CIP Interface

Technical Manual

A16760300A
02-05

TRADEMARKS

METTLER TOLEDO® and JAGXTREME® are registered trademarks of Mettler-Toledo, Inc. All other brand or product names are trademarks or registered trademarks of their respective companies.

NOTICE

This document is associated with an agency-approved product. No changes tot his document are permitted without agency approval.

FCC NOTICE

This device complies with Part 15 of the FCC Rules and the Radio Interference Requirements of the Canadian Department of Communications. Operation is subject to the following conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his or her own expense.

ORDERING INFORMATION

It is most important that the correct part number is used when ordering parts. Parts orders are machine processed, using only the part number and quantity as shown on the order. Orders are not edited to determine if the part number and description agree.

COPYRIGHTS

CUSTOMER FEEDBACK

Your feedback is important to us! If you have a problem with this product or its documentation, or a suggestion on how we can serve you better, please fill out and send this form to us. Or, send your feedback via email to: quality_feedback.mtwt@mt.com. If you are in the United States, you can mail this postpaid form to the address on the reverse side or fax it to (614) 438-4355. If you are outside the United States, please apply the appropriate amount of postage before mailing.

Your Name:	Date:
Organization Name:	METTLER TOLEDO Order Number
Address:	Part / Product Name:
	Part / Model Number:
	Serial Number:
Phone Number: ( ) Fax Number: ( )	Company Name of Installation:
E-mail Address:	Contact Name:
	Phone Number:

How well did this product meet your expectations in its intended use?		Comments:
☐	Met and exceeded my needs	
☐	Met all needs	
☐	Met most needs	
☐	Met some needs	
☐	Did not meet my needs	

PROBLEM:

UNACCEPTABLE DELIVERY:

- ☐ Shipped late
- ☐ Shipped early
- ☐ Shipped to incorrect location
- ☐ Other (Please Specify)

OUT OF BOX ERROR:

- ☐ Wrong item
- ☐ Wrong part
- ☐ Missing equipment
- ☐ Equipment failure

- ☐ Wrong documentation
- ☐ Missing documentation
- ☐ Incorrectly calibrated
- ☐ Other (Please specify)

Comments:

DO NOT WRITE IN SPACE BELOW; FOR METTLER TOLEDO USE ONLY

☐ Retail ☐ Light Industrial ☐ Heavy Industrial ☐ Systems

RESPONSE: Include Root Cause Analysis and Corrective Action Taken.

FOLD THIS FLAP FIRST

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 414 COLUMBUS, OH

POSTAGE WILL BE PAID BY ADDRESSEE

Mettler-Toledo, Inc.
Quality Manager - MTWI
P.O. Box 1705
Columbus, OH 43240
USA

Please seal with tape.

DECLARATION OF CONFORMITY

Konformitätserklärung
Déclaration de conformité
Declaración de Conformidad
Conformiteitsverklaring
Dichiarazione di conformità

We/Wir/Nous/Wij/Noi: **Mettler-Toledo, Inc.**
1150 Dearborn Drive
Worthington, Ohio 43085
USA

declare under our sole responsibility that the product,
erklären, in alleiniger Verantwortung, daß dieses Produkt,
déclarons sous notre seule responsabilité que le produit,
declaramos, bajo nuestra sola responsabilidad, que el producto,
verklaren onder onze verantwoordelijkheid, dat het product,
dichiariamo sotto nostra unica responsabilitá, che il prodotto,

Model/Type: **Jaguar and JagXtreme**

to which this declaration relates is in conformity with the following standard(s) or other normative document(s).
auf das sich diese Erklärung bezieht, mitder/den folgenden Norm(en) oder Richtlinie(n) übereinstimmt.
Auquel se réfère cette déclaration est conforme à la (aux) norme(s) ou au(x) document(s) normatif(s).
Al que se refiere esta declaración es conforme a la(s) norma(s) u otro(s) documento(s) normativo(s).
Waarnaar deze verklaring verwijst, aan de volende norm(en) of richtlijn(en) beantwoordt.
A cui si riferisce questa dichiarazione è conforme alla/e sequente/i norma/e o documento/i normativo/i.

Council directive on the harmonization of the laws of the Member states:	**standards:**	**Certificate number (if applicable)**
relating to non-automatic weighing instruments (90/384/EEC) amended by directive (93/68/EEC)	EN 45501:1992 Article 1.2.a	TC 2618
relating to electromagnetic compatibility (89/336/EEC) amended by directive (93/68/EEC; 92/31/EEC)	EN 55022, B	
relating to electrical equipment designed for use within certain voltage limits (73/23/EEC amended by directive (93/68/EEC)	EN 60950	
Relating to electrical equipment designed for use in potentially explosive atmospheres (94/9/EC) (Refer to note 1)	EN 50021 : 1999 EN 50281-1-1 : 1998	KEMA 02ATEX1023 X (Refer to note 1)

Worthington, Ohio USA, April, 2003 Mettler-Toledo, Inc.

Darrell Flocken, Manager - Weights & Measures
Office of Weights and Measures

Notes:

1. Certificate KEMA 02ATEX1023 X applies only to JagXtreme units only. Refer to Section (17) of the certificate for special conditions.

Original issue:	July, 1995	
Revised:	October, 1996	added compliance to Low Voltage Directive
	May, 2000	added JagXtreme
	April, 2003	added compliance to ATEX Directive.

Table of Contents

CIP Communications with the JAGXTREME Terminal

Overview

The JAGXTREME ControlNet interface card enables a JAGXTREME terminal to communicate to Control and Information Protocol (CIP) networks, ControlNet, and Ethernet IP (industrial protocol). CIP networks are supported by Rockwell Automation, Honeywell, ABB, and Mitsubishi process controllers.

Note: The CIP network ControlNet and Ethernet IP connections are mutually exclusive (only one connection may be configured at a time) on the JAGXTREME terminal.

The JAGXTREME ControlNet/Ethernet IP interface is classified as a CIP Adapter Class device that supports the following types of CIP messaging:

- Class 1 scheduled messaging (I/O messaging)
- Unscheduled messaging
- Unconnected messaging
- Class 3 unscheduled messaging

Common Definitions

The following sections assume a certain familiarity on the part of the user with the CIP protocol. Users who are unfamiliar with CIP may wish to download the CIP Common Specification from the Open Device Vendor's Association (ODVA) web site at: http://www.odva.org/. Chapter 1 / Volume 1 of that document (Introduction to CIP) provides an excellent user level overview of the CIP protocol. Common definitions that are used in this manual are adapted from the ODVA "Ethernet/IP Terms and Definitions":

Adapter Class	An Adapter Class product emulates functions provided by traditional rack-adapter products. This type of node exchanges real-time I/O data with a Scanner Class product. It does not initiate connections on its own.
Scanner Class	A Scanner Class product exchanges real-time I/O data with Adapter Class and Scanner Class products. This type of node can respond to connection requests and can also initiate connections on its own.
I/O Client	Function that uses the I/O messaging services of another (I/O Server) device to perform a task. Initiates a request for an I/O message to the server module. The I/O Client is a Connection Originator
I/O Server	Function that provides I/O messaging services to another (I/O Client) device. Responds to a request from the I/O Client. I/O Server is the target of the connection request.
Message Client	Function that uses the Explicit messaging services of another (Message Server) device to perform a task. Initiates an Explicit message request to the server device.
Message Server	Function that provides Explicit messaging services to another (Message Client) device. Responds to a Explicit message request from the Message Client.
Target	Destination for I/O connection or message requests. Can only respond to a request, cannot initiate an I/O connection or message.

Connection Originator

Source for I/O connection or message requests. Initiates an I/O connection or explicit message connection.

Implicit Messaging Implicit Messages are exchanged across I/O Connections with an associated Connection ID. The Connection ID defines the meaning of the data and establishes the regular/repeated transport rate and the transport class. No messaging protocol is contained within the message data as with Explicit Messaging. Implicit Messages can be point-to-point or multicast and are used to transmit application-specific I/O data. This term is used interchangeably with the term I/O Messaging.

Explicit Messaging Explicit Messages can be sent as a connected or unconnected message. CIP defines an Explicit Messaging protocol that states the meaning of the message. This messaging protocol is contained in the message data. Explicit Messages provide a one-time transport of a data item. Explicit Messaging provide the means by which typical request/response oriented functions are performed (e.g. module configuration). These messages are typically point-to-point.

I/O Messaging Used interchangeably with the term Implicit Messaging.

Unconnected Messaging

Provides a means for a node to send message requests without establishing a connection prior to data transfer. More overhead is contained within each message and the message is not guaranteed destination node resources. Unconnected Messaging is used for non-periodic requests (i.e. network "Who" function). Explicit messages only.

Connected Messaging

A connection is a relationship between two or more application objects on different nodes. The connection establishes a virtual circuit between end points for transfer of data. Node resources are reserved in advance of data transfer and are dedicated and always available. Connected messaging reduces data handling of messages in the node. Connected messages can be Implicit or Explicit.

Class 1 Messaging

In ControlNet communication protocol scheduled (cyclic) message transfer between a PLC and CIP Adapter Class device.

Class 3 Messaging

In ControlNet communication protocol unscheduled (cyclic) message transfer between a PLC and CIP Adapter Class device.

ControlNet Messaging

The ControlNet Information Protocol is the ControlNet logical message layer protocol. It operates over either the ControlNet physical layer protocol or the TCP/IP/Ethernet protocol. The JagXtreme supports two types of application-level messaging: Class 1 is Cyclic (Scheduled) Messaging, and Class 3 is Unscheduled Data Messaging.

Class 1 - Cyclic (Scheduled) Messaging

Class 1 or Cyclic Messaging is "scheduled" messaging that occurs at regular, "deterministic" timed intervals. Typically, the ControlNet data-link protocol guarantees that scheduled messages have a fixed time slot on the LAN that is always available and cannot be pre-empted. A Controller can access the Class 1 data without specific Ladder program instructions. Process Control systems typically use the cyclic messaging for transmitting real-time process data.

Cyclic messaging is always "connected" messaging. Once the connection is open, it is long-lived. Cyclic messages are continually sent across the open connection without the overhead of re-establishing the connection

From the user's perspective, scheduled messaging can be utilized to quickly and automatically transfer data between the Logix controller and the JAGXTREME terminal. This type of messaging happens at a user selectable interval in a background task of the Logix controller and is consequently asynchronous to the execution of its ladder program.

The data that is transferred between the JAGXTREME and controller using scheduled messaging is defined when the JAGXTREME ControlNet interface card is set up. This data is identified as an "Assembly Object" and consists of data for four scales. This data is arranged in "Slots"-one slot for each scale. The setup of the interface card also defines the type of scale numerical data(Gross Weight, Tare Weight, etc.) will be transferred. The types are:

Weight (Integer)
Divisions (Integer)
Extended
Floating Point.

The format of the "Slots" is described in the "Assembly Object" section of this manual.

Scheduled messaging is set up by the user within the Logix controller program at design time, and is initiated by that controller at run time. Scheduled messaging between the Logix controller and the JAGXTREME is bi-directional and takes place over a single Class 1 (Scheduled) CIP connection between the controller and one pair of the JAGXTREME terminal's Assembly Object Instances.

Notes:

- Although the JAGXTREME terminal provides six assembly object instances (3 pairs), only two Instances (one pair) are ever active at a given time, depending on how the user has configured the JAGXTREME terminal.
- Scheduled messaging is limited to JAGXTREME terminals that are physically connected to the CIP network via a JAGXTREME ControlNet interface card. Scheduled messaging to JAGXTREME terminals that are connected to a cluster network is not supported.
- A scale in the cluster network may be selected as part of the scheduled message.
- Although the Requested Packet Interval (RPI) for scheduled messaging can be set to whatever the user desires, the JAGXTREME terminal only updates its CIP Network Interface Card with weight, rate, and scale status data at the rate of 17 Hz.

Class 3 –Unscheduled Data Messaging

The controller uses Unscheduled Messaging to access the "Shared Data" memory of the JAGXTREME terminals.

The JAGXTREME terminal supports unscheduled messaging to any of the objects defined in its device profile. Such messaging is driven by message instructions located within the controller's ladder program. Whenever commanded by the controller, the controller sends simple discrete messages to the JAGXTREME terminal specifying a Class, Instance, Attribute (if applicable) and Service Code to be executed.

ControlNet sends unscheduled messages using "non-deterministic" scheduling. That is, the ControlNet protocol sends these messages in the next available time slot, rather than on a fixed schedule.

Unscheduled messaging may be either Connected or Unconnected at the discretion of the Controller and based on the architecture of the network. .

The Controller can only use unconnected messaging for routing Shared Data messages from a clustered JAGXTREME terminal through a JAGXTREME bridge terminal.

To implement a request/response message exchange using Shared Data Messaging, the Controller Ladder program must use two MESSAGE instructions. The Controller Ladder program first uses the MESSAGE "Set Attributes" instruction to write the request message, and then it uses a separate MESSAGE "Get Attributes" instruction to read the response message.

Note: The JAGXTREME is a CIP Adapter Class device. It responds to unscheduled messages, but it cannot originate them.

Supported Controller Types

The JAGXTREME terminal supports CIP messaging from the controller types described in this section.

Logix Controllers

Rockwell Logix 5000 and ProcessLogix DCS controllers can support either scheduled or unscheduled messaging to a JAGXTREME terminal over either ControlNet or Ethernet/IP.

Note: Logix controllers use different names for a variety of the data types specified within this document as JAGXTREME attributes. In addition, they do not support every data type used by the JAGXTREME terminal.

The following table provides a useful mapping between the JAGXTREME terminal's standard data types and those utilized by Rockwell Automation controllers.

JAGXTREME Data Type	Equivalent Logix 5000 Data Type	Comments
ASCII[*x*]	SINT	Format the SINT as ASCII in Logix 5000
BIT	BOOL	
BOOL8	SINT	Value will be 0 or 1.
CHAR	SINT	Format the SINT as ASCII in Logix 5000
DOUBLE	N/A	Not supported by Logix 5000 Controllers
FLOAT	REAL	32 Bit Logic 5000 Format
INT8	SINT	
INT16	INT	
INT32	DINT	
UINT8	N/A	Use SINT, but beware of sign bit
UINT16	N/A	Use INT, but beware of sign bit
UINT32	N/A	Use DINT, but beware of sign bit

PLC 5

Rockwell Automation PLC 5 controllers (Series F, Revision C.1, or later) can support either I/O or unscheduled messaging to a JAGXTREME terminal over ControlNet only. Although many PLC 5 controllers do have Ethernet ports, they do not use the Ethernet/IP protocol.

Other Controllers

The JAGXTREME terminal supports scheduled and unscheduled messaging (within the constraints of this document) from any standard ControlNet or Ethernet/IP scanner or message originator.

Network Communication Terminology

The following terminology and acronyms are used in this document.

Term	Meaning
PLC-5	Rockwell Automation PLC-5 Controller
L5K	Rockwell Automation Logix 5000 Controller
CIP	Control and Information Protocol
ControlNet	Open Device Vendor's Association's ControlNet Industrial Protocol
Ethernet/IP	Open Device Vendor's Association's Ethernet Industrial Protocol

1

JAGXTREME Data Type Definitions

The following data types are referred to throughout this document.

Term	Meaning
ASCII[*x*]	Null terminated ASCII string of length *x* (including the null terminator)
BIT	Single bit value
BOOL8	8 Bit Boolean value: 0 = False / No, 1 = True / Yes
CHAR	Single 8 bit ASCII character
DOUBLE	Double Precision (64 Bit) Floating Point Value
FLOAT	IEEE 32 bit floating point value
INT8	Signed 8 bit integer
INT16	Signed 16 bit integer
INT32	Signed 32 bit integer
UINT8	Unsigned 8 bit integer
UINT16	Unsigned 16 bit integer
UINT32	Unsigned 32 bit integer

References:

Ethernet/IP Specification Rel 1.0, June 5, 2001, ControlNet International

ControlNet Specification Release 2.0, March 31, 1998, ControlNet International

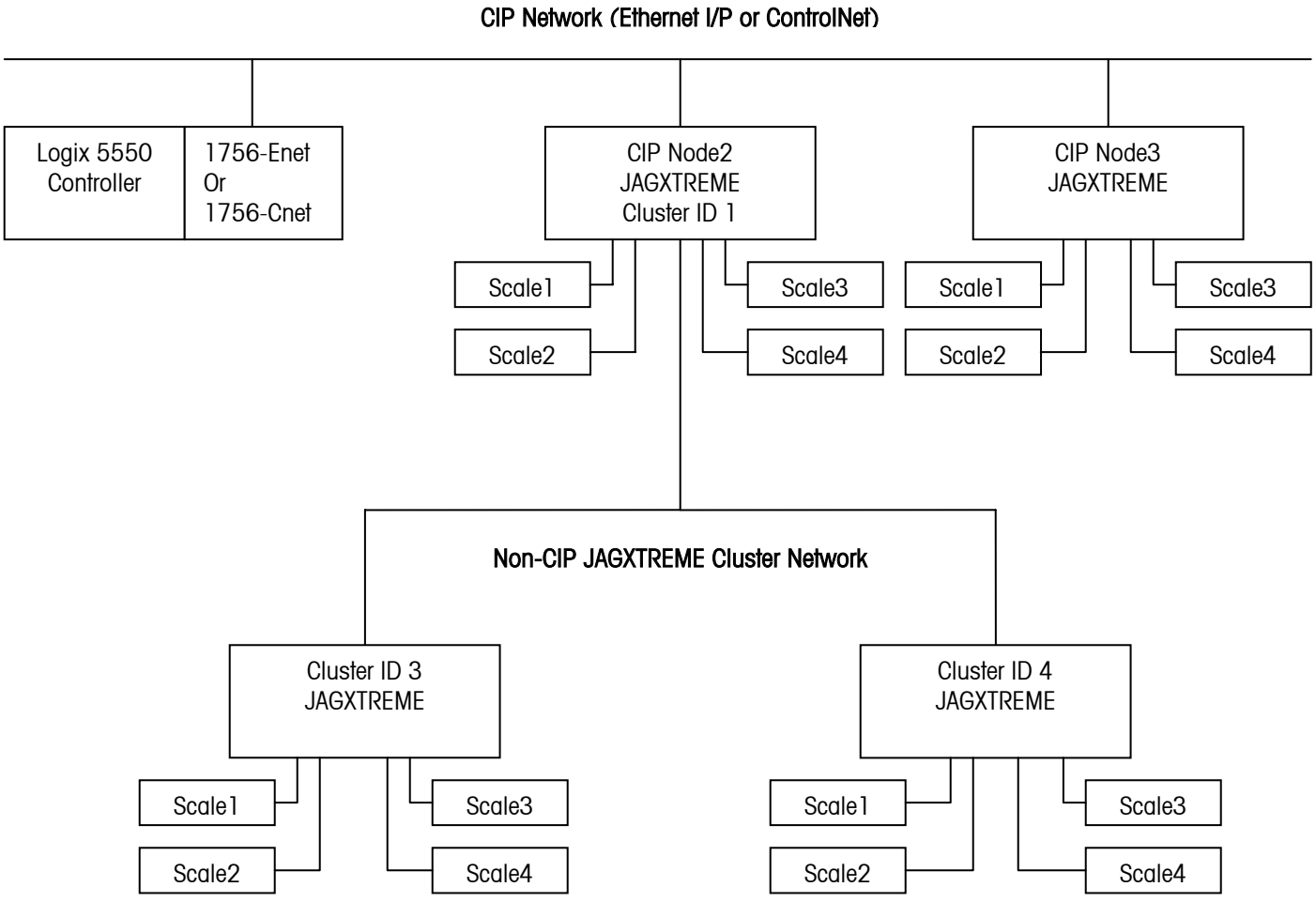
Logix5000 Data Access, 1756-RM005A-EN-E, Rockwell Automation, March 2000

JAGXTREME Technical Manual, C15896200A, METTLER TOLEDO,

Network Topology

A complete JAGXTREME system may be comprised of a variety of network topologies, from the most basic to the extremely complex. Logix controllers can communicate with several JAGXTREME terminals either directly via a CIP network or, through the use of bridging, indirectly via a non-CIP based JAGXTREME cluster network.

The following example represents a network topology in which a Logix controller can communicate with several JAGXTREME terminals both directly and indirectly through the use of bridging.



Example of a Network Topology

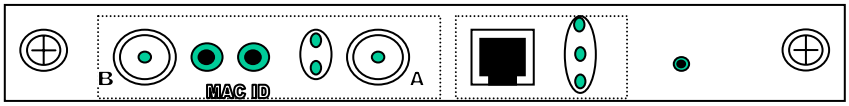
Network Topology Considerations

The network topology represented in the previous figure highlights some of the communications abilities of the JAGXTREME terminal. Naturally, your own network topology will be dictated by your specific circumstances and needs. In designing your network, however, there are a few key points to keep in mind:

- Each JAGXTREME terminal can support a maximum of four directly or indirectly connected individual scales.
- A JAGXTREME terminal can act as a bridge to other JAGXTREME terminals connected to it via a non-CIP JAGXTREME cluster network. This cluster network is limited to six JAGXTREME terminals. In the Network Topology shown, the JAGXTREME at CIP node 2 is acting as a bridge to the JAGXTREMES at Cluster IPs 3 and 4.
- Each JAGXTREME cluster network can contain a maximum of 24 scales.
- Any number of Logix controllers can reside on the CIP network and any of those controllers can communicate either directly with any JAGXTREME terminal on the CIP network, or indirectly, using unscheduled messaging, with any JAGXTREME terminal on the non-CIP JAGXTREME cluster network.
- Each JAGXTREME terminal on the CIP network may also communicate with one (and only one) Logix controller on that network via scheduled messaging. Scheduled messaging is only supported by JAGXTREMES that are directly connected to the network via a JAGXTREME ControlNet interface card.
- In the Network Topology diagram shown, the Controller may communicate via scheduled and unscheduled messaging with the JAGXTREMES directly connected to the NETWORK at CIP Nodes 2 and 3. The Logix Controller may only communicate via unscheduled messaging with the JAGXTREMES at Cluster IPs 3 and 4.
- In the Network Topology shown, any one of the scales connected to the clustered JAGXTREMES may be configured to be part of the scheduled message setup at the JAGXTREME bridge terminal.
- A scheduled message may only be configured to handle four scales

ControlNet Interface Card Setup and Use

Setting up the JAGXTREME Interface



Field Connections

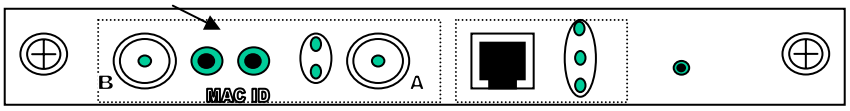
The physical media supported for ControlNet is a redundant Bayonet Nut Connector BNC connector. Field connections to the interface are via an RG-6 quad shielded coaxial cable.

The physical media supported for Ethernet IP is an RJ-45 connector. Field connections to the interface are via twisted pair Ethernet (10baseT), which is also referred to as "UTP" (for unshielded twisted pair).

Network Addressing

Addressing for the ControlNet interface is accomplished through a ControlNet Media Access Control (MAC address). A MAC address represents one physical node and is set through a set of rotary switches located on the ControlNet interface card. The address is chosen by the system designer, and then configured on each JAGXTREME terminal's ControlNet card using a small bladed screwdriver. Once configured, the MAC address or MAC ID can be viewed from the JAGXTREME terminal's setup screens in the ControlNet sub block or from the embedded web pages on the JAGXTREME terminal's communication page.

MAC ID Address



Note: Addressing for the Ethernet IP address is similar to Ethernet TCP/IP addressing.

