TPS

IDMS

TPS-IDMS INTERFACE MANUAL

This version of the **TPS-IDMS Interface** manual incorporates the enhancements provided in Releases 3.10 (TPS/ICL-DME) and 2.03 (TPS/ICL-VME) of the TPS software.

Incorporating Amendments up to Number: 3

TPS-IDMS INTERFACE MANUAL

This version of the **TPS-IDMS Interface** manual incorporates the enhancements provided in Releases 3.09 onwards (DM) and 2.02 onwards (VM) of the TPS software.

Incorporating Amendments up to number 2

CONTENTS

		Page
INTRODUCTION		
1.	PURPOSE OF THE MANUAL	iii
2.	STRUCTURE OF THE MANUAL	iii
3.	ASSOCIATED MANUALS	iii
PART 1 : IDMS INTERFACE FOR DM SYSTEMS		
1.	SUMMARY	1 - 1
1.1	IDMS FACILITIES	1 - 1
1.2	IDMS OPTIONS	1 - 1
2.	SYSTEM DESIGN CONSIDERATIONS	1 - 2
2.1	GENERAL	1 - 2
2.1.1	Threading within Database Calls	1 - 2
2.1.2	Currency Checking	1 - 2
2.2	SYSTEM START-UP	1 - 2
2.2.1	Establishing the IDMS Area	1 - 3
2.2.2	Initialising the Database	1 - 4
2.3	SYSTEM CLOSEDOWN	1 - 4
2.4	APPLICATION ROUTINE INTERFACE	1 - 5
2.5	ABORTING MESSAGES	1 - 5
2.6	RECOVERING FROM FILE FAILURE	1 - 5
2.7	RECOVERING FROM SYSTEM FAILURE	1 - 6
2.7.1	Warm Restart	1 - 6
2.7.2	Database Security	1 - 6
3.	WRITING APPLICATION ROUTINES	1 - 7
3.1	GENERAL	1 - 7
3.2	SETTING UP THE TPSIDMS PROTOCOL	1 - 7
3.3	ENVIRONMENT DIVISION	1 - 8
3.4	DATA DIVISION	1- 8
3.5	PROCEDURE DIVISION	1 - 9
3.6	DML COMMANDS	1 - 9
3.7	ERROR-STATUS HANDLING	1 - 11
4.	SYSTEM IMPLEMENTATION FOR TPS-IDMS	1 - 12
4.1	GENERATION FORMS	1 - 12
4.2	MODIFYING STANDARD TPS AR TRAINS	1 - 12
4.3	CONSOLIDATION	1 - 13
4.4	IDMS FORMATS	1 - 13
4.5	NOTE FOR ME29 USERS	1 - 13

CONTENTS

		Page
PART 2 : IDMS INTERFACE FOR VM SYSTEMS		
1.	SUMMARY	2 - 1
1.1	IDMS FACILITIES	2 - 1
1.2	IDMS OPTIONS	2 - 1
2.	SYSTEM DESIGN CONSIDERATIONS	2 - 1
2.1	SYSTEM START-UP & CLOSEDOWN	2 - 2
2.1.1	SCL Procedure TPS_IDMS_START	2 - 2
2.1.2	SCL Procedure TPS_IDMS_RUN	2 - 2
2.1.3	SCL Procedure TPS_IDMS_CLOSE	2 - 2
2.2	AMENDMENTS TO START-UP AR TRAIN	2 - 3
2.3	INSTALLING THE TPS-IDMS DATA MODULE	2 - 3
2.4	IDMS_PARAMS	2 - 3
2.5	RECOVERING FROM SYSTEM FAILURE	2 - 4
3.	WRITING APPLICATION ROUTINES	2 - 5
3.1	GENERAL	2 - 5
3.2	SETTING UP THE TPSIDMS PROTOCOL	2 - 5
3.3	BIND RUN UNIT	2 - 6
3.4	ERROR-STATUS HANDLING	2 - 7
APPENDIX A:	EXAMPLE START-UP ARs FOR DM	A - 1
APPENDIX B:	EXAMPLE START-UP AR FOR VM	B - 1
APPENDIX C:	ZIDF: STANDARD FUNCTION TO ISSUE FILE COMMANDS	C - 1
APPENDIX D:	SYSTEM DEFINITION FOR TPS-IDMS	D - 1
D.1	KEYWORD IDMS	D - 1
D.2	NOTE ON FILE MANAGEMENT IN A DM SYSTEM	D - 2

INTRODUCTION

1. PURPOSE OF THE MANUAL

The TPS-IDMS interface has been developed to provide the user of TPS with access to data held within an IDMS database in a manner compatible with the normal facilities of TPS.

The purpose of this manual is to give details of the system design considerations when designing a TPS program with the IDMS interface, to give some pointers for writing Application Routines and to specify the parameters necessary for generating the IDMS interface into the TPS program.

2. STRUCTURE OF THE MANUAL

This manual is divided into two parts. Part 1 describes the IDMS Interface for DM Systems, including considerations of System Design, the writing of Application Routines, and System Implementation.

Part 2 describes the IDMS Interface for VM Systems, including considerations of System Design and the writing of Application Routines.

3. ASSOCIATED MANUALS

There are two manuals directly associated with this manual. These are:

- **Common COBOL Interface** manual, which describes the programming method used in a TPS program.
- **Features & Facilities** manual, which describes the TPS facilities that are available to the applications designer.

PART ONE

IDMS INTERFACE FOR DM SYSTEMS

1. SUMMARY

1.1 IDMS FACILITIES

The TPS-IDMS interface provides the following facilities:

- The ability to initialise and close down the database in the TPS start-up and closedown AR Trains.
- Control of access to IDMS through a TPS request.
- Full warm restart in the event of system failure.
- The ability to 'roll back' database transactions in the event of a message being aborted.
- An on-line transaction to issue File Commands to IDMS (Non-shareable systems only).
- The automatic 'roll back' of database transactions when using a terminal in training mode.

1.2 IDMS OPTIONS

The following options are available (selected by generate options):

- IDMS Release 4.
- IDMS Release 5+ without a log record on the database.
- IDMS Release 5+ with a log record on the database.
- Shareable IDMS with a log record on the database.

Note: The start-up process for an on-line TPS program incorporating IDMS Release 6 Shareable Database differs from that for all other versions. The start-up process begins as normal, but stops before the process is complete and the program is rolled into its MTS slot. When the first message is input, the remainder of the start-up process will be performed and then the message will be processed.

Users should be aware that this will appear to increase the processing time for the first message.

