precisely

Syncsort Optimize IDMS

(EZ-IDMS)

Syncsort IDMS Synchro Cross MVS User Guide

Release 5.4

Version 1.0

May 2002

Syncsort IDMS Synchro Cross MVS User Guide
(formerly EZ-Synchro CrossMVS User Guide)

NOTICE

Syncsort Optimize IDMS, formerly called EZ-IDMS is designed to be used by licensed users of Broadcom IDMS for z/OS, as a supplement to this product.

PREFACE:

EZ-Synchro CrossMVS Option is an extension to EZ-Synchro which permits database buffer synchronization to take place between CA-IDMS jobs executing on different physical processors.

Standard EZ-Synchro achieves buffer synchronization between CA-IDMS jobs running on the same MVS system by using cross-memory services. CrossMVS option achieves buffer synchronization between CA-IDMS jobs running on different MVS systems using LU6.2 (APPC).

CrossMVS option permits users even greater flexibility in designing multi-central version CA-IDMS environments by allowing exploitation of IBM's PR/SM or Amdahl's MDF architectures, or multi-CPU environments.

CrossMVS can even be used in a Sysplex environment as an alternative to the CA-IDMS Release 14.0 Sysplex support, which relies upon the Sysplex Coupling facility to provide buffer synchronization.

CrossMVS option operates under CA-IDMS/DB release 10.1 and higher, in the IBM MVS operating environment.

RELATED DOCUMENTATION:

EZ-DB Overview
EZ-DB Integrated Installation Guide
EZ-DB Information and Error Messages

EZ-Image User Guide
EZ-Synchro® and **EZ-Share**® User Guide
DB-Megabuf User Guide
DB-Buffer Plus User Guide and Reference Guide
DB-Allocate User Guide
DB-Priority User Guide
EZ-Reorg User Guide

TABLE OF CONTENTS

FIGURES

Introduction

This manual describes the operation of the EZ-Synchro CrossMVS option from Cogito Limited. It should be read by the CA-IDMS Database Administrator and by the installer of EZ-Synchro.

This manual is intended as an addendum to the EZ-Synchro User's guide. It is assumed that the reader of this manual has already installed a base EZ-Synchro system, and is familiar with the EZ-Synchro installation procedures and EZ-Synchro operation.

To implement EZ-Synchro CrossMVS option, you must be installed at EZ-Synchro release 2.1 or higher.

The structure of this manual is as follows:-

Section 1 provides an overview of CrossMVS option.

Section 2 provides details of the CrossMVS environment.

This section describes the CrossMVS environment in detail, and provides the information necessary to generate the CrossMVS EZ-Synchro components.

Section 3 describes CrossMVS Operation.

This section describes how to start the CrossMVS facility, and provides information on the available operator commands and online display.

Section 4 describes CrossMVS information and error messages.

Appendix contains sample VTAM definition statements, as well as an overview of the CrossMVS control structures.

New Features

Many-to-Many Connections

Release 5.2 of the CrossMVS option introduces the ability to implement multiple or *many-to-many* sessions between EZ-Synchro SharedBuffer Groups. In prior releases, a particular SharedBuffer Group could only participate in a CrossMVS connection with one other SharedBuffer Group. Although any SharedBuffer Group could contain up to 32 members, this one-to-one group limitation did restrict certain configuration possibilities. For example, it was not possible in a three CPU configuration (say CPUA, CPUB, CPUC) to execute an update CA-IDMS CV on CPUA, synchronizing with retrieval CA-IDMS CVs on CPUB and CPUC.

In the new release, this, and still more flexible configurations are possible.

Overview

Where databases are shared between multiple CA-IDMS jobs running on the same machine, EZ-Synchro provides the answer to the inherent buffer synchronization problem.

With EZ-Synchro CrossMVS option, CA-IDMS databases may be shared between CA-IDMS jobs running on physically separate machines, with the same buffer synchronization provided in the standard EZ-Synchro.

EZ-Synchro introduced the SharedBuffer Group concept, where a SharedBuffer Group consists of up to thirty-two central version or local mode jobs executing on the same mainframe. Buffer synchronization between the group members is implemented using MVS cross-memory services.

With CrossMVS option, the user can define for any SharedBuffer Group, one or more "partner" group(s), each also consisting of up to thirty-two members. Each partner group may execute on a the same or a physically different CPU.

Synchronization requests are sent to the partner group(s) using APPC (LU6.2). A RECEIVE task acting on behalf of each partner group then uses MVS cross-memory services to synchronize the buffers of the applicable group members executing within the partner group.

CrossMVS option has been defined to achieve maximum performance with minimal overhead. The installation of CrossMVS option will have **no measurable overhead** on the execution of the CA-IDMS jobs using the facility.

Features

- CrossMVS Option permits EZ-Synchro SharedBuffer groups to have partner groups executing on physically different mainframes.

- CrossMVS Option uses APPC (LU6.2) for communication between EZ-Synchro SharedBuffer Groups.

- CrossMVS Option permits synchronization requests to be sent in either direction between the partner EZ-Synchro SharedBuffer Groups.

- CrossMVS Option runs in its own address space and has no impact on the performance of the CA-IDMS address spaces.

- CrossMVS Option has an online display that provides detailed information allowing fine tuning of the installation options.

- CrossMVS Option permits users even greater flexibility in the exploitation of multiple CPU configurations.

EZ-Synchro CrossMVS option operates under CA-IDMS release 10.1 and higher in the MVS operating environment. Users of CrossMVS option must be at EZ-Synchro release 2.1 or higher. Users of CrossMVS release 5.2 (including many-to-many support) must also be on EZ-Synchro release 5.2.

EZ-Synchro CrossMVS Environment.

A EZ-Synchro SharedBuffer Group consists of up to 32 CA-IDMS jobs executing on one CPU which maintain buffer synchronization through MVS cross-memory services.

When EZ-Synchro CrossMVS option is installed, a EZ-Synchro SharedBuffer Group may maintain buffer synchronization with one or more *partner* EZ-Synchro SharedBuffer Group(s), executing on the **same or different** mainframe, through APPC (LU6.2). Each unique connection between a particular SharedBuffer Group to another particular SharedBuffer Group is referred to as a CrossMVS ***Session***. Effective from release 5.2, a particular SharedBuffer Group may have multiple Sessions active simultaneously. The maximum number of sessions which may be defined for a particular Group is 31, that is to say, an IDMS CV which belongs to a SharedBuffer Group on a particular CPU could have simultaneous sessions with up to 31 other SharedBuffer Groups on the same or different CPUs.

The transfer of synchronization requests from one SharedBuffer Group to another is performed by a separate CrossMVS address space asynchronously with the processing of the CA-IDMS job. This ensures that there is no overhead nor performance degradation in the CA-IDMS address space when using this facility.

Figure 2.1 illustrates a multiple CPU environment, where an Update CV executing on CPUA uses CrossMVS to synchronise the buffers of Read CV's executing on CPUB and CPUC respectively.

