35 TCP/SNA

September 1999

In this issue

3 Easy VTAM monitor

9 OSI explained

13 An SMF termination exit for batch jobs – part 2

29 Enterprise print solutions

38 A mailbox system for SMTP under MVS TCP/IP – 7

68 TCP/SNA news

update

TCP/SNA Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: info@xephon.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions
Articles published in *TCP/SNA Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

TCP/SNA Update on-line
Code from *TCP/SNA Update* can be downloaded from our Web site at http://www.xephon.com/tcpsnaupdate.html; you will need the user-id shown on your address label.

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year's subscription to *TCP/SNA Update*, comprising four quarterly issues, costs £130.00 in the UK; $190.00 in the USA and Canada; £136.00 in Europe; £142.00 in Australasia and Japan; and £140.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the March 1991 issue, are available separately to subscribers for £33.00 ($48.00) each including postage.

Printed in England.

Easy VTAM monitor

This small utility is an easy VTAM monitor that highlights the working logical units connected to a VTAM applid. It's a simple REXX EXEC that uses TSO console service: it has no dependencies on other vendor products, and it's simple and quick.

We use the VTAM applid monitor to see which of the logical units connected to a VTAM applid are sending or receiving data. The applids we check usually are those dedicated to a product that performs software distribution. Sometimes a logical unit remains in an odd state – apparently it is active, but in effect it is not working. So, instead of using our VTAM manager and entering lots of 'D,NET,ID=....' commands, one following the other, we use this small but useful monitor. In a format similar to the display from the 'D,NET' command, we can see highlighted the working logical units and the values pertaining to send and receive changing.

This monitor uses the TSO console interface to issue the D NET commands, the GETMSG function to retrieve the response, and an ISPF table to store the logical units' information. Every time you press 'enter', a new command is issued and the table is refreshed. The monitor's main program is a REXX routine. A curious note about the table display: I couldn't use the TBDISPL command because, after a refresh, not all the table rows nor all the fields in a table row changed. So I had to use a dynamic display and modify the attributes of the varying fields each time. When you refresh the table you can also change the VTAM applid to monitor.

To use the console services you need some RACF definitions: the user who wants to open a TSO console session needs the OPERPARM segment with the following values:

- AUTH = ALL
- LEVEL = ALL
- MFORM = T S J.

Also, you need to define the resource console in the TSOAUTH class and give the user read access to it.

Sometimes, the command issued at the TSO console does not get a prompt reply. Just try it again.

MAIN CODE

```
/* REXX */
/* THE ARG IS THE VTAMID WHOSE ACTIVE SESSIONS YOU WANT TO CHECK */
/* YOU CAN CHANGE THE VTAMID AT EVERY PANEL DISPLAY OVERTYPING   */
/* THE NAME OF THE RESOURCE CURRENTLY DISPLAYED                  */

ARG VTAM

OLDVTAM = VTAM
WL = 14                    /* WINDOW LENGTH */
CUR = Ø                    /* NUMBER OF THE LAST ROW DISPLAYED */
/*********************************/
/* ISPF TABEL INITIALIZATION     */
/*********************************/
ADDRESS ISPEXEC
OK='Ø';"TBCREATE VTAMID KEYS(TERM,SID) ,
 NAMES(STATUS,SEND,RECV,SENDMOD,RECVMOD,LASTTIME) REPLACE"
OK='Ø';"TBSORT VTAMID FIELDS(TERM)"
ADDRESS TSO

/*********************************/
/* TSO CONSOLE INITIALIZATION    */
/*********************************/
USER = SYSVAR(SYSUID)
CART = SUBSTR(USER,1,2)||SUBSTR(USER,6,2)||'CONS'
ADDRESS TSO
"CONSPROF SOLDISP(NO) UNSOLDISP(NO) SOLNUM(4ØØ)"
"CONSOLE ACTIVATE"

/*********************************/
/* DISPLAY LOOP                  */
/*********************************/
FIRST = 'SI'
DO FOREVER
   IF VTAM ¬= OLDVTAM THEN DO
      ADDRESS ISPEXEC
      OLDVTAM = VTAM
      OK='Ø';"TBCREATE VTAMID KEYS(TERM,SID) ,
          NAMES(STATUS,SEND,RECV,SENDMOD,RECVMOD,LASTTIME) REPLACE"
      OK='Ø';"TBSORT VTAMID FIELDS(TERM)"
      ADDRESS TSO
      END
   TABROW = '                                                    '
   TABROW = TABROW||'                  '
   ADDRESS ISPEXEC
```

```
   "VGET (ZVERB)"
   SELECT
      WHEN ZVERB = 'DOWN' THEN DO
         ROWS = NUMROW - CURROW
         IF CURROW < WL THEN FATT = 1
         ELSE
            IF ROWS >= Ø & ROWS < WL THEN DO
               FATT = ((CURROW % WL) * WL) + 1
               "TBTOP  VTAMID"
               "TBSKIP VTAMID NUMBER("FATT")"
               END
         END
      WHEN ZVERB = 'UP' THEN DO
         IF CURROW > WL THEN CURROW = CURROW - 1
         FATT = (((CURROW % WL) - 1) * WL) + 1
         ADDRESS ISPEXEC
         "TBTOP VTAMID"
         "TBSKIP VTAMID NUMBER("FATT")"
         END
      OTHERWISE DO
         NOW = TIME(S)
         LASTTIME = NOW
         ADDRESS TSO
         "CONSOLE SYSCMD(D NET,E,ID="||VTAM||") CART("CART")"
         CALL WRITE_RESP
         IF CURROW <= WL THEN FATT = 1
         ELSE                FATT = (((CURROW - 1) % WL) * WL) + 1
         ADDRESS ISPEXEC
         "TBTOP VTAMID"
         "TBSKIP VTAMID NUMBER("FATT")"
         END
      ZVERB = ''
      "VPUT (ZVERB)"
      END /* END SELECT */
   CALL WRITE_AREA
   CUR = STRIP(CURROW,L,'Ø')
   OK='Ø 4 8';"DISPLAY PANEL(MONVTAMP)"
   TBCC = RC
   IF TBCC = 8 THEN LEAVE
   END

/***************************/
/* END TSO CONSOLE SESSION */
/***************************/
ADDRESS TSO
"CONSOLE DEACTIVATE"
"CONSPROF SOLDISP(YES) UNSOLDISP(YES)"

/******************/
/* END ISPF TABLE */
```

```
/*****************/
ADDRESS ISPEXEC
OK='Ø';"TBEND VTAMID"

EXIT

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *  */

WRITE_RESP:
NUMROW = Ø
GETCODE = GETMSG('CONMSG.','SOL',CART)
IF GETCODE <> Ø THEN DO
   SAY 'MESSAGE NOT RECEIVED'
   RETURN
   END
DO UNTIL (GETCODE <> Ø)
   DO I = 1 TO CONMSG.Ø
      PARSE VAR CONMSG.I MSG .
      IF MSG = 'IST635I' THEN DO
         NUMROW = NUMROW + 1
         PARSE VAR CONMSG.I MSG TERM STATUS SID SEND RECV .
         TERM = LEFT(TERM,8,' ')
         SEND = LEFT(SEND,4,' ')
         RECV = LEFT(RECV,4,' ')
         NEWSEND = SEND
         NEWRECV = RECV
         OK='Ø 8';ADDRESS ISPEXEC "TBEXIST VTAMID"
         IF RC = Ø THEN DO
            OK='Ø';ADDRESS ISPEXEC "TBGET VTAMID"
            IF SEND ¬= NEWSEND THEN DO
               SENDMOD = 'SI'
               END
            ELSE SENDMOD = 'NO'
            IF RECV ¬= NEWRECV THEN RECVMOD = 'SI'
            ELSE RECVMOD = 'NO'
            SEND = NEWSEND
            RECV = NEWRECV
            END
         ELSE DO
            SENDMOD = 'NO'
            RECVMOD = 'NO'
            END
         LASTTIME = NOW
         OK='Ø';ADDRESS ISPEXEC "TBMOD VTAMID ORDER"
         END
      END
   GETCODE = GETMSG('CONMSG.','SOL',CART,,1)
   END
RETURN
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

WRITE_AREA:
ENDTAB = 1
SPACES = '    '
YELL   = '2'
RED    = ''
BLUE   = 'E'
DO WHILE ENDTAB  <= WL
   "TBGET VTAMID POSITION(CURROW)"
   IF RC ¬= Ø THEN RETURN
   IF LASTTIME <> NOW THEN DO
      "TBDELETE VTAMID"
      END
   ELSE DO
      STAT = LEFT(STATUS,8,' ')
      ROW = '       '
      ROW = ROW||YELL||TERM||SPACES||BLUE||STAT||SPACES||BLUE||SID
      IF SENDMOD = 'SI' THEN RESTSEND = RED||SEND||SPACES
      ELSE                   RESTSEND = YELL||SEND||SPACES
      IF RECVMOD = 'SI' THEN RESTRECV = RED||RECV
      ELSE                   RESTRECV = YELL||RECV
      ROW = ROW||SPACES||RESTSEND||RESTRECV
      TABROW = TABROW||ROW
      ENDTAB = ENDTAB + 1
      END
   "TBSKIP VTAMID"
   IF RC ¬= Ø THEN ENDTAB = 17
   END
RETURN

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

CLEANUP:
ADDRESS TSO
"CONSOLE DEACTIVATE"
"CONSPROF SOLDISP(YES) UNSOLDISP(YES)"
ADDRESS ISPEXEC
OK='Ø';"TBEND VTAMID"
RETURN

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

ERRPROC:
IF CONDITION('C')='ERROR' & SYMBOL('OK')='VAR' THEN,
   IF WORDPOS(RC,OK)>Ø | OK='*' THEN RETURN
SIGNAL OFF SYNTAX
SIGNAL OFF NOVALUE
CALL OFF ERROR
```

```
ERROR_TYPE = CONDITION('C')
SAY ERROR_TYPE 'ALLA LINEA' SIGL ':' CONDITION('D')
IF ERROR_TYPE = 'SYNTAX' THEN SAY ERRORTEXT(RC)
IF ERROR_TYPE = 'ERROR' & SYMBOL('ZERRLM') = 'VAR',
   THEN SAY ZERRLM
CALL CLEANUP
```

MONVTAMP

```
)Attr Default(%¬_)
Ø7 type(dataout) intens(high) color(red)    caps(on)
Ø8 type(dataout) intens(low ) color(blue)   caps(on)
Ø9 type(dataout) intens(high) color(yellow) caps(on)
   % type(text  ) intens(high) color(pink)
   £ type(text  ) intens(high) color(yellow)
/* ¬ type(text  ) intens(low )                    information only  */
   # type(text  ) intens(low ) color(green)
   _ type( input) intens(high) caps(on ) just(left ) color(green)
   $ type( input) intens(high) caps(on ) just(left )
   | type( input) intens(high) caps(on) color(Yellow)
   { type(output) intens(high) caps(on) color(Yellow)

 AREA(DYNAMIC) EXTEND(ON) SCROLL(ON)
    AREA(DYNAMIC) EXTEND(Off) scroll(off)
)body expand("") CMD(ZCMD)
Command ===>_zcmd " "
-------------------------- DCPVTAM EASY MONITOR   ----------------------
--
¬
¬                          %====> |vtam                         CUR ¬/
NUMROW
¬
¬    +------------+-------------+-------------------+--------+--------+
¬    | %TERM¬     | %STATUS¬    | %SID¬             | %SEND ¬| %RECV ¬|
¬    +------------+-------------+-------------------+--------+--------+
¬

TABROW ---------------------------------------------------------------

)INIT
 &amt = CSR
)PROC
)END
```

Maria Elena Campidoglio
Systems Programmer
Cedacri Ovest (Italy)

OSI explained

The OSI (Open System Interconnection) was introduced by the International Standards Organization (ISO) in 1978, to create a standardized communications architecture for linking heterogeneous computers. The OSI was originally intended as a detailed specification of interfaces, but the committee decided to establish a common 'reference model' to guide product implementers to develop detailed interfaces that in turn could become standards.

The principal idea in the Open System Interconnection is that the process of communication between any two devices in a network can be divided into layers, where each layer has its own special set of related functions. This layered model obscures the complexities of the lower layer from the hardware or application component using it. The

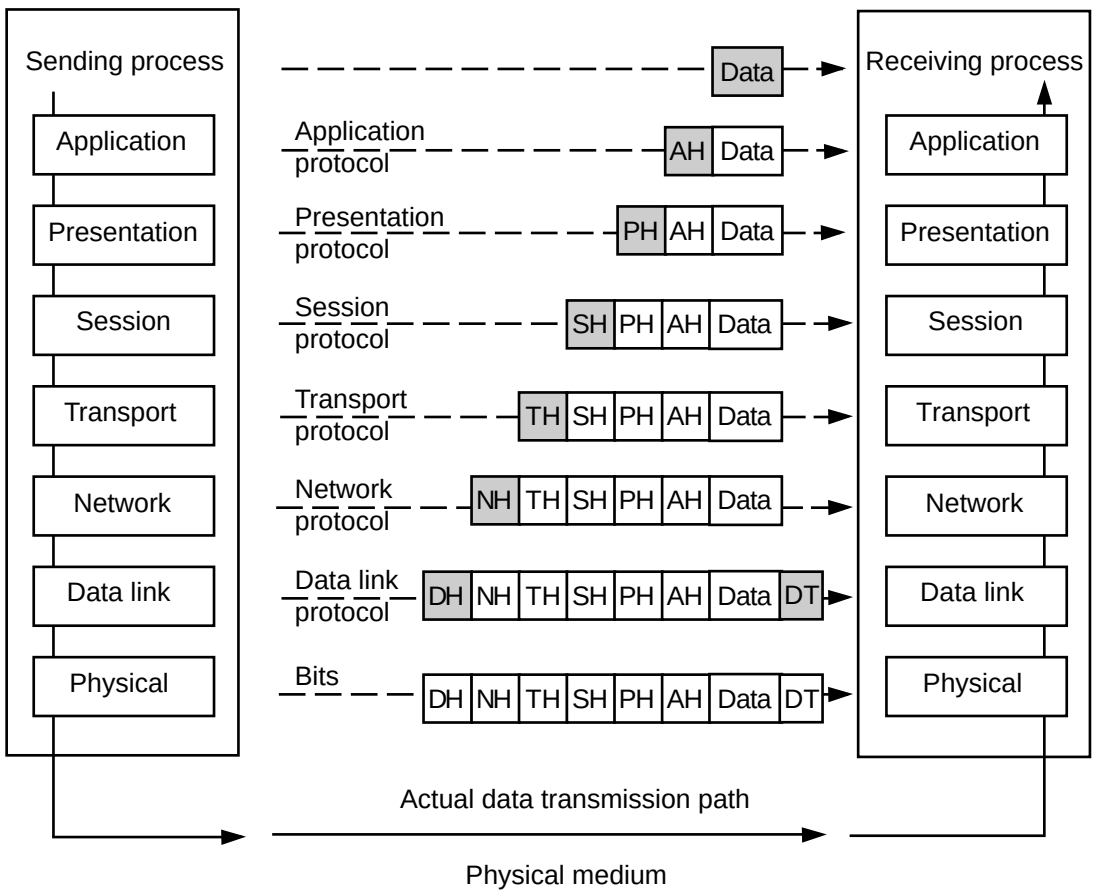

Figure 1: OSI layers

communication functions are broken down into a hierarchical set of layers, so that in any given message between devices there will be a flow of data down through each layer in a device. Each layer relies on the next lower layer to perform a more primitive function, and to obscure the details of those functions. When the data is received at the other end, it travels a reverse path up through the same layers to the receiving device. The layers are defined in such a way that changes in one layer do not also require changes in the other layers.

Designed to establish data communications standards that would promote multi-vendor interoperability, the OSI consists of seven layers, divided into two groups. The upper four layers are used when data passes from or to a user, and the lower three are used when data passes through the host computer.

The seven layers are, starting with the 'lowest' layer in the hierarchy, the Physical, Data Link, Network, Transport, Session, Presentation, and at the 'highest' level, the Application Layer. This is illustrated in Figure 1.

LAYER 1: THE PHYSICAL LAYER

The Physical Layer transmits and receives the unstructured bit stream on a physical medium. It deals with the mechanical, electrical/optical, and functional interfaces to the physical medium, and it carries the signal for all the higher layers.

LAYER 2: THE DATA LINK LAYER

This layer provides error-correction and synchronization for the Physical Layer, although its error control will not guarantee delivery of a message. A number of link layer systems break this layer down into a number of sub-layers. The layer delivers blocks of data (frames) with the necessary synchronization, error, and flow control.

LAYER 3: THE NETWORK LAYER

The Network Layer controls the routing and forwarding of data across a network. It establishes, maintains, and terminates connections, and ensures that data is sent in the right direction and to the right

connection on an out-going message, and receives incoming transmissions at the packet level.

LAYER 4: THE TRANSPORT LAYER

This layer handles end-to-end error checking and flow control, and should provide reliable and transparent data transfer between end points, with messages arriving error-free, in sequence and without duplication or loss.

The level of error detection and recovery built into this layer is dependent on the type of service it receives from the Network Layer. A reliable Network Layer will require a minimal Transport Layer; an unreliable Network Layer will require more substantive error detection and recovery.

LAYER 5: THE SESSION LAYER

The Session Layer controls the creation and termination, exchange, and dialogues of a physical connection between two devices. Session support provides name recognition, security, logging, and so on.