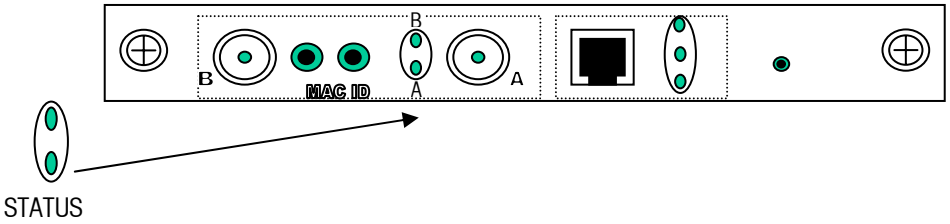
Each Ethernet IP address represents one physical node. The address is defined by the IP Address, the Subnet Mask and the Gateway Address. The JAGXTREME terminal's setup capabilities, either from the front panel or from the web pages, allow selection of the Ethernet IP address, Subnet, and Gateway. (This includes the ControlNet sub block section if using the front panel setup or the Communication page if using setup via the web pages.) The addresses are typically chosen by the system designer, and then configured on each JAGXTREME terminal.

Note: Slot addressing of the JAGXTREME ControlNet interface board is fixed.

Network Status LEDs

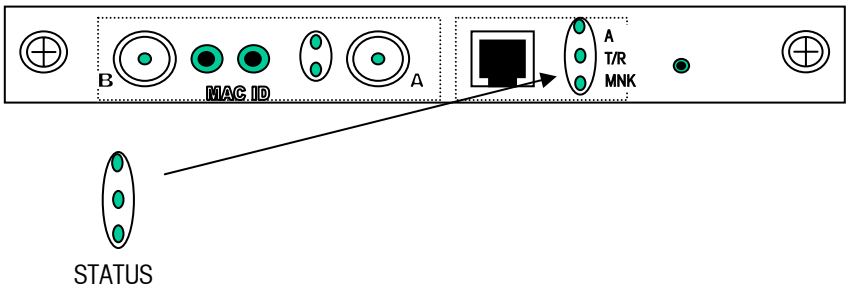
The status lights on the ControlNet interface card give you information about the card and the CIP network when you are connected. There are two sets of status lights, one for ControlNet and one for Ethernet IP. The functions of the lights are explained below.

ControlNet



A and B LED's	Cause	Action
Off	No power	None or power up
Steady Red	Faulted unit	Cycle power
Alternating Red/Green	Self-test	None
Alternating Red/Off	Incorrect node configuration Media fault	Check network address and other ControlNet configuration parameters Check media for broken cables, loose connectors missing terminators, etc
Steady Green	Normal operation	None
Flashing Green/Off	Temporary errors	None, node will self correct
Flashing Red/Off	Media fault	Check media for broken cables, loose connectors missing terminators, etc
Flashing Red/Green	Incorrect network configuration	Cycle power, check setup of JAGXTREME ControlNet card

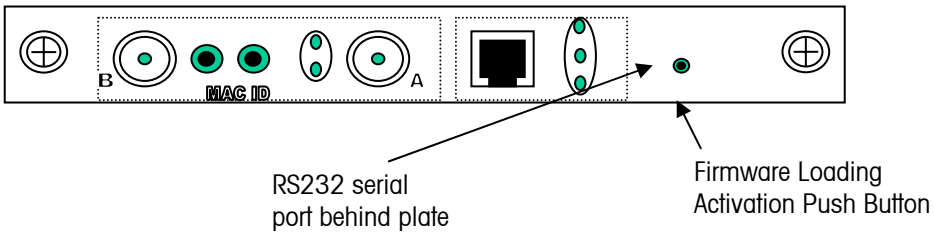
Ethernet IP



LED	Off	On
A	Interface card not active. Check power, reset JAGXTREME.	Normal Operation
T/R	No network activity. Check network activity, check network addressing.	Normal operation
LINK	Bad media. Check media connection.	Normal Operation

Firmware Loading

The JAGXTREME terminal's ControlNet interface card is a flash-based technology, enabling field upgrades of the interface cards firmware. To accomplish the firmware loading a special RS232 interface port has been added to the board as well as a "firmware loading" activation push button.



The RS232 interface port can be accessed by removing the card, attaching a serial cable, and then reinstalling the card. Once the cable is in place, a PC running EFLASH or FLASHPRO can download the new firmware.

Pin 1	Pin 2	Pin 3
TXD	RXD	GND

Download Procedure

- Press and hold the "firmware loading" activation push button.
- Start the firmware download on the PC.
- Once the downloading has started, release the "firmware loading" activation push button and wait for the download to complete.
- Restart the JAGXTREME terminal, and set the ControlNet/Ethernet IP parameters as required.

NOTES

JAGXTREME CIP Object Interface

This section defines the specific CIP objects, instances, attributes and services supported by the JAGXTREME terminal with a CIP network interface card. Only the objects that are proprietary or in some way altered from their generic behavior are documented here. Standard CIP objects, required by the CIP specification, are documented in the Ethernet/IP or ControlNet specifications. Refer to the chart below when setting up the controller communication for Scheduled or Unscheduled Messaging.

Object Class	Assembly Instance	Messaging
Assembly (0x04 hex)	1,2,3,4,9	Scheduled / Unscheduled
Scale Weight (0x64 hex)	1-5	Unscheduled
Dynamic Data (0x65 hex)	1	Unscheduled
Scale Weight Static (0x66 hex)	1-5	Unscheduled
Scale Calibration (0x67 hex)	1-5	Unscheduled
Scale Tare (0x68 hex)	1-5	Unscheduled
Setpoint (0x69 hex)	1-12	Unscheduled
System (0x6A hex)	1	Unscheduled
User Literals (0x6B hex)	1	Unscheduled
User Prompts (0x6C hex)	1	Unscheduled
User Variables (0x6D hex)	1	Unscheduled
Cluster Variables (0x6E hex)	1	Unscheduled
Basic Application (0x70 hex)	1	Unscheduled
POWERCELL Log (0x71 hex)	1	Unscheduled
Scale Calibration EEPROM (0x72 hex)	1-5	Unscheduled
Shift Adjust (0x73 hex)	1-5	Unscheduled
Cell Calibration (0x74 hex)	1	Unscheduled
Level Sensitive Discrete Status (0x75 hex)	1	Unscheduled
Edge Sensitive Discrete Status (0x76 hex)	1	Unscheduled
Level Sensitive Physical Discrete I/O Output (0x77 hex)	1	Unscheduled
Level Sensitive Physical Discrete I/O Input (0x78 hex)	1	Unscheduled
Application Interface to Remote Batching (0x79 hex)	1-2	Unscheduled

Assembly Object

Class Code

Class Code: 04 hex

The Assembly Object acts as a repository for a variety of real-time scale weight data. The Assembly Object contains data for four scales. This data is accessible via Scheduled and Unscheduled messaging.

Class Attributes

Attr	Access	Name	Data Type	Default Value
0x01	Get	Object Revision	UINT16	2
0x02	Get	Max Instance	UINT16	6

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Class/Instance	Get Attribute Single
0x10	Instance	Set Attribute Single

Assembly Object Input and Output Instances

The JAGXTREME terminal supports six different assembly object instances in total. However, only two will be active at any given time depending on the data format selected when the CIP Network Interface Card was configured.

Selected Data Format	Input Instance	Output Instance
Weight or Divisions	1	2
Extended	9	2
Floating Point	3	4

Note:

The words 'input' or 'output' used to describe assembly instances are from the perspective of the Logix AC, not from the JAGXTREME terminal. Input instances are sent from the JAGXTREME terminal to the Logix AC. Output instances are sent from the Logix AC to the JAGXTREME terminal.

Assembly Object "Slots"

Because each JAGXTREME terminal can have up to four individual scales connected to it, its assembly instances take the form of arrays of scale data, with the data from each scale occupying one element, or "Slot" in that array. The assembly instance will always be an array of four slots of data.

Assembly Object Instance 1

Assembly Object Instance 1 represents scale weight data sent to the Logix AC by the JAGXTREME terminal when the user has configured the CIP Network Interface Card for a data format of Weight or Division.

Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
3	Get	Data	Integer Input Slots [4]	N/A
4	Get	Size	UINT16	16 (Bytes) or 8 (Int)

Structure of the Integer Input Slots

Each of the four integer input slots within assembly instance 1's data attribute have the following structure:

Name	Data Type	Description
Weight	INT16	Scale Weight Data
Setpoint 1	BIT	Setpoint 1
Setpoint 2	BIT	Setpoint 2
Setpoint 3	BIT	Setpoint 3
Setpoint 4	BIT	Setpoint 4
Setpoint 5	BIT	Setpoint 5
Setpoint 6	BIT	Setpoint 6
Setpoint 7	BIT	Setpoint 7
Setpoint 8	BIT	Setpoint 8
EscapeKey	BIT	Escape Key Pressed (See Note 1)
DiscreteIn 1	BIT	Discrete Input 1
DiscreteIn 2	BIT	Discrete Input 2
DiscreteIn 3	BIT	Discrete Input 3
Scale Motion	BIT	Scale Motion
NetMode	BIT	Net Weight Mode
Updating	BIT	Update In Progress
DataOk	BIT	Scale Data Ok
Total Length:	4 Bytes	

Notes:

Bit 8 is set to "1" only on the first controller input assembly slot per terminal involved in the up to four scale data. The bit is set when the ESC key is pressed on the JAGXTREME terminal and while in operator prompt mode.

Assembly Object Instance 2

Assembly Instance 2 represents data sent to the JAGXTREME terminal by the Logix AC when the user has configured the CIP Network Interface Card for a data format of either Weight or Division or Extended Weight.

Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
3	Get / Set	Data	Integer Output Slots [4]	N/A
4	Get	Size	UINT16	16 (Bytes) or 8 (Int)

Structure of the Integer Output Slots

Each of the four integer output slots within assembly instance 2's Integer Data attribute has the following format:

Name	Data Type	Description
Load Value	INT16	Preset Tare or Setpoint Value
SelectMode	BIT[3]	Weight Mode: 0 = Gross Weight 1 = Net Weight 2 = Displayed Weight 3 = Tare Weight 4 = Setpoint 1 5 = Rate 6,7 = Reserved
LoadTare	BIT	Load preset tare on 0 to 1 transition
ClearTare	BIT	Clear tare on 0 to 1 transition
TareScale	BIT	Tare scale on 0 to 1 transition
PrintWeight	BIT	Print scale weight on 0 to 1 transition
ZeroScale	BIT	Zero scale on 0 to 1 transition
EnblSetpoint	BIT	Enable Setpoints: 1 = Enable, 0 = Disable
DisplayMode	BIT[3]	Display Mode: 0 = Normal Display Mode 1 = Display Literal 1 2 = Display Literal 2 3 = Display Literal 3 4 = Display Literal 4 5 = Display Literal 5 6 = Reserved 7 = Display Literal from Block Transfer input
DiscreteOut1	BIT	Discrete Output 1
DiscreteOut2	BIT	Discrete Output 3
DiscreteOut3	BIT	Discrete Output 2
LoadSetpoint	BIT	Load Setpoint 1 Value
Total Length:	4 Bytes	

Assembly Object Instance

Assembly Instance 9 represents scale weight data sent to the Logix AC by the JAGXTREME when the user has configured the CIP Network Interface Card for a data format of Extended Weight.

Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
3	Get	Data	Extended Integer Input Slots [4]	N/A
4	Get	Size	UINT16	16 (Bytes) or 8 (Int)

Structure of Extended Integer Input Slots

Each of the four extended integer slots within Assembly Instance 9's Data attribute has the following format:

Name	Data Type	Description
WeightData	INT16	Preset Tare or Setpoint Value
ExtWeight	BIT[4]	Bits 17 – 20 of the Extended Weight
Sign	BIT	Sign of Extended Weight: 0 = Positive, 1 = Negative
Setpoint 1	BIT	Setpoint 1
Setpoint 2	BIT	Setpoint 2
Setpoint 3	BIT	Setpoint 3
EscapeKey	BIT	Escape Key Pressed (See Note 1)
DiscreteIn 1	BIT	Discrete Input 1
DiscreteIn 2	BIT	Discrete Input 2
DiscreteIn 3	BIT	Discrete Input 3
Scale Motion	BIT	Scale Motion
NetMode	BIT	Net Weight Mode
Updating	BIT	Update In Progress
DataOk	BIT	Scale Data Ok
Total Length:	4 Bytes	

Notes:

The EscapeKey bit is set to a 1 only on the first controller input assembly slot per terminal involved in the up to four scale data. The bit is set when the **ESC** key is pressed on the JAGXTREME terminal while in the operator prompt mode.

Assembly Object Instance 3

Assembly Instance 3 represents scale weight data sent to the Logix AC by the JAGXTREME when the user has configured the CIP Network Interface Card for a data format of Floating Point.

Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
3	Get	Data	Floating Point Input Slots [4]	N/A
4	Get	Size	UINT16	32 (Bytes) or 16 (Int)

Structure of Floating Point Input

Each of the four floating-point input slots within Assembly Instance 3's data attribute has the following format:

Name	Data Type	Description
Reserved	INT8	Reserved
FPIndicator	BIT[5]	Floating Point Indicator
Integrity	BIT	Data Integrity Bit[3]
CommandAck	BIT[2]	Command Acknowledge
FPValue	FLOAT	Floating Point Value (IEEE 32 bit format)
SP1Feeding	BIT	1st Setpoint Feeding
SP2Feeding	BIT	2nd Setpoint Feeding
SP1FastFeed	BIT	1st Setpoint Fast Feeding
SP2FastFeed	BIT	2nd Setpoint Fast Feeding
SpiTolerance	BIT	1st Setpoint In Tolerance
ScaleSelected	BIT	Scale Selected" (weight on local display)
JagBasic1	BIT	JagBASIC Custom Bit 1[2]
JagBasic2	BIT	JagBASIC Custom Bit 2[2]
EscapeKey	BIT	Escape Key Pressed [1]
DiscreteIn 1	BIT	Discrete Input 1
DiscreteIn 2	BIT	Discrete Input 2
DiscreteIn 3	BIT	Discrete Input 3
Motion	BIT	Scale Motion
NetMode	BIT	Net Weight Mode
Integrity1	BIT	Data Integrity Bit 2[5]
DataOk	BIT	Scale Data Ok[3]
Total Length:	8 Bytes	

Notes:

1. The EscapeKey bit is set to a 1 only on the first controller input assembly slot per terminal involved in the up to four scale data. The bit is set when the ESC key is pressed on the JAGXTREME terminal, while in the operator prompt mode.
2. There are two custom status bits that a Jagbasic application can use to communicate special statues to the controller. The JagBasic application and controller define the meaning of these bits.
3. The Data Ok indication bit reports scale overcapacity, scale under zero, scale communication, and cluster off-line error conditions. When the controller detects an error indication, it can send a command to the JAGXTREME terminal to get the latest error status.

4. The two Data Integrity Bits always have the same polarity, and they alternate polarity on each internal update.

Command Acknowledge Field

Whenever the JAGXTREME terminal processes a scale command it has received through the Floating Point Output Instance, it acknowledges the command by incrementing the value in the two-bit Command Acknowledge field. When the value in that field reaches 3, it wraps around again to 1.

The controller may use the counting sequence of the Command Acknowledge bits to verify that the JAGXTREME terminal has successfully completed the previous command.

The default value for the Command Acknowledge bits prior to any command acknowledgement by the JAGXTREME terminal is 0.

Floating Point Input Indicator Field

The Floating Point Input Indicator field describes the type of content in the FPValue field according to the definitions in the following table:

Value	Meaning	Value	Meaning
1	Gross Weight*	12	Notch Filter Frequency
2	Net Weight*	13	1st Setpoint Coincidence*
3	Tare Weight*	14	2nd Setpoint Coincidence*
4	Fine Gross Weight*	15	1st Setpoint Dribble
5	Fine Net Weight*	16	1st Setpoint Dribble
6	Fine Tare Weight*	17	1st Setpoint Tolerance
7	Rate*	18	Primary Units-Low Increment Size
8	JagBASIC Custom Variable 1	19-28	Reserved
9	JagBASIC Custom Variable 2	29	Last JAGXTREME Error Code
10	JagBASIC Custom Variable 3	30	No Data Response – Command Success
11	Low-Pass Filter Corner Frequency	31	No Data Response – Command Failed

*These types of values may be reported in rotation.

Assembly Object Instance 4

Assembly Instance 4 represents data sent to the JAGXTREME terminal by the Logix AC when the user has configured the CIP Network Interface Card for a data format of Floating Point..

Instance Attributes

Attr ID	Access	Name	Data Type	Default Value
3	Get / Set	Data	Floating Point Output Slots[4]	N/A
4	Get	Size	UINT16	26 (Bytes) or 13 (Int)

Structure of Floating Point Output Slots.

The Floating Point Output Data consists of one reserved integer and four scale slots. The first scale slot starts at byte 2.

Bytes 0- 1	Reserved
Bytes 2-7	Slot 1
Bytes 8-13	Slot 2
Bytes 14-19	Slot 3
Bytes 20-25	Slot 4

Typically, the user assigns one slot per scale in the JAGXTREME 'Config Options' menu. However, the user can increase the bandwidth of the I/O channel for one or two scales by selecting only one or two scales in the Config Options menu. The JAGXTREME terminal uses the following criteria for assigning scale slots to scales:

- If the user selects only one scale in the "Config Options" menu, then the JAGXTREME terminal assigns slots 1 and 2 for communicating the scale data with the controller.
- When you select only two scales, then the JAGXTREME terminal assigns slots 1 and 3 for communicating scale 1 data with the controller. It assigns slots 2 and 4 for communicating scale 2 data with the controller.
- If the user selects more than two scales, then the JAGXTREME terminal assigns one scale per scale slot.

Each of the four Floating Point Output Slots within Assembly Instance 4's Data attribute has the following format:

Name	Data Type	Description
Command	INT16	Preset Tare or Setpoint Value
FPValue	FLOAT	Command dependant floating point value
Total Length:	6 Bytes	

Behavior of the Floating Point Output Instance

In order to receive Floating Point Input data from the JAGXTREME terminal, the controller must issue a command for a particular scale by setting a Scale Command in the Floating Point Output Data of Assembly Instance 4. Refer to the appendix for a list of the recognized Scale Commands.

The JAGXTREME terminal recognizes a new command from the controller whenever it sees a new value in the Scale Command register. If the command has an associated floating-point value, the controller must set the value in the floating-point output register before issuing the command.

The controller can select a rotation of up to nine floating-point input fields from each scale. For example, the controller can alternatively look at weight and rate by issuing commands to place both in the rotation. The JAGXTREME terminal stores the rotation in Shared Data so that it does not have to be reinitialized after each power cycle. When the controller does not set up an input rotation, the default input rotation consists of gross weight only.

The controller may request that the JAGXTREME terminal continually cycle among the fields of the input rotation by placing 0 in the scale command. In that case, the JAGXTREME terminal will automatically select the next field from the input rotation at the next A-to-D update.

To control the pace of the input rotation, the controller may request the next field from the input rotation by alternating between commands 1 and 2. The controller needs to change the command value so that the JAGXTREME terminal knows when the controller is requesting a new field. In this way, the controller controls when the JAGXTREME terminal switches to the next field of input rotation.

Commands 10 through 29 are "Report Data" commands. As long as one of the Report Data commands is in the Scale Command, the JAGXTREME terminal will report the requested data and will not report data from the input rotation.

When the controller requests a real-time field from the JAGXTREME terminal, the JAGXTREME terminal acknowledges the command only once but sends a new value for that field at every analog-to-digital weight update. The controller requests real-time fields either through the report data command or through the input rotation.

When the controller requests a static field from the JAGXTREME terminal, the JAGXTREME terminal acknowledges the command once and sets a new value for that field in the output register once. However, that value remains in the output register until the controller issues another command. Examples of static fields are setpoint and filter values.

After acknowledging the previous command, the JAGXTREME terminal will act on a new command only when the controller sets a new value in a Scale Command in the Controller Output message.

Scale Weight Object

Class Code

Class Code: 64 hex

The Scale Weight Object contains the full complement of scale weight data for a given scale.

Class Attributes

The Scale Weight Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1 thru 5)	Get Attribute Single
0x10	Instance (1 thru 5)	Set Attribute Single

Instances

One instance of the Scale Weight Object will exist for each of up to five possible scales connected to a given JAGXTREME terminal either directly or indirectly. The Instance will be equal to the internal scale number. Example. To read the Displayed Gross Weight for internal scale 3 (wt301) set the instance to "3"

Instance Attributes

Attr ID	Access [1]	Shared Data Name[2]	Data Type	Description
0x01 hex	Get / Set	wtn01	ASCII[13] [3]	DisplayedGrossWeight
0x02 hex	Get / Set	wtn02	ASCII[13] [3]	DisplayedNetWeight
0x03 hex	Get / Set	wtn03	ASCII[3]	DisplayedWeightUnits: lb = pounds kg = kilograms g = grams t = metric tons
0x04 hex	Get / Set	wtn04	ASCII[13] [3]	DisplayedAuxGrossWeight
0x05 hex	Get / Set	wtn05	ASCII[13] [3]	DisplayedAuxNetWeight
0x06 hex	Get / Set	wtn06	ASCII[6]	DisplayedAuxWeightUnits: lb = pounds kg = kilograms oz = ounces lb-oz = pounds & ounces ozt = troy ounces dwt = penny weights t = metric tons ton = tons or custom units name
0x07 hex	Get / Set	wtn07	CHAR	DisplayedAuxRatePeriod: N = No S = Sec M = Min H = Hour
0x08 hex	Get / Set	wtn08	ASCII[13] [3]	DisplayedRate
0x09 hex	Get / Set	wtn09	ASCII[13] [3]	DisplayedDiagnosticWeight
0x0A hex	Get / Set	wtn10	DOUBLE	LegalGrossWeight

Attr ID	Access [1]	Shared Data Name[2]	Data Type	Description
0x0B hex	Get / Set	wtn11	DOUBLE	LegalNetWeight
0x0C hex	Get / Set	wtn12	DOUBLE	AuxiliaryGrossWeight
0x0D hex	Get / Set	wtn13	DOUBLE	AuxiliaryNetWeight
0x0E hex	Get / Set	wtn14	DOUBLE	AuxiliaryRate
0x0F hex	Get / Set	wtn15	UINT8	ScaleState: 0 = Disabled 1 = Normal Weight Processing, 2 = Diagnostic 3 = Calibration, 4 = Shift Adjust 5 = Error
0x10 hex	Get / Set	wtn16	UINT8	ContinuousOutputStatusA
0x11 hex	Get / Set	wtn17	DOUBLE	FineGrossWeight
0x12 hex	Get / Set	wtn18	DOUBLE	FineNetWeight
0x13 hex	Get / Set	wtn19	UINT8	WeighingRange: 0 = Single Range 1 = Multi Range 1 2 = Multi Range 2 3 = Multi Range 3
0x14 hex	Get / Set	wtn20	DOUBLE	WIM Time Counts
0x15 hex	Get / Set	wtn21	ASCII[3][3]	WIM Weight Units: lb = pounds kg = kilograms g = grams t = metric tons

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5). The object "Instance" should be set to the scale number.
3. Right justified, null terminated string.

Dynamic Data Object

Class Code

Class Code: 65 hex

The Dynamic Data Object contains a variety of scale related data.

Class Attributes

The Dynamic Data Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance	Get Attribute Single
0x10	Instance	Set Attribute Single

Instances

Only 1 instance of the Dynamic Data Object is supported: Instance 1.