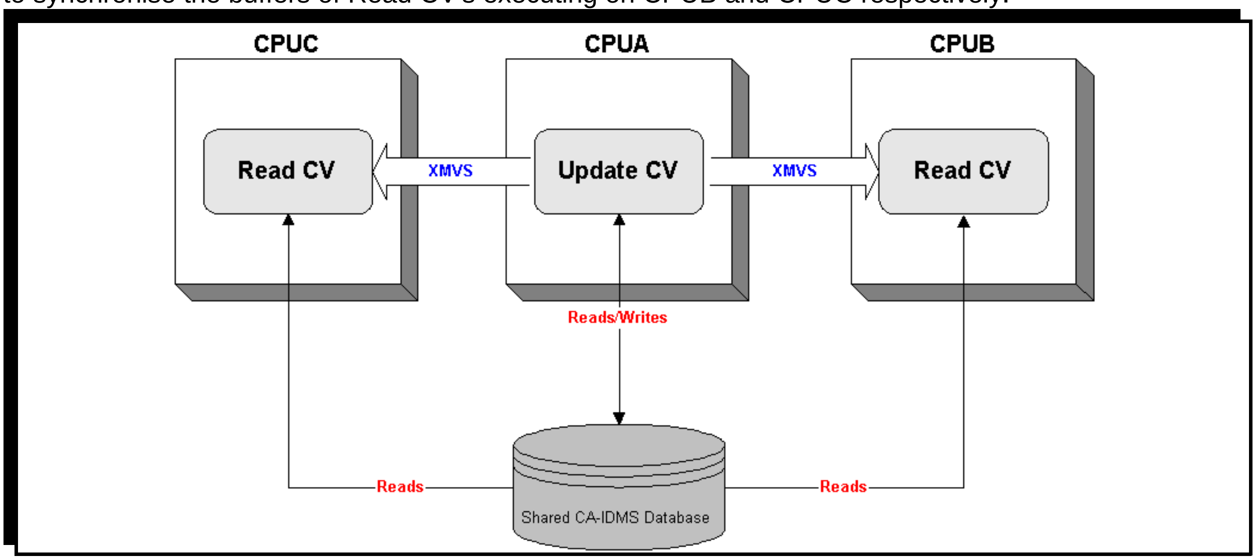


Figure 2.1 *CrossMVS connections across multi-Cpu's*.

To initiate CrossMVS synchronization, the user must initiate a CrossMVS master task (COXCPMAS) **for each session** for each EZ-Synchro SharedBuffer Group that requires the facility. For example, if SharedBuffer Group IDMSPROD on CPUA requires two sessions to connect to IDMSRTR1 on CPUB and IDMSRTR2 on CPUC, then you would start **two** CrossMVS jobs for IDMSPROD on CPUA, and **one** CrossMVS job on CPUB and CPUC respectively. The master task may be executed as a batch program or as a started task. **Figure 2.2** illustrates the EXEC statement required to submit the master task as a batch program. A mandatory EXEC parameter is the name of the EZ-Synchro SharedBuffer Group and the Session name for which this master task is to act.

```
//XMVS1     EXEC   PGM=COXCPMAS,PARM='IDMSPROD,XCPUS1     '
```

Figure 2.2 *Executing the CrossMVS master task (COXCPMAS)*

The MASTER task will then attach separate SEND and RECEIVE tasks to communicate with the corresponding SEND and RECEIVE tasks of the partner SharedBuffer Group. This is illustrated in **Figure 2.3**.

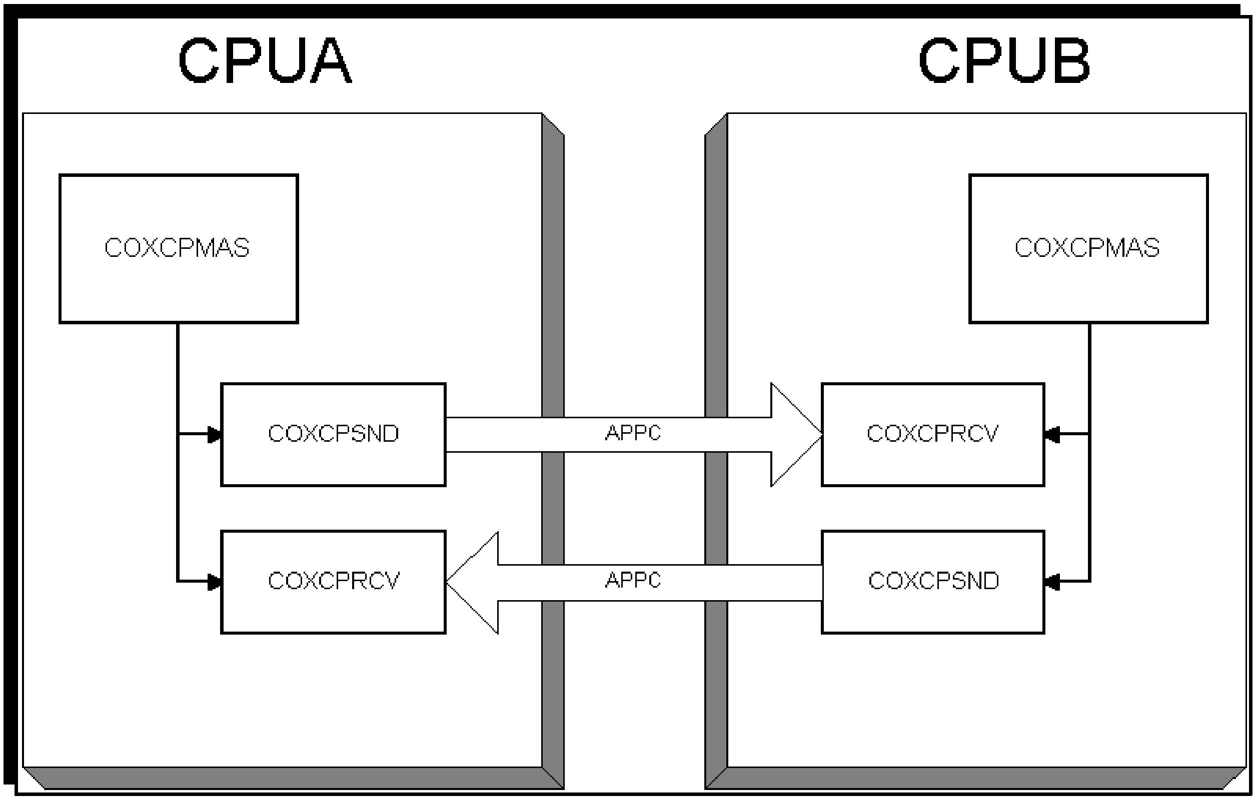


Figure 2.3 *CrossMVS Task/Subtask structure.*

Figure 2.3 illustrates the major components of CrossMVS option, namely:

- COXCPMAS The CrossMVS Master task.
- COXCPRCV The CrossMVS Receive task.
- COXCPSND The CrossMVS Send task.

The CrossMVS Master Task (COXCPMAS)

The function of the master task is as follows:

- Attach the Send subtask (COXCPSND)
- Attach the Receive subtask (COXCPRCV)
- Wait on Operator communication and process operator commands.
- Detach subtasks on shutdown.

