

Computer Communications for Ethernet Global Data – CommEGD

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.

The IC697CMM742 modules configured with Control and IC693CPU364 and IC200CPUE05 configured with VersaPro can send and receive EGD, but few software vendors support EGD yet. I am developing an Ethernet gateway for Genius networks so any computer or PC control system can read inputs or control outputs using EGD, MODBUS/UDP and serial RTU. There are several sources of RTU master programs, including free source code on our web site. The gateway requires EGD, so this web application note provides source code for an EGD server for GENIGate modules as well as GE Fanuc PLC's and I/O.

CommEGD is provided as sample source code for third party developers to integrate GE Fanuc EGD into their software products. It has the following features:

- Reads (consumes) or writes (produces) EGD to any device (Data port only, not Drives Control port)
- Simple text GEFComm.ini file defines all EGD exchanges and timing. Edit with LMSetup program
- Dynamically configures any number of devices or EGD exchanges of up to 1400 bytes each
- Software creates separate threads to handle Ethernet UDP messages, (future TCP and serial data)
- Internal computer reference table created to store I, Q, AI, AQ, M and R data received or to be sent
- Applications call gefRead/WriteComputerMemory to access internal reference tables with overrides
- Routines pack bit data on write and unpack bits to 16-bit integer on read for simple VisualBASIC use
- In the [Open Source](#) code tradition, source code is provided with a royalty free right to copy and use

Note all current source code is provided, but **the multithreaded HostEGD using the routines is not complete**. The two EGD performance testing programs, ReflectEGD and StressEGD, are complete and include source code. Check the GE Fanuc PLC Tech Support web site for updates.

EGD Data Message Format

The EGD specification covers two types of UDP datagrams: Data messages sent to the Data port 0x4746 and Control messages sent to the Control port. Control messages provide a reply message to acknowledge every request message. GE Drives systems support Control messages, but they will not be discussed here as EGD Control messages have currently not been implemented in any GE Fanuc products.

The following standard C code defines the EGD Data message format:

```
#pragma pack(2)          /* change default packing from 4 or 8 bytes to 2 */
#define GEF_EGD_UDP_DATA_PORT 0x4746 /* Letters GF are used as port for
EGD Data messages */
typedef struct {
```

```

        unsigned short PDUTypeVersion;          /* Type=13 (0Dh) in low byte,
version=1 in hi */
        unsigned short RequestID;               /* incremented every time
data produced */
        unsigned long  ProducerID;              /* The TCP/IP address of
device sending EGD */
        unsigned long  ExchangeID;             /* A unique Producer number
identifying the data */
        unsigned long  TimeStampSec;           /* Timestamp seconds since 1-
Jan-1970 */
        unsigned long  TimeStampNanoSec;       /* and number of nanoseconds
in current second */
        unsigned long  Status;                 /* In low word, upper word
reserved and set to 0 */
        unsigned long  ConfigSignature;        /* In low word, upper word
reserved and set to 0 */
        unsigned long  Reserved;               /* word set to 0 */
        unsigned char  ProductionData[1400];   /* PLC or I/O data to be sent
as EGD */
    } GEF_EGD_DATA;
GEF_EGD_DATA MessageEGD;

```

EGD Data messages are sent from Producer to Consumer on a scheduled basis. A 32-byte header with the following fields precedes each EGD data message:

Field	Description
PDUTypeVersion	has a 13 in the low byte and the current version of 1 in the high byte.
RequestID	16-bit number for each ExchangedID incremented by producer
ProducerID	TCP/IP addresses of the sender based on EGD config, not actual address
ExchangeID	Unique number for each Producer used to identify data, from EGD config
TimeStamp	Two 32-bit numbers with seconds and nanoseconds since 1-Jan-1970
Status	Message status, 1=Success, others in Table 4-3 in GFK-1541A manual
ConfigSignature	For security use, Not implemented yet and must be set to 0

