICE parameter is greater than 1 and less than or equal to the value of the SCANRATE. In all other cases the slicing is inactive.

If slicing is active then following two parameters are calculated for each scan session.

```
slice period    = SCANRATE / SCANSLICE;
```

ERMs per slice = MAX (1, (total ERMs transmitted during last scan session / SCNSLICE));

If (Total ERMs transmitted during last scan session does not exactly divide into SCANSLICE) then add 1 to ERMs per slice.

Each scan slice starts at regular interval of calculated value of "slice period". The slices are repeated until entire RBE DB is scanned when the overall SCANRATE timeout is applied for the next scan session.

The slicing mechanism is ineffective if total (wall clock) time required to generate the "ERMs per slice" exceeds the "slice period". When this happens the next slice starts immediately after the previous slice ends and does not free up any processing time between slices.
Example:

```
SCANRATE    =       300     (30 seconds)
SCANSLICE   =       0
SCANTIME    =       100     (100 milliseconds)
Average no. of ERMs per scan session = 32.
```

Above condition will cause the ACCOL Tasks with lower priority running at .1 second rate to slip. To avoid this condition set the SCANSLICE to 4. The scan session is broken down in 4 equal sub-sessions starting at 7 second interval:

```
SCANRATE    =       300
SCANSLICE   =       4
SCANTIME    =       30      (milliseconds)
Average no. of ERMs per sub session = 8.
```

Eight consecutive ERMs are generated as fast as the comm system responds to the transmission requests and then the RBE Task waits for scan slice timer (7 seconds) to elapse from the "start" of the just completed sub-session.

STOPXMIT limit

This parameter is useful in controlling the message flow and in limiting the maximum number of messages that can be lost during communication and/or intervening master RTU failures. It is also an effective tool to suspend/throttle RBE functions of a node in controlled manner.

When this parameter is set to 0 the REPORT_ACK message is not required. If a REPORT_ACK is received it is processed but does not serve any useful function. The RBE Task generates ERMs and never enters a wait for the REPORT_ACK message. If the RTU's master node or any of the intervening master node up the network hierarchy goes down the ERMs may be lost. The rate of lost messages may be slowed as when the communication is lost the BSAP Comm Subsystem does not return the buffer to the RBE

Task until the port timeout occurs and the RBE Task is unable to generate the ERMs until the previous buffer is available. (This is true only if the immediate master is not communicating).

Even though the messages are lost the actual LRVs are not lost because they are stored in the RBE DB. The RBE Manager may use the RSNs and the DEMAND_REQ to retrieve lost exception reports. However, during extended failures there is a danger of losing 128 or more ERMs. When this happens it is misleading to use the RSNs to determine the lost messages.

When the STOPXMIT parameter is set to a value from 1 thru 127 the RBE Task sends at most set number of ERMs then halts the scan session and waits for the required REPORT_ACK message, thus limiting the number of ERMs that can be lost during failures.

To minimize the number of messages lost and maintain the efficiency following strategy is recommended. Set the STOPXMIT parameter from 1-127 depending on the reliability of the network, with higher the reliability higher the limit. Once a limit is set the RBE Manager should send a REPORT_ACK several counts prior to reaching this limit. Advantage of this strategy is that the RBE Task does not have to wait for the REPORT_ACK. Disadvantage is that a periodic REPORT_ACK is required.

To suspend RBE activities of a node simply do not send the required REPORT_ACK until it is desired to restart RBE activities.

To achieve the hand shake effect set this limit to one. This results in a REPORT_ACK per ERM.

PROBLEMS AND SOLUTIONS

Error conditions are rare but when they occur it should be relatively easy to recover with built-in defense mechanism. This section describes common error conditions or RBE node status changes and suggested recovery strategies if applicable.

a. Missing ERM(s): Occasionally an Exception Report Message is lost due to temporary communication failure either in a passthru node or in the immediate master node. When this happens the RBE Manager will notice a gap in the received RSNs. Type of recovery action depends on the number of ERMs lost.

If the number is small then it is better to use the DEMAND_REQ message to retrieve the ERE associated with the missing ERM(s).

If the number is large then send the INIT_REQ (with SFC=02). This will reinitialize and generate the Init Report Messages with the current value/status' for all signals or send the ACTDCAT_REQ (with SFC=01, OP=01) to activate all signals. This results in Active Report Messages (current values) for all signals.

Specify an appropriate value for the STOPXMIT parameter / RBE module terminal to limit the maximum number of ERMs that can be lost.

b. Too Few EREs Per ERM: There are two causes for this situation.

1. The signals are not being changed frequently. As long as all expected exceptions are being reported nothing needs to bedone. However, if the RBE Task seems to be missing some exceptions then lower the value of the SCANRATE with due regard to overall performance of the ACCOL load.

2. The SCANRATE is set too low and as a result the number of changes detected during a scan session is also small.

If timing is not critical and the buffer occupancy is of importance then increase the SCANRATE value. Experiment with this value so that buffer is filled as much as possible yet at the same time exceptions are not missed for rapidly changing signals.

c. A signal is never reported: This could be due to one or more of the following reasons:

1. The signal is not active.

2. The signal is not written into. Nothing to do.

3. The signal goes into exception but returns back to normal (out of exception) by the time it is analyzed by the RBE Task. Decrease or increase the SCANRATE as appropriate so that the signal is analyzed when it is in exception.

4. If applicable, signal's deadband is set too high.

d. Degraded Performance: RBE may be spending too much time during the scan session analyzing the RBE DB and generating the ERMs.

This may be depriving other system and ACCOL tasks with equal or lower priority of the processing time. This is observed at the SCANTIME terminal, performance of the system tasks, e.g. RDB, and/or slip counts for the ACCOL tasks. If RBE is the cause then this situation can be improved by adjusting the values of SCANRATE and if necessary activating the slicing.

e. More than one ERM per scan session but only few EREs per ERM:

Must be using long format for EREs. Change to short format. When using the short format there is room for 32 analog EREs, or 75 logical EREs, or a combination of the two. The Long format is recommended only when the RBE Manager needs to resolve the signal names to the signal's MSD addresses. Once the signal

names are resolved the format type should be changed to short format (use INIT_REQ with SFC = 03 to modify the parameters only).

f. No ERMs at all: In order for the ERE/Report Messages to be generated all of the following conditions must be satisfied.

1. At least one or more signals are active. If defined, the value of the ACTIVE_1 or ACTIVE_2 terminal must be non zero.

2. One or more of the active signals are going into exception: deadband violation, logical ON/OFF, or control flags changing.

3. One or more signals are in exception when they are analyzed.

4. There are no related BSAP network problems.

Verify that all of the above conditions are satisfied. If all of these conditions are satisfied and ERMs are still not generated then reset the SCANTIME terminal and see if it changes within SCANRATE time. If it does not change then RBE is not performing its functions. It is time to reinitialize RBE. If problem persists after reinitialization then it is time to report the problem.

g. WAITING_INIT is not received by the RBE Manager:

1. Check the repeatation frequency at the TIMEOUT terminal or set by the pervious INIT_REQ. It is used as the delay between two successive WAITING_INIT messages. Set it to desired value.

2. There may be BSAP network problems.

Normally the RBE Manager should send a STATUS_REQ whenever in doubt about the status of an RBE node. If the RBE Task is waiting for the INIT_REQ it will immediately send the WAITING_INIT message otherwise it will send the STATUS_RESP with the current RBE status information.

h. An RBE node fails: When a previously failed RBE node is restored back to service it enters the RBE network as if it is coming online for the first time. The normal RBE Manger interface will or initialization MODE selection will get RBE activities started.

i. Redundancy Switchover: When the active processor of a redundant pair goes down the backup processor switches over to active role.

Following this switchover, regardless of the value of the MODE terminal, the RBE Task performs the cold start for a switchover.

COORDINATION BETWEEN RBE AND RDB

The RBE System is independent of the RDB system. As a result in the event when RBE and RDB both want to access same signal simultaneously, some coordination is necessary between these two systems. This coordination is required to guarantee that the chronologically latest report from these two system carries the most current information.

This is accomplished by a mutual exclusion strategy. Each system is assigned a dedicated "DB free/in use" flag. When this flag is reset it indicates to other that owner of this flag is either requesting the use of the DB or is currently using the DB. When one of these two systems is ready to access the RTU DB it first resets own flag indicating to other that it wants to access the DB. Now it checks other's flag to see if the DB is free (flag is set). If the DB is NOT free (flag is reset) then it waits for other to free up the DB. If it is free or when other has set the flag it is now free to access the DB. When the user has finished the DB access it sets own flag informing other that the DB is now available for access.

Only the signal accesses are covered using this lockout strategy. Control for other RDB accesses such as memory, data arrays is not required as such data are not accessed by RBE.

RBE AND THE ALARM SYSTEM

The RBE System is independent of the Alarm System. It is possible to define an analog signal as an alarm signal as well as an RBE signal. This may be desirable, for example, when a signal needs to be observed under two different deadband limits. It is strongly recommended no to use this facility. It should be noted that there is a danger of receiving a latest value for a signal via an alarm message followed by receiving an old value in an ERM. This can happen as alarm messages have higher priority and may pass the ERMs during message communication phase. In addition, a logical signal definitely should not be defined both as an alarm signal and as an RBE signal. Doing so will result in duplicate reports, one by the alarm system and one by RBE system.