LAYER 6: THE PRESENTATION LAYER

This layer, sometimes called the Syntax Layer and commonly part of the OS, determines how the data is presented to the application, eg EBCDIC or ASCII. In other words, this layer acts as a translator for the network, but can also provide data compression, encryption (passwords etc), and re-formatting.

LAYER 7: THE APPLICATION LAYER

The Application Layer acts as a window for the end user and contains a variety of commonly needed functions such as remote file access, network management, and directory services; it signifies APIs that allow applications to transparently access the services of the lower layers.

Although some applications may perform Application Layer functions, they do not take place in the Application Layer itself.

At Layer 2, hubs broadcast data over the network, while routers and switches work at Layer 3, sending data to specific nodes. However, the latest product developments include Workgroup Switches, OSI Layer 2 switches, that are intended as wiring hub replacement devices. These switches can be used to migrate workgroup networks from shared-media LANs to switched networks, supporting a single end device such as a server, workstation, or switch port. Workgroup Switches could easily replace intelligent wiring hubs as the principal building blocks for workgroup connectivity, because of their performance/cost ratios.

Further developments include Multilayer Switches. These combine the performance gains of Layer 2 devices such as Workgroup Switches, with some Layer 3 intelligence. Multilayer Switches interrogate packets at Layer 3, and dynamically build bridge tables of Layer 3 addresses. Their main use is to supplement the performance of an existing router by front-ending the router and intercepting traffic.

However, as Layer 3 intelligence is hardware-based using ASICs, the protocols supported by Layer 3 are limited to IP or sometimes IPX. Furthermore, Multilayer Switches do not exchange their data with other switches, neither do they support routing update protocols such as RIP or OSPF. Some would now question why would one want to work at Layer 4 or even above.

With Layer 4 now being referred to as Application Switching, you can define full access-control lists enabling you to filter or forward traffic based on UDP or TCP port numbers.

Layer 5 now handles data flows between specific applications by dealing with session and connection coordination. This layer is now being used by some next-generation switches specifically for handling Web and Internet traffic.

Allowing the host to communicate directly with the switch to ascertain appropriate resources for forwarding the data means working at Layer 7.

Layer 4 and 5 switching was inevitable after the hype surrounding

Layer 3 devices. Consequently, we should expect to see our networks littered with Layer *n* switches: devices that will automatically detect the type of incoming traffic and switch it to its destination at the most appropriate level.

Nick Nourse
Independent Consultant (UK)

