A Sierra Monitor Company

Driver Manual

(Supplement to the FieldServer Instruction Manual)

FS-8704-12 GE-EGD

APPLICABILITY & EFFECTIVITY

Effective for all systems manufactured after April 2010

Driver Version:	1.02
Document Revision:	3

TABLE OF CONTENTS

1 GE-EGD (ETHERNET GLOBAL DATA) DESCRIPTION

The GE-EGD (Ethernet Global Data) driver allows the FieldServer to transfer data to and from devices over Ethernet using GE-EGD (Ethernet Global Data) protocol. The FieldServer can emulate either a Server or Client.

GE Fanuc Automation and GE Drive Systems developed an Ethernet Global Data, or EGD, exchange for PLC and computer data in 1998. EGD uses UDP or datagram messages for fast transfer of up to 1400 bytes of data from a producer to one or more consumers. UDP messages have much less overhead than the streaming TCP connection used for programming or CommReq's over SRTP Ethernet. Like Genius® broadcast input or directed control messages, UDP messages are not acknowledged. They can be sent at short intervals. Chances of one or more messages being dropped are small on a local area network.

As a Client the FieldServer acts as an EGD consumer. As a Server the FieldServer acts as an EGD producer.

The IC697CMM742 Ethernet module supports both GE SRTP and GE EGD.

2 DRIVER SCOPE OF SUPPLY

2.1 Supplied by FieldServer Technologies for this driver

FieldServer Technologies PART #	Description
FS-8915-10	UTP cable (7 foot) for Ethernet connection

2.2 Provided by the Supplier of 3rd Party Equipment

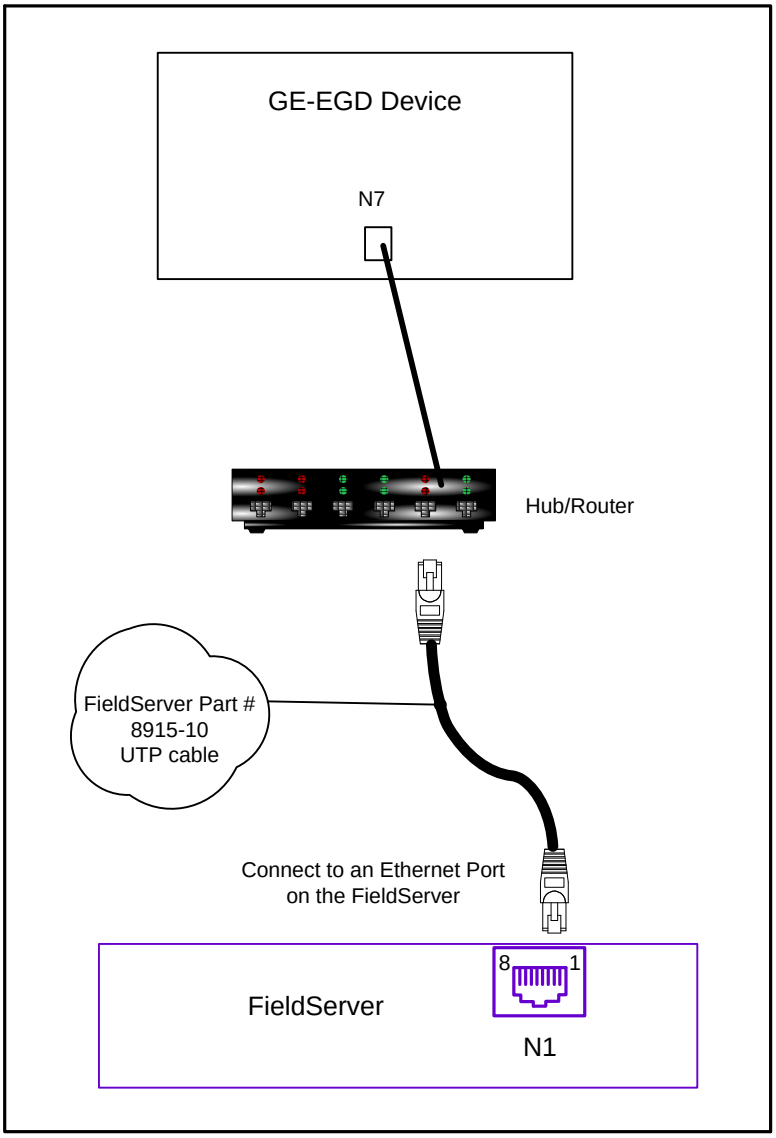
EGD capable GE communication/processor module.

The IC697CMM742 modules configured with Control and IC693CPU364 and IC200CPUE05 configured with VersaPro can send and receive EGD.

3 HARDWARE CONNECTIONS

The FieldServer is connected to the Site Ethernet as shown below.

Configure and connect the "GE TCP/IP Ethernet Interface Type 2" according to manufacturer's instructions.



4 DATA ARRAY PARAMETERS

Data Arrays are "protocol neutral" data buffers for storage of data to be passed between protocols. It is necessary to declare the data format of each of the Data Arrays to facilitate correct storage of the relevant data.