Alarm reports have higher priority in that the BSAP transmits any pending alarm reports before it sends any other message types. As a result it is recommended to define the critical signals as alarm signals so that they are reported soon after they enter the alarm state. Some not so critical signals that can now be changed from alarm to RBE type there by relieving the load on Alarm system.

REDUNDANCY AND RBE

The RBE system's support for the redundancy is very simple. The RBE DB in the backup processor is updated for the signal's active/inactive status only. All the other information, such as RBE parameters, current RSN, RSN of the last report ACKed, etc. are not updated to the backup processor. Thus the RBE signals maintain their present active/inactive state thru switchovers. All RBE signals are set to active following a cold start. From then on the active/inactive state is controlled by the RBE Manager.

Following a switchover RBE uses the current active/inactive state of the signal from the RBE DB to determine the number of active signals. Selection at the MODE terminal, even if it is set to Go Active, is ignored and the RBE task sends the WAITING_INIT message with the Sub Function Code (=02) to indicate that a switchover has occurred and it is waiting for the INIT_REQ. Subsequent initialization procedure is same as the cold start initialization. At the end of successful initialization RBE considers active signal counts to decide if it should enter Active State or Inactive State.

RBE does not update the backup processor for value/status changes to the RBE signal. Therefore, the applications that use these signals are responsible to see that the backup processor is updated for such signals. See ACCOL reference manual for recommendations on the redundancy. Generally the backup update frequency of such tasks should be set to 1 to assure that important signals are kept up to date in the backup processor every they are changed in the active processor.

EXTERNAL INTERFACE REQUIREMENTS

RBE Manager or equivalent function at central is responsible for implementing the RBE Manager functions. In order for RBE to be effective the RBE Manager must support all or applicable functions described in this document. A summary of these functions is presented below:

- Initialize the RBE Task when the WAITING_INIT request is received.
- Process received ERMs.
- Handle multiple Report Messages that may result from the INIT_REQ, DEMAND_REQ and ACTDACT_REQ.
- Generate REPORT_ACK messages as needed.
- Support, as needed, these RBE interface messages: demand request, reinitialization request, status request, activate/deactivate signal(s) request. Handle corresponding response messages.

- Handle communication failures and recover any lost messages by demanding such messages: demand request, initialization request, or activate all request.

- If and when desired control throughput by setting/resetting RBE System configuration parameters: SCANRATE, SCANSLICE, STOPXMIT, and FORMAT.

- Incorporate RBE system performance statistics as applicable.

- Include sanity and synchronization detection controls for each RBE node.

SELECTED MESSAGE INTERFACE SCENARIOS

This section describes frequently used message interface scenarios between the RBE Task and the RBE Manager. Only the RBE messages are presented. BSAP protocol messages, e.g. POLL, ACK, are not included here. For this exercise the Toolkit was used as the RBE Manager. Request messages were generated using the "READ WRITE VIA COMM SYSTEM" facility of the Toolkit.

A Data Line monitor was used on an Asynchronous line to capture the messages presented below. The message contents are in hexadecimal.

Note: In the following examples the BSAP Serial numbers may not be in ascending order from message to message because of the time lapse between message transactions.

All messages are presented in following format:

```
BSAP Link, Network Control, and Transport level
     information
RBE Message Data
BSAP Link level information
```

The source of the message from an RBE node is always the RBE Task (ID = 0A2H). Destination for these messages depends on message/response type and could be the RBE Manager (ID = 0A3H) or the requester (the Toolkit, ID = 03H in the following examples).

Cold Start Initialization with Init Report Messages

An RBE node is down loaded and performs initialization. The RBE Task generates WAITING INIT message. The RBE Manager (Toolkit in this case) sends the status request (which by the way is not required but illustrates the point that RBE Manager may send the STATUS_REQ at any time). The RBE Task responds with the WAITING_INIT message. The RBE Manager sends the INIT_REQ and asks for Init Reports (SFC=02). The RBE Task sends the init completion message (STATUS_RESP) and follows it with four Init

Report Messages that include all RBE signals and waits for the REPORT_ACK message for each Init Report Message. The RBE Manager sends the REPORT_ACKs for each Init Report Messages.

```
RBE Manager                                   RBE Task

<-------------------WAITING_INIT<-----------------
   00 10 02 80 C9 00 00 00 04 00 A3 00 00 A2 00
            82 01 E5 34
   10 03 72 E2
     Message is sent to RBE Manager (ID=0A3)


------------------->STATUS_REQ------------------->

 (This message is NOT "required"; here for
  demonstration purpose only.)
   00 10 02 81 F6 00 04 00 00 00 A2 E9 3F 03 01
            06 01
   10 03 87 1A 00

<-------------------WAITING_INIT<-----------------
   00 10 02 80 F7 00 00 00 04 00 A3 08 00 A2 00
            82 01 E5 34
   10 03 A6 1B
     Message is sent to RBE Manager (ID=0A3) even though the
STATUS_REQ was from Toolkit (ID=03)

------------------>INIT_REQ (REPORTS)------------>
   00 10 02 81 F6 00 04 00 00 00 A2 F0 0B 03 01
            03 02 64 00 01 01 00 00 01
   10 03 BE B5 00
                        Source (Requester) = 03
                        SFC = 02 (Generate Init Report Messages)
                        SCANRATE = 64H/100 (10 seconds)
                        SCANSLICE = 01 (no slicing)
                        FORMAT = 01 (short format)
                        STOPXMIT = 00 (no REPORT_ACKs for ERMs)
                        TIMEOUT = 0100H (25.6 seconds)

<-----------------STATUS_RESP<-------------------
 (and Init Report Messages in response to the INIT REQ)
   00 10 02 80 F7 00 00 00 04 40 03 F0 0B A2 01
            83 02 E5 34 01 64 00 01 01  00 2C 01 00 00 00 64 00 19
00
            64 00 19 00 00 00 00 00
   10 03 C6 F4
                        Destination (requester) = 03
                        FC = 83 (Init completion STATUS_RESP)
                        SFC = 02 (Init complete, reports follows)
                        VERSION = 34E5H
                        STATUS = 01 (cold initialization)
                        SCANRATE = 0064H/100 (10 seconds)
                        SCANSLICE = 01 (no slicing)
                        FORMAT = 01 (short format)
```

```
                        STOPXMIT = 00 (no REPORT_ACKs for ERMs)
                        TIMEOUT = 012CH (30 seconds)
                        SEQ_NUM_1 = 00 (current RSN)
                        SEQ_NUM_2 = 00 (RSN of the acknowledged RM)
                        MODE    = 00 (wait for initialization)
                        TOTAL_1 = 0064H (100 analog RBE signals)
                        TOTAL_2 = 0019H (25 logical RBE signals)
                        ACTIVE_1   =   0064H   (100   active   analog
signals)
                        ACTIVE_2   =   0019H   (25   active   logical
signals)
                        TOTAL_3   =   0000H   (#   of   analog   EREs
generated)
                        TOTAL_4 = 0000H (# of logical EREs " )


<-----------------REPORT MESSAGE # 1<------------
   00 10 02 80 F9 00 00 00 04 40 03 F0 0B A2 01
            85 03 00 04 E5 34 16 D3 30  B7 01 20 81 F8 0E 02 00 00
C3
            43 01 0F 02 00 00 C3 43 ........ 0F 10 10 02 00 00 C3
43
   10 03 19 82
                        Destination = 03 (requester)
                        SFC=03 (Init Report Message)
                        FLAG = 81 (more reports to follow, short
                     FMT)
                        RSN = 01 (first Init Report Message)

----------------->REPORT_ACK--------------------->
   00 10 02 81 FC 00 04 00 00 00 A2 F4 0B 03 01
            08 01 01
   10 03 7B C6 00
                        REPORT_ACK is required. It  uses the same
RSN as the received Report Message (RSN=01)

<-----------------REPORT MESSAGE # 2<------------
   00 10 02 80 FD 00 00 00 04 40 03 F4 0B A2 01
            85 03 00 04 E5 34 16 16 D8  30 3A 02 20 81 18 10 10 02
00
            40   0D   44   21   10   10      02   00   04   0D   44
.........................
            2F 11 02 00 00 0D 44
   10 03 D7 57
                        Destination = 03 (requester)
                        SFC=03 (Init Report Message)
                        FLAG = 81 (more reports to follow, short
                           FMT)
                        RSN = 02 (second Init Report Message)

----------------->REPORT_ACK--------------------->
   00 10 02 81 01 00 04 00 00 00 A2 F7 0B 03 01
            08 01 02
   10 03 D8 1C 00
                           REPORT_ACK is required. RSN = 02.
```

```
<-----------------REPORT MESSAGE # 3<------------
   00 10 02 80 02 00 00 00 04 40 03 F7 0B A2 01
            85 03 00 04 E5 34 16 16 DA  30 83 03 20 81 38 11 02 00
C0
            24    44    41    11    02        00    C0    24    44
..............................
            4F 12 02 00 C0 24 44
   10 03 92 A4
                        Destination = 03 (requester)
                        SFC=03 (Init Report Message)
                        FLAG = 81 (more reports to follow, short
                     FMT)
                        RSN = 03 (third Init Report Message)

----------------->REPORT_ACK--------------------->
   00 10 02 81 04 00 04 00 00 00 A2 F9 0B 03 01
            08 01 03
   10 03 9E 55 00
                             REPORT_ACK is required. RSN = 03.


<-----------------REPORT MESSAGE # 4<------------
   00 10 02 80 05 00 00 00 04 40 03 F9 0B A2 01
            85 03 00 04 E5 34 16 16 DC  30 A0 04 1D 01 58 12 02 00
40
            3A    44    61    12    02        00    40    3A    44
..............................
            96 0C 00 9F 0C 00
   10 03 48 57
                        Destination = 03 (requester)
                        SFC=03 (Init Report Message)
                        FLAG = 01 (last report, short FMT)
                        RSN = 04 (fourth Init Report Message)

----------------->REPORT_ACK--------------------->
   00 10 02 81 09 00 04 00 00 00 A2 FD 0B 03 01
            08 01 04
   10 03 A9 3F 00
                        REPORT_ACK is required. RSN = 04.
```

The RBE Task now enters the Active State where it will start generating the ERMs periodically. Next ERM will have RSN = 05.