Instance Attributes

Attr ID	Access [1]	Shared Data Name	Data Type	Description
0x02 hex[2]	Get	bd002	ASCII[60]	Board Configuration String[2]
0x03 hex	Get	bd003	ASCII[2]	Latest Key Stroke Key Source
0x04 hex	Get	bd004	ASCII[10]	EEPROM Authorization String
0x05 hex	Get	bd005	ASCII[13]	Console Software Part No
0x06 hex	Get	bd006	ASCII[13]	Analog 1 Software Part No
0x07 hex	Get	bd007	ASCII[13]	Analog 2 Software Part No
0x08 hex	Get	bd008	ASCII[13]	DigiTOL 1 Software Part No
0x09 hex	Get	bd009	ASCII[13]	DigiTOL 2 Software Part No
0x0A hex	Get	bd010	ASCII[13]	IDNET 1 Software Part No
0x0B hex	Get	bd011	ASCII[13]	Ident 2 Software Part No
0x0C hex	Get	bd012	ASCII[13]	POWERCELL Software Part No
0x10 hex	Get	bd016	ASCII[25]	POWERCELL Scale-Cell Errors: Error numbers for up to 24 power cells. Cell errors for Scales A - D
0x12 hex	Get	bd018	ASCII[25]	Scan Table: Ordered list of current POWERCELL Addresses.
0x13 hex	Get	bd019	UINT32[24]	POWERCELL Scale-Cell Counts[2]
0x14 hex[2]	Get	bd020	CHAR[25]	POWERCELL Overload State[2]
0x15 hex[2]	Get	bd021	CHAR[25]	POWERCELL Zero Drift State[2]
0x16 hex	Get	bd022	ASCII[13]	ControlNet Software Part Number
0x17 hex	Get	bd023	CHAR	Conveyor Scale Command set by JagBASIC Application to send a command to the Analog Board Conveyor: R = Reset B = Begin W = End Weigh
0x18 hex	Get / Set	bd024	ASCII[40]	JagBASIC Email Destination
0x19 hex	Get / Set	bd025	ASCII[40]	JagBASIC Email Subject Line

Attr ID	Access [1]	Shared Data Name	Data Type	Description
0x1A hex	Get / Set	bd026	UINT8	JagBASIC Email Active Status: 0 = Email cannot be sent 1 = Email can be sent
0x1E hex	Get	bd030	BIT[8]	Read Discrete Inputs
0x1F hex	Get	bd031	BIT[8]	Read Discrete Outputs
0x20 hex	Get	bd032	BIT[8]	Read Status Flags for Scale A remotely
0x21 hex	Get	bd033	BIT[8]	Read Status Flags for Scale B remotely
0x22 hex	Get	bd034	BIT[8]	Read Status Flags for Scale C remotely
0x23 hex	Get	bd035	BIT[8]	Read Status Flags for Scale D remotely
0x24 hex	Get	bd036	BIT[8]	Read Status Flags for Scale E remotely
0x25 hex	Get	bd037	BIT[8]	Read JagBASIC Custom Flags s_250 -s_s5f remotely
0x26 hex	Get	bd038	UINT8	Read Cal Switch remotely
0x38 hex	Get	bd056	UINT8	HMI Attached: Must be set to 1 by HMI after JAGXTREME sets it to 0 at power-up.
0x39 hex	Get	bd057	UINT8	Unacknowledged Error Present
0x46 hex	Get	bd070	UINT8	ControlNet Ping Status: 0 = PING_INVALID 1 = PING_POWER_UP 2 = PING_CHECK_FOR_CABLE 3 = PING_WAITING_TO_ROGUE 4 = PING_CHECK_MODERATOR 5 = PING_IM_ALIVE 6 = PING_ATTACHED_ONLINE 7 = PING_FORCED_LISTEN_ONLY 8 = PING_DUPLICATE_LISTEN_ONLY 9 = PING_CNET_TIMEOUT
0x55 hex	Get	bd085	UINT8	DisplayBoard (1 = Yes, 0 = No)
0x56 hex	Get	bd086	UINT8	AnalogBoard1 (1 = Yes, 0 = No)
0x57 hex	Get	bd087	UINT8	AnalogBoard2 (1 = Yes, 0 = No)
0x58 hex	Get	bd088	UINT8	AllenBradleyPLC (1 = Yes, 0 = No)
0x59 hex	Get	bd089	UINT8	PROFIBUS (1 = Yes, 0 = No)
0x5A hex	Get	bd090	UINT8	ControlNet (1 = Yes, 0 = No)
0x5B hex	Get	bd091	UINT8	MultiFunctionIO_1 (1 = Yes, 0 = No)
0x5C hex	Get	bd092	UINT8	PowerCell_Board_1 (1 = Yes, 0 = No)
0x5D hex	Get	bd093	UINT8	ModBus Plus (1 = Yes, 0 = No)
0x5E hex	Get	bd094	UINT8	AnalogOut (1 = Yes, 0 = No)
0x5F hex	Get	bd095	UINT8	HighPrec1 (1 = Yes, 0 = No)
0x60 hex	Get	bd096	UINT8	HighPrec2 (1 = Yes, 0 = No)
0x61 hex	Get	bd097	UINT8	MultiFunctionIO_2 (1 = Yes, 0 = No)

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

2. Additional details for this attribute provided below

Attribute 2: Board Configuration String

Attribute 2, the Board Configuration String, consists of a 15-byte entry for each of the four board slots. Each entry is formatted as follows:

Data Type	Description
CHAR[2]	Board Identifier
CHAR[13]	Board Software Serial Number (where applicable)

Attribute 13: POWERCELL Scale - Cell Counts

Attribute 13, POWERCELL Scale - Cell Counts, contains the current shift-adjusted counts for consecutive power cells in a scale. An application can request the current counts for a scale by setting trigger t_69d for Scale A, t_6ad for Scale B to 1, t_62d for Scale C, and t_63d for Scale D.

Attribute 14: POWERCELL Overload State

Attribute 14, the POWERCELL Overload State, consists of one entry each for up to 24 power cells:

0 = Cell not assigned

1 = Cell OK,

2 = Cell in Overload condition

Attribute 15: POWERCELL Zero Drift State

Attribute 15, the POWERCELL Zero Drift State, consists of one entry each for up to 24 power cells:

0 = Cell not assigned

1 = Cell OK,

2 = Cell in Zero Drift Threshold Exceeded condition

Scale Weight Static Object

Class Code

Class Code: 66 hex

The Scale Weight Static Object contains static weight data for a given scale.

Class Attributes

The Scale Weight Static Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance	Get Attribute Single
0x10 hex	Instance	Set Attribute Single

Instances

One instance of the Scale Weight Static Object will exist for each of up to 5 possible scales connected to a given JAGXTREME terminal either directly or indirectly. The instance is set to the scale number, i.e. the instance equals 3 for scale 3.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex	Get	wsn01	CHAR	ScaleModeOut: G = Gross N = Net
0x02 hex	Get	wsn02	ASCII[13] [3]	DisplayedTareWeight[3]
0x03 hex	Get	wsn03	ASCII[13] [3]	DisplayedAuxTareWeight[3]
0x04 hex	Get	wsn04	DOUBLE	FineTareWeight
0x05 hex	Get	wsn05	DOUBLE	AuxiliaryTareWeight
0x06 hex	Get	wsn06	UINT8	CurrentUnits: 1 = Primary 2 = Secondary
0x07 hex	Get	wsn07	UINT8	TareSource: 1 = Pushbutton 2 = Keyboard 3 = Autotare
0x08 hex	Get	wsn08	DOUBLE	Current Zero Counts (PB & AZM)
0x09 hex	Get	wsn09	CHAR[2]	TareSourceString: PT = Keyboard Tare, else "T "
0x0A hex	Get	wsn10	ASCII[13] [3]	DisplayedStoredWeight[3]
0x0B hex	Get	wsn11	DOUBLE	Stored Weight
0x0C hex	Get	wsn12	DOUBLE	LegalTareWeight
0x0D hex	Get	wsn13	ASCII[41]	LastScaleError: Date - Time - Error Message
0x0E hex	Get	wsn14	FLOAT	Number of Scale IO errors since calibration or reset
0x14 hex	Get	wsn20	FLOAT	Number of Weighments since last calibration or reset
0x15 hex	Get	wsn21	FLOAT	Number of Platform Overloads since calibration or reset
0x16 hex	Get	wsn22	FLOAT	Number of Platform High Impacts since calibration or reset
0x17 hex	Get	wsn23	FLOAT	Number of Zero Commands since calibration or reset
0x18 hex	Get	wsn24	FLOAT	Number zero command failures due to "out of zero range" since calibration or reset.

Attr ID	Access [1]	Name[2]	Data Type	Description
0x19 hex	Get	wsn25	UINT8	SymmetryCheckFailure: 0 = No Failure 1 = Estimate-able Symmetry Failure 2 = Estimate-able Comm Failure 3 = UnCorrectable Symmetry Failure 4 = UnCorrectable Comm Failure 5 = Estimate-able Zero Drift Failure 6 = UnCorrectable Zero Drift Failure
0x1A hex	Get	wsn26	UINT8	RunFlatCellFromSymmetryCheck: POWERCELL that was detected bad in symmetry check. If run flat is enabled, this cell is replaced using weight counts from replacement cell (0-23).
0x1B hex	Get	wsn27	UINT8	RunFlatReplacementCell: POWERCELL that is used as replacement cell in run flat operation (0-23).
0x1C hex	Get	wsn28	UINT8	CalibrationCheckFailure: 0 = No 1 = Latest calibration check failed

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5).
3. Right justified, null terminated string.

Scale Calibration Object

Class Code

Class Code: 67 hex

The Scale Calibration Object contains calibration data for a given scale.

Class Attributes

The Scale Calibration Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance(1 thru 5)	Get Attribute Single
0x10 hex	Instance(1 thru 5)	Set Attribute Single

Instances

One instance of the Scale Calibration Object will exist for each of up to five possible scales connected to a given JAGXTREME either directly or indirectly. The instance is set to the scale number, i.e. the instance equals 3 for scale 3.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex	Get / Set	csn01	UINT8	Auxiliary Display Units: 1 = Pounds 2 = Kilograms 3 = Grams 4 = Ounces 5 = Pounds and ounces 6 = Troy Ounces, 7 = Penny Weights 8 = Metric Tons 9 = Tons 10 = Custom units
0x02 hex	Get / Set	csn02	ASCII[7]	Custom Units Name
0x03 hex	Get / Set	csn03	DOUBLE	Custom Units Conversion Factor
0x04 hex	Get	csn04	CHAR	Rate Time Units: (No, Sec, Min, Hour)
0x05 hex	Get	csn05	UINT8	Rate Sample Period: If csn06 = 0,1,or 2: Specifies the number of seconds over which the rate is averaged. If csn06 = 3: Specifies the value for both frequency and sample period in number of weight updates.
0x06 hex	Get	csn06	UINT8	Rate Calculation Frequency: 0 = every second 1 = every five seconds, 2 = every half-second. 3 = calculation frequency and sample period are specified in number of weight updates, as specified in Rate Sample Time.
0x07 hex	Get	csn07	CHAR	Analog Scale Weigh In Motion (W = Yes) Valid only for the first Single or Dual Analog Scale.
0x08 hex	Get	csn08	BOOL8	IDNET Higher Precision (Use IDNET Scale "times 10" Precision)
0x09 hex	Get	csn09	ASCII[2] [3]	Power Up Timer

Attr ID	Access [1]	Name[2]	Data Type	Description
0x0A hex	Get	csn10	DOUBLE	Low Pass Filter Corner Frequency (1 Hz to 9.9 Hz by .1 Hz)
0x0B hex	Get	csn11	DOUBLE	Notch Filter Frequency (1 Hz to 9.9 Hz by .1 Hz)
0x0C hex	Get	csn12	DOUBLE	Comb Filter Frequency (1 Hz to 9.9 Hz by .1 Hz)
0x0D hex	Get / Set	csn13	DOUBLE	Print Threshold weight
0x0E hex	Get / Set	csn14	DOUBLE	Print Reset Threshold weight
0x0F hex	Get	csn15	DOUBLE	Display Update Frequency hertz
0x10 hex	Get	csn16	DOUBLE	Custom Continuous Out Update Frequency hertz
0x11 hex	Get	csn17	UINT8	Low Pass Filter Poles
0x12 hex	Get / Set	csn18	ASCII[9]	Scale ID
0x13 hex	Get / Set	csn19	UINT8	Averaging Filter Order
0x14 hex	Get	csn20	UINT8	Comb Filter Order
0x15 hex	Get / Set	csn21	CHAR	Scale Type: A = Analog Load Cells P = Power Digital Load Cells I = High Precision S = Single cell DigiTOL M = Power Module DigiTOL H = UltraResHigh L = UltraResLow U = Summing
0x16 hex	Get	csn22	UINT8	Scale Location: 0 = first unit 1 = second unit (board or COM: port)
0x17 hex	Get	csn23	CHAR	IDNetVibrationAdaptor: '0' - '9' (specific to Precision Base)
0x18 hex	Get	csn24	CHAR	IDNetWeighingProcessAdaptor: '0' - '9' (specific to Precision Base)
0x19 hex	Get	csn25	CHAR	IDNetAutomaticStabilityDetection: '0' - '9' (specific to Precision Base)
0x1A hex	Get	csn26	BOOL8	IDNetAutoZeroSetting
0x1B hex	Get	csn27	ASCII[12]	IDNetSoftwarePartNum: "xxxx-x-xxxx" string from precision base
0x1C hex	Get	csn28	CHAR[2]	IDNetIdentcode: '—' to '99' calibration count from Precision Base
0x1D hex	Get	csn29	BOOL8	Scales In Summing Scale (Add scale to summing scale)
0x1E hex	Get	csn30	ASCII[12]	Calibration Date
0x1F hex	Get	csn31	UINT32	Next Scheduled Calibration Date (in seconds since 1970 GMT)
0x20 hex	Get	csn32	UINT8	Calibration Interval In Days
0x21 hex	Get	csn33	UINT32	Calibration Interval In Weighments
0x22 hex	Get / Set	csn34	UINT8	Cal Expired Announcement: 1 = log only 2 = disable scale and alarm 3 = email alert and alarm 4 = alarm only
0x23 hex	Get / Set	csn35	UINT32	Last Calibration Date (in seconds since 1970 GMT)
0x24 hex	Get / Set	csn36	DOUBLE	Calibration Check Tolerance (Weight tolerance in primary units)
0x25 hex	Get / Set	csn37	UINT8	Number of Calibration Check Points
0x26 hex	Get / Set	csn38	UINT8	Calibration Check Failed Announcement: 1 = log only 2 = disable scale and alarm 3 = email alert and alarm 4 = alarm only

Attr ID	Access [1]	Name[2]	Data Type	Description
0x28 hex	Get / Set	csn40	UINT8	MonitorCellOverloads: 0 = No 1 = Count 2 = Count and Log
0x29 hex	Get / Set	csn41	DOUBLE	Cell Overload Threshold (in units in csn54)
0x2A hex	Get / Set	csn42	UINT8	Monitor Platform Overloads: 0,1 = Count 2 = Count and Log
0x2C hex	Get / Set	csn44	UINT8	Monitor Platform High Impacts: 0 = No 1 = Count 2 = Count and Log
0x2D hex	Get / Set	csn45	FLOAT	High Impact Weight Threshold (in primary units)
0x2E hex	Get / Set	csn46	FLOAT	High Impact Rate Threshold (in primary units / second)
0x2F hex	Get / Set	csn47	UINT8	Monitor Weighments: 0 = No 1 = Count 2 = Count and Log
0x30 hex	Get / Set	csn48	UINT8	Weighment Trigger: 0 = None 1 = Print Command 2 = Upscale Gross Weight Threshold 3 = Downscale Gross Weight Threshold 4 = Upscale Net Weight Threshold 5 = Downscale Net Weight Threshold
0x31 hex	Get / Set	csn49	DOUBLE	Weighment Threshold
0x32 hex	Get / Set	csn50	UINT8	Monitor Zero Commands: 0,1 = Count 2 = Count and Log
0x33 hex	Get / Set	csn51	UINT8	Monitor Zero Command Failures: 0,1 = Count 2 = Count and Log
0x34 hex	Get / Set	csn52	UINT8	Monitor Scale IO Errors: 0,1 = Count 2 = Count and Log
0x35 hex	Get	csn53	DOUBLE	Weighment Reset Threshold
0x36 hex	Get	csn54	UINT8	Cell Overload Units: 1 = counts 2 = primary unit
0x37 hex	Get	csn55	UINT8	Run Flat This Specific Cell
0x38 hex	Get/Set	csn56	UINT8	Threshold to Begin Symmetry: % of capacity to begin symmetry checking
0x3E hex	Get/Set	csn62	UINT8	Cell SymmetryCheck: 0 = No 1 = Count 2 = Count and Log
0x3F hex	Get/Set	csn63	UINT8	Cell Zero Drift Check: 0 = No 1 = Count 2 = Count and Log
0x40 hex	Get/Set	csn64	DOUBLE	Cell Zero Drift Check Threshold (in percent of span)

Attr ID	Access [1]	Name[2]	Data Type	Description
0x41 hex	Get/Set	csn65	UINT8	Cell Symmetry: 0 = No 1 = Radial 2 = Left-Right Pairs
0x42 hex	Get/Set	csn66	UINT8	Cell Symmetry Threshold: Percent difference (0-99)
0x43 hex	Get/Set	csn67	UINT8	Predictive Failure Announcement: 1 = log only 2 = disable scale and alarm 3 = email alert and alarm 4 = alarm only
0x44 hex	Get/Set	csn68	BOOL8	Run Flat Weight Estimation
0x55 hex	Get/Set	csn85	BOOL8	FillNoise Filter Enable
0x56 hex	Get/Set	csn86	BOOL8	Auto Print
0x57 hex	Get/Set	csn87	BOOL8	No Motion Before Print
0x58 hex	Get/Set	csn88	BOOL8	Display Rate
0x59 hex	Get/Set	csn89	BOOL8	Display Auxiliary Units
0x5A hex	Get/Set	csn90	BOOL8	Units Switch Enable
0x5B hex	Get/Set †	csn91	BOOL8	Print Interlock Enable
0x5C hex	Get/Set	csn92	BOOL8	Do IDNET Tare In Jag
0x5D hex	Get/Set	csn93	UINT8	Process Application: 0 = Low 1 = Mid 2 = High

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the 'off' position.
2. The character 'n' represents the internal scale number (1-5).
3. Right justified, null terminated string.

Scale Tare Object

Class Code

Class Code: 68 hex

The Scale Tare Object contains tare data for a given scale.

Class Attributes

The Scale Tare Static Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1 thru 5)	Get Attribute Single
0x10 hex	Instance (1 thru 5)	Set Attribute Single

Instances

One instance of the Scale Tare Static Object will exist for each of up to five possible scales connected to a given JAGXTREME terminal either directly or indirectly. The instance is set to the scale number, i.e. the instance equals 3 for scale 3.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex	Get / Set	trn01	DOUBLE	AutoTareThreshold
0x02 hex	Get / Set	trn02	DOUBLE	AutoTareResetThreshold
0x03 hex	Get / Set	trn03	DOUBLE	AutoClearTareThreshold
0x55 hex	Get / Set	trn85	BOOL8	TareEnabled
0x56 hex	Get / Set	trn86	BOOL8	PushbuttonTare
0x57 hex	Get / Set	trn87	BOOL8	KeyboardTare
0x58 hex	Get / Set	trn88	BOOL8	AutoTare
0x59 hex	Get / Set	trn89	BOOL8	AutoTareCheckMotion
0x5A hex	Get / Set	trn90	BOOL8	Auto Clear Tare
0x5B hex	Get / Set	trn91	BOOL8	Auto Clear Tare After Print
0x5C hex	Get / Set	trn92	BOOL8	Auto Clear Tare Motion
0x5D hex	Get / Set	trn93	BOOL8	Tare Interlock
0x5E hex	Get / Set	trn94	BOOL8	Display Tare
0x5F hex	Get / Set	trn96	BOOL8	Net Sign Correction

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5).

Setpoint Object

Class Code

Class Code: 69 hex

The Setpoint Object contains data for scale setpoints.

Class Attributes

The Setpoint Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance	Get Attribute Single
0x10	Instance	Set Attribute Single

Instances

One instance of the Setpoint Object may exist for each of 12 setpoints. The default assignment for setpoint B is ZERO TOLERANCE for Scale A. The default assignment for setpoint C is ZERO TOLERANCE for Scale B.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex	Get / Set	spn01	ASCII[9]	Setpoint Name
0x02 hex	Get / Set	spn02	CHAR	Setpoint Enable Button: 0 = Disabled 1 = Scale 1 2 = Scale 2 3 = Scale 3 4 = Scale 4 5 = Scale 5
0x03 hex	Get / Set	spn03	CHAR	Setpoint Target Variable: G = Gross N = Net D = Displayed R = Rate H = Gross With AutoPreact M = Net with AutoPreact, L = Learn Jog Weight Associated with a Timer Value J = Jog using Timer Value
0x05 hex	Get / Set	spn05	DOUBLE	Setpoint Coincidence Value: For most setpoint targets, this field is a weight value. For LearnJog setpoints, this field contains a time value.
0x06 hex	Get / Set	spn06	DOUBLE	Setpoint Preact Value: For most setpoint targets, this field has a weight preact value. For AutoPreact setpoints, this field contains a time value in seconds.
0x08 hex	Get / Set	spn08	DOUBLE	Setpoint DribbleValue: weight
0x0A hex	Get / Set	spn10	DOUBLE	Setpoint Tolerance Value: weight
0x56 hex	Get / Set	spn86	UC	Setpoint Fill Or Discharge: 0 = Fill 1 = Discharge
0x57 hex	Get / Set	spn87	UC	Setpoint Latching: 1 = Feed Latching Enabled

Attr ID	Access [1]	Name[2]	Data Type	Description
0x58 hex	Get / Set	spn88	UC	Setpoint Latched: 0 = Unlatched 1 = Latched

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal setpoint number. The Instance is set to the internal setpoint number..

Behavior

When an application enables "Feed Latching", the JAGXTREME O/S sets the Setpoint Latched=1 and the Setpoint Feeding=0 when it next encounters the setpoint coincidence. Then, it never resets the Setpoint Feeding=0 condition again until the application resets the Setpoint Latched=0. That is, the application must reset SetpointLatched=0 before starting a new setpoint. Jog and LearnJog setpoints are always latched when the setpoints are started, so the application must reset the latch before starting a new setpoint.

System Object

Class Code

Class Code: 6A hex

The System Object contains a variety of system data.

Class Attributes

The System Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

Only 1 instance of the System Object is supported: Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 hex	Get / Set	jag01	ASCII[3]	Current Selected Scale: First Char = L or n, 2nd = A or B
0x04 hex	Get / Set	jag04	CHAR	Market: U = USA E = European Community A = Australia C = Canada
0x05 hex	Get / Set	jag05	UINT8	DateFormat
0x06 hex	Get / Set	jag06	UINT8	TimeFormat
0x07 hex	Get / Set	jag07	ASCII[9]	JulianDate
0x08 hex	Get / Set	jag08	ASCII[9]	JulianTime
0x09 hex	Get / Set	jag09	UINT32	Consecutive Number counter
0x0A hex	Get	jag10	ASCII[41]	LastErrorMessage: Date - Time - Error Message
0xc0B hex	Get	jag11	ASCII[13]	SoftwareID
0x0D hex	Get / Set	jag13	UINT8	NumberDiscreteInputs: Number of discrete I/O's that are assigned as inputs, including 4 dedicated controller card discrete inputs plus the assigned MFIO inputs. A value 'B' assigns default value 12 = 4 controller card inputs plus 8 MFIO inputs.
0x0E hex	Get	jag14	UINT32	BRAMVersionNumber
0x0F hex	Get	jag15	UINT8	NumberOfInternalScales
0x10 hex	Get / Set	jag16	CHAR	DateSeparator
0x11 hex	Get / Set	jag17	CHAR	TimeSeparator
0x12 hex	Get / Set	jag18	ASCII[11]	ConsecutiveNumberDest
0x13 hex	Get / Set	jag19	ASCII[12]	CurrentDate
0x14 hex	Get / Set	jag20	ASCII[12]	TimeOfDay
0x15 hex	Get / Set	jag21	ASCII[11]	WeekDay
0x16 hex	Get / Set	jag22	UINT32	ConsecutiveNumberPreset

Attr ID	Access [1]	Name	Data Type	Description
0x17	Get/ Set	jag23	UINT8	CharacterSet: 0 = USA 1 = France 2 = England 3 = Germany 4 = Denmark-I 5 = Sweden 6 = Italy 7 = Spain-I 8 = Japan 9 = Norway 10 = Denmark-II 11 = Spain-II 12 = Latin America
0x18	Get / Set	jag24	UINT8	Language: 0 = English 1 = French 2 = German 4 = Spanish
0x19	Get / Set	jag25	UINT8	Keyboard: 0 = English 1 = French 2 = German 4 = Spanish
0x1A	Get	jag26	ASCII[24]	Scale Error Log Reset Time
0x1B	Get	jag27	ASCII[24]	Monitor Counts Reset Time
0x1C	Get	jag28	ASCII[24]	Monitor Log Reset Time
0x1D	Get	jag29	UINT32	Monitor Log Max Records
0x1E	Get	jag30	UINT32	Monitor Log Next Record Pointer
0x1F	Get	jag31	UINT8	Monitor Log File Full Indicator
0x20	Get	jag32	ASCII[80]	LastDemandPrintMessage
0x21	Get	jag33	ASCII[20]	JAGXTREME Serial #
0x22	Get / Set	jag34	ASCII[20]	JAGXTREME ID
0x23	Get / Set	jag35	ASCII[20]	JAGXTREME Project
0x24	Get / Set	jag36	ASCII[80]	JAGXTREME Description
0x25	Get	jag37	UINT8	Indirect Read Counter
0x26	Get	jag38	FLOAT	Power Cycle Counter

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

User Literals Object

Class Code

Class Code: 6B hex

The User Literals Object contains user-defined literal strings.

