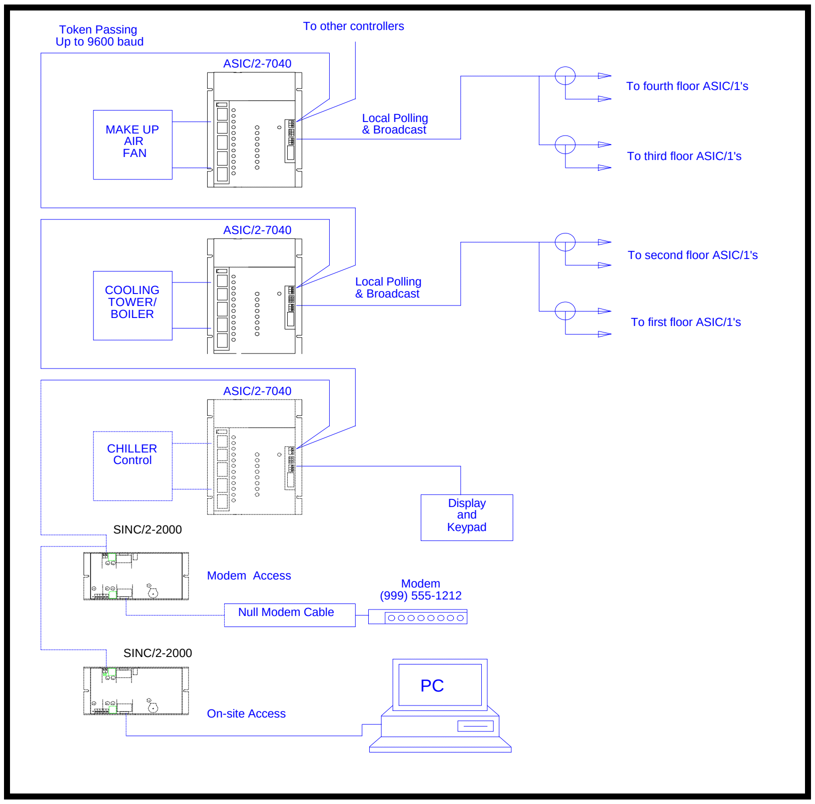
ASIC/2

Communication Protocol Revision 2.1

By ASI Controls



For further information or for the most current release of this document contact:

ASI Controls
2202 Camino Ramon
San Ramon, CA 94583
Phone: (925) 866-8808
FAX: (925) 866-1369

Customer Support: sales@asicontrols.com

Technical Support: techsupport@asicontrols.com

Visit our Web site at http://www.asicontrols.com

Contents

ASIC/2 Communication Protocol

Overview

This document defines the ASI Controls Standard Communications protocol as applied to the ASIC/2-7000, ASIC/2-7040, and ASIC/2-8040 family of configurable unitary controllers . These messages are used in firmware releases 700A.. 770A.., 740A.., 840A.., etc. This document does not contain detailed object descriptions. Each product has its own Users' Manual and Object Definitions.

Communication

The ASIC/2-7040 and ASIC/2-8040 have two independent communication busses. The system bus is the main communication bus and is available for token passing. The local bus is available for polling of controllers on the local bus for data and for broadcast of messages to those controllers. Alternatively the local bus may be used to support a Display and Keypad. The ASIC/2-7040 has green transmit and red receive LEDs on both the system and local busses that flash when communicating..

The ASIC/2-7000 has a single communication bus. This system bus is the main communication bus and is available for token passing. The ASIC/2-7000 power LED will flash when the controller is communicating. The LED is ON during non-communicating periods. The LED is OFF during transmit and receive. Thus, the LED will blink off during token passing and such activity. The LED may be off for extended periods during upload and download of objects using long messages at 1200 baud.

Baud Rate

The ASIC/2-7040 has the ability to communicate at 1200, 9600, 19,200, and 38,400 baud on both the system bus and at1200 and 9600 baud on local busses. Messages from the system bus to controllers on the local bus are passed through by receiving the entire message and retransmitting it. The system and local bus do not need to operate at the same baud rate. If either baud rate is changed the new baud rate becomes effective immediately.

Controller Addresses

The ASIC/2-7040 and ASIC/2-7000 controllers respond to messages on the system bus addressed to their System Bus Address. The ASIC/2-7040 also responds to messages on the local bus addressed to its Local Bus Address.

The ASIC/2-7040 and ASIC/2-7000 controllers have two single byte group addresses which will receive messages on the system bus. The devices also will receive messages sent to their device global address, 23,152, (5A 70h) and if System Global Enable is set, to the SC/1 global address, 23,295 (5A FFh).

Hardware Clock

The ASIC/2-7000 has the Dallas 1216E clock chip in a chip socket. TheASIC/2-7040 has a built in hardware clock in a can. If the hardware clock is enabled, the hardware clock is read at every half minute (30 s) and minute interval and is used as the master time keeper to update the software clock. If the hardware clock option is not enabled, then time is kept by the software clock which counts zero-crossings at 50 or 60 Hz.

About This Document

This ASIC/2 Protocol, DOC-1341, and Windows™ help system was produced using Microsoft® *Word* and *Doc-To-Help*®, by WexTech Systems, Inc. It was last revised or printed on 10/25/01.

ASI Controls is always working to improve our products. Should you have any questions, or suggestions that would help our products better meet your needs, or that would help us serve you better, please call, write, or e-mail to:

ASI Controls
2202 Camino Ramon
San Ramon, CA 94583
Phone: (925) 866-8808
FAX: (925) 866-1369
Customer Support: sales@asicontrols.com

Technical Support: techsupport@asicontrols.com

Visit our Web site at http://www.asicontrols.com

ASI Communication

ASI Control Philosophy

The ASI Controls philosophy is based on fully distributed intelligence, database, and communications. Each ASI Controller has intelligence in an Intel MCS-51 or Intel MCS-96 family microprocessor, non-volatile EEPROM memory, and dynamic RAM memory. The database of the controller is contained in EEPROM memory with setpoints which are preserved through power failure to allow the controller to function in a stand alone manner.

The RS-485 twisted pair communication line and two byte addressing allows complete communication with any controller on the network. The communication protocol provides access to any element of the controller data base and includes specific commands to facilitate system operation.

ASI Controls has developed a family of controller products including: the ASIC/1 series of 8 input and 8 output terminal unit controllers; the ASIC/2-7000 series of 8 input and 16 output unitary controllers; and ASIC/2-7040 series of 16 input and 20 output unitary controllers.   ASI Controls also has a series of communication products including the SINC/2-2000 System Interface and Network Controller.

Application Specific Intelligent Controllers

Basic Characteristics: The ASIC/1 and associated repeaters and communication interfaces contain an 8031, 8032, or 80C32, micro-controller, members of the Intel MCS-51 family. The ASIC/2 and associated repeaters and communication interfaces contain an 8096, 8097, or 80C196 micro-controller, members of the Intel MCS-96 family, . The communications concept of the ASIC Protocol is based on an internal Universal Asynchronous Receiver Transmitter (UART) contained in the micro-controller using the UART serial mark/space format. The particular byte format that has been adopted for the ASI Communication is one start bit, 8 data bits, no parity, two stop bits and least significant bit first. Communication baud rates up to 9600 baud are supported by the controllers.