Section Title		
Data_Arrays		
Column Title	Function	Legal Values
Data_Array_Name	Provide name for Data Array	Up to 15 alphanumeric characters
Data_Array_Format	Provide data format. Each Data Array can only take on one format.	Float, Bit, UInt16, SInt16
Data_Array_Length	Number of Data Objects. Must be larger than the data storage area required by the Map Descriptors for the data being placed in this array.	1-10, 000

Example

```
//  Data Arrays
Data_Arrays
Data_Array_Name   , Data_Array_Format   , Data_Array_Length
DA_AI_01          , UInt16              , 200
DA_AO_01          , UInt16              , 200
DA_DI_01          , Bit                 , 200
DA_DO_01          , Bit                 , 200
```

5 CONFIGURING THE FIELDSERVER AS A GE-EGD CLIENT

Historically, one uses the client-server model to describe the operation of most protocols. Recently producer-consumer model protocols have started to become more numerous. The GE-EGD (Ethernet Global Data) is a producer-consumer model protocol. In equating the two models i*t is important to regard the consumer as a passive (FieldServer) client.* Other clients typically are active and poll for new data. *The consumer is a passive client in that waits to digest new data generated by a producer.*

For a detailed discussion on FieldServer configuration, please refer to the instruction manual for the FieldServer. The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer (See ".csv" files provided with the FieldServer).

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with a GE-EGD Producer.

The configuration file tells the FieldServer about its interfaces, and the routing of data required. In order to enable the FieldServer for GE-EGD communications, the driver independent FieldServer buffers need to be declared in the "Data Arrays" section, the destination device addresses need to be declared in the "Client Side Nodes" section, and the data required from the servers needs to be mapped in the "Client Side Map Descriptors" section. Details on how to do this can be found below.

Note that in the tables, * indicates an optional parameter, with the bold legal value being the default.

5.1 Client Side Connection Parameters

Section Title		
Adapter		
Column Title	**Function**	**Legal Values**
Adapter	Adapter Name	N1,N2[1]
Protocol	Specify protocol used	ge_egd

Example

```
// Client Side Connections

Adapters
Adapter                    , Protocol
N1                         , ge_egd
```

[1] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

5.2 Client Side Node Parameters

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
IP_Address	The IP address in dot format of the EGD-Device.	Nnn.nnn.nnn.nnn Where nnn is in the range 0-255.
Protocol	Specify protocol used	ge_egd
Adapter	Specify which adapter connects to the network the EGD-device is connected to.	N1, N2[2]

Example

```
// Consumer (Passive Client) Side Nodes

Nodes
Node_Name , IP_Address , Adapter , Protocol
Node_A , 192.168.1.102 , N1 , ge_egd
```

5.3 Client Side Map Descriptor Parameters

5.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters..
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4
Data_Array_Location	Starting location in Data Array	0 to (Data_Array_Length-1) as specified in Section 4
Function	Function of Client Map Descriptor	Passive

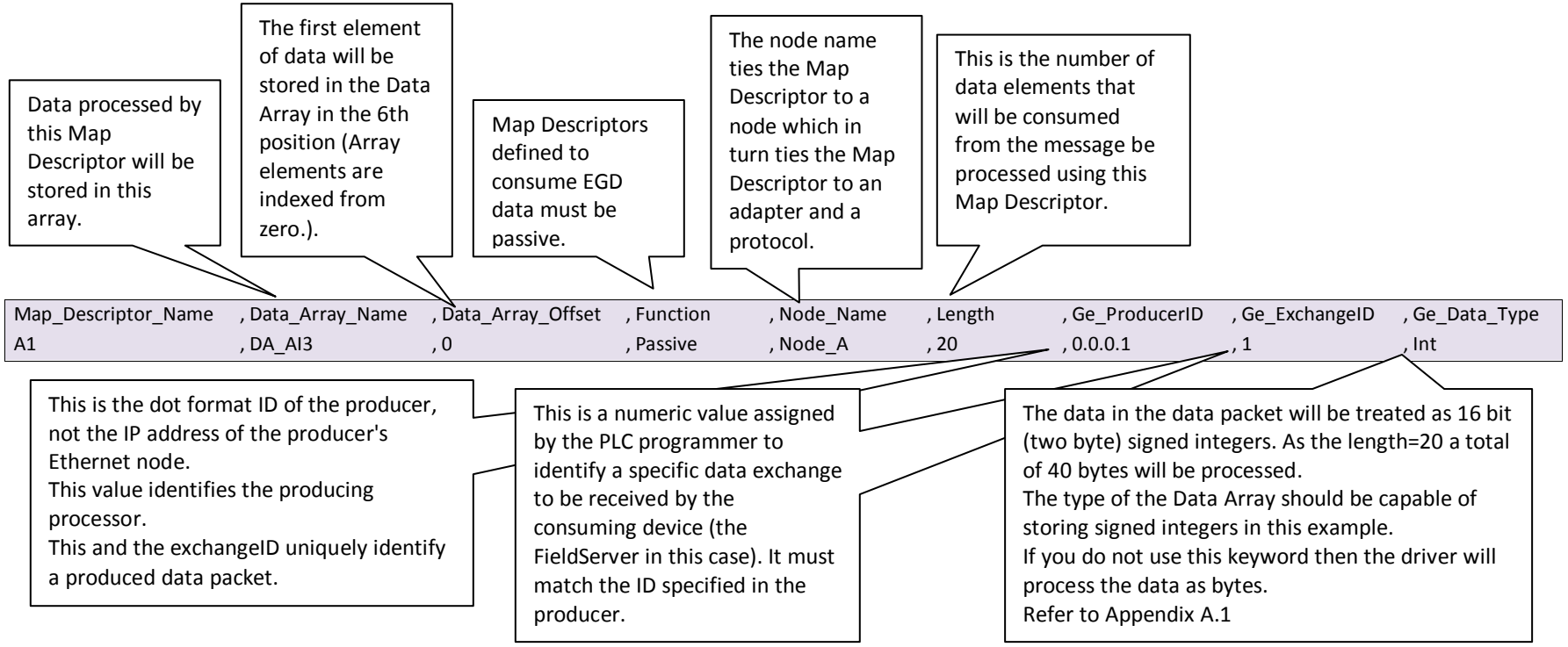
[2] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