An SMF termination exit for batch jobs – part 2

This month we continue the article looking at an SMF termination exit.

```
SCALLACT EQU   *
*        CALL ISDACTRT FOR JOB STEP COST COMPUTATION
         SPACE 1
         L     R15,EXDCOMTB        ADDRESS OF COMMON EXIT TABLE
         USING JMR,R15             PROVIDE ADDRESSABILITY
         BAS   R1,CLAMSJB          SET ADDRESS OF SJB IN GPR3
         BZ    RETURN              EXIT IF UNABLE TO LOCATE SJB
         USING SJB,R3              ESTABLISH SJB ADDRESSABILITY
GETJOBN  CLC   JMRJOB,SJBJOBNM     TEST IF SAME JOB
         BE    LOADR3              BRANCH IF SAME JOB
         ICM   R3,15,SJBSJB        OBTAIN ADDRESS OF NEXT SJB ON CHAIN
         BNZ   GETJOBN             LOOP POWER
         B     RETURN              WHY AM I HERE?
         SPACE 1
LOADR3   L     R3,SJBJCT           ADDRESS OF JOB'S JOB CONTROL TABLE
         DROP  R3,R15              REMOVE ADDRESSABILITY
         SPACE 1
         CALL  ISDACTRT,ID=12      ID INDICATES STEP CALL TO ISDACTRT
         EJECT
*************************************************************************
*         MISCELLANEOUS ROUTINES                                        *
*************************************************************************
         SPACE
         USING KEEPSECT,KEEP       PROVIDE KEEPSECT ADDRESSABILITY
         SPACE 1
RETURN   EQU   *
         SPACE 1
         TM    KEEPUSI,4           INVALID PROCESSOR USED
         BNO   NISSØØ5I            NO,SKIP MESSAGE
         MVC   MSG+2(7Ø),ISSØØ5I   MOVE IN MESSAGE
         NI    KEEPUSI,255-4       TURN OFF BIT
```

```
         PLINE
NISSØØ5I DS    ØH
         SPACE 1
         MVC   MSG+1(79),MSG       LINE OF  *'S
         BAS   PLNK,PRINTER
         SPACE 1
*        GO BACK TO INITIATOR
         SPACE 1
DEPART   L     R13,SAVELAST        PICK UP POINTER TO ABOVE AREA
CLAMQUIK L     R14,12(R13)         EXPEDITIOUS RETURN POINT
         LM    R2,R12,28(R13)      RESTORE
         SR    R1,R1               INDICATE TO WRITE THE SMF RECORD
         SR    R15,R15             INDICATE TO CONTINUE PROCESSING
         BR    R14                 RETURN
         SPACE
*********************************************************************
*      PLACE THE ADDRESS OF THE SUBSYSTEM JOB BLOCK IN REGISTER 3     *
*********************************************************************
         SPACE 1
CLAMSJB  L     R3,PSATOLD          I
         USING TCB,R3              SET TCB ADDRESSABILITY
         L     R3,TCBJSCB          ADDRESS OF JSCB
         USING IEZJSCB,R3          SET ADDRESSABILITY TO JSCB
         L     R3,JSCBACT          ADDRESS OF ACTIVE SSIB
         L     R3,JSCBSSIB         SUBSYSTEM IDENTIFICATION BLOCK
         USING SSIB,R3             SET SSIB ADDRESSABILITY
         ICM   R3,15,SSIBSUSE      ADDRESS OF SUBSYSTEM JOB BLOCK
         BR    R1                  RETURN TO CALLER
         SPACE 1
         DROP  R3                  FORGET ADDRESSABILITY
         EJECT
PRINCOST DS    ØH                  FILL LINE
         CVD   R2,TEMPD1           COUNT TO DECIMAL
         ED    COSTCNT,TEMPD1+4    TO MESSAGE
CMRCOST  CVD   R3,TEMPD1           AMOUNT TO DECIMAL
         LA    R1,COSTMASK+7       GET LAST SPOT FOR '$' + 1
         EDMK  COSTMASK,TEMPD1+4   GET TO EBCDIC
         BCTR  R1,Ø                MOVE BACK ONE
         MVI   Ø(R1),C'$'          MOVE IN DOLLAR SIGN
         B     PRINTER             PRINT IT
         EJECT
PRINTHDR EQU   *
         SPACE 1
*        THIS ROUTINE PRINTS THE FIRST LINE FOR EACH EXIT
         SPACE 1
         MVI   MSG,C'*'            SET UP FIRST STAR
         MVC   MSG+1(L'MSG-1),MSG  STAR BURST
         SPACE 2
PRINTER  EQU   *
         CLI   KEEPSMBF,Ø          SMB PRINT FLAG SET
```

```
         BNE   SMBEND              YES,SKIP NEXT
         MVI   KEEPSMBF,8          SET TO NO PRINT
         L     R15,ADDREXD         ADDRESS OF IEFACTRT PARAMETER LIST
         USING EXDDSECT,R15        PROVIDE ADDRESSABILITY TO PARM LIST
         L     R15,EXDCOMTB        ADDRESS OF COMMON EXIT TABLE
         USING JMR,R15             ESTABLISH ADDRESSABILITY
         BAS   R1,CLAMSJB          RETURN WITH ADDRESS OF SJB IN R3
         BCR   8,PLNK              RETURN IF SJB NONEXISTENT
         USING SJB,R3              ESTABLISH ADDRESSABILITY
TESTJOBN CLC   JMRJOB,SJBJOBNM     TEST IF SAME JOB
         BE    LOADJCT             BRANCH IF SO
         DROP  R15                 REMOVE ADDRESSABILITY
         ICM   R3,15,SJBSJB        OBTAIN ADDRESS OF NEXT SJB ON CHAIN
         BNZ   TESTJOBN            LOOP POWER
         BR    PLNK                WHY AM I HERE?
         SPACE 1
LOADJCT  L     R3,SJBJCT           ADDRESS OF JCT FOR JOB
         B     SFLGØ4              BRANCH AROUND DOT PRINT RESTRAINTS
*                                  SINCE IT HAS NO MEANING FOR OTHER
*                                  INSTALLATIONS SUBSCRIBING TO XEPHON
         USING JCT,R3
         CLC   JCTACCTN(4),CMRJJØT DOT TEST JOB
         BE    SFLGØ4              SET TO PRINT SMBS IF SO
         LA    R1,ATABENT          # ENTRIES
         LA    R2,ATABLE           ACCOUNT ENTRIES
SCANACTS CLC   JCTACCTN(2),Ø(R2)   IN THE TABLE
         BCR   8,PLNK              DON'T PRINT IF SO
         LA    R2,2(R2)            OTHERWISE POINT TO NEXT ENTRY AND
         BCT   R1,SCANACTS          TRY AGAIN
         EJECT
**********************************************************************
*        FOR SELECTED ACCOUNT NUMBER PREFIXES, DISALLOW PRINTING     *
*        OF JOB AND STEP STATISTICAL INFORMATION                     *
**********************************************************************
         SPACE 1
         LA    R1,RTABENT          # ENTRIES
         LA    R2,RTABLE           REMOTE ENTRIES
SCANRMTS CLC   KEEPPRRT,Ø(R2)      IN THE TABLE
         BCR   8,PLNK              DON'T PRINT IF SO
         LA    R2,2(R2)            OTHERWISE POINT TO NEXT ENTRY AND
         BCT   R1,SCANRMTS          TRY AGAIN
         SPACE 1
SFLGØ4   MVI   KEEPSMBF,X'Ø4'      SET TO PRINT SMBS
         SPACE 1
         DROP  R3
         EJECT
**********************************************************************
*        THIS ROUTINE WRITES ALL OUTPUT                              *
**********************************************************************
         SPACE 1
```

```
SMBEND   DS    ØH
         CLI   KEEPSMBF,X'Ø4'      CREATE SMB
         BNE   Ø(PLNK)              NO,RETURN
         LR    RØ,R12              SAVE ADDR OF ISDRATES
         L     R12,ADDRLCT          AND RESTORE LCT ADDR
         USING LCTDSECT,R12
         MVC   LCTPARM1,MSGADDR    MOVE ADDRESS OF MSG TO LCT
         MVC   LCTPARM2+2(2),MSGLEN MOVE LENGTH OF MSG TO LCT
         STM   R14,R12,SAVE1+24    STORE ALL REGISTERS
         LR    R6,R13              SAVE POINTER TO SAVE AREA
         CALL  IEFYS,ID=1 GO WRITE MSG ( IEFYS - SYS1.AOSB3(IEFTB724)
         LR    R13,R6              RESTORE POINTER TO SAVE AREA
         LM    R14,R12,SAVE1+24    RESTORE ALL REGISTERS
         LR    R12,RØ              RESTORE ADDRESS OF ISDRATES
         MVI   BLANK1,C' '         BLANK
         MVC   BLANK2,BLANK1             OUTPUT AREA
         BR    PLNK                RETURN TO CALLER
         SPACE 1
         USING ISDRATES,R12        REESTABLISH ADDRESSABILITY TO 'RATES
         EJECT
*********************************************************************
*        CONVERT TIME AND DATE                                      *
*********************************************************************
         SPACE 1
CONVERT  SR    R14,R14             CLEAR CORRUPTIBLE REGISTER
         D     R14,CLAM1ØØ         TIME INTO WHOLE SECONDS
         CVD   R14,DOUBLE          HUNDREDTHS TO PACKED DECIMAL
         UNPK  DOUBLE(5),DOUBLE+6(3) THENCE TO EBCDIC
         MVC   CLAMWORK+9(2),DOUBLE+1  AND INTO CLAMWORK
         SPACE 1
         SR    R14,R14             CLEAR HIGH ORDER BYTES
         D     R14,CMRHOUR         COMPUTE HOURS
         CVD   R15,DOUBLE          HOURS TO PACKED DECIMAL
         UNPK  DOUBLE(5),DOUBLE+6(3) THENCE TO EBCDIC
         MVC   CLAMWORK(2),DOUBLE+1   AND INTO CLAMWORK
         SPACE 1
         LR    R15,R14             ALIGN DIVIDEND
         SR    R14,R14             CLEAR REMAINDER REGISTER
         D     R14,CMRMINIT        COMPUTE MINUTES
         CVD   R15,DOUBLE          MINUTES TO PACKED DECIMAL
         UNPK  DOUBLE(5),DOUBLE+6(3) THENCE TO EBCDIC
         MVC   CLAMWORK+3(2),DOUBLE+1   AND INTO CLAMWORK
         SPACE 1
         CVD   R14,DOUBLE          SECONDS TO PACKED DECIMAL
         UNPK  DOUBLE(5),DOUBLE+6(3) THENCE TO EBCDIC
         MVC   CLAMWORK+6(2),DOUBLE+1  AND INTO CLAMWORK
         SPACE 1
         MVI   CLAMWORK+8,C'.'     SET PERIOD
         MVI   CLAMWORK+2,C':'       AND
         MVI   CLAMWORK+5,C':'        COLONS IN CLAMWORK AREA
```

```
         SPACE 1
         BR    R8                  RETURN TO CALLER
         EJECT
*************************************************************************
*  COMPUTE: ( NUMBER OF DAYS JOB WAS ACTIVE ) * ( TWENTY-FOUR HOURS ) *
*************************************************************************
         SPACE 1
         USING SMF30ID,R4          SET ID SECTION ADDRESSABILITY
CMGILL   SR    RØ,RØ               ZERO TIME
         LR    R1,RØ                 ACCUMULATORS
         CLC   SMF30STD,TERMDATE   COMPARE START AND STOP DATES
         BE    CGDATEQ             BRANCH IF SAME DAY
         BL    CGDATELO            BRANCH IF SPANNED AT LEAST ONE DAY
         B     4(R8)               ERROR IF JOB STOPPED BEFORE IT BEGAN
         SPACE 1
CGDATELO CLC   SMF30STD(2),TERMDATE COMPARE YEAR PORTIONS ONLY
         BE    CGYEAREQ            BRANCH IF SAME YEAR
         BH    4(R8)               ERROR IF YEARS REVERSED
         SPACE 1
         ZAP   DOUBLE,SMF30STD     ISOLATE START YEAR
         SRP   DOUBLE,64-3,Ø         AS PACKED DECIMAL NUMBER
         CVB   R15,DOUBLE          CONVERT START YEAR TO BINARY,
         TM    SMF30STD+1,1        TEST FOR LEAP YEAR
         BO    CGPERRY             BRANCH IF IMPOSSIBLE
         TM    SMF30STD+1,X'12'    RETEST FOR CERTAINTY
         BM    CGPERRY             SKIP IF NOT A LEAP YEAR
         LA    R1,1                ALLOW FOR 366-DAY YEAR
         SPACE
CGPERRY  LA    R1,365(,R1)         ADD 365 DAYS FOR YEAR CHANGE
CGYEAREQ ZAP   DOUBLE,TERMDATE+2(2) COMPUTE DIFFERENCE
         SP    DOUBLE,SMF30STD+2(2)  BETWEEN JOB START
         CVB   R15,DOUBLE            DAY AND JOB STOP DAY
         AR    R1,R15              ADD TO ADJUSTMENT FOR YEAR CHANGE
         M     RØ,CMG24            CONVERT DAYS TO HUNDREDTHS OF SEC
         SPACE
CGDATEQ  AL    R1,TERMTIME         ADD STOP TIME
         BC    12,CGOCT147         SKIP IF NO CARRY
         AL    RØ,CMRF1            ADJUST HIGH END
         SPACE
CGOCT147 SL    R1,SMF30SIT         SUBTRACT START TIME
         BC    3,CGJUN83           SKIP IF CARRY
         BCTR  RØ,Ø                ADJUST HIGH END
         SPACE
CGJUN83  LTR   RØ,RØ               CHECK HIGH END
         BM    4(R8)               ERROR IF TOTAL TIME IS NEGATIVE
         BP    4(R8)               ERROR IF TOO LARGE FOR EDIT
         CL    R1,CMG25            CHECK LOW END
         BH    4(R8)               ERROR IF TOO LARGE FOR EDIT
         LR    R15,R1              DIFFERENCE TO CONVERSION REGISTER
         BR    R8                  RETURN TO CALLER
```

```
         EJECT
*************************************************************************
*                                                                       *
*        CONVERT JULIAN DATE TO GREGORIAN DATE                          *
*                                                                       *
*************************************************************************
         SPACE 1
CMRGREG  MVC   CLAMHOLD+2(8),PATWORK EDIT PATTERN TO OUTPUT AREA
         ST    R14,CLAMWORK+4      DATE TO CLAMWORK
         LA    R14,MONTHTAB-3      PREPARE TO SCAN CONVERSION TABLE
         TM    CLAMWORK+5,1        TEST FOR POSSIBILITY OF LEAP YEAR
         BO    CMREDIT             BRANCH IF IMPOSSIBLE
         TM    CLAMWORK+5,X'12'    TEST AGAIN FOR CERTAINTY
         BM    CMREDIT             BRANCH IF NOT
         LA    R14,CMRLEAP-3       PREPARE TO SCAN LEAP YEAR CONV TABLE
CMREDIT  ED    CLAMHOLD+7(3),CLAMWORK+5 SET UP YEAR
         MVC   CLAMHOLD+6(2),CMR2Ø ASSUME THAT THE MILLENNIUM IS HERE
         CLI   CLAMWORK+4,Ø        TEST ACCURACY OF THAT ASSUMPTION
         BNE   CMR21TH             BRANCH IF NAIL WAS HIT ON ITS HEAD
         MVC   CLAMHOLD+6(2),CMR19   ELSE IT'S STILL THE 2ØTH CENTURY
CMR21TH  MVI   CLAMHOLD+5,C'/'     DATE DELIMITER
         XC    CLAMWORK(6),CLAMWORK CLEAR ALL BUT JULIAN DATE
         SR    RØ,RØ               ZERO REGISTER Ø
         CVB   R1,CLAMWORK         CONVERT JULIAN DATE TO BINARY
CMRDATE  SR    R1,RØ               CONVERT FROM JULIAN DATE TO
         LA    R14,3(R14)           MONTH AND YEAR
         IC    RØ,Ø(R14)           FETCH DAYS IN A MONTH
         CR    RØ,R1               TEST IF INCOMPLETE MONTH
         BL    CMRDATE             BRANCH IF NOT
         CVD   R1,CLAMWORK         CONVERT TO DECIMAL.
         MVO   CLAMWORK(2),CLAMWORK+6(2) SHIFT FOR EDIT
         ED    CLAMHOLD+2(3),CLAMWORK SETUP DAY
         MVI   CLAMHOLD+2,C'/'     DATE DELIMITER
         MVC   CLAMHOLD(2),1(R14)  MONTH
         BR    R8                  RETURN TO REQUESTOR
         SPACE 3
         USING ISDRATES,RAT
         SPACE 1
VISDRATE DC    V(ISDRATES)         TO ESTABLISH ADDRESSABILITY
         EJECT
         LTORG
         TITLE 'ISDACTRT - JOB AND STEP CHARGE COMPUTATION ROUTINE'
ISDACTRT CSECT
         SPACE 3
*************************************************************************
*                                                                       *
* FUNCTION:    THIS ROUTINE DOES ALL THE COST CALCULATING FOR           *
*        THE IEFACTRT SMF ACCOUNTING EXIT.  TO DO THIS IT USES          *
*        ONLY INFORMATION PASSED TO IT BY IEFACTRT AND RETURNS          *
*        ALL VALUES IN THE ARGUMENT LIST PASSED TO IT.                  *
```

```
*                                                                     *
*        THE REGISTERS R9 - R13 ARE USED AS IN IEFACTRT AND REMAIN    *
*        UNCHANGED THROUGH THIS ROUTINE.                              *
*                                                                     *
***********************************************************************
         SPACE 2
***********************************************************************
*                                                                     *
*              B E G I N   M A I N   P R O G R A M                    *
*                                                                     *
***********************************************************************
         SPACE 1
         LR    PLNK,R15               SET UP
         USING ISDACTRT,PLNK            PLNK AS BASE
         SPACE 1
*        REITERATE USINGS FOR REGISTERS PASSED FROM IEFACTRT
         SPACE 1
         USING KEEPSECT,KEEP       AREA KEPT FOR DURATION OF JOB
         USING SMFRCD30,SMF        POINTER TO SMF RECORD
         USING ISDRATES,RAT        TABLE OF RATES AND CONSTANTS
         USING WORKAREA,WORK       WORK AREA FOR EACH ENTRY
         SPACE 1
*        PICK OUT CORRECT ENTRY AND GO THERE
         SPACE 2
         NOP   *
         LH    R2,2(R14)           GET CALL ID VALUE
         B     *-8(R2)             PICK OUT CORRECT ENTRY
         B     STEPCALC            STEP TERM IF ID=12
         B     JOBCALC             JOB TERM IF ID=16
         EJECT
STEPCALC EQU   *
***********************************************************************
*                                                                     *
*        ENTERED FOR STEP TERMINATION                                 *
*                                                                     *
***********************************************************************
         SPACE 1
         SPACE 1
*        CALCULATE CRU TIMES FOR EXCPS                                *
*        TIME IS IN UNITS OF 1/100 SECOND
         SPACE 1
         L     R15,KEEPEXCP        PICK UP TOTAL EXCPS
         L     R1,DISKEXCP         GET DISK EXCPS
         A     R1,TAPEEXCP         ADD TAPE EXCPS
*        A     R1,URECEXCP         ADD UNIT RECORD EXCPS
         AR    R15,R1              ADD TO TOTAL
         ST    R15,KEEPEXCP        SAVE TOTAL EXCPS
         SPACE 2
*        CALCULATE CRU FOR CPU TIME
         SPACE 1
```

```
         L     R15,KEEPCPU         PICK UP TOTAL CPU TIME(UNFACTORED)
         L     R1,CPUTIME          GET THE STEP CPU TIME
         AR    R15,R1              ADD TO PREVIOUS TOTAL
         ST    R15,KEEPCPU         SAVE TOTAL CPU (UNFACTORED)
         SPACE 1
         LA    R2,CALFACT          POINT TO LIST OF CPU FACTORS
         LA    R15,PATSIZE         SET ATTEMPT COUNT
         SPACE 1
FINDCPUA CLC   SMF3ØSID,Ø(R2)      TEST IF CPU IDS MATCH
         BE    CFACTOR             BRANCH IF SO
         LA    R2,8(R2)            POINT TO NEXT ENTRY
         BCT   R15,FINDCPUA         AND REPEAT LOCATE ATTEMPT
         B     SKPFACTR            PASS RAW DATA TO BILLING
         SPACE 1
CFACTOR  M     RØ,4(R2)            MULTIPLY BY FACTOR-1
         SLDA  RØ,8                PUT ALL IN RØ
         LR    R1,RØ               COPY
         SPACE 1
SKPFACTR ST    R1,CALFACPU         SAVE FACTORED CPU TIME
         EJECT
CALCOST  EQU   *
         SPACE 1
*********************************************************************
*                                                                   *
*        CALCULATE STEP COST OF CPU AND EXCP'S                      *
*                                                                   *
*********************************************************************
         SPACE 1
&SCOST   SETC  'NO'
         AIF   ('&SCOST' EQ 'NO').NSCOST
         SR    R15,R15              START WITH ZERO
         L     RØ,CALFACPU         GET CPU TIME (FACTORED)
         AR    R15,RØ              ADD TO SUM SO FAR
         A     RØ,=F'99'             ADD 99/1ØØ SEC
         SRDA  RØ,32                 FOR DIVIDE
         D     RØ,CLAM1ØØ            TO GET CPU TO NEAREST WHOLE SEC
         M     RØ,CPURATE          TIMES RATE
         ST    RØ,RETOCOST         SAVE CPU COST
         SPACE 1
         SR    RØ,RØ               CLEAR COST OF EXCPS
         ST    RØ,RETXCOST         SAVE EXCP COST
         SPACE 1
         LR    RØ,R15             TOTAL CRU TO RØ
         A     RØ,=F'99'             ADD 99/1ØØ SEC
         SRDA  RØ,32                 FOR DIVIDE
         D     RØ,CLAM1ØØ            TO GET CRU TO NEAREST WHOLE SEC
         M     RØ,CRURATE          TIMES CRU RATE
         ST    RØ,RETCOST          RETURN TOTAL CRU COST THIS STEP
         ST    R15,CALFACRU        SAVE TOTAL CRU 1/1ØØ SEC
.NSCOST  ANOP
```

```
         B     THATSALL           EXIT
         EJECT
JOBCALC  EQU   *
         SPACE 1
*************************************************************************
*                                                                       *
*        THIS ROUTINE IS ENTERED FOR JOB TERMINATION                    *
*                                                                       *
*************************************************************************
         SPACE 1
*
*        CALCULATE COSTS OF LINES AND CARDS INPUT AND PUNCHED
*        LINES AND CARDS ARE ROUNDED TO NEAREST 1000 IF LT 1000
*
         SR    R0,R0               CHARGE=0
         SPACE 1
         CLC   KEEPPRRT,=H'5'      REMOTE5
         BNE   NICOLE              NO
         STH   R0,KEEPPRRT         IF SO, TREAT AS LOCAL
NICOLE   CLC   KEEPPURT,=H'5'
         BNE   ANGELA
         STH   R0,KEEPPURT
ANGELA   CLC   KEEPINRT,=H'5'
         BNE   CAROL
         STH   R0,KEEPINRT
         SPACE 1
*        COMPUTE PRINT COST
         SPACE 1
CAROL    CH    R0,KEEPPRRT         REMOTE
         BNE   PCOST               YES, NO CHARGE
         L     R1,PRNTLNES         GET LINES GENERATED
         LTR   R1,R1               ARE THERE ANY
         BZ    PCOST               NO, FINISHED
         L     R0,BREAK            MINIMUM
         CR    R1,R0               TEST FOR LESS THAN MINIMUM
         BNL   PATRICIA            BRANCH IF MORE THAN MINIMUM
         LR    R1,R0               SET TO MINIMUM
PATRICIA M     R0,PRNTRATE         MULTIPLY BY RATE
         N     R1,MASK2            SAVE 2 PLACES
         LTR   R1,R1               ANY LEFT OVER
         BZ    PCOST               NO
         A     R0,CMRF1            YES, ADD A PENNY
PCOST    ST    R0,RETLCOST         RETURN COST
         EJECT
*        COMPUTE COST OF SPECIFIC AS WELL AS NON-SPECIFIC TAPE MOUNTS
         SPACE 1
         SR    R1,R1               CLEAR VOLATILE GPR
         ICM   R1,3,KEEPTPR        OBTAIN NUMBER OF SPECIFIC MOUNTS
         ST    R1,RETSCOST         RETURN COST OF SPECIFIC MOUNTS
         BE    CMRNONSP            BRANCH IF NONE
```

```
         M     RØ,TAPSRATE         COMPUTE COST OF TAPE MOUNTS
         ST    R1,RETSCOST         RETURN COST OF SPECIFIC MOUNTS
         SPACE 1
CMRNONSP SR    R1,R1               CLEAR VOLATILE GPR
         ICM   R1,3,KEEPPTM        OBTAIN NUMBER OF NON-SPECIFIC MOUNTS
         BE    CPCOST              BRANCH IF NONE
         M     RØ,TAPNRATE         COMPUTE COST OF TAPE MOUNTS
         EJECT
*        COMPUTE PUNCH COST
*        FOR CRYING OUT LOUD - IT'S THE END OF THE MILLENNIUM!
         SPACE 1
CPCOST   ST    R1,RETNCOST         RETURN COST OF NON-SPECIFIC MOUNTS
         SR    RØ,RØ               CHARGE=Ø
         CH    RØ,KEEPPURT         REMOTE
         BNE   CCOST               YES,NO CHARGE
         L     R1,PUNCHCRD         GET CARDS GENERATED
         LTR   R1,R1               ARE THERE ANY
         BZ    CCOST               NO,FINISHED
         L     RØ,BREAK            MINIMUM
         CR    R1,RØ               TEST FOR LESS THAN MINIMUM
         BNL   CJONLYN             BRANCH IF MORE THAN MINIMUM
         LR    R1,RØ               SET TO MINIMUM
CJONLYN  M     RØ,PNCHRATE         MULTIPLY BY RATE
         N     R1,MASK2            SAVE 2 PLACES
         LTR   R1,R1               ANY LEFT OVER
         BZ    CCOST               NO
         A     RØ,CMRF1            YES,ADD A PENNY
CCOST    ST    RØ,RETCCOST         RETURN COST
         SPACE 1
*        COMPUTE CHARGE FOR CARDS READ
         SPACE 1
         SR    RØ,RØ               CHARGE=Ø
         CH    RØ,KEEPINRT         REMOTE?
         BNE   ICOST               YES, NO CHARGE
         L     R4,SMF3ØIOF         OFFSET TO IDENTIFICATION SECTION
         AR    R4,SMF              ADDRESS OF IDENTIFICATION SECTION
         USING SMF3ØID,R4          SET IDENTIFICATION SECTION REFERENCE
         SPACE 1
         CLC   SMF3ØJBN(2),=C'JJ'  DOT JOB
         BE    ICOST               YES,NO CHARGE
         L     R1,CRDSREAD         GET CARDS READ
         LTR   R1,R1               ARE THERE ANY
         BZ    ICOST               NO,FINISHED
         L     RØ,BREAK            MINIMUM
         CR    R1,RØ               TEST FOR LESS THAN MINIMUM
         BNL   ILYNJON             BRANCH IF MORE THAN MINIMUM
         LR    R1,RØ               SET TO MINIMUM
ILYNJON  M     RØ,CARDRATE         MULTIPLY BY RATE
         N     R1,MASK2            SAVE 2 PLACES
         LTR   R1,R1               ANY LEFT OVER
```

```
         BZ    ICOST               NO
         A     RØ,CMRF1            YES,ADD A PENNY
         SPACE 1
ICOST    ST    RØ,RETICOST         RETURN COST
         EJECT
*        (RE)CALCULATE OS COST - CPU+EXCP
*        FINAL COSTS ARE COMPUTED FROM RAW SMF DATA.
*        KEEPCOST WHICH IS CARRIED ALONG WITH THE
*        JOB IS NOT USED IN THE FINAL CALCULATION.
*
         SPACE 2
*        DETERMINE WHICH RATE SCHEDULE TO USE
**********************           ASSUME OLD RATE,CLEAR FOR INDEX
********************************  TEST ACCOUNT NO
**********************           USE OLD RATE OF 7 CENTS
**********************           SET INDEX FOR NEW RATE
         SPACE
*        CONVERT EXCPS TO CRU; SAVE, ROUND TO SECONDS, COMPUTE COST
         SR    R15,R15             FOR TOTAL OF EACH TIME
         SPACE
         SR    RØ,RØ               CLEAR VOLATILE REGISTER
         L     R1,KEEPEXCP         PICK UP TOTAL EXCPS
         A     R1,=F'999'          ADD 999 EXCPS
         D     RØ,=F'1ØØØ'         COMPUTE MULTIPLES OF ONE THOUSAND
         SR    RØ,RØ               CLEAR VOLATILE REGISTER
         M     RØ,=F'1ØØØ'         COMPUTE EXCPS IN THOUSANDS FOR COST
         M     RØ,IORATE           TIMES RATE/1ØØØ EXCPS
         N     R1,MASK2            ERASE LOW ORDER THREE BYTES
         LTR   R1,R1               TEST IF RESIDUAL REMAINS
         BZ    JACQUELN            BRANCH IF NOT
         A     RØ,CMRF1             ELSE A PENNY
JACQUELN ST    RØ,RETXCOST           AND STOW IT
         SPACE 1
*        CONVERT CPU TO CRU,SAVE,ROUND TO SEC,COMPUTE COST
         SPACE 1
         L     RØ,KEEPCPU          PICK UP CPU
         LA    R2,CALFACT          POINT TO LIST OF CPU FACTORS
         LA    R1,PATSIZE          SET LOOP COUNT
FINDCPU  CLC   SMF3ØSID,Ø(R2)      TEST IF CPU IDS MATCH
         BE    CFACTOR1            BRANCH IF SO
         LA    R2,8(R2)            POINT TO NEXT ENTRY
         BCT   R1,FINDCPU           AND REPEAT LOCATE ATTEMPT
         B     NOFACT              SEND UNFACTORED DATA TO E W
         SPACE 1
CFACTOR1 DS    ØH
         LR    R1,RØ               FOR MULTIPLY
         M     RØ,4(R2)            X FACTOR
         SLDA  RØ,8                PUT ALL IN RØ
NOFACT   DS    ØH                  ADD TO CONVERTED EXCP
         ST    RØ,CALFACPU         SAVE TOTAL FACTORED CPU TIME
```

```
         AR    R15,RØ              ADD TO TOTAL CRU
         A     RØ,=F'99'           ADD 99/1ØØ SEC
         SRDA  RØ,32               FOR DIVIDE
         D     RØ,CLAM1ØØ          TO GET CRU TO NEAREST WHOLE SEC
         M     RØ,CPURATE          TIMES RATE  (INDEXED)
         SLDA  RØ,8                ALL TO RØ
         ST    RØ,RETOCOST         SAVE CPU TIME COST
         EJECT
*        ROUND CRU TO SEC,COMPUTE COST
         SPACE 1
         LR    RØ,R15              TOTAL CRU
         A     RØ,=F'99'           ADD 99/1ØØ SEC
         SRDA  RØ,32               FOR DIVIDE
         D     RØ,CLAM1ØØ          TO GET CRU TO NEAREST WHOLE SEC
         M     RØ,CRURATE          TIMES CRU RATE  (INDEXED)
         SLDA  RØ,8                ALL TØ RØ
         ST    RØ,RETCOST          SAVE TOTAL BILLABLE COST(CENTS)
         ST    R15,CALFACRU        SAVE TOTAL CRU 1/1ØØ SEC
         SPACE 2
THATSALL DS    ØH
         BR    14                  RETURN
         EJECT
ISDRATES CSECT
         SPACE 1
********************
*  DOCUMENTATION   *
********************
         SPACE 1
         TAPINFO
         SPACE 2
********************
*    RATES         *
********************
         SPACE
RATABLE  DS    ØF                  --- RATE TABLE -----
CRURATE  DC    FS24'21.Ø833'       OLD RATE FOR CRU 3.8889 CENTS/ SEC
         DC    FS24'21.Ø833'       NEW RATE FOR CRU 3.472 CENTS/SEC
CPURATE  EQU   CRURATE             RATE FOR CPU TIME IN CENTS/ SEC
IORATE   DC    FS32'.ØØ51'          RATE FOR I/O EXCP/1ØØØ
*
TAPSRATE DC    F'4Ø'      (CENTS)   $ Ø.5Ø/MOUNT OF SPECIFIC VOLUME
TAPNRATE DC    F'4Ø'      (CENTS)   $ Ø.5Ø/MOUNT OF NON-SPECIFIC VOLUME
CARDRATE DC    FS32'.Ø2Ø' (CENTS)   $ Ø.Ø2Ø/1ØØØ CARDS READ
PRNTRATE DC    FS32'.Ø6Ø' (CENTS)   $ Ø.6Ø/1ØØØ LINES PRINTED
PNCHRATE DC    FS32'.Ø6'  (CENTS)   $ Ø.Ø6/1ØØØ CARDS PUNCHED
BREAK    DC    F'1ØØØ'              UNIT RECORD BREAK
*                       ---- END RATE TABLE ------
         SPACE 2
**********************
*    CPU FACTORS     *
```

```
**********************
         SPACE
CALFACT  DS    ØF
         SPACE 1
         DC    CL4'VSØ1'           SMP ID OF 14ØØ DEV
         DC    FS24'1'             CRUNEW=CPU155 * 19.9
         SPACE 1
         DC    CL4'VSØ5'           SMP ID OF 14ØØ PROD
         DC    FS24'1'             CRU3Ø33=CPU155 * 19.9
         SPACE 1
         DC    CL4'VSØ4'           SMP ID OF 14ØØ ACCENT
         DC    FS24'.718'          CRU3Ø33=CPU155 * 19.9
         SPACE 1
         DC    CL4'VSØ3'           SMP ID OF 14ØØ TECH
         DC    FS24'1'             CRU47Ø=CPU155 * 19.9
         SPACE 1
         DC    CL4'VSØ2'           SMP ID OF NEXT MAINFRAME
         DC    FS24'19.9'          CRU158=CPU155 * 19.9
PATSIZE  EQU   (*-CALFACT)/8
         SPACE 2
MASK2    DC    ØF'Ø',X'FFØØØØØØ'   DEC-HEX CONV MASK
         SPACE 1
CLAM1ØØ  DC    F'1ØØ'
CMRHOUR  DC    AL4(6Ø*6Ø)
CMRMINIT DC    AL4(6Ø)
QMARKS   DC    CL4'????'
CMRUR    DC    CL4'U/R '
CMRDYN   DC    CL4'DYN '
CMRJJØT  DC    CL4'JJØT'
UNKNOWN  DC    CL8'UNKNOWN'
CMREND   DC    CL5'  END'
         EJECT
******************
*    FORMATS      *
******************
         SPACE 1
MGPRIO   MVC   HGPRIO(*-*),1(R4)   *** EXEC ONLY ***
MGACOUNT MVC   HGACOUNT(*-*),1(R4) *** EXEC ONLY ***
MGPD     MVC   HGPD(*-*),1(R4)     *** EXEC ONLY ***
         SPACE 1
CMG24    DC    A(24*6Ø*6Ø*1ØØ)
CMG25    DC    A(25*4*6Ø*6Ø*1ØØ)
         SPACE 1
CMR19    DC    C'19'
CMR2Ø    DC    C'2Ø'
CMRFØ    DC    F'Ø'
CMRF1    DC    F'1'
CMRF1Ø24 DC    F'1Ø24'
         SPACE 1
CMR7FFF  DC    XL4'ØØØ7FFF'
```

```
CMRFFFF  DC    XL4'0000FFFF'
CMRFFF   DC    XL4'00000FFF'
         SPACE 1
CMRH4    DC    H'4'
CMRH8    DC    H'8'
CMRH10   DC    H'10'
         SPACE 2
PATSTEP# DC    X'40202120'
PATPAGE  DC    X'4020206B202120'
PATEXCP  DC    X'402020206B2020206B202120'
PATBLKSZ DC    X'4020206B202120'
PATSERV  DC    X'4020206B2020206B202120'
PATDATE  DC    X'4020206120206120 20'
PATWORK  DC    X'F021204040402120'
         EJECT
MONTHTAB DC    AL1(31),C'01'
         DC    AL1(28),C'02'
         DC    AL1(31),C'03'
         DC    AL1(30),C'04'
         DC    AL1(31),C'05'
         DC    AL1(30),C'06'
         DC    AL1(31),C'07'
         DC    AL1(31),C'08'
         DC    AL1(30),C'09'
         DC    AL1(31),C'10'
         DC    AL1(30),C'11'
         DC    AL1(31),C'12'
         SPACE 1
CMRLEAP  DC    AL1(31),C'01'
         DC    AL1(29),C'02'
         DC    AL1(31),C'03'
         DC    AL1(30),C'04'
         DC    AL1(31),C'05'
         DC    AL1(30),C'06'
         DC    AL1(31),C'07'
         DC    AL1(31),C'08'
         DC    AL1(30),C'09'
         DC    AL1(31),C'10'
         DC    AL1(30),C'11'
         DC    AL1(31),C'12'
         EJECT
*        DEFINE TITLE AND HEADER FOR JOB STATISTICAL INFORMATION
         SPACE 1
CMRISSD  DC    CL78'  JOB ACCOUNTING INFORMATION  -  O I R  -  CENTRAL F
               FACILITY RESOURCES ONLY'
CGJOBNAM DC    CL8'JOB NAME'
CGJOBNO  DC    CL4'JOB#'
CGPD     DC    CL4'P/D'
CGSYSTEM DC    CL6'SYSTEM'
CGACOUNT DC    CL12'BILLING CODE'
CGPGMR   DC    CL23'PROGRAMMER''S NAME FIELD'
```

```
CJSSTART DC    CL2Ø'    JOB START    '
CJEND    DC    CL2Ø'      JOB END     '
CJELAPSD DC    CL18' JOB ELAPSED TIME '
CMRUNOME DC    CL11' (UNKNOWN)'
         SPACE 2
*        DEFINE STEP HEADER
         SPACE 1
CJOBNAME DC    CL8'  JOB'
CSTEPNAM DC    CL8'  STEP  '
CSTEPNUM DC    CL3'NUM'
CPGMNAME DC    CL8'PGM NAME'
CSTART   DC    CL18'    STEP START    '
CEND     DC    CL18'    STEP END     '
         SPACE 2
*        DEFINE TASK TIME HEADER
         SPACE 1
CELAPSED DC    CL13'ELAPSED TIME:'
CCPUTIME DC    CL15'CPU TIME: TCB ='
CSRBTIME DC    CL1Ø'     SRB ='
         SPACE 2
*        DEFINE TASK COMPLETION CODE HEADER
         SPACE 1
CCOMP    DC    CL21'STEP COMPLETION CODE:'
CABERC   DC    CL18'ABEND REASON CODE:'
CCCC     DC    CL3'CC='
CFLUSH   DC    CL7'FLUSHED'
         SPACE 2
*        DEFINE SERVICE UNITS HEADER
         SPACE 1
CSERVICE DC    CL2Ø'SERVICE UNITS: CPU ='
CSRBSERV DC    CL5'SRB ='
CIOSERV  DC    CL5'I/O ='
CMSOSERV DC    CL5'MSO ='
         EJECT
*        DEFINE PAGING HEADER
         SPACE 2
CPPI     DC    CL2'PI'
CPPO     DC    CL2'PO'
CPPR     DC    CL2'  '
CPVI     DC    CL2'VI'
CPVO     DC    CL2'VO'
CPVR     DC    CL2'VR'
         SPACE 1
*        DEFINE COMMON PAGING HEADER
         SPACE 2
CCCSAIN  DC    CL12'CSA: PAGE-IN'
CCRECLAM DC    CL8'HYPER-IN'
CLPAIN   DC    CL12'LPA: PAGE-IN'
CLRECLAM DC    CL8'HYPER-OT'
         SPACE 2
*        DEFINE SWAPPING HEADER
```

```
         SPACE 1
CSWAPING DC    CL19'SWAPPING: SEQUENCES'
CSWAPOUT DC    CL4'OUT'
CSWAPIN  DC    CL3'IN '
CPSTOLEN DC    CL13'PAGES STOLEN:'
         SPACE 2
*        DEFINE STORAGE ALLOCATION HEADERS
         SPACE 1
CSSIZE   DC    CL24'REGION(VIRT) SIZE      :'
CSMAX    DC    CL24'MAX STORAGE ALLOCATED   '
CSMAXC   DC    CL24'TO SYSTEM (LSQA+SWA)   :'
CSREQ    DC    C'REQUESTED'
CSUABOVE DC    CL25'USED ABOVE 16MEG/EXTENDED'
CSUBELOW DC    CL25'USED BELOW 16MEG        '
CSBELOW  DC    C'BELOW 16MEG'
CSABOVE  DC    CL25'ABOVE 16MEG/EXTENDED AREA'
CSREAL   DC    CL4'REAL'
         SPACE 2
*        DEFINE I/O SECTION HEADER
         SPACE 1
CIODDNAM DC    CL8'DDNAME'
CIOADDR  DC    CL4'ADDR'
CIOBLKSZ DC    CL6'BLKSIZ'
CIOUNIT  DC    CL4'UNIT'
CIOEXCP  DC    CL11'-- EXCPS --'
         EJECT
*        DEFINE TOTAL I/O HEADER
         SPACE 1
CTIODISK DC    CL12'DISK EXCPS ='
CTIOTAPE DC    CL12'TAPE EXCPS ='
CTIOJESV DC    CL11'JES + VIO ='
CMRVIO   DC    C'VIO '
CMRVIRTC DC    C'VIRT'
CMRJES2  DC    C'JES2'
CMRPSF   DC    C'PSF '
CMRPSFPG DC    C'PAGE'
CMRDUMY  DC    C'DMY '
CMRNA    DC    C'N/A'
CMRSAME  DC    C'- SAME -'
         SPACE 2
*        DEFINE TOTAL TAPE MOUNTS HEADER
         SPACE 1
CTTMOUNT DC    CL23'TAPE MOUNTS: SPECIFIC ='
CTTMN    DC    CL14'NON-SPECIFIC ='
CTTUSED  DC    CL16'TAPE UNITS USED:'
         EJECT
```

