

# **z390 Service Oriented Architecture User Guide v1.2.00G**

## **Table of Contents**

- 1. Introduction**
- 2. z390 TCP/IP sockets support**
- 3. z390 SOA application generation support**
- 4. z390 demo SOA application**
- 5. References**
- 6. Appendix**
  - a. Demo application source code**
  - b. Demo application execution log for statically linked base line**
  - c. Demo SOA client application execution log**
  - d. Demo SOA server application execution log**
  - e. Demo SOA client server timing statistics**

## **1. Introductions**

**The z390 Portable Mainframe Assembler open source project now supports low level Service Oriented Architecture (SOA) type application generation and execution. Starting with z390 v1.2.00g, a new macro TCPIO and svc x'7C' support TCP/IP sockets messaging between client and server components written in z390 assembler. z390 client and server programs can also interact with any other clients or servers written in any language supporting compatible messaging via TCP/IP sockets.**

**Starting with v1.2.00g, z390 also includes a new SOA directory with SOA generation macro library and a demo application which can be generated and executed as either a classic statically linked application or an SOA generated client server application.**

**The SOA demo application consists of a z390 mainframe assembler main program DEMOMAIN.MLC which calls two subroutines DEMOSUB1.MLC and DEMOSUB2.MLC. DEMOSUB1 calculates the sum of two numbers with up to 34 digits in scientific notation using extended decimal floating point. DEMOSUB2 calculates the sum of two 32 bit integers.**

**To assemble, statically link, and execute the demo application run soa\demo\demostd.bat. To generate, build, and execute the demo SOA application with the client and server running on the same processor, run soa\demo\demosoa.bat. The demo SOA application client and server can also be run using two separate processors on a cable or wireless network supporting TCP/IP. See Appendix V for statistics showing the timings from each of these network configurations. The conclusion from timings is that the average overhead for TCP/IP messaging is about 2-3 milli-seconds after initialization. This overhead is negligible for services which also perform database I/O. The advantage of using the SOA architecture is that services are more easily shared across diverse applications and user networks and maintenance is simpler since server code does not need to be statically linked into client application code.**

## **2. z390 TCP/IP Sockets Support**

**The macro TCPIO has been added with operations OPEN, CLOSE, SEND, and RECEIVE to support messaging between client and server programs using TCP/IP sockets. Messages of any length can be exchanged between any client and server programs on a TCP/IP network. The server program must first open a socket port, and the client must then open the same port with the host IP address specified for the server.**

**If any TCPIO operation fails for any reason on the client or server, a non-zero return code is returned. For examples of how to use this new macro service, see the SOA generated client and server message managers (soa\demo\democmgr.prn and soa\demo\demosmgr.prn). If the TRACE and CON options are specified on the SOAGEN macro in soa\demo\demosoa.mlc, then the generated client and server message managers will display trace of all instructions including additional trace information on each TCPIO svc plus memory dump of each message sent and received. Note there are two separate logs for the client and server when running the demo SOA application. The server must be closed normally to see the generated log file. The utility DEMOSTOP.MLC can be assembled, linked, and executed to shut down the demo server by running soa\demo\demostop. The shut down server function is triggered by sending a message from the client with -1 length in the first 4 bytes of message. All other messages in the demo have 4 byte length of message in the first 4 bytes so the server and client can read variable length messages by first reading the first 4 bytes and then the rest in one TCPIO receive request which blocks task until the requested number of bytes are successfully read via TCP/IP.**

### **3. z390 SOA application generation support**

The z390 SOA application generation macro SOAGEN can be used to generate customized client and server message managers plus stubs for each service called by the client application. In addition the SOAGEN macro generates two batch commands to build the SOA application and to execute the client server SOA application. The SOAGEN macro uses the z390 PUNCH extension operands DSNNAME= and FORMAT to generate 3 or more source MLC files plus the two BAT files in one macro execution. The keyword parameters on the SOAGEN macro call are as follows:

1. **MAIN=** - name of main client program. If specified, an assembly and link of the main program with the generated service call stubs will be generated.
2. **CLIENT=** - name of the generated client message manager called from stubs.
3. **SERVER=** - name of the generated service message manager which loads and calls services based on service request messages.
4. **HOST=** - IP address of server host processor or \* for local processor
5. **PORT=** - port # for this application (must be greater than 1023)
6. **SERVICES=** - one or more sublists defining the name of each called service and the length of each parameter being passed to service. If the length is negative, that indicates the parameter is read only and the updated parameter will not be returned in response message.
7. **MACDIR=** - directory containing the SOAGEN macros
8. **GENDIR=** - directory to contain the generated source files and command files
9. **GENBLD=** - name of the generated build command file
10. **GENRUN=** - name of the generated run command file

The client message manager is generated using a call to the SOACMGR macro with the required parameters from the SOAGEN macro call. The client message manager performs the following functions when called from a client service call stub:

1. On first call, dynamically allocate the required message buffer based on the maximum service message required.
2. On first call open a TCP/IP socket connection to the server message manager using the port and IP address specified.
3. Build a send message with message length, time stamp, service name, and all the parameters required by the service. Note the service can only access the parameters passed with the length specified. If a service needs to access additional parameters such as control blocks in memory, they need to all be passed to the service.
4. Send the message from client to server message manager using specified port
5. Wait for response from the server with matching time stamp and service name plus updated parameters, and return code from service.
6. Move the returned updated parameters to the original calling list addresses.
7. exit to calling stub which exits to the calling client main application program.

The server message manager is generated using a call to the SOASMGR macro with the required parameters from the SOAGEN macro call. The server message manager performs the following functions:

1. Opens server socket for specified port and listens for incoming messages from clients.
2. If the first 4 byte message length field is -1, stop the service.
3. Look up the service name specified. If service not found, an error is generated on server log and server returns to get next message.
4. Build call parameter address less pointing to the parameters in the received message. Note this implies that all updates by the service will be made to parameter areas in the message buffer.
5. Load the service on the first call and save entry address.
6. Call the service.
7. Store the service return code in the message buffer.
8. Build return message truncated to just the updated parameters as indicated by positive lengths in the SOAGEN SERVICES parameter.
9. Send response message back to client message manager.
10. Return to wait for next service request message.
11. If client disconnect occurs, an error is send to server log and server returns to wait for next request message from new client.

Stubs for each service name called by the client application are generated using calls to the macro SOASTUB. The functions performed by the generated stubs are:

1. On first call load the client message manager and save address.
2. Call the client message manager passing the name of the service and the calling parameter list.
3. Upon return, exit to caller with return code.

If the GENBLD parameter specifies a name, then the SOAGEN macro will generate a batch command file which assembles each of the above source programs to create an executable SOA type client server application. If the GENRUN parameter specifies a name, then the SOAGEN macro will generate a batch command to start the named SOA server message manager on the same processor, and then run the client application. If the HOST parameter specifies a different processor, the generated server message manager will need to be copied to that processor and started prior to running the client application.

#### 4. Demo SOA application

z390 SOA support includes a demo SOA application in the directory SOA\DEMO consisting of a main program DEMOMAIN.MLC which calls two subroutines DEMOSUB1.MLC and DEMOSUB2.MLC. The first subroutine calculates the sum of two numbers using extended decimal floating point, and the second subroutine adds two integer numbers. As a base line, the demo application can be assembled, statically linked, and executed using the command soa\demo\demostd.bat. See appendix II for the DEMOMAIN.LOG listing showing the results of static calls to each of the two subroutines 5 times including elapsed time measurements.

The same demo application can be generated as an SOA client server application using the command soa\demo\demosoa.bat. See III and IV for DEMOMAIN.LOG and DEMOSMGR .LOG showing the demo client and server logs. Note the difference in calculated end to end service response times and the fact that WTO messages from the services appear on the DEMOSMGR log instead of the DEMOMAIN log. Response time comparisons for the demo application when run on same processor, two processors on local LAN, and two processors on wireless LAN are summarized in Appendix V.