Physical Layer: The physical layer that has been adopted for the ASIC Protocol is RS-485, half-duplex, two-wire, balanced line. The hardware provided in the controller and in all repeaters is the National Semiconductor integrated circuit, DS3695 and equivalents.

Pull-up and pull-down resistors are provided at all communication interface and repeater line terminals to enhance the quiescent line 'unbalance' when all drivers on the line are in the OFF tri-state mode. Transient protection for each line to common is provided at each controller and repeater with fast zener clamps, but no true lightning protection is provided anywhere within the system and must be provided where necessary by others. Fuses are provided on each communication line to protect the controller against improper wiring.

Communication Repeaters

The RS-485 specification includes a limit of 32 loads on any one line. This limitation is provided by specifying the number of loads that a driver must be able to support. In a typical large building environment, there may be several hundred controllers. The ASI Communications Protocol is intended to support polling of these controllers by a Host System, or by another configurable controller.

ASI Controls has developed the System Interface and Network Controller, SINC, a half-duplex RS-485 "Intelligent" Repeater to allow 100% use of the communications line. ASI Communication philosophy for getting maximum use of a two wire communication line requires that responses to request messages always follow directly after the checksum of those messages. The start bit of the response can occupy the next bit time after the second stop bit of the checksum of the request message. The SINC/1 uses a patented technique (U. S. Patent No. 4,811,195) to monitor the progress of each byte of every message and reverts to listening in both directions after the midpoint of the second stop bit of every byte. This allows 100% utilization of the line, but requires special considerations to avoid collisions. At least one (1) byte time of quiet is left on the line between message pairs.

Configurable Controllers

ASI Controls has developed a Configurable Unitary Controller, the ASIC/2-7000, which uses an 8098 micro-controller, and can operate on a system bus with other controllers.

The ASIC/2-7040 Configurable Controller has a local communication bus in addition to the system bus. The ASIC/2-7040 controller uses an 8097 or 80C196 micro-controller.

The local communication bus is used to communicate with terminal controllers, can be configured to poll for alarms, temperature and airflow conditions, modify system operation based on information received from the field, and supervise the operation of 64 or more controllers.

The system bus is a token passing bus. TheASIC/2-7040 can listen and originate communications independently and simultaneously on both the system and local busses. It monitors the progress of every message so as to remain synchronized with all messages and responses.

ASIC Protocol Collision Protection

The ASI Communications Protocol uses peer to peer token passing on the system bus. Each token player listens for its turn to receive the token. While it has the token, it can broadcast any messages such as time or remote points, and then passes the token to the next token player. The token bus is protected against message collisions. Only the token holder is authorized to initiate request messages. If there are any unexpected communications on the line, the controller drops the token and waits until the line is quiet. The token passing is quickly reinitialized.

Collision prevention is used to minimize message collisions. Collision prevention is implemented by requiring all message originators to listen to the communications line for at least one full byte length prior to originating a message. There must not be any signal on the communications line during that length of time. A software timer is reset at each serial data interrupt.

This technique prevents one originator from corrupting a message from another already in progress. Note that only originator messages are subject to collision, since responses follow directly on the heels of the requesting message. Every message to an individual controller is acknowledged.

Failures to receive an ACK (06 hex) message in response to a request usually elicits a re-transmission.

All ASI configurable controllers keep track of the framing of every message, even if the message is not addressed to them. They monitor each message byte so that they know when the request message ends and a response starts.

ASI Setup Software and the ASI DDE Server use a gap protocol that periodically leaves quiet spaces on the communication line, and listens before beginning transmission to allow other communication traffic to proceed and minimize message collisions.

More than one device may wish to originate messages on the communication line. For example, a host computer may be polling continuously, and an operator may also wish to communicate on the bus in trouble shooting or commissioning a controller. Therefore communication etiquette requires that any device originating messages should leave a quiet gap of 100 ms every 2 seconds to allow time sharing of the communication line.

RS-232 Access to the token passing system bus is controlled through a System Interface and Network Controller, SINC/2-2000. The SINC/2-2000 is configured to be a token player. Upon receiving a request for access on its Access Bus, the SINC/2-2000 at the next opportunity receives and holds the token and allows the host software to communicate for a token hold time. When the token hold time expires, access is interrupted and the token is passed. Access is restored when the SINC/2 gets the token again. This gives every controller on the system bus a timely opportunity to transmit messages. There can be multiple points of access to the token bus, each getting their share of the access time.

ASIC Protocol Message Format

The ASI message format is strictly defined and provides full identification of start of message, originator, destination, message type, framing, message body, and checksum. Each different message type has a unique definition of the contents internal to the body of the message to allow for a wide variety of instructions and commands. Once a new message type is defined it does not change. It may not be applicable to or implemented in every product.

ASI Controls publishes its protocol to assist customers in integrating the communicating control products into their applications. Over 20 vendors have successfully completed interfaces to ASI controller products.

In this document all byte values are in hexadecimal (00 to FF hex) representation unless otherwise noted (0...255 decimal). For bitwise representation the least significant bit is lsb = bit 0 = 01 hex, and the most significant bit is msb = bit 7 = 80 hex.

Message Request

The message request has 7 parts: the header, destination address, source address, message type, frame byte, message body and checksum.

Message Request:

Header, Destination, Source, Type, Frame, Body, Check

Header: The character STX , start transmission, (02 hex) is used as the first byte of all messages, regardless of the source or destination or intention of the message.

Destination: A two-byte (16 bit) **Destination Address** is used to provide a large address domain. The High Byte of the address is transmitted first, followed by the Low Byte.

The destination can be an individual controller address, a group address, or a global address. Individual addressing allows a specific device to be accessed by another device in the system. When using individual addresses two-way communications can be accomplished.

Each ASIC/1 controller has a two-byte address ranging from 1 (0001 hex) to 23,039. (59FFhex). The factory assigned address of ASIC/1 controllers is 16,000 (3E80 hex). Some addresses above this range are reserved and there are certain limitations on controller addresses:

Group Addresses: Addresses that are evenly divisible by 256 (Least Significant Byte = 0) are used for group destination.

Global Addresses: Addresses 23041 to 23295 ('5A 01 hex' through '5A FF hex') are reserved for device global addresses.

Initialization Addresses: Addresses 46,081 through 46,335 ('B4 01 hex' through 'B4 FF hex') are reserved for initialization of device addresses.

Token Passing Addresses: Addresses 32,001 through 32,255 ('7D 01'hex through '7D FF hex) are used for Peer to Peer and Token Bus addressing functions.

Other Special Addresses: Address 50,115 ('C3 C3 hex') is reserved for extended group addressing.

Source: A two-byte (16 bit) **Source Address** is used to identify the message originator. This becomes the destination address for a response message. The High Byte of the address is transmitted first, followed by the Low Byte.

Type: A single-byte (8 bits) **Message Type** is used to signify what the message is intended to accomplish. The meaning of some message types may be further qualified by the values in the message body.

Frame: A **Framing Byte**, 77 hex , is included within the format of the message.

Body: The **Message Body** of the ASI message contains all of the necessary instructions, locations and data that may be required to carry out the intent of the message type. The request message bytes are numbered sequentially starting with the first byte after the framing byte being M1, followed by M2....up to Mn, as required. The definition of each message byte in each message type is given below.