Cold start Initialization without Init Report Messages

This initialization is same as Cold Start Initialization with Init Report Messages except the Init Report Messages are not requested/generated. The STATUS_REQ was not used in this transaction. After the initialization complete (STATUS_RESP) is generated RBE enters Active State where it will start generating the ERMs periodically.

```
RBE Manager                                    RBE Task

<-----------------WAITING_INIT<------------------
   00 10 02 80 D3 00 00 00 04 00 A3 01 00 A2 00
            82 01 E5 34
   10 03 F2 4A
                        Message is sent to RBE Manager (ID=0A3)

----------------->INIT_REQ(REPORTS-------------->
   00 10 02 81 D7 00 04 00 00 00 A2 D0 41 03 01
            03 01 64 00 01 01 00 00 01
   10 03 FA AE 00
                        SFC = 01      (do   not    generate   Init
Report Messages) All other  parameters  are  same  as previous
transaction.

<------------STATUS_RESP(Init Complete)<--------
   00 10 02 80 D8 00 00 00 04 40 03 D0 41 A2 01
            83 04 E5 34 01 64 00 01 01  00 2C 01 00 00 00 64 00 19
00
            64 00 19 00 00 00 00 00
   10 D6 76 F4


                        Destination = 03 (requester)
                        All other parameters same as earlier.
```

Init Report Messages are NOT generated. The RBE Task now enters the Active State where it will start generating the ERMs periodically. Next ERM will have RSN=01.

RBE Complete Reinitialization

The RBE Manager may want to 'reinitialize' RBE at any time with different parameters and/or retrieve the current values/status' for all RBE signals. The RBE Manager sends the INIT REQ message with the appropriate Sub Function Code (generate or do not generate init reports). The INIT REQs are same as the cold start initialization described earlier except that the RBE Task is already initialized and active.

RBE Parameter Re/Initialization

The RBE Manager may decide to reinitialize the RBE system parameters, e.g. SCANSLICE, SCANTIME. It sends the INIT_REQ with appropriate subfunction code to the RBE Task. The RBE Task does not reinitialize the RBE DB. It cancels the current activity and changes only the parameters as requested and begins normal operations. The value of the current RSN is not changed. For the following example the current RSN was 34H when the INIT_REQ is sent.

```
RBE Manager                                    RBE Task

----------------->INIT_REQ(REPORTS)------------->
```

```
   00 10 02 81 2E 00 04 00 00 00 A2 72 24 03 00
            03 03 00 01 01 01 00 FF FF
   10 03 46 2D 00
                        Source = 03 (requester = Toolkit)
                        SFC = 03 (init parameters only)
                        SCANRATE = 0100H (25.6 seconds)
                        SCANSLICE = 01 (no slicing)
                        FORMAT = 01 (short format)
                        STOPXMIT = 00 (no REPORT_ACKs required)
                        TIMEOUT = 0FFFFH (6553.5 seconds)

<---------STATUS_RESP(Init Complete)<-----------
   00 10 02 80 2F 00 00 00 04 40 03 72 24 A2 01
            83 04 E5 34 03 00 01 01 01  00 FF FF 34 04 00 64 00 19
00
            64 00 19 00 14 1E B7 01
   10 03 F4 CB
                        Destination = 03 (Toolkit)
                        SFC = 04 (Init completed, only parameters)
                        Other parameters as explained earlier.
```

Init Report Messages are not generated. RSN is not reset to 0. The RBE Task now enters the Active State where it will start generating the ERMs periodically.

```
<--------------REPORT MESSAGE # 1<------------
   00 10 02 80 34 00 00 00 04 00 A3 33 01 A2 00
            85 00 00 04 E5 34 0F 16 09  26 A0 35 20 01 F8 0E 02 00
88
            BA 46 0A 0F 02 00 88 BA 46 ..... 0F 10 10 02 00 88 BA
46
   10 03 F0 F7
                        Destination = A3H (RBE Manager)
                        SFC = 00 (ERM)
                        FLAG = 01 (short format)
                        Num of EREs = 20H
                        RSN = 35H (RSN is not changed)

.........(Active State continues)
```

Download, Initialize RBE, and Go Active

The MODE terminal of the RBE module is set to Go_Active (=01). Following an RBE node initialization (after a down load only) the RBE Task initializes the RBE DB, generates the STATUS_RESP (Going Active), and enters the Active State using the parameters specified by the RBE Module terminals or defaults. There is no need for the INIT_REQ and the WAITING_INIT message is not sent to the RBE Manager.

```
RBE Manager                                    RBE Task

<---------STATUS_RESP(Init Complete)<------------
```

```
   00 10 02 80 15 00 00 00 04 00 A3 01 00 A2 00
            83 03 54 80 01 0A 00 01 01  00 A0 8C 00 00 01 09 00 0A
00
            09 00 0A 00 00 00 00 00
   10 03 3C 5D
                        Destination = A3 (RBE Manager)
                        SFC = 03 (init completed, Going Active)
                        VERSION = 8054H
                        STATUS = 01 (cold initialization)

                        Following   parameters   are   from   RBE
Module terminal or defaults:

                        SCANRATE = 000AH (1 second)
                        SCANSLICE = 01 (no slicing)
                        FORMAT = 01 (short format)
                        STOPXMIT = 00 (no REPORT_ACKs required)
                        TIMEOUT = 8CA0H (3600 seconds/1 hour)
                        SEQ_NUM_1 = 00  (RSN   =   00;   no
                  reports generated yet)
                        SEQ_NUM_2 = 00  (ACKed RSN = 00)
                        Other parameters as explained earlier.


            Init Report Messages are not  generated. The RBE Task
now enters the Active  State  where  it will start generating
the ERMs periodically.

<---------------REPORT MESSAGE # 1<--------------
   00 10 02 80 17 00 00 00 04 00 A3 01 00 A2 00
            85 00 00 04 54 80 19 16 1D  1F AA 01 03 01 A5 15 02 00
00
            C8 41 3B 1E 02 00 00 C8 41 51 1E 02 00 00 C8 41
   10 03 9D 4F
                        Destination = A3H (RBE Manager)
                        SFC = 00 (ERM)
                        RSN = 01 (the first ERM after
           initialization)

.........(Active State continues)
```

Demand Request

The RBE Manager may generate a demand request to recover multiple Exception Report Messages starting from a given RSN. If the demand request is valid then it directly results in Demand Report Messages. Like all other requests there is no specific 'demand response' for this request. Following scenario describes this interface.

The RBE Manager sends a demand request for all Exception Report Elements since the ERM with RSN = 03. The RBE Task

generates demand reports that include all EREs since RSN = 03 to the RSN = 08 (current RSN). It waits for a REPORT_ACK for each Demand Report Message. After the last Demand Report and corresponding REPORT_ACK it enters the Active State.

```
RBE Manager                                   RBE Task

----------------->DEMAND_REQ(RSN=03)-------------->
   00 10 02 81 69 00 04 00 00 00 A2 0E 27 03 00
            05 01 03
   10 03 53 3F 00
                        Requester = 03 (Toolkit)
                        RSN = 03 (requesting all EREs since RSN 03)

<--------- Demand Report Message # 1<-------------
   00 10 02 80 6A 00 00 00 04 40 03 0E 27 A2 01
            85 05 00 04 E5 34 0F 16 BF  29 51 06 20 81 F8 0E 02 00
09
            70 47 ........................... 0F 10 10 02 00 09 70
47
   10 03 BA 9B
                        Destination = 03 (Toolkit)
                        SFC = 05 (Demand Report Message)
                        RSN = 06
                        FLAG = 81 (more reports to follow, short
                     FMT)
                        Num of Demand Entries = 20H
```