The soa\demo\demosoa.bat command executes the user defined SOAGEN macro call defined in soa\demo\demosoa.mlc:

```
SOAGEN MAIN=DEMOMAIN,          MAIN CLIENT APPLICATION PGM   X
      CLIENT=DEMOCMGR,         SOA CLIENT MSG MGR NAME      X
      SERVER=DEMOSMGR,        SOA SERVER MSG MGR NAME      X
      HOST=*, (192.168.1.3)    HOST SERVER NAME (*=LOCAL)   X
      PORT=3900,              HOST SERVER PORT              X
      SERVICES=((DEMOSUB1,-45,-45,45), SERVICES WITH PARM LENX
      (DEMOSUB2,-4,-4,4)),    (NOTE -LENGTH FOR READ ONLY) X
      MACDIR=D:\WORK\Z390\SOA\MACLIB, SOA GEN MACRO DIRECTORYX
      GENDIR=D:\WORK\Z390\SOA\DEMO, DIRECTORY FOR SOA APPL   X
      GENBLD=DEMOBLD,         GENERATED BUILD BAT FILE     X
      GENRUN=DEMORUN          GENERATED RUN BAT FILE

END
```

The above SOA application generation macro call generates server message manager DEMOSMGR to run on the same host as client using HOST=\*. To generate the same application to run server on a specific host, change the HOST= parameter to specify the IP address of the desired server. The IP address of any windows PC can be viewed by type IPCONFIG from the command prompt. The same client application generated with a specific host and port, can be run on the same processor or any processor on the same network as the server.

The above SOAGEN macro call generates the following source files in the soa\demo directory using z390 PUNCH extended operands DSNAME= and FORMAT to control PUNCH output files:

1. DEMOCMGR.MLC - source macro call to SOACMGR to generate SOA client message manager for the demo application.

2. DEMOSMGR.MLC - source macro call to SOASMGR to generate SOA server message manager for the demo application.
3. SOA\_STUB\_DEMOSUB1.MLC - source macro call to SOASTUB to generate SOA stub for DEMOSUB1 service call.
4. SOA\_STUB\_DEMOSUB2.MLC - source macro call to SOASTUB to generate SOA stub for DEMOSUB2 service call.
5. DEMOBLD.BAT - generated command file to build the SOA demo application.
6. DEMORUN.BAT - generated command file to start the DEMOSMGR server on the same processor and then run the DEMOMAIN client application.

The DEMOBLD.BAT command file assembles and links the following executable client server programs:

1. DEMOCMGR.390 - client message manager
2. DEMOSMGR.390 - server message manager
3. DEMOMAIN.390 - main application program statically linked with the two stubs assembled from above stub source code.
4. DEMOSUB1.390 - service loaded and executed by DEMOSMGR when requested via message from DEMOCMGR.
5. DEMOSUB2.390 - service loaded and executed by DEMOSMGR when requested via message from DEMOCMGR.

#### 4. References:

- **SOA**
  - [IBM - Migrating to SOA](#)
  - [JavaWorld definition of XML based SOA architecture](#)
  - [Microsoft SOA](#)
  - [Software Developers Introduction to SOA](#)
  - [Sun SOA and Web Services](#)
- **TCP/IP**
  - [TCP/IP Transmission Control Protocol RFP](#)
  - [Sockets Tutorial](#)
  - [J2SE ServerSocket Class](#)
  - [J2SE Socket Class](#)
- **Host IP Addressing**
  - [J2SE InetAddress Class](#)
  - [IP Addressing RFC](#)
- **Socket Ports**
  - [Choosing a Socket Port](#)
  - [Registered Ports](#)
  - [Register a Port](#)

For latest z390 downloads and additional information visit [www.z390.org](http://www.z390.org)