Up to 1400 bytes of PLC or I/O data follows the header. Reading EGD data in a computer is very simple. All that is required to exchange Ethernet Global Data is to open a **socket** for UDP Datagrams and **bind** it to port "GF". The application uses **recvfrom** to read (consume) data produced by other devices or **sendto** to write (produce) data to other devices.

```

SOCKET HostSocket, TargetSocket;
struct in_addr HostAddress, TargetAddress, FromAddress;
struct sockaddr_in
HostSocketAddress, TargetSocketAddress, FromSocketAddress;
long ByteLength, DataLength, FromLen, TargetID;

HostSocket = socket(AF_INET, SOCK_DGRAM, 0);
memset(&HostSocketAddress, 0, sizeof(HostSocketAddress));
HostSocketAddress.sin_family = AF_INET;
HostSocketAddress.sin_port = htons(GEF_EGD_UDP_DATA_PORT);
bind(HostSocket, &HostSocketAddress, sizeof(HostSocketAddress));

```

```

    FromLen = sizeof(FromSocketAddress);
    ByteLength
=recvfrom(HostSocket, &MessageEGD, sizeof(MessageEGD), 0, &FromSocketAddress,
&FromLen);
    DataLength = ByteLength - 32;
    if ((DataLength>0)&&(MessageEGD.PDUTypeVersion==0x010D)) {
        memcpy(&FromAddress, &MessageEGD.ProducerID, sizeof(DWORD));
        printf("\nReceived %u data bytes from %s Exchange %u, Request
%u", DataLength,
(char *)inet_ntoa(FromAddress), MessageEGD.ExchangeID, MessageEGD.RequestID);
    }

```

Code to write EGD to another TCP/IP Address is just as easy. There are a few extra lines if the target is specified by name rather than by TCP/IP address. The code is shown in the ReflectEGD.c example file.

```

    TargetSocket = socket(AF_INET, SOCK_DGRAM, 0);
    TargetID = inet_addr(TargetTCPIPAddressInDottedDecimalFormat);
// More code is required if remote target specified by name, see
ReflectEGD.c

memcpy(&TargetSocketAddress, &HostSocketAddress, sizeof(HostSocketAddress));
memcpy(&TargetSocketAddress.sin_addr, &TargetID, sizeof(long));
// Fill in MessageEGD with Header and DataLength bytes of Data (code not in
ReflectEGD.c)
    MessageEGD.PDUTypeVersion = 0x010D;
    MessageEGD.RequestID++;
    MessageEGD.ProducerID =
inet_addr(HostTCPIPAddressInDottedDecimalFormat);
    MessageEGD.ExchangeID = 1;
    MessageEGD.TimeStampSec = time(NULL);
    MessageEGD.TimeStampNanoSec = 0;
    MessageEGD.Status = 1;
    MessageEGD.ConfigSignature = 0;
    MessageEGD.Reserved = 0;
    memcpy(MessageEGD.ProductionData, pDataToSend, DataLength);
    ByteLength = DataLength + 32;

sendto(TargetSocket, &MessageEGD, ByteLength, 0, &TargetSocketAddress, sizeof(Ta
rgetSocketAddress));

```

A Simple Example EGD Program (Separate from CommEGD)

CommEGD code has many features beyond EGD that may make it appear more complex than it really is. What is needed is a simple working application that shows how to receive and send EGD without configuration validation and setup details.

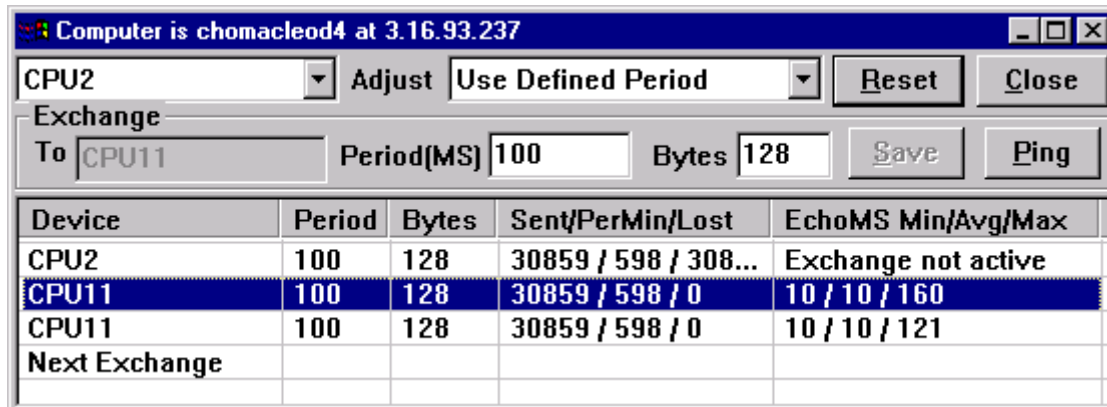
The ReflectEGD program below is a sample that accepts EGD messages from any device and sends them to another TCP/IP address. It acts like a mirror reflecting beams of light, hence the program name. The program is designed as the simplest example of reading and writing EGD messages without any configuration or other overhead. It might actually be useful to display what EGD messages are being sent to a specified PLC if you ran it on a computer set to the same TCP/IP address as the PLC.

