
TPS
Overlay
Optimiser

CONTENTS

1. INTRODUCTION

The Overlay Optimiser is designed to improve the performance of a TPS System by optimising the disc accesses required for the transfer of overlays. In a typical TPS program the majority of Application Routines (both TPS and user routines) are overlaid. Using the standard overlay package it is not exceptional for a COBOL Application Routine to require several (perhaps up to ten or more) transfers to bring it in; this may lead to a disproportionate number of disc accesses on the drive holding the Program File.

The run-time Optimiser package ensures that most overlay units are brought in with no more than two transfers, with small units coming in as a single transfer.

Run-time statistics are available to monitor the actual number of transfers taken for each overlay unit. In addition the Optimiser supports the use of a duplicate Program File so that overlay accesses may be split between two disc drives, and automatic fallback provided if one of the program files fails.

To achieve this, the optimiser package consists of the following elements:-

- a batch utility to convert the Program File to a more efficient format (this is described in Section 2)

- a batch utility to load the reformatted program into store (this is described in Section 3)

- run-time overlay software for inclusion into the TPS program (this is described in Section 4)

Details of the generation options required to incorporate the Overlay Optimiser into a TPS system are given in Section 5.

2. CONVERTING THE PROGRAM FILE

2.1 GENERAL DESCRIPTION

Some understanding of the structure of a standard Direct Access Program File is required in order to appreciate the necessity of converting the Program File in order to provide more efficient overlay handling.

An overlay unit of a binary program is held as a separate subfile on Direct Access devices. Each unit contains one or more organisational buckets and, optionally, one or more binary buckets. The binary buckets may be interspersed with the organisational buckets. In order that the standard run-time package may have a small buffer, only 32 words of each 128 word organisational bucket is utilised. Each section of data within a unit (e.g. lower variables, lower presets, program) is defined as either a "type 0" record, where the data is held within the organisational record itself or a "type 5" record which provides pointers to data held in binary buckets which may then require additional transfers to bring in at run-time. Any data section of more than 27 words is defined as a "type 5" record, owing to the restriction that only 32 words are used per bucket. An investigation of some COBOL Application Routine overlay units showed that most units consist of seven distinct sections of which only one, the program part, is of a size greater than about 100 words.

To meet the main objective of the Optimiser package, the reduction of disc accesses, a new format Program File called an Overlay File is produced. This contains the following features:

- a change to the file structure to allow full use of all words in the organisational buckets
- "type 0" records for data sections of more than 27 words are used
- "type 5" data binary buckets are held following all organisational buckets for the relevant unit.

The utility #TPSN is provided to convert a standard program file to the Optimised format in an Overlay File. The parameters required for the conversion are described below. Full operating instructions are given in the TPS Operating Manual Part 5.

2.2 ALLOCATING A FILE RESIDENCY ON DISC

A residency to hold the reformatted program (and optionally to hold a duplicate) must be allocated on disc.

2.2.1 File Identity

The Overlay File must be given a file name of the form TPSFASTOxxxx (TPSFASTDxxxx for a duplicate). 'xxxx' may be assigned any value valid within file names. The generation number allocated is not significant; the Overlay Optimiser Utilities always open the highest generation of the file online.

2.2.2 Bucket Size

A bucket size of 1, 2 or 4 blocks may be chosen; as a general rule the larger bucket sizes will result in more efficient disc overlaying at run-time. The most suitable bucket size will depend on the average size of overlay units, and the store available within the TPS program to hold the overlay buffer. As a guide, the average TPS COBOL Application Routine with a memory size of 2000 words may be contained in one 4-block organisation bucket and one binary data block (which will be read with multi-bucket transfers).

2.2.3 Size Of The File

The Overlay File will be extended (together with its duplicate) as required. It may therefore be any size other than zero length. However a rough guide for the size is that of the current Program File output by #XPCK.

2.2.4 Integrity Code

If the TFILE option is to be used the integrity code of the file (and duplicate if used) must be 2.

2.3 INPUT PARAMETERS

Control parameters defining the program conversion are input on cards.

Format Of The Cards

The first four characters of each card are the program identifier TPSN. These are followed (without space or separator) by a four-character section and sequence number starting from 0101.

Each item is identified by a Keyword, which in most cases has an alternative shortened form, and which may have an associated parameter. A parameter to a Keyword must be on the same card as the Keyword, separated from it by any number of spaces and from the following Keyword by a comma.