Class Attributes

The User Literals Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1)	Get Attribute Single
0x10	Instance (1)	Set Attribute Single

Instances

The User Literals Object has only a single instance; Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 through 0x32	Get / Set	lit01 through lit050	ASCII[40]	User Literals 1 - 50

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

User Prompts Object

Class Code

Class Code: 6C hex

The User Prompts Object contains user-defined prompt strings.

Class Attributes

The User Prompts Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1)	Get Attribute Single
0x10	Instance (1)	Set Attribute Single

Instances

The User Prompts Object has only a single instance; Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 through 0x14	Get / Set	pmt01 through pmt20	ASCII[16]	User Prompts 1 - 20

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the 'off' position.

User Variables Object

Class Code

Class Code: 6D hex

The User Variables Object contains user-defined variables.

Class Attributes

The User Variables Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1)	Get Attribute Single
0x10	Instance (1)	Set Attribute Single

Instances

The User Variables Object has only a single instance—Instance 1.

Instance Attributes

Attr ID	Access ☐	Name	Data Type	Description
0x01 through 0x14	Get / Set	var01 through var020	CHAR[47]	User Variables 1 - 20: USER_VARIABLE structure
0x51	Get / Set	var81	UINT8	Variables in Use: (0-20)
0x52	Get / Set	var82	UINT8	Prompt Looping Mode: 0 = No Loop 1 = Loop

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

Cluster Variables Object

Class Code

Class Code: 6E hex

The Cluster Variables Object contains Cluster variables.

Class Attributes

The Cluster Variables Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

The Cluster Variables Object has only a single instance; Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 hex through 0x14 hex	Get / Set	clv01 through clv20	ASCII[40]	Cluster Variables 1 – 20

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

Behavior

Cluster Variable fields may contain Jog Tables for the Jog setpoints. The fields have numbers in string format.

Cluster variables 1-10 are the weight values.

Cluster Variables 11-20 are the associated timer values.

Basic Application Object

Class Code

Class Code: 70 hex

The Basic Application Object contains a variety of data related to the JagBASIC Application.

Class Attributes

The Basic Application Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

Only 1 instance of the Basic Application Object is supported: Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 hex	Get / Set	bas01	ASCII[20]	Program 1
0x02 hex	Get / Set	bas02	ASCII[20]	Program 2
0x03 hex	Get / Set	bas03	ASCII[20]	Program 3
0x04 hex	Get / Set	bas04	ASCII[20]	Program 4
0x05 hex	Get / Set	bas05	ASCII[20]	Program 5
0x06 hex	Get / Set	bas06	ASCII[20]	Program 6
0x07 hex	Get / Set	bas07	ASCII[20]	Program 7
0x08 hex	Get / Set	bas08	ASCII[20]	Program 8
0x09 hex	Get / Set	bas09	ASCII[20]	Program 9
0x0A hex	Get / Set	bas10	UINT8	Keyboard Source: 0 = None 1 = Keypad 2 = Keyboard 3 = Both
0x0B hex	Get / Set	bas11	UINT8	Display Destination: 0 = None 1 = Lower Display 2 = Serial Port
0x0C hex	Get / Set	bas12	DOUBLE	Programmable Tare Weight Scale A
0x0D hex	Get / Set	bas13	DOUBLE	Programmable Tare Weight Scale B
0x0E hex	Get / Set	bas14	FLOAT	Custom Output A1 From PLC
0x0F hex	Get / Set	bas15	ASCII[4]	Custom Output A2 From PLC
0x10 hex	Get / Set	bas16	FLOAT	Custom Output A3 From PLC
0x11 hex	Get / Set	bas17	ASCII[4]	Custom Output A4 From PLC

Attr ID	Access [1]	Name	Data Type	Description
0x12 hex	Get / Set	bas18	FLOAT	Custom Input A1 To PLC (High Speed)
0x13 hex	Get / Set	bas19	ASCII[4]	Custom Input A2 To PLC (High Speed)
0x14 hex	Get / Set	bas20	FLOAT	Custom Input A3 To PLC
0x15 hex	Get / Set	bas21	ASCII[4]	Custom Input A4 To PLC
0x16 hex	Get / Set	bas22	FLOAT	Custom Output B1 From PLC
0x17 hex	Get / Set	bas23	ASCII[4]	Custom Output B2 From PLC
0x18 hex	Get / Set	bas24	FLOAT	Custom Output B3 From PLC
0x19 hex	Get / Set	bas25	ASCII[4]	Custom Output B4 From PLC
0x1A hex	Get / Set	bas26	FLOAT	Custom Input B1 To PLC (High Speed)
0x1B hex	Get / Set	bas27	ASCII[4]	Custom Input B2 To PLC (High Speed)
0x1C hex	Get / Set	bas28	FLOAT	Custom Input B3 To PLC
0x1D hex	Get / Set	bas29	ASCII[4]	Custom Input B4 To PLC
0x1E hex	Get / Set	bas30	FLOAT	Custom Output C1 From PLC
0x1F hex	Get / Set	bas31	ASCII[4]	Custom Output C2 From PLC
0x20 hex	Get / Set	bas32	FLOAT	Custom Output C3 From PLC
0x21 hex	Get / Set	bas33	ASCII[4]	Custom Output C4 From PLC
0x22 hex	Get / Set	bas34	FLOAT	Custom Input C1 To PLC (High Speed)
0x23 hex	Get / Set	bas35	ASCII[4]	Custom Input C2 To PLC (High Speed)
0x24 hex	Get / Set	bas36	FLOAT	Custom Input C3 To PLC
0x25 hex	Get / Set	bas37	ASCII[4]	Custom Input C4 To PLC
0x26 hex	Get / Set	bas38	FLOAT	Custom Output D1 From PLC
0x27 hex	Get / Set	bas39	ASCII[4]	Custom Output D2 From PLC
0x28 hex	Get / Set	bas40	FLOAT	Custom Output D3 From PLC
0x29 hex	Get / Set	bas41	ASCII[4]	Custom Output D4 From PLC
0x2A hex	Get / Set	bas42	FLOAT	Custom Input D1 To PLC (High Speed)
0x2B hex	Get / Set	bas43	ASCII[4]	Custom Input D2 To PLC (High Speed)
0x2C hex	Get / Set	bas44	FLOAT	Custom Input D3 To PLC
0x2D hex	Get / Set	bas45	ASCII[4]	Custom Input D4 To PLC
0x2E hex	Get / Set	bas46	FLOAT	Custom Output E1 From PLC
0x2F hex	Get / Set	bas47	ASCII[4]	Custom Output E2 From PLC
0x30 hex	Get / Set	bas48	FLOAT	Custom Output E3 From PLC

Attr ID	Access [1]	Name	Data Type	Description
0x31 hex	Get / Set	bas49	ASCII[4]	Custom Output E4 From PLC
0x32 hex	Get / Set	bas50	FLOAT	Custom Input E1 To PLC (High Speed)
0x33 hex	Get / Set	bas51	ASCII[4]	Custom Input E2 To PLC (High Speed)
0x34 hex	Get / Set	bas52	FLOAT	Custom Input E3 To PLC
0x35 hex	Get / Set	bas53	ASCII[4]	Custom Input E4 To PLC
0x36 hex	Get / Set	bas54	FLOAT	Programmable Tare Weight Scale C
0x37 hex	Get / Set	bas55	FLOAT	Programmable Tare Weight Scale D
0x38 hex	Get / Set	bas56	FLOAT	Programmable Tare Weight Scale E
0x55 hex	Get / Set	bas85	BOOL8	Auto Start Enabled
0x56 hex	Get / Set	bas86	BOOL8	Escape Enabled
0x57 hex	Get / Set	bas87	BOOL8	Select Enabled
0x58 hex	Get / Set	bas88	BOOL8	Manual Start Enabled
0x59 hex	Get / Set	bas89	BOOL8	Manual Stop Enabled

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

POWERCELL Log Object

Class Code

Class Code: 71 hex

The POWERCELL Log Object contains POWERCELL information.

Class Attributes

The POWERCELL Log Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

The POWERCELL Log Object has only a single instance—Instance 1.

Instance Attributes

Attr ID	Access [1]	Name	Data Type	Description
0x01 hex	Get	pcl01	UINT32[24]	Number IO Errors- Cell (1-24)
0x03 hex	Get	pcl03	UINT32[24]	Current Zero Counts- Cell (1-24)
0x05 hex	Get	pcl05	UINT32[24]	Number Cell Overloads- Cell (1-24)
0x06 hex	Get	pcl06	UINT32[24]	Num Symmetry Failures- Cell (1-24)
0x07 hex	Get	pcl07	UINT32[24]	Number Zero Drift Failures- Cell (1-24)

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.

Shift Adjust Object

Class Code

Class Code: 73 hex

The Shift Adjust Object contains the shift adjust constants for all 24 cells.

Class Attributes

The Shift Adjust Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1 thru 5)	Get Attribute Single
0x10 hex	Instance (1 thru 5)	Set Attribute Single

Instances

One instance of the Shift Adjust Object will exist for each of up to five possible scales connected to a given JAGXTREME terminal either directly or indirectly.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex through 0x10 hex	Get	san01 through san16	UINT32	Shift Constants for cell 1 through 16
0x11 hex through 0x18 hex	Get	sxn17 through sxn24	UINT32	Shift Constants for cell 17 through 24

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5). The Instance is equal to the scale number.

Cell Calibration Object

Class Code

Class Code: 74 hex

The Cell Calibration Object contains calibration data for every cell.

Class Attributes

The Cell Calibration Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

The Cell Calibration Object supports one single instance; Instance 1

Instance Attributes

Attr ID	Access [1]	Name [2]	Data Type	Description
0x01 hex	Get	ccn01	UINT32[24]	Calibrated Zero Counts-Cell 1-24
0x02 hex	Get	Ccn02	UINT32[24]	Calibrated Span Counts-Cell 1-24

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5).

Level Sensitive Discrete Status Object

Class Code

Class Code: 75 hex

The Level Sensitive Discrete Status Object contains level sensitive logical discrete I/O data.

Class Attributes

The Level Sensitive Discrete Status Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1)	Get Attribute Single
0x10 hex	Instance (1)	Set Attribute Single

Instances

The Level Sensitive Discrete Status Object supports one single instance; Instance 1.

Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01 hex	Get	s_200	BOOL8	MotionOut_A
0x02 hex	Get	s_201	BOOL8	CenterOfZero_A
0x03 hex	Get	s_202	BOOL8	OverCapacity_A
0x04 hex	Get	s_203	BOOL8	UnderZero_A
0x05 hex	Get	s_204	BOOL8	NetMode_A
0x06 hex	Get	s_205	BOOL8	ScaleCriticalError_A
0x07 hex	Get	s_206	BOOL8	StoredWeightMode_A
0x08 hex	Get	s_207	BOOL8	ScaleSelected_A
0x09 hex	Get	s_208	BOOL8	MotionOut_B
0x0A hex	Get	s_209	BOOL8	CenterOfZero_B
0x0B hex	Get	s_20a	BOOL8	OverCapacity_B
0x0C hex	Get	s_20b	BOOL8	UnderZero_B
0x0D hex	Get	s_20c	BOOL8	NetMode_B
0x0E hex	Get	s_20d	BOOL8	ScaleCriticalError_B
0x0F hex	Get	s_20e	BOOL8	StoredWeightMode_B
0x10 hex	Get	s_20f	BOOL8	ScaleSelected_B
0x11 hex	Get	s_210	BOOL8	SetpointFeeding_1
0x12 hex	Get	s_211	BOOL8	SetpointFastFeeding_1
0x13 hex	Get	s_212	BOOL8	SetpointWithinTolerance_1
0x15 hex	Get	s_214	BOOL8	SetpointFeeding_2
0x16 hex	Get	s_215	BOOL8	SetpointFastFeeding_2
0x17 hex	Get	s_216	BOOL8	SetpointWithinTolerance_2
0x19 hex	Get	s_218	BOOL8	SetpointFeeding_3
0x1A hex	Get	s_219	BOOL8	SetpointFastFeeding_3
0x1B hex	Get	s_21a	BOOL8	SetpointWithinTolerance_3
0x1D hex	Get	s_21c	BOOL8	SetpointFeeding_4
0x1E hex	Get	s_21d	BOOL8	SetpointFastFeeding_4
0x1F hex	Get	s_21e	BOOL8	SetpointWithinTolerance_4
0x21 hex	Get	s_220	BOOL8	SetpointFeeding_5
0x22 hex	Get	s_221	BOOL8	SetpointFastFeeding_5
0x23 hex	Get	s_222	BOOL8	SetpointWithinTolerance_5

Attr ID	Access	Name	Data Type	Description
0x25 hex	Get	s_224	BOOL8	SetpointFeeding_6
0x26 hex	Get	s_225	BOOL8	SetpointFastFeeding_6
0x27 hex	Get	s_226	BOOL8	SetpointWithinTolerance_6
0x29 hex	Get	s_228	BOOL8	SetpointFeeding_7
0x2A hex	Get	s_229	BOOL8	SetpointFastFeeding_7
0x2B hex	Get	s_22a	BOOL8	SetpointWithinTolerance_7
0x2D hex	Get	s_22c	BOOL8	SetpointFeeding_8
0x2E hex	Get	s_22d	BOOL8	SetpointFastFeeding_8
0x2F hex	Get	s_22e	BOOL8	SetpointWithinTolerance_8
0x31 hex	Get	s_230	BOOL8	SetpointFeeding_9
0x32 hex	Get	s_231	BOOL8	SetpointFastFeeding_9
0x33 hex	Get	s_232	BOOL8	SetpointWithinTolerance_9
0x35 hex	Get	s_234	BOOL8	SetpointFeeding_10
0x36 hex	Get	s_235	BOOL8	SetpointFastFeeding_10
0x37 hex	Get	s_236	BOOL8	SetpointWithinTolerance_10
0x39 hex	Get	s_238	BOOL8	SetpointFeeding_11
0x3A hex	Get	s_239	BOOL8	SetpointFastFeeding_11
0x3B hex	Get	s_23a	BOOL8	SetpointWithinTolerance_11
0x3D hex	Get	s_23c	BOOL8	SetpointFeeding_12
0x3E hex	Get	s_23d	BOOL8	SetpointFastFeeding_12
0x3F hex	Get	s_23e	BOOL8	SetpointWithinTolerance_12
0x42 hex	Get	s_241	BOOL8	NodeOnLine_1
0x43 hex	Get	s_242	BOOL8	NodeOnLine_2
0x44 hex	Get	s_243	BOOL8	NodeOnLine_3
0x45 hex	Get	s_244	BOOL8	NodeOnLine_4
0x46 hex	Get	s_245	BOOL8	NodeOnLine_5
0x47 hex	Get	s_246	BOOL8	NodeOnLine_6
0x4B hex	Get	s_24a	BOOL8	PLC Online
0x4E hex	Get	s_24d	BOOL8	HostOnLine_3
0x4F hex	Get	s_24e	BOOL8	HostOnLine_2
0x50 hex	Get	s_24f	BOOL8	HostOnLine_1
0x51 hex	Get	s_250	BOOL8	PLC_CustomStatus1_Scale_A
0x52 hex	Get	s_251	BOOL8	PLC_CustomStatus2_Scale_A
0x53 hex	Get	s_252	BOOL8	PLC_CustomStatus1_Scale_B
0x54 hex	Get	s_253	BOOL8	PLC_CustomStatus2_Scale_B
0x55 hex	Get	s_254	BOOL8	PLC_CustomStatus1_Scale_C
0x56 hex	Get	s_255	BOOL8	PLC_CustomStatus2_Scale_C
0x57 hex	Get	s_256	BOOL8	PLC_CustomStatus1_Scale_D
0x58 hex	Get	s_257	BOOL8	PLC_CustomStatus2_Scale_D
0x59 hex	Get	s_258	BOOL8	PLC_CustomStatus1_Scale_E
0x5A hex	Get	s_259	BOOL8	PLC_CustomStatus2_Scale_E
0x5B hex	Get	s_25a	BOOL8	JagBASIC Custom Status 1
0x5C hex	Get	s_25b	BOOL8	JagBASIC Custom Status 1
0x5D hex	Get	s_25c	BOOL8	JagBASIC Custom Status 1
0x5E hex	Get	s_25d	BOOL8	JagBASIC Custom Status 1
0x5F hex	Get	s_25e	BOOL8	JagBASIC Custom Status 1
0x60 hex	Get	s_25f	BOOL8	JagBASIC Custom Status 1
0x62 hex	Get	s_261	BOOL8	WeightDataOK_A
0x63 hex	Get	s_262	BOOL8	RateSetpointOK_A
0x64 hex	Get	s_263	BOOL8	EstimatedWeight_A
0x65 hex	Get	s_264	BOOL8	PrintingInProgress_A
0x6A hex	Get	s_269	BOOL8	WeightDataOK_B
0x6B hex	Get	s_26a	BOOL8	RateSetpointOK_B

Attr ID	Access	Name	Data Type	Description
0x6C hex	Get	s_26b	BOOL8	EstimatedWeight_B
0x6D hex	Get	s_26c	BOOL8	PrintingInProgress_B
0x71 hex	Get	s_270	BOOL8	MotionOut_C
0x72 hex	Get	s_271	BOOL8	CenterOfZero_C
0x73 hex	Get	s_272	BOOL8	OverCapacity_C
0x74 hex	Get	s_273	BOOL8	UnderZero_C
0x75 hex	Get	s_274	BOOL8	NetMode_C
0x76 hex	Get	s_275	BOOL8	ScaleCriticalError_C
0x77 hex	Get	s_276	BOOL8	StoredWeightMode_C
0x78 hex	Get	s_277	BOOL8	ScaleSelected_C
0x7A hex	Get	s_279	BOOL8	WeightDataOK_C
0x7B hex	Get	s_27a	BOOL8	RateSetpointOK_C
0x7C hex	Get	s_27b	BOOL8	EstimatedWeight_C
0x7D hex	Get	s_27c	BOOL8	PrintingInProgress_C
0x81 hex	Get	s_280	BOOL8	MotionOut_D
0x82 hex	Get	s_281	BOOL8	CenterOfZero_D
0x83 hex	Get	s_282	BOOL8	OverCapacity_D
0x84 hex	Get	s_283	BOOL8	UnderZero_D
0x85 hex	Get	s_284	BOOL8	NetMode_D
0x86 hex	Get	s_285	BOOL8	ScaleCriticalError_D
0x87 hex	Get	s_286	BOOL8	StoredWeightMode_D
0x88 hex	Get	s_287	BOOL8	ScaleSelected_D
0x8A hex	Get	s_289	BOOL8	WeightDataOK_D
0x8B hex	Get	s_28a	BOOL8	RateSetpointOK_D
0x8C hex	Get	s_28b	BOOL8	EstimatedWeight_D
0x8D hex	Get	s_28c	BOOL8	PrintingInProgress_D
0x91 hex	Get	s_290	BOOL8	TareScaleError_A[1]
0x92 hex	Get	s_291	BOOL8	ClearTareScaleError_A[1]
0x93 hex	Get	s_292	BOOL8	PrintScaleError_A[1]
0x94 hex	Get	s_293	BOOL8	ZeroScaleError_A[1]
0x95 hex	Get	s_294	BOOL8	SwitchToPrimUnitsError_A[1]
0x96 hex	Get	s_295	BOOL8	SwitchToSecondUnitsError_A[1]
0x97 hex	Get	s_296	BOOL8	SwitchToOtherUnitsError_A[1]
0x98 hex	Get	s_297	BOOL8	ApplySetupError_A[1]
0x99 hex	Get	s_298	BOOL8	RestartSetpointsError_A[1]
0x9A hex	Get	s_299	BOOL8	RestartRateCalculationError_A[1]
0x9B hex	Get	s_29a	BOOL8	RestartFilterError_A[1]
0x9C hex	Get	s_29b	BOOL8	ResetSetpointCoincidenceError_A[1]
0x9D hex	Get	s_29c	BOOL8	DisableScaleError_A[1]
0x9E hex	Get	s_29d	BOOL8	CaptureRawCountsError_A[1]
0x9F hex	Get	s_29e	BOOL8	WriteCal.ToEEPromError_A[1]
0xA1 hex	Get	s_2a0	BOOL8	TareScaleError_B[1]
0xA2 hex	Get	s_2a1	BOOL8	ClearTareScaleError_B[1]
0xA3 hex	Get	s_2a2	BOOL8	PrintScaleError_B[1]
0xA4 hex	Get	s_2a3	BOOL8	ZeroScaleError_B[1]
0xA5 hex	Get	s_2a4	BOOL8	SwitchToPrimUnitsError_B[1]
0xA6 hex	Get	s_2a5	BOOL8	SwitchToSecondUnitsError_B[1]
0xA7 hex	Get	s_2a6	BOOL8	SwitchToOtherUnitsError_B[1]
0xA8 hex	Get	s_2a7	BOOL8	ApplySetupError_B[1]
0xA9 hex	Get	s_2a8	BOOL8	RestartSetpointsError_B[1]
0xAA hex	Get	s_2a9	BOOL8	RestartRateCalculationError_B[1]
0xAB hex	Get	s_2aa	BOOL8	RestartFilterError_B[1]

Attr ID	Access	Name	Data Type	Description
0xAC hex	Get	s_2ab	BOOL8	ResetSetpointCoincidenceError_B
0xAD hex	Get	s_2ac	BOOL8	DisableScaleError_B[1]
0xAE hex	Get	s_2ad	BOOL8	CaptureRawCountsError_B[1]
0xAF hex	Get	s_2ae	BOOL8	WriteCal.ToEEPromError_B[1]
0xB1 hex	Get	s_2b0	BOOL8	TareScaleError_S[1]
0xB2 hex	Get	s_2b1	BOOL8	ClearTareScaleError_S[1]
0xB3 hex	Get	s_2b2	BOOL8	PrintScaleError_S[1]
0xB4 hex	Get	s_2b3	BOOL8	ZeroScaleError_S[1]
0xB5 hex	Get	s_2b4	BOOL8	SwitchToPrimUnitsError_S[1]
0xB6 hex	Get	s_2b5	BOOL8	SwitchToSecondUnitsError_S[1]
0xB7 hex	Get	s_2b6	BOOL8	SwitchToOtherUnitsError_S[1]
0xB8 hex	Get	s_2b7	BOOL8	CustomPrintError_1
0xB9 hex	Get	s_2b8	BOOL8	CustomPrintError_2
0xBA hex	Get	s_2b9	BOOL8	CustomPrintError_3
0xBB hex	Get	s_2ba	BOOL8	CustomPrintError_4
0xBC hex	Get	s_2bb	BOOL8	CustomPrintError_5
0xBD hex	Get	s_2bc	BOOL8	AlarmOutput
0xC0 hex	Get	s_2bf	BOOL8	JagBasicEnabled
0xC1 hex	Get	s_2c0	BOOL8	TareScaleError_C[1]
0xC2 hex	Get	s_2c1	BOOL8	ClearTareScaleError_C[1]
0xC3 hex	Get	s_2c2	BOOL8	PrintScaleError_C[1]
0xC4 hex	Get	s_2c3	BOOL8	ZeroScaleError_C[1]
0xC5 hex	Get	s_2c4	BOOL8	SwitchToPrimUnitsError_C[1]
0xC6 hex	Get	s_2c5	BOOL8	SwitchToSecondUnitsError_C[1]
0xC7 hex	Get	s_2c6	BOOL8	SwitchToOtherUnitsError_C[1]
0xC8 hex	Get	s_2c7	BOOL8	ApplySetupError_C[1]
0xC9 hex	Get	s_2c8	BOOL8	RestartSetpointsError_C[1]
0xCA hex	Get	s_2c9	BOOL8	RestartRateCalculationError_C[1]
0xCB hex	Get	s_2ca	BOOL8	RestartFilterError_C[1]
0xCC hex	Get	s_2cb	BOOL8	ResetSetpointCoincidenceError_C[1]
0xCD hex	Get	s_2cc	BOOL8	DisableScaleError_C[1]
0xCE hex	Get	s_2cd	BOOL8	CaptureRawCountsError_C[1]
0xCF hex	Get	s_2ce	BOOL8	WriteCal.ToEEPromError_C[1]
0xD1 hex	Get	s_2d0	BOOL8	TareScaleError_D[1]
0xD2 hex	Get	s_2d1	BOOL8	ClearTareScaleError_D[1]
0xD3 hex	Get	s_2d2	BOOL8	PrintScaleError_D[1]
0xD4 hex	Get	s_2d3	BOOL8	ZeroScaleError_D[1]
0xD5 hex	Get	s_2d4	BOOL8	SwitchToPrimUnitsError_D[1]
0xD6 hex	Get	s_2d5	BOOL8	SwitchToSecondUnitsError_D[1]
0xD7 hex	Get	s_2d6	BOOL8	SwitchToOtherUnitsError_D[1]
0xD8 hex	Get	s_2d7	BOOL8	ApplySetupError_D[1]
0xD9 hex	Get	s_2d8	BOOL8	RestartSetpointsError_D[1]
0xDA hex	Get	s_2d9	BOOL8	RestartRateCalculationError_D[1]
0xDB hex	Get	s_2da	BOOL8	RestartFilterError_D[1]
0xDC hex	Get	s_2db	BOOL8	ResetSetpointCoincidenceError_D[1]
0xDD hex	Get	s_2dc	BOOL8	DisableScaleError_D[1]
0xDE hex	Get	s_2dd	BOOL8	CaptureRawCountsError_D[1]
0xDF hex	Get	s_2de	BOOL8	WriteCal.ToEEPromError_D[1]
0xE1 hex	Get	s_2e0	BOOL8	TareScaleError_E[1]
0xE2 hex	Get	s_2e1	BOOL8	ClearTareScaleError_E[1]
0xE3 hex	Get	s_2e2	BOOL8	PrintScaleError_E[1]
0xE4 hex	Get	s_2e3	BOOL8	ZeroScaleError_E[1]
0xE5 hex	Get	s_2e4	BOOL8	SwitchToPrimUnitsError_E[1]

Attr ID	Access	Name	Data Type	Description
0xE6 hex	Get	s_2e5	BOOL8	SwitchToSecondUnitsError_E[1]
0xE7 hex	Get	s_2e6	BOOL8	SwitchToOtherUnitsError_E[1]
0xE8 hex	Get	s_2e7	BOOL8	ApplySetupError_E[1]
0xE9 hex	Get	s_2e8	BOOL8	RestartSetpointsError_E[1]
0xEA hex	Get	s_2e9	BOOL8	RestartRateCalculationError_E[1]
0xEB hex	Get	s_2ea	BOOL8	RestartFilterError_E[1]
0xEC hex	Get	s_2eb	BOOL8	ResetSetpointCoincidenceError_E[1]
0xED hex	Get	s_2ec	BOOL8	DisableScaleError_E[1]
0xEF hex	Get	s_2ee	BOOL8	WriteCal.ToEEPromError_E[1]
0xF1 hex	Get	s_2f0	BOOL8	MotionOut_E
0xF2 hex	Get	s_2f1	BOOL8	CenterOfZero_E
0xF3 hex	Get	s_2f2	BOOL8	OverCapacity_E
0xF4 hex	Get	s_2f3	BOOL8	UnderZero_E
0xF5 hex	Get	s_2f4	BOOL8	NetMode_E
0xF6 hex	Get	s_2f5	BOOL8	ScaleCriticalError_E
0xF7 hex	Get	s_2f6	BOOL8	StoredWeightMode_E
0xF8 hex	Get	s_2f7	BOOL8	ScaleSelected_E
0xFA hex	Get	s_2f9	BOOL8	WeightDataOK_E
0xFB hex	Get	s_2fa	BOOL8	RateSetpointOK_E
0xFC hex	Get	s_2fb	BOOL8	EstimatedWeight_E
0xFD hex	Get	s_2fc	BOOL8	PrintingInProgress_E

Notes:

1. The JAGXTREME operating system sets this field to report success (=0) or error (=1) when an application uses a corresponding discrete field to trigger a command in the JAGXTREME operating system.

Edge Sensitive Discrete Status Object

Class Code

Class Code: 76 hex

The Edge Sensitive Discrete Status Object contains edge sensitive logical discrete I/O data. Edge sensitive bit fields only trigger events when a 1 is written to the field.

Class Attributes

The Edge Sensitive Discrete Status Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1)	Get Attribute Single
0x10	Instance (1)	Set Attribute Single

Instances

The Edge Sensitive Discrete Status Object supports one single instance—Instance 1.

Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01 hex	Get / Set	t_600	BOOL8	MasterControlRelay: 1 = Set physical discrete outputs active, 0 = Set all physical discrete outputs to 0
0x03 hex	Get / Set	t_603	BOOL8	DisableErrorDisplay
0x04 hex	Get / Set	t_604	BOOL8	DisableNumericDisplay
0x06 hex	Get / Set	t_606	BOOL8	Restart Network
0x0A hex	Get / Set	t_60a	BOOL8	DisableSetup
0x0B hex	Get / Set	t_60b	BOOL8	DisableKeypad
0x0C hex	Get / Set	t_60c	BOOL8	IgnoreQWERTY_PositionKeys
0x0E hex	Get / Set	t_60e	BOOL8	DisableRunFlat
0x0F hex	Get / Set	t_60f	BOOL8	Alarm Output Acknowledge
0x13 hex	Get / Set	t_613	BOOL8	WeightUpdated_Cv
0x14 hex	Get / Set	t_614	BOOL8	WeightUpdated_D[1]
0x15 hex	Get / Set	t_615	BOOL8	WeightUpdated_E[1]
0x16 hex	Get / Set	t_616	BOOL8	SetpointInstalled_C[1]
0x17 hex	Get / Set	t_617	BOOL8	SetpointInstalled_D[1]
0x18 hex	Get / Set	t_618	BOOL8	SetpointInstalled_E[1]
0x19 hex	Get / Set	t_619	BOOL8	CalibrationComplete_C[1]
0x1A hex	Get / Set	t_61a	BOOL8	CalibrationComplete_D[1]
0x1B hex	Get / Set	t_61b	BOOL8	CalibrationComplete_E[1]
0x1C hex	Get / Set	t_61c	BOOL8	StartSetpointJogTimer[1]
0x1D hex	Get / Set	t_61d	BOOL8	StartPromptList[1]
0x1E hex	Get / Set	t_61e	BOOL8	Request to Stop Jag Basic Program[1]
0x1F hex	Get	t_61f	BOOL8	Setup Started by Web Browser[1]
0x20 hex	Get / Set	t_620	BOOL8	TareScale_C[2]
0x21 hex	Get / Set	t_621	BOOL8	ClearTareScale_C[2]
0x22 hex	Get / Set	t_622	BOOL8	PrintScale_C[2]
0x23 hex	Get / Set	t_623	BOOL8	ZeroScale_C[2]
0x24 hex	Get / Set	t_624	BOOL8	SwitchToPrimaryUnits_C[2]
0x25 hex	Get / Set	t_625	BOOL8	SwitchToSecondUnits_C[2]
0x26 hex	Get / Set	t_626	BOOL8	SwitchToOtherUnits_C[2]
0x27 hex	Get / Set	t_627	BOOL8	ApplySetup_B[2]

Attr ID	Access	Name	Data Type	Description
0x28 hex	Get / Set	t_628	BOOL8	RestartSetpoints_C[2]
0x29 hex	Get / Set	t_629	BOOL8	RestartRateCalculation_C[2]
0x2A hex	Get / Set	t_62a	BOOL8	RestartFilter_C[2]
0x2B hex	Get / Set	t_62b	BOOL8	ResetSetpointCoincidence_C[2]
0x2C hex	Get / Set	t_62c	BOOL8	DisableScale_Cv
0x2D hex	Get / Set	t_62d	BOOL8	CaptureRawCounts_C[2]
0x2E hex	Get / Set	t_62e	BOOL8	WriteCalibrationToEEProm_C[2]
0x2F hex	Get / Set	t_62f	BOOL8	ResetPredictiveFailures[2]
0x30 hex	Get / Set	t_630	BOOL8	TareScale_D[2]
0x31 hex	Get / Set	t_631	BOOL8	ClearTareScale_D[2]
0x32 hex	Get / Set	t_632	BOOL8	PrintScale_D[2]
0x33 hex	Get / Set	t_633	BOOL8	ZeroScale_D[2]
0x34 hex	Get / Set	t_634	BOOL8	SwitchToPrimaryUnits_D[2]
0x35 hex	Get / Set	t_635	BOOL8	SwitchToSecondUnits_D[2]
0x36 hex	Get / Set	t_636	BOOL8	SwitchToOtherUnits_D[2]
0x37 hex	Get / Set	t_637	BOOL8	ApplySetup_D[2]
0x38 hex	Get / Set	t_638	BOOL8	RestartSetpoints_D[2]
0x39 hex	Get / Set	t_639	BOOL8	RestartRateCalculation_D[2]
0x3A hex	Get / Set	t_63a	BOOL8	RestartFilter_D[2]
0x3B hex	Get / Set	t_63b	BOOL8	ResetSetpointCoincidence_D[2]
0x3C hex	Get / Set	t_63c	BOOL8	DisableScale_D[2]
0x3D hex	Get / Set	t_63d	BOOL8	CaptureRawCounts_D[2]
0x3E hex	Get / Set	t_63e	BOOL8	WriteCalibrationToEEProm_D[2]
0x3F hex	Get / Set	t_63f	BOOL8	ResetPredictiveFailures[2]
0x40 hex	Get / Set	t_640	BOOL8	TareScale_E[2]
0x41 hex	Get / Set	t_641	BOOL8	ClearTareScale_E[2]
0x42 hex	Get / Set	t_642	BOOL8	PrintScale_E[2]
0x43 hex	Get / Set	t_643	BOOL8	ZeroScale_E[2]
0x44 hex	Get / Set	t_644	BOOL8	SwitchToPrimaryUnits_E[2]
0x45 hex	Get / Set	t_645	BOOL8	SwitchToSecondUnits_E[2]
0x46 hex	Get / Set	t_646	BOOL8	SwitchToOtherUnits_E[2]
0x47 hex	Get / Set	t_647	BOOL8	ApplySetup_E[2]
0x48 hex	Get / Set	t_648	BOOL8	RestartSetpoints_E[2]
0x49 hex	Get / Set	t_649	BOOL8	RestartRateCalculation_E[2]
0x4A hex	Get / Set	t_64a	BOOL8	RestartFilter_E[2]
0x4B hex	Get / Set	t_64b	BOOL8	ResetSetpointCoincidence_E[2]
0x4C hex	Get / Set	t_64c	BOOL8	DisableScale_E[2]
0x4D hex	Get / Set	t_64d	BOOL8	CaptureRawCounts_E[2]
0x50 hex	Get / Set	t_650	BOOL8	SelectScale_C
0x51 hex	Get / Set	t_651	BOOL8	SelectScale_D
0x52 hex	Get / Set	t_652	BOOL8	SelectScale_E
0x56 hex	Get / Set	t_656	BOOL8	EmailAlertMessage_1
0x57 hex	Get / Set	t_657	BOOL8	EmailAlertMessage_2
0x58 hex	Get / Set	t_658	BOOL8	EmailAlertMessage_3
0x59 hex	Get / Set	t_659	BOOL8	EmailAlertMessage_4
0x88 hex	Get / Set	t_688	BOOL8	WeightUpdated_A[1]
0x89 hex	Get / Set	t_689	BOOL8	WeightUpdated_B[1]
0x8B hex	Get / Set	t_68b	BOOL8	Web Pages Disable Error Display[1]
0x8C hex	Get / Set	t_68c	BOOL8	SetpointInstalled_A[1]
0x8D hex	Get / Set	t_68d	BOOL8	SetpointInstalled_B[1]
0x8E hex	Get / Set	t_68e	BOOL8	CalibrationComplete_A[1]
0x8F hex	Get / Set	t_68f	BOOL8	CalibrationComplete_B[1]

Attr ID	Access	Name	Data Type	Description
0x90 hex	Get / Set	t_690	BOOL8	TareScale_A[2]
0x91 hex	Get / Set	t_691	BOOL8	ClearTareScale_A[2]
0x92 hex	Get / Set	t_692	BOOL8	PrintScale_A[2]
0x93 hex	Get / Set	t_693	BOOL8	ZeroScale_A[2]
0x94 hex	Get / Set	t_694	BOOL8	SwitchToPrimaryUnits_A[2]
0x95 hex	Get / Set	t_695	BOOL8	SwitchToSecondUnits_A[2]
0x96 hex	Get / Set	t_696	BOOL8	SwitchToOtherUnits_A[2]
0x97 hex	Get / Set	t_697	BOOL8	ApplySetup_A[2]
0x98 hex	Get / Set	t_698	BOOL8	RestartSetpoints_A[2]
0x99 hex	Get / Set	t_699	BOOL8	RestartRateCalculation_A[2]
0x9A hex	Get / Set	t_69a	BOOL8	RestartFilter_A[2]
0x9B hex	Get / Set	t_69b	BOOL8	ResetSetpointCoincidence_A[2]
0x9C hex	Get / Set	t_69c	BOOL8	DisableScale_A[2]
0x9D hex	Get / Set	t_69d	BOOL8	CaptureRawCounts_A[2]
0x9E hex	Get / Set	t_69e	BOOL8	WriteCalibrationToEEProm_A[2]
0x9F hex	Get / Set	t_69f	BOOL8	ResetPredictiveFailures[2]
0xA0 hex	Get / Set	t_6a0	BOOL8	TareScale_B[2]
0xA1 hex	Get / Set	t_6a1	BOOL8	ClearTareScale_B[2]
0xA2 hex	Get / Set	t_6a2	BOOL8	PrintScale_B[2]
0xA3 hex	Get / Set	t_6a3	BOOL8	ZeroScale_B[2]
0xA4 hex	Get / Set	t_6a4	BOOL8	SwitchToPrimaryUnits_B[2]
0xA5 hex	Get / Set	t_6a5	BOOL8	SwitchToSecondUnits_B[2]
0xA6 hex	Get / Set	t_6a6	BOOL8	SwitchToOtherUnits_B[2]
0xA7 hex	Get / Set	t_6a7	BOOL8	ApplySetup_B[2]
0xA8 hex	Get / Set	t_6a8	BOOL8	RestartSetpoints_B[2]
0xA9 hex	Get / Set	t_6a9	BOOL8	RestartRateCalculation_B[2]
0xAA hex	Get / Set	t_6aa	BOOL8	RestartFilter_B[2]
0xAB hex	Get / Set	t_6ab	BOOL8	ResetSetpointCoincidence_B[2]
0xAC hex	Get / Set	t_6ac	BOOL8	DisableScale_B[2]
0xAD hex	Get / Set	t_6ad	BOOL8	CaptureRawCounts_B[2]
0xAE hex	Get / Set	t_6ae	BOOL8	WriteCalibrationToEEProm_B[2]
0xAF hex	Get / Set	t_6af	BOOL8	ResetPredictiveFailures[2]
0xB0 hex	Get / Set	t_6b0	BOOL8	TareScale_S[2]
0xB1 hex	Get / Set	t_6b1	BOOL8	ClearTareScale_S[2]
0xB2 hex	Get / Set	t_6b2	BOOL8	PrintScale_S[2]
0xB3 hex	Get / Set	t_6b3	BOOL8	ZeroScale_S[2]
0xB4 hex	Get / Set	t_6b4	BOOL8	SwitchToPrimaryUnits_S[2]
0xB5 hex	Get / Set	t_6b5	BOOL8	SwitchToSecondUnits_S[2]
0xB6 hex	Get / Set	t_6b6	BOOL8	SwitchToOtherUnits_S[2]
0xC0 hex	Get / Set	t_6c0	BOOL8	SelectScale_A
0xC1 hex	Get / Set	t_6c1	BOOL8	SelectScale_B
0xC2 hex	Get / Set	t_6c2	BOOL8	SelectOtherScale
0xC3 hex	Get / Set	t_6c3	BOOL8	DemandCustomPrint_1
0xC4 hex	Get / Set	t_6c4	BOOL8	DemandCustomPrint_2
0xC5 hex	Get / Set	t_6c5	BOOL8	DemandCustomPrint_3
0xC6 hex	Get / Set	t_6c6	BOOL8	DemandCustomPrint_4
0xC7 hex	Get / Set	t_6c7	BOOL8	DemandCustomPrint_5
0xCC hex	Get / Set	t_6cc	BOOL8	CustomCommand1
0xCD hex	Get / Set	t_6cd	BOOL8	CustomCommand2
0xCE hex	Get / Set	t_6ce	BOOL8	CustomCommand3
0xCF hex	Get / Set	t_6cf	BOOL8	CustomCommand4
0xE0 hex	Get / Set	p_6e0	BOOL8	DiscreteInputRisingEdge_1[1]
0xE1 hex	Get / Set	p_6e1	BOOL8	DiscreteInputRisingEdge_2[1]

Attr ID	Access	Name	Data Type	Description
0xE2 hex	Get / Set	p_6e2	BOOL8	DiscreteInputRisingEdge_3[1]
0xE3 hex	Get / Set	p_6e3	BOOL8	DiscreteInputRisingEdge_4[1]
0xE4 hex	Get / Set	p_6e4	BOOL8	DiscreteInputRisingEdge_5[1]
0xE5 hex	Get / Set	p_6e5	BOOL8	DiscreteInputRisingEdge_6[1]
0xE6 hex	Get / Set	p_6e6	BOOL8	DiscreteInputRisingEdge_7[1]
0xE7 hex	Get / Set	p_6e7	BOOL8	DiscreteInputRisingEdge_8[1]
0xE8 hex	Get / Set	p_6e8	BOOL8	DiscreteInputRisingEdge_9[1]
0xE9 hex	Get / Set	p_6e9	BOOL8	DiscreteInputRisingEdge_10[1]
0xEA hex	Get / Set	p_6ea	BOOL8	DiscreteInputRisingEdge_11[1]
0xEB hex	Get / Set	p_6eb	BOOL8	DiscreteInputRisingEdge_12[1]
0xF0 hex	Get / Set	p_6f0	BOOL8	DiscreteInputFallingEdge_1[1]
0xF1 hex	Get / Set	p_6f1	BOOL8	DiscreteInputFallingEdge_2[1]
0xF2 hex	Get / Set	p_6f2	BOOL8	DiscreteInputFallingEdge_3[1]
0xF3 hex	Get / Set	p_6f3	BOOL8	DiscreteInputFallingEdge_4[1]
0xF4 hex	Get / Set	p_6f4	BOOL8	DiscreteInputFallingEdge_5[1]
0xF5 hex	Get / Set	p_6f5	BOOL8	DiscreteInputFallingEdge_6[1]
0xF6 hex	Get / Set	p_6f6	BOOL8	DiscreteInputFallingEdge_7[1]
0xF7 hex	Get / Set	p_6f7	BOOL8	DiscreteInputFallingEdge_8[1]
0xF8 hex	Get / Set	p_6f8	BOOL8	DiscreteInputFallingEdge_9[1]
0xF9 hex	Get / Set	p_6f9	BOOL8	DiscreteInputFallingEdge_10[1]
0xFA hex	Get / Set	p_6fa	BOOL8	DiscreteInputFallingEdge_11[1]
0xFB hex	Get / Set	p_6fb	BOOL8	DiscreteInputFallingEdge_12[1]

Notes:

1. The JAGXTREME O/S sets these discrete triggers to indicate an event. An application must set the reset the field to 0 before the same event will trigger again.
2. Applications can set these fields to trigger a command within the JAGXTREME O/S. The JAGXTREME O/S will set the field to 0 when it is done processing the command. It will set a corresponding error bit to indicate when there is an error in processing the command.

Level Sensitive Physical Discrete I/O Output Object

Class Code

Class Code: 77hex

The Level Sensitive Physical Discrete I/O Output Object contains:

Class Attributes

The Level Sensitive Physical Discrete I/O Output Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1)	Get Attribute Single
0x10	Instance (1)	Set Attribute Single

Instances

Only one instance of the Level Sensitive Physical Discrete I/O Output Object is supported: Instance 1.

Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01 hex	Get / Set	p_500	BOOL8	PhysicalDiscreteOutput_1
0x02 hex	Get / Set	p_501	BOOL8	PhysicalDiscreteOutput_2
0x03 hex	Get / Set	p_502	BOOL8	PhysicalDiscreteOutput_3
0x04 hex	Get / Set	p_503	BOOL8	PhysicalDiscreteOutput_4
0x09 hex	Get / Set	p_508	BOOL8	PhysicalDiscreteOutput_5
0x0A hex	Get / Set	p_509	BOOL8	PhysicalDiscreteOutput_6
0x0B hex	Get / Set	p_50a	BOOL8	PhysicalDiscreteOutput_7
0x0C hex	Get / Set	p_50b	BOOL8	PhysicalDiscreteOutput_8
0x0D hex	Get / Set	p_50c	BOOL8	PhysicalDiscreteOutput_9
0x0E hex	Get / Set	p_50d	BOOL8	PhysicalDiscreteOutput_10
0x0F hex	Get / Set	p_50e	BOOL8	PhysicalDiscreteOutput_11
0x10 hex	Get / Set	p_50f	BOOL8	PhysicalDiscreteOutput_12

Level Sensitive Physical Discrete I/O Input Object

Class Code

Class Code: 78hex

The Level Sensitive Physical Discrete I/O Input Object contains:

Class Attributes

The Level Sensitive Physical Discrete I/O Input Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance	Get Attribute Single
0x10	Instance	Set Attribute Single

Instances

Only Instance 1 of the Level Sensitive Physical Discrete I/O Input Object is supported.

Instance Attributes

Attr ID	Access	Name	Data Type	Description
0x01 hex	Get	p_100	BOOL8	PhysicalDiscreteInput_1
0x02 hex	Get	p_101	BOOL8	PhysicalDiscreteInput_2
0x03 hex	Get	p_102	BOOL8	PhysicalDiscreteInput_3
0x04 hex	Get	p_103	BOOL8	PhysicalDiscreteInput_4
0x09 hex	Get	p_108	BOOL8	PhysicalDiscreteInput_5
0x0A hex	Get	p_109	BOOL8	PhysicalDiscreteInput_6
0x0B hex	Get	p_10a	BOOL8	PhysicalDiscreteInput_7
0x0C hex	Get	p_10b	BOOL8	PhysicalDiscreteInput_8
0x0D hex	Get	p_10c	BOOL8	PhysicalDiscreteInput_9
0x0E hex	Get	p_10d	BOOL8	PhysicalDiscreteInput_10
0x0F hex	Get	p_10e	BOOL8	PhysicalDiscreteInput_11
0x10 hex	Get	p_10f	BOOL8	PhysicalDiscreteInput_12

JagBASIC Remote Batching Variables

Class Code

Class Code: 79 hex

The JagBASIC Remote Batching Variables Object provides access to JagBASIC remote batching variables

Class Attributes

The JagBASIC Remote Batching Variables Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E	Instance (1,2)	Get Attribute Single
0x10	Instance (1,2)	Set Attribute Single

Instances

The JagBASIC Remote Batching Variables Object has 2 supported instances, which provide access to up to 198 remote batching variables.

Instance 1 Attributes

Attr ID	Access ☐	Name	Data Type	Description
0x01 hex	Get / Set	bx101	UINT8	Remote Batching Variable 1, Also Autotune Setup State.
0x02 hex	Get / Set	bx102	UINT8	Remote Batching Variable 2, Also Autotune Setup State.
0x03 hex	Get / Set	bx103	UINT8	Remote Batching Variable 3, Also Autotune Setup State.
0x04 hex	Get / Set	bx104	UINT8	Remote Batching Variable 4, Also Autotune Setup State.
0x05 hex	Get / Set	bx105	UINT8	Remote Batching Variable 5, Also Autotune Setup State.
0x06 hex through 0x62 hex	Get / Set	bx106 through bx198	UINT8	Remote Batching Variables 6 through 98

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the 'off' position.

Instance 2 Attributes

Attr ID	Access ☐	Name	Data Type	Description
0x01 hex through 0x62 hex	Get / Set	bx201 through bx298	UINT8	Remote Batching Variables 99 through 198

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the 'off' position.

Scale Calibration - EEPROM Object

Class Code

Class Code: 7e hex

The Scale Calibration EEPROM Object contains calibration data for a given scale.

Class Attributes

The Scale Calibration EEPROM Object does not support any class attributes.

Common Services

Service Code	Class/Instance Usage	Service Name
0x0E hex	Instance (1 thru 5)	Get Attribute Single
0x10 hex	Instance (1 thru 5)	Set Attribute Single

Instances

One instance of the Scale Calibration EEPROM Object will exist for each of up to five possible scales connected to a given JAGXTREME terminal either directly or indirectly.

Instance Attributes

Attr ID	Access [1]	Name[2]	Data Type	Description
0x01 hex	Get	cen01	UINT8	Address Of First Load Cell
0x02 hex	Get	cen02	UINT8	Number of Load Cells
0x03 hex	Get	cen03	UINT8	Primary Units: 1 = pounds 2 = kilograms 3 = grams 4 = metric tons
0x04 hex	Get	cen04	CHAR	Primary Number Ranges
0x05 hex	Get	cen05	DOUBLE	Primary Low Increment Size
0x06 hex	Get	cen06	DOUBLE	Primary Mid Increment Size
0x07 hex	Get	cen07	DOUBLE	Primary High increment Size
0x08 hex	Get	cen08	DOUBLE	Primary Low Mid Threshold
0x09 hex	Get	cen09	DOUBLE	Primary Mid High Threshold
0x0A hex	Get	cen10	DOUBLE	Primary Scale Capacity
0x0B hex	Get	cen11	UINT8	Secondary Units: 1 = pounds 2 = kilograms 3 = grams 4 = metric tons
0x0C hex	Get	cen12	CHAR	Secondary Number Ranges
0x0D hex	Get	cen13	DOUBLE	Secondary Low Increment Size
0x0E hex	Get	cen14	DOUBLE	Secondary Mid Increment Size
0x0F hex	Get	cen15	DOUBLE	Secondary High Increment Size
0x10 hex	Get	cen16	DOUBLE	Secondary Low Mid Threshold
0x11 hex	Get	cen17	DOUBLE	Secondary Mid High Threshold
0x12 hex	Get	cen18	DOUBLE	Secondary Scale Capacity
0x13 hex	Get	cen19	CHAR	Calibration Units: 1 = primary 2 = secondary
0x14 hex	Get	cen20	UINT32	Zero Calibration Counts
0x15 hex	Get	cen21	UINT32	High Calibration Counts

Attr ID	Access [1]	Name[2]	Data Type	Description
0x16 hex	Get	cen22	DOUBLE	High Calibration Weight
0x17 hex	Get	cen23	UINT32	Mid Calibration Counts
0x18 hex	Get	cen24	DOUBLE	Mid Calibration Weight
0x19 hex	Get	cen25	DOUBLE	Gravity Adjust
0x1A hex	Get	cen26	FLOAT	Motion Stability Sensitivity in D
0x1B hex	Get	cen27	UINT8	Motion Stability Time Period: (1 = 3 sec, ..., 7 = 10sec)
0x1C hex	Get	cen28	ASCII[13]	Scale Serial Number
0x1D hex	Get	cen29	UINT8	Calibration Counter 1
0x1E hex	Get	cen30	UINT8	Calibration Counter 2
0x1F hex	Get	cen31	UINT8	A to D Update Rate
0x20 hex	Get	cen32	UINT8	Over Capacity Divisions
0x55 hex	Get	cen85	BOOL8	Linearity Correction Enable
0x56 hex	Get	cen86	BOOL8	Over Capacity Blanking
0x57 hex	Get	cen87	BOOL8	Multirange Mode
0x58 hex	Get	cen88	UINT8	Shift Adjust Mode: 0 = Cell 1 = Pair

Notes:

1. Settable only when the JAGXTREME terminal's Legal-for-Trade switch is in the OFF position.
2. The character 'n' represents the internal scale number (1-5). Right justified, null terminated string.

Setting up Scheduled Messaging to the JAGXTREME Terminal

Scheduled Messaging (for EtherNet I/P or ControlNet Networks)

This section provides the user with an example of how to set up a Logix 5550 controller for scheduled messaging to a JAGXTREME terminal. The network used in the example is EtherNet I/P, but the techniques and steps used in setting up messaging are similar for ControlNet networks as well. Where significant differences exist between the set up procedure for the two networks, they will be noted during the course of the example.

To set up scheduled messaging to the JAGXTREME terminal:

- Configure the JAGXTREME terminal's CIP Network Interface card.
- Configure the controller to initiate an I/O connection to the JAGXTREME terminal.
- Schedule the I/O connection (ControlNet only).

Configuring the JAGXTREME terminal's CIP Network Interface Card

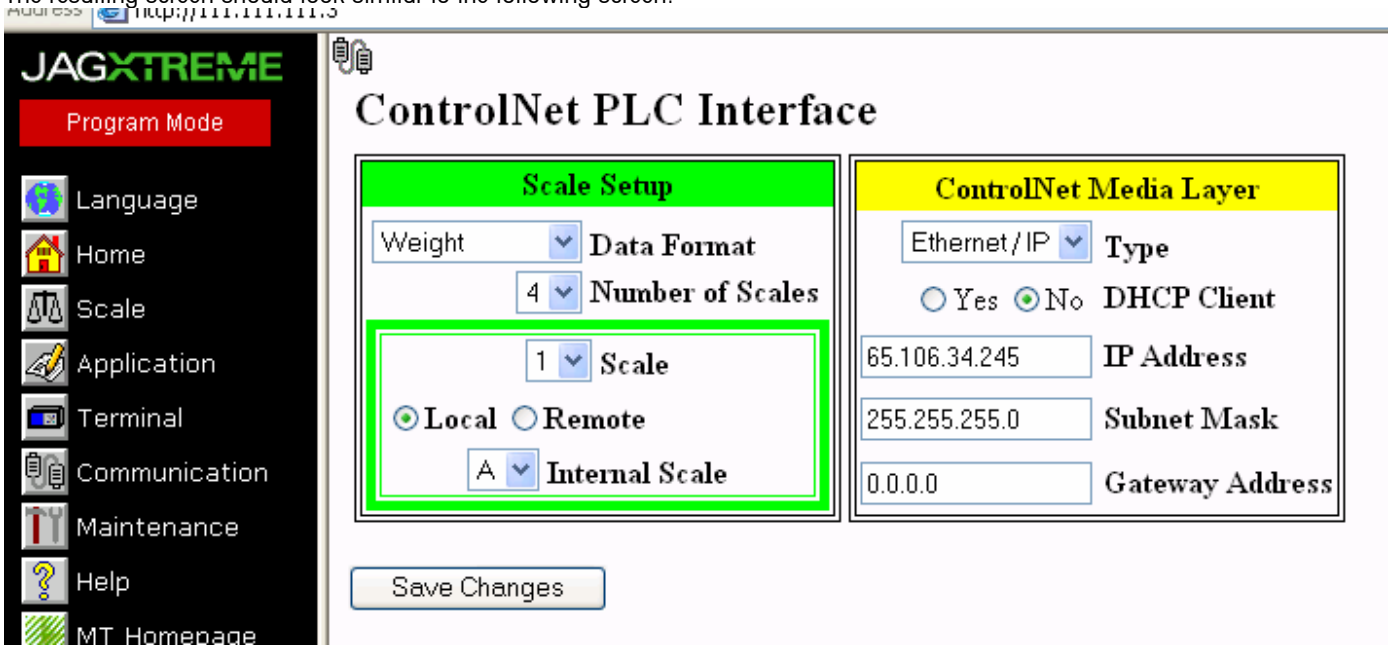
Before communications can take place to the JAGXTREME terminal's CIP Network Interface Card, the card must be configured in terms of:

- The type of network being used.
- The network address (IP or MAC) of the CIP Network Interface Card.
- The format of the I/O data the JAGXTREME should transmit.

All of these tasks can be accomplished via the CIP PLC Interface page of the JAGXTREME terminal's built in web server. To get there, simply browse the JAGXTREME terminal's internal web page using any popular web browser using the IP default or previously configured IP address. From the home page:

- Select **Communications** from the left hand panel.
- Select **CIP PLC Communications** from the list presented in the right portion of the screen.
- Press the **Program Mode** button in the left column of the screen in order to enable the controls on the **CIP PLC Communications** screen presented in the right hand portion of your screen.

The resulting screen should look similar to the following screen.



CIP Network Interface Configuration Screen

The CIP Network PLC Interface screen provides two forms side by side. Once a selection is made, press the **Save Changes** button beneath the forms to save the settings within the JAGXTREME terminal and then press the **Run Mode** button in the left hand pane to return the scale to the run mode.

The Scale Setup Form

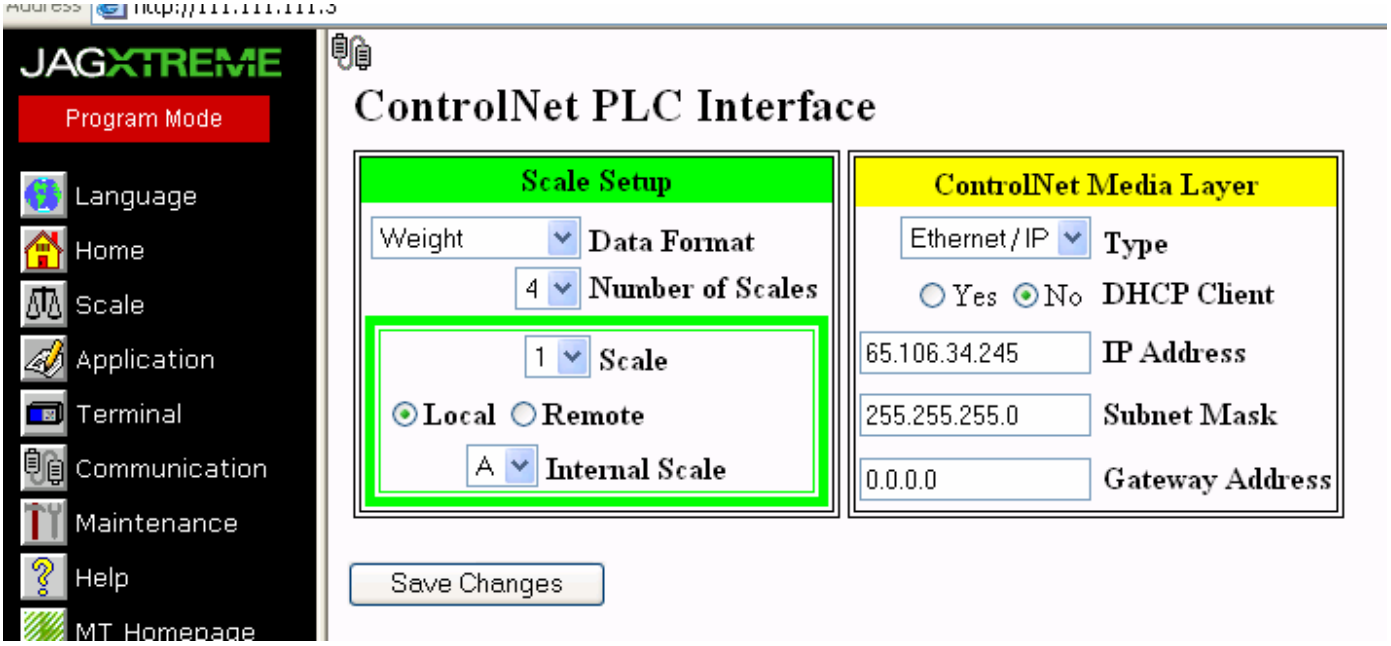
The Scale Setup Form allows you to select the format of the data you would like the JAGXTREME terminal to transmit and receive via scheduled messaging, as well as the number of scales to be included in the JAGXTREME terminal's I/O message data. The Data Format choices are Weight, Divisions, Extended and Floating Point. If Weight or Divisions is selected, then the numerical data included in the scheduled message will be Integers.

Note: The Data Format selection made here determines the exact Assembly Instances and data size to use when configuring the Logix 5000.

The lower portion of the form allows you to specify the way each scale is connected to the JAGXTREME terminal (i.e. Local –connected to the CIP network, or Remote-connected to JAGXTREME cluster network). A scheduled message contains an array of scale information. Since four scales may be selected there are four elements within the array. These elements are referred to as "Slots". Slot 1 contains Scale 1 data, Slot 2 contains Scale 2 data,etc. The types of information contained within a scheduled message is dependent on the Data Format selected.

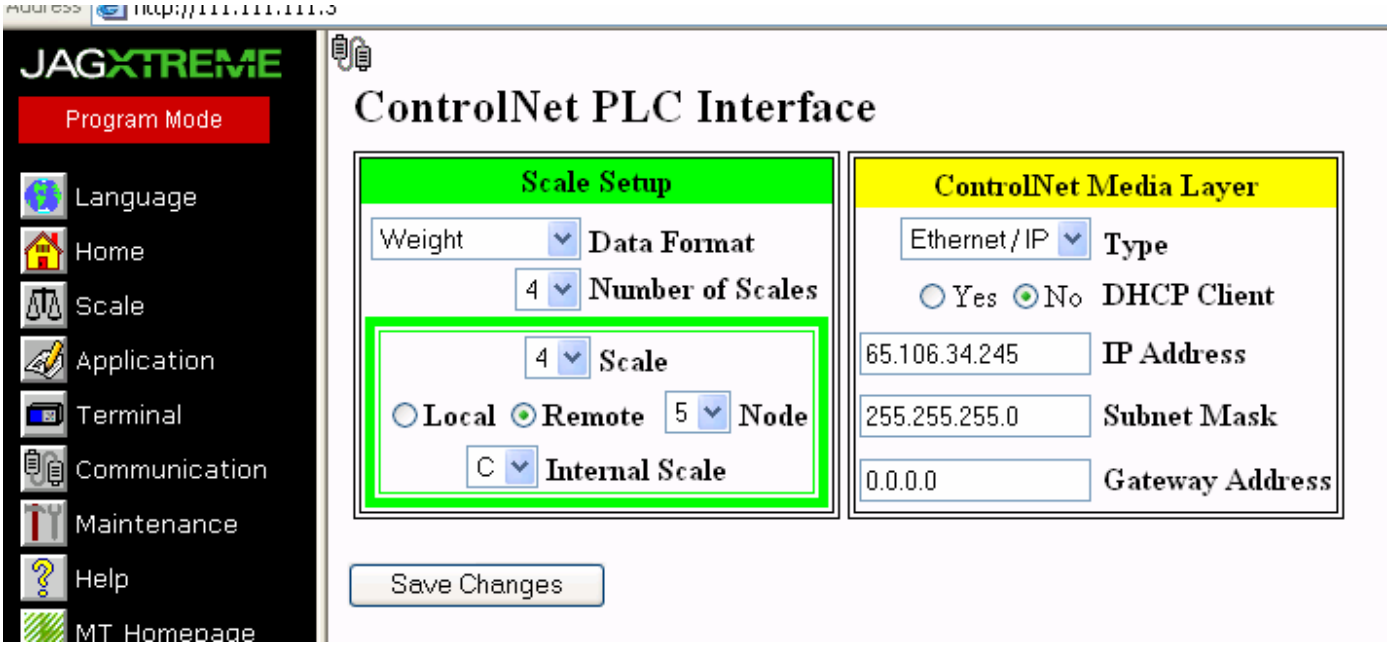
Each scale that is part of the scheduled message must be mapped to a respective internal scale.

There will be four slots of data even if there is only one scale.



Local Scale Connection

The diagram above shows Scale 1 of the scheduled message mapped to Internal Scale A. Scale 1 could be mapped to any one of the Local Internal Scales.



Remote Scale Connection

In the Remote Scale Connection diagram shown, there are 4 scales in the scheduled message. Weight data from "Scale C" of the JAGXTREME at Cluster Node 5 is mapped to the Scale 4 slot of the scheduled message.

The CIP Network Media Layer Form

The CIP Network Media Layer Form allows you to choose either Ethernet/IP or ControlNet as the underlying network media. Once you have chosen a media layer, the remainder of the form changes to provide you with access to the network address data appropriate for the media layer chosen.

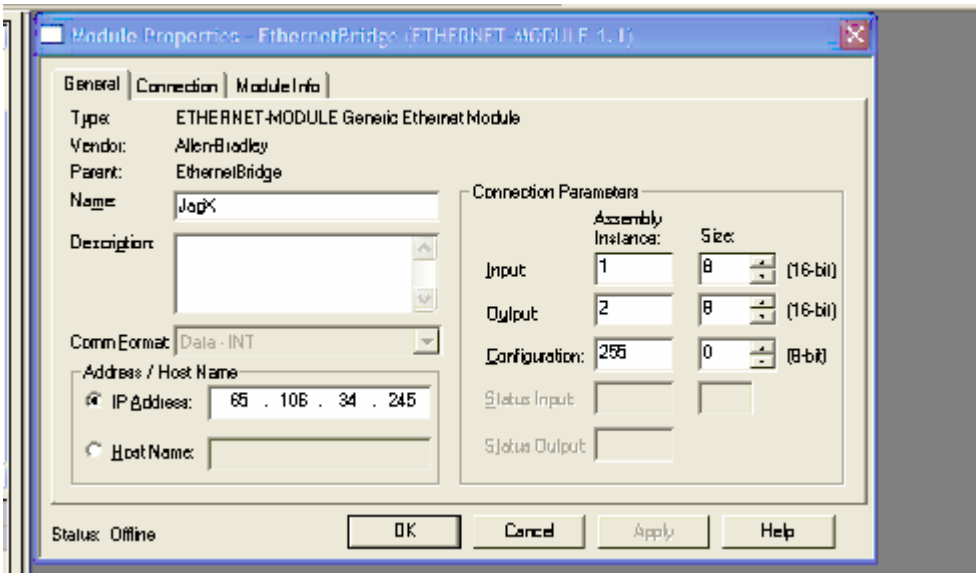
Configuring the Logix Controller

In order for the Logix controller to be able to open and maintain an I/O connection with the JAGXTREME terminal, the connection must be configured within the Logix controller.

To set up the connection:

- Open the Logix 5000 program file and right clicking on the appropriate Ethernet bridge within the controller's I/O configuration tree.
- Select **New Module** from the pop-up menu.
- Select **Generic Ethernet Module** from the Select Module Type dialog box and press **OK**.

The resulting dialog box should look something like the one shown in the following figure.



Module Properties-General Page

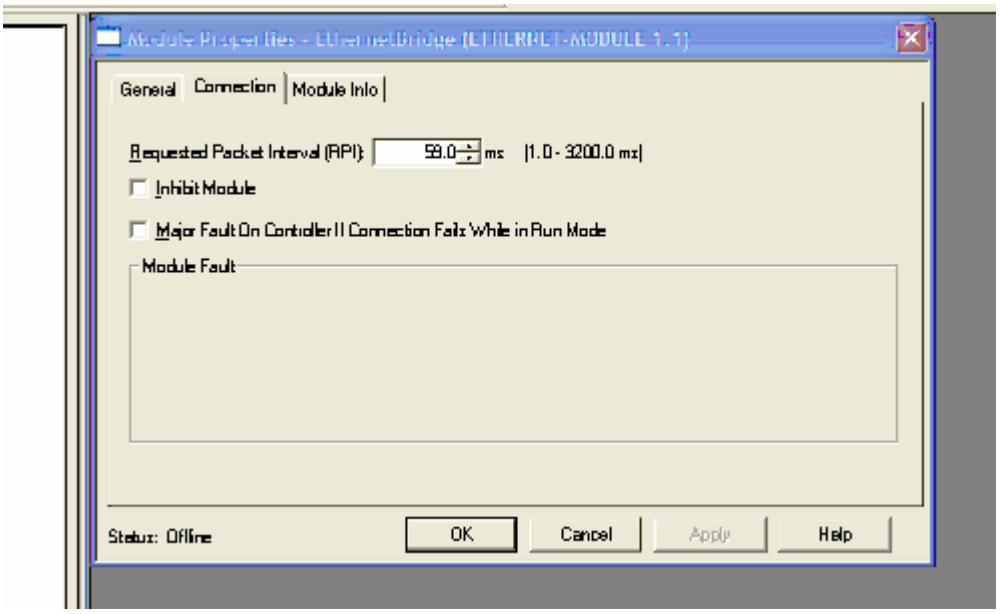
The dialog box shown in the previous figure allows you to configure the I/O connection to the JAGXTREME. The Instances and Sizes shown came from the previous table and are based on the choice of data format made earlier when we configured the CIP Network Interface Card from the JAGXTREME terminal's configuration web page.

Notice that the sizes entered within the Connection Parameters of the dialog box are in INT, since that is the Comm Format selected. See the following diagram for size information.

Data Format	Input Instance	Size (Bytes)	Output Instance	Size (Bytes)
Weight	1	16	2	16
Divisions	1	16	2	16
Extended	9	16	2	16
Floating Point	3	32	4	26

Data Format to Assembly Instance Mappings

Note: The JAGXTREME terminal does not use an Assembly Configuration Instance, so it does not matter what number is entered for that selection as long as the size is 0. Once you have entered your selections into the dialog box shown above, click the **Connection** button. The resulting dialog box should look something like the following:



Module Properties-Connection Page

Specifying an Requested Packet Interval (RPI)

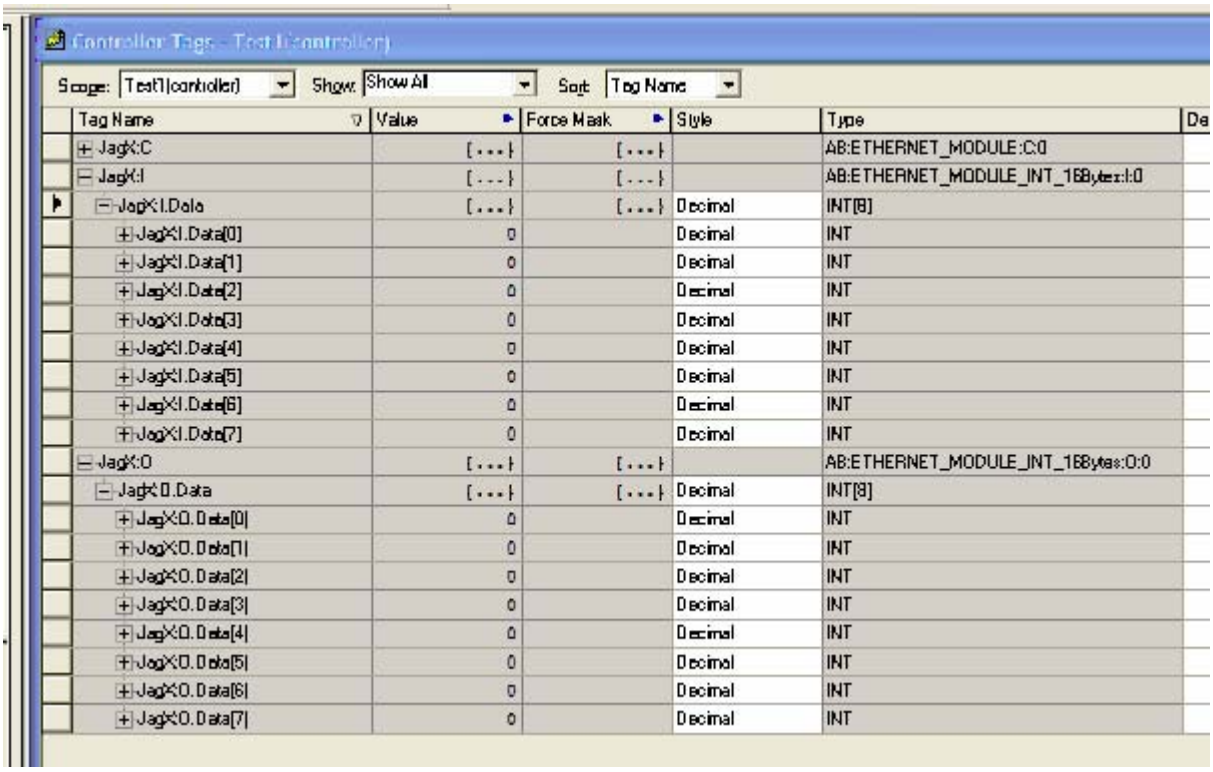
The Module Properties dialog box allows you to select the Requested Packet Interval (RPI) for the connection. This is the interval, in milliseconds, at which the Logix controller and the JAGXTREME terminal will exchange data over the I/O connection.

Note: Although the JAGXTREME terminal supports any RPI selected. The scale weight data it reports is only updated at the rate of 17Hz.

Once an RPI selection has been made, press the **Finish** button to complete the configuration of the I/O connection.

Input and Output Tags

Note that after the I/O connection has been configured, the RSLOGIX program will automatically create the Input and Output arrays tags for the connection.
For this example the tags will be identified as Jagx.I.Data and Jagx.O.Data. See the following for an example of these tags.



Running the Connection

Now that the connection has been configured, it is ready to be exercised. To accomplish this, download the program into the Logix controller.

ControlNet Only

If you are using a ControlNet network, the network must be scheduled using Rockwell Software's RS Networx for ControlNet. This step is not required for Ethernet/IP networks. For details on how to use RSNetworx for ControlNet, refer to the RSNetworx user manual and online help system, or contact Rockwell Software for support.

Note: The Logix controller does not need to be in Run mode for the connection to operate. The connection will open and run as soon as you complete downloading it to the controller.

If you would like to stop the connection, simply go back to the dialog box shown in the previous figure (right-click the JAGXTREME module in the controller's I/O tree and select Properties from the pop-up menu) and click the Inhibit check box.

Working With JAGXTREME I/O Data

While previous sections of this manual detailed the exact format of the data contained in each of the JAGXTREME terminal's objects and instances, it is often preferable within a Logix 5000 ladder program to be able to reference individual data items by symbolic name rather than as offsets into a block of data. The following section explains how to convert JAGXTREME I/O data into a format that is easier to use within the Logix 5000 controller.

Data Types

In Logix 5000 programming, tags are named data items that may consist of a variety of data types: from a single INT, to arrays of structures containing a variety of mixed data types. While the most basic data types are defined by Logix 5000, the user is also free to define new data types to suit theirown needs.

Using User Defined Data Types

The use of special user-defined data types to represent JAGXTREME I/O data is an excellent way to provide clear, efficient access to JAGXTREME I/O data within a Logix 5000 ladder program. In order to make use of user-defined data types to represent JAGXTREME I/O data, the user must:

- Determine the structure of the data type.
- Create the data type.
- Create a tag of that type.
- Put data into the tag

Determining the Structure of the Data Type

The JAGXTREME terminal's assembly instances present the data from several scales arranged into Slots. Accordingly, the data can be viewed as an array of 4 identical user-defined data types rather than a single large data type containing repetitively named fields. For example, the Integer Input data from Assembly Instance 1 might be represented as follows:

Tag Name	Data Type
MyJagData	IntInputSlotType[4]

Where:

MyJagData is a tag we create to hold the Integer Input data from Assembly Instance 1 of the JAGXTREME terminal.

IntInputSlotType is a user-defined data type that exactly duplicates the structure of a single Slot of integer input data from the JAGXTREME terminal.

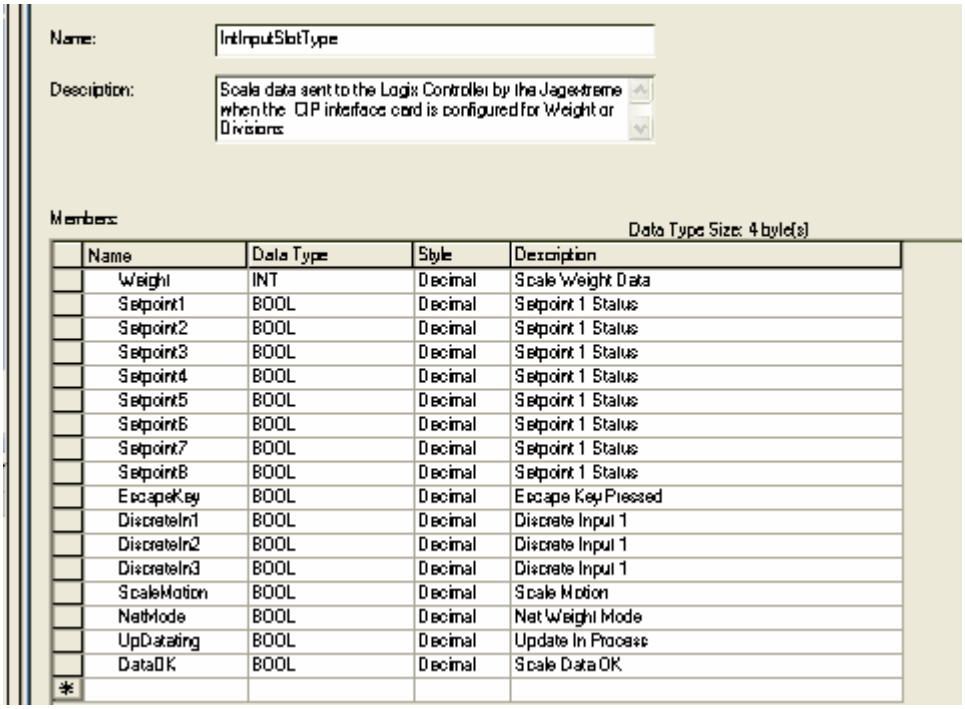
Creating a User Defined Data Type

Once a desired data type structure has been determined, the actual creation of the user-defined data type is relatively straightforward.

For Integer Input data from Assembly Instance 1, we will create a user-defined data type to represent a single slot of scale data, and later create a tag consisting of an array of 4 of those user-defined tag types.

To create a new data type, simply right click on **User Defined Data Types** in the left hand window of the Logix 5000 programming software and select **New Data Type** from the pop up menu that appears.

A "Data Type" dialog box will appear which allows you to completely define your custom data type. The following dialog box shows the structure of the **User Defined Data Type** "IntInputSlotData.

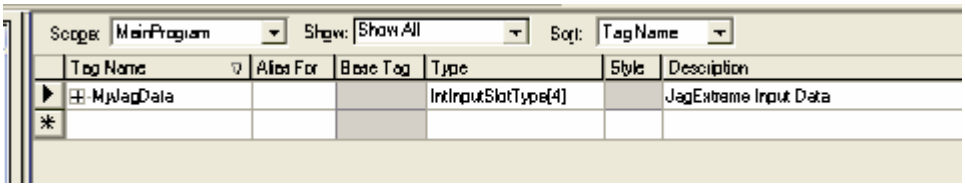


Name	Data Type	Style	Description
Weight	INT	Decimal	Scale Weight Data
Setpoint1	BOOL	Decimal	Setpoint 1 Status
Setpoint2	BOOL	Decimal	Setpoint 1 Status
Setpoint3	BOOL	Decimal	Setpoint 1 Status
Setpoint4	BOOL	Decimal	Setpoint 1 Status
Setpoint5	BOOL	Decimal	Setpoint 1 Status
Setpoint6	BOOL	Decimal	Setpoint 1 Status
Setpoint7	BOOL	Decimal	Setpoint 1 Status
Setpoint8	BOOL	Decimal	Setpoint 1 Status
EscapeKey	BOOL	Decimal	Escape Key Pressed
DiscreteIn1	BOOL	Decimal	Discrete Input 1
DiscreteIn2	BOOL	Decimal	Discrete Input 1
DiscreteIn3	BOOL	Decimal	Discrete Input 1
ScaleMotion	BOOL	Decimal	Scale Motion
NetMode	BOOL	Decimal	Net Weight Mode
UpDating	BOOL	Decimal	Update In Process
DataOK	BOOL	Decimal	Scale Data OK

Creating a Tag of the User Defined Data Type

In order to make use of the new data type created in the step above, a tag of that type must be created. The creation of the tag is done in the same way as any other Logix 5000 tag.

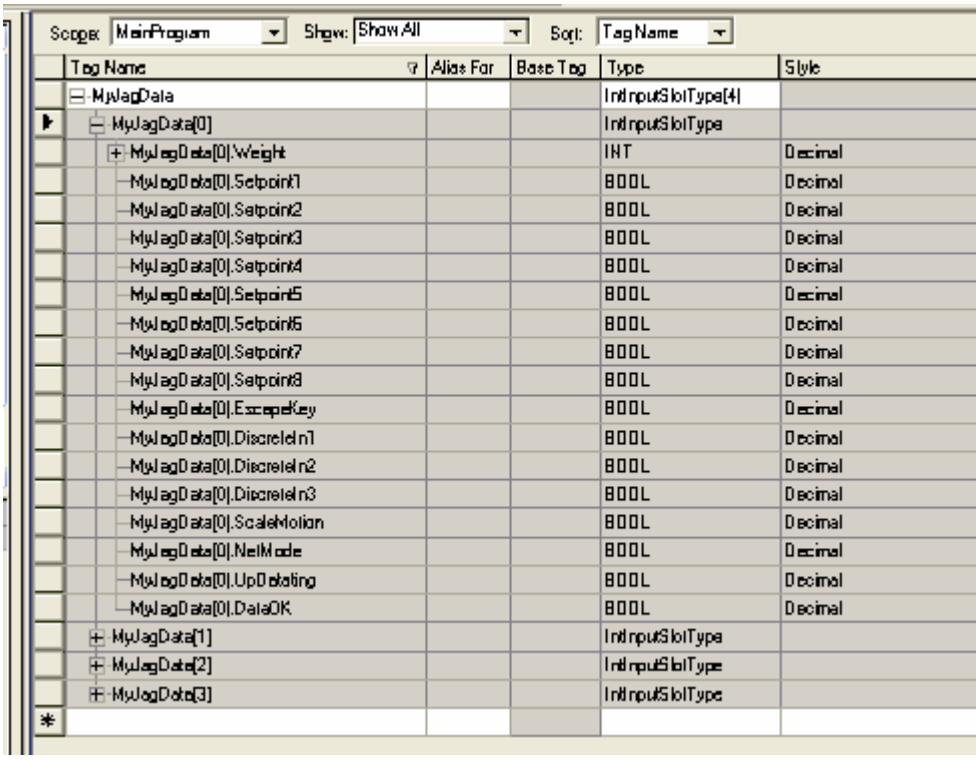
The figure below shows a tag named MyJagData consisting of an array of four IntInputSlotTypes.



Tag Name	Alias For	Base Tag	Type	Style	Description
MyJagData			IntInputSlotType[4]		JagExtreme Input Data

Creating an Array of User Defined Data Types

The following figure shows an expanded view of the tag named MyJagData. Note that the data for JagExtreme scale A is accessed as MyJagData[0]. Scale A weight is MyJagData[0]. Weight. Scale D weight is MyJagData[3].Weight.

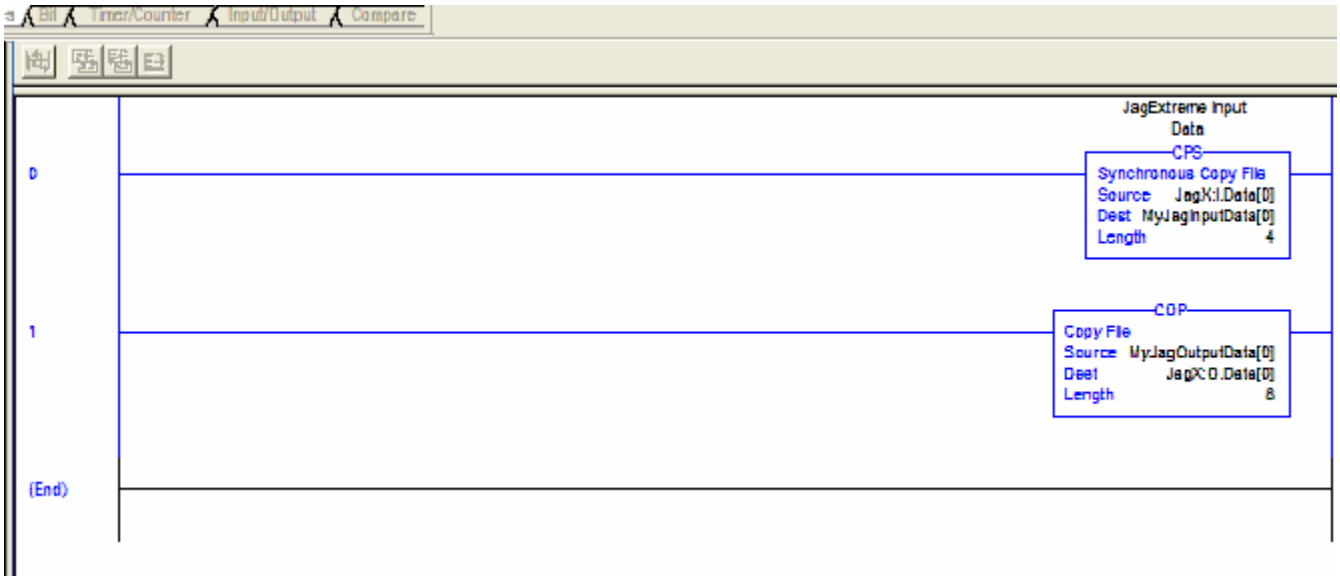


Jag Data Tag Example

Putting Data Into the Tag

Now that a tag of the user-defined type has been created, all that remains is to actually use it. When unscheduled messaging is being used, the tag can be referenced as the source or destination of the Msg command.

In the case of I/O messaging, however, the Logix 5000 controller will not allow such a direct approach. Instead, it will only receive data into or send data from the tags it automatically created when the I/O connection was configured. Consequently, the data must be copied between those tags and any user-defined tags within the ladder logic of the program in the following diagram.



In the diagram, the tag JagX:I.Data was automatically created by the program when the CIP connection was configured. The data from JagX:I.Data is copied into the four structures, MyJagInputData[0].

Once the data has been copied into the user-defined tag in such a fashion, its individual non-floating point fields are accessible to the remainder of the ladder program by their symbolic names. For instance, MyJagData[0].Weight will contain the current weight from the scale occupying Slot 1 of the JAGXTREME terminal's Assembly Instance 1, etc.

Note: See below for special instructions regarding the handling of floating point I/O data.

Handling Floating Point Data

The JAGXTREME terminal's floating point Assembly Instances represent a special case in terms of the way the I/O data must be handled. When a user data type is created which consists of mixed atomic data types, the Logix 5000 controller attempts to align certain individual data members on 32 bit address boundaries for the purposes of processor efficiency. It does this by padding those user-defined data types immediately ahead of the floating-point values.

Padding is a phenomenon by which invisible bytes are transparently inserted into the user-defined data type in order to attain the desired data alignment. An unfortunate consequence of this padding however, is that the resulting structure is functionally incorrect for the intended purpose in that it is oversized and the location any data starting with the floating point value is incorrect.

Avoiding Padding of User Defined Data Types

The following technique can be used to prevent Logix 5000 from surreptitiously padding user-defined data types containing misaligned floating-point values.

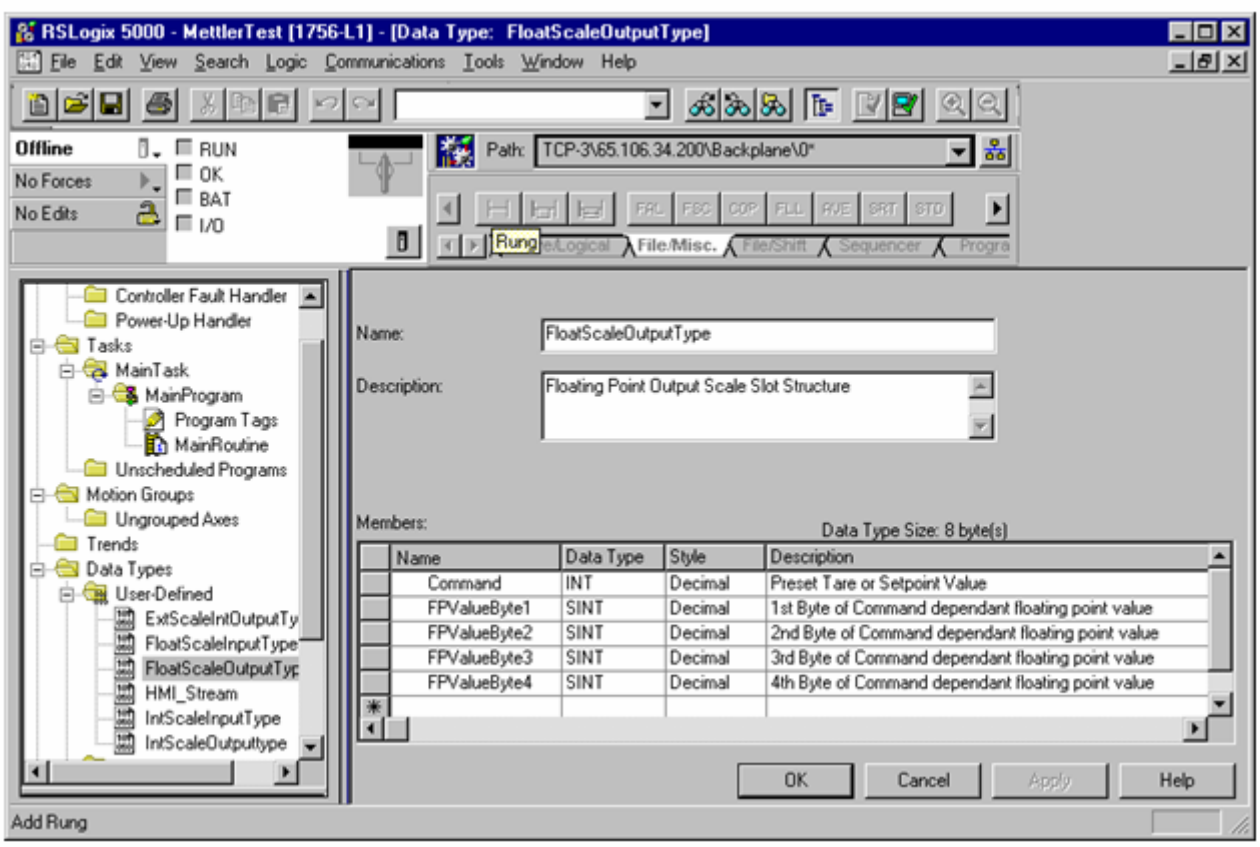
When defining a user-defined data type, which contains a floating-point value, do not define the actual floating-point value as a REAL. Instead, define four separate data members of type SINT in sequence.

In order to access the floating-point value, use the Logix 5000 Copy command to copy those four SINTS to another tag, which is of type REAL.

Example:

Instance 4 of the JAGXTREME terminal's Assembly Object (the Floating Point Output Instance) provides a good example of the application of the technique described above.

The figure that follows shows the proper data type definition for a single slot of the JAGXTREME terminal's Assembly Instance 4 data.

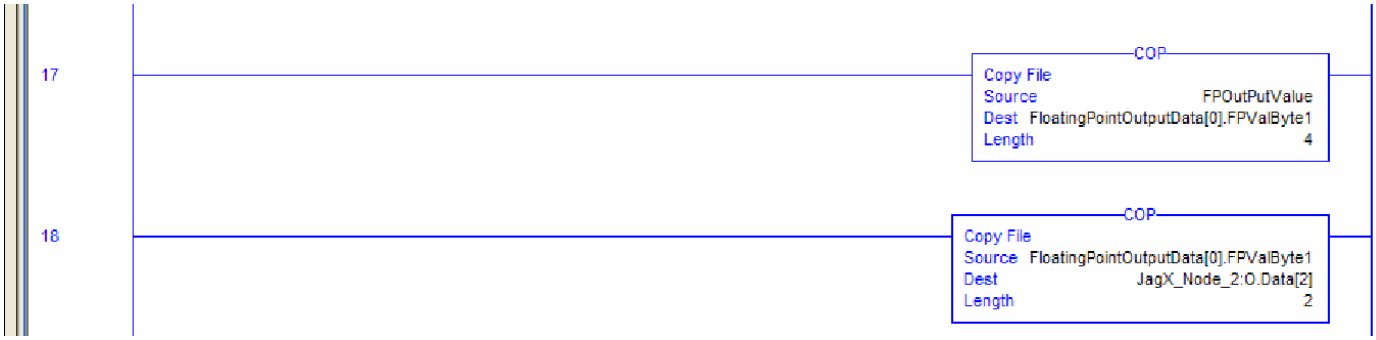


User Defined Data Type With a Floating Point Value

Note that the floating-point value in Figure 6-4 above has been defined as a series of 4 consecutive SINTs rather than a single REAL. Note also that the Logix 5000 controller still incorrectly miscalculates the length of the user-defined data type as 8 bytes rather than 6. This is due to padding at the end of the data type, but will not be an issue when the data is used.

The figure below shows how to load the 4 consecutive bytes defined above with a REAL before sending the data to the JAGXTREME terminal.

In this example, a tag "FloatingPointOutputData[4] has been defined using the Data Type of the User-Defined tag (FloatScaleOutputType) that was just created.



Preparing Floating Point Output Data for the JAGXTREME

Rung 17 in the figure above copies a REAL (real tag FPOutputValue) value into the four consecutive bytes assigned for the Floating Point Output value within the FloatingPointOutputData tag. Rung 18 copies one slot of the Floating point value to the JAGXTREME terminal's I/O output tag.