The CrossMVS Send Task (COXCPSND)

The function of the Send Task is to establish a session with the corresponding Receive task, and send synchronization requests using LU6.2.

The CrossMVS Receive Task (COXCPRCV)

The function of the receive task is to receive synchronization requests from the partner SharedBuffer Group, and to invoke the buffer flush routine in each applicable group member using MVS cross-memory services. From this point of view, the Receive task behaves exactly like any other member of a EZ-Synchro group.

Defining the CrossMVS Environment.

This section describes the steps necessary to implement a EZ-Synchro environment.

The COXMVS table Overview.

The definition of the CrossMVS environment is defined primarily through the definition of a COXMVS table, although certain elements may be defined or overridden via run-time parameters. The COXMVS table is generated using the #COXMVS macro. The following illustrates the three classes of #COXMVS macro which may be combined to generate a COXMVS table:

```
#COXMVS     TYPE=INIT,<init parameters>
#COXMVS     TYPE=LU,<LU parameters>
#COXMVS     TYPE=END
```

To generate a valid COXMVS table, you would code a #COXMVS ENTRY=INIT statement followed by multiple TYPE=LU statements as required. The table is always terminated with a TYPE=END statement.

Figure 2.4 shows an example of a COXMVS table definition including the SESSION parameter introduced in Release 5.2.

```
#COXMVS     TYPE=INIT,....
            #COXMVS     TYPE=LU,CPUID=CPUA,GROUP=IDMSPROD,SESSION=XCPUS1,
                        SENDLU=PRODS1SA,RECVLU=PRODS1RA
            #COXMVS     TYPE=LU,CPUID=CPUA,GROUP=IDMSPROD,SESSION=XCPUS2,
                        SENDLU=PRODS2SA,RECVLU=PRODS2RA
            #COXMVS     TYPE=LU,CPUID=CPUB,GROUP=IDMSPROD,SESSION=XCPUS1,
                        SENDLU=PRODS1SB,RECVLU=PRODS1RB
            #COXMVS     TYPE=LU,CPUID=CPUC,GROUP=IDMSPROD,SESSION=XCPUS2,
                        SENDLU=PRODS2SC,RECVLU=PRODS2RC
            #COXMVS     TYPE=END
```

Figure 2.4 *Example of #COXMVS table definition.*

The above example illustrates the definition of an environment where SharedBuffer Group IDMSPROD on CPUA will synchronize with Group IDMSPROD on CPUB and Group IDMSPROD on CPUC.

The generated COXMVS table may either be hard-linked with the CrossMVS master task (COXCPMAS), or dynamically loaded from the CrossMVS jobs STEPLIB. If using dynamic loading, the table must be generated with a name of COXCPTAB.

Figure 2.5 shows the full syntax of the #COXMVS macro.

```
#COXMVS TYPE=type,      ( INIT | LU | END )
      SENDMAX=200,
      FREEMAX=100,
      TIMER=10,
      ERTIME=500
      CPUID=cpuid,
      GROUP=EZ-Synchro-group,
      SESSION=session-name,
      SENDLU=sendlu,
      RECVLU=recvlu
```

Figure 2.5 *The #COXMVS macro parameters.*

The specified parameters are as follows:

TYPE:

TYPE=INIT: This #COXMVS table parameter signals the start of a COXMVS table definition. There should be one and only one TYPE=INIT parameter in each table.

TYPE=LU: This parameter will define, for a given CPU and EZ-Synchro group, the names of the SEND and RECEIVE LU names to be used. There may be any number of TYPE=LU parameters, depending on how many CPUs and EZ-Synchro SharedBuffer Groups are in the configuration.

TYPE=END: This #COXMVS table parameter signals the end of the COXMVS table definition. There should be one and only one TYPE=END parameter in each table.

SENDMAX:

The parameter specifies the maximum number of elements on the send queue. When the active number gets above this, elements are not added to the send queue until the active number drops below this number. This prevents excessive use of ECSA.

This parameter is only valid on the TYPE=INIT statement.
The default value for this parameter is 200.

FREEMAX:

Requests for cross-mvs buffer synchronization are queued in the extended common storage area (ECSA) subpool 241. minimize the number of OS calls for GETMAINs and FREEMAINs a pool of used elements is maintained. This parameter defines the maximum number of free elements that will be kept.

This parameter is only valid on the TYPE=INIT statement.
The default value for this parameter is 100.

TIMER: Specifies the interval that buffer synchronization requests are sent to the partner SharedBuffer Group. It must be specified in 100ths of second.

This parameter is only valid on the TYPE=INIT statement.
The default value for this parameter is 10. (0.10 Seconds).

ERTIME: Specifies the interval that LU6.2 commands are retried after errors. It must be specified in 100ths of second.

This parameter is only valid on the TYPE=INIT statement.
The default value for this parameter is 500. (5.00 Seconds).

CPUID: *cpuid* The id of the CPU for this entry. The CPU id must match that defined in the field CVTSNAME. The maximum size of this parameter is 8 characters.

This parameter is only valid on the TYPE=LU statement.

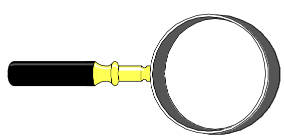
GROUP: *EZ-Synchro-group* The DB-Synchro SharedBuffer group name for this entry. The maximum size of this parameter is 8 characters.

This parameter is only valid on the TYPE=LU statement.

SESSION: *session-name* The name to be used to identify this CrossMVS SESSION. The maximum size of this parameter is 8 characters. The session name must commence with the characters XCPU, followed by any 1-4 alphanumeric characters.

For example, XCPUS1, and XCPUS2 are valid session names.

This parameter is only valid on the TYPE=LU statement.



The combination of CPUID, GROUP and SESSION must be unique in any given environment.

SENDLU: *sendlu* The LU name to be used for the SEND task for this specified GROUP/SESSION on this CPU. The maximum size of this parameter is 8 characters.

This parameter is only valid on the TYPE=LU statement.

RECVLU: *recvlu* The LU name to be used for the RECEIVE task for the specified GROUP/SESSION on this CPU. The maximum size of this parameter is 8 characters.

This parameter is only valid on the TYPE=LU statement.

Sample JCL for assembling and Link-Editing the COXMVS table is included in the EZ-DB installation source library member INSTSYNC. For reference, the required procedure is documented here.

```
//                  EXEC ASMHCL
//ASM.SYSLIB        DD
//                  DD    DSN=SYS1.MACLIB,DISP=SHR
//                  DD    DSN=EZ-DB.srclib,DISP=SHR
//                  DD    DSN=caidms.srclib,DISP=SHR
//ASM.SYSIN DD      *
                    #COXMVS statements....
//LKED.SYSLMOD      DD    DSN=EZ-DB.loadlib,DISP=SHR
//LKED.SYSIN        DD    *
  NAME coxmvs-name
//
```

Figure 2.6 *COXMVS Table assembly and Link-edit JCL.*

The information shown in lower-case should be modified as follows:

EZ-DB.srclib The name of the library into which you unloaded the EZ-DB installation tape source library.

caidms.srclib The name of the library which contains the standard CA-IDMS source members (macros).