The sample code fragments in the previous section are from the ReflectEGD.c file. You can load this file into a C compiler editor or any other editor such as NotePad. The comment lines at the start describe how to run the program to display EGD being sent to your computer by a GE Fanuc PLC. You can enter another TCP/IP on the command line to resend received EGD to another device.

A Simple EGD Performance Test Program (Also separate from CommEGD)

Both ReflectEGD and CommEGD are 32-bit console mode programs that lack a graphic interface with continuous updates. Customers also ask for EGD performance data. StressEGD is a simple program

to display EGD performance data by sending EGD messages between computers and recording the minimum and maximum time and messages per minute. The only screen in StressEGD is shown below:



Device	Period	Bytes	Sent/PerMin/Lost	EchoMS Min/Avg/Max
CPU2	100	128	30859 / 598 / 308...	Exchange not active
CPU11	100	128	30859 / 598 / 0	10 / 10 / 160
CPU11	100	128	30859 / 598 / 0	10 / 10 / 121
Next Exchange				

Before running StressEGD, open a command prompt window on other computers on your network and start ReflectEGD programs specifying the echo option on the command line: **reflectegd echo** Record the computer name or TCP/IP address displayed on each computer. Start the StressEGD program and click on the Next Exchange line. Enter the other computer TCP/IP address or name in the edit box after To and the Exchange Period (10 to 3600000) and Byte Length (1 to 1400) edit boxes and click on the Save button. You can click the Ping button to test communication to the computer address or name in the To edit box.

StressEGD sends EGD messages to other computers based on the Period. ReflectEGD programs on other computers echo the same data back and StressEGD displays message turnaround times in the last column. You can use the Adjust list box to try different message rates: Defined period, Half or Double the defined period or 0 for as fast as possible. Click the Reset button after adjusting the period to reset the message Sent Counts and Echo Times at the bottom display.

If you create a GEFComm.ini EGD configuration file (discussed in the next section), the list box at the upper left contains all devices with defined EGD exchanges. File GEFComm.ini for the screen above is:

```
[CPU1]
TCPIP = chomacleod4 ; Computer where StressEGD is run
Consume1CPU2 = 200,%R100 ; consume from ExchangeID 1, CPU2
Consume1CPU11= 200,%R101 ; Timeout and status location not used
Consume2CPU11 = 200,%R102 ; Need Consume line for each exchange
[CPU2]
TCPIP = chomacleod1 ; computer where ReflectEGD was not run
Produce1 = 100,%A11(64) ; 1 is Producer ExchangeID, 100MS Period
[CPU11]
TCPIP = 3.16.89.33 ; computer where ReflectEGD was running
Produce1 = 100,%A11(64) ; 1 is Producer ExchangeID for this CPU
Produce2 = 100,%A1100(64) ; Period also 100 MS, 128 byte message
```

StressEGD lists all exchanges at the bottom with turnaround times for computers running the ReflectEGD program. It displays "Exchange not active" for PLC's or devices not running ReflectEGD. The exchange count limit of 1000 is set by GEF_MAX_EXCHANGE_COUNT in the StressEGD.c file

Configuration File for the CommEGD Program

The examples above show how reading and writing EGD data is simple on a computer. The problem is writing code to handle many exchanges from different devices. Control and VersaPro PLC programmers have EGD configuration integrated into the product, but this does not help computer users writing their own custom applications. What is required is a simple text file to define all EGD exchanges between all PLC's, computers and I/O devices such as the Genius gateways.