Demand Report Messages (SFC=05) are sent to the requester (=03). First Demand Report Message has RSN = 06. Continuation flag (8X) indicates more reports to follow.

```
----------------->REPORT_ACK(RSN=06)-------------->
   00 10 02 81 6E 00 04 00 00 00 A2 12 27 03 00
            08 01 06
   10 03 89 5A 00
                        Required REPORT_ACK (RSN=06).

<--------- Demand Report Message # 2<-------------
   00 10 02 80 6F 00 00 00 04 40 03 12 27 A2 01
            85 05 00 04 E5 34 0F 16 C3  29 8C 07 20 81 18 10 10 02
00
            09 70 47 ........................... 2F 11 02 00 09 70
47
   10 03 92 67
                        Destination = 03 (Toolkit)
                        SFC = 05 (Demand Report Message)
                        RSN = 07H
                        FLAG = 81 (more reports to follow, short
                     FMT)
                        Num of Demand Entries = 20H

----------------->REPORT_ACK(RSN=07)-------------->
   00 10 02 81 72 00 04 00 00 00 A2 15 27 03 00
            08 01 07
   10 03 40 04 00
```

```
                        Required REPORT_ACK (RSN=07).

<-------- Demand Report Message # 3<--------------
   00 10 02 80 73 00 00 00 04 40 03 15 27 A2 01
            85 05 00 04 E5 34 0F 16 C5  29 B2 08 20 81 38 11 02 00
0F
            70 47 ............................. 4F 12 02 00 0F 70
47
   10 03 87 C5
                        Destination = 03 (Toolkit)
                        SFC = 05 (Demand Report Message)
                        RSN = 08
                        FLAG = 81 (more reports to follow, short
                      FMT)
                        Num of Demand Entries = 20H

------------------>REPORT_ACK(RSN=08)-------------->
   00 10 02 81 75 00 04 00 00 00 A2 17 27 03 00
            08 01 08
   10 03 20 7C 00
                        Required REPORT_ACK (RSN=08).

<------- Demand Report Message # 4<---------------
   00 10 02 80 76 00 00 00 04 40 03 17 27 A2 01
            85 05 00 04 E5 34 0F 16 C7  29 B7 08 1C 01 58 12 02 00
60
            70 47 .........................8D 0C 80 96 0C 80 9F 0C
80
   10 03 90 65
                        Destination = 03 (Toolkit)
                        SFC = 05 (Demand Report Message)
                        RSN = 08


                        FLAG = 01 (Last report, short format)
                        Num of EREs = 1CH

------------------>REPORT_ACK(RSN=08)-------------->
   00 10 02 81 78 00 04 00 00 00 A2 19 27 03 00
            08 01 08
   10 03 CC 7D 00
                        Required REPORT_ACK (RSN=08).
```

The RBE Task now resumes Active State and sends normal reports during future scan sessions. RSN will start at 09:

```
<-------Exception Report Message # n<-----------
   00 10 02 80 79 00 00 00 04 00 A3 6F 02 A2 00
           85 00 00 04 E5 34 0F 16 C9  29 51 09 20 01 F8 0E 02 00
34
           73 47 ........................... 0F 10 10 02 00 34 73
47
   10 03 94 15
                        Destination = A3H (RBE Manager)
                        SFC = 00 (Exception Report Message)
                        RSN = 09     (next RSN  after  the  RSN of
                                     the   last   Demand   Report
                                     Message)
                        FLAG = 01 (short format)
                        Num of EREs = 20H

<-------Exception Report Message  n+1<-----------
   00 10 02 80 7B 00 00 00 04 00 A3 70 02 A2 00
           85 00 00 04 E5 34 0F 16 C9  29 53 0A 20 01 18 10 10 02
00
           34 73 47 ........................... 2F 11 02 00 34 73
47
   10 03 30 64
                        Destination = A3H (RBE Manager)
                        SFC = 00 (Exception Report Message)
                        RSN = 0AH    (next RSN  after  the  RSN of
                                     the   last   Exception   Report
                                     Message)
                        FLAG = 01 (short format)
                        Num of EREs = 20H

......... (Active State continues)
```

Status Request/Response

The RBE Manager may send the STATUS_REQ at any time. In reply the RBE Task returns the current parameters, statistics, and status information. In following example the STATUS_REQ is sent when the RBE Task is waiting for the REPORT_ACK after it has sent the Active Report Message for the ACTDACT_REQ:

```
RBE Manager                                   RBE Task

------------------->STATUS_REQ------------------->


   00 10 02 81 EE 00 04 00 00 00 A2 A0 07 03 00
           06 01
   10 03 3F 5F 00
                        Requester = 03 (Toolkit)

<-------------------STATUS_RESP<-------------------
   00 10 02 80 EF 00 00 00 04 40 03 A0 07 A2 01
           86 05 E5 34 07 00 02 01 01  00 2C 02 3C 27 00 64 00 19
00
```

```
          62 00 17 00 66 05 4A 00
   10 03 8B D4
                        Destination = 03 (Toolkit)
                        SFC    = 05 (in response to STATUS_REQ)
                        STATUS = 07  (processing  ACTDACT_REQ, i.e.
                                     waiting  for  the   REPORT_ACK
                                     for a Active Report Message)
                        RSN    = 3CH (RSN for which it is waiting)
                        Other parameters as explained earlier.

........ (current state continues)
```

Activate/Deactivate RBE Signal(s)

Occasionally it may be necessary to deactivate and at a later time reactivate selected signals. As described by the following scenario the ACTDACT_REQ can be used to accomplish this task. The RBE Manager can use the activate command to reinitialize the RBE DB with the current value and at the same time collect these values via the Active Report Message for all RBE signals, all analog RBE signals, all logical RBE signals, or selected RBE signals.

```
RBE Manager                                   RBE Task

--------------------->ACTDACT_REQ----------------->
          (Selected signals; four entries)
   00 10 02 81 05 00 04 00 00 00 A2 CF 4A 03 00
          07 04 08 02 1C 0F 02 E2 0B  01 25 0F 01 EB 0B 02 2E 0F
02
          F4 0B 01 37 0F 01 FD 0B
   10 03 5C 13 00
                        Requester = 03 (Toolkit)
                        SFC = 04 (selected signals)
                        Num of entries = 08
                        Activate = 01       Deactivate = 02
                            MSDs                MSDs
                            0F25                0F1C
                            0BEB                 0BE2
                            0F37                0F2E
                            0BFD           0BF4

<--------------------ACTDACT_RESP<-----------------
   00 10 02 80 06 00 00 00 04 40 03 CF 4A A2 01


          87 04 00
   10 03 12 8C
                        Destination = 03 (Toolkit)
                        STATUS =    00    (no                errors;
                                          signals       de/activated
                                          successfully;       Active
                                          Report          Message(s)
                                          follows)
```

```
<----------Activate Report Message<-------------
   00 10 02 80 08 00 00 00 04 40 03 CF 4A A2 01
           85 07 00 04 E5 34 10 10 16  52 17 A2 15 04 01 25 0F 02
30
           FA 33 49 EB 0B 80 37 0F 02 30 FA 33 49 FD 0B 80
   10 03 6F 87
                        Destination = 03 (Toolkit)
                        SFC = 07 (Active Report Message)
                        FLAG = 01 (Last report, short format)
                        RSN = 15H
                        Num of EREs = 04
                        MSD     FLAG    Data
                        0F25H   02      49,33,FA,30 (analog signal)
                        0BEBH   80       (logical    signal;    data
                                       in FLAG)
                        0F37H   02      49,33,FA,30 (analog signal)
                        0BFDH   80       (logical    signal;    data
                                       in FLAG)

------------------->REPORT_ACK----------------->
   00 10 02 81 33 00 04 00 00 00 A2 EB 4A 03 00
           08 01 15
   10 03 A5 88 00
                        REPORT_ACK  is  required.  It  must  have  the
same RSN as the received Active Report Message ( = 15H).

........ (RBE Task continues in active state).
```

Waiting for an Acknowledgement

The RBE Task waits for a REPORT_ACK message in the following four conditions:

a. After every Init Report Message.
b. After every Demand Report Message.
c. After every Active Report Message.
d. After the RBE Task has sent "n" ERMs since the last ACKed ERM. Where n = STOPXMIT limit.

In the first three conditions the REPORT_ACK must specify the same RSN as received in the Report Message. In the fourth condition the REPORT ACK can specify any valid RSN that is either equal to the current RSN or between the RSN of the last REPORT_ACK message and the current RSN.

In the following example the STOPXMIT limit is set to 20.

```
RBE Manager                                      RBE Task

--------------------->REPORT ACK------------------->
   00 10 02 81 03 00 04 00 00 00 A2 71 27 03 38
           08 01 78
   10 03 C7 00 00
```

The REPORT_ACK acknowledges all Report Messages up to RSN = 78H/120. REPORT_ACK does not have any response.

```
<-------------Exception Report Message<--------
   00 10 02 80 04 00 00 00 04 00 A3 7C 00 A2 00
            85 00 00 04 E5 34 19 16 2E 23 52 79 20 01 .........
```

Normal report (SFC = 00) continues. RSN is 79H/121. The normal reports are sent to the RBE Manager only (ID = 0A3H).

```
.............
.............
.............

<-------------ERM<--------
   00 10 02 80 2A 00 00 00 04 00 A3 8F 00 A2 00
            85 00 00 04 E5 34 19 16 32 23 B7 0C 04 01 .........
```

Exception report (SFC = 00) continues. RSN is 0CH/12. This makes it 20th ERM since last REPORT_ACK (RSN = 122-127,00-12). The RBE task enters a wait as STOPXMIT (=20) limit is reached.

```
-------------------->STATUS_REQ--------------->
   00 10 02 81 A5 00 04 00 00 00 A2 FF 27 03 38
            06 01
   10 03 7C 37 00
```

The request is from Toolkit (=03).

```
<--------------------STATUS_RESP<-----------------
   00 10 02 80 A6 00 00 00 04 40 03 FF 27 A2 01
            86 05 E5 34 06 0A 00 01 01  14 2C 01 0C 78 00 64 00 19
00
            64 00 19 00 AC 0D AC 00
   10 03 CE 0C
```

Response for the STATUS_REQ (SFC = 05) is sent to the requester (=03). RBE Task is in Actively Scanning state (=06) where it is waiting for the REPORT_ACK for the reports up to RSN = 0C.