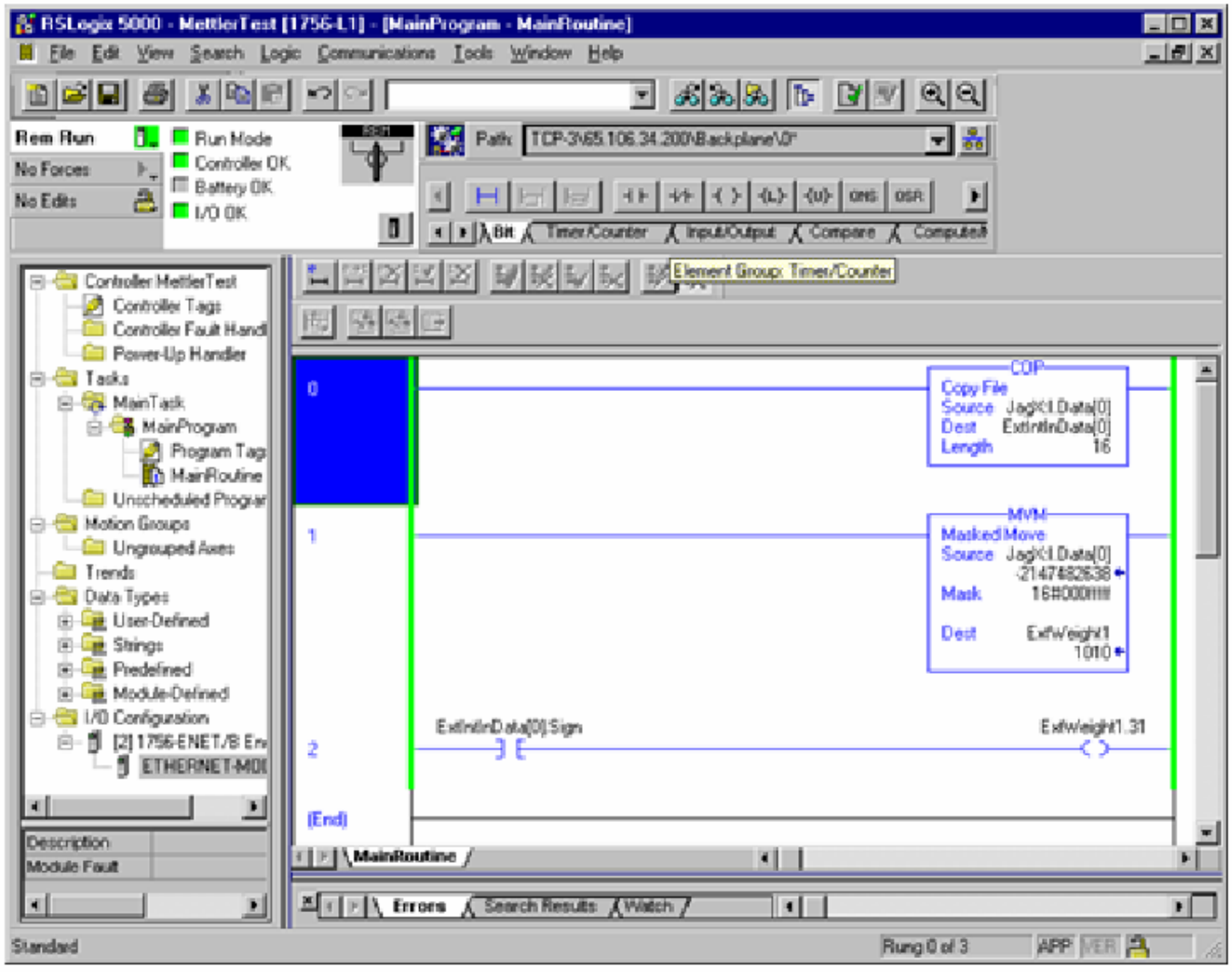
Note that the length of the data to be copied in Rung 18 above is four bytes which is the correct size of one Floating Point Output Slot.

Handling Extended Integer Input Data

The JAGXTREME terminal's Extended Integer Input Data represents a special case in terms of data access in that its actual weight data is comprised of a 21 bit signed integer, which crosses the boundary between 2 individual INTs. This differs from the Logix 5550 DINT data type, which is a standard signed 32-bit integer.

Converting 21 Bit INTs to 32 Bit INTs in Ladder Logic

The JAGXTREME terminal's 21 bit signed integer format can be converted to the Logix 5550's standard 32 bit signed integer format by the ladder logic shown in the figure that follows.



Converting a 21 Bit Integer to a 32 Bit Integer

The Masked Move instruction on rung 1 copies the least significant 20 bits of the JAGXTREME terminal's 21 bit Extended Weight data to a temporary tag named ExtWeight1.

The Logic on rung 2 sets or clears the most significant bit of ExtWeight1 (i.e. the sign bit) depending on the value of the Sign bit (bit 21) of the JAGXTREME terminal's Extended Weight data.

Note that the logic shown above will execute continuously on each scan of the ladder program.

Sending Unscheduled Messages to the JAGXTREME Terminal

The Logix 5550 controller's Msg instruction is an output instruction, which can be located on any rung of the user's ladder logic program. It can be used to communicate with JAGXTREME terminals, which are located directly on the same CIP network as the controller as well as those located on JAGXTREME cluster networks, which are accessible to the controller through the use of bridging.

Use of the Msg instruction involves the following steps:

- Creating the Msg instruction.
- Configuring the Msg instruction.
- Configuring the communications.
- Executing the Msg instruction.

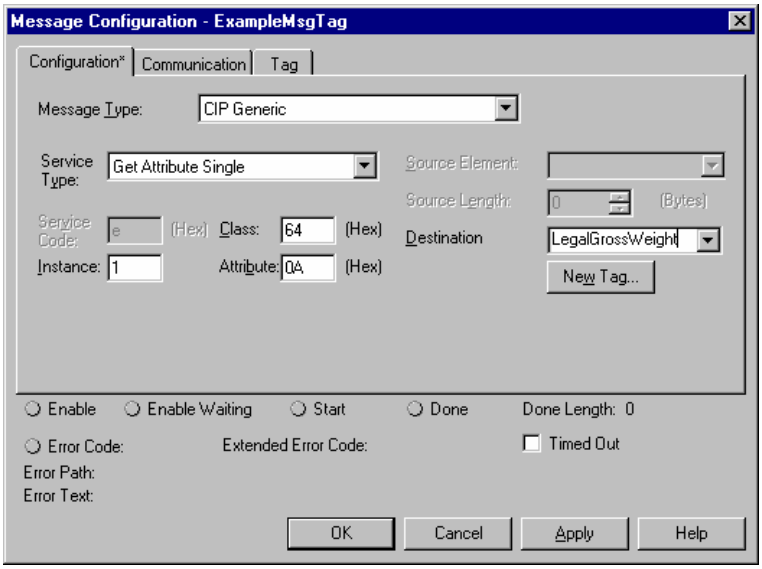
The example that follows takes you through the steps required to get the Legal Gross Weight parameter from a scale attached to a JAGXTREME terminal.

Creating the Msg Instruction

The Msg instruction is created and added to a run in the same manner as any other Logix 5550 output instruction. For help on creating the instruction and locating it on a rung, refer to the Logix 5000 programming software's online help system.

Configuring the Msg Instruction

The figure that follows below shows the Configuration tab of an Msg Configuration dialog box, which has been properly filled out to read the Legal Gross Weight from Scale #1, attached to a JAGXTREME terminal.



Msg Configuration Dialog Box

Message Type - The Message Type must always be CIP Generic when communicating with the JAGXTREME terminal.

Service Type - The Service type specified may be either Get Attribute Single to read data or Set Attribute Single to write data.

Class, Instance and Attribute - The Class, Instance and Attribute fields specify the Class, Instance and Attribute of the data to be read from or written to the JAGXTREME and can be obtained from Section 4 of this document. The selections shown identify the LegalGrossWeight data from Scale #1 of the JAGXTREME terminal's Scale Weight Object.

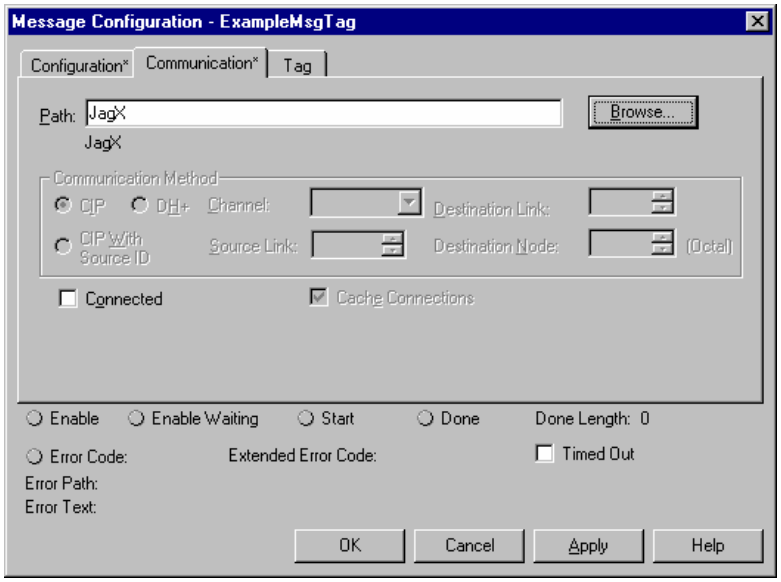
Source / Destination - The Source and Destination fields identify the source or destination tags for data to be written to or read from the JAGXTREME, depending on the Service

Type selected. Since the Service Code specified in the example above is Get Attribute Single, the Source and Source Length fields are grayed. Had the Set Attribute Single Service Type been specified, the Destination field would be grayed and the Source and Source Length field would be active instead. Note that when using the Set Attribute Single service, the Source Length field specifies the length of data to write to the JAGXTREME terminal in bytes.

Configuring the Communications

The Communications tab of the Msg Configuration dialog box allows the user to specify a communications path to a specific JAGXTREME as well as the type of communications to be utilized.

The figure below shows the Communications tab of an Msg Configuration dialog box.



Communications Configuration Tab

Connected / Cache Connections - The Connected and Cache Connection check boxes determine the type of messaging to be utilized.

Path - The Path field specifies the communications path that the Controller is to use to access the desired JAGXTREME terminal. How the path is specified differs depending on whether the desired JAGXTREME terminal is directly or indirectly connected to the controller.

Directly Connected JAGXTREME Terminals

Directly connected JAGXTREME terminals are those which reside directly on the same CIP network as the controller. In Figure 2-1 in Section 2 of this document, nodes 1 and 2 represent JAGXTREME terminals, which are directly connected to the controller over the same CIP network.

To specify a communications path to a directly connected JAGXTREME terminal, simply press the Browse button on the Communications tab of the Msg Configuration dialog box and select the desired JAGXTREME terminal from the I/O Configuration tree which will be presented. In the previous figure, the directly connected JAGXTREME terminal named JAGX has been selected as the communications path.

Indirectly Connected JAGXTREME Terminals

Indirectly connected JAGXTREME terminal's are those which reside on a non CIP JAGXTREME cluster network which is accessible to the controller via a direct connection to one of the JAGXTREME terminal's within that Cluster Network. In the following Network Diagram, Cluster ID 3 and 4 represent JAGXTREME Cluster Id's which are indirectly connected to the controller, through the JAGXTREME terminal at Cluster ID 1, which acts as a bridge between the two networks.

Specifying a communications path to an Indirectly connected JAGXTREME terminal is a bit more complicated than specifying one to a Directly connected JAGXTREME terminal as it involves the use of a comma delimited path string. (See Specifying:Communications Details (Communications tab) in the Logix 5000 programming software's help system for details.) The communications path to an indirectly connected JAGXTREME terminal takes the following form:

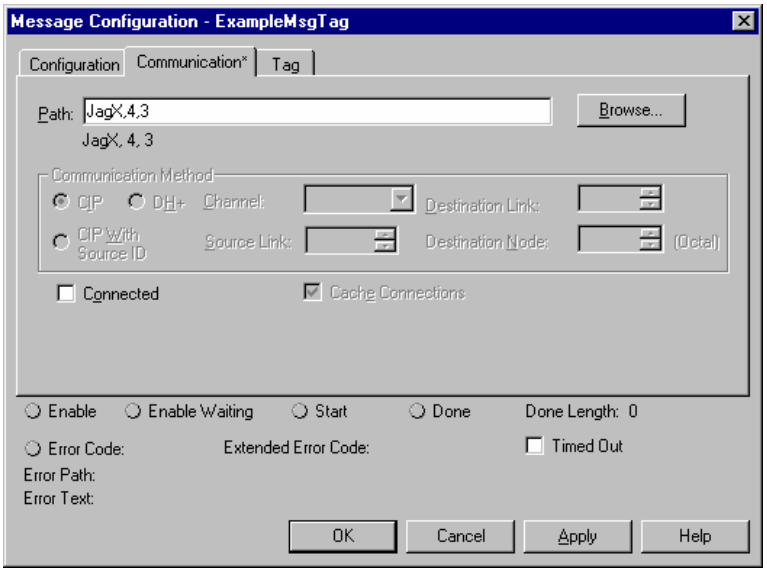
<BridgeJag>,<Port>,<RemoteJag>

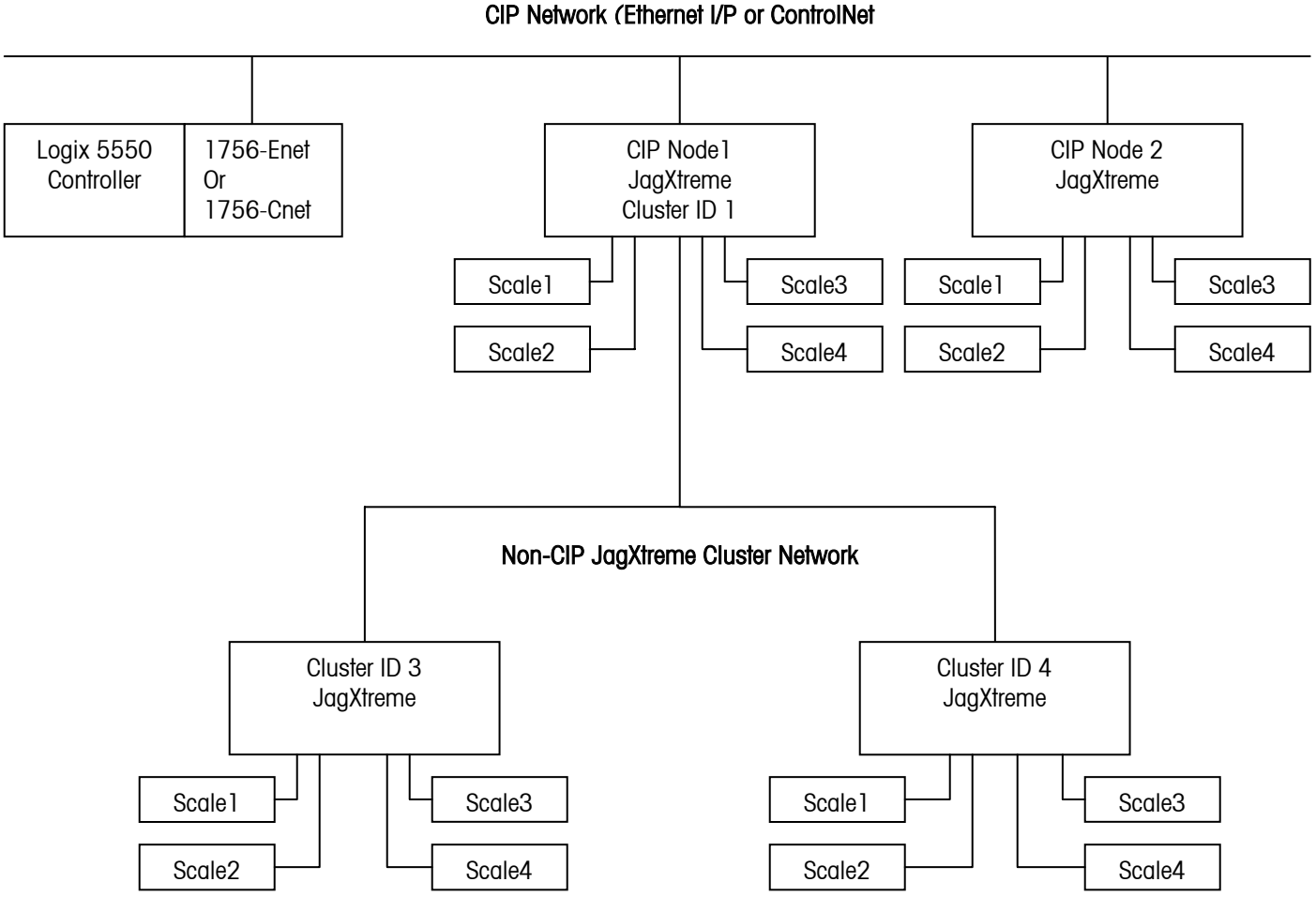
<BridgeJag> - The path to the JAGXTREME terminal which is acting as a bridge between the 2 networks. The path to this JAGXTREME is specified exactly as in the example for directly connected JAGXTREME terminals provided earlier.

<Port> - The port number by which the message is to leave the JAGXTREME terminal to get to the Cluster Network. This value will always be 4.

<RemoteJag> - The JagXtreme Cluster ID number of the Indirectly connected JAGXTREME terminal on the non CIP Cluster Network (1 - 6)

For example, the figure that follows shows the proper communications path for a JAGXTREME terminal, which is Cluster ID number 3 of a non CIP Cluster Network, connected to the CIP network by the JAGXTREME terminal referred to as JAGX.





Example of a Network Topology

NOTES

Appendix A

Floating Point Output Command Values

Command		Meaning (see notes at end of table)
Dec	Hex	
0	0	Report next field from input rotation at next A-to-D update *(6)*
1	01	Report next field from input rotation *(5,6)*
2	02	Report next field from input rotation *(5,6)*
3	03	Reset Input Rotation
10	0A	Report Gross Weight *(1,6)*
11	0B	Report Net Weight *(1,6)*
12	0C	Report Tare Weight *(1,6)*
13	0D	Report Fine Gross Weight
14	0E	Report Fine Net Weight *(1,6)*
15	0F	Report Fine Tare Weight *(1,6)*
16	10	Report Rate *(1,6)*
17	11	Report JagBASIC Custom Input 1 to Controller *(1,6)*
18	12	Report JagBASIC Custom Input 2 to Controller *(1,6)*
19	13	Report Low-Pass Filter Corner Frequency *(1)*
20	14	Report Notch Filter Frequency *(1)*
21	15	Report 1st Setpoint Coincidence Value *(1,3)*
22	16	Report 2nd Setpoint Coincidence Value *(1,3)*
23	17	Report 1st Setpoint Dribble Value *(1,3)*
24	18	Report 2nd Setpoint Dribble Value *(1,3)*
25	19	Report 1st Setpoint Tolerance Value *(1,3)*
27	1B	Report JagBASIC Custom Input 3 to Controller *(1)*
28	1C	Report JagBASIC Custom Input 4 to Controller *(1)*
29	1D	Report Error After Error Indication *(1)*
30	1E	Report Primary Units - Low Increment Size *(1)*
40	28	Add Gross Weight to Input Rotation
41	29	Add Net Weight to Input Rotation
42	2A	Add Tare Weight to Input Rotation
43	2B	Add Fine Gross Weight to Input Rotation
44	2C	Add Fine Net Weight to Input Rotation
45	2D	Add Fine Tare Weight to Input Rotation
46	2E	Add Rate to Input Rotation
47	2F	Add JagBASIC Custom Input 1 to Controller to Input Rotation
48	30	Add JagBASIC Custom Input 2 to Controller to Input Rotation
60	3C	Set Programmable Tare *(2)*
61	3D	Set Push-Button Tare
62	3E	Clear Command
63	3F	Print Command
64	40	Zero Command
65	41	Select Scale A
66	42	Select Scale B
67	43	Select Other Scale
68	44	Custom Print 1 Command
69	45	Custom Print 2 Command
70	46	Custom Print 3 Command

Command		Meaning (see notes at end of table)
Dec	Hex	
71	47	Custom Print 4 Command
72	48	Custom Print 5 Command
73	49	Set Low-Pass Filter Corner Frequency *(2)*
74	4A	Set Notch Filter Frequency *(2)*
75	4B	Reset Escape Key
78	4E	Disable Error Display
79	4F	Enable Error Display
80	50	Set Normal Display Mode
81	51	Display Literal 1
82	52	Display Literal 2
83	53	Display Literal 3
84	54	Display Literal 4
85	55	Display Literal 5
87	57	Display Literal in Shared Data Block of Message
88	58	Disable Numeric Display
89	59	Enable Numeric Display
90	5A	Set Discrete Output 1 = ON
91	5B	Set Discrete Output 2 = ON
92	5C	Set Discrete Output 3 = ON
93	5D	Set Discrete Output 4 = ON
100	64	Set Discrete Output 1 = OFF
101	65	Set Discrete Output 2 = OFF
102	66	Set Discrete Output 3 = OFF
103	67	Set Discrete Output 4 = OFF
110	6E	Set 1st Setpoint Coincidence Value *(2,3)*
111	6F	Set 1st Setpoint Dribble Value *(2,3)*
112	70	Set 1st Setpoint Tolerance Value *(2,3)*
114	72	1st Setpoint = Enabled *(3)*
115	73	1st Setpoint = Disabled *(3)*
116	74	1st Setpoint = Gross Weight *(3)*
117	75	1st Setpoint = Net Weight *(3)*
118	76	1st Setpoint = Rate *(3)*
119	77	1st Setpoint = Filling *(3)*
120	78	1st Setpoint = Discharging *(3)*
121	79	1st Setpoint = Latching Enabled *(3)*
122	7A	1st Setpoint = Latching Disabled *(3)*
123	7B	1st Setpoint = Reset Latch *(3)*
130	82	Set 2nd Setpoint Coincidence Value *(2,2)*
131	83	Set2nd Setpoint Dribble Value *(2,3)*
134	86	2nd Setpoint = Enabled *(3)*
135	87	2nd Setpoint = Disabled *(3)*
136	88	2nd Setpoint = Gross Weight *(3)*
137	89	2nd Setpoint = Net Weight *(3)*
138	8A	2nd Setpoint = Rate *(3)*
139	8B	2nd Setpoint = Filling *(3)*
140	8C	2nd Setpoint = Discharging *(3)*
141	8D	2nd Setpoint = Latching Enabled *(3)*
142	8E	2nd Setpoint = Latching Disabled *(3)*
143	8F	2nd Setpoint = Reset Latch *(3)*
150	96	Set JagBASIC Custom Output 1 From Controller *(4)*
151	97	Set JagBASIC Custom Output 2 From Controller *(4)*
152	98	Set JagBASIC Custom Output 3 From Controller *(4)*

Command		Meaning (see notes at end of table)
Dec	Hex	
153	99	Set JagBASIC Custom Output 4 From Controller *(4)*
160	A0	Apply Scale Setup
161	A1	Write Scale Calibration Parameters to EEPROM
162	A2	Disable Tare from JAGXTREME Control Panel, by Scale
163	A3	Enable Tare from JAGXTREME Control Panel, by Scale
166	A6	Disable Keyboard Tare from JAGXTREME Control Panel, by Scale
167	A7	Enable Keyboard Tare from JAGXTREME Control Panel, by Scale
168	A8	Select Scale C
169	A9	Select Scale D

Notes:

1. Command requires the JAGXTREME terminal to report a specific value in the controller input message. As long as one of these commands is in the Scale Command, the JAGXTREME terminal will respond with the requested data and not with data from the input rotation.
2. Command requires a floating-point value output from the controller to the JAGXTREME terminal. The JAGXTREME terminal reflects back the floating-point value in the floating-point register of input message to the controller. The JagBASIC application and the controller define the format of the data that has a length of four bytes.
3. Command that the controller uses to select the next field from the input rotation. The controller must alternate between these two commands to tell the JAGXTREME terminal when to switch to the next field of the input rotation.
4. Command that requests real-time fields from the JAGXTREME terminal. The JAGXTREME terminal updates the input register to the controller at the A-to-D update rate of scale.
5. Command that the controller uses to select the next field from the input rotation. The controller must alternate between these two commands to tell the JAGXTREME when to switch to the next field of the input rotation.
6. Command that requests real-time fields from the JAGXTREME terminal. The JAGXTREME terminal updates the input register to the controller at the A-to-D update rate of the scale.

Setpoints

Setpoint numbers are relative to each scale in the JAGXTREME terminal according to the following mapping.

Scales on JAGXTREME Terminal	Scale A Setpoints	Scale B Setpoints	Scale C Setpoints	Scale D Setpoints
1	1 & 2	N/A	N/A	N/A
2	1 & 2	3 & 4	N/A	N/A
3	1 & 2	3 & 4	5 & 6	N/A
4	1 & 2	3 & 4	5 & 6	7 & 8

Appendix B

Scheduled Integer Messaging

NOTES

METTLER TOLEDO
1900 Polaris Parkway

Columbus, Ohio 43240
Phone: (US and Canada) (800) 786-0038
(614) 438-4511
Phone: (International) (614) 438-4888
www.mt.com

P/N: **A16760300A**
(02/05).00

Printed in USA