The GEFComm.ini text file has been expanded to include plant-wide EGD exchanges. The LMSetup program has defined plant-wide communications using [Section] names based on a three-character device type followed by a number, such as "GIO1" or "PLC3". Types are

CPU A computer with an Ethernet UDP/IP application to exchange EGD

PLC 90-30 CPU364 or 90-70 with CMM742 card configured to exchange EGD

GIO Ethernet to Genius gateway with one or more GEN104 daughter-cards

EIO Ethernet I/O rack configured to send EGD inputs and to receive directed control outputs

The following Key names have been added for EGD and unsolicited CommReq 2010 from PLC's.

ProduceN = PeriodMS,List of %Addresses(Length) of data to send. N is ExchangeID 1 to 9999

ConsumeNSection = TimeoutMS,List of %Addresses(Length) to receive ProduceN in [Section]

Primary = PrimaryPLCSection under backup PLC sections to duplicate I/O exchanges

ReceiveSection = CPUAddress1=PLCAddress1(Length), CPUAddress2=PLCAddress2, etc

GENIX = Genius Config file name where **X** is from 1 to 4 for local PCIM cards

Number **N** in the ProduceN is the ExchangeID number that must be unique under each Producer section. Every section associated with Ethernet devices has a key **TCPIP** specifying the device TCP/IP address in dotted decimal or DNS computer name format. This address is used for the ProducerID when EGD is sent from this device. The **Section** is the section name of the device producing the EGD that is being consumed by this section, such as PLC3 or CPU2.

ProduceN and **ConsumeNSection** key values start with a millisecond Producer Period or Consumer Timeout. These may be followed by a list of addresses. The list of addresses can be any memory type in a PLC section. The first address in PLC (and maybe CPU) lists is reserved for the 16-bit exchange status word. The GIO and EIO sections are limited to %I, %Q, %AI and %AQ while the CPU sections allow these I/O types plus %M and %R for internal memory. Addresses are generally followed by a length in bits or words in () parentheses. EGD only transferred byte data, so discrete addresses are automatically adjusted to start on a byte boundary with a length that is a multiple of 8 bits.

The ProduceM and ConsumeMSection keys should match up across the entire file or the program displays a warning when the file is loaded. The Address list is required at one side of the exchange, but is optional at the other side unless addresses have to be changed to avoid overlaps. Generally the Producer side has the address list and the Consumer list is optional except when the consumer is GIO or EIO. For directed control to I/O, the GIO or EIO Consumer defines the list while it is optional for the Producer. For CPU or PLC devices the Consumer list may be shorter than the Producer and portions of the consumed data can be skipped by adding one or more Skip(ByteCount) fields to the address list.

The **Primary** key is placed in backup PLC or CPU sections to identify the primary controller. The I/O sections, GIO, EIO or others are set to consume directed control messages from the primary controller. This is used to set up duplicate exchanges to backup controllers. The I/O producer/consumer lines must still be duplicated. It would be nice to eliminate this, but it would take 2 extra scans on the config file.

Generally lists have discrete data preceding analog or word data, but this is not required. In the example below, a redundant pair of PLC's are exchanging EGD for synchronization and also controlling 2 Genius gateways while a computer is receiving both PLC and gateway I/O data.

```
[PLC3]
TCPIP = chomacleod4      ; use computer names or numbers like 3.1.1.7
Produce1 =50,%R100       ; list from GIO1 Consumer
Produce2 = 50,%R101      ; list from GIO2 Consumer
Consume1GIO1=150,%R102   ; list from GIO1 Producer
Consume1GIO2=150,%R103   ; list from GIO2 Producer
Produce3 = 80,%R103,%R201(200),%M1(512)
Consume3PLC4=200,%R104
GENI1 = MyGENI.cfg       ; load config file for local PCIM card
[PLC4]
TCPIP=3.1.1.8
```

```

Primary=PLC3
Produce1 =50,%R100      ; list from GIO1 Consumer
Produce2 = 50,%R101      ; list from GIO2 Consumer
Consume1GIO1=150,%R102 ; list from GIO1 Producer
Consume1GIO2=150,%R103 ; list from GIO2 Producer
Produce3 = 80,%R103,%R401(200),%M513(512)
Consume3PLC3=200,%R104
[CPU1]
TCPIP=3.1.1.2
Consume1GIO1=200
Consume1GIO2=200
Consume3PLC3=250
Consume3PLC4=250
[GIO1]
TCPIP=3.1.1.4
Produce1 =50, %I1(256),%AI1(60)
Consume1PLC3=150,%Q1(128),%AQ1(32)
[GIO2]
TCPIP=3.1.1.5
Produce1= 50,%I257(320),%AI70(24)
Consume2PLC3= 150,Q129(64),%AQ40(16)

```

The **ReceiveSection** key has been added to handle unsolicited Ethernet CommReq 2010 in the future, but it has not been programmed. If the CPUAddress= part is omitted, PLCAddress is used in the computer.