## 5. Appendix

### Appendix I: Demo application source code

```
*****
* Copyright 2006 Automated Software Tools Corporation *
* This source code is part of z390 assembler/emulator package *
* The z390 package is distributed under GNU general public license *
* Author - Don Higgins *
* Date - 12/26/06 *
*****
* 12/31/06 RPI 523 CODE DEMOMAIN CALLING DEMOSUB1/DEMOSUB2
* 01/08/07 RPI 523 ADD TIMING IN MICRO-SEC AND PERFORM TWICE
*****
* 1. DEMOMAIN CALLS DEMOSUB1 TO CALCULATE AND DISPLAY SUM OF 2
* DISPLAY SCIENTIFIC NOTATION VALUES USING CFD AND CTD MACROS.
* FOR CONVERSION TO/FROM LD EXTENDED DFP FORMAT FOR ADDITION
* 2. DEMOMAIN CALLS DEMOSUB2 TO CALCULATE AND DISPLAY SUM OF 2
* INTEGER VALUES.
* 3. RUN SOA\DEMO\DEMOSTD.BAT TO ASSEMBLE, STATICALLY LINK, AND
* EXECUTE DEMO APPLICATION AS STANDARD LOCAL SINGLE PROCESS.
* 4. RUN SOA\DEMO\DEMOSOA.BAT TO GENERATED, ASSEMBLE, LINK, AND
* EXECUTE DEMO APPLICAITON USING SOA CLIENT SERVER TO ALLOW
* RUNNING THE TWO CALLED SUBROUTINES AS SERVICES RUNNING ON
* SEPARATE PROCESS ON SAME OR ANY TCP/IP CONNECTED PLATFORM.
*****
COPY ASMMSP
DEMOMAIN SUBENTRY
WTO 'DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION'
LA R11,5 REPEAT 5 TIMES TO CHECK TIMING CONSISTENCY
LOOP EQU *
BAL R12,START_TIME
CALL DEMOSUB1,(DFP1,DFP2,DFP3),VL
MVC DSUM1,DFP3
WTO MF=(E,WTOMSG1)
BAL R12,STOP_TIME
IF (CLC,DFP3,NE,DFP4)
WTO 'DEMOMAIN DEMOSUB1 DFP SUM INVALID - ABORTING'
ABEND 111
ENDIF
BAL R12,START_TIME
CALL DEMOSUB2,(INT1,INT2,INT3),VL
L R0,INT3
CVD R0,PWORK
MVC DSUM2,MASK2
ED DSUM2,PWORK+4
WTO MF=(E,WTOMSG2)
BAL R12,STOP_TIME
IF (CLC,INT3,NE,INT4)
WTO 'DEMOMAIN DEMOSUB2 INT SUM INVALID - ABORTING'
ABEND 111
ENDIF
BCT R11,LOOP
WTO 'DEMOMAIN ENDED OK'
SUBEXIT
START_TIME DS 0H
```

```

        TIME  NS,NS_START
        BR    R12
STOP_TIME DS 0H  SHOW SRERVICE TIME IN MICRO-SECONDS
        TIME  NS,NS_STOP
        LG    R1,NS_STOP
        SG    R1,NS_START
        DSG   R0,=FD'1000'
        CVD   R1,PWORK
        MVC   DMICS,MICS_MASK
        ED    DMICS,PWORK+3
        WTO   MF=(E,SHOW_MSG)
        BR    R12
NS_START DC    D'0' START TOD IN NANO-SECONDS
NS_STOP  DC    D'0' END   TOD IN NANO-SECONDS
SHOW_MSG DC    AL2(SHOW_END-*,0),C'SERVICE TIME IN MIRCO-SEC ='
DMICS    DC    C' ZZZ,ZZZ,ZZZ'
SHOW_END EQU   *
MICS_MASK DC   X'40202020',C',',X'202020',C',',X'202120'
        LTORG
WTOMSG1 DC    AL2(WTOEND1-*,0),C'DEMOMAIN DEMOSUB1 DFP SUM='
DSUM1   DC    CL45' '
WTOEND1 EQU   *
WTOMSG2 DC    AL2(WTOEND2-*,0),C'DEMOMAIN DEMOSUB2 INT SUM='
DSUM2   DC    C' Z,ZZZ,ZZZ'
WTOEND2 EQU   *
MASK2   DC    C' ',X'20',C',',X'202020',C',',X'202120'
PWORK   DC    PL8'0'
DFP1    DC    CL45'1.1'
DFP2    DC    CL45'2.2'
DFP3    DC    CL45' '
DFP4    DC    CL45'3.3'  VERIFY SUM VALUE
INT1    DC    F'1'
INT2    DC    F'2'
INT3    DC    F'0'
INT4    DC    F'3'
        EQUIREGS
        END