5.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the node names specified in Section 5.2
Length	Number of points being consumed. For Bit values this represents the number of bytes (i.e. number of points divided by 8)	1 - 1000
Ge_ProducerId	This identifies the GE device producing the EGD data. Although in decimal dot format, it is not an IP address and does not necessarily correspond to the IP address of the GE-Ethernet port producing the message. It corresponds to the producer ID configured for the CPU producing the data. The default value is typically the same as the IP address of the producer but the value can be changed and it is possible for one device to have multiple Ethernet interfaces and hence multiple IP addresses. Any change to the producerID must be matched by a similar change in the consumer's configuration.	Nnn.nnn.nnn.nnn Where nnn are in the range 0-255.
Ge_ExchangeId	Used with the ProducerID, to uniquely identify a packet of EGD data. The driver uses these two parameters to match a produced data packet with one or more passive Map Descriptors.	Integer values >= 1
Ge_Data_Type*	Each produced data packet contains raw packed data. Nothing in the message identifies the structure or type of the incoming data. The Driver therefore cannot differentiate between byte, integer, real ... numbers and requires the specification of this keyword to unpack the data buffer.	**Byte**, Bit, Word, Dword, Int , Long, Float (4 byte IEEE real number) or Double (8 byte IEEE real number).
Ge_Offset*	If the producer has been configured to produce data of multiple types in one data packet then multiple Map Descriptors are required to decode them. The Ge_Offset is used to point to the first byte in the data packet to be processed by the Map Descriptor. Typically the Map Descriptor for the 2nd, 3rd ... Map Descriptors associated with one data packet will be non-zero.	**0,** Any positive integer

5.3.3 Map Descriptor Example 1: - Simple Consumer Map Descriptor

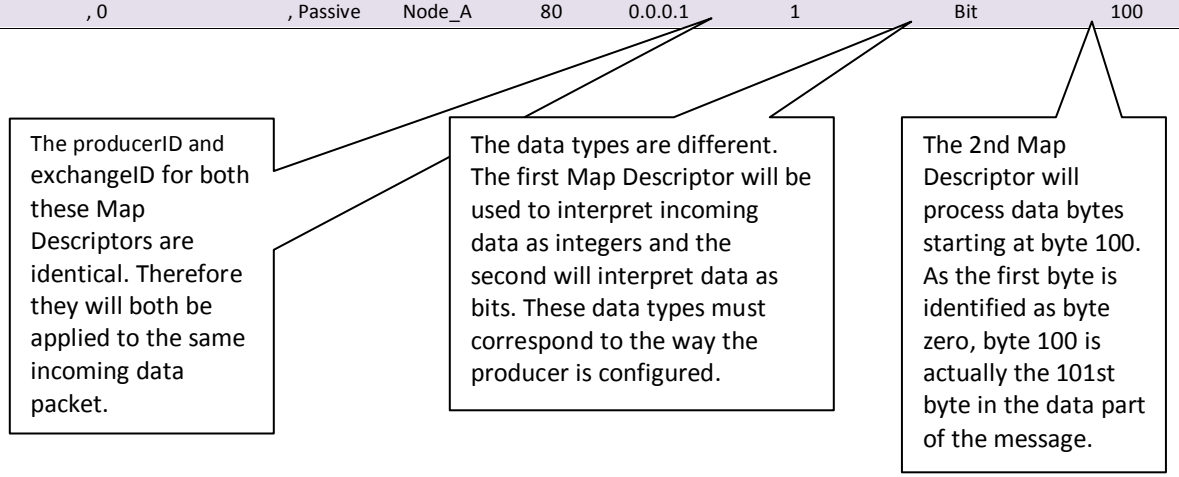
In this example the basics required for each consumer Map Descriptor are explained.



5.3.4 Map Descriptor Example 2: - Multiple Consumer Map Descriptor

In this example we assume that one produced data packet (produced by 0.0.0.1 and identified as exchange 1) contains different types of data elements making up the single exchange. This is configured when configuring EGD for the producer. The arrangement of data must correspond exactly with the configuration of the Map Descriptors used to consume the data. The following two Map Descriptors imply that the exchange contains at least 180 bytes of data and that the first 40 bytes contain 20 word values and that bytes 100 to 179 contain bit values. We cannot deduce what bytes 40-99 contain.