The **GENI** (and future PCIF) key defined the configuration file for Genius PCIM cards (and future 90-30 PIF300 and PIF400 cards) that are installed on the local computer to send or receive EGD.

CommEGD Program Operation

The CommEGD program is configured using the same GEFComm.ini file used for LMSSetup and the GEFComm serial, Genius and Ethernet library. The file defines Producer and Consumer exchanges and update times for every computer, PLC, I/O drop and Genius gateway on the network. The program defines internal reference tables with %I, %Q, %AI, %AQ, %R and %M data based on the highest address used. The user application calls the same gefReadPLCMemory and gefWritePLCMemory that were defined for the GENIlib library in 1993. The only different is PLC has been changed to Computer and the first parameter with the PLC number has been dropped.

The CommEGD functions are:

```

Status = gefReadComputerMemory (SNPMemoryType, StartAddress, DataLength,
DataArray)
Status = gefWriteComputerMemory (SNPMemoryType, StartAddress, DataLength,
DataArray)
ExchangeCount = gefEGDLoadConfig (DefaultHostTCPIP, MaxExchangeCount)
Status = gefEGDSuspend (fSuspendTransfer)
Status = gefEGDStatus (fWrite, IndexBase0, &ExchangeStatus, nBytes, pText)
Status = gefEGDConfig (fWrite, IndexBase0, &ExchangeConfig, nBytes, pText)
Status=gefEGDMemoryList
(fWrite,&ExchangeConfig,MemoryCount,pMemoryList,nBytes,pText)

```

Function parameter definitions are:

SNPMemoryType - SNP Memory Type like 8 for registers has predefined name PLCMemoryTypeR
StartAddress - Starting address for specified memory type, starts at 1
DataLength - Length in 16-bit words or bits for discrete data with MemoryType>PLCMemoryTypeAQ
DataArray - Address for transferred data (ByRef for VB), bit data stored as 1 bit per 16-bit word
DefaultTCPUPHost - Default TCP/IP address to load from config file, Use 0 for current computer
MaxExchangeCount - Maximum number of EGD exchanges on this computer, Use 0 to close
ExchangeCount - Returns the number of defined, active exchanges. Can disable or

change config

fSuspendTransfer - Call with 1/TRUE to stop transfer to computer memory, 0/FALSE to restart

fWrite - Flag set to 0/FALSE to read data or set to 1/TRUE to write data back (3 may Disable)

IndexBase0 - Index for Exchange data from 0 to ExchangeCount-1

ExchangeStatus - Read message counts and last time stamp or write to reset counts or change Enable

ExchangeConfig - Can read or change EGD configuration. Can also enumerate all defined exchanges

nBytes - Maximum number of text bytes returned with starting at pText, Use 0 if text not required

pText - Start of text buffer to return status or config information at text for display, NULL if not needed

MemoryCount - Number of memory segments in the pMemoryList, limit of 100 in PLC exchange

pMemoryList - Array of memory types, addresses and length of data in an EGD exchange message

The **gefEGDLoadConfig** routine must be called once at the start of a users program to load EGD config information from the GEFComm.ini file and allocate memory. This routine sets up the separate thread to handle Ethernet communications. The DefaultHostTCPIP is normally 0 and the MaxExchangeCount set higher than the number of EGD exchanges for this computer. You should call the same routine with the second parameter = 0 at the end of your program to close the thread and unload Ethernet communications.

When gefEGDLoadConfig is first called, it makes three passes through the GEFCom.ini config file.

1. Find section matching default or current computer TCP/IP address or its Primary section if a backup
2. Save all Consume lines in this section and those in other sections that refer to this section or Primary
3. Save all Produce lines linked to saved Consume lines

If a Primary line points to another controller, Produce and Consume lines associated with GIO or EIO from that controller are duplicated for this section. The memory list defined under I/O sections always take precedence, but controllers may choose to map memory to alternate addresses.