EZ-DB.loadlib The name of the run-time load library containing the EZ-DB modules.

This library must be included in the STEPLIB concatenation of the CrossMVS execution JCL.

coxmvs-name The name of this EZ-Synchro® COXMVS table.

If this COXMVS table is to be dynamically loaded from STEPLIB, then it must be link-edited with a name of COXCPTAB.

As an alternative, you are permitted to hard-link the COXMVS table with the module COXCPMAS. In this case, the *coxmvs-name* may be any valid 1-8 character name.

Defining the Group-to-Group Connections

When a Master task starts, it obtains its applicable GROUP name and SESSION from the EXEC parameter. It can also determine the CPU Id of the machine on which it is executing from the MVS CVT.

The Master Task needs to find an entry for its Group and Session on this CPU. There should also be an entry with the same Session name on this or another CPU. For example, using the table defined in **Figure 2.4**, the master task started for IDMSPROD SESSION XCPUS1 on CPUA will find its Send and Receive LU names PRODS1SA and PRODS1RA respectively. It will then find the corresponding Session XCPUS1 entry - in this case IDMSPROD on CPUB. The partners Send and Receive LU names can thus be found.

The combination of CPUID, GROUP and SESSION has to be unique. The new SESSION parameter introduced in release 5.2 provides additional flexibility compared to earlier releases, in that the COXMVS table may be used to define connections between groups on the same CPU. In earlier releases this was only possible using the run-time overrides.

For example, to establish a session between two groups on the same CPU, the Group names must be different. However, A Session between 2 Groups on different CPUs, the Group Names could be the same.

Overriding Send/Receive LU names

The following illustrates the run-time parameter definition.

```
//RUN  EXEC PGM=COXCPMAS,
//     PARM='Group-Name,Session-Name,MySend,MyReceive,HisSend,HisReceive'
```

Figure 2.7 *COXCPMAS runtime parameters.*

If any of the LU names are specified then ALL must be specified.

The first two parameters *Group-Name* and *Session-Name* are mandatory.

NOTE: All parameters must be 8 characters. If less than 8, pad to the right with spaces. For example, `PARM='SBGRPR52,XCPUS1  '`

If you do not specify the LU names on the EXEC PARM, then COXCPMAS will search the COXMVS table for its own and partners SEND and RECEIVE LU names. There MUST be an entry in the table matching this *Group-name* and *Session-name* on *this* CPU. There should also be another entry with the same *Session-name.* The *cpuid* may or may not be the same. If the *cpuid* is the same, the *Group-name* must be different. If the *cpuid* is different, the *Group-name* may or may not be the same.

If you do specify the LU names on the EXEC PARM, then COXCPMAS will not search the COXMVS table at all. COXCPMAS will simply attach the Send (COSXPSND) and Receive (COXCPRCV) subtasks, which will open the SEND and Receive LUs respectively. These will attempt to connect to the corresponding partner Send and Receive LUs as defined on the EXEC parm.

The CrossMVS SharedBuffer table (ShrdBUFR).

Effective from release 4.1 of EZ-DB, there are three permissible classes of synchronization, each controlled by its own ShrdBUFR table. The three classes of synchronization are:

- EZ-Synchro® Update Synchronization (ShrdUPDT).
- EZ-Share® SharedBuffer Read Synchronization (ShrdREAD).
- EZ-Synchro® CrossMVS Synchronization (ShrdXMVS).

By default, EZ-Synchro® Update synchronization takes place even if no ShrdUPDT table is generated. However, for EZ-Share® or CrossMVS synchronization, you MUST generate a ShrdREAD and ShrdXMVS table respectively, specifying which Areas/Files require synchronization.

Figure 2.8 shows an example of a ShrdXMVS table that may be generated to define the CrossMVS synchronization requirement of a particular job.

```
#SHRBUFR    TYPE=START,NAME=SHRDXMVS,TAB=TOTAL
#SHRBUFR    TYPE=AREA,NAME=SYSTEM.*
#SHRBUFR    TYPE=AREA,NAME=USERSEG.DB-*
#SHRBUFR    TYPE=END
```

Figure 2.8 *Example of a CROSSMVS ShrdBUFR table.*

Unless you specify the MEMBER= parameter, it will be assumed that you wish to synchronize the Area with all active members in the partner group that share the same physical dataset.

You must specify the name of the appropriate table to be used by the CA-IDMS job either through the Driver Options table parameter (SBXMVS), or the run-time parameter (X=).

Refer to the EZ-Synchro and EZ-Share Guide for full details about the ShrdBUFR tables and specifying run-time parameters.

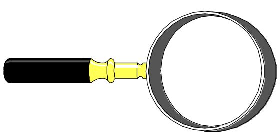
Figure 2.9 shows an example of a display received in response to the SHARE /X command.

```
EZ-DB (R5.3) SharedBuffer Status                          V0200 16:36:38 92:330
MODE: Display CrossMVS Synchronization
Area Name                   PGRP * Filename F#  12345678901234567890123456789012
--------------------------------------------------------------------------------
RUN-DDLDML                     0   DICTDB    1  ................................
DDLDML                         0   SGENDB    1  ................................
DDLDCMSG                       0   DMSGDB    1  ................................
DDLDCLOG                       0   DLOGDB    1  ................................
DDLDCSCR                       0   DSCRDB    1  ................................
DDLDCQUE                       0   DQUEDB    1  ................................
DDLDCRUN                       0   DRUNDB    1  ................................
                                                ................................
                                                ................................
                                                ................................
                                                ................................
                                                ................................
                                                ................................
                                                ................................
--------------------------------------------------------------------------------
NO MORE AREAS TO DISPLAY
<CLEAR> EXIT <PF8> DOWN
ENTER NEW COMMAND BELOW>
```

Figure 2.9 *Database synchronization table display*.

It should be noted that, unlike standard EZ-Synchro, CrossMVS does not know which datasets are physically shared until the request reaches the partner group. Therefore, the SHARE /X display will show all of the areas selected by the ShrdXMVS table, not just the areas which are also currently shared by other active jobs.

Whenever any of the areas included in the ShrdXMVS table are updated, a synchronization request (send-element) is sent to the partner group. The RECEIVE task, acting for the partner group will perform cross-memory synchronization to any active member of the group that has opened the same physical file (in retrieval mode).



It should be noted that it is possible to cause unnecessary synchronization requests to be sent by improper definition of the ShrdXMVS table.

SharedBuffer Group table definition.

Any EZ-Synchro SharedBuffer Group that participates in CrossMVS activity (either as a sender, receiver or both) must have an entry defined with a name *session-name.* i.e., there must be a SharedBuffer Group entry for each possible session. It follows that, as there can only be a maximum of 32-entries in a group, the number of sessions limits the number of other entries that may exist in the group.