The first Keyword on a card follows the eight character header, separated by at least one space. No comma is needed after the last parameter on a card.

Any number of cards may be used and the Keywords may be defined in any order.

Parameters

1. PROGRAM or PROG

   This parameter is mandatory and specifies the name of the program to be converted.

   Format: PROG programname

   programname = 4 character name of the program to be converted.

2. PFNAME or PFN

   This parameter is optional and specifies the Input Program File to be used. If not present the name is taken from the programname specified for the PROG parameter.

   Format: PFNAME name

   name = Program file for the Input File. (i.e.file name is 'PROGRAM name'). The name specified may be up to 4 alphanumeric characters with the first character alphabetic.

3. OFNAME or OFN

This parameter is optional and specifies the Original Overlay File into which the program is to be converted. If not present the name is taken from the programname specified for the PROG parameter.

Format: OFNAME name

name = Name for the Original Overlay File (i.e.file name is 'TPSFASTOname'. The name specified may be up to 4 alphanumeric, space or hyphen characters.

4. DUPLICATE or DUPL

This parameter is optional and indicates that a duplicate Overlay File is to be created in the file TPSFASTDxxxx already allocated by the user where xxxx is the name defined by Keyword OFNAME.

Format: DUPLICATE

5. TFILE

This parameter is optional. If present it specifies the name of the *TABLES File output from XPCK. Its presence causes information to be read from the *TABLES File and added onto the Overlay File for use by TP3K. Also the CUELIST information taken from the *TABLES File is printed in two sorted listings on the line printer, one in address sequence and one in name sequence.

Format: TFILE filename

filename = up to 12 alphanumeric, space or hyphen characters with the first character alphabetic.

6. NOCUES or NOC

This parameter is optional. If it is present the CUE listing normally provided if TFILE is specified will be suppressed.

Format: NOCUES

7. STATISTICS or STAT

This parameter is optional. If it is present and TFILE is specified, statistics for overlay areas will be printed.

Format: STATISTICS

The statistics output are for use with TP3K and the description of output is given in Part 2 of this Manual.

8. PEND

This parameter is mandatory and indicates the end of parameter input.

Format: PEND

Examples

1. To convert program TPS1 from a Disc File PROGRAM TPS1 into an Overlay File TPSFASTOTPS1

```
TPSN0101  PROG TPS1, PEND
```

2. To convert program TPS2 from Disc File PROGRAM COMP into an Overlay File TPSFASTOLIVE with duplicate and sorted CUE list using *TABLES File TPS2TABLES.

```
TPSN0101  PROGRAM TPS2, PFNAME COMP
TPSN0102  OFNAME LIVE, DUPLICATE
TPSN0103  TFILE TPS2TABLES, PEND
```

3. LOADING THE REFORMATTED PROGRAM

3.1 GENERAL DESCRIPTION

The binary program in Optimised format is not compatible with standard loaders. A utility, #TPSO, is therefore provided to load the reformatted program. This utility also caters for the use of a duplicate Overlay File, so that overlay accesses may be split between two disc drives, and automatic fallback provided if one of the Overlay Files fails. If a duplicate Overlay File is to be used it must have been created by TPSN in the same run as the original Overlay File.

The loader program opens the Overlay File, and if required the duplicate. The permanent part of the program is loaded into core with the overlay table; the loader then destroys itself and halts with the standard HALTED:-LD. At this stage the program is the same as if it had been loaded by a standard loader (except that words 14-19 contain different values). The Overlay File is left assigned as unit 0, and the duplicate, if used, as unit 63.

The parameters required by the loader are described below. Full operating instructions for the loader #TPSO are given in the TPS Operating Manual, Part 5.

3.2 INPUT PARAMETERS

Control parameters specifying the program to be loaded are input on cards.

Format Of The Cards

The first four characters of the card are the program identifier TPSO. These are followed (without space or separator) by a four-character section and sequence number starting from 0101.

Each item is identified by a Keyword, which in most cases has an alternative shortened form, and which may have an associated parameter. A parameter to a Keyword must be on the same card as the Keyword, separated from it by any number of spaces and from the following Keyword by a comma.

The first Keyword on a card follows the eight character header, separated by at least one space. No comma is needed after the last parameter on a card.

Any number of cards may be used and the Keywords may be defined in any order.

Parameters

1. OFNAME or OFN

This parameter is mandatory and specifies the input Overlay File to be used.

Format: OFNAME name

name = Name for the Original Overlay File (i.e. file name is 'TPSFASTOname'). The name specified may be up to 4 alphanumeric, space or hyphen characters.

2. DUPLICATE or DUPL

This parameter is optional, required only if a duplicate Overlay File is to be used. The filename used is TPSFASTDxxxx where xxxx is the name specified for the OFNAME parameter.

Format: DUPLICATE

3. PEND

This parameter is mandatory and indicates the end of parameter input.

Format: PEND

Example

To load program TPS1 from TPSFASTOLIVE. The duplicate file is required.

```
TPS00101 OFNAME LIVE, DUPL, PEND
```

4. THE RUN-TIME PACKAGE

The run-time package consists of a single subroutine TPSOVHANDLR which replaces the standard overlay handling subroutine %EROLE.

The main features of the new package are:-

- most units may be brought in with two transfers: one for the organisational data and one multi-bucket transfer for the binary data (note that this is achieved where 4-block buckets are used for the program file).

- some small units will come in as one transfer if the total code is less than 500 words.

- the use of a duplicate Overlay File is supported. This is used for the binary data blocks whilst the main Program File is used for organisational data giving an approximate 50-50 split of transfers. If either file has a transfer fail the other is used for all transfers.

- run-time statistics are kept of the number of transfers per disc file.

- optionally, statistics are kept of the number of transfers required to bring in a unit.

The run-time package requires a buffer of the bucket size of the new program file. In addition an area may be allocated to hold the statistics for each unit. These areas are set up by Generation Parameters as described in Section 5 below. Details of the retrieval of the statistics maintained by the Overlay Optimiser are given in the Programming Manual, Part 1, Section 3.17.5.

5. GENERATING A TPS SYSTEM INCLUDING THE OVERLAY OPTIMISER

GENERATION PARAMETERS

The Generation Forms should be completed as described in the System Generation Manual. The following amendments are required to the standard generation parameters in order to include the Overlay Optimiser.

Form 5

To set up an overlay buffer for the Overlay Optimiser, the bucket size in words of the new program file must be specified as the overlay area in Section 7 (first parameter, macro £TCOV).

Form 8

To specify that the Overlay Optimiser is to be incorporated, the parameter 'OPTIMISER' is included in Section 4 (macro £TCCC). A parameter specifying the number of overlay areas/units for which run-time statistics are to be collected may also be specified (in range 0-9999). If the parameter is omitted, no run-time statistics are collected. Where a parameter 'n' is specified, an area of length 2n + 3 words is allocated to hold the statistics.

Form 10

Where the option to open a duplicate program file is used, the Highest Disc Unit reference number used may not exceed 62 (as URN 63 is used for the duplicate file).

Form 26

This form must not be used in conjunction with the Overlay Optimiser.

PART II IN-FLIGHT APPLICATION ROUTINE CHANGES

1. INTRODUCTION

1.1 OVERVIEW

The standard consolidation procedures for a minor amendment to an individual Application Routine requires two operations. These are the compilation of the routine, followed by a consolidation of the full TPS program. This second operation can be a lengthy process, but can now be avoided by amending the Application Routine 'In-Flight'. This facility is designed to reduce the need for frequent consolidation of the full TPS program. It allows the replacement of an amended Application Routine without the need to close down the system or reconsolidate the program.

Application Routine amendment In-Flight, involves three operations:

1. Pre-process the semi-compiled TPS control routine.
2. Consolidate and append a semi-compiled Application Routine onto the Optimised Overlay File.
3. Utilise the new version of the Application Routine.

1.1.1 Pre-processing

Before the semi-compiled TPS control routine is consolidated, it must be pre-processed by program TP3C. TP3C will incorporate internal control information in the TPS control routine. New message types, associated Application Routine trains and Application Routines, can be introduced into the control routine through the System Definition File belonging to TP3C. These introductions can be included without regenerating or compiling the full system.

When program TP3C has processed the semi-compiled TPS control routine, XPCK is run, followed by TPSN. The TPSN utility converts a standard program file to the optimised format in an Overlay File. Refer to Section 2, Part I of this manual for the conversion parameters.

1.1.2 Consolidating And Appending A Semi-compiled A.R.

After the Optimised Overlay File has been produced, TP3K is used to consolidate a semi-compiled Application Routine and append it to this file, as a new version of an existing routine. During this process the system does not need to be closed down. The system will continue to use the existing routine and will be unaware of the new version.

The free standing overlay consolidator #TP3K, is described in Section 2 of Part II of this manual. The Operating Manual contains instructions for operating the #TP3K utility.

1.1.3 Using The New A.R

Once the new version of the Application Routine has been produced, it can be brought into use by the ZIAR function. This function is provided in the on-line program (refer to Section 3 for an explanation of this procedure).

1.2 EXPANDING AN OVERLAY AREA

The Disc File contains units which are pulled into an overlay area by the Overlay System. The consolidator determines the size of each unit held in the overlay area.

Each unit is divided into 16 relativiser areas, or 'classes', each class referring to an area of store within the program itself. After an Overlay File has been produced by TPSN, the values of each class can be accessed by TP3C.

TP3C contains parameters which can be used to expand the overlay area on the Overlay File. Once TP3C has accessed the statistics of each unit size from the Overlay File, it is possible to expand the size of each unit. This is achieved by applying a percentage or "expansion factor" to each unit, so the values of each class within the unit are increased proportionately. For example, if several of the classes within a unit needed to be expanded by a variety of amounts to take more code, the percentage applied to the whole unit would need to be sufficient to expand the class requiring most space. By having an expansion factor of 10%, each class is increased by 10%, maintaining correct ratios with other classes.

TP3C applies the given expansion factor to information received from the previous Overlay File and creates a new Control File. This must then be consolidated and optimised. The consolidator selects each class in turn, determines the maximum value of that class throughout different units, and allocates space in the overlay area.

Note that a TPS program must be consolidated when an Application Routine grows beyond the current capacity of its overlay area. Consolidating a TPS program has the effect of 'tidying up' the Overlay File.

1.3 A.R. SEMI-COMPILED LIBRARY

It is the responsibility of the user to maintain the Application Routine semi-compiled library in step with the versions used on-line.

2. CONSOLIDATING AN APPLICATION ROUTINE

2.1 GENERAL DESCRIPTION

TP3K reads the semicompiled Application Routine as output by any COBOL compiler, and consolidates it into an overlay unit using information held on the Overlay File. TP3K then appends the new Overlay unit to the Overlay File so that it may be incorporated in-flight by the message type ZIAR.

Before the A.R. can be appended to the Overlay File by TP3K, it must meet the following conditions:

1. The Application Routine must be the only routine within its overlay unit.

2. The Application Routine must have been compiled by the same compiler as used prior to the last time of consolidation by program XPCK.

3. The semi-compiled form of the Application Routine must either be in an output file from the compiler or in a standard XPEU format semicompiled library (i.e. not an XFYZ format library).

2.2 INPUT PARAMETERS

Control parameters defining the Application Routine and files to be used are input on cards.

Format Of The Cards

The first four characters of each card are the program identifier TP3K. These are followed (without space or separator) by the four-character section and sequence number starting from 0101.

Each item is identified by a keyword which in most cases has an alternative shortened form, and which may have an associated parameter. A parameter to a keyword must be on the same card as the keyword, separated from it by any number of spaces and from the following keyword by a comma.

The first keyword on a card follows the eight character leader, separated by at least one space. No comma is needed after the last parameter on a card.

Any number of cards may be used and the keywords may be defined in any order.

Parameters

1. AR

This parameter is mandatory and specifies the name of the Application Routine which is to be consolidated and appended to the Overlay File.

Format : AR name

name = name of the Application Routine

2. SCFILE or SCFI

This parameter is mandatory and specifies the file containing the semicompiled version of the Application Routine as output by the compiler.

Format : SCFILE filename

filename = up to 12 alphanumeric, space or hyphen characters with the first character alphabetic.

3. OFNAME or OFN

This parameter is mandatory and specifies the Original Overlay File (originally output by utility #TPSN) onto which the Application Routine is to be appended.

Format : OFNAME name

name = name for the Original Overlay File (i.e. filename is 'TPSFASTOname'. The name specified may be up to 4 alphanumeric, space or hyphen characters.

4. DUPLICATE or DUPL

This parameter is optional, but must be entered if a duplicate Overlay File is in use. The filename is TPSFASTDname where name is the name specified for the OFNAME parameter.

Format : DUPLICATE

5. LIST

This parameter is optional and indicates that a list of all Application Routines available for switching is to be printed.

6. STATS

This parameter is optional and indicates that size statistics for the Application Routine being consolidated are to be printed.

7. PEND

This parameter is mandatory and indicates the end of parameter input.

Format : PEND

Example

To consolidate Application Routine ABCDAR01 in semicompiled file ABCDAR01SEMI into an Overlay File TPSFASTOTPS1 with a duplicate TPSFASTDTPS1

```
TP3K0101 AR ABCDAR01
TP3K0102 SCFILE ABCDAR01SEMI
TP3K0103 OFN,DUPL,PEND
```

3. INTRODUCING THE APPLICATION ROUTINE INTO THE ONLINE PROGRAM

The on-line function ZIAR switches into use the Application Routine appended to the Overlay File by the TP3K utility.

FORMAT DISPLAYED IN RESPONSE TO ENTRY OF ZIAR

```
ZIAR]

    Enter name of A.R. to be switched...[          ]

    Enter action required.............[ ]: T to Temporarily switch AR
                                            P to Permanently switch AR
                                            C to Cancel last switch
```

ACTION AT THE TERMINAL

The name of the A.R. is entered and a character corresponding to the Action required:

T to temporarily switch the A.R.
The new version of the A.R. is used until the action C (cancel) is requested or until the program is reloaded.

P to permanently switch the A.R.
The new version of the A.R. is used, even when the program is reloaded. Action C (cancel) may be requested if it becomes necessary to revert to the original version.

ACTION BY THE SYSTEM ON RECEIPT OF THE COMPLETED SCREEN

If the system is able to carry out the requested action the response 'CONFIRMED' is displayed.

Otherwise the system responds with one of the following error messages:

1. FILE IN WRONG FORMAT
   The Overlay File was not produced by the utility TPSN.

2. FILE UNAVAILABLE
   The Overlay File cannot be opened. Either:
   a) the Overlay File has not been defined to the system as a Logical File (using keyword LF in the System Definition)

   or b) the Overlay File is currently in use by another program (e.g. TP3K) - try again.

3. INSUFFICIENT DATA
A field has not been entered.

4. INVALID DATA
A.R. name invalid, or action entered is not T, P or C.

5. INVALID LOGICAL FILE SET
The number of physical files described by the keyword LF in defining the Overlay File is not consistent with the number used by the TPSN program.

The options acceptable for an Overlay File are:

"original file" or "original file + duplicate"

6. INVALID REQUEST
In response to a 'P' or 'T' request:

The A.R. named has already been permanently switched.

7. SPECIFIED RECORD NOT ON FILE
The A.R. named has not been consolidated by the TP3K utility.

8. UNABLE TO PROCEED AT PRESENT
The ZIAR function is currently in use at another terminal - try again.

4. STATISTICS

4.1 STATISTICS

There are two sets of statistics the user can request:

1. Overlay File summary
2. Usage of the 16 classes for the Application Routine being consolidated

Both sets of statistics are used to discover the size of units and classes, and to determine by what percentage the units should be expanded.

4.1.1 Overlay File Summary

The user can request statistics from program TPSN which indicate the usage and maximum space available, including the expansion factor for each overlay area.

Enter the parameter STAT to request a summary of the Overlay File from the TPSN program.

The example below lists the amount of space used in each class above the maximum space allowed.

The expansion factor parameters for TP3C were:

```
EXPAND (1,50),EXPAND(2,10)
```

EXAMPLE 1

TPSN/55A OVERLAY FILE SUMMARY CREATED ON 03/09/82 AT 10/35/44

O'AREA		LW	LV	LR	LP	LT	RR	UP	UR	UV	RC	ORR	OPR	CLV	CLP	CUP	CLV
USED	1	0	0	2	171	24	91	138	0	0	0	269	3291	0	94	622	0
MAX	1	0	0	3	257	36	137	207	1	0	0	408	5937	0	141	933	0
USED	2	0	0	0	12	7	0	0	0	0	0	1	1500	0	0	0	0
MAX	2	0	0	1	14	8	1	1	1	0	0	3	1650	0	1	1	0

4.1.2 Usage Of The 16 Classes

The user can request statistics from program TP3K, which indicate how the 16 classes of each Application Routine being consolidated, are being used.

The parameter STAT (or STATS) requesting a statistical output is described in Section 2.2 of this Manual.

The example below lists the amount of space each class has used and how much is spare. The user can then determine by how much (percentage), the whole unit will have to be expanded to meet his future requirements. The example also lists the Application Routines on the specified file, indicating whether the new version of the Application Routine is to be temporarily or permanently used.

The parameter LIST is described in Section 2.2 of this Manual.

EXAMPLE 2

```
TP3K/6A                          TPS SINGLE A.R. CONSOLIDATOR          03/09/82 AT 15/24/08

TP3K0101 AR STAR1,SCFI SEMI STAR1,OFN MTPS,STAT,LIST,PEND

AREA 1,UNIT 3,AR STAR1 CONSOLIDATED OK

          LW   LV   LR   LP    LT   RR   UP    UR   UV   RC   ORR   OPR    CLV   CLP   CUP   CLV
USED       0    0    2  171     2   91  138     0    0    0    35  1376      0     0     0     0
SPARE      0    0    1   85    34   46   69     1    0    0   373  3561      0   141   933     0

AR NAME          AREA    UNIT    TYPE

STAR1               1       3    TEMPORARY

STAR2               1       5    PERMANENT
```

5. GENERATING A TPS SYSTEM INCLUDING THE IN-FLIGHT A.R. AMENDMENT FACILITY

5.1 GENERATION PARAMETERS

The Generation Forms should be completed as described in the System Generation Manual. The following amendments are required to the standard generation parameters in order to include the In-Flight A.R. Amedment Facility. (Note: The macros must be defined in the correct order).

Form 2 (Macro £TCAR)

Include the TPS routine TPSZIAR:

```
£TCAR      TPSZIAR
```

Note 1 : Unit 1 of any overlay area is reserved for this facility.

2 : The facility cannot handle more than 20 different overlay areas or 1000 different overlay units.

Form 4 (Macro £TCTR)

Include the train for message type ZIAR:

```
TZIAR £TCTR     TPSZIAR,TPSSTDOPAR
```

Form 10 (Macro £TCLN)

Add an extra parameter to the £TCLN macro directly following Section 5, specifying L.F.N of Overlay File.

Form 11 (Macro £TCDA)

Include the Overlay File as a Logical File with access method LBNF.

Form 23 (Macro £TCMT)

Include the Message Type Table entry for message type ZIAR:

```
£TCMT     ZIAR,N,Y,N,Y,Y,N,Y,N,N,N,TZIAR,,NOETSA,NODI
```

5.2 SYSTEM FILE

5.2.1 TP3M Users

The Overlay File must be defined to the System Definition File by using keyword LF and obtaining the default definition for Classification = OVERLAY.

Note that the presence or absence of a duplicate must be kept in line with the parameters to utility TPSN. Also the store chain used must be of a suitable size.

5.2.2 TPSM Users

The Overlay File must be defined by TPSM parameters, including the following:

```
TYPE OVERLAY
DUPL        (keep in step with parameters to utility TPSN)
MODE (LBNF) (must not be open at start of day)
ACCE (READ,WRITE)
MONI NONE
WORK (2,0,2,0) if duplicate, or WORK (1,0,1,0) if no duplicate
```

5.3 DEFINING THE SYSTEM FILE AND PROGRAM NAME AS REQUIRED BY TP3C

Program TP3C expects to obtain Program and System File details from the System Definition File that defines the system being created.

The definition details described below are the minimum required to run program TP3C and must be present within the S.D. File input to that program.

Full explanations of the Keyword.Qualifier are given in the System Definition Manual.

TPS2 users will not have a System Definition File therefore they should use the free standing S.D. program TP3S to define the required details.

A guide line for the Keywords and Qualifiers to be used is given below. The Parameter Settings are those to be returned on the screen displayed in response to the entry of the Keyword.Qualifier.

KEYWORD.QUALIFIER	PARAMETER SETTINGS
STOR.CHAIN	Cell size = 128 Number of cells = 3
SMAN.	Message Area Size = 128 User TCR = 128 Store Chain 1 = CHAIN
FAM.BUCKET S	(standard definition obtained by entering 'S' is required)
LF.SYSTEM	Classification = SYSTEM (This field should be sent to obtain the default parameter settings for a System File). Original file id = physical system file identity Store Chain = CHAIN
LFR.SYSTEM	(Default parameter settings are adequate)
FMAN.	Concurrent LF's Open = 1 Number of Tags = 3
FMAN.	Logical File Names = SYSTEM
PCON.	Program name = user online program name

page 2

Note: Keywords MT, AR, ART may be used to introduce new Message Types, Application Routines and Application Routine Trains without regeneration or compilation.

PART III

THE STORE BUFFERED OVERLAY SYSTEM

1. INTRODUCTION

An alternative run-time package for the optimised overlay system is available.

The Store Buffered Overlay system uses a core buffer to store nominated units of an overlay area when they are first accessed from disc. These units are subsequently overlaid from the store buffer when required, and subsequently reduce the number of disc accesses to the program file.

Certain 'long' requests within TPS (e.g. read/write a record) will cause the overlay to be freed if the curret overlay unit is being handled from the store buffer. This can be allowed to happen as the time taken to bring in an overlay from store is considerbly less than the time taken to read a record from disc. This greatly improves the use of overlay areas and significantly reduces the possibility of deadly embraces arising through overlaying.

THE STORE BUFFERED OVERLAY SYSTEM

2. GENERATION REQUIREMENTS

The Store Buffered Overlay system can only be used with the TPS optimised overlay system (TPS2 or TPS3).

2.1.1 Parameter Requirements

Two extra parameters are required on Form 8 (£TCCC), following the OPTIMISER parameter and the OPTIMISER run-time statistics, if present. The extra parameters are:

STOREBUFFER,n

where n is the size of the required store buffer for holding units in store.

2.2.2 Use of Buffer Space

If buffer space is available, all units with a unit number of 500 or greater (irrespective of area) will be held in store.

All units with a unit number less than 500 will be overlayed from disc regardless of the availability of buffer space.

It will be necessary to change the £TCAY and/or the £TCAR macros and the AR information on the System Definition file. This is to amend the unit numbers of routines which are required to use this facility to greater than 500.

3. SIZE OF STORE BUFFER

The following is an extra parameter which can be input to TPSN to assist in the calculation of the size of the store buffer:

BSIZE

This parameter will cause the following to be printed:

- A table of all units (by area/unit) with the space these units will occupy in the store buffer.
- The size of buffer reflecting all units with a unit number greater than 500 and the buffer work space used by the Store Buffered Overlay system (currently 128 words).

4. ACTION OF THE STORE BUFFERED OVERLAY SYSTEM

For units with a unit number of less than 500, the Store Buffered Overlay system will work identically to the standard TPS optimised run-time package.

For units with a unit number of 500 or above, the system will move a copy of these units into the system buffer, providing there is sufficient space. All subsequent requests for these units will cause the unit to be moved into its area from the buffer. The system will not attempt to add units to the buffer when it is full.

When 'long' TPS requests are made, (e.g. reads, writes), the system will check whether the unit number of the AR is 500 or greater and that it is not locked. If this is the case, the system will free the unit and place the demand for the AR at the end of the demand list. This enables another suitable thread to use the overlay area. Note that use of the overlay area can become inefficient if the unit is not in the store buffer because an incorrect size of store buffer was specified.

When 'long' requests are made in a single threading system, the action of freeing the unit and placing the demand for the AR at the end of the demand list, can become marginally inefficient. However, providing the unit is in the store buffer, this inefficiency will be heavily outweighed by the increase in speed of bringing in a unit.

5. ADDITIONAL DIAGNOSTICS

When switch 10 is on, the system will record additional diagnostic information in the diagnostic buffer. This will consist of three extra variations to the OVERLAY CONTROL 2 entry (indent 0404):

- o DB entry when placing a unit into the store buffer.
- o DB entry when bringing a unit in from the store buffer.
- o DB entry when forcing a unit out and re-queueing the demand.

5.1.1 DB entry When Placing a Unit Into the Store Buffer

Word	Contents
0	Zero (CB address)
1	Zero (AR number)
2	Area/unit
3	"INS "
4	"0404" (id/length)

5.1.2 DB Entry When Bringing a Unit in From The Store Buffer

Word	Contents
0	Zero (CB address)
1	Number of times unit accessed from store (AR number)
2	Area/unit
3	"STOR"
4	"0404" (id/length)

5.1.3 DB Entry When Forcing a Unit Out and Re-queueing the Demand

Word	Contents
0	CB address
1	AR number
2	Area/unit
3	"RE-Q"
4	"0404" (id/length)