The **gefEGDStatus** routine is used to read or reset status information for a specified EGD exchange. The first parameter is set to FALSE/0 to read or TRUE/1 to write back data. Applications may want to write back status to reset exchange counts or to enable or disable EGD messages. This is like using Genius I/O CommReq #8 to enable or disable directed control from redundant controllers.

There are several structures defined in the CommEGD.h include file. EGD status data accessed by the gefEGDStatus routine is:

```
typedef struct {
    DWORD   ProducerTCPIP; // return TCP/IP address for this exchange
    DWORD   ExchangeID;    // return unique ID for this Producer
    DWORD   ExchangeCount; // sent or received, can 0 if fWrite is TRUE
    DWORD   ErrorCount;    // Timeouts for consumer, other
    producer errors
    DWORD   TimeStampSec;  // EGD time stamp for last message received
    DWORD   TimeStampNanoSec;
    long    TimeTillTransferEGD; // MilliSec remaining, minus if
    overdue;
    WORD    RequestID;     // Producer increments when EGD
    message sent
    short   EnableExchange; // set to 1 to enable, 0 to disable if fWrite
} GEF_EGD_EXCHANGE_STATUS;
```

The **gefEGDConfig** and **gefEGDMemoryList** routines are used to read or write configuration data for a specified EGD exchange. Configuration data includes the following:

```
typedef struct {
    short   SNPMemoryType; // SNP Memory Type, %AI=10, %I=16, etc
```

```

        short   StartAddress;   // Addresses start at 1
        short   DataLength;     // In words or bits if SNPMemoryType
> %AQ
} GEF_PLC_MEMORY_LIST;
typedef struct {
        DWORD   ProducerTCPIP;  // If equal to computer TCP/IP, it is
producer
        DWORD   ExchangeID;     // Unique number for each producer
from 1 to N
        short   DeviceType;     // 0=CPU, 1=PLC, 2=GPIO, 3=EIO, etc
        short   DeviceNumber;   // 1 to 9999
        DWORD   ConsumerTCPIP;  // Set to 0 if this computer is consuming
        DWORD   ProducerPeriod; // 10 to 3600000 (1 hour) milliseconds
        DWORD   ConsumerTimeout; // 10 to 3600000 millisecfor timeout,
0=none
        short   DataByteLength; // configured transfer length, EGD limit 1400
        short   MemoryListCount; // same as AddressSegment count
        long    PLCStatusTypeAddress; // SNP Type in upper, address in
lower
        short   DataBytesReceived; // Set when first EGD message
received
} GEF_EGD_EXCHANGE_CONFIG;

```

You can enumerate all defined exchanges by calling either routine with an IndexBase0 stepping from 0 to 1 less than the ExchangeCount returned by the gefEGDLoadConfig routine.

```

char Text[4096];
for (Index=0; Index<ExchangeCount; Index++) {
        if (gefEGDExchange(FALSE, Index, &ExchangeConfig,4096,Text)>0) {
                printf(&quot;\nExchange %u = %s&quot;;Index, Text);
        }
}

```

The separate EGD processing thread may be accessing configuration and internal reference table data at the same time a user application is calling CommEGD routines. This is not normally a problem when reading tables or writing to word memory that EGD is sending. There may be a very small chance of conflicts writing to discrete data and there is a much larger chance that writing status or config data back will conflict with the EGD thread accessing to that same data. To prevent memory conflicts, you should call the **gefEGDSuspend** (TRUE) before calling routines to write data. When the routine returns, the EGD thread has acknowledged that it has stopped accessing internal memory. Call gefEGDSuspend (FALSE) when you are done to restart EGD transfers.

The gefEGDStatus, gefEGDConfig and gefEGDMemoryList routines give you complete control over configuring EGD exchanges dynamically similar to Series 90-70 logic driven Dynamic EGD setup using SvcReq #44. The configuration only applies to the local computer and not to a remote PLC. Note Genius gateways will accept remote EGD configuration from the computer for a single point of connect.

Sample User Application Program - HostEGD

A short console mode HostEGD application was developed to demonstrate calls to the CommEGD routines. To test, you need at least one other system that can produce and consume EGD. If you do not have a GE Fanuc IC697CMM742 with the Control programmer or IC693CPU364 with VersaPro™, you can use two computers connected by Ethernet running the HostEGD program.

Create or update the GEFComm.ini file in the same directory as the HostEGD program. Add sections for each PLC or computer and include the TCPIP line and all Produce and Consume lines under each section. For PLC sections, Produce lines are *ProduceM=Period,StatusWord,Address(Length),Address(Length),,,* For Consume lines, the memory list can be omitted if it is to be the same as the Produce list. You need to include the ExchangeID and Producer name and at least the Timeout and StatusWord after the = sign.

A sample GEFCComm.ini exchanging 200 registers between computer and PLC every 100 milliseconds is:

```
[CPU1]                ; First computer running HostEGD
TCPIP = mycomputername ; computer name or TCP/IP address
Produce1 = 100,R201(200) ; 100 MS period, no status, send from R201
Consume1PLC1 = 300      ; 300 MS timeout, receive R1(200) set by PLC
[PLC1]                 ; PLC configured for EGD
TCPIP = 3.1.1.4         ; Address of PLC, can also use PLC DNS name
Produce1 = 100,R400,R1(200) ; 100 MS period, Status=R400, send 200 R'2
Consume1CPU1 = 300,R401 ; 300 MS timeout,status=R401,receive
R201(200)
```

If you are not able to configure a PLC for EGD, change PLC1 to CPU2 above and the TCPIP address under the [CPU2] section to the second computer name. You must also drop the R400 and R401 status words as computers use gefEGDStatus rather than the EGD status words used in the PLC.

Run HostEGD on both computers using the same GEFCComm.ini file on both. The gefEGDLoadConfig routine will find the correct section based on the local computer name and set up all EGD exchanges. Remember the GEFCComm.ini file is designed to define all PLC, computer and distributed I/O on the plant network which eliminates maintaining separate config files for EGD (a Global config for Global Data).

Starting HostEGD on CPU1 will show 2 exchanges. Enter a 0 or 1 at the command prompt to view status, config and reference table data for either exchange. Type in any address followed by a length in () to view any data in reference table format, such as R201(100). To set any data, enter an address followed by an = sign and 1 or more values, such as R1=1,2,7,4,5,-600 or Q25=1100010101010 to set one or more bits.

Files included in the CommEGD.zip file are:

File Name	Description of file
HostEGD.c	Source code for test program with routines to parse input and show reference tables
HostEGD.dsp/dsw	HostEGD Project files for Microsoft™ Visual C++™ v6.0 compiler
HostEGD.exe	Compiled program. Edit GEFCComm.ini to add EGD exchanges & TCPIP addresses
GEFCComm.ini	Sample communication definition text file, can update with any text editor
CommEGD.doc	This document in Microsoft® Word™ format
CommEGD.c	Source code for the EGD routines
CommEGD.h	Include file with data structures and function prototypes for EGD routines
MyGENI.cfg	Sample Genius network config file for Genius gateway
ReflectEGD.c	The complete source code for the EGD reflection program discussed on page 2
ReflectEGD.exe	Compiled version of the reflector. Comments at start of source tell how to program
StressEGD.c/rc	Source code for graphic program to send EGD to ReflectEGD and record times
StressEGD.dsp/dsw	StressEGD Project files for Microsoft Visual C++ v6.0 compiler
StressEGD.exe	Compiled program. Edit GEFCComm.ini to add EGD exchanges & TCPIP addresses

The current code is most useful in a C/C++ environment where the source code can be linked into the application. In the future, the CommEGD routines will be linked into the GEFComm.dll with headers defined as GEFComm.h for C or GEFComm.bas for VisualBasic™. This will allow 32-bit Windows® applications supporting dll's to use the routines.

Programs call gefReadComputerMemory or gefWriteComputerMemory to access local reference tables. The routines use PLC type reference tables but it is expected that programs will use the memory type C structures or Basic user defined types that define element names for every point using PLC or I/O config tag names. Routines are available to create C and Basic header files from names in the GEFComm.ini file.

Closing Comments

Note ReflectEGD, StressEGD, CommEGD and HostEGD are not GE Fanuc products. Please contact your GE Fanuc Sales or Application engineer or local PLC distributor for information on GE Fanuc PLC products.