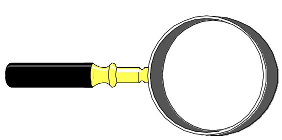
The following figure shows an SSB display for such a group.

```
EZ-DB (R5.3) SharedBuffer Group Status Display        V0200 16:36:05 92:330

Group Name: IDMSR120  Creation Date: 11/05/92 16:02        PC: CV    Stack: 500
Current CV: IDMSRTR3  ShrdUPDT: DBSYUPDT  ShrdREAD:        ShrdXMVS: DBSYXMVS

--------------------------------------------------------------------------------
 #  MEMBER    S SVC CV# 12345678901234567890123456789012  R    RD-HITS    WR-HITS
--------------------------------------------------------------------------------
 1  XCPUS1    A   .   . .3.5............................  .          0          0
 2  XCPUS2    .   .   . ................................  .          0          0
 3  IDMSR12   R 241  12 ..5.............................  .          0       1025
 4  IDMSDEV   .   0   0 ................................  .          0          0
 5  IDMSRTR3  R 238   1 3...............................  .          0        918
 6  IDMSBAT1  .   0   0 ................................  .          0          0
 7*           .   0   0 ................................  .          0          0
 8*           .   0   0 ................................  .          0          0
 9*           .   0   0 ................................  .          0          0


--------------------------------------------------------------------------------
<ENTER> REFRESH  <CLEAR> EXIT  <PF12> MODE SWITCH         <PF8> NEXT <PF7> PRIOR
ENTER NEW COMMAND BELOW>
```

Figure 2.10 *Group Table Display (SSB Task)*

The position of the Session entries in the group is immaterial, but for consistency it may be useful if they are always specified first.

In the example, it can be seen that there are <u>two</u> XMVS sessions defined for this Group, namely XCPUS1 and XCPUS2, only the first of which is currently active. The RECEIVE task for Session XCPUS1 *has* received buffer synchronization requests (*send elements*) from its partner group for IDMSR12 and IDMSRTR3. It can be seen that XCPUS1 has *connected* to these group members and performed cross-memory synchronization requests.



Technical Note: The "Status" indicator in the SSB display will appear as "A" for Active as opposed to "R" for Running. Although it appears as a member of the group, it is not actually an IDMS job.

Refer to the EZ-Synchro User Guide for full details on the SBGROUP table definition.

EZ-Synchro CrossMVS Operation.

Starting the CrossMVS Driver.

The CrossMVS driver may be started as a batch program or as a started task. JCL required to run the program is shown in **Figure 3.1**.

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

//*  START CROSS-CPU PROCESSING - GROUP IDMSXCPU                      *
//*********************************************************************
//*
//*   THIS JOB STARTS CROSS-CPU PROCESSING FOR GROUP IDMSXCPA ON CPUA.
//*   IT WILL CONNECT TO GROUP IDMSXCPB ON CPUB.
//*   IDMSXCPA: USES XCPUAPS1 AND XCPUAPR1 AS SEND/RECEIVE LU'S.
//*   IDMSXCPB: USES XCPUBPS1 AND XCPUBPR1 AS SEND/RECEIVE LU'S.
//*
//*-------------------------------------------------------------------
//* STEP XCPUA - START EZ-SYNCHRO CROSS-CPU FACILITY
//*-------------------------------------------------------------------
//XCPUA    EXEC  PGM=COXCPMAS,
//***---         PARM='IDMSXCPA,SESSION1'
//               PARM='IDMSXCPA,SESSION1,XCPUAPS1,XCPUAPR1,XCPUBPS1,XCPUBPR1'
//STEPLIB  DD DSN=crosscpu.loadlib,DISP=SHR
//*
//SNAPMAS  DD SYSOUT=X,HOLD=YES
//SNAPSND  DD SYSOUT=X,HOLD=YES
//SNAPRCV  DD SYSOUT=X,HOLD=YES
//DEBUGMAS DD SYSOUT=X,HOLD=YES
//DEBUGSND DD SYSOUT=X,HOLD=YES
//DEBUGRCV DD SYSOUT=X,HOLD=YES
//SYSLGSND DD SYSOUT=X,HOLD=YES
//SYSLGRCV DD SYSOUT=X,HOLD=YES
//SYSABEND DD SYSOUT=X,HOLD=YES
//*
//*-------------------------------------------------------------------
//*      END OF COXCPMAS JOB
//*-------------------------------------------------------------------
```

Figure 3.1 *Starting the CrossMVS driver program.*

crosscpu.loadlib is the library containing the executable CrossMVS option modules.

The SYSOUT files must all be included as specified otherwise the Master task will Abend at startup.

Note: A member (COXCPRUN) containing this sample installation job-stream can be found in the EZ-DB installation tape source file.

The CrossMVS master task will attach the Send and Receive tasks which will attempt to connect to the applicable partner Send and Receive Tasks. If the partner CrossMVS facility has not been started, error messages will be written to the log at regular intervals until a successful link-up is established.

In normal circumstances, you should endeavour to start and end the CrossMVS facility on each CPU at the same time.

Specifying Run-time Parameters.

As you can see in **Figure 3.1**, it is necessary to specify a parameter on the COXCPMAS EXEC statement. The minimum requirement is that you specify the SharedBuffer group and Session name. These are always the first two parameters specified. Optionally, you may also specify the names of the SEND and RECEIVE LUs for this job and the partner job that represents the other side of this Session. If specified, the syntax is as shown in **Figure 3.2**.

```
//    EXEC PGM=COXCPMAS,
//    PARM='group-name,session-name,MySend,MyReceive,HisSend,HisReceive
```

Figure 3.2 *COXCPMAS runtime parameters.*

group-name The name of the SharedBuffer group for this CrossMVS master task. This is a mandatory parameter.

session-name The name of the CrossMVS session. This is a mandatory parameter. There should be another COXCPMAS task started on this or another CPU with the same Session name.

The session name must commence with the characters XCPU followed by 1-4 additional characters. For example, XCPUS1 is a valid session name,

If any of the following are specified, then ALL must be specified.

MySend The LU name to be used by the SEND task for this group/session.

MyReceive The LU name to be used by the RECEIVE task for this group/session.

HisSend The LU name to be used by the SEND task of the partner group/session.

HisReceive The LU name to be used by the RECEIVE task of the partner group/session..

Note: Each parameter must be 8 alpha-numeric characters. Pad with blanks if necessary.

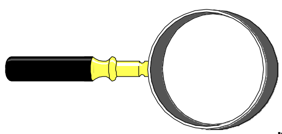
If you do not specify the LU names on the EXEC parm, then the Master task will search the COXMVS table for its own and partners SEND and RECEIVE LU names. If they are not defined in the table, the program will abend.

XMVS Online Display Program.

To display the EZ-Synchro CrossMVS status and statistics display, enter the task code XMVS at the DC/UCF "Enter Next Task Code" prompt". The syntax of the command is as follows:-

XMVS <*session-name*>

session-name is an optional parameter. If omitted, the XMVS task will initially display the first session defined for the current EZ-Synchro SharedBuffer Group. You may display other sessions applicable to this Group (in the order they were created) by using the <**PF8**> key. If you specify a *session-name* on the XMVS command line, then the XMVS task will initially display that particular session.



Each session is represented by an XCPUCNTL. block. The first session is anchored in the SharedBuffer Groups SSB (Subsystem Block). Once created, an XCPUCNTL block will be retained until the next IPL. Therefore, the XMVS task may display both *active* and *inactive* sessions. When a session starts, the COXCPMAS task will re-use the existing XCPUCNTL for the session if it exists, or allocate it if it does not.