Any RSN from 079H-07FH,00H-0CH is a valid RSN at this point.

```
------------------>REPORT_ACK (RSN=7FH)--------->
   00 10 02 81 B6 00 04 00 00 00 A2 0F 28 03 38
            08 01 7F
   10 03 94 5C 00
```

ERMs up to the RSN=7FH are acknowledged. This leaves 13 more messages that are not acknowledged (RSN = 00-0CH). The RBE Task resumes scan process.

```
<-------------Exception Report Message<--------
```

```
   00 10 02 80 B7 00 00 00 04 00 A3 90 00 A2 00
           85 00 00 04 E5 34 19 16 B6 23 52 0D 20 01 .........
```

The normal reports are generated until the report with RSN = 19/013H is transmitted when the RBE Task waits for the REPORT_ACK message. At this point 20 ERMs are pending acknowledgement (ERMs with RSN = 00 to 19). Of course the RBE Manager could send a REPORT_ACK before the RBE Task gets to this limit.

.....(actively scanning state continues)

Requests with Error

When an erroneous message is received the RBE Task generates an ERROR_RESP message to the RBE Manager. It contains the error code and a copy of the received message. Following this error response the RBE Task continues the current state.

```
RBE Manager                                    RBE Task

----------------------->INIT_REQ---------------->
   00 10 02 81 98 00 04 00 00 00 A2 2A 08 03 00
           33 01 2C 01 01 01 00 2C 01
   10 03 CB 5B 00
```

The request with a bad function code (=33) is received from the Toolkit.

```
<-------------------ERROR_RESP<------------------
   00 10 02 80 99 00 00 00 04 40 03 2A 08 A2 01
           FF F6 33 01 2C 01 01 01 00 2C 01
   10 03 7C F1
```

Error Response (FC = 0FFH) is sent to the requester (=03) indicating the bad function code error (SFC = 0F6H/-10).

```
--------->ACTDACT_REQ (Bad Sub Function Code)------>


   00 10 02 81 D0 00 04 00 00 00 A2 4E 08 03 00
           07 05 02 1C 0F 02 E2 0B
   10 03 3D F2 00
```

The request with a bad sub function code (=05) is received from the Toolkit.

```
<-------------------ERROR_RESP<------------------
   00 10 02 80 D1 00 00 00 04 40 03 4E 08 A2 01
           FF F5 07 05 02 1C 0F 02 E2 0B
   10 03 C2 43
```

Error Response (FC = 0FFH) is sent to the requester (=03) indicating the bad sub function code error (SFC = 0F5H/-11).

FUNCTION, STATUS, AND ERROR CODES

FUNCTION AND SUB FUNCTION CODES

REQUESTS:

FC	SFC	Usage
03	01	INIT_REQ, Full initialization, No Reports.
03	02	INIT_REQ, Full Initialization, Generate Init Report Messages.
03	03	INIT_REQ, reinitialize parameters only.
05	01	DEMAND_REQ, generate Demand Report Messages.
06	01	STATUS_REQ, generate status report.
07	01	ACTDACT_REQ, activate/deactivate all signals.
07	02	ACTDACT_REQ, activate/deactivate all analog signals.
07	03	ACTDACT_REQ, activate/deactivate all logical signals.
07	04	ACTDACT_REQ, activate/deactivate selected signal(s).
08	01	REPORT_ACK, acknowledgement for all Report Messages up to the specified RSN.

RESPONSES:

FC	SFC	Usage
82	01	WAITING_INIT, RBE Task waiting for INIT_REQ following "cold start".
82	02	WAITING_INIT, RBE Task waiting for INIT_REQ following a "switchover".
82	03	WAITING_INIT, RBE Task waiting for INIT_REQ after reinitialization failed due to error in the previous INIT_REQ.
83	01	STATUS_RESP, indicates initialization completed, Report Messages are not generated (result of INIT_REQ[No Reports]).
83	02	STATUS_RESP, indicates initialization completed, Init Report Messages follows (result of INIT_REQ[Generate Reports]).
83	03	STATUS_RESP, indicates initialization completed, Going Active (result of RBE Module's MODE terminal set to Go_Active), Init Report Messages are not generated.
83	04	STATUS_RESP, indicates parameter re-initialization completed, Init Report Messages are not generated (result of INIT_REQ[Parameters Only]).

85	00	ERM, result of scan session.
85	03	Init Report Message, result of INIT_REQ.
85	05	Demand Report Message, result of DEMAND_REQ.
86	05	STATUS_RESP, result of the STATUS_REQ.
87	01	ACTDACT_RESP, result of ACTDACT_REQ.
FF	XX	ERROR_RESP, result of error in received request message. XX is a negative one byte number in hex. See Error Codes in the next section (note: the error codes are decimal values)..

STATUS AND ERROR CODES

Some of the error /status codes may be visible for a very short period as the same signal is used to reflect the current processing state.

STATUS CAUSE & REMEDY

01 Init State: RBE Task is in the initialization state following a cold start. It is waiting for the INIT_REQ message: Send a valid INIT_REQ message.

-OR-

Currently processing the received INIT_REQ message and may be waiting for the REPORT_ACK for an Init Report Message.

02 Init State: Same as above except the state is following a switchover.

03 Init State: RBE Task had received an INIT_REQ and started the reinitialization. However, there was an error in the received request and now it is waiting for a proper INIT_REQ message. Or it is currently processing the received INIT_REQ message and may be waiting for the REPORT_ACK for an Init Report Message.

Send a valid INIT_REQ message or the required REPORT_ACK message.

04 Inactive State: All RBE signals are in inactive state. Send either a valid ACTDACT_REQ and activate one or more signals or reload the RTU.

05 Active Idle State: RBE Task is Active and waiting for the SCANRATE timeout to occur.

Nothing to do. Good healthy state. If reports are not being generated and are concerned then monitor

following RBE module terminals and as long as one of them is changing within a SCANRATE period then the RBE Task is performing its function:

STATUS - Toggles between 05 and 06.
SCANTIME - Value changes.
TOTAL_3 - Value increases.
TOTAL_4 - Value increases.

06 Actively Scanning State: The RBE Task is at present in scan session. The scan session includes RBE DB scan and analysis, report preparation and transmission, waiting for a REPORT_ACK, or waiting between two slices.

07 Processing Activate / Deactivate request. This includes waiting for the REPORT_ACK for the Active Report Message.

08 Processing Demand Request. This includes waiting for the REPORT_ACK for the Demand Report Message.

Following error codes are displayed at the RBE Module status terminal and may be overwritten by the next state described above.

ERROR CAUSE & REMEDY

-1 RBE is not defined. Bad RBE structures. The MCB for the RBE Module is not found or the total of analog and logical signal is 0. This indicates error with ACCOL tools.

-2 Can not allocate dedicated buffers. Problem with the ACCOL load or the firmware.

-3 Can not queue I/O requests to the system. Problem with the ACCOL load or the firmware.

-9 Bad Message. A local message is received. Only global messages are supported. The request is rejected.

Do not send any local messages to the RBE Task.

-10 Bad Function Code. The Function code in the received message is not valid. See above for valid request function codes. The request is rejected and error response message is sent to the requester.

Correct the function code and send the request again.

-11 Bad Sub Function Code. The Sub Function code in the received message is not valid. See above for valid request sub function codes. The request is rejected and error response message is sent to the requester.

Correct the sub function code and send the request again.

-12 Bad Scan Period. The Scan Period Value in the INIT_REQ is 0. The request is rejected, an error response message is sent to the requester, an WINITIN_INIT message is sent to the RBE Manager, and remains in the reinitialization state waiting for the INIT_REQ.

Set the Scan Period to a valid value between 1 and 65535 and send the request again.

-13 Bad STOPXMIT limit. The STOPXMIT limit value in the INIT_REQ is > 127. The request is rejected, an error response message is sent to the requester, an WINITIN_INIT message is sent to the RBE Manager, and remains in the reinitialization state waiting for the INIT_REQ.

Set the limit from 00-127 and send the request again.

-14 Bad Report Format Type. The Format Type in the INIT_REQ is not 01 or 02. The request is rejected, an error response message is sent to the requester, an WINITIN_INIT message is sent to the RBE Manager, and remains in the reinitialization state waiting for the INIT_REQ.

Set the type to 01 (Short Format) or 02 (Long Format) and send the request again.

-15 Bad parameter (Number of Entries) in ACTDACT_REQ. It is set to 0 or is greater than 65. The request is rejected and an error response message is sent to the requester.

Set the parameter to actual number and send the request again.

-16 Bad parameter. OP Code in ACTDACT_REQ is not 01 or 02. The request is rejected and an error response message is sent to the requester.

Set the parameter to actual Op Code (01 = Activate / 02 = Deactivate) and send the request again.

-17 Bad entry in the ACTDACT_REQ. Either the Op Code or the MSD address of an entry in this request is invalid. The erroneous entry is returned in the ACTDACT_RESP. All other entries are processed normally.

Correct the error and retry. All other entries were handled successfully so they do not have to be specified again.

-18 Bad RSN in DEMAND_REQ. The parameter is set to > 127. The request is rejected and an error response message is sent to the requester.

Set the parameter from 00-127 and send the request again.