Editor's note: this article will be concluded in the next issue.

Enterprise print solutions

INTRODUCTION

Tremendous investments have been made in SNA applications (IBM host system resident applications designed to communicate over SNA networks) running on IBM mainframe and mid-range host systems. This is where the majority of the world's data resides and large enterprises in areas such as banking, finance, insurance, transport, manufacturing, and retail are depending on these applications for their day-to-day business. Although quite a few new applications are being developed on Windows NT and Unix platforms, the IBM host systems still remain the main servers for many of these companies.

The main vehicle for communicating with these IBM servers has been the use of separate SNA networks. With the incredible growth of TCP/IP-based intranets and the success of the Internet, companies often end up with two parallel networks, and network administrators are faced with increasing demands to consolidate corporate networks down to one single (TCP/IP) network to cut costs.

A lot has been said about technologies used to make SNA applications communicate over TCP/IP-based networks, often focusing on methods to connect a browser or TN3270/TN5250 emulator to the hosts or shipping entire SNA packets over IP networks. This article provides a technical overview of the most common methods used for enterprise SNA applications printing over TCP/IP networks.

WHAT'S THE PROBLEM?

SNA and TCP/IP stem from different backgrounds with different requirements. SNA has been developed, fine-tuned, and proven in mission-critical business networks with tough demands on uptime, security, availability, response times, reliability, and control. TCP/IP was originally designed to provide flexible, open, any-to-any communication services for universities.

SNA and TCP/IP are based on different communication foundations. SNA applications are built assuming that an SNA network is available. Regular off-the-shelf IP routers are not able to route SNA the way IBM mainframes and FEPs (Front End Processors) are. Various SNA-over-IP encapsulation methods have been proposed. They are available but come with additional costs. Regular LAN attached printers used in Windows/Unix environments do not speak SNA.

SNA and TCP/IP use different printer command languages. SCS (SNA Character Stream) and IPDS (Intelligent Printer Data Stream) emulations are not available in standard PC printers.

SNA and TCP/IP use different character sets for encoding text data. The native mode of encoding text data on IBM host systems is EBCDIC. Regular desktop printers use ASCII.

REQUIREMENTS AND CHALLENGES

In general, the requirements boil down to being able to deliver SNA applications' print data over IP networks to local and remote printers. The demands on speed, reliability, efficiency, and control naturally vary depending on the nature of the print data. A company with an occasional need to print out e-mail hard copies cannot be compared with a bank or insurance company printing large volumes of cheques and/or invoices.

A number of different technologies have surfaced through the years, fulfilling the demands above to different levels, from simply forwarding entire SNA packets encapsulated in TCP/IP to schemes that terminate the SNA session on the host system, convert the data to ASCII, and use standard TCP/IP print methods. All have their own advantages and drawbacks.

Although a lot of effort has been spent and continues to be spent on improvements, there is still no method available that will fully match all aspects of SNA. Keep in mind that SNA enjoys the advantage of having been fine-tuned for the last three decades in this environment. However, lately a set of more sophisticated, cost-effective, and very promising methods such as TN3270E and TN5250E have been

introduced. Both built on native TCP/IP but with SNA applications, display and printer data flows in mind, bringing them close enough to real end-to-end SNA functionality for most users.

TODAY'S MENU

Looking at the solutions available today, we are able to identify a number of alternatives at different networking layer levels. First of all, a couple of methods that carry SNA packets complete with THs (Transmission Headers) and RHs (Request Headers) more or less untouched over the WAN (Wide Area Network) deserve to be mentioned:

- Frame Relay (RFC 1490)
- MPTN (Multi Protocol Transport Networking)
- DLSw (Data Link Switching).

These methods do require a full SNA stack at the client. They have been discussed in numerous articles and books and will not be covered in this article. A good reference is *Re-engineering IBM Networks* (A Gurugé, 1996).

Instead, we will focus on native TCP/IP print methods used with SNA applications printing. The most popular ones in use today may be summarized as:

- LPR/LPD
- Reverse Telnet
- PPR/PPD
- TN3270E
- TN5250E.

LPR/LPD

LPR/LPD (Line Printer Requester/Line Printer Daemon) is a TCP/IP-based print method that stems from the Unix world. The official

specification used today is RFC 1179. Print data, which is normally ASCII encoded, is sent from the LPR to the LPD print server. In the IBM host system arena this method often relies on host-resident software translating the SCS/IPDS EBCDIC encoded print data to ASCII. Once the conversion is done, standard LPR/LPD is used to carry the data to the LPD/printer.

The good thing with LPR/LPD is that it is a widely used and available technology that may be applied over any IP network. However, when used with SNA applications printing, the list of drawbacks is quite extensive. It lacks print job acknowledgment. Once the print data is transmitted to the LPD, the host will consider the job done. This does not necessarily mean that the job is actually printed. Another disadvantage of this method is the ageing and not very detailed or concise specification, leaving the door open for incompatible implementations of the protocol. Some implementations will not allow the user to select page ranges or multiple copies of a document to be printed. Other implementations may require PCL emulation in the target ASCII printer.

Still, LPR/LPD is a popular method for printing in Unix as well as IBM environments. It is available on all major platforms and although there are a number of drawbacks in the IBM environment it is an inexpensive and widely-available method. It may very well be an appropriate choice for the occasional user.

Pros	*Cons*
Open standard	Requires host processor cycles and resources translating to ASCII
TCP/IP based	No control /feedback of print jobs
Simple	Uni-directional
Inexpensive	Vague specification
Regular TCP/IP routers may be used	No error recovery
Widely available and used	No printer device description on AS/400
Integral part of OS/400	
Small footprint	

Figure 1: Pros and cons of LPR/LPD

A couple of vendors do offer EBCDIC to ASCII conversion software and/or LPR implementations for the mainframe environment. The list of available products includes: Network Print Facility (IBM), VPS (LRS), and TCPaccess (Interlink). Looking at the AS/400 world, the HPT (Host Print Transform) EBCDIC to ASCII conversion utility and LPR/LPD are standard features of current OS/400 levels. The pros and cons of LPR/LPD are summarized in Figure 1.

REVERSE TELNET

Reverse Telnet is another protocol with roots in the Unix world. It is a straightforward method that is based on simply transferring data safely to/from TCP ports that is now being used for printing purposes. This approach, sometimes called 'raw TCP/IP' or 'direct sockets printing', eliminates some of the shortcomings of LPR/LPD. This is achieved by the driver implementation in the IBM host system taking advantage of the bi-directional status reporting capabilities of PJL/PCL printers.

Pros	*Cons*
Open standard	Requires host processor cycles and resources translating to ASCII
TCP/IP based	Requires PJL/PCL-capable laser printers
Simple	Limited control and error recovery
Inexpensive	Semi bi-directional
Regular TCP/IP routers may be used	
Integral part of OS/400	
Small footprint	

Figure 2: Pros and cons of Reverse Telnet

This TCP-based method was introduced in the AS/400 world with OS/400 Version 3.7. It is available in the mainframe environment as well. One example of a product offering it is the popular VPS (VTAM Print Support) system from Levi, Ray, and Shoup.

Although Reverse Telnet delivers some advantages compared with LPR/LPD, it still suffers from not being built with SNA applications

printing in mind. The pros and cons of Reverse Telnet are shown in Figure 2.

PPR/PPD

Starting with PSF/MVS Version 2.2 and OS/400 Version 3.1, IBM introduced a new TCP/IP-based print method designed for IPDS (Intelligent Printer Data Stream) printing called PPR/PPD (Page Printer Requester/Page Printer Daemon). Although a limited set of commands and replies is specified, this method features built-in bi-directional capabilities. Combined with the strong status reporting mechanisms of IPDS, good run-time control and monitoring of print jobs is provided. As with the LPR/LPD structure, the requester (PPR) application sends print data to the server (PPD).