```

```

*****
* Copyright 2006 Automated Software Tools Corporation          *
* This source code is part of z390 assembler/emulator package *
* The z390 package is distributed under GNU general public license *
* Author - Don Higgins                                       *
* Date   - 12/26/06                                         *
*****
* 12/31/06 RPI 523 CODE EXAMPLE APPLICATION DEMOSUB1 ROUTINE
*****
* CALC SCIENTIFIC DISPLAY PARM1 + PARM2 = PARM3 USING EXTENDED DFP
*****

```

```

DEMOSUB1 SUBENTRY
        LM    R3,R5,0(R1          LOAD 3 PARM ADDRESSES
        WTO   'DEMOSUB1 ENTERED'
        CFD   CFD_LD,IN=(R3),OUT=0  F0,R2 = LD PARM1
        CFD   CFD_LD,IN=(R4),OUT=1  F1,R3 = LD PARM2
        AXTR  0,0,1                F0,R2 = LD PARM1 + LD PARM2

```

```

CTD   CTD_LD,IN=0,OUT=DSUB  DISPLAY VALUE OF SUB
WTO   MF=(E,WTOMSG)        DISPLAY PARM3 = LD PARM1 + LD PARM2
MVC   0(L'DSUB,R5),DSUB     UPDATE PARM3 FROM DSUB
WTO   'DEMOSUB1 EXITING'
SUBEXIT
WTOMSG DC  AL2(WTOEND-*,0),C'DEMOSUB1 SUM='
DSUB   DC  CL45' '
WTOEND EQU  *
EQUIREGS
END

```

```

*****
* Copyright 2006 Automated Software Tools Corporation *
* This source code is part of z390 assembler/emulator package *
* The z390 package is distributed under GNU general public license *
* Author - Don Higgins *
* Date - 12/26/06 *
*****
* 12/31/06 RPI 523 CODE EXAMPLE APPLICATION CALLED DEMOSUB2 ROUTINE
*****
* CALC INT PARM1 + INT PARM2 = INT PARM3
*****
DEMOSUB2 SUBENTRY

```

```

LM     R3,R5,0(R1)          GET 3 PARM ADDRESSES
WTO   'DEMOSUB2 ENTERED'
L      R0,0(R3)             LOAD  INT PARM1
A      R0,0(R4)             ADD   INT PARM2
ST     R0,0(R5)             STORE INT PARM3
CVD   R0,PWORK
MVC   DSUM,MASK
ED     DSUM,PWORK+4
WTO   MF=(E,WTOMSG)        DISPLAY PARM3 = INT PARM1 + INT PARM2
WTO   'DEMOSUB2 EXITING'
SUBEXIT
WTOMSG DC  AL2(WTOEND-*,0),C'DEMOSUB2 SUM='
DSUM   DC  C' Z,ZZZ,ZZ9'
WTOEND EQU  *
MASK   DC  C' ',X'20',C',',X'202020',C',',X'202120'
PWORK  DC  PL8'0'
EQUIREGS
END

```

**Appendix II: Demo application execution log for statically linked base line**

**EZ390I V1.2.00h Current Date 01/10/07 Time 15:35:11**  
**EZ390I Copyright 2006 Automated Software Tools Corporation**  
**EZ390I z390 is licensed under GNU General Public License**  
**EZ390I program = DEMOMAIN**  
**EZ390I options =**  
**DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 2,893**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 2,144**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 3,985**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 1,782**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 3,582**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 980**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 5,431**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**

DEMOMAIN DEMOSUB2 INT SUM= 3  
SERVICE TIME IN MIRCO-SEC = 881  
DEMOSUB1 ENTERED  
DEMOSUB1 SUM=3.3  
DEMOSUB1 EXITING  
DEMOMAIN DEMOSUB1 DFP SUM=3.3  
SERVICE TIME IN MIRCO-SEC = 9,388  
DEMOSUB2 ENTERED  
DEMOSUB2 SUM= 3  
DEMOSUB2 EXITING  
DEMOMAIN DEMOSUB2 INT SUM= 3  
SERVICE TIME IN MIRCO-SEC = 810  
DEMOMAIN ENDED OK  
EZ390I Stats total instructions = 561  
EZ390I Stats current date 01/10/07 time 15:35:11  
EZ390I Stats total seconds = 0  
EZ390I Stats instructions/sec = 5100  
EZ390I total errors = 0  
EZ390I return code(DEMOMAIN)= 0

**Appendix III: Demo SOA client application execution log**

**EZ390I V1.2.00h Current Date 01/10/07 Time 15:37:11**  
**EZ390I Copyright 2006 Automated Software Tools Corporation**  
**EZ390I z390 is licensed under GNU General Public License**  
**EZ390I program = DEMOMAIN**  
**EZ390I options =**  
**DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 696,814**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 7,318**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 7,260**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 4,189**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 10,341**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 5,176**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 4,830**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 4,763**  
**DEMOMAIN DEMOSUB1 DFP SUM=3.3**  
**SERVICE TIME IN MIRCO-SEC = 3,268**  
**DEMOMAIN DEMOSUB2 INT SUM= 3**  
**SERVICE TIME IN MIRCO-SEC = 2,968**  
**DEMOMAIN ENDED OK**  
**EZ390I Stats total instructions = 1317**  
**EZ390I Stats current date 01/10/07 time 15:37:12**  
**EZ390I Stats total seconds = 0**  
**EZ390I Stats instructions/sec = 1719**  
**EZ390I total errors = 0**  
**EZ390I return code(DEMOMAIN)= 0**

**Appendix IV: Demo SOA server application execution log**

**EZ390I V1.2.00h Current Date 01/10/07 Time 15:37:12**  
**EZ390I Copyright 2006 Automated Software Tools Corporation**  
**EZ390I z390 is licensed under GNU General Public License**  
**EZ390I program = DEMOSMGR**  
**EZ390I options =**  
**DEMOSMGR STARTED**  
**DEMOSMGR REPLY ANY CHAR TO TERMINATE**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 2,466**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 2,053**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 3,814**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 1,646**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 7,460**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 1,291**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 2,216**  
**DEMOSUB2 ENTERED**  
**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 1,163**  
**DEMOSUB1 ENTERED**  
**DEMOSUB1 SUM=3.3**  
**DEMOSUB1 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 1,385**  
**DEMOSUB2 ENTERED**

**DEMOSUB2 SUM= 3**  
**DEMOSUB2 EXITING**  
**SERVICE TIME IN MIRCO-SEC = 895**  
**TCPIO SERVER CONNECTION LOST FOR PORT 3900**  
**DEMOSMGR CLOSING DOWN AT USER REQUEST**  
**EZ390I Stats total instructions = 1111**  
**EZ390I Stats current date 01/10/07 time 15:38:39**  
**EZ390I Stats total seconds = 87**  
**EZ390I Stats instructions/sec = 12**  
**EZ390I total errors = 0**  
**EZ390I return code(DEMOSMGR)= 0**

## Appendix V: z390 Demo SOA Application Timing Statistics

### Z390 Service Oriented Architecture Demo Timings in milli-seconds as of v1.2.00g

Event	Static Link	Single CPU	LAN	Wireless LAN
Initialize server	0	500	500	500
Initialize connection	0	36	36	36
DEMOSUB1 service time	2	2	2	2
DEMOSUB1 total time	2.2	5	5	5
DEMOSUB2 service time	1.5	1.5	1.5	1.5
DEMOSUB2 total time	1.7	3	5	5

#### Conclusions:

1. Static linking will always be faster, but makes sharing services more difficult.
2. For SOA TCP/IP messaging the average overhead is about 2-3 milli-seconds.
3. The major advantage of SOA is that services can be easily shared and maintained.

#### Notes:

1. Initialize server time occurs on first call to service after startup and includes loading services.
2. Initialization connection time only occurs on first call after starting client to establish connection.
3. Service time is the time to execute service on the server as measured by DEMOSMGR.MLC
4. Total time is end to end average time excluding first as measured by DEMOMAIN..MLC
5. Single CPU timings are for both client and server running on same 3.0 GHZ Dell PC
6. 100 MB LAN timing is for client on one PC and server on another PC on same 100 MB LAN
7. Wireless LAN timing is for client on PC Laptop connected to LAN server via wireless router.
8. No changes were made to programs generated by SOA\DEMO\DEMOSO.A.BAT.
9. It may be necessary to add IP address of client to server security system like Norton NIS.