```
EZ-SYNCHRO (R5.3)      *** CROSS-CPU STATISTICS ***         V0010 18:05:21 91.025

  CROSS-CPU STATUS: ACTIVE      SEND INTERVAL:     0.10   RETRY INTERVAL:   5.00
      SESSION NAME: XCPUS1

  # SENDS:                          6       # RECEIVES:                        0

  # ELEMENTS SENT:                186       # ELEMENTS RECEIVED:               0

 :-------  SEND QUEUE STATUS  --------:   :-------  FREE QUEUE STATUS  ---------:
  # ELEMENTS ON SEND QUEUE:         1     # ELEMENTS ON FREE QUEUE:          2
  SEND QUEUE HIGH WATER:            3     FREE QUEUE HIGH WATER:             3
  SEND QUEUE MAXIMUM:             200     FREE QUEUE MAXIMUM:              100
  # TIMES AT MAXIMUM:               0
 :------------------------------------:   :------------------------------------:

       SEND LU:XCPUAPS1   >>>>>>>>>>>>>>                 XCPUBPR1:RECV LU
       RECV LU:XCPUAPR1                 <<<<<<<<<<<<<<<  XCPUBPS1:SEND LU

<ENTER> REFRESH   <CLEAR> EXIT   <PF8> NEXT SESSION
```

Figure 3.3 *The XMVS status display.*

Figure 3.3 illustrates the information displayed by the XMVS task.

The following describes the information displayed by the XMVS task.

Status: Active-The CrossMVS driver task has been started for this SharedBuffer group/Session.
Not Active-The CrossMVS driver task has not been started for this SharedBuffer group/Session.

Send Interval: The interval (in seconds) between LU6.2 Sends to the partner group.

The default send interval is defined by the TIMER parameter in the COXMVS table definition, and may be dynamically changed using an operator command.

Retry Interval: The interval (in seconds) between attempts to retry after LU6.2 errors.

The default send interval is defined by the ETIMER parameter in the COXMVS table definition, and may be dynamically changed using an operator command.

Session Name: The name of the CrossMVS Session.

 # Sends: The number of buffers which have been sent by the CrossMVS Send task to the partner Receive task.

 # Elements Sent: The number of synchronization requests sent to the partner group. This will be greater than # SENDS since a buffer may contain multiple send-elements.

 # Receives: The number of buffers received from the partner group.

 # Elements Received: The number of buffer synchronization requests received from the partner group. This will be greater than # RECEIVES since a buffer may contain multiple elements.

SEND QUEUE STATUS

 # Elements on Send Queue: The number of elements on the send queue waiting to be sent.

Send Queue High Water: The highest number of send queue elements waiting to be sent.

Send Queue Maximum: The maximum number of elements allowed on the active queue. This value is defined in the #COXMVS table, SENDMAX parameter.

 # Times at Maximum: The number of times EZ-Synchro was unable to add a synchronization request to the active queue because the maximum number of send elements were already active.

As soon as the number of elements drops below the maximum, synchronization requests will continue being sent.

FREE QUEUE STATUS

 # Elements on Free Queue: The number of elements on the free queue.

Free Queue High Water: The highest number of elements on the free queue.

Free Queue Maximum: The maximum number of elements allowed on the free queue. This value is defined in the #COXMVS table, FREEMAX parameter.

CrossMVS Operator Commands.

This section describes the commands that can be used to communicate with the CrossMVS Master task from the Operator console (or authorized SDSF session).

In all cases, *jobname* refers to the jobname or started task name of the executing COXCPMAS submission.

```
/F jobname,DISPLAY,TIME
```

This command displays the interval that buffer synchronization requests are sent to the partner EZ-Synchro group. The time displayed is in seconds.

The following message will appear on the operator console:

```
COXCPMAS:APPC012C-NORMAL TIME INTERVAL=XXX.XX SECONDS
```

```
/F jobname,DISPLAY,ETIME
```

This command displays the interval that LU6.2 commands are retried after errors.

The following message will appear on the operator console:

```
COXCPMAS:APPC013C-ERROR TIME INTERVAL=XXX.XX SECONDS
```

```
/F jobname,MOD,TIME=01.50
```

This command can be used to modify the interval that buffer synchronization requests are sent over the LU6.2 link. The time specified is in seconds.

The following message will be written to the operator console:

```
COXCPMAS:APPC010C-NORMAL TIME INTERVAL CHANGED; OLD=XXX.XX SECONDS,
NEW=XXX.XX SECONDS
```

```
/F jobname,MOD,ETIME=01.50
```

This command can be used to modify the interval that LU6.2 commands are retried after errors.

The following message will be written to the operator console:

```
COXCPMAS:APPC011C-ERROR TIME INTERVAL CHANGED; OLD=XXX.XX SECONDS,
      NEW=XXX.XX SECONDS
```

```
/F jobname,SHUTDOWN          -OR-    /P jobname
```

This command causes the normal shutdown of the CrossMVS facility on this CPU. (The command must be issued on both CPU's to shut down both sides).

The following messages should appear on the console:

```
COXCPMAS:APPC004C-OPERATOR COMMAND: SHUTDOWN
COXCPMAS:APPC003C-OPERATOR REQUESTED STOP
COXCPMAS:APPC007C-PROCESSING OF OPERATOR COMMAND HAS COMPLETED
```

```
/F jobname,TEST
```

This command causes a WTO message to be displayed on the operators console (and system log) of the other CPU. This validates that the CrossMVS SEND task can communicate with the corresponding RECEIVE task. To completely test the installation of CrossMVS, issue this command on both sides.

The following messages should appear on the console of each CPU:

```
COXCPSND:APPC020S-TEST MESSAGE SENT VIA LU lu-name TO PARTNER RECEIVE
         PROCESS BY COXCPSND -SEND PROCESS.

COXCPRCV:APPC019S-TEST REQUEST FROM SEND PROCESS CONFIRMED ON
      RECEIVE LU lu-name
```

Error Codes and Messages

Information Messages.

MASTER TASK (COXCPMAS)