Enjoying extensive support from IBM in both host system drivers as well as the actual printers/print servers, this method has become a *de facto* industry standard for IPDS over native TCP/IP. It is available on all IBM strategic platforms including MVS and OS/400 and also optionally available for most later IBM network printer models as well as from third-party printer and print server vendors.

Being based on native TCP/IP, supported by existing printers/print servers, delivering SNA-like control of print jobs, and not requiring host resources for translating print data to ASCII, this method has given IBM's page printer language IPDS a renaissance in the last few years.

The major drawback of this method is the fact that it is proprietary. The specifications are not publicly available, leaving the door open for

Pros	*Cons*
TCP/IP based	Proprietary
Bi-directional	IPDS only
De facto industry standard for IPDS over TCP/IP	
Both mainframe and AS/400 environments	
Small footprint	
Regular TCP/IP routers may be used	

Figure 3: Pros and cons of PPR/PPD

incompatible implementations. Another disadvantage is that it is used with IPDS only, leaving a large number of existing SNA applications out in the cold. The pros and cons of PPR/PPD are summarized in Figure 3.

TN3270E

TN3270 is a TCP/IP Telnet-based protocol used to carry SNA RU (Request Unit) data untouched between IBM host systems and TN3270 clients over IP networks. While the original specifications from the mid-80s targeted display traffic, the TN3270E standard (RFC 1647 in 1994) featured a number of improvements, including support for printing.

The TN3270E data stream is created by a TN3270E server that basically replaces the SNA THs (Transmission Headers) and RHs (Request Headers) with TN3270E headers and ships the RU (Request Unit) data using TCP. The server may be implemented as a software package running on the mainframe itself, a router, or other server hardware. Numerous product offerings are NT- or Unix-based. The result is a highly efficient block-oriented protocol built with SNA

Pros	*Cons*
Open standard	5250 formatting features not available in AS/400 environments
TCP/IP based	No support for LU 6.2 data streams
Bi-directional	
Print job acknowledgments	
Regular TCP/IP routers may be used	
Small/moderate footprint	
Many client and server vendors	
IPDS support	
No host resident EBCDIC–ASCII translation	
Efficient/fast	

Figure 4: Pros and cons of TN3270E

applications' display and printer data flows in mind that comes close enough to real end-to-end SNA functionality for most users. It is bi-directional by nature and does provide the possibility of transmitting positive and negative numbered packet acknowledgments – making SNA-like control and management of print jobs possible. Both IPDS and non-IPDS printing is supported.

A wide range of TN3270E solutions are available from Apertus Technologies, Attachmate, Axis Communications, Bay Networks, Bus-Tech, Cisco, Data Interface Systems, Eicon Technology, Hummingbird, IBM, Interlink, Microsoft, Novell, OpenConnect Systems, etc.

TN3270E is a popular technology today and is expected to grow rapidly in the next few years. Leading market research institutes estimate a doubling from today's 8-10 million users in just two to three years. A summary of the pros and cons of TN3270E is shown in Figure 4.

TN5250E

With the introduction of OS/400 Version 4.2 in early 1998, IBM introduced a number of enhancements to the OS/400 Telnet implementation. This TN5250 extension sports a number of new features including support for SNA like SCS (SNA Character Stream) printing over native TCP/IP. Two new 'terminal types' may be negotiated for printing purposes:

- IBM-3812-1 for SBCS (Single Byte Character Set)
- IBM-5553-B01 for DBCS (Double Byte Character Set).

The 5553 type makes it possible to print Japanese, Korean, and traditional and simplified Chinese character set-based reports.

TN5250E shares all the major advantages with TN3270E and additionally offers automatic configuration of printer devices. Although the basic specifications look really promising, a few limitations of the current implementation can be identified, one being the limited set of printer types that may be negotiated. As no matrix printers are in the list of supported devices, users may run into problems with reports

specifically designed for such printers. Another drawback is that IPDS printing is only possible by using HPT (Host Print Transform) on the host system converting IPDS print data to ASCII.

An indication of IBM's commitment to TN5250E is the fact that upgrades/PTFs for OS/400 Version 3.2 and later will be made available, enabling this technology to a majority of the existing AS/400 installations. Also, the specifications originally driven by IBM have been submitted to IETF. Another sign of the interest in this method is

Pros	*Cons*
Open standard	AS/400 environment only
TCP/IP based	IBM 3812-1 and 5553-B01 only
Bi-directional	Matrix printers may not be negotiated
Print job acknowledgments	No IPDS unless HPT is used
Regular TCP/IP routers may be used	Not widely used (yet)
DBCS supported	
Efficient/fast	
Small/Moderate footprint	
No host EBCDIC-ASCII translation	
Automatic configuration	
User specified device names	
Integral part of OS/400	

Figure 5: Pros and cons of TN5250E

the attention paid by several client vendors. After being available for only a few months, a number of TN5250E clients for display and/or printing have already been announced, including: AXIS 570/670 (External Print Server), BOS (PC SW), Hummingbird (PC SW), IBM Client Access (PC SW), IBM PCOMM (PC SW), Wall Data Rumba (PC SW). A summary of the pros and cons of TN5250E is shown in Figure 5.

Inge Persson
Product Manager
Axis Communications (Sweden)

A mailbox system for SMTP under MVS TCP/IP – 7

This issue we continue the code for the implementation of a mailbox system for SMTP, based on ISPF functions.