IDMS INTERFACE FOR DM SYSTEMS

2. SYSTEM DESIGN CONSIDERATIONS

2.1 GENERAL

A TPS program that is to include the TPS-IDMS interface should be designed to the same principles as a normal TPS program as described in the TPS **Common COBOL Interface** manual. It is possible to have a TPS program that supports both normal housekeeping files and an IDMS database. The IDMS database will use unit reference numbers above the highest unit reference number defined to TPS.

The IDMS files are handled solely by the IDMS package; they are not defined in any way to TPS. The user must ensure that the disc unit reference numbers used by TPS do not clash with those used by IDMS.

The TPS-IDMS interface routine does not check the logic of calls to the IDMS package except to ensure that:

- a READY is issued before any other IDMS request is issued
- a FINISH is issued before the end of a message pair

A thread using IDMS that ends without a FINISH will be aborted by TPS.

2.1.1 Threading within Database Calls

It is important that designers should realise that, even if the TPS program is a multi-threading one, it is only possible to single-thread within database calls. In fact once one thread has made its initial call to the database handler, then no other thread will be allowed access to the database until either the completion of the first thread (if the database was written to), or the FINISH was issued (if the database was read-only accessed).

2.1.2 Currency Checking

If a transaction retrieves records in one message pair and then needs access to those records in a subsequent message pair, it should store the database keys of the records in the TCR during the first message pair, and then retrieve the records again at the start of the next. It is critical that the records retrieved in this way establish all the IDMS currencies required to process the message correctly.

The current versions of IDMS leave currencies set between messages. Thus it is that testing which only simulates a single user situation can appear to work correctly, but with multiple users serious

database corruption can occur. This is because the testing situation described is making use of currencies established in the previous message pair. These records will not be locked between message pairs, and so the final phase of the transaction must check (where necessary) that no unexpected changes have occurred to the record owing to the activities of other terminals.

2.2 SYSTEM START-UP

The TPS start-up procedure is modified in order to include two extra application routines. Both these application routines must be written in COBOL by the user, and are incorporated into the TPS program and included within the standard TPS Start-up AR Train via the relevant implementation macros (see Section 4.1).

Examples of these two ARs before input to the IDMS pre-processor are given in Appendix A.

2.2.1 Establishing the IDMS Area

The function of the first application routine is to set up the IDMS area in WORKING STORAGE SECTION and to pass the address of this area to TPS so that other ARs may access this area via LINKAGE SECTION. This routine must be named DBAREAS (so that it will be invoked in the standard Start-up AR Train).

The area is set up in Working Storage (as distinct from LINKAGE SECTION) so that it may have real existence in store and may hold pre-set values where relevant; other ARs will access it through a description in LINKAGE SECTION.

So that TPS can pass the address of the real area to the 'using' ARs, the DBAREAS AR must first pass it to TPS. This is done by incorporating the following CALL into the PROCEDURE DIVISION of DBAREAS:

```
CALL TPSIDMSAREA USING area-name.
```

DBAREAS must be a permanent AR, to ensure that the database area is in store when required. It may also be appropriate to use this AR to issue File Commands (as described in the IDMS DML Manual). The functions performed by these commands may be omitted (defaulting to options in the Schema definition) or instigated from a terminal (using the standard function to issue file commands (ZIDF), as described in Appendix C). If neither of these approaches is desirable, the functions can be performed via calls to IDMSFILEPAR made within user logic.

2.2.2 Initialising the Database

The function of the second application routine is to initialise the database by using the appropriate BIND statements and to carry out any other database initialisation required by the user. This routine must be named DBSTART (so that it will be invoked in the standard Start-up AR Train). It will address the database area by LINKAGE SECTION, and therefore has the PROTOCOL line:

```
IDMS-RECORDS WITHIN LINKAGE
```

It may be convenient to use this AR to make initial accesses to data to ensure that the database has actually been opened successfully.

Any optional components of IDMS code can also be included or excluded from the program. In the BASIC version of IDMS, this can be done conveniently by including within DBSTART one of the calls described in the IDMS DML manual.

N.B: The option to exclude file commands should not be used since TPS expects these to be present. Unless a READ ONLY protocol is required, use of the IDMS roll-back, or 'quick before image' file (QBIF) is mandatory in a TPS environment. The TPS generation ensures that this option is included. The QBIF file must be handled via File Commands since the Schema definition does not recognise this file.

Note: The presence of the QBIF file suppresses the recording of before images on the IDMS Journal file as these are no longer required.

The main IDMS Options appropriate to a TPS user are journalising on disc, or no journalising.

Note: For SHAREABLE versions of IDMS, the selection of optional components is made when the DBM is compiled.

2.3 SYSTEM CLOSEDOWN

The TPS closedown procedure is modified in order to include an extra application routine. This extra routine is supplied by TPS as a part of the IDMS interface and calls an IDMS routine to close the IDMS journal file correctly. This routine will have been incorporated into the TPS program as described in Section 2.2. To be included within the standard TPS Closedown AR Train, the parameter IDMS must be inserted on the relevant implementation macro (see Section 4.1).

2.4 APPLICATION ROUTINE INTERFACE

Application routines are written as normal in COBOL, but they may contain IDMS statements. These IDMS statements will be expanded by the IDMS pre-processor and will automatically generate calls to the TPS interface routine. The re-entry indicator is set up by the IDMS pre-processor; the first call to an IDMS routine will set the re-entry indicator to 1, the second call will set the re-entry to 2 and so on.

If it is required to make calls to TPS other than to the IDMS interface, these calls must use re-entry points with higher values than those used by the IDMS calls unless using the direct interface. Further, it is not possible to make a database request and a transfer to another application routine in the same call to TPS; this must be done by a separate call to TPSCNXT.

Further details on writing ARs are included in Section 3.

2.5 ABORTING MESSAGES

If for any reason (either a Minor Logic Error or the application routine calling TPSCULE2), a transaction is aborted, TPS will issue a call to the IDMS package which will initiate the Automatic Roll Back facility. This is consistent with the normal TPS abort procedure where all updates carried out previously by the transaction are reversed. This is accomplished by incorporating a TPS-supplied AR in the Abort AR Train. This routine will have been incorporated into the TPS program as described in Section 2.2. To be included within the TPS Standard Abort AR Train, the parameter IDMS must be inserted on the relevant implementation macro (see Section 4.1).

Note: In the event of a Minor Logic Error, the first 50 words of the Schema Control Area are journalised in addition to the usual journalisation.

It is not required to issue a FINISH prior to a call of TPSCULE2, since TPS will do this as a normal part of abort procedures.

2.6 RECOVERING FROM FILE FAILURE

As the database files are completely handled by IDMS, TPS will not be able to take any action in the event of a failure of a database file. This must be handled by the normal IDMS dump and Journal System.