-19 Bad RSN in REPORT_ACK. The received RSN in the REPORT_ACK message is not equal to the current RSN or it is not between the last RSN ACKed and the current RSN. The request is rejected and an error response message is sent to the requester.

Send the REPORT_ACK with a valid RSN.

-20 After a switchover the RBE Task has detected a mismatch either between the Value of the ACTIVE_1 terminal and the actual number of active analog signals in the RBE DB or between the value of the ACTIVE_2 terminal and the actual number of active logical signals in the RBE DB. This is just a warning The ACTIVE_1 and ACTIVE_2 terminals are set to actual number of active signals found in the RBE DB.

It may be desirable to activate all signals if the active counts are smaller then the total signal counts (remember this will generate ERE for all signals).

-21 Bad Port. The message is received over the Pseudo Slave port. The RBE does not support the communication over this port.

Use the Slave port and send the request again.

-1XX Internal communication error code. The Comm system has returned a buffer to the RBE Task with error code XX. Generally this is 03 that indicates the comm system could not transmit the requested message successfully. Other values of the XX can be found in the BSAP section of this manual.

The Report Message that could not be transmitted is lost. The RBE Task continues the scan process. The process of report generation slows down as the RBE Task depends on the returned buffer from the Comm subsystem. The Comm subsystem holds onto each buffer until it times out waiting for a Poll at the Slave port.

QUICK REFERENCE

This section lists some commonly used general information for quick lookup:

- RBE Task Function Code for Message exchange is 0A2H.
- RBE Manager Function Code for message exchange is 0A3H.
- Communication is supported only thru the Slave port.
- Only global message communication is supported.
- 0 BSAP Network Global Address for the network central node with the RBE Manager is assumed to be 00.
- All WAITING_INIT messages are sent to the RBE Manager at the network central.
- All ERMs are sent to the RBE Manager at the network central.
- All Init, Demand, and Active Report Messages for INIT_REQ, DEMAND_REQ, and ACTDACT REQ are sent to the requester (source of the request).
- All other response messages are sent to the source of the corresponding request message.
- A Report Message may contain multiple number of EREs for the analog and/or logical signals.
- Size of the short format analog ERE is 7 bytes.
- Size of the short format logical ERE is 3 bytes.
- Size of the long format ERE is size of short format ERE plus name string "basename.extension.attribute" plus null terminator byte.
- The status byte of an ERE includes a flag that indicates whether the ERE is for an analog signal or for a logical signal.
- Maximum number of data bytes space for ERE in a Report Message is 225.

- Maximum number of EREs in a Report Message is 32 if all of the EREs were analog reports.

- Maximum number of EREs in a Report Message is 75 if all of the EREs were logical reports.

RBE PERFORMANCE AND CPU LOAD REQUIREMENTS

This section presents the performance measurements and defines the CPU time requirements for the RBE and the Template Collection systems.

All times are in percentage of CPU time. These measurements were obtained under following conditions:

- A 3330 DPC running at 12 Mhz.

- Gateway polling at the rate of 20 polls per second.

- ACCOL load with:

  - RBE set to go active following download. No Stop_XMIT Limit. SCANRATE set for 1 second.
  - 1000 Analog signals and 500 Logical signals in the RBE database, number of signals changed varied for different cases.
  - Independent control for number of Analog and number of Logical signals that can be changed and reported.
  - An ACCOL Task activated and collected measurements.
  - #RTTIME.000 set to -100 seconds for each measurement thus each measurement represented percentage of CPU time.
  - Five measurements were obtained for each set of signals. First and last measurement were dropped and an average was taken for the middle three readings. Resulting averages are presented in the following tables.

- Template requests generated once per second (same rate as the RBE scan period).

- Either RBE or the Template Collection function active at any given time.

- Same cases measured under the RBE and Template Collection systems for direct comparison.

- No other activity in the 3330.

- Basic Idle time with the ACCOL load running, RBE and Template Collection systems off, and the active comm system with the Gateway polling is 92.67%.

Even though the DPC may have enough capacity to generate the large number of RBE Report Messages the throughput is also dependent upon the network characteristics, e.g. speed of the communication link and poll rate. For example, all conditions being equal within two DPCs, the DPC connected to 1 Mb RASCAL link and poll rate of 10 polls per seconds will be able to generate more RBE Report Messages then the DPC connected to an 9600 Baud Asynchronous link with a poll rate of 1 poll per second.

Measurement results for various cases are presented in table format along with the corresponding line graphs. One table and it's graph combines results for different cases of the RBE analog signals. Two additional tables and corresponding graphs combine the results of RBE and Template Collection systems for analog and logical signals.

In the final section a formula is presented which is based on the measurements presented below and other measurements. It can be used to to determine the approximate CPU overhead for any number of RBE signals. As stated earlier this CPU time is the total time required for RBE processing. This time represents RBE DB scan and deadband analysis, exception preparation, and report communication, including the communication system overhead. It does not include overhead for any other function in the DPC.

Analog Signals: Deadband Analysis But No Reports

Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed and their deadbands were analyzed for exception condition but the RBE Report Messages were not generated (deadbands were set too high and no exceptions were detected).

Analog Signals: No Deadband Analysis With Reports

Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed. Changes were made to the Control Inhibit flag so deadband analysis was not performed. Each signal in a given set generated an exception report.

Each group of 32 analog signals generated one RBE Report Message, thus results in the following table also correspond to 1 to 10 RBE Report messages respectively.

Analog Signals: Deadband Analysis With Reports
13.3.
Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed. Each

signal's change was analyzed against its deadband for exception detection. The exception reports were generated for each signal.

Each group of 32 analog signals generated one RBE Report Message, thus results in the following table also correspond to 1 to 10 RBE Report messages respectively.

Analog Signals: Combined

Following table combines the results from the previous three cases:
deadband analysis but no report messages, report messages without deadband analysis, and deadband analysis with report messages.

Analog Signals: Template Collection

Following table includes results for the analog signals using the Template Collection system. No other activity was in progress in the DPC during these measurements. To facilitate easy comparison the template requests included the same number of signals that were used in the RBE measurements (even though it is possible to fit 38 analog signals in a single template response compare to 32 signals in the RBE Report Message).

Analog Signals: RBE and Template Collection

Following table includes the RBE results for the deadband analysis with reports and the Template Collection results.

The comparison is slightly biased towards the RBE as number of signals in a template response is 38 compare to the 32 signals in an RBE Report Message.

Analog Signals: Delta = RBE - Template Collection

Following table presents the previous table in different form. It gives the difference between the results of RBE system and Template Collection system. It shows a clear advantage for RBE system when the rate of template collection and the RBE scan periods are the same.

Additional benefit may result if only a subset of all signals is in exception on every scan period. This will result in fewer number of exceptions and may result in reduced number of RBE Report Messages.

Logical Signals: RBE

Following table includes results for 5 sets of logical signals. During these measurements the logical signals were changed. An

exception report was generated for each logical signal. Each group of 75 logical signals generated one RBE Report Message, thus measurements in the following table also correspond to 1 to 5 RBE Report messages respectively.

Logical Signals: Template Collection

Following table includes results for the logical signals using the Template Collection system. No other activity was in progress in the DPC during these measurements. To facilitate easy comparison the template requests included the same number of signals that were used in the RBE measurements (even though only 48 logical signals are included in a single template response compared to 75 signals in the RBE Report Message).

Logical Signals: RBE and Template Collection

Following table combines the RBE and the Template Collection results for the logical signals.

The results may not provide exact comparison as the number of signals that can be returned in the template response is 48 compare to the 75 signals in an RBE Report Message.

Logical Signals: Delta = RBE - Template Collection

Following table presents the previous table in different form. It gives the difference between the results of RBE system and Template Collection system. It shows a clear advantage for RBE system when the rate of template collection and the RBE scan periods are the same.

Additional benefit may result if only a subset of all signals is in exception on every scan period. This will result in fewer number of exceptions and may result in reduced number of RBE Report Messages.

An Equation For RBE Load Requirements

The following equation is provided to calculate the approximate percentage of the CPU time required by the RBE system to process any given number and any given mix of analog and logical exceptions.

a. 0.973% To send the prepared RBE Report Message; including the communication system times.

b. 0.062% To analyze an analog signal and corresponding Deadband for an exception.

c. 0.018% To prepare and log the exception report in the RBE Report Message.

d. 0.0262% To analyze a logical signal and log the resulting exception report in the RBE Report Message.

% of CPU time required by the RBE =

(# of analog signals changed AND deadbands analyzed * 0.062) + (# of analog exceptions * 0.018) + (# of logical exceptions generated * 0.0262) + ((INT(((# of analog exceptions * 7) + (# of logical exceptions * 3) + 225) / 225)) * 0.973)

Example: 150 Analog RBE signals:
40 signals changed that has deadbands but only 35 of these signals require exception reports.
25 signals changed that do not have assigned deadbands.
200 Logical RBE signals:
110 signals changed but only 80 of these signals require exception reports.

% CPU time required by the RBE =

(40*0.062) + ((35+25)*0.018) + (80*0.0262) + (INT((((30+25)*7)+(80*3)+225)/225)*0.973) = 8.575%

- Basic Idle time with the ACCOL load running, RBE and Template Collection systems off, and the active comm system with the Gateway polling is 92.67%.