CLIST ZCC (edit macro) to construct CC-receiver(s) in current mail:

```
 /*                                                                   */
 /* ZCC
 /* Edit macro to enter cc-receiver in current edit-dataset to SMTP;
 /* CC-address data is build.
 /*  Called from MAILDIRS & ZSMTPN or directly as Edit Macro.
 /*
 /*  Parameters (keyword):
 /*   CALLER : identification of calling function if called as
 /*            subroutine.
 /*  Parameters (optional) as macro parameters:
 /*   MEMOCC : If indicated will be used as Mail CC-receiver.
 /*   CCDM   : If indicated will be used as Mail CC-receiver's Domain.
 /*
 /*   SUBROUTINES/EDIT MACROES:
 /*    %COMPLIB
 /*    %MAILDIRS
 /*    %MAILSENS
 /*   PANEL USED     : INST81B
 /*
 /*   Utilities used:
 /*    SLEEP
 /*    TSOLINE1
 /*                                                                   */
PROC Ø CALLER(DUMMY) DEBUG(nEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ATTN DO
 SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

END
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 WRITE ======> Entering &SYSICMD <======
END
IF &FLUSH = FLUSH THEN DO
 SET &ZEDSMSG = &str(Function interrupted)
 ISPEXEC SETMSG MSG(ISRZØØ1)
```

```
 EXIT CODE(Ø)
END
SET &RET = Ø
SET &SDEBUG = &DEBUG
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE ====> Sorry only executable under ISPF (&SYSICMD).
 EXIT CODE(16)
END
IF &SYSNEST = NO THEN DO
 ISREDIT MACRO (MEMOCC,CCDM) PROCESS
END
ELSE DO
 ISPEXEC VGET (MEMOCC,CCDM)
END
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC VGET (ZSCREEN)
ISREDIT (SAVE) = USER_STATE
ISREDIT (CHANGED) = DATA_CHANGED
ISREDIT (MEMBER) = MEMBER
ISREDIT (DSNAME) = DATASET
SET &ATSIGN = &STR(@)
SET &NETDLM = &STR(%)
SET &DS = &STR(&DSNAME)
IF &MEMBER NE &STR() THEN DO
 SET &DS = &STR(&DSNAME(&MEMBER))
END
IF &MAXCC = Ø AND &STR(&MEMBER) NE &STR() THEN DO
 IF &CHANGED = YES THEN DO
  SET &SMAXCC = &MAXCC
  SET &RET = Ø
  ISPEXEC LMINIT DATAID(DID) DATASET('&STR(&DSNAME)') ENQ(SHRW)
  ISPEXEC LMOPEN DATAID(&DID) OPTION(INPUT)
  SET &LMRET = &RET
  IF &LMRET = Ø THEN DO
   SET &RET = Ø
   ISPEXEC LMMLIST DATAID(&DID) STATS(YES) MEMBER(MEMBER) OPTION(LIST)
   SET &RLVERS = &STR(&ZLVERS)
   SET &RLMOD = &STR(&ZLMOD)
   SET &RLMDATE = &STR(&ZLMDATE)
   SET &RLMTIME = &STR(&ZLMTIME)
   SET &RLUSER = &STR(&ZLUSER)
  END
  SET &LMRET = &RET
  IF &LMRET > 8 THEN DO        /* ACCEPT MAX RC8 FROM LMMLIST         */
   SLEEP 1
   WRITE &STR(===>) Error &LMRET LMMLIST (&SYSICMD).
  END
  SET &MAXCC = &SMAXCC
  SET &RET = Ø
  ISREDIT SAVE
```

```
  SET &SAVERET = &RET
  IF &RET = 4 THEN DO        /* NEW MEMBER */
   SET &RET = Ø
   SET &SAVERET = Ø
   SET &MAXCC = Ø
  END
  IF &SAVERET = 2Ø THEN DO
   %COMPLIB &STR('&DSNAME') SHR
   SET &MAXCC = Ø
   IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
    WRITE ======> Reentering &SYSICMD <======
   END
   /* TRY TO SAVE ONCE MORE                                          */
   ISREDIT SAVE
  END
  SET &SMAXCC = &MAXCC
  IF &STR(&RLUSER) = RECEIVE OR &STR(&RLUSER) = MAILED OR +
  &STR(&RLUSER) = REPLY OR &STR(&RLUSER) = SAVED THEN DO
    /* SAVE OF STATISTICS DURING EDIT WORKS ONLY ON ISPF V.4          */
   ISPEXEC LMMSTATS DATAID(&DID) MEMBER(&NRSTR(&MEMBER)) +
   VERSION(&RLVERS) MODDATE(&STR(&RLMDATE)) +
   MODTIME(&STR(&RLMTIME)) USER(&RLUSER)
  END
  ISPEXEC LMMLIST DATAID(&DID) OPTION(FREE)
  ISPEXEC LMCLOSE DATAID(&DID)
  ISPEXEC LMFREE  DATAID(&DID)
  SET &MAXCC = &SMAXCC
 END
END
IF &MAXCC = Ø AND &STR(&MEMBER) = &STR() THEN DO
 IF &CHANGED = YES THEN DO
  SET &RET = Ø
  ISREDIT SAVE
 END
END
IF &MAXCC = Ø THEN DO
 IF &STR(&SYSNSUB(1,&MEMOCC)) = &STR() OR +
 &STR(&SYSNSUB(1,&CCDM)) = &STR() THEN DO
  ISPEXEC VPUT (MEMOCC,CCDM)
  ISPEXEC DISPLAY PANEL(USER81B)
  IF &STR(&DEBUG) = &STR() THEN DO
   SET &DEBUG = &SDEBUG
  END
  ISPEXEC VGET (MEMOCC,CCDM)
  ISPEXEC VGET (RESP)
  IF &RESP = CANCEL THEN DO
   SET &RESP = &STR()
   ISPEXEC VPUT (RESP)
   ISPEXEC CONTROL DISPLAY LINE START(14)
   SLEEP 1
```

```
  WRITE ====> CC-address cancelled.
  ISREDIT USER_STATE = (SAVE)
  SET &RET = Ø
  SYSCALL FREERSC DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
 END
END
IF &STR(&SYSNSUB(1,&MEMOCC)) = &STR(?) AND &STR(&SYSISPF) = ACTIVE +
AND &SYSENV = FORE THEN DO
 SET &RET = Ø
 %MAILDIRS CALLER(&SYSICMD) DEBUG(&STR(&DEBUG)) /* SEARCH DIRECTORY */
 SET &DIRSRET = &RET
 IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
  WRITE ======> Reentering &SYSICMD <======
 END
 IF &DIRSRET = Ø THEN DO
  ISPEXEC VGET (DIRRC,DIRDM)
  /*                                                                */
  SET &STRING = &STR(&SYSNSUB(1,&DIRRC))
  SET &RSTRING = &STR()     /* CLEAR FOR CALL TO SUBROUTINE RETURN */
  SET &RET = Ø
  SYSCALL VARSTRNG RSTRING STRING(&STR('&SYSNSUB(1,&STRING)')) +
  DEBUG(&DEBUG)
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  SET &MEMOCC = &STR(&SYSNSUB(1,&RSTRING))
  /*                                                                */
  SET &STRING = &STR(&SYSNSUB(1,&DIRDM))
  SET &RSTRING = &STR()     /* CLEAR FOR CALL TO SUBROUTINE RETURN */
  SET &RET = Ø
  SYSCALL VARSTRNG RSTRING STRING(&STR('&SYSNSUB(1,&STRING)')) +
  DEBUG(&DEBUG)
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  SET &CCDM = &STR(&SYSNSUB(1,&RSTRING))
  /*                                                                */
  SET &DIRRC = &STR()
  SET &DIRDM = &STR()
  ISPEXEC VPUT (DIRRC,DIRDM)
 END
 IF &STR(&SYSNSUB(1,&MEMOCC)) = &STR() +
 THEN DO /* exit from recursive call */
  ISPEXEC CONTROL DISPLAY LINE START(14)
  ISREDIT USER_STATE = (SAVE)
  SET &RET = Ø
  SYSCALL FREERSC DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
  SET &RET = &LASTCC
  EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
```

```
 END
END
IF &STR(&SYSNSUB(1,&MEMOCC)) = &STR(?) THEN DO
 ISPEXEC CONTROL DISPLAY LINE START(14)
 SLEEP 1
 WRITE ====> No Receiver selected from directory, no CC-address built +
 (&SYSICMD).
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
 SET &RET = &LASTCC
 EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
END
ISREDIT CURSOR = 1 Ø
SET &SUBJTX = &STR(Subject:)
SET &LSUBJTX = &LENGTH(&STR(SUBJECT:))
SET &MAXCC = Ø
SET &SYNTAX = &STR()
SET &RET = Ø
ISREDIT SEEK '&SUBJTX' 1
IF &RET = Ø THEN DO
 ISREDIT (SUBJR,SUBJC) = CURSOR
 SET &SMAXCC = &MAXCC
 SET &RET = Ø
 ISREDIT LABEL &SUBJR = .SUBJ 1
 IF &RET > 8 THEN DO     /* allow replace of label */
  IF &RET > &SMAXCC THEN DO
   SET &SMAXCC = &RET
  END
 END
 SET &MAXCC = &SMAXCC
 SET &LASTLBL = &STR(.SUBJ)
 ISREDIT (NAMBEGIN,NAMFROW) = CURSOR
 ISREDIT (NAMFREC) = LINE &STR(&SYSNSUB(1,&NAMBEGIN))
 SET &LENNAM = &LENGTH(&STR(&SYSNSUB(1,&NAMFREC)))
 SET &SYSDVAL = +
 &SUBSTR(&NAMFROW+&LSUBJTX:&LENNAM,&STR(&SYSNSUB(1,&NAMFREC)))
 SET &SYSDVAL = &STR(&SYSNSUB(1,&SYSDVAL))
 READDVAL &A1 &A2 &A3 &A4 &A5 &A6 &A7 &A8 &A9
 SET &SU = &STR(&SYSNSUB(1,&A1) &SYSNSUB(1,&A2) &SYSNSUB(1,&A3) +
 &SYSNSUB(1,&A4) &SYSNSUB(1,&A5) &SYSNSUB(1,&A6) &SYSNSUB(1,&A7) +
 &SYSNSUB(1,&A8) &SYSNSUB(1,&A9))
END
ELSE DO
 ISPEXEC CONTROL DISPLAY LINE START(14)
 SLEEP 1
 WRITE ====> &SUBJTX    missing, no CC-address built (&SYSICMD).
 SLEEP 1
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
```

```
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
 SET &RET = &LASTCC
 EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
END
%MAILSENS RC('&STR(&SYSNSUB(1,&MEMOCC))') +
DOMAIN('&STR(&SYSNSUB(1,&CCDM))') +
ID('&STR(&SYSNSUB(1,&SU))') CCREQ(YES) +
DS(&STR(&DS)) DEBUG(&STR(&DEBUG))
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 WRITE ======> Reentering &SYSICMD <======
END
ISPEXEC VGET (DOMAIN)
ISREDIT CURSOR = 1 Ø
SET &RET = Ø
ISREDIT SEEK 'Rcpt to:<' 1 .ZF &STR(&LASTLBL)
IF &RET NE Ø THEN DO
 ISPEXEC CONTROL DISPLAY LINE START(14)
 SLEEP 1
 WRITE ====> Rcpt to:    missing, no CC-address built (&SYSICMD).
 SLEEP 1
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
 SET &RET = &LASTCC
 EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
END
ELSE DO
 ISREDIT (RCPTTOR,RCPTTOC) = CURSOR
 SET &SMAXCC = &MAXCC
 SET &RET = Ø
 ISREDIT LABEL &RCPTTOR = .B 1
 IF &RET > 8 THEN DO     /* allow replace of label */
  IF &RET > &SMAXCC THEN DO
   SET &SMAXCC = &RET
  END
 END
 SET &MAXCC = &SMAXCC
 IF &STR(&CALLER) NE &STR(ZSMTPN) THEN DO
  SET &SMAXCC = &MAXCC
  SET &MAXRP = 256
  ISPEXEC VPUT (MAXRP)      /* Info to inline subroutine */
  SET &RV = RV
  SET &RP = Ø
  DO WHILE &RP < &MAXRP
   SET &RP = &RP + 1
   ISPEXEC VGET (RV&RP)
   SET &C = &STR(&SYSNSUB(2,&&RV&RP))
   IF &STR(&SYSNSUB(1,&C)) NE &STR() THEN DO
    SET &&RV&RP = &STR()     /* clear variables */
```

```
  ISPEXEC VPUT (RV&RP)
 END
 ELSE DO
  SET &RP = &MAXRP
 END
END
SET &RA = .ZF
SET &RR = &STR(&LASTLBL)
/*                                                                  */
SET &SEARCH = &STR(To:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &TOCC = &RET
SET &RP = Ø
DO WHILE &RP < &MAXRP
 SET &RP = &RP + 1
 ISPEXEC VGET (RV&RP)
 SET &C = &STR(&SYSNSUB(2,&&RV&RP))
 IF &SYSLC(&STR(&SYSNSUB(1,&C))) = &STR() THEN DO
  SET &RP = &MAXRP
 END
 ELSE DO
  IF &SYSLC(&STR(&SYSNSUB(1,&C))) = +
  &SYSLC(&SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN)) THEN DO
   SET &RP = &MAXRP
   ISPEXEC CONTROL DISPLAY LINE START(14)
   SLEEP 1
   WRITE ====> Cc: &SYSLC(&SYSNSUB(1,&MEMOCC)&ATSIGN+
   &SYSNSUB(1,&DOMAIN)) not inserted, duplicate of To (&SYSICMD).
   SLEEP 1
   ISREDIT USER_STATE = (SAVE)
   SET &RET = Ø
   SYSCALL FREERSC DEBUG(&STR(&DEBUG))
   /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
   SET &RET = &LASTCC
   EXIT CODE(Ø)
  END
 END
END
 /*                                                                 */
SET &SEARCH = &STR(Cc:)
SET &RET = Ø
SYSCALL SCANFR &SEARCH &RA &RR &NETDLM &ATSIGN DEBUG(&STR(&DEBUG))
  /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
SET &RET = &LASTCC
SET &CCCC = &RET
SET &RP = Ø
DO WHILE &RP < &MAXRP
 SET &RP = &RP + 1
```

```
    ISPEXEC VGET (RV&RP)
    SET &C = &STR(&SYSNSUB(2,&&RV&RP))
    IF &SYSLC(&STR(&SYSNSUB(1,&C))) = &STR() THEN DO
     SET &RP = &MAXRP
    END
    ELSE DO
     IF &SYSLC(&STR(&SYSNSUB(1,&C))) = +
     &SYSLC(&SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN)) THEN DO
      SET &RP = &MAXRP
      ISPEXEC CONTROL DISPLAY LINE START(14)
      SLEEP 1
      WRITE ====> Cc: &SYSLC(&SYSNSUB(1,&MEMOCC)&ATSIGN+
      &SYSNSUB(1,&DOMAIN)) not inserted, duplicate of CC (&SYSICMD).
      SLEEP 1
      ISREDIT USER_STATE = (SAVE)
      SET &RET = Ø
      SYSCALL FREERSC DEBUG(&STR(&DEBUG))
      /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
      SET &RET = &LASTCC
      EXIT CODE(Ø)
     END
    END
   END
    /*                                                                */
   SET &MAXCC = &SMAXCC
  END
  SET &TRRET = Ø
  SET &RET = Ø
  ISREDIT LINE_AFTER .B = +
  "&syslc(Rcpt to:<&SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN)>)"
  SET &TRRET = &RET
  IF &TRRET > 4 THEN DO
   SET &RET = Ø
  ISREDIT LINE_AFTER .B = +
  '&syslc(Rcpt to:<&SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN)>)'
   SET &TRRET = &RET
  END
  IF &TRRET > 4 THEN DO
   ISPEXEC CONTROL DISPLAY LINE START(14)
   SLEEP 1
   WRITE ====> Rcpt to:    not inserted, no CC-address built (&SYSICMD).
   SLEEP 1
   ISREDIT USER_STATE = (SAVE)
   SET &RET = Ø
   SYSCALL FREERSC DEBUG(&STR(&DEBUG))
   /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
   SET &RET = &LASTCC
   EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
  END
 END
 ISREDIT CURSOR = 1 Ø
```

```
SET &RET = Ø
ISREDIT SEEK 'TO: ' 1 .ZF &STR(&LASTLBL)
IF &RET NE Ø THEN DO
 ISPEXEC CONTROL DISPLAY LINE START(14)
 SLEEP 1
 WRITE ====> To:         missing, no CC-address built (&SYSICMD).
 SLEEP 1
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
 SET &RET = &LASTCC
 EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
END
ELSE DO
 ISREDIT (CCPTTOR,CCPTTOC) = CURSOR
 SET &SMAXCC = &MAXCC
 SET &RET = Ø
 ISREDIT LABEL &CCPTTOR = .C 1
 IF &RET > 8 THEN DO     /* allow replace of label */
  IF &RET > &SMAXCC THEN DO
   SET &SMAXCC = &RET
  END
 END
 SET &MAXCC = &SMAXCC
 SET &TRRET = Ø
 SET &RET = Ø
 ISREDIT LINE_AFTER .C = +
 "C&syslc(c:       &SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN))"
 SET &TRRET = &RET
 IF &TRRET > 4 THEN DO
  SET &RET = Ø
 ISREDIT LINE_AFTER .C = +
 'C&syslc(c:       &SYSNSUB(1,&MEMOCC)&ATSIGN&SYSNSUB(1,&DOMAIN))'
  SET &TRRET = &RET
 END
 IF &TRRET > 4 THEN DO
  ISPEXEC CONTROL DISPLAY LINE START(14)
  SLEEP 1
  WRITE ====> Cc:         not inserted, no CC-address built (&SYSICMD).
  SLEEP 1
 END
 ELSE DO
  IF &SYSNEST = NO THEN DO
   TSOLINE1                /* CLEAR SCREEN */
  END
 END
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
```

```
  SET &RET = &LASTCC
  EXIT CODE(Ø)   /* keep cursor inline for insert, dont set cc1 */
 END
END
ELSE DO
 SET &ZEDLMSG = +
 &STR(=========> Execution could not be performed <==========)
 ISPEXEC SETMSG MSG(ISRZØØ1)
 ISREDIT USER_STATE = (SAVE)
 SET &RET = Ø
 SYSCALL FREERSC DEBUG(&STR(&DEBUG))
 /* SYSCALL RETURN CODE DOES NOT TAKE ERROR ROUTINE */
 SET &RET = &LASTCC
 EXIT CODE(Ø)   /* keep cursor inline for insert, don't set CC1 */
END
 /*                                                                  */
 /*  INLINE SUBROUTINES                                              */
 /*                                                                  */
VARSTRNG: +
PROC 1 RSTRING STRING() DEBUG(NEBUG)
 /*                                                                  */
 /* INLINE VARSTRNG ROUTINE; TRUNCATE TRAILING BLANKS                */
 /*                                                                  */
 CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
 ATTN DO
  SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

 END
 ERROR DO
  SET &RET = &LASTCC
  RETURN
 END
 IF &SYSCAPS(&STR(&DEBUG)) NE DEBUG THEN DO
  ISPEXEC VGET (DEBUG)
 END
 IF &STR(&DEBUG) = &STR() THEN DO
  SET &DEBUG = NEBUG
 END
 IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
  CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 END
 IF &FLUSH = FLUSH THEN DO
  SET &DEBUG = NEBUG
  ISPEXEC VPUT (DEBUG) SHARED
  SET &ZEDSMSG = &str(Function interrupted)
  ISPEXEC SETMSG MSG(ISRZØØ1)
  RETURN CODE(Ø)
 END
 ISPEXEC CONTROL ERRORS RETURN
 SYSREF &RSTRING
 SET &RSTRING = &STR(&SYSNSUB(1,&STRING))
```

```
 IF &STR(&SYSNSUB(1,&STRING)) = &STR() THEN DO
  RETURN CODE(Ø)
 END
 SET &RET = Ø
 SET &ALLBLANKS = +
 &STR(                                                                  -
                                                                        -
                                                                        -
                                                                        )
 IF &LENGTH(&SYSNSUB(1,&STRING)) > Ø THEN DO
  SET &SRCHSPC = &STR( )    /* look for last blank */
  SET &STARTSPC = 1
  SET &LOCSPC = 1
  SET &LASTSPC = Ø
  SET &LENRCP = &LENGTH(&STR(&SYSNSUB(1,&STRING)))
  SET &MAXT = &LENRCP
  SET &R = Ø
  DO WHILE &LOCSPC > Ø AND &STARTSPC <= &LENRCP AND &R < &MAXT
   SET &R = &R + 1
   SET &LOCSPC = +
   &SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&STRING)),&STARTSPC)
   IF &LOCSPC > Ø THEN DO
    SET &STARTSPC = &LOCSPC + 1
    SET &LASTSPC = &LOCSPC
    IF &LASTSPC > Ø AND &LENRCP > &EVAL(&LASTSPC + 1) +
    THEN DO /* REST BLANKS */
     IF &SUBSTR(&LASTSPC:&LENRCP,&STR(&SYSNSUB(1,&STRING))) = +
     &SUBSTR(1:&LENRCP-&LASTSPC-1,&STR(&ALLBLANKS)) THEN DO
      SET &STRING = +
      &SUBSTR(1:&LASTSPC-1,&STR(&SYSNSUB(1,&STRING)))
      SET &R = &MAXT
     END
    END
   END
  END
 END
 SET &RSTRING = &STR(&SYSNSUB(1,&STRING))
 RETURN CODE(Ø)
END
 /*                                                                    */
SCANFR: +
PROC 5 SEARCH RA RR NETDLM ATSIGN DEBUG(NEBUG)
 /*                                                                    */
 /* INLINE ROUTINE TO LOCATE CC-RECEIVERS                              */
 /*                                                                    */
 CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
 ATTN DO
  SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

 END
 ERROR DO
```

```
 SET &RET = &LASTCC
 RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) NE DEBUG THEN DO
 ISPEXEC VGET (DEBUG)
END
IF &STR(&DEBUG) = &STR() THEN DO
 SET &DEBUG = NEBUG
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &FLUSH = FLUSH THEN DO
 SET &ZEDSMSG = &str(Function interrupted)
 ISPEXEC SETMSG MSG(ISRZØØ1)
 RETURN CODE(Ø)
END
ISPEXEC CONTROL ERRORS RETURN
 /* set up a variable of length at least the same as maxlrecl of mail*/
SET &COMMABLANKS = +
&STR(,                                                                  -
                                                                        -
                                                                        -
                                                                        )
ISPEXEC VGET (HOSTNAME,SMTPNODE,OWNNODE,NICKOWN)
SET &OWNNET = &STR(&OWNNODE..&SMTPNODE)
SET &OWNALT = &STR(&OWNNODE..&NICKOWN)
ISREDIT CURSOR = 1 Ø
SET &RCPTTX = &STR('&SEARCH')
SET &LRCPTTX = &LENGTH(&STR(&SEARCH))
SET &MAXCNT5 = 512   /* SET LOOP CONTROL TO MAX 512 SCANS */
SET &CNT5 = Ø
SET &TORET = Ø
DO WHILE &TORET = Ø AND &CNT5 < &MAXCNT5
 SET &CNT5 = &CNT5 + 1
 IF &CNT5 = &MAXCNT5 THEN DO
  WRITE ====> Error, loop on CNT5 terminated (&SYSICMD).
 END
 SET &DM = &STR()
 SET &RET = Ø
 ISREDIT SEEK &RCPTTX 1 &RA &RR
 SET &TORET = &RET
 IF &TORET = Ø THEN DO
  ISREDIT (RCPBEGIN,RCPFROW) = CURSOR
  ISREDIT (RCPFREC) = LINE &STR(&SYSNSUB(1,&RCPBEGIN))
  SET &LENRCP = &LENGTH(&STR(&SYSNSUB(1,&RCPFREC)))
  SET &SYSDVAL = +
  &SUBSTR(&RCPFROW+&LRCPTTX:&LENRCP,&STR(&SYSNSUB(1,&RCPFREC)))
  SET &SYSDVAL = &STR(&SYSNSUB(1,&SYSDVAL))
  READDVAL &A1 &A2 &A3 &A4 &A5 &A6 &A7 &A8 &A9 &A1Ø &A11 &A12
  SET &SRCHSPC = &STR(,)     /* look for last comma */
```

```
SET &STARTSPC = 1
SET &LOCSPC = 1
SET &LASTSPC = Ø
SET &MAXT = &LENRCP
SET &R = Ø
DO WHILE &LOCSPC > Ø AND &STARTSPC <= &LENRCP AND &R < &MAXT
 SET &R = &R + 1
 SET &LOCSPC = +
 &SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&RCPFREC)),&STARTSPC)
 IF &LOCSPC > Ø THEN DO
  SET &STARTSPC = &LOCSPC + 1
  SET &LASTSPC = &LOCSPC
 END
END
IF &LASTSPC > Ø THEN DO      /* is comma followed by all blanks    */
 IF &SUBSTR(&LASTSPC:&LENRCP,&STR(&SYSNSUB(1,&RCPFREC))) = +
 &SUBSTR(1:&LENRCP-&LASTSPC-1,&STR(&COMMABLANKS)) THEN DO
  ISREDIT CURSOR = &EVAL(&RCPBEGIN + 1) Ø   /* continue search     */
  SET &RCPTTX = &STR(P'=')                  /* for any character   */
  SET &LRCPTTX = Ø
 END
 ELSE DO
  SET &RCPTTX = &STR('&SEARCH')            /* else look for        */
  SET &LRCPTTX = &LENGTH(&STR(&SEARCH))    /* next &SEARCH         */
 END
END
ELSE DO
 ISREDIT CURSOR = &EVAL(&RCPBEGIN + 1) Ø   /* continue search      */
 SET &RCPTTX = &STR('&SEARCH')
 SET &LRCPTTX = &LENGTH(&STR(&SEARCH))
END
SET &MAXCNT = 16
SET &N = Ø
SET &A = A
DO WHILE &N < &MAXCNT
 SET &N = &N + 1
 IF &N = &MAXCNT THEN DO
  WRITE ====> Error, loop on N terminated (&SYSICMD).
 END
 SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N)))
 IF &SUBSTR(1:1,&STR(&SYSNSUB(2,&&A&N))) = &STR(<) THEN DO
  SET &&A&N = &SUBSTR(2:&LENRC,&STR(&SYSNSUB(2,&&A&N)))
 END
 SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N)))
 IF &SUBSTR(&LENRC:&LENRC,&STR(&SYSNSUB(2,&&A&N))) = &STR(>) THEN DO
  SET &&A&N = &SUBSTR(1:&LENRC-1,&STR(&SYSNSUB(2,&&A&N)))
 END
 SET &LENRC = &LENGTH(&STR(&SYSNSUB(2,&&A&N)))
 SET &CC = &STR(&SYSNSUB(2,&&A&N))
 SET &SRCHSPC = &STR(&ATSIGN) /* look for at sign */
 SET &STARTSPC = 1
```

```
SET &LOCSPC = +
&SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&CC)),&STARTSPC)
IF &LOCSPC > Ø THEN DO
 SET &CC = &SUBSTR(1:&LOCSPC-1,&STR(&SYSNSUB(2,&&A&N)))
 SET &DM = &SUBSTR(&LOCSPC+1:&LENRC,&STR(&SYSNSUB(2,&&A&N)))
 SET &LENCC = &LENGTH(&STR(&SYSNSUB(1,&CC)))
 SET &SRCHSPC = &STR(&NETDLM) /* look for %-sign */
 SET &STARTSPC = 1
 SET &LOCSPC = +
 &SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&CC)),&STARTSPC)
 IF &LOCSPC > Ø THEN DO     /* nje-network address                */
  SET &US = &SUBSTR(1:&LOCSPC-1,&STR(&SYSNSUB(1,&CC)))
  SET &NET = &STR()
  SET &NET = &SUBSTR(&LOCSPC+1:&LENCC,&STR(&SYSNSUB(1,&CC)))
  IF &STR(&SYSLC(&OWNNET)) = &STR(&SYSLC(&SYSNSUB(1,&NET))) THEN DO
   SET &CC = &STR(&SYSNSUB(1,&US))
   SET &DM = &STR(&SYSNSUB(1,&OWNNODE))
  END
  IF &STR(&SYSLC(&OWNALT)) = &STR(&SYSLC(&SYSNSUB(1,&NET))) THEN DO
   SET &CC = &STR(&SYSNSUB(1,&US))
   SET &DM = &STR(&SYSNSUB(1,&OWNNODE))
  END
 END
 IF &STR(&SYSNSUB(1,&DM)) NE &STR() THEN DO
  SET &RV = RV
  SET &RP = Ø
  ISPEXEC VGET (MAXRP)
  DO WHILE &RP < &MAXRP
   SET &RP = &RP + 1
   IF &RP = &MAXRP THEN DO
    WRITE ====> Error, loop on RP terminated (&SYSICMD).
   END
   SET &RN = 1
   ISPEXEC VGET (RV&RN)
   SET &C = &STR(&SYSNSUB(2,&&RV&RN))
   DO WHILE &STR(&SYSNSUB(1,&C)) NE &STR() AND &RN < &MAXRP
    IF &STR(&SYSLC(&SYSNSUB(1,&CC))&ATSIGN+
    &SYSLC(&SYSNSUB(1,&DM))) = &STR(&SYSLC(&SYSNSUB(1,&C))) THEN DO
     SET &RP = &MAXRP
     SET &RN = &MAXRP
     SET &RETCODE = Ø
    END
    ELSE DO
     SET &RN = &RN + 1
     ISPEXEC VGET (RV&RN)
     SET &C = &STR(&SYSNSUB(2,&&RV&RN))
    END
   END
   IF &RP < &MAXRP THEN DO
    SET &&RV&RN = +
    &SYSLC(&STR(&SYSNSUB(1,&CC)&ATSIGN&SYSNSUB(1,&DM)))
```

```
        ISPEXEC VPUT (RV&RN)
        SET &RP = &MAXRP
        SET &RETCODE = Ø
       END
      END
     END
    END
    SET &P = &N + 1
    IF &STR(&SYSNSUB(2,&&A&P)) = &STR() THEN DO
     SET &N = &MAXCNT
    END
   END
  END
 END
 RETURN CODE(&RETCODE)
END
  /*                                                                  */
FREERSC: +
PROC Ø DEBUG(NEBUG)
 /*                                                                  */
 /* INLINE RETURN ROUTINE TO FREE EVT ALLOCATED RESOURCES            */
 /*                                                                  */
 CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
 ATTN DO
  SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

 END
 ERROR DO
  SET &RET = &LASTCC
  RETURN
 END
 IF &SYSCAPS(&STR(&DEBUG)) NE DEBUG THEN DO
  ISPEXEC VGET (DEBUG)
 END
 IF &STR(&DEBUG) = &STR() THEN DO
  SET &DEBUG = NEBUG
 END
 IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
  CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 END
 IF &FLUSH = FLUSH THEN DO
  SET &ZEDSMSG = &str(Function interrupted)
  ISPEXEC SETMSG MSG(ISRZØØ1)
  RETURN CODE(Ø)
 END
 ISPEXEC CONTROL ERRORS RETURN
 ISREDIT (CHANGED) = DATA_CHANGED
 ISREDIT (MEMBER) = MEMBER
 ISREDIT (DSNAME) = DATASET
 IF &CHANGED = YES AND &STR(&MEMBER) NE &STR() THEN DO
  SET &SMAXCC = &MAXCC
```

```
 SET &RET = Ø
 ISPEXEC LMINIT DATAID(DID) DATASET('&STR(&DSNAME)') ENQ(SHRW)
 ISPEXEC LMOPEN DATAID(&DID) OPTION(INPUT)
 SET &LMRET = &RET
 IF &LMRET = Ø THEN DO
  SET &RET = Ø
  ISPEXEC LMMLIST DATAID(&DID) STATS(YES) MEMBER(MEMBER) OPTION(LIST)
  SET &RLVERS = &STR(&ZLVERS)
  SET &RLMOD = &STR(&ZLMOD)
  SET &RLMDATE = &STR(&ZLMDATE)
  SET &RLMTIME = &STR(&ZLMTIME)
  SET &RLUSER = &STR(&ZLUSER)
 END
 SET &LMRET = &RET
 IF &LMRET > 8 THEN DO        /* ACCEPT MAX RC8 FROM LMMLIST         */
  SLEEP 1
  WRITE &STR(===>) Error &LMRET LMMLIST (&SYSICMD).
 END
 SET &MAXCC = &SMAXCC
 SET &RET = Ø
 ISREDIT SAVE
 SET &SAVERET = &RET
 IF &RET = 4 THEN DO        /* NEW MEMBER */
  SET &RET = Ø
  SET &SAVERET = Ø
  SET &MAXCC = Ø
 END
 IF &SAVERET = 2Ø THEN DO
  %COMPLIB &STR('&DSNAME') SHR
  SET &MAXCC = Ø
  IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
   WRITE ======> Reentering &SYSICMD <======
  END
  /* TRY TO SAVE ONCE MORE                                          */
  ISREDIT SAVE
 END
 SET &SMAXCC = &MAXCC
 IF &STR(&RLUSER) = RECEIVE OR &STR(&RLUSER) = MAILED OR +
 &STR(&RLUSER) = REPLY OR &STR(&RLUSER) = SAVED THEN DO
   /* SAVE OF STATISTICS DURING EDIT WORKS ONLY ON ISPF V.4          */
  ISPEXEC LMMSTATS DATAID(&DID) MEMBER(&NRSTR(&MEMBER)) +
  VERSION(&RLVERS) MODDATE(&STR(&RLMDATE)) +
  MODTIME(&STR(&RLMTIME)) USER(&RLUSER)
 END
 ISPEXEC LMMLIST DATAID(&DID) OPTION(FREE)
 ISPEXEC LMCLOSE DATAID(&DID)
 ISPEXEC LMFREE  DATAID(&DID)
 SET &MAXCC = &SMAXCC
END
ISPEXEC VGET (RS)
IF &STR(&RS) = &STR() THEN DO
```

```
 SET &RS = .ZL
END
 /* If external macro call to ZCC after ZREPLY label .RS at level 1  */
 /* cannot be seen                                                   */
ISREDIT (RSLINE) = LINENUM &RS
IF &RSLINE = Ø THEN DO
 ISREDIT CURSOR = 1 Ø
 SET &RET = Ø
 ISREDIT SEEK &STR('_______ Reply Separator ')
 IF &RET = Ø THEN DO
  ISREDIT (SUBJR,SUBJC) = CURSOR
  SET &SUBJR = &SUBJR - 1
  SET &SMAXCC = &MAXCC
  SET &RET = Ø
  ISREDIT LABEL &SUBJR = .RS 1
  IF &RET > 8 THEN DO     /* allow replace of label */
   IF &RET > &SMAXCC THEN DO
    SET &SMAXCC = &RET
   END
  END
  SET &MAXCC = &SMAXCC
  SET &RS = .RS
 END
 ELSE DO
  SET &RS = .ZL
 END
END
ISREDIT CURSOR = 1 Ø
SET &RCPTTX = &STR('To:')
SET &RET = Ø
ISREDIT SEEK &RCPTTX 1 .ZF &RS
IF &RET = Ø THEN DO
 ISREDIT (TOBEGIN,TOFROW) = CURSOR
 SET &TOBEGIN = &TOBEGIN - 1
 SET &SMAXCC = &MAXCC
 SET &RET = Ø
 ISREDIT LABEL &TOBEGIN = .TC 1
 IF &RET > 8 THEN DO     /* allow replace of label */
  IF &RET > &SMAXCC THEN DO
   SET &SMAXCC = &RET
  END
 END
 SET &MAXCC = &SMAXCC
 ISREDIT EXCLUDE .ZF .TC P'^' ALL
END
ISREDIT (L) = LINENUM &RS
SET &COL = 1
ISREDIT CURSOR = (L,COL)
SET &MEMOCC = &STR()
SET &CCDM = &STR()
SET &DOMAIN = &STR()
```

```
 ISPEXEC VPUT (MEMOCC,CCDM,DOMAIN)
 RETURN CODE(Ø)
END
  /*                                                                  */
```