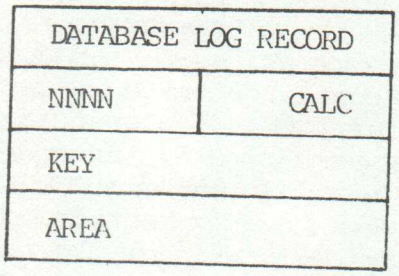
2.7 RECOVERING FROM SYSTEM FAILURE

2.7.1 Warm Restart

The normal warm restart options are available to TPS programs incorporating the TPS-IDMS interface. If the Reprocess Input Log option is specified, then the IDMS database must be restored to its start-of-day state before the warm restart is commenced. If the Reverse or Reverse and Reprocess options are specified, then any IDMS transactions will be reversed using the Automatic Roll Back Facility. Naturally, transactions to be treated in this manner must be defined as 'clean after restart' in the System Implementation. As TPS effectively single-threads database operations, there can only ever be one current transaction involving IDMS calls.

2.7.2 Database Security

The security of the database can now depend on a log record written to the database by the TPS program. The log record should be defined as part of the user database as follows:



The record details are:

```
Key.
     Program-name    PIC X(4)
     Channel         PIC X(4)   (zeroes)

Data.
     MSN             PIC 9(12)  COMP SYNC
     TERMNO          PIC 9(6)   COMP SYNC
```

The user written routine DBSTART should be amended to pass to TPS the addresses of the name and data area of the database log record. This is done by calling the following routine in DBSTART:

```
CALL TPSIDMSREC USING SRnn
                      dataname
```

where SRnn is the IDMS record identifier as defined by the user installation.

3. WRITING APPLICATION ROUTINES

3.1 GENERAL

Application routines to be incorporated in a TPS-IDMS program should be written to the standards and conventions for normal TPS ARs.

If the IDMS database is to be accessed in an application routine, the IDMS statements should be included and the source should be processed by the IDMS pre-processor DMLC before being compiled.

3.2 SETTING UP THE TPSIDMS PROTOCOL

Before any COBOL/DML modules for use with TPS are translated, the CLUC utility must be run to supplement the standard issue BATCH CLUC data by the addition of the TPSIDMS protocol and versions of the IDMS-STATUS and IDMS-STATISTICS procedure for use with this protocol.

```
+T COBOL        TPSIDMS
                MOVE @DMLSEQ TO DML-SEQUENCE
                MOVE @OCCUR  TO RECORD-OCCUR
                MOVE @PROG   TO PROGRAM-NAME
                CALL "@SS"
**              ENTER IDMS "TPSIDMS" USING @FUNC1
                                           @SSCTRL
                                           @SSNAME
                                           @REC
                                           @SET
                                           @REALM
                                           @DATANAME
                                           @RECBUF
                                           @OCCUR2
                                           @FUNC2
                IF DML-SEQUENCE = ZERO NEXT SENTENCE
                ELSE GO TO EXIT-PARA.

IDMS-@DMLSEQ.

*  TPS IDMS RE-ENTRY POINT @DMLSEQ

    IF ERROR-STATUS EQUAL TO @STATUS

+C COBOL        IDMS-STATUS TPSIDMS
```

** N.B: If RANGE COBOL is used, Subschema names of less than 8 characters will not compile (the IDMS pre-processor pads them out with trailing spaces). If names of less than 8 characters are to be used in these circumstances, then CALL "@xx" must be replaced by ENTER IDMS "@xx".

What follows is identical to the corresponding portion of the standard BATCH protocol. This data is most easily prepared by editing the issued CLUC data. In non-George 3 environments the issued CLUC data has a language code of FORTRAN, and hence may be edited using the TPS IDS Editor, or, for those who do not have IDS, XMED.

It is also necessary to change the picture of DML-SEQUENCE to be PIC 9(4).

It should be noted that some IDMS verbs are actioned in-line and others require an exit to TPS (optionally via the direct interface) and that DML-SEQUENCE is set to zero if no EXIT PROGRAM is required.

3.3 ENVIRONMENT DIVISION

All application routines that are to include DML statements should include the following section in their ENVIRONMENT DIVISION:

```
IDMS-CONTROL SECTION.
PROTOCOL.
        MODE IS TPSIDMS DEBUG
        IDMS-RECORDS WITHIN LINKAGE
        LEVELS INCREMENTED BY 1.
```

3.4 DATA DIVISION

The following statements should be included at the start of the DATA DIVISION:

```
SCHEMA SECTION.
DB                              (DB statement as described in the ICL
                                IDMS COBOL manual)
```

The LINKAGE SECTION should contain all the TPS areas defined in the Control Routine. The entry for the IDMS area should be as follows:

```
01   IDMS-AREA.
```

The IDMS pre-processor DMLC will then insert all the definitions of the IDMS records etc. into the LINKAGE SECTION at this point. It is usually more convenient if this is the last entry in the LINKAGE SECTION.

3.5 PROCEDURE DIVISION

It is strongly recommended that the direct interface, TPSCCONTROL, be used with IDMS application routines. This will avoid any problems that may arise when mixing TPSIDMS calls with other TPS calls within the same application routine.

In addition, the GO TO ..... DEPENDING statement may be omitted, thus eliminating the need to check that the correct DML-SEQUENCE and paragraph names have been generated.

Users who continue to use the indirect interface must ensure that the first statement is a GO TO .... DEPENDING. However, because of the way in which DML commands are expanded, it should be of the form:

```
GO TO  IDMS-0001
       IDMS-0002
            etc.
       DEPENDING ON re-entry
```

There must be one entry for each DML command in the AR. If there are calls to TPS other than those generated by DML commands, these should use re-entry points after those used by IDMS. For example, if an AR contains three DML commands, the first call to a TPS routine should use a re-entry parameter of 04. The fourth label in the DEPENDING clause will then indicate the appropriate procedure, which may of course logically precede those entry points indicated by the IDMS supplied labels.

3.6 DML COMMANDS

The DML commands used in an Application Routine must always be executed unconditionally - they must not follow an IF in the same sentence. The IDMS pre-processor DMLC starts a new paragraph after each such statement, so the logic of the AR must not be affected by this. If the DML command is always in a sentence by itself, it will always translate correctly. A DML IF (IF set EMPTY or IF set MEMBER) will translate satisfactorily.

Note: A DML command must not be part of a PERFORMed paragraph or section.

EXAMPLES

The DML command:

```
FIND CALC PRODUCT.
```

(if it is the third DML command in the AR) will generate:

```
*    FIND CALC PRODUCT
     MOVE 0003 TO DML-SEQUENCE
     ENTER "TPSIDMS" USING 2032
                           SR631.
     IF DML-SEQUENCE = ZERO NEXT SENTENCE
     ELSE GO TO EXIT-PARA.

IDMS-0003.

* TPS IDMS RE-ENTRY POINT 0003
```

Similarly, the sentence:

```
IF ITEM EMPTY GO TO ORD-EMPTY.
```

becomes (if it is the seventh DML command in the program):

```
*    IF ITEM EMPTY
     MOVE 0007 TO DML-SEQUENCE
     ENTER "TPSIDMS" USING 2064
                           ITEM.
     IF DML-SEQUENCE = ZERO NEXT SENTENCE
     ELSE GO TO EXIT-PARA.

IDMS-0007.

*    TPS IDMS RE-ENTRY POINT 0007.
     IF ERROR-STATUS EQUAL TO 0000
     GO TO ORD-EMPTY.
```

However, a sentence like

```
IF  X > 0  FIND CALC PRODUCT
    GO TO LABEL.
```

will clearly translate to give results different from the programmer's intentions.

3.7 ERROR-STATUS HANDLING

The programmer should note that only the ERROR-STATUS values listed below are ever returned to the Application Routine - all others are handled by the system software and will normally cause Minor Logic Error 2760.

1601 (response to IF - only accessed by generated COBOL).

0705 CONNECT)
) Attempting to violate a DUPLICATES NOT ALLOWED
0805 MODIFY) restriction.
)
1205 STORE)

0307 FIND/OBTAIN: End of set or realm

0326 FIND/OBTAIN: Record not found

0230 ERASE attempted when record still owns others.

The error reply giving rise to 2760 can be seen by looking at the relevant DB entry for the IDMS request.

4. SYSTEM IMPLEMENTATION FOR TPS-IDMS

4.1 GENERATION FORMS

Include the parameter IDMS on the following macros:

£TCAY (Form 1) - to generate AR calls

£TCTY (Form 3) - to generate AR Trains (should go on both lines of Section 2, on Section 6 and on Section 7).

£TCMY (Form 23a) - message type table entry

N.B: If any of the ARs are written in RANGE COBOL, then IDMS should not be included on Form 1, but all the IDMS ARs must be entered longhand on Form 2, with those written in RANGE COBOL carrying the additional parameter RAR.

Include the following macro in Section 1 of Form 10.2:

£TCDI - to set IDMS options

To include the IDMS area in the calling sequence, insert the parameter TPSIDMSADD on the £TCCU macro on Form 25.

4.2 MODIFYING STANDARD TPS AR TRAINS

If users wish to modify the Standard TPS AR Trains for start-up, closedown or abort, the following amendments should be made to the relevant Trains:

Cold Start Only	- Include DBAREAS, DBSTART and TPSIDMSCHK immediately before TPSSETSF.
Cold start & Warm Restart	- Include DBAREAS, DBSTART and TPSIDMSCHK twice, i.e. immediately before each occurrence of TPSSETSF.
Close-Down	- Include TPSIDMSCLS immediately preceding TPSCLOSE3.
Abort	- Include TPSIDMSABT immediately preceding TPSABORT2.

Note: Users who wish to include their own routines within standard TPS AR Trains should macro-expand the appropriate £TCTY macro, via utility #TPSA or #TPSB, to list the standard AR Train before defining the Train longhand in their user system.

4.3 CONSOLIDATION

To consolidate a TPS-IDMS program, the following components are required in the order indicated:

*IN ED	TPS Master Routine	
*LIB ED	Application Routines	
*LIB ED	TPS Libraries	
*IN ED	Semicompiled DMCL	) **
*IN ED	Semicompiled Subschema (BASIC only)	)
*LIB ED	IDMS Library	
*LIB ED	COBOL Library	

** These particular routines must be compiled as follows:

#PROGRAM xxxxnn/CONTROL(15AM,22AM,DBM,EBM)

The IDMS Library used will depend on the machine, as follows:

1900/2900 DME		- SUBGROUPIDMS, release 3.5 or higher, including the optional extra roll-back facility
ME29	BASIC	- IDMSLIB
	SHAREABLE	- IDMSSHARLIB

4.4 IDMS FORMATS

The incorporation of the standard function to issue file commands (ZIDF - see Appendix C) is performed through System Definition. Enter in the keyword.qualifier field the component name 'FORMAT.ZIDF', and 'S' in the Special Directive field. The ZIDF format will be displayed for confirmation, and pressing HOME and SEND will incorporate this standard function into the System Definition file.

Note: ZIDF is not available in SHAREABLE IDMS systems.

4.5 NOTE FOR ME29 USERS

To load and/or open the database for MTS access, users should run the OPEN-DATABASE procedure, as follows:

OPDB(databasename)

This procedure can be run at any time, but users are recommended to include it in their TPS load procedures.

PART TWO

IDMS INTERFACE FOR VM SYSTEMS

1. SUMMARY

1.1 IDMS FACILITIES

The TPS-IDMS interface provides the following facilities:

- The ability to initialise and close down the database in the TPS Start-up and Closedown AR Trains.
- Control of access to IDMS from a TPS application routine.
- Full warm restart in the event of system failure.
- The ability to 'roll back' database transactions in the event of a message being aborted.
- The automatic 'roll back' of database transactions when using a terminal in training mode.

1.2 IDMS OPTIONS

The following options are available:

- Access to and recovery of the database appearing to IDMS as a 'TP virtual machine'. This option relies on TPS to co-ordinate recovery with IDMS and thus requires the database to be unloaded during warm restart.
- Access to and recovery of the database appearing to IDMS as a 'batch virtual machine' with a log record defined within the schema. This option should be used if the database is to be shared amongst various TP services.

2. SYSTEM DESIGN CONSIDERATIONS

Regardless of which IDMS option is in use, three SCL procedures must be written and compiled into a library which is assigned with execute access at TPS load time. These procedures are called by TPS during initialisation and closedown, and are as follows:

```
TPS_IDMS_START  )
                ) - at start-up
TPS_IDMS_RUN    )

TPS_IDMS_CLOSE    - at closedown
```

2.1 SYSTEM START-UP & CLOSEDOWN

2.1.1 SCL Procedure TPS_IDMS_START

Pre IDMS 400, this procedure is called during start-up in order to load the IDMS service. It is provided for those users who are using a shared service and have chosen the 'TP virtual machine' option; it should contain a RUN_JOB command to initiate an 'IDMSSTARTUP' procedure call within a batch VM. This batch VM then becomes the IDMS service VM.

In all other cases pre IDMS 400, this procedure should simply log a message to the journal using 'SMSG' and then exit.

For IDMS 400, the TPS_IDMS_START procedure is not called, even if the 'TP virtual machine' option is being used. It must, however, be loaded, since it is not known until start-up time which version of IDMS (i.e. pre 400 or 400) is being used. Thus, for IDMS 400, a dummy TPS_IDMS_START hook must exist.

Note: For IDMS 400, the IDMS service VM must be started (using TPS_IDMS_START or START_DB_SERVICE) prior to the TPS service.

2.1.2 SCL Procedure TPS_IDMS_RUN

This procedure is called during start-up to connect the TPS VM to IDMS. It will also start an unshared service. It should contain a call either to 'IDMS(X)RUN' if using a shared service, or to 'IDMS(X)UNSHAREDRUN' if using an unshared service. The following parameters to either of these commands are mandatory:

SERVICE - identifies the service description to be used.

SUBSCHEMANAME - identifies one or more sub-schemas to be loaded and prepared for use.

CONTINGENCY - should be set to 'NO'.

N.B: If using an unshared IDMS service, a call to 'IDMS(X)FILES' must precede the 'IDMS(X)UNSHAREDRUN' command.

2.1.3 SCL Procedure TPS_IDMS_CLOSE

This procedure gives the user the option to close IDMS each time TPS is closed cleanly. It should contain a call either to 'IDMS(X)CLOSEDOWN' if using a shared service, or to 'IDMS(X)ENDRUN' if using an unshared service.

If this option is not required, the procedure should simply log a message to the journal using 'SMSG' and then exit.

2.2 AMENDMENTS TO START-UP AR TRAIN

If the 'TP virtual machine' option is being used, no amendments to the TPS Start-up AR Train are required. However, if a single sub-schema is being used, it is advisable to include an extra application routine to issue the BIND RUN UNIT and BIND RECORD IDMS calls to save having to call them in each message pair. The area to hold the sub-schema control block must be part of the Worksheet (e.g. dynamic store attached to a fixed store link). For the position in the Start-up AR Train in which to insert this AR, see below.

If the 'batch virtual machine' option is being used, an extra AR is required for start-up to issue the BIND RUN UNIT and BIND RECORD IDMS call for the TPS log record. However, if a single sub-schema is in use, this AR should include all the required BIND RECORD IDMS calls as discussed for the 'TP virtual machine' above. An example of such an AR is given in Appendix B.

Regardless of which option is being used, the extra AR should be inserted immediately after TPSSTART4 in cold start and TPSWARM9 in warm restart. The name of the AR Train component as defined to System Definition is 'ART.TPSARTCOLD'. For details of the ART component, see the **System Building Procedures** manual.

2.3 INSTALLING THE TPS-IDMS DATA MODULE

If the 'TP virtual machine' option is being used, the file ICL8TCTPSIDMSDATA must be copied into :SYSTEM.STANDARD.ICLSTDSW (LIBRARY 3) to install the TPS-IDMS Data Module.

2.4 IDMS_PARAMS

Pre IDMS 400, if the 'TP virtual machine' option is being used, this procedure is required and is called during warm restart. It should contain a call to 'IDMSFILES' for each file used by the IDMS interface. The IDMS service name as specified to System Definition is passed as a parameter to this procedure, and a response parameter is returned from it.

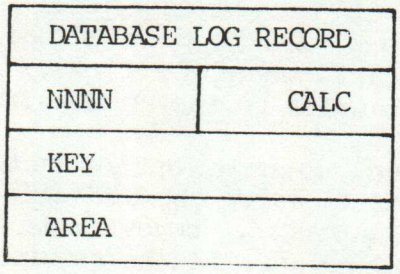
For IDMS 400, this procedure need not be provided.

Note: For examples of IDMS_PARAMS, see the IDMS documentation.

2.5 RECOVERING FROM SYSTEM FAILURE

If the 'TP virtual machine' option is being used, recovery is achieved by TPS passing to IDMS a list of committed phase identifiers, and IDMS will roll back any unfinished success units. Before this can be achieved the database must be closed.

If the 'batch virtual machine' option is being used, the security of the database depends on a log record written by TPS. The log record should be defined as part of the user database as follows:



The record details are:

```
Key.
     Service-name    PIC X(4).
     Channel         PIC X(4).
Data.
     MSN             PIC S9(18) USAGE COMP.
     TERMNO          PIC S9(8)  USAGE COMP.
```

The name of the log record as defined to the database schema must be the same as the one defined to the IDMS component of System Definition.

It should be noted that all contingencies that are allowed to be trapped are handled by TPS and, if required, the message is rolled back and all locks are released. However, as the 'ABANDON VM' contingency is masked, this command should be avoided, since in certain cases database locks could be left set.

IDMS INTERFACE FOR VM SYSTEMS

3. WRITING APPLICATION ROUTINES

3.1 GENERAL

Application routines to be incorporated in a TPS-IDMS interface should be written to the standards and conventions for normal TPS ARs. If the IDMS database is to be accessed in an AR, the IDMS statements should be included and the source should be processed by the IDMSDMLC procedure in order to call the DMLC pre-processor before compiling the AR.

3.2 SETTING UP THE TPSIDMS PROTOCOL

In order for the IDMS CLUC utility to set up correctly the IDMS directives for the translation of COBOL/DML programs for use with TPS, the standard CLUC data supplied by ICL must be amended. This amendment introduces the protocol, TPSIDMS, and versions of the IDMS-STATUS and IDMS-STATISTICS procedures for use with this protocol.

```
+T COBOL            TPSIDMS
                     MOVE @DMLSEQ TO DML-SEQUENCE
                     MOVE @OCCUR TO RECORD-OCCUR
                     MOVE @FUNC1 TO TPS-DML-ID
                     CALL "TPSIDMS" USING IDBMSCOM (@FUNC1)
                                          @SSCTRL
                                          @SSNAME
                                          @REC
                                          IDBMSCOM (@DUMMYR)
                                          @SET
                                          @REALM
                                          @DATANAME
                                          @DATA2
                                          @DATA3
                                          @RECBUF
                                          @OCCUR2
                                          IDBMSCOM (@DUMMY1)
                                          IDBMSCOM (@FUNC2)
                                          TPS-DML-ID
                      IF ERROR-STATUS EQUAL TO @STATUS

+C COBOL          IDMS-STATUS     TPSIDMS
   *    ********************************************************
    IDMS-STATISTICS SECTION.
   *    ********************************************************
    ICL8TC-STATS-ENTRY.
        ACCEPT STATS-WORK-AREA FROM IDMS-STATISTICS.
    ICL8TC-STATS-EXIT.
        EXIT.
   *    ****************************************
```

```
 IDMS-STATUS SECTION.
*    ****************************************
 ICL8TC-STATUS-ENTRY.
     IF ERROR-STATUS EQUAL TO ZEROS OR "1601"
         GO TO ICL8TC-STATUS-EXIT.
     MOVE ERROR-STATUS TO ERROR-STATUS-VAL.
     IF DB-CALL-ABORT
         ADD 1 TO DB-ABORT-CALL-NO
         PERFORM IDMS-ABORT.
     SUBTRACT 1 FROM DB-ABORT-CALL-NO.
     CALL "ICLSTDSW.ICL9IDMSFLAG" USING ERROR STATUS-VAL.
* Replace nnnn by the required user logic error code in
* the range 7000 - 7776.
     MOVE nnnn TO TPS-IDMS-LE.
     CALL "TPSCULE2" USING TPS-IDMS-LE.
 ICL8TC-STATUS-EXIT.
     EXIT.
```

Two fields must also be added to SUBSCHEMA-CTRL, such that the FILLER field immediately following SUBSIDIARY-ERROR is replaced by the following:

```
04  TPS-IDMS-LE     PIC S9(8) COMP SYNC.
04  TPS-DML-ID      PIC S9(8) COMP SYNC.
04  FILLER          PIC X(16).
```

BIND Run Unit

If multiple sub-schemas within the database are to be shared within TPS, it is required to issue BIND RUN UNIT and BIND RECORD statements for each message pair, as any TPS virtual machine is capable of processing any message. The TPS-IDMS interface now traps each BIND RUN UNIT call and, if the sub-schema name and control block address are the same as the previous BIND RUN UNIT processed by that VM, return is made immediately to the AR with an ERROR-STATUS value of zeros.

3. WRITING APPLICATION ROUTINES

3.1 GENERAL

Application routines to be incorporated in a TPS-IDMS interface should be written to the standards and conventions for normal TPS ARs. If the IDMS database is to be accessed in an AR, the IDMS statements should be included and the source should be processed by the IDMSDMLC procedure in order to call the DMLC pre-processor before compiling the AR.

3.2 SETTING UP THE TPSIDMS PROTOCOL

In order for the IDMS CLUC utility to set up correctly the IDMS directives for the translation of COBOL/DML programs for use with TPS, the standard CLUC data supplied by ICL must be amended. This amendment introduces the protocol, TPSIDMS, and versions of the IDMS-STATUS and IDMS-STATISTICS procedures for use with this protocol.

```
+T COBOL            TPSIDMS
                     MOVE @DMLSEQ TO DML-SEQUENCE
                     MOVE @OCCUR TO RECORD-OCCUR
                     MOVE @FUNC1 TO TPS-DML-ID
                     CALL "TPSIDMS" USING IDBMSCOM (@FUNC1)
                                          @SSCTRL
                                          @SSNAME
                                          @REC
                                          @IDBMSCOM (@DUMMYR)
                                          @SET
                                          @REALM
                                          @DATANAME
                                          @DATA2
                                          @DATA3
                                          @RECBUF
                                          @OCCUR2
                                          IDBMSCOM (@DUMMY1)
                                          IDBMSCOM (@FUNC2)
                                          TPS-DML-ID
                       IF ERROR-STATUS EQUAL TO @STATUS

+C COBOL          IDMS-STATUS     TPSIDMS
      *    ************************************************
       IDMS-STATISTICS SECTION.
      *    ************************************************
       ICL8TC-STATS-ENTRY.
           ACCEPT STATS-WORK-AREA FROM IDMS-STATISTICS.
       ICL8TC-STATS-EXIT.
           EXIT.
      *    ****************************************
```

```
 IDMS-STATUS SECTION.
*    ****************************************
 ICL8TC-STATUS-ENTRY.
     IF ERROR-STATUS EQUAL TO ZEROS OR "1601"
         GO TO ICL8TC-STATUS-EXIT.
     MOVE ERROR-STATUS TO ERROR-STATUS-VAL.
     IF DB-CALL-ABORT
         ADD 1 TO DB-ABORT-CALL-NO
         PERFORM IDMS-ABORT.
     SUBTRACT 1 FROM DB-ABORT-CALL-NO.
     CALL "ICLSTDSW.ICL9IDMSFLAG" USING ERROR STATUS-VAL.
* Replace nnnn by the required user logic error code in
* the range 7000 - 7776.
     MOVE nnnn TO TPS-IDMS-LE.
     CALL "TPSCULE2" USING TPS-IDMS-LE.
 ICL8TC-STATUS-EXIT.
     EXIT.
```

Two fields must also be added to SUBSCHEMA-CTRL, such that the FILLER field immediately following SUBSIDIARY-ERROR is replaced by the following:

```
04  TPS-IDMS-LE      PIC S9(8) COMP SYNC.
04  TPS-DML-ID       PIC S9(8) COMP SYNC.
04  FILLER           PIC X(16).
```

3.3 BIND RUN UNIT

If multiple sub-schemas within the database are to be shared within TPS, it is required to issue BIND RUN UNIT and BIND RECORD statements for each message pair, as any TPS virtual machine is capable of processing any message. The TPS-IDMS interface now traps each BIND RUN UNIT call and, if the sub-schema name and control block address are the same as the previous BIND RUN UNIT processed by that VM, return is made immediately to the AR with an ERROR-STATUS value of zeros.

3.4 ERROR-STATUS HANDLING

The programmer should note that only the ERROR-STATUS values listed below are ever returned to the application routine. All others are handled by the system software and will normally cause Minor Logic Error 2760.

1601 (response to IF - only accessed by generated COBOL).

0705 CONNECT)
) Attempting to violate a DUPLICATES NOT ALLOWED
0805 MODIFY) restriction.
)
1205 STORE)

0307 FIND/OBTAIN: End of set or realm.

0326 FIND/OBTAIN: Record not found.

0230 ERASE attempted when record still owns others.

The exceptions are minor codes 72 (timeout (IDMS 400)), 93 and 94 (deadlock and dual update), in which cases the message is rolled back and retried up to 3 times by the system; the application is not marked as unavailable. If the attempt is still unsuccessful, a message is displayed at the sending terminal inviting the user to re-submit the transaction. (If it is required to redefine the number of retries, or to alter the text of the message, contact TPS Support for details.)

The error reply giving rise to LE 2760 can be seen by looking at the relevant DB entry for the IDMS request.

APPENDIX A

EXAMPLE START-UP ARs FOR DM

These examples contain the source that should be input to the IDMS pre-processor. The LINKAGE SECTION will obviously depend upon that set up within the user's TPS program.

A.1 DBAREAS : Application routine to set up IDMS area in Working Storage and pass the address of this area to TPS.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DBAREAS.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  ICL-ME29.
OBJECT-COMPUTER.  ICL-ME29.
IDMS-CONTROL SECTION.
PROTOCOL.
     IDMS-RECORDS WITHIN WORKING-STORAGE
                  LEVELS INCREMENTED BY 1.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
SCHEMA SECTION.
DB SAMPSUBS WITHIN SAMPSCHM.
WORKING-STORAGE SECTION.
    01 IDMS-AREA-DEF.
LINKAGE SECTION.
    01 LA-REPLY                    PIC X(4).
    01 LB-REENTRY                  PIC 9(6) COMP SYNC.
    01 LC-MESSAGE.
       03 FILLER                   PIC X(4).
    01 LD-TCR                      PIC X(4).
    01 LE-ASA1.
       03 NXT-PAR                  PIC 9(6) COMP SYNC.
    01 IDMS-AREA                   PIC X(4).
    PROCEDURE DIVISION USING LA-REPLY
                             LB-REENTRY
                             LC-MESSAGE
                             LD-TCR
                             LE-ASA1
                             IDMS-AREA.

    START-LABEL.

        CALL "TPSCCONTROL".

        CALL "TPSIDMSAREA" USING IDMS-AREA-DEF.

        MOVE 1 TO NXT-PAR.

        CALL "TPSCNXT" USING NXT-PAR.
****
```

EXAMPLE START-UP ARs FOR DM

A.2 DBSTART

Application routine to issue BIND commands to the IDMS data base.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  DBSTART.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  ICL-ME29.
OBJECT-COMPUTER.  ICL-ME29.
IDMS-CONTROL SECTION.
PROTOCOL.
       MODE IS TPSIDMS DEBUG
       IDMS-RECORDS WITHIN LINKAGE
       LEVELS INCREMENTED BY 1.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
SCHEMA SECTION.
DB SAMPSUBS WITHIN SAMPSCHM.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
   01 LA-REPLY                  PIC X(4).
   01 LB-REENTRY                PIC 9(6) COMP SYNC.
   01 LC-MESSAGE.
      03 FILLER                 PIC X(4).
   01 LD-TCR                    PIC X(4).
   01 LE-ASA1.
      03 NXT-PAR                PIC 9(6) COMP SYNC.
   01 IDMS-AREA.

   PROCEDURE DIVISION USING LA-REPLY
                            LB-REENTRY
                            LC-MESSAGE
                            LD-TCR
                            LE-ASA1
                            IDMS-AREA.

   START-LABEL.

       CALL "TPSCCONTROL".

       COPY IDMS SUBSCHEMA-BINDS.

       MOVE 1 TO NXT-PAR.

       CALL "TPSCNXT" USING NXT-PAR.
****
```

APPENDIX B

EXAMPLE START-UP AR FOR VM

Application routine to issue BIND commands to the IDMS data base.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  TPSIDMSBINDS
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  ICL-ME29.
OBJECT-COMPUTER.  ICL-ME29.
IDMS-CONTROL SECTION.
PROTOCOL.
       MODE IS TPSIDMS DEBUG
       IDMS-RECORDS MANUAL
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
SCHEMA SECTION.
DB SAMPSUBS WITHIN SAMPSCHM.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
    01 LA-REPLY                    PIC X(4).
    01 LE-REENTRY                  PIC 9(6) COMP SYNC.
    01 LC-MESSAGE.
       03 FILLER                   PIC X(4).
    01 LD-TCR                      PIC X(4).
    01 LE-ASA1.
       03 NXT-PAR                  PIC 9(6) COMP SYNC.
    01 IDMS-AREA.
      02 COPY IDMS SUBSCHEMACTRL.

    PROCEDURE DIVISION USING LA-REPLY
                             LB-REENTRY
                             LC-MESSAGE
                             LD-TCR
                             LE-ASA1
                             IDMS-AREA.


    START-LABEL.

        CALL "TPSCCONTROL".

        COPY IDMS SUBSCHEMA-BINDS.

        MOVE 1 TO NXT-PAR.

        CALL "TPSCNXT" USING NXT-PAR.
****
```

APPENDIX C

STANDARD FUNCTION TO ISSUE FILE COMMANDS

(DM ONLY)

Identifier: ZIDF

Function: Issue a File Command to IDMS.

Format displayed in Response to ZIDF:

```
ZIDF]
    ENTER IDMS FILE COMMAND
   [                              ]
```

Action at the Terminal:

Enter ZIDF in the first four character positions of the screen. On receipt of the above format, the required File Command should be entered. For details of the File Commands, see the ICL manual IDMS COBOL Data Manipulation Language on 1900 Series Computers.

Action by the System on Receipt of the Request:

The system will check for Master Status if this has been specified as a requirement for this message type at System Implementation time. (This is the recommended option). The above format will be displayed.

Action by the System on Receipt of the Completed Screen:

The input File Command will be issued to IDMS. The database will be locked whilst the File Command is being carried out. If the database is already locked when this transaction is initiated, it will wait until the database is free.

It is not necessary to open the database files in this way. A file will be opened the first time it is accessed. However, ZIDF must be used to open both the Journal and Automatic Rollback files at start of day, unless this has been provided for in an application routine as suggested in Part 1, Section 2.2.

e.g.

```
ZIDF]
    ENTER IDMS FILE COMMAND
    [*JL, IDMSJOURNAL(DA6)      ]
```

APPENDIX D

SYSTEM DEFINITION FOR TPS-IDMS

D.1 KEYWORD IDMS

The incorporation of TPS-IDMS software into a user's system involves procedures to modify the system definition to cater for it. These procedures require the setting of various parameters interactively through keyword IDMS in System Definition, as follows:

SERVICE NAME:

TYPE: Item

Enter the name (up to 8 alpha-numeric characters) of the IDMS service to which the TPS system is to attempt connection. Mandatory if Recovery Type is STANDARD.

Note: For DM systems, this entry is purely documentary.

SERVICE TYPE:

TYPE: Selected Item

Enter one of the following:

BAS or BASIC	if the IDMS facilities are to be built into and used exclusively by the TPS system here being defined.
SHAR or SHAREABLE	if the IDMS facilities are to be provided by a separate program with which the TPS system here being defined (and possibly other programs) will communicate.

Note: For VM systems, this entry is ignored.

RECOVERY TYPE:

TYPE: Selected Item

Enter one of the following:

STD or STANDARD	if recovery is to be achieved by TPS passing to IDMS a list of committed phase identifiers, and IDMS rolling back any unfinished success units. Selection of STD makes Service Name mandatory.

CHKP or CHECKPOINT — if the security of the database is to depend on a log record written by the TPS system. The log record must be defined as part of the user database and its name must be entered in the Checkpoint Name parameter.

CHECKPOINT NAME:

TYPE: Item

If Recovery Type is specified as CHECKPOINT, then enter the name (up to 16 characters, the first alphabetic, the rest alpha-numeric) of the TPS log record as defined in the IDMS Schema; otherwise, leave this field blank.

D.2 NOTE ON FILE MANAGEMENT IN A DM SYSTEM

IDMS files should not be defined to TPS in any way. However, it is important that the unit reference numbers used by the IDMS system are higher than the number of physical files which satisfy all the following conditions:

- defined to TPS in keywords LOGICAL FILE and LF RESILIENCE
- listed in keyword FILE MANAGEMENT
- concurrently open

INDEX

A

Aborting Messages, **1-5**
Application Routines
  BIND Run Unit (TPS/ICL-VME), **2-6**
  DATA DIVISION, **1-8**
  DML Commands, **1-9**
  ENVIRONMENT DIVISION, **1-8**
  ERROR-STATUS Handling (TPS/ICL-DME), **1-11**
  ERROR-STATUS Handling (TPS/ICL-VME), **2-7**
  LINKAGE SECTION, 1-8
  PROCEDURE DIVISION, **1-9**
  TPSIDMS Protocol (TPS/ICL-DME), **1-7**
  TPSIDMS Protocol (TPS/ICL-VME), **2-5**
Application Routines (TPS/ICL-DME), **1-7**
Application Routines (TPS/ICL-VME), **2-5**
AR Interface (TPS/ICL-DME), **1-5**

B

BIND Run Unit (TPS/ICL-VME), **2-6**

C

Checkpoint Name, D-2
CLUC (TPS/ICL-DME), 1-7
CLUC (TPS/ICL-VME), 2-5
COBOL Interface Routines
  TPSCCONTROL, 1-9
  TPSCNXT, 1-5
  TPSCULE2, 1-5
COBOL Interface Routines (TPS/ICL-DME), **1-5**
Consolidation for IDMS, **1-13**
Currency Checking, **1-2**

D

DATA DIVISION, **1-8**
Database
  AR Access, 1-7
  Calls and Threading, **1-2**
  File Failure (TPS/ICL-DME), 1-5
  Initialisation, **1-4**
  Load/Open on ME29, 1-13
  Security, **1-6**
  System Failure (TPS/ICL-DME), 1-6
DB Entries, 1-11
DBAREAS, 1-3, 1-12, **A-1**
DBSTART, 1-4, 1-6, 1-12, **A-2**
DML Commands, **1-9**

E

ENVIRONMENT DIVISION, **1-8**
ERROR-STATUS Handling (TPS/ICL-DME), **1-11**
ERROR-STATUS Handling (TPS/ICL-VME), **2-7**

F

Facilities (TPS/ICL-VME - IDMS), **2-1**
Facilities (TPS/ICL-DME - IDMS), **1-1**
File Failure (TPS/ICL-DME), 1-5
FILE MANAGEMENT (FMAN), D-2

INDEX

F (cont'd)

Files
IDMS Journal, 1-4
Management (TPS/ICL-DME), **D-2**
QBIF, 1-4

G GO TO ... DEPENDING, 1-9

I IDMS (IDMS), **D-1**
IDMS Area, **1-3**
IDMS Commands
FINISH, 1-2, 1-5
READY, 1-2
IDMS Library (TPS/ICL-DME), 1-13
IDMS Library (TPS/ICL-VME), 2-3
IDMS Options (TPS/ICL-DME), **1-1**
IDMS Options (TPS/ICL-VME), **2-1**
IDMS_PARAMS, **2-3**
Initialising the Database, **1-4**
Installing TPS-IDMS Data Module, **2-3**

L LF RESILIENCE (LFR), D-2
Library
IDMS (TPS/ICL-DME), 1-13
IDMS (TPS/ICL-VME), 2-3
Log Record (TPS/ICL-DME), 1-6
Log Record (TPS/ICL-VME), 2-4
LOGICAL FILE (LF), D-2

M Messages
Aborting, **1-5**
Minor Logic Error, 1-5, 1-11, 2-7
Modifying TPS AR Trains for IDMS, **1-12**

O Options
IDMS (TPS/ICL-DME), **1-1**
IDMS (TPS/ICL-VME), **2-1**

P PERFORM, 1-9
PROCEDURE DIVISION, **1-9**

Q QBIF, 1-4

R Recovery
Database Security, **1-6**
from File Failure (TPS/ICL-DME), **1-5**
from System Failure (TPS/ICL-DME), **1-6**
from System Failure (TPS/ICL-VME), **2-4**
Warm Restart, **1-6**
Recovery Type, D-1

INDEX

S

SCL
  TPS_IDMS_CLOSE(), 2-1, **2-2**
  TPS_IDMS_RUN(), 2-1, **2-2**
  TPS_IDMS_START(), 2-1, **2-2**
Service Name, D-1
Service Type, D-1
System Closedown (TPS/ICL-DME), **1-4**
System Closedown (TPS/ICL-VME), **2-2**
System Definition for IDMS, **D-1**
System Design Considerations (TPS/ICL-DME), **1-2**
System Design Considerations (TPS/ICL-VME), **2-1**
System Failure (TPS/ICL-DME), 1-6
System Failure (TPS/ICL-VME), **2-4**
System Generation (TPS/ICL-DME), **1-12**
System Implementation for TPS/ICL-DME - IDMS, **1-12**
System Start-up
  Amending AR (TPS/ICL-VME), **2-3**
  Example AR for TPS/ICL-VME, **B-1**
  Example ARs for TPS/ICL-DME, **A-1**
  IDMS 6 Shareable, 1-1
System Start-up (TPS/ICL-DME), **1-3**
System Start-up (TPS/ICL-VME), **2-2**

T

Terminal Functions, **1-13**
  ZIDF, 1-3, 1-13, **C-1**
Terminal Functions (TPS/ICL-DME), **C-1**
Threading within Database Calls, **1-2**
TPS Batch Programs
  #TPSA, 1-12
  #TPSB, 1-12
TPSIDMS Protocol (TPS/ICL-DME), **1-7**
TPSIDMS Protocol (TPS/ICL-VME), **2-5**
TPSIDMSABT, 1-12
TPSIDMSCHK, 1-12
TPSIDMSCLS, 1-12
Training Mode, 1-1

U

Unit Reference Numbers for IDMS, 1-2