Check: The **Checksum** is the simple algebraic sum of ALL bytes of the message, up to but not including the Check byte, modulo 256. The Checksum is appended to the message as the last character.

Message Response

The message response has 6 parts: the header, destination address, source address, acknowledgement, message body and checksum.

Message Response:

Header, Destination, Source, Ack, Body, Check

Header: The first byte of the response message is STX (02 hex).

Destination: A two-byte (16 bit) **Destination Address** is used to return the response message. This is the source address from the request message. The High Byte of the address is transmitted first, followed by the Low Byte.

Source: A two-byte (16 bit) **Source Address** is used to identify the message responder. This is the destination address from the request message. The High Byte of the address is transmitted first, followed by the Low Byte.

ACK: Almost all message types sent to an individual controller destination require a response . In all responding messages, this byte is used to acknowledge (ACK) the receipt of the valid message.

Body: The Response Body of the message follows the acknowledgement and contains response data. There is a limit of 127 bytes of returned data. The response message bytes are numbered sequentially starting with the first byte after the acknowledge byte being M1, followed by M2....up to Mn, as required. The meaning of response message bytes for each message type is defined below.

Check: The Checksum is the simple algebraic sum of ALL bytes of the message, up to but not including the Check byte, modulo 256. The Checksum is appended to the message as the last character.

Note: If an ASIC/1 receives a response checksum with a value of 02 hex and the next message is sent to a new address, then the new address may not respond. The new address has already received 02 hex which it believes is a header. If the checksum is not received in 220 milliseconds, the ASIC/1 communications sequence will abort. ASIC/2 configurable controllers remain in synchronization with all messages and responses and do not exhibit this behavior.

ASI Basic Host Messages

The ASI Setup Software has a mode of operation which is referred to as ASI Basic Host. With Basic Host one can explicitly send protocol commands and inspect the response message in hexadecimal representation. Basic Host is not user friendly, however it can be extremely valuable in trouble shooting difficult job problems.

Basic Host mode is reached by pressing <Alt+D> from the main menu screen of ASIC/1 Setup Software, or by pressing <Alt+F3> from SETSYS Setup Software. Basic Host assumes that the destination address is the current device address being used by the Setup Software. Basic Host asks for the message type, MT, followed by the message bytes, M1...M6, all in hexadecimal. Each entry is made followed by pressing <Enter>. When the final message byte has been entered, it will then display the response message bytes.

```
 Header, Destination, Source, Type, Frame,      Body    , Check
STX    Dst1 Dst2    Src1 Src2   MT FR M1 M2 M3 M4 M5 M6 Chk
```

Spy

The ASI Setup Software has a mode of operation which is referred to as Spy. With spy, one can observe byte by byte the protocol request and response messages in hexadecimal representation. Spy breaks each line at a new 02 hex which it interprets as a STX, start of transmission. Spy is not user friendly, however it can be extremely valuable in trouble shooting communication problems.

Controller Addressing

Device Addresses

Each controller has a 2 byte address that allows it to be directly addressed with commands on the communications line. When the controller recognizes its address, it will then process the message and deliver the appropriate response message. Device addresses that are evenly divisible by 256 are reserved for Group addressing. The ASIC/2-7040 has separate device addresses for the system bus and the local bus.

Group Addresses

The message destination can be a Group Address. Each controller can be assigned a separate single byte group address, 1..255. The ASIC/2-7000 and ASIC/2-7040 have two single byte group addresses. Group addressing is used to send a one way communication to a specific group of devices in the system. This allows a single message to be communicated to a predefined group of controllers. No response is made by any devices listening to a message sent to a group address.

The destination address in the message is constructed by using the group address as the high byte, and zero as the low byte. "Group" addresses 00 .. 255 (00 ...FF hex) can be assigned. Consequently, addresses which are a multiple of 256, for example 0, 256, 512, etc., are reserved for group addresses and should not be used for controller device addresses.

Care must be taken in sending commands to a group destination address. Only controllers of a single type should be assigned to the same group, because each type has different parameter assignments. For example, ASIC/1-8015 VAV controllers may have different setpoint assignments than ASIC/1-4300 Heat Pump Controllers.

For earlier ASIC/1-8010 controllers with EPROM firmware versions 137F through 137L one further restriction applies to group addresses. The group address should not be the same as the most significant byte of any individual controller address.

Initialization Addresses

Addresses 46,081 through 46,335 ('B4 01 hex' through 'B4 FF hex') are reserved for initialization of device addresses. These addresses are typically used with message type, 42h, Get address, to return the assigned device address of a controller. It is used with a hardware interlock in the ASIC/1 controllers. The ASIC/2 controllers do not use a hardware interlock.

Address 46,112 (B4 20 hex) is used with SINC/2-2000 Controller to retrieve the Device Address.

Address 46,165 (B4 55 hex) is used with a hardware interlock on ASIC/1-8X55 controllers to perform certain commands such as installing a new controller address and loading the default table of parameters.

Address 46,192 (B4 70 hex) is used with a ASIC/2-7000 and ASIC/2-7040 Controller to retrieve the Device Address.

Address 46,260 (B4 B4 hex) is used with a hardware interlock on ASIC/1 terminal unit controllers to perform certain commands such as installing a new controller address and loading the default table of parameters.

Address 46,334 ('B4 FE hex) is used on the system bus of a System Controller to retrieve the System Bus Address.

Address 46,335 (B4 FF hex) is used on the local bus of a System Controller to perform certain commands such as retrieving and installing the local bus system controller address.

Global Addresses

All messages to ASI controller may be transmitted with a special destination address of '5A XX hex' which is interpreted by all listening devices as though the message carried their specific address. The Global address is fixed in each ASI controllers

Addresses 23,041 through 23,295 (5A 01 hex through 5A FF hex) are reserved for device global addresses. Only the destination address can be global. All messages sent to the global address associated with a particular device will be received and acted on. No response is made by any listening devices to a 'Global' message.

Use of global address should be restricted to time synchronization (message type 38h), Set operating state (message type 10h), Set Emergency State (message type 12h). Other messages should be used only if all controllers on the system are the same type.

All controllers receive the message sent to a global address on the network and act on it unless they have been programmed to ignore it . Global broadcast messages are always broadcast 3 times with a gap of approximately 50 ms between each repeated message.

Recent versions of firmware allow for Device Global addressing which allows for global downloads of parameters and setpoints to all controllers of a particular type, without affecting other controllers which use the same parameter location for a different purpose. The following Global Addresses have been defined:

Address 23,045	(5A 05 hex)	ASIC/1-8055 VAV Controllers (155A1.9..)
Address 23,061	(5A 15 hex)	ASIC/1-8015, -4015 VAV Controllers(150E..,154E..)
Address 23,077	(5A 25 hex)	ASIC/1-8255, Fan Coil Controllers (255A16..)
Address 23,093	(5A 35 hex)	ASIC/1-8355, PAC Controllers (355A1.7..)

Address 23,107 (5A 43 hex) ASIC/1-4300 Heat Pump Controllers (304G..)
Address 23,125 (5A 55 hex) ASIC/1-8x55 Controllers(155A,175A, 255A, 355A)
Address 23,130 (5A 5A hex) All ASIC/1 terminal unit controllers.
Address 23,152 (5A 70 hex) ASIC/2-7000, -7040 Controllers (700A..,740A..)
Address 23,154 (5A 72 hex' ASIC/1-7210 Heat Pump Controllers (721C...)
Address 23,157 (5A 75 hex) ASIC/1-7510 Single Zone Controllers (751A..)
Address 23,158 (5A 76 hex) ASIC/1-7610 Multi-stage Controllers (761A...)
Address 23,170 (5A 82 hex) ASIC/1-8200, -8205 Fan Coil Controllers(251A...)
Address 23,295 (5A FF hex) System Global SC/1, ASIC/2 (907A..,700A..)

ASI Protocol Message Types

The ASIC/1 message format for the terminal unit control products includes a reasonable number of message types that provide for future needs. Eight distinct classes of message types are presently included in the ASI Communication Protocol: Poll, Download, Command, Response ,Table, Object, Token, and Display.

Poll Messages

Poll messages include requests for a single byte or multiple bytes of data from dynamic RAM, static EEPROM and program PROM memory. All data in the controller can be accessed with poll messages. The number of bytes of data that can be obtained with one request is limited, for practical reasons, to about 64 data bytes. Use of absolute read commands requires knowledge of the memory locations in the specific ASIC/1 firmware version and is not recommended.

Download Messages

Download messages transmit data to static or dynamic locations in the controller.

Data downloaded to static EEPROM will remain in memory, regardless of power loss. Due to this, repetitive downloading of setpoints and variables is not necessary. EEPROM should not be written to more than 10,000 times in a lifetime. In general, messages which write to EEPROM should be sent only in response to an operator request. EEPROM data can be read an unlimited number of times.

The number of bytes of data that can be downloaded to an ASIC/1 terminal unit controller in a single download message is limited to 4. A message to write to EEPROM causes a delay of 20ms/byte of written data from receipt of the response checksum in responding to a new message. When downloading a series of bytes, a delay of at least 1 second should be included before initiating the next message to that same controller to allow for the completion of the EEPROM write. The controller will not listen to any messages until the EEPROM write has been completed.

In the SC/1 and ASIC/2 controllers the number of bytes of data that can be downloaded with one message is limited to 64 data bytes. Static EEPROM data in the SC/1 and ASIC/2 controllers is written to a mirror image of EEPROM in dynamic RAM memory. Values of static EEPROM data that have changed are updated to non-volatile memory in background. Therefore, it is important that power to the controller NOT be turned off for at least 90 seconds after a change of static EEPROM data to allow the change to be transferred to non-volatile memory. No delays between writes are needed in the ASIC/2 controllers.

Command Messages

Command Messages result in a change of the operation of the controller. This activity may not directly change a value in the controller memory, but includes the setting or resetting of logical flags, or writing to dynamic locations that change the response of the controller.   This is the principal method of changing the operation of the ASIC/1 controller.

In the ASIC/2 configurable controllers commands are often executed by writing to the Action attribute of an object using an Object Write Message

Note: Certain ASIC/1 commands, e.g. emergency mode, and output overrides in the ASIC/1, write to EEPROM, so that the override will be maintained through power failure. EEPROM should not be written to more than 10,000 times in a lifetime. In the ASIC/1-8X55 family of controllers. output override write to RAM.

Response Messages

Almost all request messages sent to an individual controller require a response from the destination. For Poll Messages the response data is contained within the body of the response message starting with the first response message byte, M1. For Download and Command Messages the response is often a simple acknowledgement. There is a limit of 127 bytes of returned data.

Header, Destination, Source, ACK, Response Body , Check
STX Dst1 Dst2 Src1 Src2 ACK   M1   M2   ...   Mn Chk

If a request message is improperly formed, does not have the correct framing byte, does not have the correct number of bytes, or has a check sum error, the controller does not response.

Table Messages

Table messages are used to read (Message Type 7Eh)and write (Message Type 7Dh) data in the controller. The data includes one or more values from tables of setpoints and parameters, to read RAM variables containing current values and status information, and to initiate other functions. The structure of the table messages are uniform with every ASI controller that implements them, but the meaning of table entries may be different with each product line. In the ASIC/2-7000 and ASIC/2-7040 table messages are used to read and write one or more values from pages of memory.

Object Messages

Data Object messages (90h/91h) are used to read (Message Type 91h)and write (Message Type 90h)structured blocks of data to and from ASIC/2-7000 and ASIC/2-7040 Configurable Controllers. One or more EEPROM setpoints, enable flags, and other parameters and RAM variables containing current values and status information can be read or written in a single command. Each data object provides information for a specific control object in the controller. The structure of the data objects is uniform with every Controller. The meaning of each data object is specific and is defined in the object definitions for that controller.

Token Messages

Token Messages are used on the system bus to pass and acknowledge receipt of the token among controllers in the range of address from 32001 to 32225. The SINC/2 also uses these messages to request and release access to the token bus

Display Messages

Display Messages are used by the Display and Keypad to receive data from and pass data to a properly configured ASIC/2-7000 or ASIC/2-7040 controller. Display messages are originated by the DAK-002 controller and use the ASIC/2 Service Address as the destination address. The ASIC/2 Service Address is separate from the device address.

Message Framing

It is necessary for the host computer to listen to every message to keep track of message framing. Every ASI Protocol request message begins with a header byte, STX (02hex) followed by 4 bytes with the Destination address (HI,LO) and Source address (HI,LO). The next two bytes consist of the Message Type followed by the Framing byte. The framing byte is 77 hex. If byte 7 of the message is not a valid framing byte, then it is not a request message. The message body then follows with the message being terminated with a check sum. When a request message is identified, the message type can be saved, and the number of expected bytes can be determined. The expected byte count of the response message can also be determined from the message type.

Every ASI Protocol request and response message has a determined message length. Basic ASI Messages have fixed length of up to 11 bytes, some of which can be determined from a data table. ASI Table messages and certain basic host messages have a length that can be determined from reading a specific data length byte in the message body.

The response message begins with a header byte STX (02 hex) followed by 4 bytes with the Destination address (HI,LO) and Source address (HI,LO). The next byte consists of the acknowledge which is ACK (06 hex) . If framing is not kept orderly, the 06 hex value could be a valid message type 06h of a request message. Use of message type 06, Read multiple bytes from EEPROM, is discouraged. If the 06 hex is found in the 6th message byte, then it is generally assumed that it is a response message. By examining the succeeding byte, the presence of the framing character can be used to affirm or deny the 06 hex as a command or response. In addition, by examining the previous message type, one can determine the expected response message byte count, or location of the byte count for variable length messages. Continued processing of the message with the validated checksum, will verify that the communication monitoring is still properly framed.

If framing is lost, it is necessary to monitor the line for a new header byte, STX (02 hex), and re-synchronize to maintain proper message framing.

ASI Protocol Format

Each ASIC/1 Protocol Message Type is presented in the following format for ease of reference:

7Eh Get Standard Table Value

This command gets values from predefined tables. The tables will have different meaning from product to product.

Message body:
- M1 = Table Number
- M2 = Starting Byte
- M3 = Number of Bytes

Response:
- M1 = 7E
- M2 = Table Number
- M3 = Starting Byte
- M4 = Number of Bytes to return
- M5 ... Mn = Data bytes returned.

"7Eh" represents the message type. It is the message type that would be entered in using "ASI Basic Host" software.

"Get Standard Table Value ..." is a short statement describing the general function of the message type.

"Message body: M1 = ..." states the options that are available in the message body for the message type. "Table Number" represents the data that would be entered for message byte M1 in using the "ASI Basic Host" software. A short statement describing the specific function of a message entry may also be included.

"Response: " is the ASIC/1 response and what it represents. More than one response message body byte M1, M2, .., may be returned if the message type allows for it. If no message body bytes are returned, then the response is limited to "ACK" for acknowledge.

Firmware Revisions

To the extent possible backward compatibility is preserved from firmware version to version. The firmware revisions and the changes and features added to each are indicated at the end of this document. Each message type is identified as to the firmware revisions which are covered.

(137K...137Q) indicates that it applies from FW 137K through FW 137Q.

(907A.....,700A...) indicates that it applies from SC/1-9040 FW 907A and ASIC/2-7000 FW 700A and all subsequent releases.

(200A.. Only) indicates that it applies only to the SINC/2-2000 FW200A and subsequent releases.

ASIC/2 Protocol Messages

Group 1: Read and Modify Memory Directly

This group of message types is typically used only for diagnostics by ASI Controls. Knowledge of individual memory address locations is required and unpredictable from one version of firmware to another. These messages are Not used with ASIC/2 configurable controllers.

Group 2: Read and Modify the ASIC/1 State

10h Set/Reset Operating State

This message writes a value to Override State in Utility object, index zero, UTIL-0,Attr 0. (770A..,700A..) This message writes to RAM.

The specific action to be taken based on the Override State must be configured by the user.

0 = No Override
1 = Unoccupied
2 = Occupied
3 = Night Setback
4 = Morning Ready

Message body:

M1 =	01 - No Action
	02 - No Action
	03 - Set Override State to 4 = Night Setback
	04 - Set Override State to 3 = Morning Ready
	05 - Set Override State to 2 = Occupied
	06 - Set Override State to 1 = Unoccupied
	07 - Restore Override State to 0 = No Override
	08 - No Action
	09 - No Action
	10 - No Action

Response: ACK

12h Set/Reset Emergency State

This message writes a value to Emergency State in Utility object, index zero, UTIL-0, Attr 1. (770A..,700A..) This message writes to RAM.

The specific action to be taken based on the Emergency State must be configured by the user.

Message body:

M1 = 1 - Set Emergency State to 1
2 - Set Emergency State to 2
3 - Restore Emergency State to 0 = No Emergency
4 - No Action
5 - No Action
6 - No Action
7 - Call Request 2 Clear

Response: ACK

16h Set/Reset Demand Status (700A..,770A..)

This message is used to set the demand level and demand group. This message writes a value to Demand Status in Utility object, index zero, UTIL-0, Attr 2. (770A..,700A..) This message writes to RAM.

The specific action to be taken based on the Demand Status must be configured by the user.

Message body:

M1 = LO Byte - Demand Level
0 - Set Demand Level = 0
1 - Set Demand Level = 1
2 - Set Demand Level = 2
3 - Set Demand Level = 3
4 - Set Demand Level = 4
5 - Set Demand Level = 5
6 - Set Demand Level = 6

M2 = High Byte - Demand Group
[0, ..,6] Demand Group

Response: ACK
M1 = 16

Group 4: Messages to Handle Inputs

33h Return Raw Input Data

This reads the input channel using the Analog to Digital Converter at unity (1x) gain. This is for diagnostic use only. (700A..)

Message body:

M1 = Input Channel to read

1..8	- ASIC A channel 0..7 Input Index 0..7
17..20	- ASIC/2-7000 only 10 bit ADC Microprocessor ACH4..7, Input Index 0..2 and Unregulated supply voltage
21..22	- ASIC A MPX 8,9 Calibration

Response:
M1 = Integer output of the analog to digital converter. (FW700A.)

The result of 8 or 10 bit ADC is right adjusted in the returned byte.

Group 5: Time Related Messages

38h Synchronize Real Time Clock

Real Time Clock Information using time in "Host" computer. (FW700A..,FW740A..)

Message body:
M1 = Day, 01...07 where 1 = Monday (81...87 hex represent holidays)
M2 = Hours, 0...23 decimal
M3 = Minutes, 0...59 decimal
M4 = Seconds, 0...59 decimal

Response: ACK

39h Return Real Time Clock Data

This message operates on the local bus only. (FW700A..)

Message body: none

Response:
M1 = Day, 01...07 where 1 = Monday (81...87 hex represent holidays)
M2 = Hours, 0...23 decimal
M3 = Minutes, 0...59 decimal
M4 = Seconds, 0...59 decimal

3Fh Time and Date Messages

A set of time and date messages are used to get and set clock time and calendar date.

3Fh, M1 = 01 Set Real Time Clock

This message tells the controller to set its software clock to the time specified in the message body. If Update Hardware Clock is enabled then this message will also set a new time in the hardware clock. (FW700A..,FW740A..,FW200A..)

Message body:
M1 = 1
M2 = 7
M3 = seconds (range: 0...59;)
M4 = minutes (range: 0...59; 00...3B hex)
M5 = hours (range: 0...23; 00...17 hex)
M6 = day of month (range: 1...31; 1...1F hex)
M7 = month (range: 1...12; 01..0C hex)
M8 = year (range: 0...99; 01...63 hex)
M9 = Day of Week
LSNBL = Day, 01...07 where 1 = Monday
bits4,5,6 - Special Day Status (0..7) (FW740C)
bit 7 = Holiday
Response: ACK

3Fh, M1 = 05 Return Current Time and Date

This message causes the controller or host to respond with its current time and date in the format shown below.(FW700A..,,FW740A..,FW200A..)

Message body:
M1 = 5
M2 = 0
Response:
M1 = seconds (range: 0...59; 00...3B hex)
M2 = minutes (range: 0...59; 00...3B hex)
M3 = hours (range: 0...23; 0...17 hex)
M4 = day of month (range: 1...31; 01...1F hex)
M5 = month (range: 1...12; 01..0C hex)
M6 = year (range: 0...99; 00...63 hex)
M7 = day of week (range: 1...7, 1 = Monday)
bit 7 =1 if holiday.

Group 6: General Housekeeping Messages

42h Return Device Address

Requesting the controller address requires 'B4 70h' destination address and returns the Device Address.

Message body: None
Response:
M1 = High order address byte
M2 = Low order address byte

48h Reset Controller

The controller software clock will lose synchronization during a reset of power. The Reset message causes the controller to respond as if power had been turned off and turned on. The Reset message never gets a response message. . If a hardware clock is installed and enabled the hardware clock time will be read at reset.

Local or System Bus

Message body: None

Response: None

4Ah Who Are You?

Return Product and Firmware Version and Revision. This message returns 10 ASCII characters from PROM. The product number is a unique 4 digit ASCII number that identifies the product, e.g. 7000. The version number is a two digit ASCII number starting with 10 which reflects subsequent changes in hardware or firmware which are backwardly compatible with previous versions. The firmware revision level reflects the changes to firmware with added features and bug fixes, 700A, 740A, etc.

ASIC/2-7040 FW 704A Rev 1.2	returns "704012740A"
ASIC/2-7000 FW 700G Rev 1.4	returns "700014700G"

Message body:
M1 = 1
Response:

M1...M4 = product number in ASCII (4 bytes)
M5...M6 = product version in ASCII (2 bytes)
M7...M10 = PROM revision number in ASCII (4 bytes)

Group 7: Display Messages

The 50h Request data and 51h Send Data Messages are used by the Display and Keypad, DAK-001, an DAK-002 to communicate with theASIC/2-7000 and ASIC/2-7040 controllers that has the appropriate DAK and Service Addresses.(FW700A.., FW740A..) The format of the DAK Messages are shown below. The ASIC/2-7000 will respond to these messages. For details please consult the DAK Communication Protocol Manual, DOC-1364.

50h DAK Request Data

Message body:

M1 = n - Request Type
M2 = 0 - Request Byte Count remaining
M3 = Index Number (Optional)
CHK

Response: ACK

M1 = Request Type
M2 = n - Response Data Byte Count remaining
M3 = Request Data 1.
...
Mn+2 = Request data n
CHK

The following Upload Messages Types M1 have been defined:

M1 = 1 - Initialize Communication
M1 = 2 - Get Display Manager Label
M1 = 3 - Get Display List Label
M1 = 4 - Get Display Item Data
M1 = 5 - Get Schedule Data
M1 = 6 - Get Holiday Data
M1 = 7 - Get Alarm Data
M1 = 8 - Get Password Data
M1 = 9 - Get Authorization Level

51h Download DAK Data

Message body:

M1 = 1 - Download Type
M2 = n - Download Byte Count remaining
M3 = Download Data 1.
...
Mn+2 = Download data n
CHK

Response: ACK

M1 = 1 - Download Type
M2 = 0 - Response Data Byte Count remaining
CHK

The following Download Messages Types M1 have been defined:

M1 = 1 - Parameter
M1 = 2 - (reserved)
M1 = 3 - (reserved)

M1 = 4 - Overrides
M1 = 5 - Schedule
M1 = 6 - Holiday
M1 = 7 - Alarm
M1 = 8 - Password

5Fh, M1 = 01 Set Real Time Clock

This message tells the controller to set its software clock to the time specified in the message body. If Update Hardware Clock is enabled then this message will also set a new time in the hardware clock. (FW700A..,FW502A) This is modeled on the MT 3Fh, M1 = 01 message except that it uses the ASIC/2 Service address. .

Message body:
M1 = 1
M2 = 7
M3 = seconds (range: 0...59;)
M4 = minutes (range: 0...59; 00...3B hex)
M5 = hours (range: 0...23; 00...17 hex)
M6 = day of month (range: 1...31; 1...1F hex)
M7 = month (range: 1...12; 01..0C hex)
M8 = year (range: 0...99; 01...63 hex)
M9 = day of week (range: 1...7, 1 = Monday)
Response: ACK

5Fh, M1 = 05 Return Current Time and Date

This message causes the controller or host to respond with its current time and date in the format shown below.(FW700A..,502A) This is modeled on the MT 3Fh, M1 = 05 message except that it uses the ASIC/2 Service address.

Message body:
M1 = 5
M2 = 0
Response:
M1 = seconds (range: 0...59; 00...3B hex)
M2 = minutes (range: 0...59; 00...3B hex)
M3 = hours (range: 0...23; 0...17 hex)
M4 = day of month (range: 1...31; 01...1F hex)
M5 = month (range: 1...12; 01..0C hex)
M6 = year (range: 0...99; 00...63 hex)
M7 = day of week (range: 1...7, 1 = Monday)
bit 7 =1 if holiday.

Group 10: Table Messages

The Table messages are used to change parameters and setpoints. These messages have been implemented as data table entries using Set/Get Table messages (7Dh/7Eh) These table messages are byte oriented. That is each entry is assumed to consist of a single byte of data. As new controllers are developed the meaning of the data table entries for the different controllers will be different.

7Dh Set Standard Table

This message defines a set of variables which can be set or reset.

Message body:

M1 = Table Number
M2 = Starting Byte
M3 = Number of Bytes
M4 ... Mn = Data bytes (limited by message buffer size)

Response:

M1 = 7D
M2 = Table Number
M3 = Starting Byte
M4 = Number of Bytes actually set

7Eh Get Standard Table Value

This message gets values from predefined tables. The tables will have different meaning from product to product.

Message body:

M1 = Table Number
M2 = Starting Byte
M3 = Number of Bytes

Response:

M1 = 7E
M2 = Table Number
M3 = Starting Byte
M4 = Number of Bytes to return
M5 ... Mn = Data bytes returned.

Memory Configuration.

In the configurable controller the table messages are used for diagnostic purposes only and reference 256 byte pages of memory. The configuration of memory depends on the hardware. The Table message is also use download the allocation to the controller.

RAM Memory(8k or 32 kbytes)

The SC/1-9040 and ASIC/2-7000 FW700A..G use 8 k RAM memory chips. These RAM Memory 32kx8 28 Pin DIP chips are identified as follows: Hitachi HM6264LP

The ASIC/2-7000 FW700H.. and ASIC/2-7040 FW700A.. use32 k RAM memory chips. These RAM Memory 32kx8 28 Pin DIPchips are identified as follows: Hitachi HM62256LP, SONY CXK58256PM or CXK58256AP, NEC uPD43256AC, MOSEL MS6256L.

EEPROM Memory (2k or 8kbytes)

The early SC/1-9040 and ASIC/2-7000 FW700A.. use 2 k EEPROM memory chips. These EEPROM Memory 2kx8 28 Pin DIP chips are identified as follows: SEEQ 2817

The later SC/1-9040 and ASIC/2-7000 FW700A.. use 8 k EEPROM memory chips. These EEPROM Memory 8kx8 28 Pin DIP chips are identified as follows: ATMEL 28C64, SEEQ or 28C64

Memory Assignment - ASIC/2-7000 FW700A..FW700G

The ASIC/2-7000 FW700A..G and SC/1-9040 FW907A..use 8 k RAM and has the following assignment of memory.

Absolute Page	
	Table 0. Internal RAM;
A000h	Table 1 to 11 Dynamic Object data in External RAM ;
	Table 12 Communication Queues;
AC00h	Table 13 to 28 Image of Static EEPROM object data in External RAM;
BC00h	Table 29 to 32 Processor stack (1 kbyte) in External RAM ;
C000h	Table 33 to 49 External EEPROM object data (4 kbyte).

Table 13 begins with 16 bytes of reserved data. The next 64 bytes contain the static allocation object data. The static data for the remaining allocated objects follows up to a maximum of 4 kbytes.

Memory Assignment - ASIC/2-7000 FW700H..

The ASIC/2-7000 FW700H uses 32 k RAM and has the following assignment of memory.

Absolute Page	
	Table 0. Internal RAM;
8000h	Table 1 to 28 Dynamic Object data in External RAM ;
9C00h	Table 29 to 60 RAM Image of Static EEPROM object data (8192 bytes).
BC00h	Table 61 Communication Queues;
	Table 61 to 64 Processor Stack in External RAM
C000h	Table 65 to 96 External EEPROM object data (8192 bytes)

Table 29 begins with 16 bytes of reserved data. The next 64 bytes contain the static allocation object data. The static data for the remaining allocated objects follows up to a maximum of 8 kbytes.

Memory Assignment - ASIC/2-7040 FW740A

The ASIC/2-7040 FW740A uses 32 k RAM and has changed the assignment of memory.

Absolute Page

	Table 0. Internal RAM;
0100h	Table 01 to 31 Utility Space
	Table 32 to 39 ROM Tables
2800h	Table 40 to 92 Dynamic Object data in External RAM (13,312 bytes);
5C00h	Table 92 Communication Queues;
	Table 93 to 95 Processor stack (768 bytes) in External RAM ;
6000h	Table 96 to 127 Image of Static EEPROM object data in External RAM;
8000h	Table 128 to 159 External EEPROM object data (8 kbyte)

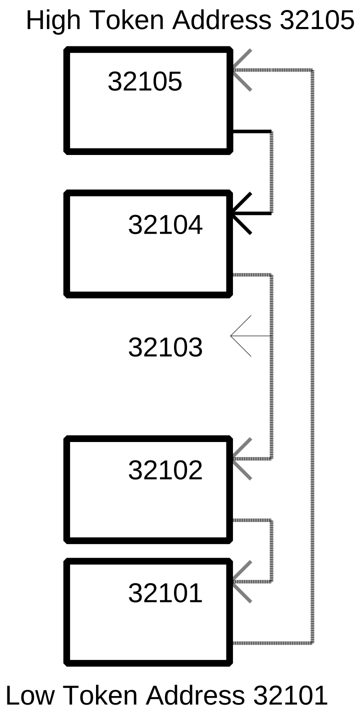
Table 96 begins with 16 bytes of reserved data. The next bytes contain the static allocation object data. The static data for the remaining allocated objects follows up to a maximum of 8 kbytes.

Allocation

A special table message is used to download the allocation to the controller. Table message 7Dh is used with Table number = ABh (171 decimal) and Entry CDh (205 decimal) followed by the Byte Count which is 2 bytes times the total number of objects. A word of data is then downloaded for the number of indices for each object starting with object 0, object 1, through the last object.

The combination lock is automatically destroyed when the allocation is downloaded with the 7Dh message. After waiting 60 seconds, power to the controller is turned off and on, resetting the controller. The combination lock can be restored by writing the value 10 decimal to Object 1-SYSTEM, Index 0, Attribute 4. From the allocation object one can determine the location, page and offset in memory where each object starts. Such messages are for Diagnostic purposes only.

Group 11: Token Passing Messages



Controllers that participate in the token passing process must have a System Bus Address in the range 32001 to 32255 and the Token Enable must be set to Yes. Furthermore the address should be within the High Token Address and Low Token Address range.

Each controller which has Token Enable acknowledges the token passing message when it is addressed to it and then passes the token to the next lower addressed controller until the Low Token Address is reached. The token is then passed to the controller with the High Token Address. The decision to pass the token is made after transmitting all remote point or time messages or on the expiration of the Token Hold Timer.

Upon passing the token to the next controller, the controller listens for the next controller to acknowledge that the token was successfully passed. If it hears a valid response, it assumes that the token has been successfully passed and reverts to listening mode.

If it does not hear a valid response, it assumes the token message was garbled and reissues the Pass Token message for a total of 3 times. After three failures, the controller then issues a Pass Token message to the controller with an address one lower until the Token Low Address fails to respond. It will then begin with Token High Address and continue until it reaches its own address. If unable to successfully pass the token, it then drops the token and goes into a listening mode.

Note: For best response of the token network it is important that token devices are addressed in sequence. Missing device addresses between the Low Token Address and the High Token Address cause unnecessary attempts to pass the token to non-existent controllers.

80h Pass Token

Permission to transmit is passed to the Next Station when there are no more messages to transmit, or the token holding time has been exceeded.

Destination Address:	Next Station (HI,LO)
Source Address:	This Station (HI,LO)
Message Type:	80 hex
Authority:	77 hex
Message body:	None
Response:	ACK

Fault Control

If the token holder hears any other message traffic on the communication line, it immediately drops the token and goes to listening mode. If the token is lost, so that it is no longer being passed from controller to controller, each controller listens to the system bus until its Token Lost Timer times out. The first controller to time out claims the token and begins transmitting. The Token Lost Time is calculated based on the System Bus Address, so that the controller with the highest address times out first.

Token Hold Interval

The Token Hold Interval is the maximum time that a controller can hold the token and transmit data on the system bus. On receiving the token the controller broadcasts time if needed, and then any Remote Points that are ready. When finished transmitting it passes the token. When the Token Hold Timer times out the token is passed to the next controller immediately and pending remote points must wait until the next token round.

Group 12: Object Messages

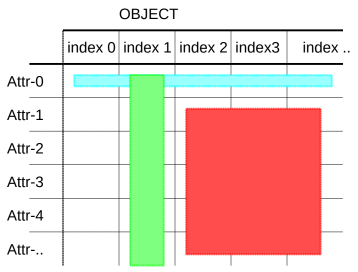
A family of (9Xh) object messages are implemented specifically to meet the data handling requirements of ASIC/2 Configurable Controllers and SINC/2 System Interface and Network Controllers. Data structures in the configurable controller are treated as objects.

Data objects, OBJECT(attr,index), are represented as two dimensional data tables which can be written to or read back using Message Types 90h and 91h. Each data record has fields which correspond to specifically defined attributes. To use memory efficiently, more than one parameter may be packed into a single attribute. Each attribute, corresponds to a data field. The first attribute is typically used for the present value. All data attributes for a given object are of the same data type: byte, or word.

Two Data Types are implemented:

1 - Single byte data
2 - Word (2 byte) data

Multiple instances with the identical structure are identified by an index number, 0 to 255. For example, the INPUT object can have 16 indexes, each with the same attributes including present value, previous value, hi and low alarm values, etc.



Each request message consists of message header bytes, message overhead bytes, a data block if required, and a check sum. The header contains the destination and source addresses and message type. The overhead bytes contain the type of data, byte or word, the object number, starting index, starting attribute, number of indices and the number of attributes. Multiple attributes of an index, multiple indexes of a single attribute, or multiple indices of multiple attributes may be requested. The data block in a single message is limited to 64 bytes.

Each response message consists of response header bytes, response overhead bytes, a data block if required, and a check sum.

The data is returned in the following order: first index, all attributes; next index, all attributes; etc. If the requested data block exceeds 64 bytes, the data will be truncated at the last complete index that does not overrun the buffer. Because of the limited data block size it is usually most efficient to request multiple attributes on an index by index basis.

Several errors are possible. A request for multiple attributes from a single index may find fewer attributes than have been requested. A request from a single index may exceed 64 data bytes. A request for multiple attributes from multiple indices may find fewer indices or attributes than have been requested. For all of these errors the return byte count will be set at zero and no data will be returned.

The interpretation of the data will depend on the specific data object and application. Data words will be transmitted low order byte first (LO,HI) following Intel standard usage.

Note: Only the Source and Destination Addresses are transmitted hi byte first (HI,LO) following ASI communication protocol practice.

90h Set Object Data Message

The set object value, 90h, request message has 7 required message header bytes: STX, DST1, DST2, SRC1, SRC2, MT, and FR.

The set object request message has 8 additional required message overhead bytes. The message overhead specifies the data type, the data object number to write the data to, the starting index , the starting attribute , the number of indexes, the number of attributes, the number of data bytes in the data block, followed by a second framing byte. The number of data bytes transmitted is used by the receiving controller to determine where to find the checksum. The number of data bytes does NOT count the second framing byte M8 = 77 hex which is used to synchronize transmission.

The data bytes are then transmitted followed by a checksum.

The checksum is the arithmetic sum (modulo 256) of all message bytes including the initial STX and through the last data byte.

90h Set Object Data - Request

The Configurable Controller message type 90h is used to set data object values.

Message Header:

STX	= 02 hex - start of message
DST1	= Destination Address ADDR_A HI
DST2	= Destination Address ADDR_A LO
SRC1	= Source Address ADDR_B HI
SRC2	= Source Address ADDR_B LO
MT	= 90 hex - Message Type
FR	= 77 hex - Framing Byte

Message Overhead:

M1 = Data Type

1 - Single byte table
2 - Word (two byte table)

M2 = Object Number
M3 = Starting Index
M4 = Starting Attribute
M5 = Number of Indices
M6 = Number of Attributes
M7 = Data Block (Number of Data Bytes to be transmitted.)
M8 = 77 hex - Framing Byte

Data Block:

M9 ... M72 = Data bytes (limited to 64 bytes)

Message Checksum:

CHK = Checksum

90h Set Object Data - Acknowledge

The set object, 90h, acknowledge response message has 7 response header bytes: STX, DST1, DST2, SRC1, SRC2, ACK, MT. The response header echoes the message type following the acknowledge byte. NOTE: ASI Protocol messages which have a simple acknowledgement will place the checksum byte immediately following the ACK. If the controller is expecting a response to a 90h message, then 90 hex appears following the ACK.

The set object, 90h, response message has 7 additional required overhead bytes. The response overhead echoes the request and returns the number of indices, attributes, and data bytes actually accepted. The data type requested must agree with data type for table, or else the data will not be accepted and the number of data bytes actually set will be returned as 0. The actual data type for the object in question will be returned with the response.

The set object, 90h, response message has no data bytes. The response overhead is followed by a checksum. The checksum is the arithmetic sum (modulo 256) of all message bytes including the initial STX and through the last response overhead byte.

Response Header:

STX = 02 hex - start of message
DST1 = Destination Address ADDR_B HI
DST2 = Destination Address ADDR_B LO
SRC1 = Source Address ADDR_A HI
SRC2 = Source Address ADDR_A LO
ACK = 06 hex - Acknowledge Byte

Response Overhead:

M1 = 90 hex - Response Message Type
M2 = Data Type (actual)
M3 = Object Number
M4 = Starting Index
M5 = Starting Attribute
M6 = Number of Indices (actual)
M7 = Number of Attributes (actual)
M8 = Data Block (Number of Data Bytes actually written.)

Response Checksum:

CHK = Checksum

91h Get Object Data Message

The Configurable Controllermessage type 91h is used to get data object values. The message specifies the data type, the object number to get the data from, the starting index, the starting attribute, the number of indices, the number of attributes, followed by the checksum.

The get object, 91h, request message has 7 required message header bytes: STX, DST1, DST2, SRC1, SRC2, MT, and FR.

The get object, 91h, request message has 7 additional required message overhead bytes. The message overhead specifies the data type, the data object number to write the data to, the starting index, the starting attribute, the number of indices, the number of attributes, followed by the number of data bytes in the data block to be returned. The number of data bytes requested is used by the receiving controller to verify that the data request is valid.

The request overhead is followed by a checksum. The checksum is the arithmetic sum (modulo 256) of all message bytes including the initial STX and through the last request overhead byte.

91h Get Object Data - Request

Message Header:

STX = 02 hex - start of message
DST1 = Destination Address ADDR_A HI
DST2 = Destination Address ADDR_A LO
SRC1 = Source Address ADDR_B HI
SRC2 = Source Address ADDR_B LO
MT = 91 hex - Message Type
FR = 77 hex - Framing Byte

Message Overhead:

M1 = Data Type
- 1 - Single byte object
- 2 - Word (two byte object)

M2 = Object Number
M3 = Starting Index
M4 = Starting Attribute
M5 = Number of Indices
M6 = Number of Attributes
M7 = Data Block (Number of Data Bytes to be returned.)

Message Checksum:

CHK = Checksum

91h Get Object data - Acknowledge and Response

The get object, 91h, response message has 7 additional required response header bytes: STX, DST1, DST2, SRC1, SRC2, ACK, MT. The response header echoes the message type following the acknowledge byte. NOTE: ASI Protocol messages which have a simple acknowledgement will place the checksum immediately following the ACK (06 hex). If the controller is expecting a response to a 91h message, then 91 hex appears following the ACK.

The get object, 91h, response message has 7 required overhead bytes. The response overhead echoes the request and returns the number of data bytes actually to be transmitted. The data type requested must agree with data type for table, or else the data will not be returned and the number of data bytes actually sent will be returned as 0. The actual data type for the object in question will be returned with the response. The number of data bytes does NOT count the second framing byte M8 = 77 hex which is used to synchronize transmission.

The data bytes are then transmitted followed by a checksum. The checksum is the arithmetic sum modulo 256 of all message bytes including the initial STX and through the last data byte.

Response Header:

STX = 02 hex - start of message
DST1 = Destination Address ADDR_B HI
DST2 = Destination Address ADDR_B LO
SRC1 = Source Address ADDR_A HI
SRC2 = Source Address ADDR_A LO
ACK = 06 hex - Acknowledge Byte

Response Overhead:

M1 = 91 hex - Response Message Type
M2 = Data Type (actual)
M3 = Object Number
M4 = Starting Index
M5 = Starting Attribute
M6 = Number of Indices (actual)
M7 = Number of Attributes (actual)
M8 = Data Block Size (Number of Data Bytes actually returned.)
M9 = 77 hex - Framing Byte

Response Data Block:

M10 ... M73= Data bytes
(limited to 64 table data bytes)

Response Checksum:

CHK = Checksum

Firmware Release History

ASIC/2-8040 Configurable Controller (FW 840)

ASIC/2-8040 FW840A Rev. 1.0 22 March1996

ASIC/2-7040 Configurable Controller (FW 740)

ASIC/2-7040 FW740A Rev. 1.0 29 March1994

ASIC/2-7000 Configurable Controller (FW 700)

ASIC/2-7000 FW700I Rev. 1.0 Released 10 June 1993

1) Firmware to accommodate 32 kbyte dynamic memory
2) Object 26 Counter
3) Object 27 Static Trend
4) Object 28 Event Manager
5) Object 29 Event Log

ASIC/2-7000 FW700H Rev. 1.0 Released 21 April 1993

Firmware to accommodate 32 kbyte dynamic memory

ASIC/2-7000 FW700G Rev. 1.0 Released 16 March 1993

Modified Firmware for support of DAK-002 FW700B including

ASIC/2-7000 FW700F Rev. 1.0 Preliminary 08 Oct 1992

90h message will write to EE with Combo-Lock No Good.

ASIC/2-7000 1.0 FW700A 5 Dec 1991

Revised for ASIC/2-7000 hardware

SC/1-9XXX System Controller(FW907)

SC/1 Ver. 1.0 FW 907E, 3 Dec 1992

RAM 2558 Bytes, EEPROM 4080 Bytes

SC/1 Ver. 1.0 FW 907C, 21 April 1992

Added Message type 68H and 69h to local pass-through

SC/1 Ver. 1.0 FW 907A, 25 July 1991

First release of System Controller.