CLIST MAILEDIT for editing mail in mailbox dataset:

```
 /*                                                                  */
 /* MAILEDIT
 /*  Edit SMTP Mailbox Dataset
 /*   Called from Panel INST81
 /*
 /*   SUBROUTINES/EDIT MACROS:
 /*    %EDITRECV
 /*
 /*   Utilities used:
 /*                                                                  */
PROC Ø DEBUG(nEBUG) LOGDS(SMTP)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ATTN DO
 SET &FLUSH = FLUSH         /* NEXT STATEMENT MUST BE NULL LINE     */

END
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
SET &RET = Ø
IF &STR(&DEBUG) NE DEBUG THEN DO
 ISPEXEC VGET (DEBUG)
 IF &STR(&DEBUG) = &STR() THEN DO
  SET &DEBUG = NEBUG
 END
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 WRITE ======> Entering &SYSICMD <======
END
IF &FLUSH = FLUSH THEN DO
 SET &ZEDSMSG = &str(Function interrupted)
 ISPEXEC SETMSG MSG(ISRZØØ1)
 EXIT CODE(4)
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE ====> Sorry only executable under ISPF (&SYSICMD).
 EXIT CODE(16)
END
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC VGET (MEMODS MEMOMB)
SET &MAXSMTPLR = 251
SET &BLKSMTPLR = 27861
```

```
SET &LDSN = &LENGTH(&STR(&MEMODS))
IF &SUBSTR(1:1,&STR(&MEMODS)) = &STR(') THEN DO
 SET &MEMODS = &SUBSTR(2:&LDSN,&STR(&MEMODS))
END
SET &LDSN = &LENGTH(&STR(&MEMODS))
IF &SUBSTR(&LDSN:&LDSN,&STR(&MEMODS)) = &STR(') THEN DO
 SET &MEMODS = &SUBSTR(1:&LDSN - 1,&STR(&MEMODS))
END
IF &SYSDSN(&STR('&MEMODS')) = &STR(DATASET NOT FOUND) THEN DO
 IF &STR(&MEMODS) = &SYSUID..&LOGDS THEN DO
  ALLOC FI(MEMOFILE) NEW CATALOG SPACE(1 3) CYLINDERS DIR(45) +
  RECFM(F B) LRECL(&MAXSMTPLR) BLKSIZE(&BLKSMTPLR) +
  MGMTCLAS(STANDARD) STORCLAS(STANDARD) +
  DA('&SYSUID..&LOGDS') REUSE
  ALLOC FI(MEMOFILE) DA('&MEMODS(DUMMY)') SHR REUSE
  OPENFILE MEMOFILE OUTPUT
  CLOSFILE MEMOFILE
  FREE FI(MEMOFILE)
 END
END
SET &SYSOUTTRAP = 9999
SET &RET = Ø
LISTC ENT(&STR('&MEMODS')) ALIAS
SET &ALRET = &RET
SET &SYSOUTTRAP = Ø
IF &SYSDSN(&STR('&MEMODS')) NE OK OR &ALRET = Ø THEN DO
 ISPEXEC SETMSG MSG(INST343)
 EXIT CODE(4)
END
SET &SYSOUTTRAP = 9999
LISTD '&STR(&MEMODS)'
SET &SYSOUTTRAP = Ø
SET &SYSDVAL = &STR(&SYSOUTLINE3)
IF &SUBSTR(1:1,&STR(&SYSDVAL)) = &STR(-) THEN DO
 SET &SYSDVAL = &STR(&SYSOUTLINE4)
END
READDVAL &RF &LR &BL &DS
IF &LR NE &MAXSMTPLR THEN DO
 ISPEXEC SETMSG MSG(INST348)
END
IF &DS NE PO AND &DS NE POU THEN DO
 ISPEXEC SETMSG MSG(INST342)
END
SET &DS = &STR(&MEMODS)
IF &STR(&MEMOMB) NE &STR() THEN DO
 SET &DS = &STR(&DS(&MEMOMB))
END
%EDITRECV                    /* take edit recovery */
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 WRITE ======> Reentering &SYSICMD <======
END
```

```
ISPEXEC EDIT DATASET(&STR('&DS'))
FREE FI(MEMOFILE)
EXIT CODE(Ø)
 /*                                                                  */
```

CLIST USERNAME gets user name from RACF and saves it into ISPF profile:

```
 /*
 /* USERNAME
 /*  Return Name of User from RACF
 /*
 /*  PARAMETERS:
 /*   USER:     DEFAULTS TO CALLER'S USER-ID
 /*   USERNAME: ISPF PROFILE VARIABLE CONTAINING NAME OF USER FROM RACF
 /*
 /*  RETURN CODE:
 /*   Ø : NAME FOUND
 /*   8 : NAME NOT FOUND
 /*                                                                  */
PROC Ø USER() DEBUG(nEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 WRITE ======> Entering &SYSICMD <======
END
IF &FLUSH = FLUSH THEN DO
 SET &ZEDSMSG = &str(Function interrupted)
 ISPEXEC SETMSG MSG(ISRZØØ1)
 EXIT CODE(Ø)
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE ====> Sorry only executable under ISPF (&SYSICMD).
 EXIT CODE(16)
END
ISPEXEC CONTROL ERRORS RETURN
SET &STATUS = &STR()
SET &RET = Ø
SET &SYSOUTTRAP = 99999
LU &STR(&USER)
SET &SYSOUTTRAP = Ø
IF &RET = Ø THEN DO
 SET &MAXLINE = &SYSOUTLINE
 SET &ALLBLANKS = +
 &STR(                                                               -
                                                                     -
                                                                     -
```

```
                                                                      )
SET &LENNAME = &LENGTH(&STR(NAME=)
SET &LENOWN  = &LENGTH(&STR(OWNER=)
SET &NAMEUSED = NO
SET &Q = Ø
DO WHILE &Q < &MAXLINE
 SET &Q = &Q + 1
 SET &SYSDVAL = &STR(&SYSNSUB(2,&&SYSOUTLINE&Q))
 SET &SYSDVAL = &STR(&SYSNSUB(1,&SYSDVAL))
 SET &A = &STR(&SYSNSUB(1,&SYSDVAL))
 READDVAL &A1 &A2 &A3 &A4 &A5 &A6
 IF &NAMEUSED = NO THEN DO
  IF &LENGTH(&STR(&SYSNSUB(1,&A))) > &EVAL(&LENDFGR+&LENOWN) THEN DO
   SET &NAMESTR = &SYSINDEX(&STR(NAME=),&STR(&SYSNSUB(1,&A)),1)
   IF &NAMESTR > Ø THEN DO
    SET &NAMESTR = &NAMESTR + &LENNAME
    SET &NAMEEND = +
    &SYSINDEX(&STR(OWNER=),&STR(&SYSNSUB(1,&A)),&NAMESTR)
    IF &NAMEEND > Ø THEN DO
     SET &NAMEEND = &NAMEEND - 1
    END
    ELSE DO
     SET &NAMEEND = &LENGTH(&STR(&SYSNSUB(1,&A)))
    END
    SET &USERNAME = &SUBSTR(&NAMESTR:&NAMEEND,&STR(&SYSNSUB(1,&A)))
    IF &LENGTH(&SYSNSUB(1,&USERNAME)) > Ø THEN DO
     SET &SRCHSPC = &STR( )    /* look for last blank */
     SET &STARTSPC = 1
     SET &LOCSPC = 1
     SET &LASTSPC = Ø
     SET &LENRCP = &LENGTH(&STR(&SYSNSUB(1,&USERNAME)))
     SET &MAXT = &LENRCP
     SET &R = Ø
     DO WHILE &LOCSPC > Ø AND &STARTSPC <= &LENRCP AND &R < &MAXT
      SET &R = &R + 1
      SET &LOCSPC = +
      &SYSINDEX(&STR(&SRCHSPC),&STR(&SYSNSUB(1,&USERNAME)),&STARTSPC)
      IF &LOCSPC > Ø THEN DO
       SET &STARTSPC = &LOCSPC + 1
       SET &LASTSPC = &LOCSPC
       IF &LASTSPC > Ø AND &LENRCP > &EVAL(&LASTSPC + 1) +
       THEN DO /* REST BLANKS */
        IF &SUBSTR(&LASTSPC:&LENRCP,&STR(&SYSNSUB(1,&USERNAME))) = +
        &SUBSTR(1:&LENRCP-&LASTSPC-1,&STR(&ALLBLANKS)) THEN DO
         SET &USERNAME = +
         &SUBSTR(1:&LASTSPC-1,&STR(&SYSNSUB(1,&USERNAME)))
         SET &R = &MAXT
        END
       END
      END
     END
```

```
     END
     ISPEXEC VPUT (USERNAME) PROFILE
     SET &NAMEUSED = YES
    END
   END
  END
  IF &NAMEUSED = YES THEN DO
   SET &Q = &MAXLINE
  END
 END
END
ELSE DO
 SET &USERNAME = &STR()
END
IF &NAMEUSED = YES THEN DO
 IF &SYSNEST = NO THEN DO
  WRITE &STR(&SYSNSUB(1,&USERNAME))
 END
 EXIT CODE(Ø)
END
ELSE DO
 EXIT CODE(8)
END
 /*                                                                  */
```

CLIST ZSMTPI (edit macro) to set ISPF note lines in received mail:

```
 /*
 /* ZSMTPI
 /* Edit macro to set edit messages in received SMTP-note.
 /*  Called from ZSMTPR
 /*                                                                  */
PROC Ø DEBUG(nEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &SYSCAPS(&STR(&DEBUG)) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
 WRITE ======> Entering &SYSICMD <======
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE ====> Sorry only executable under ISPF (&SYSICMD).
 EXIT CODE(16)
END
IF &SYSNEST = NO THEN DO
 ISREDIT MACRO PROCESS
END
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC VGET (ZSCREEN)
```

```
IF &SYSENV = FORE THEN DO
 SET &DATA = +
 &STR('To reply:        enter ZREPLY and hit ENTER.')
 ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
 SET &DATA = +
 &STR('To save mail:    hit END (PF3/PF15).')
 ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
 SET &DATA = +
 &STR('To forward mail: enter ZMAIL and hit ENTER.')
 ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
 SET &DATA = +
 &STR('To delete mail:  enter ZDELETE and hit ENTER.')
 ISREDIT LINE_BEFORE .ZF =  MSGLINE MASKLINE + &DATA
 ISREDIT CURSOR = 1 Ø
END
EXIT CODE(1)
 /*                                                                  */
```

The CLIST EDITRECV to test and take pending ISPF edit recovery. This CLIST should always be called just before entering ISPF edit from a CLIST:

```
 /*                                                                  */
 /* EDITRECV                                                         */
 /*  FETCHES PENDING EDIT RECOVERY IF ANY FROM ISPEXEC EDIT          */
 /*                                                                  */
PROC Ø DEBUG(NEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 EXIT
END
ISPEXEC CONTROL ERRORS RETURN
ISPEXEC VGET (ZAPPLID)
ISPEXEC EDREC INIT    /* INIT RECOVERY TABLE IF NOT EXISTING */
SET &RECOV =  NO
DO WHILE &RECOV = NO
 SET &RET = Ø
 ISPEXEC EDREC QUERY
 IF &RET = 4 THEN DO
  ISPEXEC CONTROL DISPLAY REFRESH
  ISPEXEC DISPLAY PANEL(ISREDMØ2)
  IF &STR(&ZEDCMD) = &STR() THEN DO
   ISPEXEC EDREC PROCESS
  END
```

```
  ELSE DO
   IF &SUBSTR(1:1,&STR(&ZEDCMD)) = C THEN DO
    ISPEXEC EDREC CANCEL
   END
   IF &SUBSTR(1:1,&STR(&ZEDCMD)) = D THEN DO
    ISPEXEC EDREC DEFER
   END
  END
 END
 ELSE DO
  SET &RECOV = YES
 END
END
EXIT
```

The edit macro ZDELETE to delete current edit-member:

```
 /*
 /* ZDELETE
 /*  EDIT MACRO TO DELETE CURRENT MEMBER
 /*
PROC Ø
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &SYSNEST = NO THEN DO
 ISREDIT MACRO (DEBUG) PROCESS
 ISPEXEC CONTROL ERRORS RETURN
END
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH NOLIST CONLIST SYMLIST
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE SORRY ONLY EXECUTABLE UNDER ISPF
 EXIT CODE(16)
END
ISREDIT (DSNAME) = DATASET
ISREDIT (MEMBER) = MEMBER
IF &SYSPREF.. = . THEN DO
 IF &SYSDSN(&STR('&DSNAME')) = &STR(DATASET NOT FOUND) THEN DO
  SET &DSNAME = &STR(&SYSUID..&DSNAME)
 END
END
IF &MEMBER.. = . THEN DO
 IF &SYSDSN('&DSNAME') ¬= &STR(DATASET NOT FOUND) THEN DO
  ALLOC FI(@@@@@@@@) DA('&DSNAME') OLD REUSE
  TSOEXEC DEL '&DSNAME' FI(@@@@@@@@)
 END
 ELSE DO
  ISPEXEC VGET (ZDLVOL ZDLCAT)
```

```
  IF &STR(&ZDLVOL) ¬= MIGRAT AND &STR(&ZDLVOL) ¬= &STR() THEN DO
   SET &VL = &STR(VOLUME(&ZDLVOL))
   ALLOC FI(@@@@@@@@) DA('&DSNAME') UNIT(SYSALLDA) &STR(&VL) OLD REUSE
   TSOEXEC DEL '&DSNAME' FI(@@@@@@@@)
  END
 END
END
ELSE DO
 IF &SYSDSN('&DSNAME(&MEMBER)') ¬= &STR(MEMBER NOT FOUND) THEN DO
  ISPEXEC LMINIT DATAID(DID) DATASET('&DSNAME') ENQ(SHRW)
  ISPEXEC LMOPEN DATAID(&DID) OPTION(OUTPUT)
  SET &RET = Ø
  ISPEXEC LMMDEL DATAID(&DID) MEMBER(&MEMBER) NOENQ
  IF &RET > Ø THEN DO
   SET &MAXCC = Ø
   SET &RET = Ø
   ISPEXEC LMMDEL DATAID(&DID) MEMBER(&MEMBER)
  END
  ISPEXEC LMCLOSE DATAID(&DID)
  ISPEXEC LMFREE DATAID(&DID)
 END
 ELSE DO
  ISPEXEC VGET (ZDLVOL ZDLCAT)
  IF &STR(&ZDLVOL) ¬= MIGRAT AND &STR(&ZDLVOL) ¬= &STR() THEN DO
   SET &VL = &STR(VOLUME(&ZDLVOL))
   SET &RET = Ø
   ISPEXEC LMINIT DATAID(DID) DATASET('&DSNAME') ENQ(SHRW) &STR(&VL)
   ISPEXEC LMOPEN DATAID(&DID) OPTION(OUTPUT)
   SET &RET = Ø
   ISPEXEC LMMDEL DATAID(&DID) MEMBER(&MEMBER) NOENQ
   IF &RET > Ø THEN DO
    SET &MAXCC = Ø
    SET &RET = Ø
    ISPEXEC LMMDEL DATAID(&DID) MEMBER(&MEMBER)
   END
   ISPEXEC LMCLOSE DATAID(&DID)
   ISPEXEC LMFREE DATAID(&DID)
   /* MEMBER DID NOT EXIST, SO IGNORE ANY RETURN CODE, JUST ISSUE CAN
   IF &RET > Ø THEN DO
    SET &MAXCC = Ø
   END
  END
  ELSE DO
   SET &MAXCC = Ø
  END
 END
END
IF &MAXCC = Ø THEN DO
 FREE FI(@@@@@@@@)
 IF &MEMBER.. ¬= . THEN DO
  SET &MEM = &STR((&MEMBER))
```

```
 END
 SET &ZEDLMSG = +
 &STR(DATASET &DSNAME.&MEM DELETED)
 ISPEXEC SETMSG MSG(ISRZØØ1)
 ISREDIT CANCEL
END
ELSE DO
 FREE FI(@@@@@@@@)
 SET &ZEDLMSG = +
 &STR(=========> DELETION COULD NOT BE PERFORMED <==========)
 ISPEXEC SETMSG MSG(ISRZØØ1)
END
EXIT CODE(Ø)
```

CLIST ISPFCPY copies member with ISPF statistics:

```
 /*
 /* ISPFCPY
 /*  COPIES A MEMBER WITH ISPF AND SETS ISPF STATISTICS
 /*   CAN BE USED AS A GENERAL ISPF COPY FUNCTION.
 /*   EXAMPLE OF JCL FOR EXECUTION:
 /*   //S1 EXEC ISPFBAT,
 /*   // PARM.ISPFBAT='ISPSTART CMD(%ISPFCPY &TDFMEM.ØØ IN OUT)',
 /*   //ISPFBAT.IN  DD DSN=*.SAVEPGM.SYSUT2,DISP=(OLD,DELETE)
 /*   //ISPFBAT.OUT DD DSN=XXX.YYYY.ZZZZZZZZ,DISP=SHR
 /*
 /*  PARAMETERS:
 /*  1: INPUT MEMBER             (POSITIONAL)
 /*  2: INPUT DDNAME OR DSNAME   (POSITIONAL)
 /*  3: OUTPUT DDNAME OR DSNAME  (POSITIONAL)
 /*  KEYWORD TYPE
 /*     TYPE   : DDNAME OR DSNAME   (KEYWORD) DEFAULT: DDNAME
 /*     REPLACE: REPLACE OR NOREPLACE COPY OPTION; DEFAULT: REPLACE
 /*
 /*  INPUT DDNAME MUST BE PREALLOCATED IF TYPE=DDNAME.
 /*  OUTPUT DDNAME MUST BE PREALLOCATED IF TYPE=DDNAME.
 /*  IF TYPE=DSNAME, THE FILES NEED NOT BE PREALLOCATED.
 /*
PROC 3 MEMBER INDD OUTDD TYPE(DDNAME) REPLACE(REPLACE) DEBUG(DEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST NOCAPS
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &STR(&DEBUG) = DEBUG OR &STR(&DEBUG) = D THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST NOCAPS
END
IF &STR(&TYPE) NE DDNAME AND &STR(&TYPE) NE DSNAME THEN DO
 WRITE ERROR TYPE NOT DDNAME OR DSNAME
 EXIT CODE(16)
END
```

```
IF &SYSENV NE FORE THEN DO
 WRITE &STR(MEMBER=&MEMBER)
END
IF &TYPE = DDNAME THEN DO
 LISTDSI &OUTDD FILE
 SET &OUTDSN = &SYSDSNAME
 SET &LMIDD = DDNAME(&STR(&INDD))
 SET &LMODD = DDNAME(&STR(&OUTDD))
END
ELSE DO
 SET &OUTDSN = &OUTDD
 SET &LMIDD = DATASET(&STR('&INDD'))
 SET &LMODD = DATASET(&STR('&OUTDD'))
END
IF &SYSENV NE FORE THEN DO
 WRITE &STR(OUTPUT DSNAME=&OUTDSN)
END
IF &STR(&REPLACE) = NOREPLACE THEN DO
 SET &REPLACE = &STR()
END
SET &RET = Ø
ISPEXEC LMINIT DATAID(DID) &LMIDD ENQ(SHR)
IF &RET NE Ø THEN DO
 WRITE ERROR OPENING &STR(&INDSN) RC=&RET
 EXIT CODE(16)
END
SET &RET = Ø
SET &DIDI = &DID
ISPEXEC LMOPEN DATAID(&DIDI) OPTION(INPUT)
IF &RET NE Ø THEN DO
 WRITE ERROR ACCESSING &STR(&INDSN) RC=&RET
 EXIT CODE(16)
END
SET &RET = Ø
ISPEXEC LMINIT DATAID(DID) &LMODD ENQ(SHRW)
IF &RET NE Ø THEN DO
 WRITE ERROR ACCESSING &STR(&OUTDSN) RC=&RET
 EXIT CODE(16)
END
SET &DIDO = &DID
SET &RET = Ø
ISPEXEC LMOPEN DATAID(&DIDO) OPTION(INPUT)
IF &RET NE Ø THEN DO
 WRITE ERROR OPENING OUTPUT&STR(&OUTDD) RC=&RET
 EXIT CODE(16)
END
SET &RET = Ø
ISPEXEC LMMFIND DATAID(&DIDO) MEMBER(&NRSTR(&MEMBER)) STATS(YES)
SET &FINDRET = &RET
IF &FINDRET NE Ø OR &ZLINORC = &STR() THEN DO
```

```
 SET &NEW = YES
 SET &ZLINORC = Ø
 SET &ZLCNORC = Ø
 SET &ZLMNORC = Ø
 SET &ZLVERS = 1
 SET &ZLMOD = Ø
 SET &ZLCDATE = &STR(&SYSSDATE)
END
IF &FINDRET = Ø AND &STR(&REPLACE) NE REPLACE THEN DO
 ISPEXEC LMCLOSE DATAID(&DIDI)
 ISPEXEC LMCLOSE DATAID(&DIDO)
 ISPEXEC LMFREE DATAID(&DIDI)
 ISPEXEC LMFREE DATAID(&DIDO)
 EXIT CODE(12)
END
IF &NEW NE YES THEN DO
 SET &ZLMOD = &ZLMOD + 1
END
SET &ZLMDATE = &STR(&SYSSDATE)
SET &ZLMTIME = &STR(&SYSSTIME)
SET &ZLUSER = &SYSUID
ISPEXEC LMCLOSE DATAID(&DIDO)
SET &RET = Ø
ISPEXEC LMCOPY FROMID(&DIDI) +
TODATAID(&DIDO) TOMEM(&NRSTR(&MEMBER)) &STR(&REPLACE)
SET &COPYRET = &RET
IF &COPYRET = 2Ø THEN DO
 ISPEXEC LMOPEN DATAID(&DIDO) OPTION(OUTPUT)
 SET &RET = Ø
 ISPEXEC LMMDEL DATAID(&DIDO) MEMBER(&NRSTR(&MEMBER)) NOENQ
 IF &RET > Ø THEN DO
  SET &MAXCC = Ø
  SET &RET = Ø
  ISPEXEC LMMDEL DATAID(&DIDO) MEMBER(&NRSTR(&MEMBER))
 END
 ISPEXEC LMCLOSE DATAID(&DIDO)
 %COMPLIB &STR(&OUTDSN) SHR               /* COMPRESS */
 SET &RET = Ø
 ISPEXEC LMCOPY FROMID(&DIDI) FROMMEM(&NRSTR(&MEMBER)) +
 TODATAID(&DIDO) TOMEM(&NRSTR(&MEMBER)) &STR(&REPLACE)
 SET &COPYRET = &RET
END
IF &COPYRET NE Ø THEN DO
 IF &SYSENV NE FORE THEN DO
  WRITE &NRSTR(&MEMBER) GOT ERROR FROM COPY RC=&RET
 END
END
IF &COPYRET = Ø THEN DO
 SET &RET = Ø
 IF &NEW = YES THEN DO
  ISPEXEC LMMSTATS DATAID(&DIDO) MEMBER(&NRSTR(&MEMBER))
```

```
 END
 ELSE DO
  ISPEXEC LMMSTATS DATAID(&DIDO) MEMBER(&NRSTR(&MEMBER)) +
  CREATED(&ZLCDATE) INITSIZE(&ZLINORC) MODLEVEL(&ZLMOD)
 END
 IF &RET NE Ø THEN DO
  WRITE ERROR SETTING ISPF STATISTICS RC=&RET
  SET &COPYRET = 12
 END
END
ISPEXEC LMCLOSE DATAID(&DIDI)
ISPEXEC LMFREE DATAID(&DIDI)
ISPEXEC LMFREE DATAID(&DIDO)
EXIT CODE(&COPYRET)
```

CLIST WAITUNTJ will wait until the specified address space arrives:

```
  /*
  /* WAITUNTJ
  /*  CLIST TO WAIT UNTIL A SPECIFIED ADDR SPACE IS/BECOMES ACTIVE;
  /*  IF THE ADDRESS SPACE IS ALREADY ACTIVE, RETURN WITH CC Ø.
  /*
  /*  PARAMETER JOBNAME:  ADDRESS SPACE TO BE WAITED FOR
  /*  PARAMETER WAIT   :  MAX WAIT TIME (DEFAULT TO UNLIMITED)
  /*
  /*  HALT RETURNS CC=Ø WHEN ADDRESS SPACE TO WAIT FOR BECOMES ACTIVE
  /*  HALT RETURNS CC=4 AFTER MODIFY, THUS LOOP CAN BE TERMINATED
  /*   USING: F ADSPNM,CONTINUE
  /*  HALT RETURNS CC=8 WHEN MAX TIME EXPIRED
  /*
PROC 1 JOBNAME WAIT(9999999) DEBUG(NEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST CAPS
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
SET &CNT = Ø
SET &ASVTRET = 8
SET &HALTRET = 4
DO WHILE &ASVTRET = 8 AND &HALTRET NE Ø AND &CNT < &WAIT
 SET &CNT = &CNT + 1
 SET &RET = Ø
 HALT 1
 SET &HALTRET = &RET
 IF &HALTRET > 12 THEN DO
  EXIT CODE(&HALTRET)
 END
```

```
 SET &RET = Ø
 ASVTFIND &STR(&JOBNAME)
 SET &ASVTRET = &RET
END
IF &ASVTRET = 4 THEN DO
 /* ADDRESS SPACE IS ACTIVE      */
 EXIT CODE(Ø)
END
IF &HALTRET = Ø THEN DO
 /* WAIT TERMINATED BY MODIFY    */
 EXIT CODE(4)
END
 /* MAX WAIT EXCEEDED            */
EXIT CODE(8)
END
```

Editor's note: this article will be concluded in the next issue.

Nils Plum
Systems Programmer (Denmark)

Call for papers

Why not share your expertise and earn money at the same time? *TCP/SNA Update* is looking for REXX EXECs, macros, CLISTs, program code, etc, that experienced networkers have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2.

More information about how to contribute is contained in our free booklet entitled *Notes for contributors*. Ask us and we will send you a copy, or download it from our Web site – www.xephon.com/contnote.html.

Sterling Software has announced Version 5.0 of SOLVE:Netmaster for TCP/IP, its network management software.

The new version of the OS/390-based product, with IP diagnostic and performance capabilities, now gets Web-based access capabilities and remote access facilities. It also incorporates access control technology, to help secure transactions by IP address, port, user-id, and time of day or week.

Version 5.0 also adds diagnostic support for IBM's 2216/2210 router line and associated Telnet 3270 servers, plus better security features enabling network administrators to define user identification, access, and authority levels.

And the proactive alert monitor has been given advanced controls which provide filtering and profiling of network events, and include improved graphical presentation.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, Sterling Court, Eastworth Road, Chertsey, Surrey, KT16 8DF, UK.
Tel: (0181) 867 8000.
URL: http://www.sterling.com.

* * *

HDS has announced a new VisionBase ESCON adapter, which allows its VisionBase PC servers to act as co-processors to its Skyline Trinium and Pilot Series mainframes.

It's said to be optimized for LAN gateway consolidation, data warehousing, data mining, and ERP server access of mainframe data. It uses the Multi-Path Channel Plus (MPC+) protocol to boost throughput and lower mainframe processing cycles, allowing both SNA and TCP/IP communication with the mainframe.

For further information contact:
Hitachi Data Systems, PO Box 54996, E Central Expressway, Santa Clara, CA 95056-0996, USA.
Tel: (408) 970 1000.
Hitachi Data Systems, Sefton Park, Bells Hill, Stoke Poges, Bucks, SL2 4HD, UK.
Tel: (01753) 618000.
URL: http://www.hdshq.com.

* * *

Cisco Systems has begun shipping its Cisco Transaction Connection (CTRC), providing TCP/IP end users and servers with access to DB2 databases using SNA. The software and router system based on Cisco IOS software is designed to replace Unix and NT gateways for database access.

CTRC uses the DRDA protocol to access remote databases using a standard messaging format over TCP/IP or SNA. It converts TCP/IP data requests into SNA messages and forwards them to the host.

Included with CTRC is a site licence for the StarSQL Enterprise Edition (EE) driver, which transfers data from host resources into desktop and Unix server-based applications. End users can access DB2 from tailored or ODBC-based desktop applications.

For further information contact:
Cisco, 5305 Gulf Drive, Suite 1, New Port Richey, FL 34652, USA.
Tel: (813) 817 0131.
URL: http://www.cisco.com.