Even though the DPC may have enough capacity to generate the large number of RBE Report Messages the throughput is also dependent upon the network characteristics, e.g. speed of the communication link and poll rate. For example, all conditions being equal within two DPCs, the DPC connected to 1 Mb RASCAL link and poll rate of 10 polls per seconds will be able to generate more RBE Report Messages then the DPC connected to an 9600 Baud Asynchronous link with a poll rate of 1 poll per second.

Measurement results for various cases are presented in table format along with the corresponding line graphs. One table and it's graph combines results for different cases of the RBE analog signals. Two additional tables and corresponding graphs combine the results of RBE and Template Collection systems for analog and logical signals.

In the final section a formula is presented which is based on the measurements presented below and other measurements. It can be used to to determine the approximate CPU overhead for any number of RBE signals. As stated earlier this CPU time is the total time required for RBE processing. This time represents RBE DB scan and deadband analysis, exception preparation, and report communication, including the communication system overhead. It does not include overhead for any other function in the DPC.

Analog Signals: Deadband Analysis But No Reports

Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed and their deadbands were analyzed for exception condition but the RBE Report Messages were not generated (deadbands were set too high and no exceptions were detected).

Analog Signals: No Deadband Analysis With Reports

Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed. Changes were made to the Control Inhibit flag so deadband analysis was not performed. Each signal in a given set generated an exception report.

Each group of 32 analog signals generated one RBE Report Message, thus results in the following table also correspond to 1 to 10 RBE Report messages respectively.

Analog Signals: Deadband Analysis With Reports

13.3.

Following table includes results for 10 sets of analog signals. During these measurements the analog signals were changed. Each signal's change was analyzed against its deadband for exception detection. The exception reports were generated for each signal.

Each group of 32 analog signals generated one RBE Report Message, thus results in the following table also correspond to 1 to 10 RBE Report messages respectively.

Analog Signals: Combined

Following table combines the results from the previous three cases:
deadband analysis but no report messages, report messages without deadband analysis, and deadband analysis with report messages.

Analog Signals: Template Collection

Following table includes results for the analog signals using the Template Collection system. No other activity was in progress in the DPC during these measurements. To facilitate easy comparison the template requests included the same number of signals that were used in the RBE measurements (even though it is possible to fit 38 analog signals in a single template response compare to 32 signals in the RBE Report Message).

Analog Signals: RBE and Template Collection

Following table includes the RBE results for the deadband analysis with reports and the Template Collection results.

The comparison is slightly biased towards the RBE as number of signals in a template response is 38 compare to the 32 signals in an RBE Report Message.

Analog Signals: Delta = RBE - Template Collection

Following table presents the previous table in different form. It gives the difference between the results of RBE system and Template Collection system. It shows a clear advantage for RBE

system when the rate of template collection and the RBE scan periods are the same.

Additional benefit may result if only a subset of all signals is in exception on every scan period. This will result in fewer number of exceptions and may result in reduced number of RBE Report Messages.

Logical Signals: RBE

Following table includes results for 5 sets of logical signals. During these measurements the logical signals were changed. An exception report was generated for each logical signal. Each group of 75 logical signals generated one RBE Report Message, thus measurements in the following table also correspond to 1 to 5 RBE Report messages respectively.

Logical Signals: Template Collection

Following table includes results for the logical signals using the Template Collection system. No other activity was in progress in the DPC during these measurements. To facilitate easy comparison the template requests included the same number of signals that were used in the RBE measurements (even though only 48 logical signals are included in a single template response compared to 75 signals in the RBE Report Message).

Logical Signals: RBE and Template Collection

Following table combines the RBE and the Template Collection results for the logical signals.

The results may not provide exact comparison as the number of signals that can be returned in the template response is 48 compare to the 75 signals in an RBE Report Message.

Logical Signals: Delta = RBE - Template Collection

Following table presents the previous table in different form. It gives the difference between the results of RBE system and Template Collection system. It shows a clear advantage for RBE system when the rate of template collection and the RBE scan periods are the same.

Additional benefit may result if only a subset of all signals is in exception on every scan period. This will result in

fewer number of exceptions and may result in reduced number of RBE Report Messages.

An Equation For RBE Load Requirements

The following equation is provided to calculate the approximate percentage of the CPU time required by the RBE system to process any given number and any given mix of analog and logical exceptions.

a. 0.973% To send the prepared RBE Report Message; including the communication system times.

b. 0.062% To analyze an analog signal and corresponding Deadband for an exception.

c. 0.018% To prepare and log the exception report in the RBE Report Message.

d. 0.0262% To analyze a logical signal and log the resulting exception report in the RBE Report Message.

% of CPU time required by the RBE =

(# of analog signals changed AND deadbands analyzed * 0.062) + (# of analog exceptions * 0.018) + (# of logical exceptions generated * 0.0262) + ((INT(((# of analog exceptions * 7) + (# of logical exceptions * 3) + 225) / 225)) * 0.973)

Example: 150 Analog RBE signals:
40 signals changed that has deadbands but only 35 of these signals require exception reports.
25 signals changed that do not have assigned deadbands.
200 Logical RBE signals:
110 signals changed but only 80 of these signals require exception reports.

% CPU time required by the RBE =

(40*0.062) + ((35+25)*0.018) + (80*0.0262) + (INT((((30+25)*7)+(80*3)+225)/225)*0.973) = 8.575%

Appendix E
NETTOP File Header Structure
(Not Applicable to Open BSI 3.0 or *newer*)

```
/*+ nettop.h - definitions for NETTOP program and Net. files user routines
 *
 *      This file contains the #defines and structure definitions used by
 *      the NETTOP program and by the Network Topology Files user access
 *      routines.  These structure definitions define the NETFILE.DAT
 *      record structures. (The same structures are used for working versions
 *      of the file, ie. xxx.NET.)  Structure definitions for the Network
 *      Topology cross reference files are also included.
 *
 *      netfildef - NETFILE.DAT (and .NET file) definitions
 *      xrefdef - XREF file record structure definitions
 *      ntuser_errors - error codes returned by Net. Top. file user routines
 *
 *      Note:
 *      This file was modified for BSI / ACCOL III tools project.  The
 *      typedefs were changed use the standard types provided by that project.
 *

/*+ netfildef - NETFILE.DAT (and .NET file) definitions
 *
 * FUNCTION
 *      This section includes the NETFILE.DAT (and .NET file) record structure
 *      definitions.
-*/
#ifdef BSI_PRAGMA_PACK
#pragma pack(1)
#endif

#define FLBASNAM_LEN 8              /* file base name length                */
#define MAXCFE 8                /* Maximum number of CFE nodes allowed      */
#define NDNAME_LEN 4                /* node name length                     */
#define NTNUMLEVELS 7               /* number of network levels, includes lev 0 */
#define MAXALMZON 32                /* max alarm zones allowed.             */
#define NOTNODEFLG (NTNUMLEVELS + 1)    /* if in 1st byte of rec, indicates */
                                    /*     not node record */

/* Network level data */

typedef struct level_data {
   UCHAR lev_shiftcnt;    /* shift count for this level               */
   UCHAR lev_mask;              /* mask for this level                  */
} LEVEL_DATA;
/*
 *    NETFILE header record
 *    Contains basic information required to construct network routing table
 */
typedef struct ntheadrec {  /* record 0 structure                       */
   UCHAR maxlevel;              /* neg. of max level num (to allow sorting) */
```

```
   USHORT ntzero;                  /* 0 fill (to allow sorting)                */
   LEVEL_DATA leveldata[NTNUMLEVELS];      /* level masks and shift counts     */
   UCHAR ntvers;                   /* current node routing table vers number   */
   UCHAR ntreletime[5];            /* date/time of last release via NETTOP     */
   UCHAR ntloadtime[5];            /* date/time of last route load             */
    /* (2 times above implemented only on VAX CONSOLE versions of NETTOP) */
   UCHAR cfescanmsk;               /* indicates which CFE's currently on scan  */
   USHORT ntrecmax;                /* maximum record number this file    */
   USHORT masrecnum;               /* master node record number                */
   CHAR gladvlflg;                 /* if YES, global addresses are valid       */
   CHAR sortvlflg;                 /* if YES, sort order is valid              */
   CHAR accessflg;                 /* if YES, NETFILE.DAT may be accessed      */
   UCHAR lev_max[NTNUMLEVELS]; /* level slave max as entered by user           */
   UCHAR ntflvers;                 /* network files version number             */
   CHAR netname[FLBASNAM_LEN + 1]; /* 1st 8 chars of netfile used to           */
                                   /* create current rel.  */
   /* 53 bytes used */
   UCHAR ntheadspare[11];
   /* 64 bytes used */
} NTHEADREC;
/*
 *   Node descriptor record
 *   Contains node routing information concerned with one particular node
 *
 *         Note:    this structure and the Node type definitions are mirrored in
 *                  the file BSI_NET.H.
 */
typedef struct ntnoderec {
   UCHAR nodelevel;                    /* node level (0-6), in .NET file set*/
                                       /*     to NOTNODEFLG if node deleted */
   USHORT globadr;                     /* global node address              */
   UCHAR locadr;                       /* local address (0-127)        */
   UCHAR alrmzone;                     /* alarm area number (zone)         */
   CHAR nodename[NDNAME_LEN + 1];  /* node name                            */
   CHAR nodefile[FLBASNAM_LEN + 1]; /* node ACO file base name             */
   USHORT loadvers;                    /* load ID or vers num (info only)  */
   CHAR nodedesc[32];                  /* node descriptor                  */
   UCHAR nodetype;                     /* node type: 3000, 3320, 3350, ... */
   CHAR predname[NDNAME_LEN + 1];  /* predecessor name                     */
   UCHAR nodestatus;               /* node status: on/off scan, avail/not avail*/
   UCHAR slavednum;                    /* num of slaved nodes defined      */
   /* 61 bytes used */
   UCHAR nodespare[3];
   /* 64 bytes used */
} NTNODEREC;

/* Node Type Definitions */
/* 0x80 and above is reserved for Enterprise PLC node types */

#define NDTYP_UNDEF  0
#define NDTYP_3330_4  1
#define NDTYP_3320    2
#define NDTYP_3350_4  3
#define NDTYP_3740    4
#define NDTYP_3380_4  5
#define NDTYP_CFE_4   6
```

```
#define NDTYP_CONSOLE       7
#define NDTYP_NETMON        8
#define NDTYP_3508      9
#define NDTYP_3330_5    10
#define NDTYP_3350_5    11
#define NDTYP_3380_5    12
#define NDTYP_3335_5    13
#define NDTYP_3745      14
#define NDTYP_VAX_ENT       15
#define NDTYP_IBM_ENT       16
#define NDTYP_GENESIS       17
#define NDTYP_PEI       18
#define NDTYP_3755      19
#define NDTYP_3308      20
#define NDTYP_3310      21
#define NDTYP_MASK  (BIT0 | BIT1 | BIT2 | BIT3 | BIT4)

/* Node status byte usage definitions */

#define NDSCN_OFF 0
#define NDSCN_ON BIT0
#define NDSCN_MASK BIT0

#define NDFILVAL_NO 0
#define NDFILVAL_YES BIT1
#define NDFILVAL_MASK BIT1
/*
 *    Alphabetic Index Structure Definitions, currently only used by NETTOP
 */
typedef struct ndnaminx {           /* alphabetic index entry structure */
   CHAR nminx_node[NDNAME_LEN + 1];         /* node name                    */
   USHORT nminx_rec;                        /* file record number this node     */
} NDNAMINX;

typedef struct ntindxrec {  /* alphabetic index record structure     */
   UCHAR  ntnotndflg;               /* set to NOTNODEFLG, indicates not node rec*/
   UCHAR  ntchrcnt;                 /* bytes used this record               */
   UCHAR  ntstrcnt;                 /* number of structures this record     */
   USHORT ntnxtlnk;                 /* record number of next record this type   */
   UCHAR ntreccon[59];  /* all other data, index entries            */
} NTINDXREC;

/* note: index entry structure is 7 bytes, 59/7 = 8 index entries per record */

#define NTIX_HD_SIZE 5
#define NTIX_NOTNDFLG ntinxrec.ntnotndflg
#define NTIX_CHRCNT ntinxrec.ntchrcnt
#define NTIX_STRCNT ntinxrec.ntstrcnt
#define NTIX_NXTLNK ntinxrec.ntnxtlnk
#define NTIX_RECCON ntinxrec.ntreccon
/*
 *    Union of all Record Types in Net File
 */
#define NTFLRECSZ 64            /* Net file record size                 */

typedef union {
```

```
   NTHEADREC nthedrec;
   NTNODEREC ntnodrec;
   NTINDXREC ntinxrec;
   UCHAR ntbytrec[NTFLRECSZ];
} NTFILEREC;
/*
 *   VMS record, used by NETTOP only
 */
typedef struct {
   SHORT ntrecind;      /* record number this entry - sign for written   */
   USHORT ntbcklnk;     /* link for previously accessed                  */
   USHORT ntfwdlnk;     /* link for next accessed                        */
   NTFILEREC ntrec;     /* actual NETFILE record                         */
} NTRECORD;
/*
 *   NETFILE location descriptor, used by NETTOP only
 */
typedef struct {
   USHORT ntrecord;     /* defines record number this structure          */
   UCHAR ntstrnum;      /* structure number this this structure          */
} NT_LOCATION;                  /* structure used to define current struct location */

/*+ xrefdef - XREF file record structure definitions
 *
 * FUNCTION
 *          XREF record structure definitions for RTUXREF.DAT
 *          and GLADXREF.DAT files
 *
-*/
/*
 *   Node XREF record
 */
typedef struct xrefrec {
   CHAR xrfndname[NDNAME_LEN + 1]; /* node name                          */
   USHORT xrfglobadr;           /* global node address, calc. by NETTOP   */
   UCHAR xrfndtype;             /* node type: 3000, 3320, 3350, 3380, 3600..*/
   UCHAR xrfntvers;             /* current routing table version number   */
   USHORT xrfntrec;             /* NETFILE record number this node        */
   UCHAR xrfspare[5];           /* spare to pad to 16 bytes */
} XREFREC;

#define RTUXREF_MODE 1
#define GLADXREF_MODE 2


/*+ ntuser_errors - error codes returned by Net. Top. file user access routines
 *
 * FUNCTION
 *          This section include the error codes returned by the Network Topology
 *          file user access routines.
 *
-*/

/*
 *   Return status codes for Network Topology file user functions
 */
```

```
#define NTERR_ACCESS254
        /* -2 Failure:      New Network Top. files being released, Access error */
#define NTERR_HDREAD        253
        /* -3 Failure:      Error reading NETFILE.DAT header record.    */
#define NTERR_INVNRTVER 252
        /* -4 Failure:      Node Routing Table version number mismatch.         */
#define NTERR_NETFLCL       251
        /* -5 Failure:      Error closing Network Topology file.               */
#define NTERR_NETFLOP       250
        /* -6 Failure:      Error opening Network Topology file.               */
#define NTERR_NETFLRD       249
        /* -7 Failure:      Error reading NETFILE.DAT.                         */
#define NTERR_XREFFLRD      248
        /* -8 Failure:      Error reading XREF file.                  */
#define NTERR_XREFNOREC 247
        /* -9 Failure:      Specified cross reference record not found.    */
#define NTFAILURE       255
        /* -1 Failure:      Unsucessful completion of function         */
#define NTSUCCESS       1
        /* 1 Successful completion of function.                        */

#ifdef BSI_PRAGMA_PACK
#pragma pack()
#endif
```

This page is intentionally left blank

Glossary

A glossary of commonly used terms is provided here to aide the reader in understanding this document.

ACK	-	Acknowledge Protocol message, message was successfully received
BSAP	-	Bristol Synchronous/ Asynchronous Communication Protocol
CRC	-	Cyclic Redundancy Checksum word
Data Highway	-	Bristol proprietary high speed multidrop, multimaster data highway
DLE	-	Protocol character (10H)
ETX	-	Protocol character (03H), End of Text
ISO	-	International Standards Organization
Master	-	Node on the network that has one or more Slave nodes, initiates all message transfers with its slaves
MSD	-	Master Signal Directory, data base in Node for all signals, lists, and data arrays
NAK	-	Not-Acknowledge Protocol message, message could not be received because no buffers were available (not applicable to a POLL message)
NSB	-	Node Status Byte (see Chapter 8 for description)
NME	-	Number of data Elements in a Request/Response message
NRT	-	configuration table for network topology, Node Routing Table
NETTOP	-	Name of software program used to produce the network's Node Routing Table. (NOTE: Not used with Open BSI 3.0

		or *newer*; those versions use NetView, instead.)
PEI		Portable Engineer's Interface. Any computer (typically a PC) running ACCOL Tools software.
Peer to Peer	-	main data transfer mechanism between nodes on the Network 3000
Poll	-	Protocol message used by a master node to obtain data/alarm information from a slave node
Protocol	-	rules to follow to be communicate on the Network-3000
Pseudomaster	-	Device that can be connected to a Pseudoslave port and request, or receive data from a node on the Network
Pseudoslave	-	Name of the port that can be connected to a Pseudomaster device
RDB	-	Remote Data Base Access system
RER	-	Return Error Response code, found in response message
Slave	-	Node on the network that has a Master, sends a message to its master upon receipt of a POLL message
SOH	-	Protocol character (02H), Start of Header

This page is intentionally left blank

Headquarters:
Emerson Process Management
Remote Automation Solutions
6005 Rogerdale Road
Houston, TX 77072 U.S.A.
T +1 281 879 2699 | F +1 281 988 4445
www.EmersonProcess.com/Remote

Europe:
Emerson Process Management
Remote Automation Solutions
Unit 8, Waterfront Business Park
Dudley Road, Brierly Hill
Dudley UK DY5 1LX
T +44 1384 487200 | F +44 1384 487258
www.EmersonProcess.com/Remote

North American/Latin America:
Emerson Process Management
Remote Automation Solutions
6005 Rogerdale Road
Houston TX USA 77072
T +1 281 879 2699 | F +1 281 988 4445
www.EmersonProcess.com/Remote

Middle East/Africa:
Emerson Process Management
Remote Automation Solutions
Emerson FZE
P.O. Box 17033
Jebel Ali Free Zone – South 2
Dubai U.A.E.
T +971 4 8118100 | F +971 4 8865465
www.EmersonProcess.com/Remote

Asia-Pacific:
Emerson Process Management
Remote Automation Solutions
1 Pandan Crescent
Singapore 128461
T +65 6777 8211| F +65 6777 0947
www.EmersonProcess.com/Remote

Remote Automation Solutions

EMERSON.
Process Management