```
COXCPMAS:APPC003C-OPERATOR REQUESTED STOP
COXCPMAS:APPC004C-OPERATOR COMMAND:
COXCPMAS:APPC005C-UNKNOWN OPERATOR MODIFY COMMAND ENTERED
COXCPMAS:APPC007C-PROCESSING OF OPERATOR COMMAND HAS COMPLETED
COXCPMAS:APPC008C-PLEASE SPECIFY OPERAND IN THE FORM TIME=99.99
COXCPMAS:APPC009C-PLEASE SPECIFY OPERAND IN THE FORM ETIME=99.99
COXCPMAS:APPC010C-NORMAL TIME INTERVAL CHANGED; OLD=XXX.XX SECONDS,
NEW=XXX.XX SECONDS
COXCPMAS:APPC011C-ERROR TIME INTERVAL CHANGED; OLD=XXX.XX SECONDS,
      NEW=XXX.XX SECONDS
COXCPMAS:APPC012C-NORMAL TIME INTERVAL=XXX.XX SECONDS
COXCPMAS:APPC013C-ERROR TIME INTERVAL=XXX.XX SECONDS
COXCPMAS:APPC016C-XXXXXXX SUBTASK ENDED WITH SYSTEM COND. CODE OF:XXXX;
COXCPMAS:APPC017C-XXXXXXX SUBTASK ENDED MORE THAN XXXX TIMES.
COXCPMAS:APPC018C-PARMS SHOULD BE IN THE FORM 'GROUP,MYSENDLU,
         MYRECEIVELU,HISSENDLU,HISRECEIVELU' -PLEASE RESUBMIT JOB.
COXCPMAS:APPC019C-UNABLE TO RESTART SUBTASK DUE TO USER COND. CODE
COXCPMAS:APPC020C-TIME INTERVAL MUST BE 00.01 SECONDS AT MINIMUM
COXCPMAS:APPC021C-ERROR TIME INTERVAL MUST BE 00.01 SECONDS AT MINIMUM
COXCPMAS:APPC024C-UNKNOWN FIELD TO BE MODIFIED.
COXCPMAS:APPC025C-UNKNOWN FIELD TO BE DISPLAYED.
COXCPMAS:APPC026C-COXCPMAS START AT XXXXXXXXXXXXXXXXXXXXXX
```

SEND TASK (COXCPSND)

```
COXCPSND:APPC006E-LU6.2 ERR:XXXXXXXX XXXXXXXX RQ=XX QU=XX R15=XX R0=XX
         PR=XXXX SC=XXXX RCV1=XX SI=XXXXXXXX SO=XXXXXXXX
COXCPSND:APPC017S-UNABLE TO OPEN PLU XXXXXXXX - ERRFLG: XX
COXCPSND:APPC018S-SETLOGON ERROR FOR PLU XXXXXXXX - FDBK:XXXXXX
COXCPSND:APPC019S-UNEXPECTED ECB POSTED AT ADDRESS XXXXXXXX - PROGRAM
         COXCPSND LOADED AT XXXXXXXX
COXCPSND:APPC020S-TEST MESSAGE SENT VIA LU XXXXXXXX TO PARTNER RECEIVE
         PROCESS BY COXCPSND -SEND PROCESS.
COXCPSND:APPC021S-INVALID BUFFER FLUSH REQUEST. PAGE GROUP:XXXX
         CVNAME:XXXXXXXX PAGE: XXXXXXXX
COXCPSND:APPC021S-ZERO LENGTH SEND BUFFER. LOOK FOR INVALID PAGE # TO
         ACCOUNT FOR IT.
```

RECEIVE TASK (COXCPRCV)

```
COXCPRCV:APPC006E-LU6.2 ERR:12345678 12345678 RQ=12 QU=12 R15=12 R0=12
         PR=1234 SC=1234 RCV1=12 SI=12345678 SO=12345678
COXCPRCV:APPC017S-UNABLE TO OPEN PLU 12345678 - ERRFLG: 12
COXCPRCV:APPC018S-SETLOGON ERROR FOR PLU 12345678- FDBK:123456
COXCPRCV:APPC019S-TEST REQUEST FROM SEND PROCESS CONFIRMED ON
      RECEIVE LU 12345678
```

Error Messages.

`COXCPMAS: ERROR FOLLOWING COGINDEX CHAIN`

This error may occur if you have specified an invalid SharedBuffer Group name. Another reason for this error is that you have started up the CrossMVS master task before any IDMS CV in the applicable SharedBuffer Group has started after an IPL. The first IDMS CV to start after an IPL will initialize the SharedBuffer Groups SSB control block. Therefore, you must start at least one CV after an IPL to avoid this error. Start a CV then re-submit the CrossMVS master task.

`COXCPMAS: WRONG LENGTH PARAMETER PASSED. ABENDING`

Invalid parameter format. Correct the EXEC parm and re-submit.

`COXCPMAS: UNABLE TO OPEN DEBUG FILE. ABENDING`

The DBUGMAS DD statement is missing from the JCL.

`COXCPMAS: UNABLE TO OPEN SNAP FILE. ABENDING`

The SNAPMAS DD statement is missing from the JCL.

`COXCPMAS: COXMVS TABLE NOT LINKED. ABENDING`

The COXMVS table has not been link-edited with the COXCPMAS program. Refer to CrossMVS user guide. Correct situation and resubmit job.

`COXCPMAS: COXMVS TABLE ERR; LU FOR THIS JOBS GROUP AND CPU NOT FOUND`

COXCPMAS could not find an entry in the COXMVS table for this group on this CPU.

`COXCPMAS: COXMVS TABLE ERR; LU FOR PARTNER JOBS GROUP AND CPU NOT FOUND`

COXCPMAS could not find an entry in the COXMVS table for the partner group.

`COXCPMAS: UNABLE TO OPEN DEBUG FILE. ABENDING`

The DEBUGMAS DD statement is missing from the JCL.

`COXCPMAS: DUPLICATE JOB. EXITING PROGRAM.`

The job has been submitted twice.

`COXCPRCV:UNABLE TO OPEN DEBUG FILE'`

The DEBUGRCV DD statement is missing from the JCL.

`COXCPRCV:UNABLE TO OPEN SNAP FILE'`

The SNAPRCV DD statement is missing from the JCL.

COXCPRCV:UNABLE TO OPEN SYSLOG FILE

The SYSLGRCV DD statement is missing from the JCL.

COXCPRCV:UNABLE TO OPEN DEBUG FILE'

The DEBUGRCV DD statement is missing from the JCL.

COXCPRCV:UNABLE TO OPEN SNAP FILE'

The SNAPRCV DD statement is missing from the JCL.

COXCPRCV:UNABLE TO OPEN SYSLOG FILE

The SYSLGRCV DD statement is missing from the JCL.

Appendix.

VTAM Definitions.

The following VTAM definition statements are provided for your information as a guide to your Network programmer when setting up the VTAM LU's for CrossMVS option. These definitions work as coded! If you have any problems connecting or communicating between LU's, first check that the VTAM definitions are as defined in the following.

VBUILD TYPE=APPL (CPUA)

```
APXCPUA  VBUILD TYPE=APPL
XCPUAPS1 APPL   ACBNAME=XCPUAPS1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUAPR1 APPL   ACBNAME=XCPUAPR1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUASS1 APPL   ACBNAME=XCPUASS1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUASR1 APPL   ACBNAME=XCPUASR1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
```

VBUILD TYPE=APPL (CPUB)

```
APXCPUB  VBUILD TYPE=APPL
XCPUBPS1 APPL   ACBNAME=XCPUBPS1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUBPR1 APPL   ACBNAME=XCPUBPR1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUBSS1 APPL   ACBNAME=XCPUBSS1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
XCPUBSR1 APPL   ACBNAME=XCPUBSR1,MODETAB=EDJMODTB,DLOGMOD=LU62BIND,      X
                EAS=1,APPC=YES,DMINWNL=2,DMINWNR=0,DSESLIM=2,            X
                AUTOSES=1,AUTH=(SPO)
```

CROSS-DOMAIN RESOURCE (CPUA)