Map_Descriptor_Name	, Data_Array_Name	, Data_Array_Offset	, Function	Node_Name	Length	Ge_ProducerID	Ge_ExchangeID	Ge_Data_Type	ge_offset
A1	, DA_AI3	, 0	, Passive	Node_A	20	0.0.0.1	1	Int	0
A2	, DA_DI1	, 0	, Passive	Node_A	80	0.0.0.1	1	Bit	100



6 CONFIGURING THE FIELDSERVER AS A GE-EGD SERVER

6.1 Server Side Connection Parameters

Section Title
Connections

Column Title	Function	Legal Values
Adapter	Adapter Name	N1,N2
Protocol	Specify protocol used	ge_egd

Example

```
Adapters
Adapter     , Protocol
N1          , ge_egd
```

6.2 Server Side Node Parameters

Section Title
Nodes

Column Title	Function	Legal Values
Node_Name	Provide name for Node	Up to 32 alphanumeric characters
IP_Address	The IP address in dot format of the EGD-Device.	Nnn.nnn.nnn.nnn Where nnn is in the range 0-255.
Protocol	Specify protocol used	ge_egd
Adapter	Specify which adapter connects to the network the EGD-device is connected to.	N1, N2[3]

Example

```
//  Producer(Active Server) Side Nodes

Nodes
Node_Name   , IP_Address,      , Adapter   , Protocol
node_A      , 192.168.1.102,   , N1        , ge_egd
```

[3] Not all ports shown are necessarily supported by the hardware. Consult the appropriate Instruction manual for details of the ports available on specific hardware.

6.3 Server Side Map Descriptors

Only one Map Descriptor may be configured for each ExchangeID. Each produced exchange is thus limited to one data type and to data from one Data Array. This is different from the configuration of consumer Map Descriptors.

6.3.1 FieldServer Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Map_Descriptor_Name	Name of this Map Descriptor	Up to 32 alphanumeric characters..
Data_Array_Name	Name of Data Array where data is to be stored in the FieldServer	One of the Data Array names from Section 4
Data_Array_Location	Starting location in Data Array	0 to (Data_Array_Length-1) as specified in Section 4
Function	Function of Client Map Descriptor	Wrbc

6.3.2 Driver Specific Map Descriptor Parameters

Column Title	Function	Legal Values
Node_Name	Name of Node to fetch data from	One of the Node names specified in Section 6.2.
Length	Length of Map Descriptor	1 - 1000
Ge_ProducerId	This identifies the GE device producing the EGD data. Although in decimal dot format, it is not an IP address and does not necessarily correspond to the IP address of the GE-Ethernet port producing the message. It corresponds to the producer ID configured for the CPU producing the data. The default value is typically the same as the IP address of the producer but the value can be changed and it is possible for one device to have multiple Ethernet interfaces and hence multiple IP addresses. Any change to the producerID must be matched by a similar change in the consumer's configuration.	Nnn.nnn.nnn.nnn Where nnn are in the range 0-255.
Ge_exchangeId	This and the producerID uniquely identify a packet of EGD data. Thus, the consumer uses these two parameters to update. Any change to the exchangeID must be matched by a similar change in the consumer's configuration.	Integer values >= 1
Ge_data_type	Each produced data packet contains raw packed data. This keyword is used to tell the driver how to pack the data into the message. Thus data can be read from a BIT array in the FieldServer and sent as words for storage in %R (register memory) in the GE-PLC. Any change to the data type must be matched by a similar change in the consumer's configuration.	Refer to Appendix C.1.

6.3.3 Timing Parameters

Column Title	Function	Legal Values
Scan_Interval	Rate at which data is produced. This is the equivalent of the producer interval.	>0.1s

6.3.4 Map Descriptor Example.

Map_Descriptor_Name	, Data_Array_Name	, Data_Array_Offset	, Function	, Node_Name	, Length	, Scan_Interval	,Ge_producerID	, ge_exchangeID	, Ge_data_type
A1	, DA_AI3	, 0	, Wrbc	, Node_A	, 100	, 5.0s	, 0.0.0.1	, 1	, %R

Only a **Wrbc** can be used to produce data. The other write functions are not periodic.

Consider this as the producer interval.

The consumer must be configured to have the same producerID and exchangeID. These two fields are the only way it has of differentiating one set of produced data from another.

Defines how data is packed into the data part of the message.
In this example data will be packed words (unsigned 16 bit integers) suitable for storage in register memory in the GE PLC's.
Appendix C.1 contains a full list.

Appendix A. Vendor Information

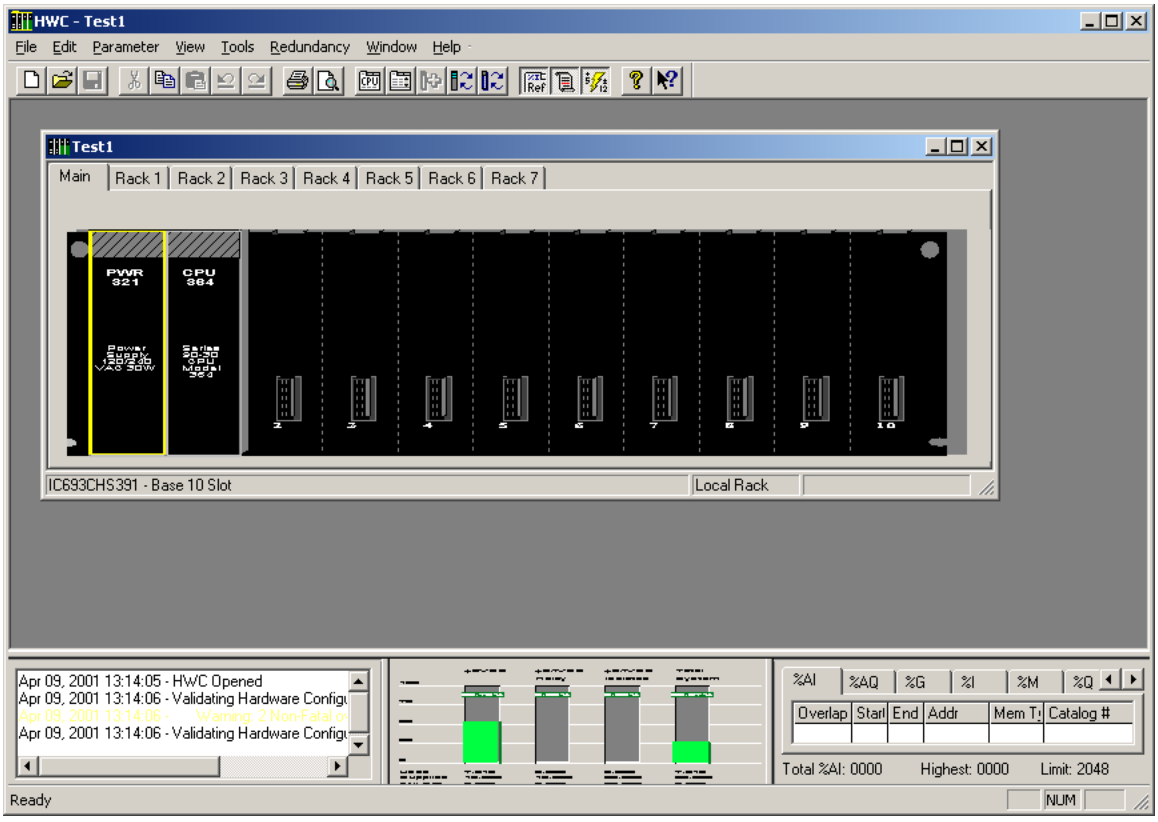
Appendix A.1. Enable the FieldServer to read data from a 90-xx PLC.

Appendix A.1.1. Use Versapro to configure/look at the EGD configuration.

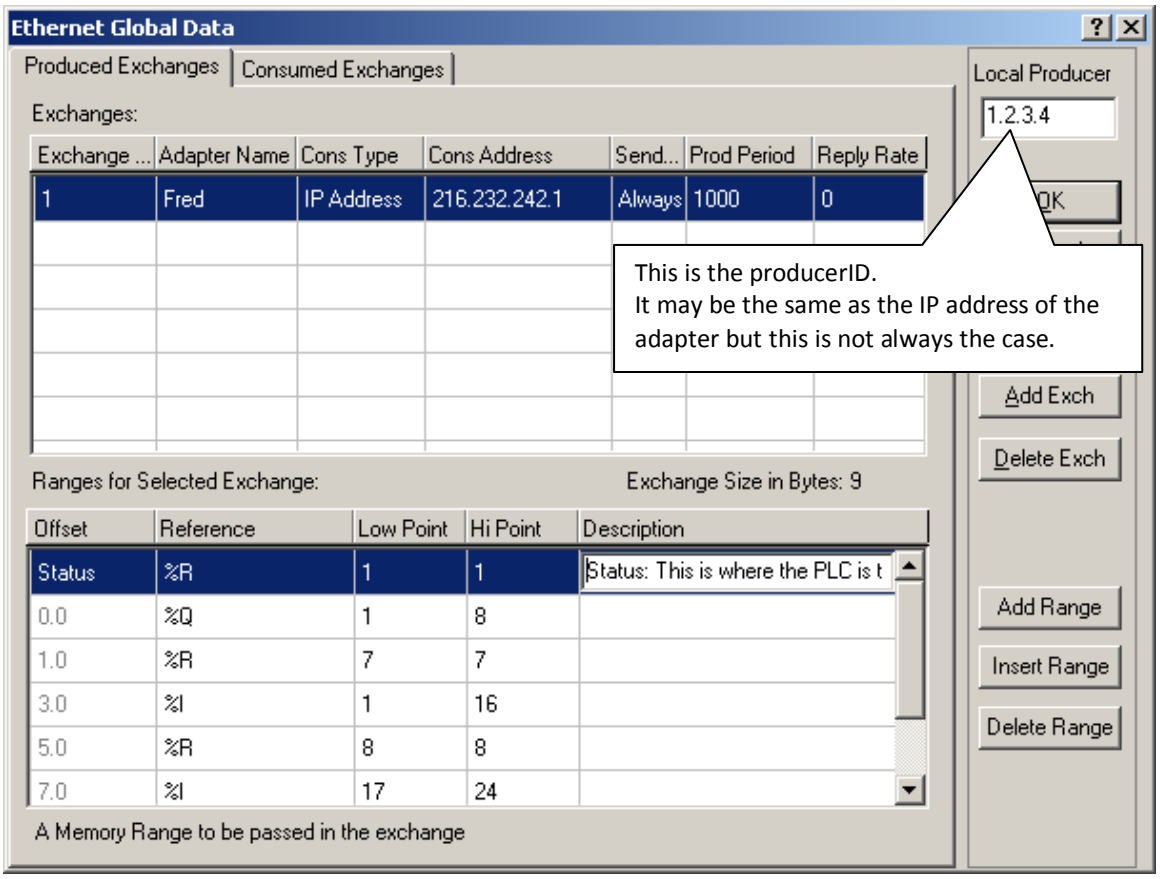
Produced data must be produced for a specific consumer (Specific IP address). Thus a new exchange must be created in the PLC that will produce data for the FieldServer.

Since the EGD data packet is not structured, the FieldServer cannot decode the data ranges without the Map Descriptors. It is therefore important that the data ranges in the produced exchange correspond to the Map Descriptors in the CSV file.

- Go online.
- View Menu, Hardware Configuration. (Launches HWC program).
- HWC. Edit. Rack Operations. EGD Configuration.

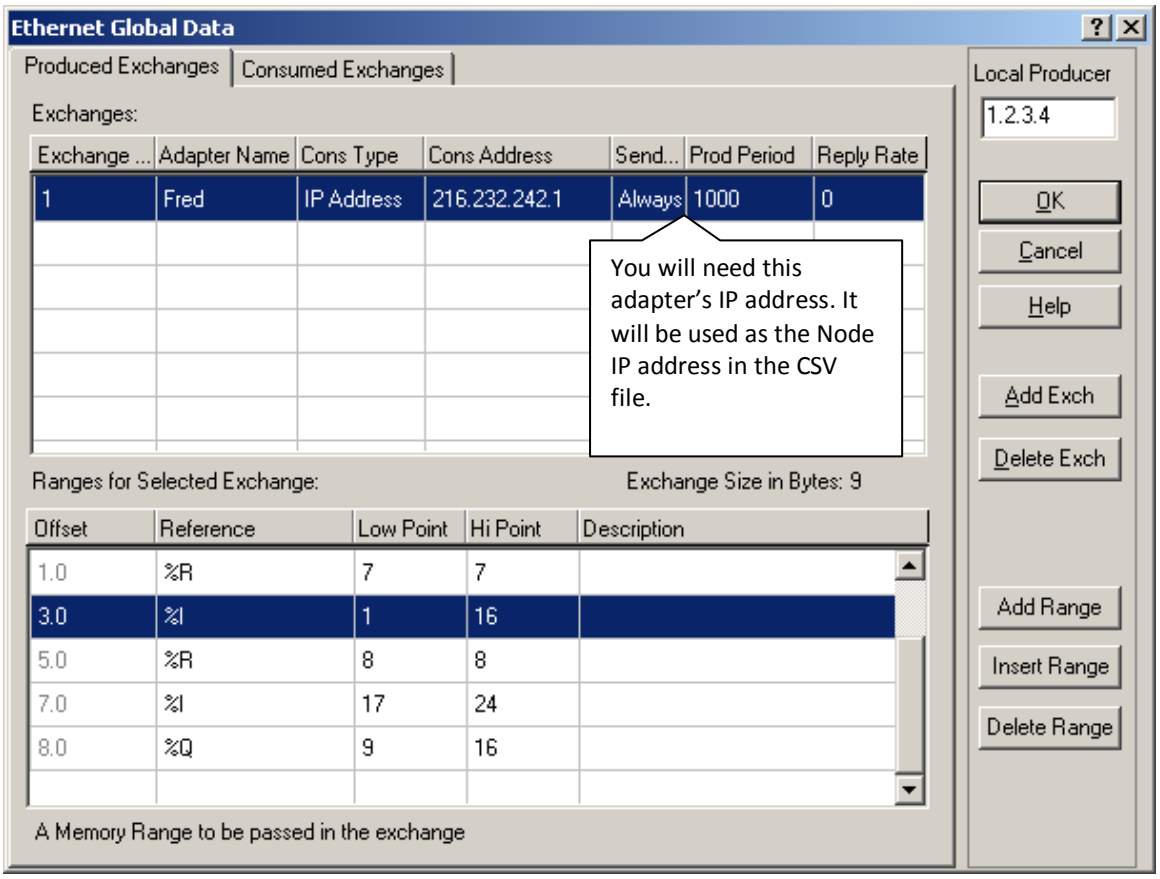


- Add an exchange. Set the CONS ADDRESS equal to the IP address of the FieldServer.



- Note the Local Producer address. Typically it will be the same as the IP of the closest GE Ethernet port. You can override this.
- Add Ranges. Record the offset and reference for each data range in the exchange.
- Save your work.
- Close HWC.
- Stop the processor.
- Store the Hardware settings to the PLC
- Put the processor back in run mode (must be running to produce.)

A second screen image shows that this exchange actually has an additional range at offset 8.



Appendix A.1.2. Create a CSV file that will consume the produced data.

An example is shown on the following page.

```
Adapters
Adapter     , Protocol
N1          , ge_egd
```

```
Nodes
Node_Name   , IP_Address      , Adapter   , Protocol
PLC90-30    , 216.232.242.3   , N1        , ge_egd
```

```
Nodes
Node_name   , Protocol
null_node   , ge_egd
```

This is the IP address of the producing port. You can obtain this by using the Versapro HWC program and double clicking on the Module with the adapter shown in the EGD configuration. (Fred, in this example) Now look for the Ethernet port address.

```
Data_Arrays
Data_Array_Name   , Data_Format   , Data_Array_Length
DA_AO_01          , Float         , 200
DA_AI_00          , BYTE          , 100
DA_AI_01          , BIT           , 100
DA_AI_02          , UINT16        , 100
DA_AI_03          , UINT32        , 100
DA_AI_04          , SINT16        , 100
DA_AI_05          , SINT32        , 100
DA_AI_06          , FLOAT         , 100
DA_AI_07          , FLOAT         , 100
EGD_DIAG          , UINT32        , 100
EGD_STATS         , UINT32        , 100
```

```
Map_Descriptors
Map_Descriptor_Name,   Data_Array_Name,   Node_Name
egd-ii              ,  EGD_DIAG         ,  Null_Node
egd-stats           ,  EGD_STATS        ,  Null_Node
```

```
Map_Descriptors
Map_Descriptor_Name , Data_Array_Name , Data_Array_Offset , Function , Node_Name , Length , Ge_producerId , Ge_exchangeId , Ge_data_type , Ge_offset
Q1                  , DATA_Q          , 0                 , Passive  , PLC90-30  , 1      , 1.2.3.4       , 1             , %q           , 0
R1                  , DATA_R          , 0                 , Passive  , PLC90-30  , 1      , 1.2.3.4       , 1             , %r           , 1
I1                  , DATA_R          , 0                 , Passive  , PLC90-30  , 2      , 1.2.3.4       , 1             , %u           , 3
R2                  , DATA_R          , 1                 , Passive  , PLC90-30  , 1      , 1.2.3.4       , 1             , %r           , 5
I2                  , DATA_R          , 2                 , Passive  , PLC90-30  , 1      , 1.2.3.4       , 1             , %i           , 7
Q2                  , DATA_R          , 1                 , Passive  , PLC90-30  , 1      , 1.2.3.4       , 1             , %q           , 8
```

Must correspond to the 'Local Producer' in the EGD configuration. Not necessarily the IP address of the producer port.

Refer to Appendix C.1 to see how many items are being transmitted. Note that the %Q, %I references are actually byte references and not bit references as they are always produced in multiples of 8 and are always byte aligned.

These data types must correspond to the references in the EGD range configuration.

These offsets must correspond to the offsets in the EGD configuration.

Appendix B. Troubleshooting

Appendix B.1. ProducerID with FieldServer device as Producer

During testing it has been observed that a 90-30 PLC required that the *ge_ProducerID* parameter was set to the same value as the IP Address of the FieldServer.

Appendix B.2. Produced Time Stamp

The GE-EGD (Ethernet Global Data) driver always sets the timestamp of produced data to the time of the Field Server Device. The nanoseconds portion of the time stamp is always set to zero.

Appendix B.3. Status Values

The status of the EGD Exchange may be monitored in the GE PLC. The status value is well documented in GFK-1541 Chapter 4.4. During testing, using the Field Server device as a producer and the GE Device as a consumer the following status values were observed.

0 -> The exchange had never been consumed
1 -> Normal
4 -> Length of produced and consumed exchange not equal - Different messages with the same exchange ID.
6. -> Timeout.

Appendix B.4. Error Messages

Multiple protocol drivers may exist on a FieldServer. Each driver may produce its own error messages and the FieldServer itself may produce error messages.

Message	Action
EGD:#1 Error. Can't init UDP.	This is a fatal error. The FieldServer needs to be re-initialized or you need technical support from FieldServer Technologies.
EGD:#2 Error. Can't get a socket.	
EGD:#3 Error. Protocol does not support active polling. Change function for mapDesc=<%s>	The rdbc/rdb/rdbx functions are not supported by this protocol. The device you wish to poll must be configured to 'produce' its data and this driver will 'consume' the data using passive Map Descriptors.[4]
EGD:#4 Error. Producer ID required for mapDesc=<%s>	Each Map Descriptor requires a producerID.[4]
EGD:#5 Error. Exchange ID required for mapDesc=<%s>	Each Map Descriptor requires an exchangeID.[4]
EGD:#6 FYI. No data type specified. Defaulted to <Byte>	This is a warning only. You can eliminate the warning by editing the CSV file.[4]
EGD:#7 FYI. Data type not recognized. Defaulted to <Byte> for mapDesc=<%s>	

[4] Edit the CSV file, download to the FieldServer and restart the FieldServer for the changes to take effect.

Message	Action
EGD:#8 Error. Don't know GE Data Type(%d) for mapDesc=<%s>	An illegal data type has been used.[4]
EGD:#9 Error. Incoming data from ip=<%s> producerID=<%s> exchangeID=(%d) is being abandoned.	An EGD producer has sent a data packet to the FieldServer but the driver cannot find a passive Map Descriptor to use to process and store the incoming data. It's possible that the producer has been incorrectly configured and that the packet was not intended for the FieldServer. Alternatively, make a new Map Descriptor which will handle this data.
EGD:#10 Error. Don't know GE Data Type (%d) for mapDesc=<%s>	An illegal data type has been used.[4]
EGD:#11 FYI. You could have used a mapDesc called <egd-ii> to expose diagnostic info.	This message requires no action. Refer to Appendix B.5 of this manual to expose driver internal diagnostic data.
EGD:#12 Invalid IP. Too many characters.	IP address is more than 15 characters in length.[4]
EGD:#13 Invalid IP <%s>	Insufficient points in the IP address. [4]
EGD:#14 Error. The mapDesc called <egd-stats> is too short	Increase the data length parameter for this Map Descriptor Make sure the Data Array is long enough too.
EGD:#15 FYI. You could have used a mapDesc called <egd-stats> to expose diagnostic info	Refer to Appendix B.6 for more info.

Appendix B.5. EGD-ii (EGD Internal Indications)

This driver can expose data from the most recently consumed message and some additional diagnostic information. A special Map Descriptor is required. The driver recognizes the Map Descriptor by its name which must be "***EGD-ii***" which stands for EGD Internal Indications.

The following example shows how this special Map Descriptor can be configured.

```
Nodes
Node_name,                  Protocol
null_node,                  ge_egd

Data_Arrays
Data_Array_Name,            Data_Format,            Data_Array_Length
EGD_DIAG,                   UINT32,                 100

Map_Descriptors
Map_Descriptor_Name,        Data_Array_Name,        Node_name
egd-ii,                     EGD_DIAG,               null_node
```

This Map Descriptor instructs the driver to use the Data Array EGD_DIAG to store driver specific data. Only one of these Map Descriptors may be specified per FieldServer.

The driver stores the following data.

Array Element	Contents
0-31	The first 32 bytes of the most recently received UDP packet received on port 0x4746 (The GE EGD port).
32	PDUTypeVersion
33	RequestID
34	ProducerID[5]
35	ExchangeID
36	TimeStampSec
37	TimeStampNanoSec
38	Status[6]
39	ConfigSignature
40	Reserved
41	Source IP Address[1]

Appendix B.6. Driver Stats

EGD producers produce data messages for Server devices to consume. The type and frequency of the messages depends on the producer configuration. The driver counts all incoming messages of interest as the PLC_READ_MSG_RECD statistic. Other legal messages which do not contain data of interest are discarded and are counted as the MSG_IGNORED statistic. The PLC_READ_MSG_RECD statistic is incremented once by each Map Descriptor which extracts data from an incoming message. Thus, one incoming message and three associated Map Descriptors would cause the statistic to increase by three (when viewed from the connection's point of view.)

This driver can expose some driver statistics by writing data to a Data Array. A special Map Descriptor is required. The driver recognizes the Map Descriptor by its name which must be "***EGD-stats***".

The following example shows how this special Map Descriptor can be configured.

```
Nodes
Node_Name           , Protocol
Null_Node           , ge_egd

Data_Arrays
Data_Array_Name     , Data_Format       , Data_Array_Length
EGD_STATS           , UINT32            , 100

Map_Descriptors
Map_Descriptor_Name , Data_Array_Name   , Node_Name
egd-stats           , EGD_STATS         , Null_Node
```

This Map Descriptor instructs the driver to use the Data Array EGD_STATS (in this example) to store driver specific statistics. Only one of these Map Descriptors may be specified per FieldServer.

The driver stores the following data.

[5] As a UINT32. Not in dot format
[6] Read section 4.4 of GE-Fanuc document GFK-1541 for more information.

Array Element	Contents
0	Messages Produced
1	Bytes Produces
2	Messages Received
3	Bytes Received
4	Messages Consumed
5	Messages Ignored

Appendix C. Reference

Appendix C.1. Data Types

Each produced data packet contains up to 1400 bytes of unstructured data. The specification of the Ge_data_type in the Map Descriptor tells the driver how to interpret these raw data bytes.

The minimum data unit processed is a byte. This is the case even when the data type is specified as bit. This is because EGD producers cannot produce a single bit. When bits are produced the producer determines the closest byte boundary and sends a minimum of 8 bits.

The following data types are recognized by the driver

Byte	
Bit	(translated as 8bits aligned with a byte boundary)
Word	(unsigned 16bit integer)
Dword	(unsigned 32bit integer)
Int	(signed 16bit integer)
Long	(signed 32bit integer)
Float	(translated as an IEEE 4 byte real number)
Double	(translated as an IEEE 8 byte real number)

The following GE Specific data types are also recognized.

Type	Description P-ProducerC-Consumer
%R	Register memory in word mode P/C
%AI	Analog input memory in word mode P/C
%AQ	Analog output memory in word mode P/C
%I	Discrete input memory in byte mode P/C
%Q	Discrete output memory in byte mode P/C
%T	Discrete temporary memory in byte mode P/C
%M	Discrete momentary memory in byte mode P/C
%SA	Discrete system memory group A in byte mode P/C
%SB	Discrete system memory group B in byte mode P/C
%SC	Discrete system memory group C in byte mode P/C
%S	Discrete system memory in byte mode P
%G	Discrete global data table in byte mode P/C

If you use the RUI editor and view the Map Descriptors online it may appear that the driver changed the data type but in fact all that it has done is changed the display to a synonym.