SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

052015

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER: 137

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Program Library Standards Manual".

(1) Program Number (to be filled in by SPLA) 360D-05.2.015

(2) System Type (machine) .....

(3) Search Key .....

(4) Programming Language ..... Assembler

(5) Author's Name and Address ..... Software Development Division
Standard Oil Co. (Indiana)
200 E. Randolph Drive
~~Chicago, Illinois 60601~~

(6) Direct Inquiries to Name and Address ..... (if different than Author)

(7) Title of Program ..... Inter-system Shared Enque

(8) Submitter's Installation Membership Code..... IN

(9) Submitter's Own Program Identification and Suffix(Optional)...

(10) Primary Subject Code..... 05.2

(11) Operating or Monitor System Required OS REL 21

(12) New or Revision Code (if revision, show prior Program Number in Item 1).. N

(13) Year Completed..... 1976

(14) Date of Submittal..... 12-19-75

(15) Documentation (number of original pages submitted)..... 44

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

Inter-system Shared Enque, as implemented by Standard Oil Co., is designed to replace the reserve of a complete volume with an ENQ across systems for only the resources required. This is accomplished by ENQUING the same resource in both systems via a channel to channel adapter.

(Please attach additional pages if necessary).....Total pages attached____

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date M Elkins 1/8/72

(18) Signature of Installation Addressee_____

December 16, 1975

SHARE Program Library Agency
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina 27709

Dear Sirs:

Enclosed is the tape, documentation and submittal form for the Inter-System Shared Enque program.

To the best of my knowledge, my program is free of any proprietary, secret, or confidential information belonging to any person or organization. I have not used the work, plans, procedures, systems, programs or names of any companies or individuals.

TAPE KEY FOR VD0056

This is a standard labeled tape with the following format:

DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)

1st File:	Header Label
2nd File:	DSN=TPD.CTCSHENQ.SOURCE
3rd File:	Trailer Label
4th File:	Header Label
5th File:	DSN=TPD.CTCSHENQ.MACRO
6th File:	Trailer Label

Both datasets are in the correct "SYSIN" format for IEBUPDTE.

Yours truly,

Phil Smith

CTC GENERAL LOGIC FLOW

This manual is intended to explain the basic system logic of the Channel-to-Channel Access Method and delineate the design objectives. If more detailed knowledge of the system is needed, the CSECT pre-comments for the module of interest should be consulted.

Design Objectives

The basic design of the CTC Access Method was obtained with the following objectives in mind:

1. Capable of handling multiple CTC devices
2. MCS support between systems
3. Avoid master/slave relationship between processors
4. Handle multiple users on each CTC device
5. Minimum of ØS modifications
6. Require as little operator intervention as possible.

Task Structure

The resulting system is composed of one controlling task and two daughter tasks for each CTC device that is made active. These two tasks, the Supervisor and the Command Processor, synchronize all I/O and allow operator communication between processors, respectively.

In addition, the Channel End/Abnormal End Appendage and the CTC Attention Routine trap out all CTC interrupts and direct them to the CTC Supervisor.

System Work Area

Each CTC device has an associated work area to be used by the responsible CTC system tasks. This work area, obtained and initialized by the control routine, contains ECB lists, buffers, DCB's, TCB pointers and a Read/Write Table. The address of this work area is placed in the UCB for that device.

Read/Write Table

The Read/Write Table is the principle control block used by the CTC. It contains pointers to the users' READ and WRITE DECB's and to the user's system status ECB. Through this table all I/O is synchronized between processors and the status of users in both systems is maintained.

Adapter Logical States

Throughout the CTC documentation, CPU's will be referred to as X or Y. In this terminology, CPU X is "this system" while CPU Y is "the other system". Thus, if a CTC device is operational between two systems, each sees himself as X and the other as Y.

In the UCB there is an indication of which system is sending data and which is receiving. Since CPU X is always "this system", we see that if the sending control indicators say X, then this system is issueing its writes and the other system is responding with its available reads. If the UCB indicates Y, then this system is receiving and Y is sending. If the UCB indicates idle, then neither CPU is sending and the adapter is idle.

CTC Command Codes

The channel program opcodes for the CTC are:

READ	XXXX XX10
WRITE	XXXX XX01
CONTROL	XXXX X111
NOP	XXXX X011

The remainder of the bits are used by the CTC Access Method to communicate additional information to CPU Y. The high half-byte of each op-code reflects the user id associated with this channel program. Thus a Read CCW for CTCUSER6 will have an opcode of X'62'. By sensing the adapter, system Y can tell the operation to be performed and who is performing it. The CTC system reserves the use of users 0, 1 and 15 for itself. Thirteen identifiers remain for users of the Access Method. A list of the CTC opcodes follows:

Read by user X	X'X2'	, X=1-E
Write by user X	X'X1'	, X=1-E
Sense	X'04'	
Wakeup after an IPL	X'f7'	
Wakeup without IPL	X'FF'	
Switch Control (Request/Release)	X'07'	
Readtap for user X	X'X7'	, X=1-E
Close for user X	X'Xf'	, X-1-F
Close for CTC	X'0F'	

Control Routine

The Channel-to-Channel Control Routine is the mother task for all active CTC devices. When a request is made by the operator to start a CTC device, the Control Routine validates the request, initializes a CTC system work area, and ATTACH'es a Supervisor and Command Processor. In order to validate a start request, the Control Routine maintains a table of valid CTC's for each system and their associated UCB's. This table is provided in the coding of the CTCTL macro instruction which will expand into the Control Routine. By coding the Control [illegible] macro, it can be tailored to each system requirements easily. For a discussion of other Control Routine macro options, see Attachment I.

In order to monitor I/O activity on all active CTC's, the Control Routine issues a STIMER upon being entered for the first time. At the end of each time interval if no data has been transferred on a CTC device, it requests the Command Processor to send a "protocol I/O" in order to stimulate adapter activity. At the end of the subsequent time interval, if the protocol I/O hasn't completed, the status of the other system is requested from the operator.

The Control Routine regains control if either the Supervisor or Command Processor terminates. The Command Processor is reinstated the first time it abends. If the Supervisor abends, the CTC system is terminated for that device. However, at the time of a second abend in the system, the CTC is terminated for all devices.

Command Processor

The Command Processor acts as a CTC user. When posted by the Control Routine, it sends operator messages to the other system or schedules protocol I/O. If it is posted by the Supervisor, it sends any system status changes to the master console or displays operator messages from the other system. The buffers used by the Command Processor are found in the CTC work area. The MCS feature of the CTC is provided by using keywords for route codes to be sent across to the other system. Valid routing codes are listed in the CTC Operator Interface.

Supervisor

The Supervisor is the principle CTC task in that all work is received and processed in this module. At the time the Supervisor is attached,

he must notify the CTC system in Y that he is available for work. He does this by sending a "wakeup" control command. This will allow Y to post all his users of the new status of this CTC. Work processing then commences. When there is no work from either processor, the Supervisor "sits idle" by waiting on an Attention and a Work ECB.

When the Work ECB is posted, the Supervisor must first "request control" of the adapter by sending a "switch" control command. Once in the X logical state, he sends all work across. Work for the Supervisor consists of:

1. Sending a readtap notifying the other system of a READ just issued by a user in this system.

2. Sending a "close" control command informing the other system of a user closing in X.

3. Sending user data scheduled by a WRITE macro.

When all work has been sent, another "switch" command is sent releasing adapter control and returning it to idle. As the work requests complete, the users are posted and the Read/Write Table is updated.

The READ and WRITE ECB's are posted only to I/O completion. Any change in system status (e.g., other system stopped, user in Y closed) will be reflected by posting the system status ECB for the corresponding user or all users depending on the resulting status.

OS Modifications

1. The UCB's for the CTC devices have to be zapped to indicate the CTC device type. This is X'40F0'.

OS Modifications by Control Routine When Started:

1. The address of the CTC Attention routine is placed into the attention table and its index is placed into the UCB.

2. The CTC saves the original TIOT and uses its own. At task termination, the original TIOT is restored.

PP:kl
June 1972

CTCTL - CONTROL ROUTINE MACRO

The CTCTL macro will expand into a CTC Control Routine tailored to the requested specifications. Optional generating parameters for this macro include:

DF=n — Maximum number of users on a CTC device if omitted DF=4 is assumed.

SUPR= — Module name of CTCAM Supervisor. If omitted SUPR=CTCAMSUP is assumed.

CMD= — Module name of CTCAM Command Processor. If omitted CMD=CTCAMCMD is assumed.

SYS= — Code generated for either 370 or 360 CPU. Default is SYS=370.

ATTNØF= — Index into attention table for attention routine. Default ATTNØF=8.

ØSATØF= — Index into attention table for devices that don't have an attention routine. This index will point to a BR14 instruction. At the time CTC is stopped, this value is placed into the UCBATI field. The default value is ØSATØF=0.

SNAP=YES — YES is the only permitted value. This generates code to allow snaps to be taken (see CTC Operator Interface). If omitted, the snap code will not be generated.

LIST OF CTC MODULES

*CTCAMCTL	Control Routine
*CTCAMSUP	Supervisor
*CTCAMCMD	Command Processor
CTCOPEN	Open Module
IGC020XA	Close Module
*CTCRDWRT	Read/Write Routine
*CTCCHECK	Check Routine
*CTCERASE	Erase I/O Routine
*CTCPURGE	Purge RQE Routine
*CTCCHABE	Channel End/Abnormal End Appendage
*CTCATTEN	Attention Routine

* CSECT in CTCAMCTL Load Module

CTCAMWKA - CTC WORK AREA

```
WKA       DSECT
          CTCAMWKA
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*                              C T C A M W K A
*
* THIS DSECT DESCRIBES THE CTCAM SYSTEM WORK ARAE. ONE IS OBTAINED
* BY THE CONTROL ROUTINE CTCAMCTL FOR EACH CTC DEVICE STARTED.
*
* * * * * *
*
* AUTHOR - PHIL PRESTON   3/20/72
*
* * * * * *
*
* THIS IS A CONFIDENTIAL AMOCO PRODUCTION PROGRAM
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

CTCSTART EQU   *
*
* WORK AREA CONTROL INFORMATION
*
CTCHAIN  DS    F * CHAIN FIELD TO NEXT WORK AREA
CTCTCBAD DS    F * TCB FOR THIS SUPERVISOR
CTCMDTCB DS    F * CMD PROC TCB ADDRS
CTCUCBAD DS    H = UCB ADDRS FOR THIS CTC DEVICE
CTCLENTH DS    H * LENGTH OF THIS WORKAREA
CTCSYSNO DS    F * SYSTEM ON OTHER SIDE OF ADAPTER
*
* SYSTEM SAVE AREAS
*
CTCSUPSA DS    18F * SUPERVISOR SAVE AREA
CTCMDPSA DS    18F = COMMAND PROCESSOR SAVE AREA
CTCOPNLS EQU   CTCMDPSA+12 USE SAVE AS WORK AREA
CTCATTCH EQU   CTCMDPSA+20 USE SAVE AS WORK AREA
*
CTCPGLST DS    CL16 * PURGE PARM LIST FOR CTCPURGE
*
* SYSTEM DCB'S
*
CTCDCB0  DS    CL88 * SUPERVISOR DCB
CTCDCB1  DS    CL88                 PROCESSOR DCB
*
```

```
* SYSTEM READ/WRITE TABLE PREFIX
*
RWTSTART DS    0F
RWTUSERS DS    F
RWTDMRTN DS    F
RWTWKECB DS    F
RWTATECB DS    F
RWTCPECB DS    F
RWTCURRE DS    F
RWTIOCNT DS    F * COUNTER OF UESR POSTS FROM I/O ACCROSS ADAP
RWTERCNT DS    F * ERROR CNTR.ADDED WHEN CHANNEL CHK OCCURS
*
* SYSTEM ECB LISTS
*
CTCSECBL DS    0CL16 * SUPERVISOR ECB LIST
CTCURECB DS    F * CURRENT IOB ECB POINTER
CTCSYECB DS    F * SUPV SYSTEM ECB FOR CTL TO POST
CTCATECB DS    F * ATTENTION ECB POINTER
CTCWKECB DS    F * WORK ECB POINTER
*
CTCPECBL DS    0CL16 * COMMAND PROCESSOR ECB LIST
CTCTLECB DS    F * ECB FOR CTL RTN TO POST
CTCPSECB DS    F * SYSTEM ECB FOR SUPV TO POSY
CTCPRECB DS    F * READ ECB
CTCPWECB DS    F * WRITE ECB
*
CTCSENSE DS    D * SENSE CCW
*
* COMMAND PROCESSOR DECB'S
*
CTCRDECB DS    7F * CMD PROC READ DECB
CTCWDECB DS    7F * CMD PROC WRITE DECB
*
* SYSTEM BUFFERS
*
         DS    0D
CTCIBUF  DS    0CL72 * MODIFY BUFFER
FREE     EQU   X'80' * INDICATES BUFFER FREE FOR COMMAND
CTCUCMID DS    CL2 * UCMID OF REQUESTING CONSOLE
CTCBUFLN DS    H * LENGTH OF COMMAND
CTCDATA  DS    CL70 * COMMAND
*
CTCRDBUF DS    CL72 * CMD PROC READ BUFFER
*
CTCWTBUF DS    CL72 * CMD PROC WRITE BUFFER
*
CTCWTOBF DS    CL72 * CMD PROC WTO BUFFER
WORKLNGH EQU   *-CTCSTART
*
```

CTCDMRTN - CTC DATA MANAGEMENT ROUTINE POINTERS

```
DMRTN     DSECT
          CTCDMRTN
*
* THE FOLLOWING DESCRIBES THE DATA MGT POINTER AREA OF THE RWT
*
RWTRWRTN DS    F
RWTCHECK DS    F
RWTERASE DS    F
RWTPURGE DS    F * PURGE ROUTINE ADDRESS
RWTCEAP  DS    F
RWTATTN  DS    F
```

CTCENTRY - CTC READ WRITE TABLE

```
RWT       DSECT
          CTCENTRY
*
*
* THE FOLLOWING IS THE FORMAT OF EACH USER ENTRY
*
RWTENTRY DS    0F
RWTRDECB DS    F
RWTWDECB DS    F
*
* THE BIT VALUES OF THE HIGH BYTE OF BOTH DECB POINTERS ARE :
RWTIOTAP EQU   X'80'
RWTIOGEN EQU   X'40'
RWTEXCP  EQU   X'20'
*
RWTUSECB DS    F
*
RWTSTAT1 DS    C
RWTOPENX EQU   X'80'
RWTOPENY EQU   X'40'
RWTLASTE EQU   X'20'
RWTCLOSE EQU   X'10' * CLOSE ISSUED
CTCWAKE  EQU   X'04'
*
RWTSTAT2 DS    C
RWTSTAT3 DS    C
*
RWTDDNAM DS    0C
RWTSTAT4 DS    C
ENTRYEND EQU   *
ENTRYLEN EQU   ENTRYEND-RWTENTRY
```

CTCTIOT - CTC TASK INPUT/OUTPUT TABLE

```
TIOT      DSECT
          CTCTIOT
TIOELNGH  DS    X
          DS    3X
TIOEDDNM  DS    CL8
          DS    F
TIOEFRST  DS    F
TIOTFEND  DS    F
```

CTCDEB - CTC DATA EXTENT BLOCK

```
DEB       DSECT
          CTCDEB
DEBNMSUB  DS    0X
DEBTCBAD  DS    F
DEBAMLNG  DS    0X
DEBDEBAD  DS    F
DEBOFLGS  DS    0X
DEBIRBAD  DS    F
DEBOPATB  DS    X
DEBQSCNT  DS    X
          DS    H
DEBNMEXT  DS    0X
DEBUSRPG  DS    F
DEBPRIOR  DS    0X
DEBECBAD  DS    F
DEBDEBID  DS    0X
DEBDCBAD  DS    F
DEBEXSCL  DS    0X
DEBAPPAD  DS    F
DEBDVMOD  DS    0X
DEBUCBAD  DS    F
DEBEND    EQU   *
DEBLNTH   EQU   DEBEND-DEBNMSUB
```

CTCUCB - CTC UNIT CONTROL BLOCK

```
UCB       DSECT
          CTCUCB
*********************************************************************
*
*                        - - CTC UNIT CONTROL BLOCK - -
*
* THIS DSECT DESCRIBES THE UNIT CONTROL BLOCK FOR THE CHANNEL-TO-
* CHANNEL ADAPTER.FOR FURTHER INFORMATION REGARDING THE COMMON
* SEGMENT SEE THE MVT CONTROL BLOCKS MANUAL.
*
*********************************************************************
*
*     THIS  IS  A  CONFIDENTIAL  AMOCO  PRODUCTION  PROGRAM
*
*********************************************************************


UCBCOMON DS     0CL24 * COMMON SEGMENT
SRTEJBNR DS     C * INTERNAL JOB NUMBER
SRTECHAN DS     C * ALLOCATION CHANNEL MASK
UCBID    DS     C * UCB IDENTIFIER
SRTESTAT DS     C * STATUS BYTE A
UCBCHA   DS     C * CHANNEL ADDRESS
UCBUA    DS     C * UNIT ADDRESS
UCBFL1   DS     C * FLAG BYTE 1
UCBDTI   DS     C * INDEX TO DEVICE TABLE
UCBETI   DS     C * ERROR RTN KEY
UCBSTI   DS     C * STATISTICS TABLE INDEX
UCBLCI   DS     C * CHANNEL TABLE INDEX
UCBATI   DS     C * ATTENTION TABLE INDEX
UCBWGT   DS     C * FLAGS AND MASK
UCBNAME  DS     CL3 * UNIT NAME
UCBTYP   DS     F * DEVICE TYPE
UCBLTS   DS     H * LAST RQE
UCBSNS   DS     H * SENSE INFORMATION
```

```
* CTCAM DEVICE DEPENDENT SEGMENT

UCBCTC   DS    F * ADDR OF RESIDENT USER TABLE
UCBCTCRE EQU   X'80' * CTC IN REINSTATEMENT
UCBRFRSH EQU   X'40' * OTHER SYSTEM RETURNED TO READY
UCBCTFL1 DS    C
UCBABEND EQU   X'80' * CTC HAS ABENDED.RECURSION=1
UCBPCTC  EQU   X'40'
UCBOSS   EQU   X'20'
UCBOSD   EQU   X'10'
UCBIPLWK EQU   X'08'
UCBDTFR  EQU   X'04'
UCBADAPX EQU   X'03'
UCBADAPY EQU   X'02'
UCBADAPI EQU   X'00'
*
UCBCTFL2 DS    C
UCBCEAP  EQU   X'80'
UCBINT   EQU   X'40'
UCBSUP   EQU   X'20'
UCBATTN  EQU   X'10'
UCBRQE   EQU   X'08'
UCBTIMER EQU   X'04' * TIMER CODE IN CONTROL
UCBCMD   EQU   X'02' * COMMAND PROCESSOR ACTIVE
*
UCBCTFL3 DS    C
UCBWTOR  EQU   X'80' * STOP,DOWN,RETRY WTOR OUT TO OPER
UCBFSTIM EQU   X'40' FIRST TIME FOR DATA TRANSFER
UCBCTFL4 DS    C
```

CTCIOB - CTC INPUT/OUTPUT BLOCK

```
IOB       DSECT
          CTCIOB
IOBIOFLG  DS    0X
IOBIOBA   DS    F
IOBECB    DS    F
IOBFLAG1  DS    XL1
IOBFLAG2  DS    XL1
IOBSENS0  DS    XL1
IOBSENS1  DS    XL1
IOBECBCC  DS    0X
IOBECBPT  DS    F
IOBFLAG3  DS    0X
IOBCSW    DS    2F
IOBSIOCC  DS    0X
IOBSTART  DS    F
IOBDCBPT  DS    F
IOBRESTR  DS    F
IOBINCAM  DS    H
IOBERRCT  DS    H
IOBSEEK   DS    2F
IOBCHPGM  DS    6F
IOBEND    EQU   *
IOBLNTH   EQU   IOBEND-IOBIOFLG
IOB12     EQU   IOBLNTH+IOBLNTH
```

CTCDECBD - CTC DATA EVENT CONTROL BLOCK

```
DECB      DSECT
          CTCDECBD
DECBDSCT  DSECT
          DS     OF .  ALIGNMENT
DECSDECB  DS     F  .  EVENT CONTROL BLOCK
DECTYPE   EQU    *  .  TYPE OF I/O REQUEST
DECBNXT   DS     H  .  ADDRESS OF NEXT DECB
DECLNGTH  DS     H  .  LENGTH OF KEY AND DATE
DECDCBAD  DS     A  .  ADDRESS OF DCB
DECAREA   DS     A  .  ADDRESS OF KEY DATA OR USER CP
DECIOBPT  DS     A  .  ADDRESS OF IOB
DECIOCNT  DS     F  .  SYNC COUNTER FOR I/O
```

CTCDCB - CTC DATA CONTROL BLOCK

```
         CTCDCB
*****************************************************************************
*                                                                           *
*                        - - CTC DATA CONTROL BLOCK - -                     *
*                                                                           *
* THIS DSECT DESCRIBES THE DCB FOR THE CHANNEL-TO-CHANNEL ADAPTER           *
* THE FIRST 20 BYTES ARE FOR CTC DEVICE DEPENDENT USE AND THE               *
* ACCESS METHOD INTERFACE IS FOR BSAM.                                      *
*                                                                           *
*****************************************************************************
*                                                                           *
*       THIS IS A CONFIDENTIAL AMOCO PRODUCTION PROGRAM                     *
*                                                                           *
*****************************************************************************

*                       DCB SYMBOLIC DEFINITION FOR
*                       BSAM-BPAM

IHADCB   DSECT

*                       DEVICE INTERFACES

*                   DIRECT ACCESS DEVICES

DCBRELAD DS     A
DCBKEYCN DS     AL1
DCBFDAD  DS     CL8
         ORG    *-1
DCBDVTBL DS     A
         DS     H
DCBTRBAL DS     AL2
*                       ACCESS METHOD COMMON INTERFACE

         ORG    IHADCB+16
DCBKEYLE DS     BL1
DCBDEVT  DS     08L1
DCBREL   DS     AL3
DCBBUFNO DS     08L1
DCBBUFCB DS     A
DCBBUFL  DS     H
DCBDSORG DS     BL2
DCBIOBAD DS     A
```

```
*                          FOUNDATION EXTENSION

DCBBFTEK DS     0BL1
DCBBFALN DS     0BL1
DCBHIARC DS     0BL1
DCBEODAD DS     A
DCBRECFM DS     0BL1
DCBEXLST DS     A

*                          FOUNDATION BEFORE OPEN

         ORG    IHADCB+40
DCBDDNAM DS     CL8
DCBOFLGS DS     BL1
DCBIFLG  DS     BL1
DCBMACR  DS     BL2

*                          FOUNDATION AFTER OPEN

         ORG    IHADCB+40
DCBTIOT  DS     BL2
DCBMACRF DS     BL2
DCBIFLGS DS     0BL1
DCBDEBAD DS     A
DCBREAD  DS     0A
DCBWRITE DS     0A

*                          QSAM-BSAM-BPAM COMMON INTERFACE

         ORG    IHADCB+52
DCBOPTCD DS     0BL1
DCBGEPR  DS     0A
DCBPERR  DS     0A
DCBCHECK DS     A
         ORG    IHADCB+56
DCBIOBL  DS     0BL1
DCBSYNAD DS     A
         ORG    IHADCB+60
DCBCIND1 DS     BL1
DCBCIND2 DS     BL1
DCBBLKSI DS     H
         ORG    IHADCB+64
DCBWCPO  DS     BL1
DCBWCPL  DS     BL1
DCBOFFSR DS     BL1
DCBOFFSW DS     BL1
DCBIOBA  DS     A
         ORG    IHADCB+80
DCBUSASI DS     BL1
DCBBUFOF DS     BL1
```

```
*                    BSAM-BPAM INTERFACE

          ORG    IHADCB+72 M0018
DCBNCP    DS     0BL1
DCBEOBR   DS     A
DCBEOBW   DS     A
DCBDIRCT  DS     H
DCBLRECL  DS     H
          ORG    IHADCB+84
DCBCNTRL  DS     0A
DCBNOTE   DS     0A
DCBPOINT  DS     A

*                         CTCAM DEVICE DEPENDENCY
          ORG    IHADCB
DCBUSERX  DS     H
          DS     C
DCBCTCID  DS     C
DCBECBPT  DS     F
DCBCTECB  DS     F
DCBCTRSV  DS     CL8
          ORG    IHADCB+84
DCBERASE  DS     F
```

CIB - COMMAND INPUT BUFFER

```
CIB      DSECT
         IEZCIB
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*        COMMAND INPUT BUFFER MAPPING MACRO
*
*              RELEASE 20     02-16-70
*
         DS    0D -      CIBPTR
*
CIBNEXT  DS    A -       ADDRESS OF NEXT CIB IN QUEUE (ZERO FOR LAST)
CIBVERB  DS    C -       COMMAND VERB CODE
CIBSTART EQU   X'04' -   COMMAND CODE FOR START
CIBMODFY EQU   X'44' -   COMMAND CODE FOR MODIFY
CIBSTOP  EQU   X'40' -   COMMAND CODE FOR STOP
CIBMOUNT EQU   X'0C' -   COMMAND CODE FOR MOUNT
CIBLEN   DS    FL1 -     LENGTH IN DOUBLEWORDS OF CIB INCLUDING CIBDATA
         DS    XL4 -     RESERVED FOR CSCB COMPATIBILITY
CIBTJID  DS    CL2 -     TSO TERMINAL JOB IDENTIFIER
CIBCONID DS    C -       IDENTIFIER OF CONSOLE ISSUING COMMAND
         DS    X -       RESERVED
CIBDATLN DS    H -       LENGTH IN BYTES OF DATA IN CIBDATA
CIBDATA  DS    CL8 -     DATA FROM COMMAND OPERAND
*        (LENGTH OF CIBDATA IS A MULTIPLE OF EIGHT BYTES
*        DEPENDING ON THE VALUE CONTAINED IN CIBLEN)
*              START -   FOURTH POSITIONAL PARAMETER (PARMVALUE)
*              MODIFY -  RESIDUAL OPERAND IMAGE FOLLOWING COMMA
*                        TERMINATING FIRST POSITIONAL PARAMETER
*              STOP -    NONE (CIB GENERATED ONLY TO GIVE CONSOLE ID)
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
```

CTC USER INTERFACE

Use of the Channel-to-Channel Access method is much like that of BSAM. However, only system tasks are accepted as valid users. If a task in problem program state attempts to use the CTC, he will abend at the time he issues his first READ or WRITE.

The Channel-to-Channel adapter should be viewed as any other I/O device. It must have a DCB coded within the user's program and it must reference a DD card. Each user will be assigned a user identification number from 2 to 14. This number will be used by the CTC to synchronize I/O between this user and the user with the same id on the other system.

Other aspects of BSAM are also apparent. I/O is scheduled for execution by the user through the READ and WRITE macros; and a CHECK routine has been provided to allow standard use of SYNAD exits. An additional feature has been provided in the ERASE routine. This allows a user to "un-queue" a READ or WRITE under certain conditions.

There are three ECB's for each user that the CTC system posts. The READ and WRITE ECB's are posted upon their I/O completion, and the status ECB is posted when a change in system status is encountered. Since a change may occur at any time (e.g. other system goes down), the status ECB should always be waited on.

A brief discussion follows on the use of each of these macro instructions.

OPEN

CTCOPEN (DCBNAME)

The CTC OPEN routine will open one CTC DCB at a time. Since a typical user will issue both READ's and WRITE's, the DCB's are opened for INOUT. The CTCOPEN routine gives a return code in register 15 upon completion. This code should be used to determine the status of the other system. Values for these codes are listed in Attachment I.

The DCB for the CTC should be specified as follows:

DCBNAME DCB DDNAME=CTCXXXXn,LRECL=mmm,BLKSIZE=mmm,DSØRG=PS,MACRF=(R,W)

Where:

n - Assigned user id (2 thru E)

XXXX - Any alphanumeric characters

Since the Access Method does not provide blocking, the LRECL field is for user information only. It should be the same as the blocksize.

The expanded DCB will be 88 bytes in length. In addition to typical BSAM DCB contents, OPEN routine will provide a pointer to the user's status ECB in the second word. This ECB will be posted by the CTC system when any change in status is encountered in either processor X or Y.

The DDcard format is:

//CTCUSERn DD UNIT=uuu where uuu=address of the CTC device

READ

READ DECBNAME,DI,DCBNAME,BUFNAME,{'S' / length},'S',0,0

Explanation of the coding of this macro is in the IBM Data Management Macros Manual. Data transfer between processors does not necessarily occur when the READ is issued. The function of the READ is to schedule a read request in the CTC Read/Write Table and notify processor Y of its presence. Data transfer will not occur until a WRITE is issued by Processor Y.

The READ will return a code to the user in register 15 indicating whether it was scheduled. If it was scheduled successfully, the return code indicates the status of the other system. The list of return codes from READ and WRITE can be found in Attachment I.

WRITE

WRITE DECBNAME,DI,DCBNAME,BUFNAME,{'S' / length},'S',0,0

Explanation of the coding of this macro is in the IBM Data Management Macros Manual. The function of the WRITE is to schedule a write request in the CTC Read/Write Table. If at that time a READ has been issued by processor Y, data transfer occurs and users in both processors are posted of the completions. Otherwise, the WRITE is enqueued to wait on the read from processor Y. A return code is in register 15 for the user to analyze. The same return codes are available for both READ's and WRITE's. See Attachment I for a complete list.

The DECB used by the CTC is seven words in length. The sixth word is used at the time a user's ECB is posted. A fullword binary counter value, incremented by one for each post, is placed there to enable the user to decide which DECB, the READ or the WRITE, was posted first. The seventh word is initially zeroes and remains unused by the CTC system.

Both a READ and a WRITE can be outstanding to the CTC from each user at any one time. However, an attempt at scheduling a second READ or WRITE before the first has completed will result in a return code from the READ/WRITE routine and the second request will not be scheduled.

CHECK

CHECK DECBNAME

The CTC CHECK routine provides the same service as does the BSAM CHECK. If the user has specified a SYNAD routine and the post code from the I/O completion is not 00, the SYNAD routine will be entered following standard SYNAD conventions.

The CHECK routine can only wait on one ECB. Thus, its use will vary depending on the user's system design since the system status ECB should also be waited on. That is, if the user's READ's and WRITE's are issued by a task other than the one monitoring system status, CHECK will be useful, otherwise, it is recommended that a multiple WAIT be used rather than CHECK.

ERASE

ERASE DECBNAME [,PURGE]

From time to time it may be necessary to dequeue a previously scheduled READ or WRITE. This I/O may have already been issued to the channel or it may have just been queued into the Table. The user may dequeue this request by issueing an ERASE. If the PURGE option is not used and the request has already been sent to the channel, a return code in register 15 will indicate the request cannot be satisfied. If, however, PURGE is specified, the RQE will be purged for that request.

A request to ERASE a READ which has already tapped the other system will not be serviced even if the PURGE option is specified.

CLOSE

CLOSE (DCBNAME)

This will close the DCB and cause notification to be sent to processor Y of the action. Any outstanding I/O requests are ERASE'd. If the user's program abends, abend will issue the CLOSE automatically.

DSECTS

Two macros used by the CTC modules may prove helpful to users. These are CTCDECBD which will provide an expansion of the CTC DECB, and CTCDCB for the DCB; DSECT statements must be provided for these macros.

RETURN CODES AND POST CODES TO CTC USERS

RETURN CODES

The following is a list of return codes from various CTC user modules. All return codes are right justified in register 15.

CTCOPEN

0 - DCB opened successfully - Y user already open

4 - DCB opened successfully - Y user not open

8 - CTC not available for this UCB

12 - UCB specified is not a CTC UCB

16 - Another user is already open on this UCB with the same user id.

20 - DDNAME is not in TIOT

24 - DCB already open.

READ/WRITE

0 - READ/WRITE scheduled successfully

4 - User not yet open in Y

8 - CTC to Y down

12 - Unable to schedule multiple READ/WRITE

16 - CTC in X not started on this UCB

20 - Open has not been done on this UCB

24 - Invalid DDname

28 - Can't handle zero length buffer

ERASE

0 - Successfully erased

4 - I/O already sent to channel

8 - Unable to ERASE read due to read tap

12 - Invalid DECB address

POST CODES

The following POST codes are valid in the READ or WRITE DECB's:

0 - Successful data transfer

4 - Incorrect length condition received

40 - This DECB has been ERASE'd.

The following POST codes reflect system status change and will be in the users status ECB:

8 - User in Y closed

12 - CTC in X closed

16 - CTC in Y going down

20 - CTC in Y coming up after an IPL

24 - CTC in Y coming up without an IPL

28 - User in Y now open

32 - Processor Y in STOP mode

36 - Processor Y down

All post codes are in the leftmost byte of the ECB.

OPERATOR INTERFACE

The CTC (Channel-to-Channel) is used to connect two channels together so that users in two CPU's can communicate with each other. The CTC system operates as a system task in each machine that is connected to another with a Channel-to-Channel adapter. Each CTC system task is capable of handling up to sixteen adapters and provides the necessary software so that users can establish communications between two CPU's. All of the users of the CTC are also system tasks; therefore, the CTC must be started before any of the tasks that use it.

Starting the CTC System

The CTC is started using the procedure

```
S CTC.C
```

NOTE: The modifier is not necessary, but will allow shorter type-ins.

The start procedure can be modified as follows:

R=16	Region size
S='PARMS'	Start Parameters -
	(xxx,n) where xxx is the system number,
	N is the number users allowed.
	NOTE: The system number refers to the machine system number on the other side of the adapter.

EXAMPLES:

A. S CTC.C,S='739,2,287,3'

Start a CTC to system 739 with 2 users and one to 287 with 3 users.

B. S CTC.C,S='287,1'

Start one CTC to system 287 with one user.

C. S CTC,S='(739,,287)'

Start one CTC to 739 and 287 with the default number of users.

NOTE: Parenthesis may be used as desired.

After the CTC system has been started, an individual adapter can be started by using the modify command

FC,S=XXX,N

The format of start is the same as in the Parm field.

EXAMPLES:

D. FC,S=287

Start 287 with the default users.

E. FC,S=(287,,739,5)

Start 287 with the default number of users and 739 with five.

Stopping the CTC System

The CTC system can be stopped, or an individual adapter can be stopped without affecting other adapters that are operating.

To stop the entire CTC system ---

PC (or, if started without the modifier, PCTC)

To stop a single adapter ---

FC,P=XXX where XXX is the system number.

NOTE: If this terminates the last adapter, the CTC system will terminate.

EXAMPLE:

F. FC,P=287

Stops the adapter to system 287.

Display Status

The status of the CTC system or an individual adapter can be displayed by issuing---

FC,D=A Displays all of the adapters that have been started in the system.

FC,D=XXX Displays only the adapter specified by the system XXX.

The format of the display is

CTC018I SYSTEM=XXX,UCB=UUU ,DOWN / ,STOPPED

Sending Messages

The CTC can be used to send messages between two consoles on different machines. The operator must specify the system number and the console to which the message is to be sent.

The format of the command is ---

FC,XXX,CC,MESSAGE

Where CC is ---

MC - Master Console

PA - Printer Area

TA - Tape Area

TL - Tape Library

TP - Teleprocessing Area

BD - Broadcast to all Consoles.

EXAMPLES:

G. FC,287,MC,MESSAGE TO BE SENT TO MASTER CONSOLE ON 287.

H. FC,288,BD,MESSAGE TO BE SENT TO ALL CONSOLES ON 288.

Adapter Timeout

The CTC system sends a protocol I/O to each adapter at given time intervals to determine if the other CTC is still operating. If the other system fails to respond, the operator is asked the status of the other system in the following message---

CTC099I WHAT IS THE STATUS OF SYSTEM XXX-REPLY U,S,OR D

The conditions under which these replies should be made are

U - 1) Reply U if the status of the other system is unclear. No action is taken and if the other system does not respond in another interval, the message will be repeated.

2) Reply U if the CTC in the other system is down, but will be restarted.

S - 1) Reply S only if the other system is stopped (in manual mode).

D - 1) Reply D only if the other system will be IPL'ed before the CTC system is started.

CTC MESSAGES

CTC001I INVALID START REQUESTED

Explanation: START field on parm or in modify was invalid.

Response: Enter correct parameter.

CTC002I (SSS) INVALID USER FIELD

Explanation: User field in a start command contained invalid digits.

Response: Enter correct USER field.

CTC003I (SSS) TOO MANY USERS REQUESTED

Explanation: More users were requested than could be handled. Thirteen is the maximum number allowed.

Response: Reduce number of users.

CTC004I (SSS) CTC ALREADY STARTED ON UCB XXX

Explanation: An attempt was made to start a CTC on a UCB that already had one started on it.

Response: If there is not another CTC started on this UCB, the UCB must be cleared out. An IPL will do this. (Words seven and eight must be - 00000000 08000000)

CTC005I CTC SYSTEM ERROR-CODE=X

Explanation: An error occurred in the CTC system as a result of a logic error.

- X=0 Unable to locate DDNAMES
- X=1 Maximum number of CTC exceeded

Response: Call system programmer.

CTC006I (SSS) DEVICE ON XXX NOT OPERATIONAL

Explanation: The adapter on UCB XXX is (a) switched off or (b) there is a hardware malfunction.

Response: If it is (a), turn it on and retry. If it is (b), call a CE.

CTC007I CTC SYSTEM IS STOPPING

Explanation: The system has no adapters operating and is stopping.

Response: None.

CTC008I CTC SYSTEM IS STARTING

Explanation: The CTC system will accept modifies from the console and is starting the adapters specified in the parm field.

Response: Enter desired commands.

CTC009I SYSTEM XXX IS NOT ACTIVE

Explanation: A reference was made to an adapter that was not active.

Response: Enter correct system number.

CTC010I (SSS) WTO BUFFER BUSY - TRY AGAIN

Explanation: A previous message that was to be sent to another CPU has not been sent.

Response: Retry message.

CTC013I (SSS)INVALID RESPONSE - RETRY ASSUMED

Explanation: (a) A correct response to CTC 099 was not entered after three attempts. (b) A response of D or S was entered, but the system was in run mode.

Response: None.

CTC014I SYSTEM XXX INVALID

Explanation: System XXX is not a valid system number.

Response: Enter the correct system number.

CTC017I INVALID MESSAGE

Explanation: Message format entered was invalid.

Response: Correct and retry.

CTC018I SYSTEM=SSS,UCB=UUU {,DOWN / ,STOPPED}

Explanation: Response to a display status.

Response: None.

CTC019I (SSS)CTC SUBTASK ABENDED - XXXXXX

Explanation: A subtask of the CTC abended with a completion code of xxxxxx.

Response: If the supervisor abended, that adapter will be stopped. If the command processor abended, it will be reinstated. If two abends occur, the system will abend. Notify system programmer.

CTC020I (SSS) CTC COMMAND PROCESSOR REINSTATED

Explanation: The Command Processor abended but was reinstated.

Response: Notify system programmer.

CTC099A WHAT IS THE STATUS OF SYSTEM XXX- REPLY U,S, OR D

Explanation: The system specified by XXX has not responded to protocol I/O.

Response: Reply one of the following:

U - a) Status of other system is unclear
b) CTC in other system is down, but will be restarted.

S - a) Only if the other system is in manual mode.

D - a) Only if the other system will be IPL'ed before the CTC is started again.

CTC100I SYSTEM XXX STOPPED

Explanation: The system specified by XXX is stopped.

Response: None.

CTC101I SYSTEM XXX DOWN

Explanation: The system specified by XXX is down.

Response: None.

CTC102I SYSTEM XXX CTC OPERATIONAL

Explanation: System XXX is ready to communicate with the other system.

Response: None.

CTC103I SYSTEM XXX CTC TERMINATING

Explanation: System XXX has issued a P CTC

Response: None.

CTC104I SYSTEM XXX CTC IS DEAD

Explanation: The CTC to system XXX has been closed.

Response: None.

CTC105I SYSTEM XXX CTC I/O ERROR

Explanation: I/O Error occurred.

Response: Possible hardware malfunction. See if adapter is still enabled.

CTC106I SYSTEM XXX CTC UP AFTER IPL

Explanation: The CTC is ready to communicate with the other systems.

Response: None.

CTC107I SYSTEM XXX CTC ACCEPTING CMD$

Explanation: The CTC is ready to accept commands for consoles on other machines.

Response: Modify commands for the console task will be processed.

INTER-SYSTEM SHARED ENQUE

Standard Oil Company (Indiana)
Manual # CSD0082
January 1973

CONTENTS

S H A R E D E N Q U E F U N C T I O N A L D E S C R I P T I O N

In a single system (CPU) environment, IBM's OS/MVT provides protection of serially reusable resources via the ENQ macro. Each task issuing this macro generates requests which are added to unique queues for each requested resource. The task then waits until its requests are at the top of all queues where the request was for exclusive control or, if the request was for shared control, until its requests share the top of each queue with other shared requests. Only the requested resources are affected allowing other resources (e.g. datasets on the same volume) to be accessed by other jobs.

IBM's OS/MVT implementations of this resource protection when two or more systems (CPUs) are sharing direct access devices is by use of the reserve macro. The reserve macro performs an ENQ in the originating system and when the resource is obtained, a hardware reserve of the spindle (volume) is performed; this locks out all other CPUs from accessing any data set on the reserved volume. Other tasks running in the system that issues the reserve are not, however, locked out from the volume. If any job running in other CPUs needs access to data sets on a locked out volume, they are prevented from running until the reserving CPU releases the volume.

If only the specific data sets needed were reserved, jobs in any CPU which need different datasets on the same volume could run; this is the basic concept implemented via Inter-System Shared Enque (SHRENQ).

Inter-System Shared Enque, as implemented by Standard Oil Company, is designed to replace the reserve of a complete volume with an ENQ across systems for only the resources required. This is accomplished by ENQing the same resource in both systems.

SHRENQ provides all the functions and options of single system enques, and in addition, changes certain reserves to ENQs when they are for application oriented resources. Reserves on system oriented resources such as those issued by catalog management and DADSM are not affected and remain reserves.

SHRENQ also provides an additional option, not supported in IBM single system enques, which permits converting an exclusive ENQ to a shared ENQ.

Inter-System Shared Enque consists of a Supervisor system task which controls a Reader/Writer task and a Monitor task. The Reader/Writer task interfaces with the Channel-to-Channel Access Method and reads and writes ENQ lists to and from other systems. The Monitor task orders these lists, releases tasks performing ENQs in its system to complete the ENQ function and performs the ENQ function for lists representing tasks in another system.

Core requirements for this system are about 16K bytes for the system task plus the additional amount of System Queue Space required to hold enque elements and messages to and from other systems.

DEFINITIONS AND MISCELLANEOUS INFORMATION

This chapter provides definitions for terms used throughout this manual. Also included in this chapter is some miscellaneous information on SHRENQ.

Definitions

SHRENQ Running Solo - If SHRENQ is running solo, active CPU count=0 and SHRENQ is not operational. SHRENQ is not sending and receiving ENQs or DEQs to the other CPU. Jobs in this system can continue to run without their ENQs going across to the other CPU. SHRENQ will not place jobs issuing ENQs in any wait state, although ENQ itself may do so. All software reserves cause the hardware reserve feature to be used.

SHRENQ Operational - Active CPU count not 0, and SHRENQ is not running solo. SHRENQ is sending and receiving ENQs and DEQs to the other CPU. All running jobs in this CPU pass through SHRENQ when issuing ENQs, at which time all the across-system names (see Table 1 this chapter) are sent to the other CPU. If, however, SHRENQ is in start-up mode, SHRENQ will place these jobs in a wait state until SHRENQ has started-up. This is only true when active CPU count is greater than zero.

Active CPU Count - The number of active other CPUs not including this CPU. The number of other CPUs which are receiving our ENQs and sending us their ENQs. This can be obtained with a F SHRENQ,D=S command.

CPU X, CPU Y - CPU X is this CPU. CPU Y is the other CPU. These are used to distinguish the difference between this CPU and the other CPU. Regardless of what system you are on, it is CPU X, the other CPU is Y.

Envoys - SHRENQ utilizes ENVOY control blocks which are used as phantom TCB, and SVRB for ENQs originated in the other CPU.

SYSOTHER - SHRENQ when running, has an ENQ up for "SYSOTHER C1". Any time an ENQ originating in this CPU is waiting for the ENQ to be satisfied in the other system, a QEL is placed on "SYSOTHER" QCB chain.

Q-msg - This is the message that SHRENQ sends/receives to/from the other system to invoke ENQs, DEQs, etc. Any time an ENQ, DEQ, etc. is to be sent to the other CPU (CPU Y), a Q-msg is constructed and given to CTC for read/write processing.

Across-System ENQs - Some ENQ names are sent to the other CPU and some are not. The names that do go to the other system are known "across-systems" (i.e. any ENQ name sent or received to/from other CPU is an across-system ENQ). See Table 1 for names to be known across-systems.

TABLE I

Miscellaneous Information

Table of ENQ major names to be known across-systems. These major names, when ENQ'd by some job, will be sent across-systems if SHRENQ is operational.

SYSDSN	DSTAPEA
SYSIEWL	SYSPSWRD
SYSIGGLG	
USERENQ	

The following minor names are the only exceptions for major name SYSDSN. When a major name of SYSDSN is followed by one of these minor names, it is not sent across to the other CPU:

SYSCTLG	Rnn. (RJE DD * data set)
DAP.JCB	TPD.IØTRACE
STAND.ALONE.DUMP	
T.TURNARØN	

SHRENQ REGION SIZE = 16K

Version I of SHRENQ is for two-CPU support only.

OPERATING INSTRUCTIONS

This chapter contains instructions on how to use SHRENQ, operating practices, and operator commands. To fully understand the terms used, reader should first reference chapter on miscellaneous items and definitions.

SHRENQ operates as a system task. The console operator has the capability to:

1. Start or stop SHRENQ
2. Display the status of SHRENQ
3. Invoke a Snap Dump of the SHRENQ system.

Starting SHRENQ

In order for SHRENQ to send and receive ENQs/DEQs between systems, CTC must be running. However, SHRENQ can be started without CTC running and it will wait for CTC to be started.

To start SHRENQ, type in "S SHRENQ"

This command will start a system task named SHRENQ. If CTC is not currently running in the system, message "SHRENQ21-CTC CLOSED, SHRENQ IDLE" will appear. SHRENQ will then enter a wait state running solo until CTC is started. The system may continue to operate normally if desired. Once CTC is started or if CTC is already active, message "SHRENQ03-SHRENQ STARTING" will appear. This indicates that SHRENQ is starting up but is not yet operational until message "SHRENQ06-SHRENQ READY" appears.

During the start-up period, SHRENQ attempts to establish communication with the other CPU. The next message to appear will be "SHRENQ09-SHRENQ WAITING ON CPU Y TO START SHRENQ". If CPU Y (the other CPU) does not start SHRENQ, this CPU will wait (running solo) until SHRENQ is started in CPU Y.

If SHRENQ is started in both systems, message "SHRENQ06-SHRENQ READY" should appear shortly after message "SHRENQ09---", message "SHRENQ06" indicates both CPUs have established communication, ENQs in each system have been sent to the other system and SHRENQ is now operational.

In order for SHRENQ to start-up, both CPUs can be dirty. (i.e., one or both CPUs may have ENQs which are to be known across-systems.) However, SHRENQ cannot start if both systems contain the same across-system names with disp of OLD.

It is possible for both CPUs to have many running jobs and start SHRENQ; however, they may not have old ENQs (to be known across-systems) with the same name.

If start-up is impossible because both CPUs contain the same ENQ'd names, messages will appear in both systems. Different messages will appear in each system. In one CPU the message "SHRENQ01-SHRENQ CANNOT START BECAUSE OF EXISTING ENQ,S IN OTHER CPU(S)" will appear. This indicates that both CPUs contain the same ENQ'd name and, therefore, SHRENQ cannot start. In the other CPU message "SHRENQ01-SHRENQ CANNOT START BECAUSE OF ENQ,S IN THIS CPU BY THE FOLLOWING JOB(S)"/ will appear. This message will be followed by a list of one or more jobnames that must finish, or release some resource before SHRENQ can start.

In both CPUs these messages will be followed by message (WTOR) "SHRENQ02-REPLY U WHEN CLEAN OR C TO CANCEL SHRENQ". This condition indicates both systems have identical ENQs to go to the other CPU. If SHRENQ is not to be run, reply "C" in both systems and SHRENQ will terminate. If SHRENQ is to be run, one of the systems must wait for jobnames listed to finish or cancel the jobs. Both systems will continue to run with SHRENQ as solo at this point. When the jobnames listed are terminated, reply U in both systems, and SHRENQ will restart the start-up process beginning with message "SHRENQ03-SHRENQ STARTING".

If CPU Y should go down while SHRENQ is starting up, messages "SHRENQ10---" or "SHRENQ11---" will appear and SHRENQ will wait for CPU Y to come up.

To obtain the status of SHRENQ at any time, type in: F SHRENQ,D=S. Reference chapter on SHRENQ messages for description or responding messages.

Stopping SHRENQ

SHRENQ may be stopped at any time by issuing a "P SHRENQ" command. If SHRENQ is starting up when a stop is issued, the other CPU will enter a wait state until stopped or SHRENQ is started again. If SHRENQ is operational (not in start-up mode), in CPU Y, and SHRENQ is stopped in CPU X, CPU Y will clean up all ENQs from CPU X and will enter a wait state running solo, until the other CPU starts SHRENQ.

Any time SHRENQ is stopped, message "SHRENQ20-SHRENQ FOLDS" will appear indicating SHRENQ has terminated.

Modify Commands

The modify SHRENQ commands have the following format:

F SHRENQ, [command=operand]

where:

F SHRENQ is the ØS modify command.

command=operand are valid commands used by SHRENQ.

Four types of modify commands exist.

1. Display status of SHRENQ

   F SHRENQ,D=S

   This command causes several messages to appear denoting the status of SHRENQ. Reference chapter on SHRENQ messages for further information.

2. Dump core used by SHRENQ

   F SHRENQ,D=C

3. ØPEN CTC

   F SHRENQ,

4. Display Q-Heads

   F SHRENQ,D=Q

   This command causes a HEX display of Q-Heads for the Monitor-Q, Reader/Writer-Q and Envoy-Q. This command is a debugging aid.

Commands number 2, 3, and 4 are not necessary except for debugging.

Operating Practices

It is most desirable to start SHRENQ immediately after IPL. This will insure that the system is clean and SHRENQ can start. If this is not possible, SHRENQ can be started at any time, and will be operational provided there are no identical across-system names ENQ'd in both systems.

Neither SHRENQ or CTC should ever be stopped.

S H R E N Q M E S S A G E S

This chapter contains messages issued by SHRENQ; their description, format and meaning. Each message is preceeded by "SHRENQxx" where xx is the unique message-id. The following cross-reference list is provided to associate the message-id with a particular module which issues that message.

MODULE/CSECT	MESSAGE PREFIX NO.	
SNQSTART	SHRENQ01	
	SHRENQ02	
	SHRENQ03	
	SHRENQ05	
	SHRENQ06	
	SHRENQ09	
	SHRENQ10	
	SHRENQ11	
SHRENQ/SUPERVISOR (SNQSUPER)	SHRENQ20	
	SHRENQ21	
	SHRENQ22	
SNQMODFY	SHRENQ30	SHRENQ37
	SHRENQ31	SHRENQ38
	SHRENQ32	SHRENQ39
	SHRENQ33	SHRENQ40
	SHRENQ34	SHRENQ41
	SHRENQ35	SHRENQ42
	SHRENQ36	

"SHRENQ01-SHRENQ CANNOT START BECAUSE OF EXISTING ENQ,S IN OTHER CPU(S)"

SHRENQ cannot start up because of one of the following:

1. Both CPUs contain identical across-system ENQ names with DISP=OLD
2. Asynchronous ENQs exist.
3. 1 CPU contains DISP=OLD, other CPU has DISP=SHR on across-system name(s).

Action: Wait for jobnames listed in other CPU to terminate.

"SHRENQ01-SHRENQ CANNOT START BECAUSE OF ENQ,S IN THIS CPU BY THE FOLLOWING JOB(S)"

This message will be followed by a list of one or more jobnames that are preventing SHRENQ from starting.

Action: Wait for those jobs listed to terminate or cancel the jobs.

"SHRENQ02- REPLY U WHEN CLEAN OR C TO CANCEL SHRENQ"

Always appears after "SHRENQ01" msg.

Action: 1. Wait for jobnames listed (in one CPU) to terminate and reply U to attempt restart.

2. Reply C if SHRENQ not to be run (stops SHRENQ).

"SHRENQ03 - SHRENQ STARTING:

SHRENQ is starting to come up, but not yet operational (start-up routine has been entered). This message is informational only.

"SHRENQ05 - NUCLEUS INVALID"

SHRENQ cannot run because system is not IPLed from modified nucleus which contains ENQ/DEQ Hooks.

Action: IPL from correct nucleus.

"SHRENQ06 - SHRENQ READY"

SHRENQ has completed start-up functions and is sending/receiving ENQs etc. to/from other CPU.

"SHRENQ06 - SHRENQ ENDING"

SHRENQ has been stopped by operator or was starting and SHRENQ in CPU Y stopped. SHRENQ is either going to terminate or restart.

"SHRENQ09 - SHRENQ WAITING ON CPU Y TO START SHRENQ"

Start up in process attempting to establish communication with SHRENQ in other CPU. - System is running solo (ENQs are allowed).

Action: Start SHRENQ in other CPU or ignore if SHRENQ not to be run in other CPU. If SHRENQ started in other system msg "SHRENQ06" should appear shortly. This message is informational only.

"SHRENQ10 - CPU Y DOWN, SHRENQ IDLE"

CPU Y status is down as declared by CTC.

- SHRENQ allows ENQs

"SHRENQ11 - CPU Y STOPPED, SHRENQ WAITING"

CPU Y Status is stopped as declared by CTC.

- SHRENQ not allowing ENQ. Jobs issuing ENQs to go across are placed in a wait state.
- SHRENQ waits for change in CPU Y status.

"SHRENQ20 - SHRENQ FOLDS"

SHRENQ has been stopped by operator or reply 'C' to "SHRENQ02" msg.

- SHRENQ is dead indeed

"SHRENQ20 - SHRENQ ABEND"

SHRENQ has been stopped by the operator and/or a subtask has abended in the start-up or come-down process.

"SHRENQ21 - CTC CLOSED, SHRENQ IDLE"

CTC DCB has been closed because of one of the following:

1. Our CTC went down
2. CPU Y IPL'd - we've already started up

3. CPU Y dropped (went down)

If 1 - SHRENQ does stimer waiting for CTC to restart.

In any case, SHRENQ is running solo.

"SHRENQ22 UNEXPLAINED ERROR - X"

Where X is a number 1, 2, or 3. Some error has occurred. The type of error is denoted by the number in the message.

1. No ENVOY found for DEQ received from another CPU.
2. No QEL on SYSOTHER found for a "resources acquired" message received from another CPU.
3. Invalid message type given to the Monitor.

In any case, SHRENQ will invoke a snap dump and continue to run.

"SHRENQ30 - INVALID MODIFY COMMAND"

Valid Modify Commands are:

F SHRENQ,D=C	To Snap Dump
F SHRENQ,D=S	To Display Status
F SHRENQ,	To Open CTC DCB

"SHRENQ31 - CTC OPEN"

CTC DCB is open.

"SHRENQ32 - CTC CLOSED"

CTC DCB is closed.

"SHRENQ33 - WAITING FOR CPU Y TO START SHRENQ"

SHRENQ during start-up mode doing as msg states.

"SHRENQ34 - SHRENQ INOPERATIVE"

SHRENQ not sending/receiving ENQs to/from other CPU.

"SHRENQ35 - SHRENQ OPERATIONAL"

SHRENQ is sending/receiving ENQs to/from other CPU.

"SHRENQ36 - SHRENQ FOLDING"

SHRENQ is terminating.

SHRENQ37 - SHRENQ RESTARTING"

SHRENQ is restarting itself because of one of the following:

1. CTC has gone down/came up
2. SHRENQ in Y gone down/came up
3. CPU Y gone down/came up.

"SHRENQ39 ACTIVE CPU'S=XX"

XX=# CPUs communicating with this SHRENQ

XX=0 No CPUs; we're running solo

XX>0 Means SHRENQ is operational

"SHRENQ40 - UNEXPLAINED ERR HAS OCCURRED"

Some error has occurred, SHRENQ will continue to run.

"SHRENQ41 - WAITING ON CTC TO START"

CTC DCB is closed, SHRENQ trying to open every 15 seconds. Can't open until CTC is started.

"SHRENQ42 - WAITING ON CPU Y TO START CTC"

CTC has gone down in the other CPU. SHRENQ queues up ENQs to go across until CTC in the other system is started, or a Reply Down or Stop has been given to CTC.

SHRENQ MODULES

This chapter contains module descriptions and task flow of SHRENQ. All of the modules for SHRENQ are listed below in brief form:

	MODULE	CSECT	FUNCTION
1.	SNQSUPER	SNQSUPER	Supervisor
	SNQSUPER	SNQMONTR (Via IDENTIFY)	Monitor
	SNQSUPER	SNQSUPER	ENQ-DEQ Hook Code
2.	SNQSTART	SNQSTART	SHRENQ Start-Up Routine
3.	SNQRDWTR	SNQTASK	Reader/Writer Task
4.	SNQFINSH	SNQFINSH	Come-Down Clean-Up Routine
5.	SNQMODFY	SNQMODFY	Process Modify Commands
6.	SNQTABLE	SNQTABLE	Determines major/minor names to be known across systems.

MODULE NAME: SNQSUPER

CSECT NAME: SNQSUPER (SUPERVISOR)

ENTRY FROM: System task control routine via start command.

EXIT TO: O/S when stop command is issued.

FUNCTIONS: (Main Supervisor for SHRENQ)

This routine identifies SNQMONTR, attaches SNQSTART, opens, closes CTC DCB, and waits on 1 of 3 ECB's.

- Stop/modify ECB
- CTC status ECB
- Subtask exit ECB

If stop is issued, the Supervisor posts all subtasks to exit, waits for their completion and then exits. If a modify command is issued, a link SVC is issued to SNQMODFY to process the modify command. After return from SNQMODFY, the Supervisor does a QEDIT and waits again.

If the subtask ECB is posted, this indicates a subtask has abended at which time the Supervisor restores hooks to ENQ-DEQ code within the nucleus, and performs functions as though a stop command had been issued.

If the CTC status ECB is posted, the Supervisor performs different functions depending on whether SHRENQ is in start-up mode or operational mode.

MODULE NAME: SNQSUPER (Monitor)

CSECT NAME: SNQSUPER - Identified as SNQMONTR

ENTRY FROM: Attached by SNQSTART

EXIT TO: O/S if a Stop command is issued.

FUNCTION:

Waits to be posted by the Reader/Writer (SNQRDWTR) that there is work. When the Monitor is posted, it DEQ's the Q-msg from the Monitor-Q and processes it according to type. These messages are from the other CPU and types are:

ENQ List - Monitor issues an ENQ

DEQ List - Monitor issues a DEQ

HOLD MSG - for ENQ use and test - Holds up all other msgs until the ENQ has been processed.

RET CODES -Return codes generated by the other CPU as a result of an ENQ we sent across.

RESOURCES ACQUIRED - An ENQ we sent across has been successfully ENQ'd in other CPU.

Once all the Q-msgs are processed, the monitor frees the message, scans the envoy-Q for envoys to be purged and waits for work.

MODULE NAME: SNQSUPER (ENQ/DEQ Hooks)

CSECT NAME: SNQSUPER (ENQ-DEQ Hook Code)

ENTRY FROM: ENQ-DEQ code within the nucleus. Entry points are: ENQ ENTRY, ENQ EXIT, DEQ ENTRY, DEQ EXIT, TASK SWITCH, ABEND.

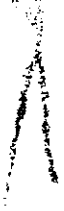
FUNCTIONS: ENQ ENTRY - Send ENQ across if not for an envoy and if it is to be known across-systems. Send Hold msg (by placing on Reader/Writer-Q) for RET=USE or CHANGE, wait for results. Construct QELs for SYSOTHER if resource is waiting on other CPU. If ENQ for envoy, replace regs with phantom TCB, SVRB pointers and exit.

ENQ EXIT - If for an envoy, exit.
If the name is to be known across systems, the "RET=" parm of the ENQ list is checked. If RET= HAVE or zero, the monitor is posted and control is returned to the nucleus.

If RET=TEST, the issuing task is placed in a wait state until the return codes are received from the other CPU. If RET=USE or CHANGE, an ENQ list is constructed for the ENQs which were successful in this CPU and the issuing task is placed in a wait until the return codes are received from the other CPU.

DEQ ENTRY - Return to nucleus if not an envoy. If envoy calling replace regs with phantom TCB,SVRB pointers.

DEQ EXIT - Exit if envoy calling. Uses code in the Monitor to construct a DEQ list message and places it on Reader/Writer-Q (if it is known in other CPU).



TASK SWITCH - Called by envoys only when a new QEL has received control (Popped to top). Uses code in Monitor to construct a "resources acquired" message to go to other CPU.

ABEND - Uses code in Monitor to construct an "abend" message to go to other CPU.

NOTE:

All messages sent to other CPU are done so by placing them on the Reader/Writer-Q.

MODULE NAME: SNQSTART (Read/Write Start-Up Routine)

CSECT NAME: SNQSTART

ENTRY FROM: Attached by SHRENQ Supervisor

EXIT TO: XCTL to SNQFINSH if SHRENQ cannot start-up.

XCTL to SNQRDWTR if SHRENQ can start up.

SVC 3 exit back to Supervisor if invalid nucleus.

FUNCTIONS:

SNQSTART performs all initialization functions for SHRENQ to start up. SNQSTART will always be in control at the same time in both CPUs. Functions include: Load SNQTABLE, initalize control blocks in common, construct and chain SYSOTHER QCBs, establish hooks with ENQ-DEQ code in the nucleus, issue hand-shake msg with other CPU and wait for other CPU to respond, determine up, down, stopped status of other CPU. The remainder of its functions are performed in non-solo mode which means no ENQs or DEQs (which are to be known as across-systems) are allowed. Any requester of these is placed in a wait state until start-up is complete.

After communication is established with other CPU, SNQSTART determines by scanning the QCB chains if this CPU is clean or dirty. "Clean" means there are no exclusive ENQs to be known across-systems. "Dirty" means there are exclusive ENQs (and/or reserves) to be known across-systems.

In order for SNQSTART to continue start-up functions, both CPUs cannot contain the same ENQ'd name. If this occurs, SNQSTART issues a WTOR indicating one system must be cleaned up and restarts itself after the operator has responded.

Each CPU receives QCBs-QELs from other CPU, chains them in this system's QCB chain and constructs envoys. Once each CPU's QCB is received, SQNSTART frees its buffers, attaches the Monitor, posts any ENQ-DEQ requesters and XCTL's to SNQRDWTR.

Each CPU that is dirty sends all eligible QCBs-QELs across to other CPU(s). The eligible (to be known as across-systems) QCBs are determined by linking to SNQTABLE. Any QELs that are sent across and are waiting for resources in this CPU are also entered on the SYSOTHER-Q.

Any exclusive reserves to be sent across are released and sent across as an ENQ System.

After all across-system QCBs have been sent across, SNQSTART frees its buffers, attaches the Monitor, posts any waiting ENQ-DEQ requesters, and XCTL's to SNQRDWTR.

During start-up, it is possible for the other CPU or CTC to go down. If this occurs, SNQSTART cleans up everything it's done, sets to run solo (allows ENQs and DEQs) and waits for other CPU to come back up.

MODULE NAME: SNQRDWTR (Reader/Writer Task)

CSECT NAME: SNQRDWTR

ENTRY FROM: XCTL'd to from SNQSTART

EXIT TO: XCTL's to SNQFINSH when a stop SHRENQ command has been issued, or if SHRENQ restarts itself (i.e. - if other CPU goes down, this CPU restarts SHRENQ).

FUNCTIONS:

SNQRDWTR issues reads and writes to CTC. It acts as an interface between the Monitor, and CTC or the other CPU. As reads/writes are processed, SNQRDWTR places the appropriate Q-msgs on the Monitor-Q, gets and frees buffers, determines transmission order and waits for work.

MODULE NAME: SNQFINSH (Read/Write Finish)

CSECT NAME: SNQFINSH

ENTRY FROM: XCTL'd to from SNQSTART if a stop SHRENQ command is issued or if SHRENQ is restarting.
Also XCTL'd to from SNQSTART if SHRENQ can't start-up or restart.
Also link to from SNQSTART.

EXIT TO: Return to caller if linked to from SNQSTART.
Normal exit if XCTL'd to which causes subtask exit.

FUNCTIONS:

SNQFINSH performs partial or total clean-up functions if link to for partial clean-up (i.e. - clean up other CPU's blocks only). SNQFINSH purges all envoys and their QELs by linking (via BALR 14,15) to code within the nucleus (ENQ-DEQ). If any QELs from this system were sent across as an ENQ "systems", meaning originally it was a reserve, the UCB reserve count is incremented in order to reinstate the reserve. Next, it frees the Monitor-Q and the Reader-Writer-Qs, frees any IRB Blocks, and returns.

If XCTL'd to for total clean-up, SNQFINSH performs all the above functions plus detaching the monitor, restoring ENQ-DEQ Hooks, DEQs and frees SYSOTHER QCBs and decrements active CPU count.

MODULE NAME: SNQMODFY (Process Modify Commands)

CSECT NAME: SNQMODFY

ENTRY FROM: Linked to by SNQSUPER (Supervisor) when a modify command has been issued.

EXIT TO: Return to SNQSUPER (Supervisor).

FUNCTIONS:

This routine processes modify SHRENQ commands to display the status of SHRENQ or to take a snap dump if requested.

MODULE NAME: SNQTABLE (Across-systems name table)

CSECT NAME: SNQTABLE

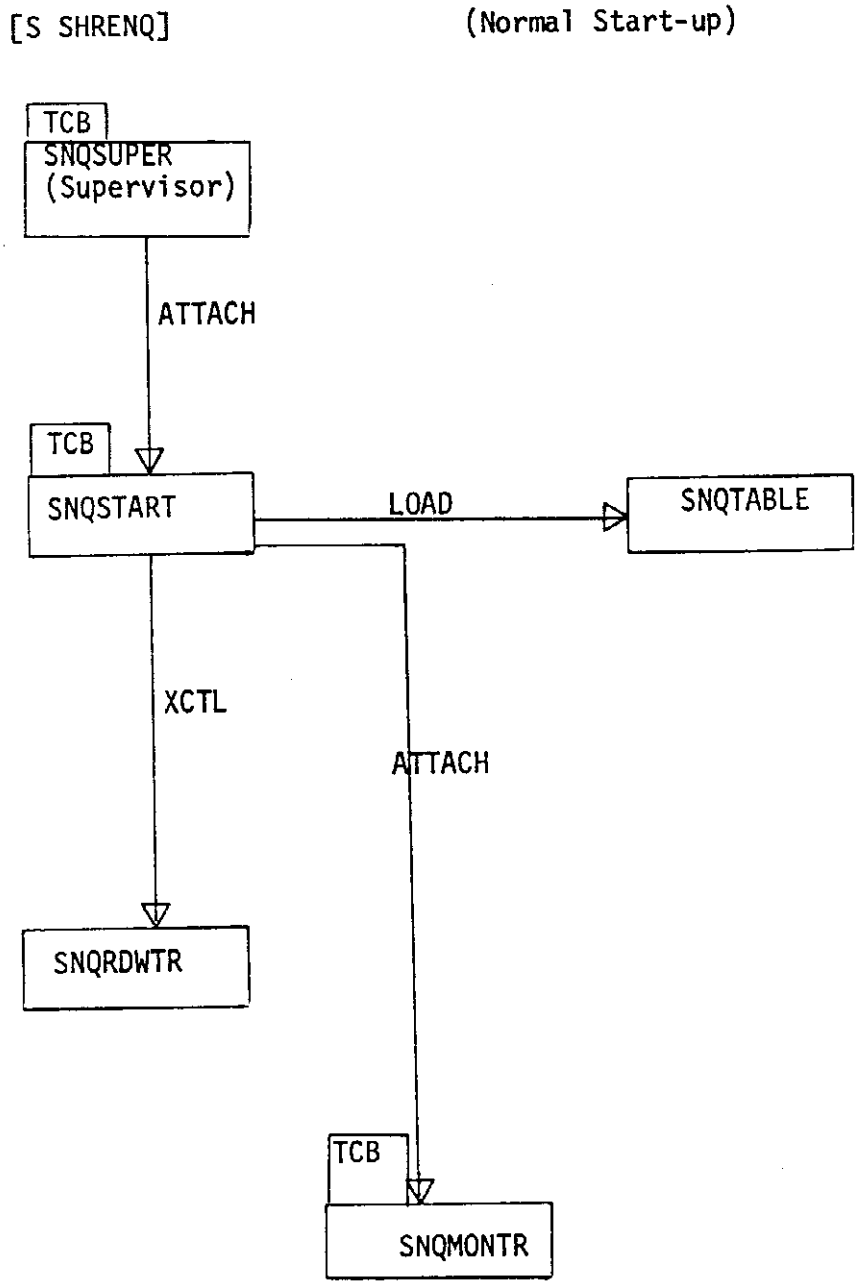
ENTRY FROM: Loaded by SNQSTART, entered via BALR 14, 15 from ENQ entry hook, DEQ entry hook, and from SNQSTART.

EXIT TO: Return to caller

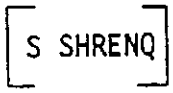
FUNCTIONS:

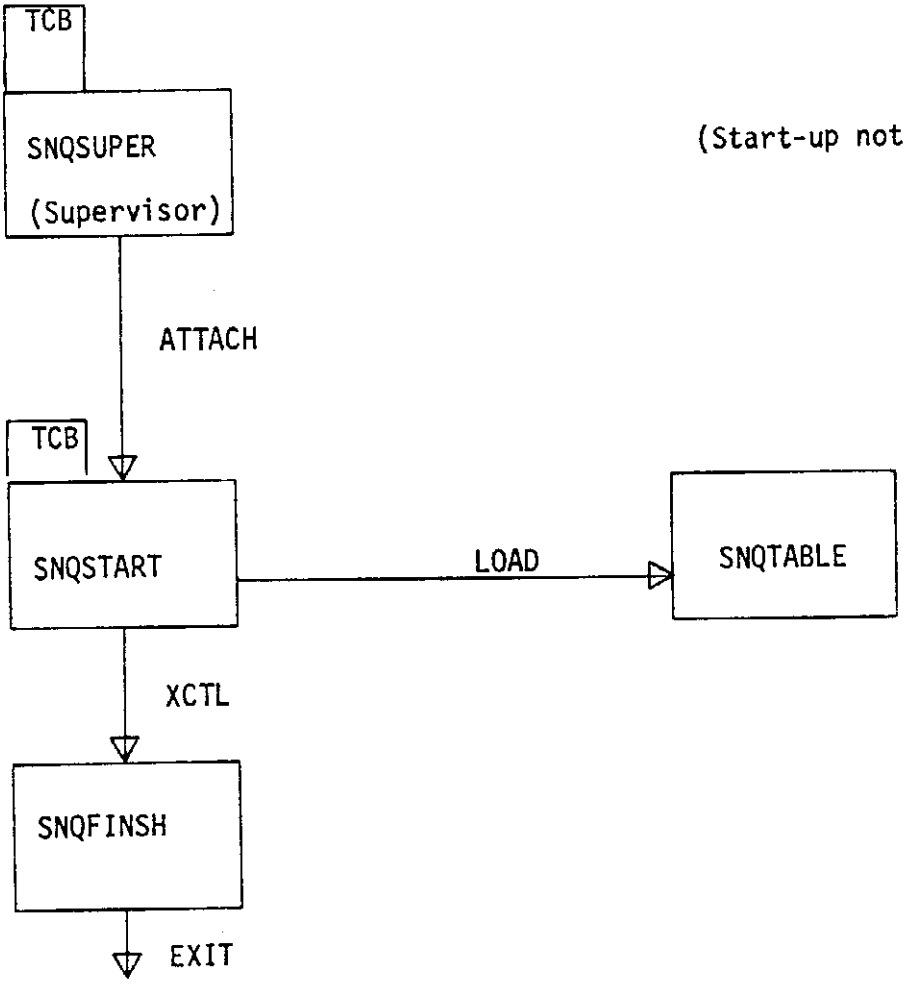
This routine scans the major/minor name table (located in this CSECT) to see if the major/minor name passed by the caller is to be known across-systems. This CSECT contains a list of major names to be known across-systems, and for each major name, a list of minor names to include and/or exclude.

"SHRENQ TASK FLOW"



Task Flow during normal start-up. SHRENQ (Supervisor) attaches SNQSTART. SNQSTART loads SNQTABLE, attaches the monitor which was identified by the Supervisor, XCTL's to SNQRDWTR.





Task Flow when start-up is impossible because operator says to cancel or a "P SHRENQ" was issued, or CTC goes down.

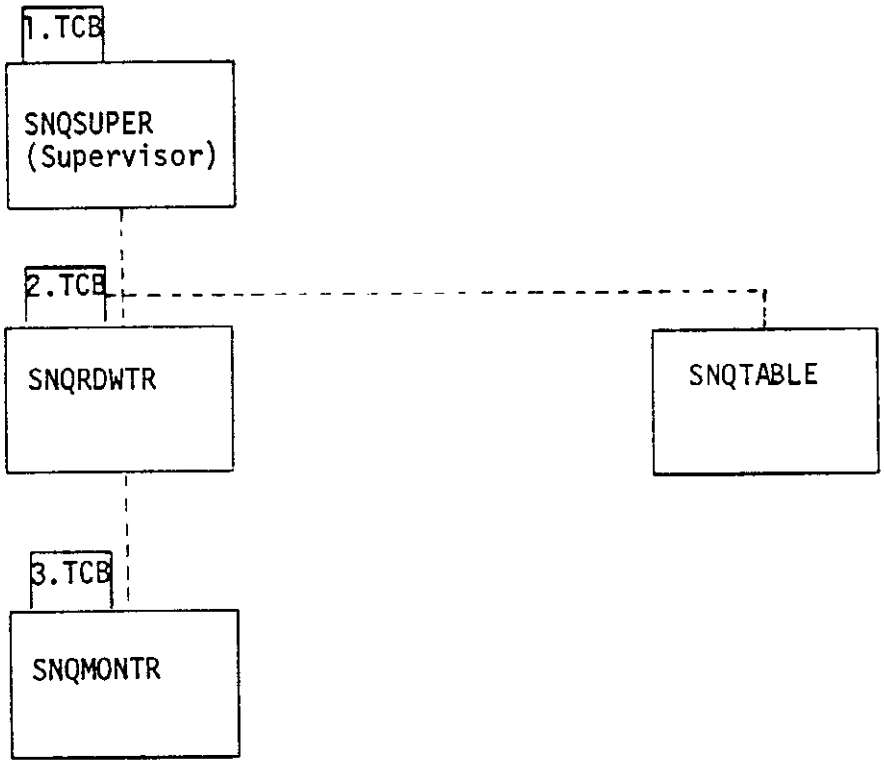
SHRENQ (Supervisor) attaches SNWSTART, SNQSTART Loads SNQTABLE, XCTL's to SNQFINSH.

SNQFINSH cleans-up and exits.

SHRENQ (Supervisor) either restarts or exits.

S SHRENQ

(Task structure after start-up complete)

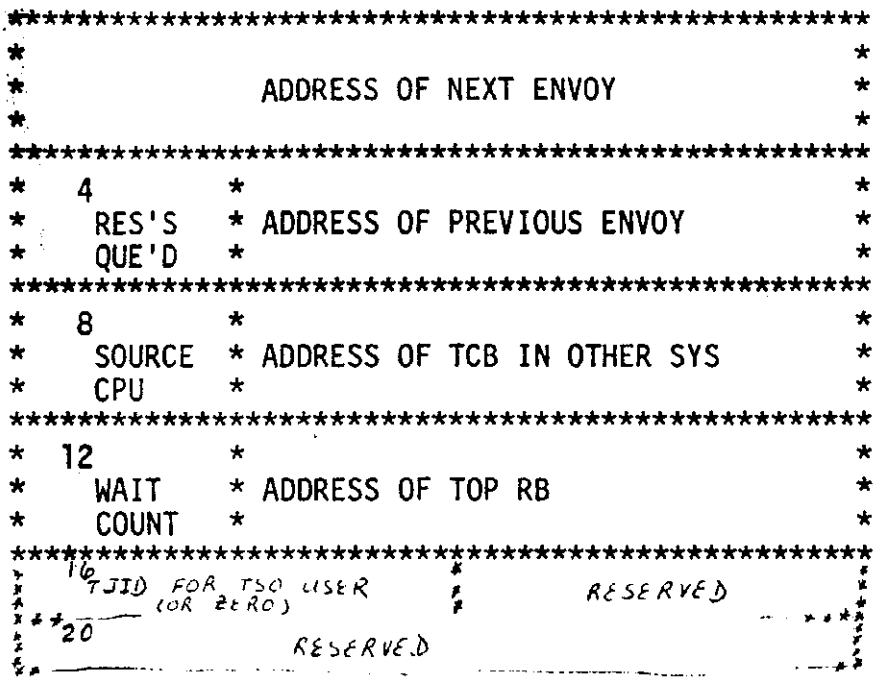


After start-up is complete and SHRENQ is operational, 3 TCB's exist.

1. SNQSUPER - Load module contains Supervisor, Hook code, and Monitor.
2. SNQRDWTR - Performs I/O with CTC.
3. SNQMONTR - Code is in SNQSUPER Load Module.

Envoy Control Block

This Control Block is constructed and used by SHRENQ. There is one envoy for each ENQ received from the other CPU. They are chained together in SQS, the first envoy pointer is located in SHRENQ common data area which is part of load module SHRENQ. The QEL for the other CPU's ENQ points to its envoy for a phantom TCB and SVRB address.




```
ENVFWD    DS      F                    NEXT ENVOY ON LIST
ENVRESCT  DS      OX                 COUNT OF RESOURCES ENQUED ON
*                                       BY THIS ENVOY
ENVBCK    DS      F                    PREVIOUS ENVOY ON LIST
ENVRLTCB  DS      F                  ADDRESS IN OTHER SYSTEM OF TCB
*                                       REPRESENTED BY THIS ENVOY
*                                    HIGH BYTE INDICATES WHICH CPU
ENVWTCT   DS      OX                 NUMBER OF RESOURCES ON WHICH THIS
*                                       ENVOY IS STILL WAITING
ENVTOPRB  DS      F                    TOP RB (RB IN ENVOY UNLESS
*                                       ASYNCHRONOUS ENTRY OCCURS)
ENVLL     EQU     *-ENVOY            LENGTH OF ENVOY CONTROL BLOCK


*****************************************
*                                       *
*          ENVLL MUST REMAIN            *
*           A MULTIPLE OF 8             *
*                                       *
*****************************************

ENVTJID   DS      H                    TJID FOR TSO USER (OR ZERO)
ENVRSRVD  DS      6X                   RESERVED
```

Q-MESSAGES SENT ACROSS CTC BY SHRENQ

SHRENQ sends and receives 14 different types of messages via CTC. This chapter describes the different types of messages and how they are used.

Messages one through six are constructed and read or written by the SHRENQ Start-Up Routine (SNQSTART). The only time these messages are used is during start-up. For further details see QMSSG DSECT description.

1. HAND-SHAKE MESSAGE - Denoted by X'FF' in high-order byte of message. Length is 8 bytes.

This message is used to establish communication with the other CPU. One is read and written from each CPU.

2. ENQs WERE SUCCESSFUL - Denoted by X'28' in Q-msg type field. Length is 8 bytes.

This message is used to tell the other CPU that this CPU has successfully established across-system ENQs sent by the other CPU.

3. ENQs NOT SUCCESSFUL - Denoted by X'2C' in Q-msg type field. Length is 8 bytes.

This message is used to tell the other CPU that this CPU could not ENQ the across-system names sent by the other CPU. This would occur if asynchronous ENQs exist, or if an ENQ with DISP=OLD that was sent across already existed in this system.

4. QCB CHAIN MESSAGE - Denoted by X'30' in Q-msg type field. Length is 8 bytes.

During start-up, each system must send across some existing ENQs and reserves. This is done with a QCB chain message. The contents of the message are the actual major, minor QCBs and QELs.

5. ASYNCHRONOUS ENQ MSG - Denoted by X'34' in Q-msg type field. Length is 8 bytes.

This message is used if a CPU contains asynchronous ENQs. It is used to tell the other system that asynchronous ENQs exist and, therefore, SHRENQ cannot come up.

6. END-OF-COMMING-UP - Denoted by X'38' in Q-msg type field. Length is 8 bytes.

This is used to tell the other CPU that this CPU has finished sending its ENQs.

NOTE: When "ENQs" are used in the above 6 message types, it actually means the major, minor QCB and QEL chains. In the remaining message type explanations, "ENQs" refer to an ENQ, DEQ, Reserve parameter list as constructed by an ENQ macro, and not the QCBs themselves. The only time QCB,QEL control blocks are sent across is during start-up.

Messages 7 through 14 are used by SHRENQ after it is up and running in both systems. These messages are constructed by the Monitor, ENQ-DEQ Hook code (which uses the Monitor code) or the Reader/Writer task. If the message is constructed by the Monitor or the ENQ-DEQ Hook code, it is placed on the Reader/Writer Queue to be read or written. In other words, all these message types are read or written to CTC by the Reader/Writer subtask (SNQRDWTR).

7. REQUEST FOR LARGE BUFFER - Denoted by X'00' in Q-msg type field. Length is 8 bytes.

If an ENQ or DEQ list exceeds 96 bytes, the message is sent across first to tell the other CPU a larger message is coming. The longer length is in the first half-word QMID (see DSECT of QMSSG).

8. ENQ LIST - Denoted by X'04' in Q-msg type field. Minimum length is 96 bytes but could be larger depending on the size of the ENQ list.

This message contains an ENQ parm list to be ENQ'd in the receiving system. When a task in this CPU issues an ENQ, the ENQ SVC branches to the SHRENQ "ENQ Entry" Hook code located in SNQSUPER module. If the major-minor names are to be known across systems, the Hook code constructs an ENQ parm list Q-msg and places it on the Reader/Writer Queue to be sent to the other system. When the other CPU receives the ENQ msg, the Reader Writer places in on the Monitor-Q to be ENQ'd in that CPU.

9. DEQ LIST - Denoted by X'08' in Q-msg type field. Length is same as ENQ list.

This message contains a DEQ parm list to be DEQ'd in the receiving CPU. The flow of operation is the same as that of ENQ except that the DEQ

message is contructed at "DEQ EXIT" hook.

10. HOLD MESSAGE - Denoted by X'0C' in Q-mgs type field. Length is 16 bytes.

When a task issues an ENQ (RET=USE or CHANGE), a "Hold msg" is sent first to tell the other CPU's monitor to hold all processing until another message with the same TCB address and TJID is received. After the hold message is sent, the ENQing task completes the ENQ attempt in its CPU, then the actual ENQ list is sent.

11. ABEND OCCURRED FOR THIS TCB - Denoted by X'10' in Q-msg type field. Length is 16 bytes.

When a task abends, the "Abend Hook" code is entered and this type message is constructed and placed on the Reader/Writer Q to be sent to the other system. When the other system receives it, the other system purges all resources belonging to that TCB, and TJID.

12. RETURN CODES - Denoted by X'14' in Q-msg type field. Length is variable.

When a task issues an ENQ (RET=TEST) the task is placed in a wait until return codes are received from the other system. This type message contains the return codes.

13. RESOURCES ACQUIRED - Denoted by X'18' in Q-msg type field. Length is 16 bytes.

When a task issues an ENQ (RET=HAVE, or WAIT), the ENQ message is sent across. The task is then placed in a wait state by placing a QEL on "SYSOTHER" chain. When the other CPU has ENQ'd all the resources in the list, it sends back a resources acquired message to inform the originating CPU of its success. The "SYSOTHER" QEL is then DEQ'd.

14. NULL MSSG FOR SEQUENCING - Denoted by X'1C' in Q-msg type field.
Length is 8 16 bytes.

This message type is currently not being used.

SHRENQ Proclib Procedures

The following JCL statements must be installed in SYS1.PROCLIB with IEBUPDTE. This procedure is used for SHRENQ system task and must be named SHRENQ.

```
// PROC   p=SNQSUPER,R=16,CTC=501
//  EXEC  PGM=&p,REGION=&R.K
//STEPLIB DD DSN=TSPC.SHRNQLOD,DISP=SHR
//   DD  DSN=T.CTCAM.LOAD,DISP=SHR
//CTCUSER2  DD  UNIT=&CTC
//SNAPDD  DD  SYSOUT=A,SPACE=(TRK,(1,200))
//SYSABEND  DD  SYSOUT=A,SPACE=(TRK,(1,200))
```

Symbolic parms include:

&P	which is for testing only, Default is SNQSUPER
&R	which is Region Size, Default is 16K
&CTC	which is unit address of the channel-to-channel adapter. Default should be the address for the system in which this procedure resides.

DD Cards Include:

STEPLIB	Must define the load data sets for SHRENQ Load Library and CTC Load Library.
CTCUSER2	Must define CTC unit
SNAPDD	Used for Snap Dumps
SYSABEND	Used for dumps.

ATTACHMENT I

RETURN CODES AND POST CODES TO CTC USERS

RETURN CODES

The following is a list of return codes from various CTC user modules. All return codes are right justified in register 15.

CTCOPEN

0 - DCB opened successfully - Y user already open

4 - DCB opened successfully - Y user not open

8 - CTC not available for this UCB

12 - UCB specified is not a CTC UCB

16 - Another user is already open on this UCB with the same user id

20 - DDNAME is not in TIOT

24 - DCB already open.

READ/WRITE

0 - READ/WRITE scheduled successfully

4 - User not yet open in Y

8 - CTC to Y down

12 - Unable to schedule multiple READ/WRITE

16 - CTC in X not started on this UCB

20 - Open has not been done on this UCB

24 - Invalid DDname

28 - Can't handle zero length buffer

Computer Center Operations
Work Procedure

Work Section: Computer Operations

Subject: CTC and SHRENQ

Background: Channel-to-channel (CTC) is an access method which provides capabilities for Inter-System CPU Communication. Shared ENQUEUE (SHRENQ) interfaces with CTC and is designed to replace the reserve of a complete volume with an ENQ across systems for only the data sets required. CTC and SHRENQ are to be active during all periods of production processing on the two systems designated for Commercial job execution. Additional information about CTC and SHRENQ is documented in Standard Oil Company (Indiana) Manuals CSD0082 and CSD0084.

Purpose and Scope: These procedures provide instructions to start and stop CTC and SHRENQ, and they explain recovery procedures for those times when problems are encountered with either CTC or SHRENQ.

A. Starting CTC:

Master Console Operator (System X)	1. Starts CTC. 2. Receives message 'CTC008I CTC SYSTEM IS STARTING'. 3. Notifies System Y to start CTC.
Master Console Operator (System Y)	1. Starts CTC. 2. Receives messages 'CTC008I, CTC SYSTEM IS STARTING', 'CTC102I SYSTEM X CTC OPERATIONAL', and 'CTC107I SYSTEM X CTC ACCEPTING CMD'.
Master Console Operator (System X)	1. Receives messages 'CTC102I SYSTEM Y CTC OPERATIONAL' and 'CTC107I SYSTEM Y CTC ACCEPTING CMD'.
Master Console Operators	1. Acknowledge above messages and determine which system will start SHRENQ first.

B. Starting SHRENQ:

Master Console Operator (System X)	1. Starts SHRENQ.
	2. Receives messages 'SHRENQ03 SHRENQ STARTING' and 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ'.
	3. Acknowledges the above messages and notifies System Y to start SHRENQ.
Master Console Operator (System Y)	1. Starts SHRENQ.
	2. Receives messages 'SHRENQ03 SHRENQ STARTING', 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ', and 'SHRENQ06 SHRENQ READY'.
Master Console Operator (System X)	1. Receives message 'SHRENQ06 SHRENQ READY'.
Master Console Operators	1. Acknowledge message 'SHRENQ06 SHRENQ READY' and proceed with normal system activity.

C. Stopping SHRENQ (both systems):

Master Console Operator (System X)	1. Notifies System Y of intentions.
	2. Stops SHRENQ.
	3. Receives message 'SHRENQ20 SHRENQ FOLDS'.
Master Console Operator (System Y)	1. Receives messages 'SHRENQ03 SHRENQ STARTING', and 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ'.
	2. Stops SHRENQ.
	3. Receives messages 'SHRENQ06 SHRENQ ENDING' and 'SHRENQ20 SHRENQ FOLDS'.
Master Console Operators	1. Acknowledge the above messages and determine which system will stop CTC first.

D. Stopping CTC (both systems):

Master Console Operator (System X)	1. Notifies System Y of intentions.
	2. Stops CTC.
	3. Receives messages 'CTC104I System Y CTC IS DEAD' and 'CTC007I CTC SYSTEM IS STOPPING'.
Master Console Operator (System Y)	1. Receives message 'CTC103I SYSTEM X CTC TERMINATING'.
	2. Stops CTC.
	3. Receives messages 'CTC104I SYSTEM X CTC IS DEAD' and 'CTC007I CTC SYSTEM IS STOPPING'.
Master Console Operators	1. Acknowledge above messages and inform the Lead Operator that CTC and SHRENQ are not active.
Lead Operator	1. Insures that Commercial jobs are not being processed in both systems concurrently.

E. Stopping SHRENQ and CTC (System X only for IPL, IDLE, etc.):

Master Console Operator (System X)	1. Notifies System Y of intentions.
	2. Stops SHRENQ.
	3. Receives messages 'SHRENQ06 SHRENQ ENDING' and 'SHRENQ20 SHRENQ FOLDS'.
Master Console Operator (System Y)	1. Receives messages 'SHRENQ03 SHRENQ STARTING', and 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ'.
Master Console Operators	1. Acknowledge the above messages.
Master Console Operator (System X)	1. Notifies System Y of intentions to stop CTC.
	2. Stops CTC.
	3. Receives messages 'CTC103I SYSTEM Y CTC IS DEAD' and 'CTC007I CTC SYSTEM IS STOPPING'.

	4. Completes the task for which SHRENQ and CTC were stopped.
Master Console Operator (System Y)	1. Receives messages 'CTC103I SYSTEM X CTC TERMINATING'.
	2. Acknowledges the message and resumes normal processing activity.

F. Invalid stops of CTC or SHRENQ:

1. CTC stopped on System X, while SHRENQ remains active on System X, and both CTC and SHRENQ remain active on System Y:

Master Console Operator (System X)	1. Starts CTC.
	2. Receives messages 'CTC008I CTC SYSTEM IS STARTING', 'CTC102I SYSTEM Y CTC OPERATIONAL', 'SHRENQ03 SHRENQ STARTING', and 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ'.
Master Console Operator (System Y)	1. Receives messages 'CTC102I SYSTEM X CTC OPERATIONAL' and 'CTC107I SYSTEM X CTC ACCEPTING CMD'.
	2. If message 'CTC099I WHAT IS THE STATUS OF SYSTEM XXX-REPLY U, S, OR D' is received, replies U.
Master Console Operator (System X)	1. Stops SHRENQ.
	2. Receives messages 'SHRENQ06 SHRENQ ENDING' and 'SHRENQ20 SHRENQ FOLDS'.
Master Console Operator (System Y)	1. Receives messages 'SHRENQ06 SHRENQ ENDING', 'SHRENQ03 SHRENQ STARTING', and 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ'.
Master Console Operator (System X)	1. Starts SHRENQ.
	2. Receives messages 'SHRENQ03 SHRENQ STARTING', 'SHRENQ09 SHRENQ WAITING ON CPU 'Y' TO START SHRENQ', and 'SHRENQ06 SHRENQ READY'.
Master Console Operator (System Y)	1. Receives message 'SHRENQ06 SHRENQ READY'.

Master Console Operators	1. Acknowledge the above messages and resume processing.

2. CTC and SHRENQ stopped on System X while CTC and SHRENQ remain active on System Y:

Master Console Operator (System Y)	1. Stops SHRENQ.
	2. Receives messages 'SHRENQ06 SHRENQ ENDING' and 'SHRENQ20 SHRENQ FOLDS'.
Master Console Operator (System X)	1. Starts CTC.
	2. Receives messages 'CTC008I CTC SYSTEM IS STARTING' and 'CTC102I SYSTEM Y CTC OPERATIONAL'.
Master Console Operator (System Y)	1. Receives messages 'CTC102I SYSTEM X CTC OPERATIONAL' and 'CTC107I SYSTEM X ACCEPTING CMD'.
Master Console Operators	1. Acknowledge the above messages and determine the system to start SHRENQ first.
	2. Follow the instructions outlined in B, above.

G. Abnormal termination of CTC (System X) with SHRENQ active:

Master Console Operator (System X)	1. Stops SHRENQ.
	2. Notifies System Y of problem.
Master Console Operator (System Y)	1. Stops SHRENQ.
	2. Stops CTC.
Master Console Operators	1. Starts CTC and SHRENQ as outlined in A and B, above.
	NOTE: If SHRENQ will not stop on a system on which CTC has terminated, that system must be IPL'ed.

H. Abnormal termination of CTC (System X) with SHRENQ not active:

Master Console Operator (System X)	1. Notifies System Y of the problem.

Master Console Operator (System Y)	1. Stops SHRENQ.
	2. Stops CTC.
Master Console Operators	1. Start CTC and SHRENQ as outlined in A and B, above.

I. Abnormal termination of SHRENQ (System X):

Master Console Operator (System X)	1. Notifies System Y of the problem.
	2. Follows the necessary procedures for the shutdown and IPL of the system.
Master Console Operator (System Y)	1. Stops SHRENQ.
Master Console Operator (System X)	1. Notifies System Y upon receipt of messages 'SHRENQ03 SHRENQ STARTING' and 'SHRENQ09 SHRENQ WAITING on CPU 'Y' TO START SHRENQ' (NOTE: AUTOIPL with start CTC and SHRENQ).
Master Console Operator (System Y)	1. Starts SHRENQ.
Master Console Operators	1. Acknowledge message 'SHRENQ06 SHRENQ READY' and resume normal processing.

J. SHRENQ unexplained error:

Master Console Operators	1. Stop SHRENQ as outlined in C, above.
	2. Start SHRENQ as outlined in B, above.

K. CTC cannot communicate - System Y status uncertain:

Master Console Operator (System X)	1. Receives message 'CTC099A WHAT IS THE STATUS OF SYSTEM XXX - REPLY U, S, OR D'.
Master Console Operators	1. Determine reason for receiving the message, attemp to correct.
Master Console Operator (System X)	1. Replies U to each occurrence of message 'CTC099A' for 10 minutes.
	a. After 10 minutes, replies D to the message.

	2. Stops SHRENQ.
	3. Stops CTC.
	4. Informs System Y of the above actions.
Master Console Operator (System Y)	1. If the System is recovered, receives message 'CTC099A WHAT IS THE STATUS OF SYSTEM XXX - REPLY U, S, OR D'.
	2. Replies D to the message.
	3. Stops SHRENQ.
	4. Stops CTC.
Master Console Operators	1. Start CTC and SHRENQ as outlined in A and B, above.
Master Console Operator (System Y)	1. If System is not recovered, message 'CTC099A' will be received (AUTOIPL will have started CTC and SHRENQ.).
	2. Replies D to message.
	3. Informs System X of status.
Master Console Operator (System X)	1. Starts CTC and SHRENQ.
Master Console Operators	1. Acknowledge all CTC or SHRENQ messages to obtain successful communication and resume processing.

L. Special instructions:

Master Console Operators	1. Hold the Queue whenever any CTC or SHRENQ problems are encountered.
	2. Record all stoppages or terminations of CTC or SHRENQ immediately in the System Incident Log.
	3. Accumulate documentation and support dumps for all problems and route them to the Center Support/Center Maintenance Group.

****