```
*
*   THIS CDRSC DEFINITION IS TO ALLOW ACCESS TO XCPUA LU 6.2 SESSIONS
*
CDRSXCPA  VBUILD TYPE=CDRSC
XCPUAPS1  CDRSC  CDRM=EDJ01,ISTATUS=ACTIVE
XCPUAPR1  CDRSC  CDRM=EDJ01,ISTATUS=ACTIVE
XCPUASS1  CDRSC  CDRM=EDJ01,ISTATUS=ACTIVE
XCPUASR1  CDRSC  CDRM=EDJ01,ISTATUS=ACTIVE
```

CROSS-DOMAIN RESOURCE (CPUB)

```
*
*   THIS CDRSC DEFINITION IS TO ALLOW ACCESS TO XCPUB LU 6.2 SESSIONS FROM CPUA
*
CDRSXCPB  VBUILD TYPE=CDRSC
XCPUBPS1  CDRSC  CDRM=EDJ02,ISTATUS=ACTIVE
XCPUBPR1  CDRSC  CDRM=EDJ02,ISTATUS=ACTIVE
XCPUBSS1  CDRSC  CDRM=EDJ02,ISTATUS=ACTIVE
XCPUBSR1  CDRSC  CDRM=EDJ02,ISTATUS=ACTIVE
```

MODE TABLE ENTRY

```
***********************************************************************
* MODE TABLE ENTRY FOR TESTING LU6.2 SESSIONS BETWEEN CPU'S            *
***********************************************************************
          SPACE 1
LU62BIND  MODEENT LOGMODE=LU62BIND,                                     +
                TYPE=X'00',                                             +
                FMPROF=X'13',                                           +
                TSPROF=X'07',                                           +
                PRIPROT=X'B3'                                           +
                SECPROT=X'B3'                                           +
                COMPROT=X'70B1',                                        +
                PSNDPAC=X'00'                                           +
                SRCVPAC=X'00',                                          +
                SSNDPAC=X'04',                                          +
                RUSIZES=X'8585'                                         +
                PSERVIC=X'060200000000000000004000'
          SPACE 1
***********************************************************************
```

This is the MODETAB definition referred to in the prior APPL definitions.

Data Structures (ECSA usage).

The following figure (**Figure A.1**) illustrates the control block structure used by the CrossMVS facility.

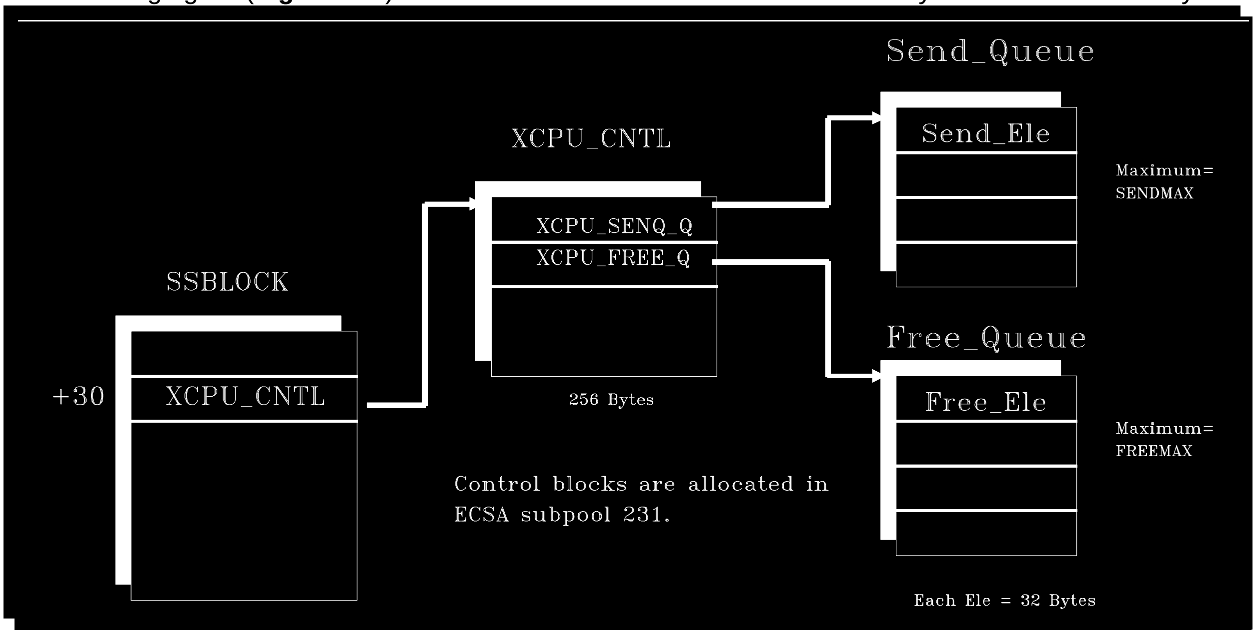


Figure A.1 *CrossMVS option control structures.*

XCPU_CNTL Block.

There is one XCPU_CNTL block (length 512 bytes) for each Session for each EZ-Synchro SharedBuffer Group that uses the CrossMVS facility. The XCPU_CNTL block is allocated the first time a particular Session starts for a given EZ-Synchro group. The first XCPU_CNTL is anchored in the groups SSBLOCK. Subsequent XCPU_CNTL blocks are chained off the first. This control block is used to maintain statistical information as well as anchors to the CrossMVS Send-Queue and Free-Queue.

If a EZ-Synchro group is regenerated (*new-copied*), the address of the first XCPU_CNTL block is copied into the new SSBLOCK. The XCPU_CNTL blocks are only deleted at system IPL.

SEND_QUEUE and FREE_QUEUE

Each request for buffer synchronization causes a *Send-Element* to be placed on the *Send-Queue*. The length of one element is 92-bytes. The maximum number of elements on the *Send-Queue* is defined by the SENDMAX parameter of the #COXMVS macro.

To avoid excessive GETMAIN and FREEMAIN requests, the CrossMVS option maintains a pool of *Free-Elements*. When a *Send-Element* is required, an available element is taken from the *Free-Queue*, initialized, and placed on the *Send-queue*. If there is no available element on the *Free-Queue* then the *Send-Element* is GETMAINed from ECSA (subpool 241).

After the *Send-Element* has been sent, the used element is placed on the *Free-Queue* provided that the current number of elements on the *Free-Queue* does not exceed the number specified by the FREEMAX parameter of the #COXMVS macro. If the *Free-Queue* is already full, the element is FREEMAINed.

COXCPSND Send Buffers.

The actual transmission of buffer synchronization requests to the partner group is performed by the SEND task (COXCPSND) using alternate SEND Buffers. COXCPSND takes elements off the *Send-Queue* and places them in its own SEND Buffer. Each Send Buffer can hold up to 256 *Send-Elements*. This is why, when you observe the CrossMVS statistics, the statistic **# Sends** is less than **# Elements Sent** and **# Receives** is less than **# Elements Received**. A buffer is sent either when it is full, or when the send time interval expires.

NOTES: