UNIVAC 494

TRANSACTION CONTROL SYSTEM

1. PREFACE

This writing is intended as a programmer's reference manual on the Transaction Control System (TCS). It provides a comprehensive source of information concerning the programming of Transaction Processing Segments (TPS) to operate under the control of the U494 TCS.

Included in this manual are sections explaining the basic concepts of the TCS and the mechanics employed to implement these concepts. These sections are followed by a more detailed explanation of the mechanics of the system. The latter part of the manual contains the Glossary, Formats of Control Tables, and a detailed explanation (definition, format, examples) of the functions available to the user.

1.1 ---------------------- TABLE OF CONTENTS --------------------------

TABLE OF CONTENTS (CONTINUED)

2. GENERAL DESCRIPTION

The Transaction Control System (TCS) is a random storage, transaction processing, oriented system made up of modular routines that have been designed to run under Omega. The TCS is the controlling agent between Transaction Processing Segments (TPS's) written by the user, the flow of messages from input through the Transaction Processing Segments to output, and Omega.

With minor exceptions TCS will operate under Omega as a normal JOB. The TCS JOB stream is comprised of a main task, which is resident for the duration of the Normal Operation of the system, and a supplemental task, which is loaded when needed, for "off-line" recovery. The TCS will operate under Omega as a single user.

The first part of the TCS is the start-up portion. The starter will provide the ability to assign and request terminal lines and allow the installation to call the various functions necessary for a particular configuration of lines and equipment. All control and queuing of messages is performed by Omega's communication program. Omega provides entries for user's own-code routines to do classification of messages, assign priorities for queuing and specify which TPS(s) will operate on them. The extent of the processing done at this level will vary depending on the needs of the installation.

The second part of the TCS is concerned with the processing of messages, the updating of files and obtaining information from the files to process the messages. The tool used in processing is a Transaction Processing Segment (TPS). This is a program that will run as an activity under the TCS. The TPS(s) is generated using the normal language available to the U-494 and is collected via the standard Omega procedures. Along with the TPS is a Transaction Area (TAR). The TAR is used as a control and scratch area for the Transaction to hold the input message, perform calculations, store intermediate results for subsequent TPS's, and build output messages. There is a standard set of rules that must be followed in the preparation of the TPS(s) and TAR(s) and the rules vary somewhat with the particular type of TPS being prepared. This will be covered in detail later on in the manual.

Before the files can be used by the TPS(s) they must be generated for the system. The TCS assumes that a batch program running under Omega has been used for this normal off-line operation. Once these files have been created they are registered with Omega's master file directory system as permanent files. The files will be available through the TCS by referring to them by user number-file number.

All functions which operate on the files in their entirety such as copying them or sorting them, are considered to be batch type functions and cannot easily be accomplished under TCS simultaneously with Transaction Processing.

The TCS does the assignment and regulation of the mnemonics used by Omega to refer to the files, and adjusts the number of files that can be referenced by users at one time. The regulation of mnemonics is necessary since all users of the TCS are considered to be a single user by Omega and only one set of mnemonics is available.

In order for the TPS to acquire messages, manipulate files, write messages and other aspects necessary for processing, the TCS has a number of service requests available to the user. It is necessary to use the TCS service requests in most cases, instead of Omega's, because the TCS is a normal JOB considered by Omega to be a single user. TCS, however, may have a number of users and this has placed certain restrictions and conventions on the service requests available to the TPS programmer.

The TCS provides the ability to initiate TPS's on a TIME-OF-DAY basis. When the appropriate time is reached, the specified TPS is initiated. Control of this operation is done through the Timer Table by the Starter routine. Entries in the Timer Table are not made dynamically. The Table is established prior to start up time. It is possible, however, to change the TIME-OF-DAY of an entry in the table by a user service request.

The last part of the TCS is file recovery. The basic file recovery system used is to capture, on tape, dumps of the files and to record images of any portions of the files that have changed by writing "after-look records" onto magnetic tape. The file recovery procedure is a combination of reloading the files up to a certain point and updating them with any changes made after that time.

File recovery is also concerned with portions of the files that have become unreadable. In the case of duplex files much of the recovery procedure is automatically built into the error routines, where portions of files are re-created as they are determined to be unreadable. In certain cases in a duplex system both copies will go bad and in this case they are treated as a simplex file and recovery procedures are initiated from the audit trail.

3. FUNCTIONAL DESCRIPTION

3.1 MESSAGE PROCESSING

The message enters the system from a line terminal and is handled by Omega's communication program and an optional user own-code routine. The own-code routine may determine which random storage Input Queue Omega's communication director will store the message. At some time prior to putting the message on the Input Queue an operation will be performed to write the message on the Audit trail tape and to perform message recovery procedures. The user own-code routine must establish for this message a starter value, according to some parameter of the message, which will point to a particular entry in the starter control table of the TCS. The starter control table will indicate, at this entry, the TPS to be used to process this message and the amount of core needed for the TAR.

At fixed time intervals and whenever a transaction is terminated, the starter searches the input queues, beginning at the highest priority, for a queue that isn't busy. When the starter finds a queue that isn't busy and contains a message, it activates the Get Next Transaction routine. This routine reads in the message from the input queue to an input queue buffer in core. The size of the Input Queue Buffer is determined at system generation time, and all or part of the message may be contained in the buffer. Input Queue Buffers may also be created by other routines which the user may add to TCS.

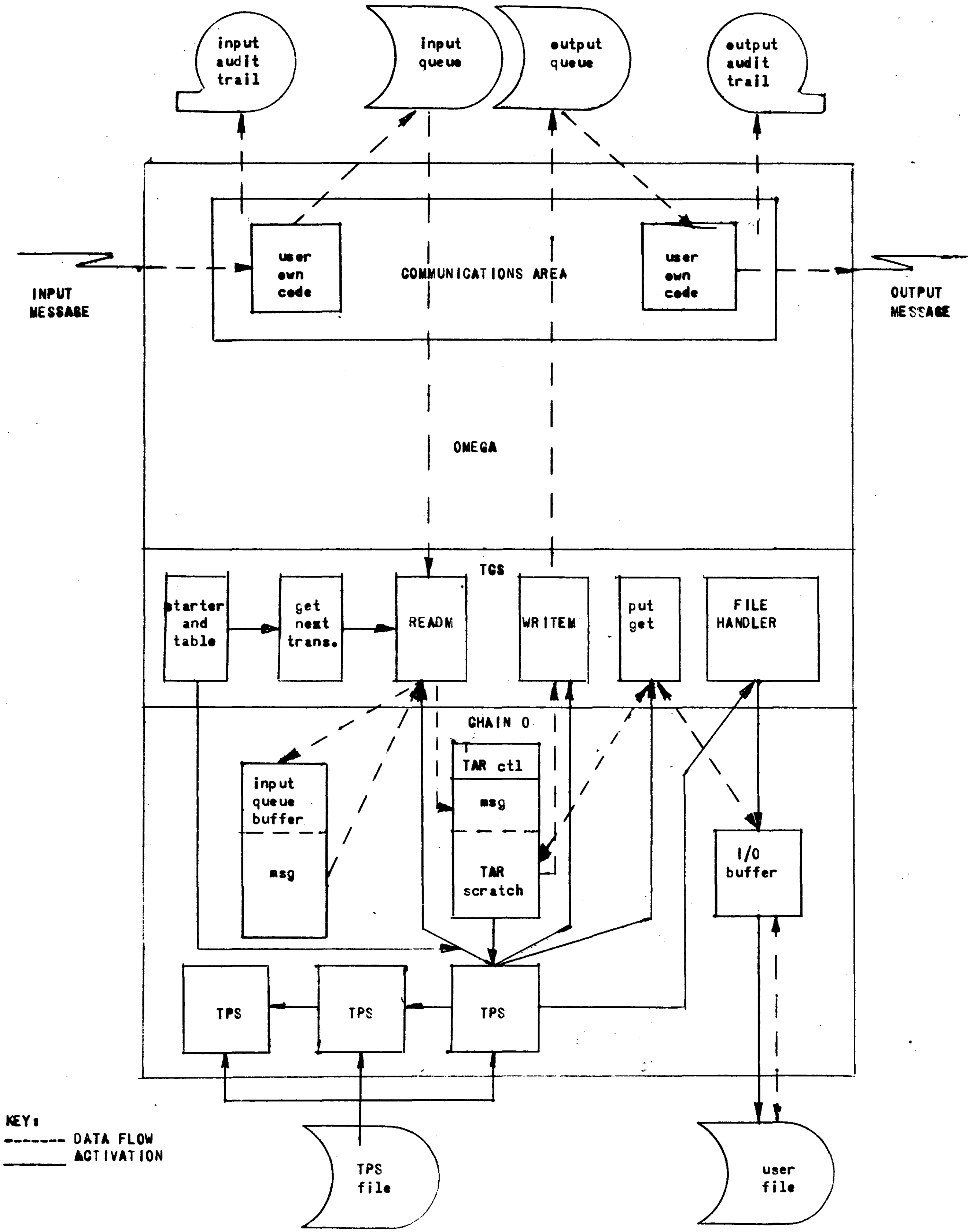
As the message is being read into the Input Queue Buffer, the TCS is loading the correct type of TPS (see Types of TPS) as indicated in the starter control table, establishing the TAR to be used with this TPS, setting the RIR and PLR limits, and finally registering the TPS and TAR as an activity.

The TPS acquires the message through a READM Service request. The READM request transfers a specified amount of the message into the requesting TAR working storage area. Subsequent READM's will transfer the remaining part of the message into the TAR until an end of message is reached.

The TPS processes the message using whatever service requests it needs. If another TPS is required to process this message the current TPS gives the required request to activate the next TPS for processing.

When the processing of a message has completed the user initiates a function (CLEAN UP) which insures a proper completion for the message.

When the TPS(s) has finished processing the message, and an output is required, a WRITEM passes the output message from the TAR to Omega for transfer to a random storage Output Queue. The WRITEM request also sets up a control block for this message indicating its destination and links it to other control blocks for the purpose of recovery. The message is transferred from the Output Queue to the output device via Omega's Communication Program and an optional user own-code routine. The amount of processing by the user own-code routine depends upon the needs of the particular installation.



input audit trail
input queue
output queue
output audit trail
INPUT MESSAGE
user own code
COMMUNICATIONS AREA
user own code
OUTPUT MESSAGE
OMEGA
TCS
starter and table
get next trans.
READM
WRITEM
put get
FILE HANDLER
CHAIN 0
input queue buffer
msg
TAR ctl
msg
TAR scratch
I/O buffer
TPS
TPS
TPS
KEY:
------- DATA FLOW
_______ ACTIVATION
TPS file
user file

3.2 TAR LAYOUT

In the TCS, the control area for a transaction is referred to as the TAR. Basically, there can be two parts to a TAR, a control portion and a working storage portion. The control portion always exists; the working storage portion's existence depends on the type of TPS used to process the transaction.

The control portion is not accessible to a TPS. The information contained in this portion is maintained by TCS routines. Some of the information has to do with statistics regarding the particular transaction while other parts provides linkages to maintain the control necessary. Various TCS routines are re-entrant and they therefore require additional storage for information regarding the transaction. These areas are provided for the control portion of a TAR. A TAR control portion of 100_8 words exists for every transaction while being processed.

A working storage portion exists for TPS(s) that require a separate working storage (See Types of TPS section). The first 10_8 words of this portion is considered the COMMUNICATION AREA for the transaction. The remainder of this portion is scratch area placed at the disposal of the TPS(s) routines in order to process the transactions. This area provides a common place to communicate data between TPS's, to disect input messages and to build output messages. For TPS(s) not requiring a separate working storage the COMMUNICATION and scratch areas are located with the particular TPS(s).

The Program Lock Register's (PLR) lower-lock (LL) and upper lock (UL) are set to the bounds of the working storage portion if it exists. The length of the working storage portion is variable in multiples of 100_8 words and dependent on the type of associated transaction.

If it becomes necessary for the user to enlarge his TAR area, a service request is available to perform this function. It is also possible for the user to contract his TAR area by use of a service request. Their formats are explained in the section entitled User Functions.

The design of the system determines the size of the TAR working storage. A Type 1 TPS knows its TAR size.

3.3 TYPES OF TPS

There are two types of TPS that can be written by the user. One of the types can further be categorized into two modes. The distinguishing factor is the relationship of the TAR Control area to the TPS(s) area.

A Type 1 TPS is not contiguous to its TAR control area. The relative index register (RIR) established to execute this processing segment is set to the base of the TPS (See fig. 1&2). For this type of TPS the PLR's lower-lock (LL) and upper-lock (UL) are set to the bounds of the associated TAR scratch area. These register settings are made when the TPS is activated for processing. With the LL and UL set on the TAR scratch area, the integrity of all other TAR's, TPS's, etc. is maintained. It further demands that all reading/writing of data must be done from the TAR scratch area. Use of B-register is further defined when using the Type 1 TPS. Within the TPS, BØ-B3 must be used; when accessing the TAR scratch area B4-B7 must be used.

Type 1 TPS can be written in one of two modes: re entrant or non re entrant. A re entrant Type 1 TPS can process more than one transaction concurrently (see Fig. 1). In this mode, one TPS could be processing as many transactions as the system allows to be active concurrently. Control information is kept in the transactions TAR area. The TPS's integrity is maintained by Omega. The remaining chore of setting the PLR and RIR registers to the proper values for the multi-TAR, one TPS atmosphere is handled by Omega as it switches amongst activities (i.e. TAR's). One can easily see that this type-mode TPS is used to process high volume transactions where a one TPS/TAR relationship would be very binding on the system.

A non re entrant TYPE 1 TPS can process only one transaction concurrently. This is due to the fact that this type-mode allows for overlays of secondary processing segments in the TPS area (see Fig. 2). Although the integrity of the TPS coding is maintained by Omega, the overlays cause a change in environment which make the TPS non re entrant. A TPS that requires a large amount of core when active and/or is infrequently used are prime candidates for this type-mode TPS.

The above mentioned TPS modes can be used together to process any one transaction; however, they cannot call a TYPE 2 TPS. The processing of any one transaction must be by either TYPE 1 or TYPE 2 TPS.

A TYPE 2 TPS's area is contiguous to its respective TAR control area. The Relative Index Register (RIR) and the Program Lock Register's lower-lock (LL) are set to base address of the TPS. The upper-lock (UL) is set to the upper bounds of the TPS area (see Fig. 3). With the LL and UL set to the same area as the TPS, the TPS maintains its data and control information within; therefore, the TPS's integrity is not maintained and cannot process more than one transaction concurrently. There are no restrictions on the use of B registers when using this TYPE TPS. The processing of low volume transactions should be by this type TPS. As stated previously a TYPE 2 TPS cannot call a TYPE 1 TPS.

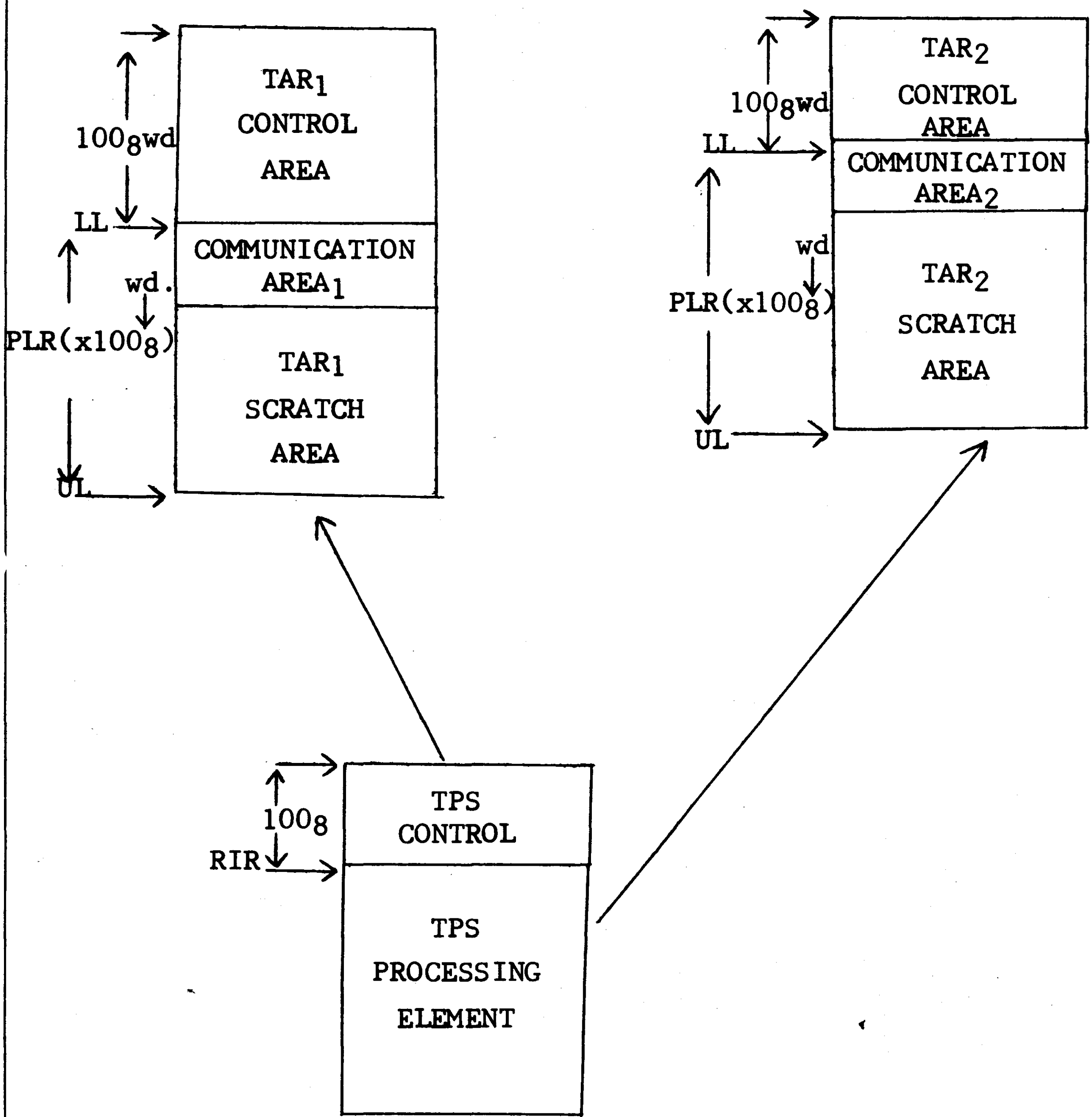


FIGURE 1

TYPE 1 TPS - RE ENTRANT

Example of one TPS being used to process more than one transaction or TAR.

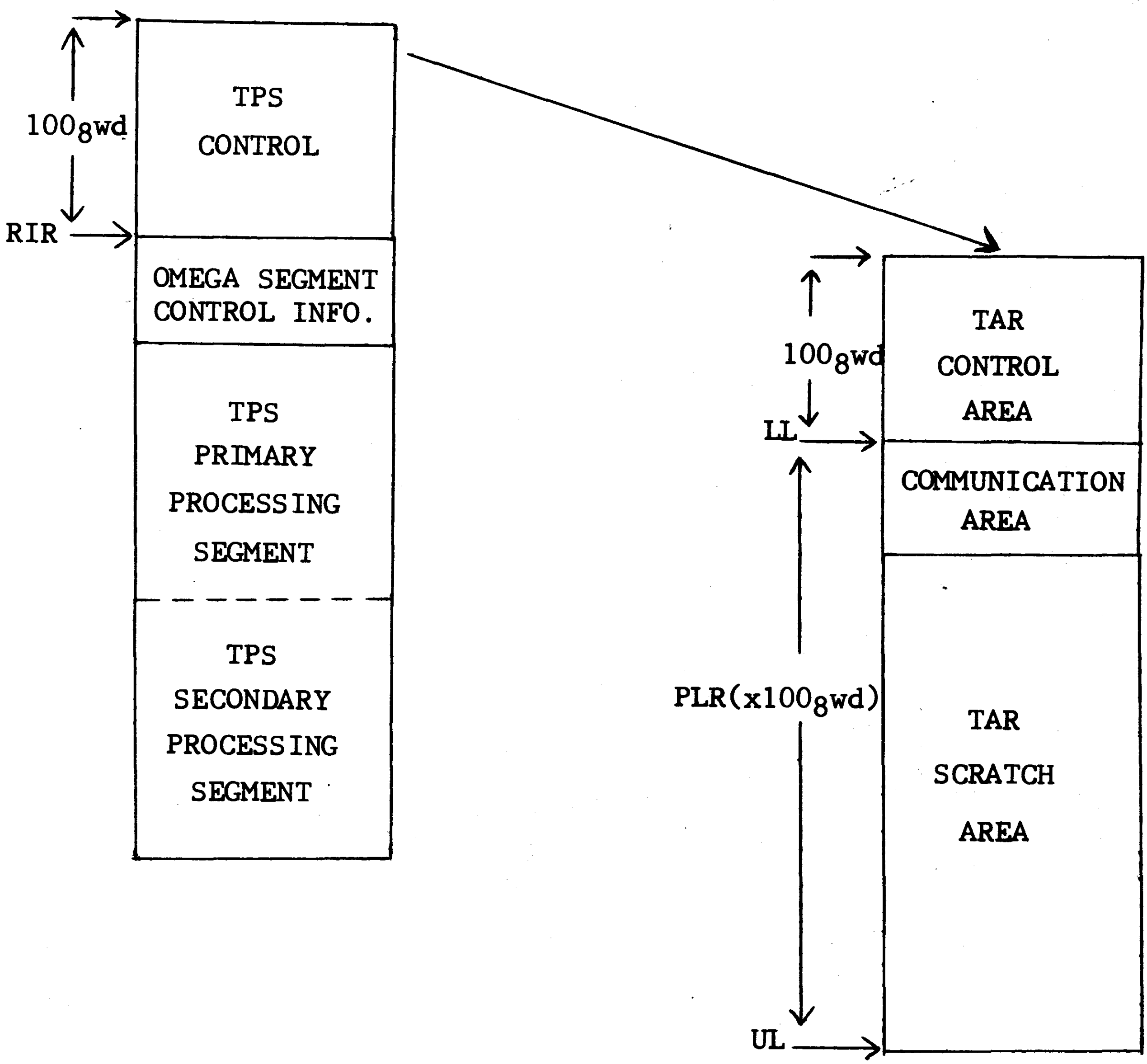


FIGURE 2

TYPE 1 TPS - NON RE ENTRANT

An example of 4 segmented type 1 TPS's which can only process one transaction at a time.

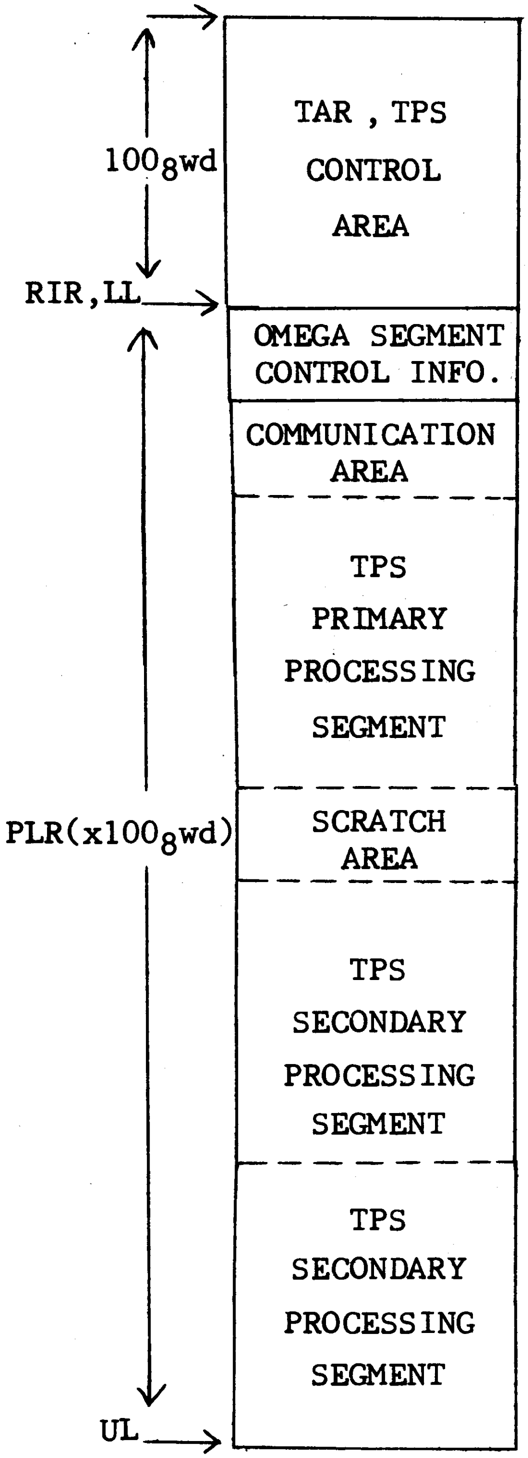


FIGURE 3 - TYPE 2 TPS

Arrangement of TPS segments and scratch area is exemplary - Omega segmentation rules apply. The communication area is mandatory though.

3.4 TPS PREPARATION

Prior to coding a TPS a few decisions have to be made. These decisions will determine the basic ground rules under which the TPS must be written.

Initially a decision must be made regarding the TYPE of TPS that is to be written. This decision is based mainly on the activity of the related transactions. Once this decision has been made the TPS writer is aware of the settings of the Relative Index Register (RIR) and the Program Lock Register's lower-lock (LL) and upper-lock (UL). He also is now aware of any special way in which the B-registers must be used. The location of the Communication Area is also dependent on the TYPE of TPS. Once its position is determined a review of data conveyed in this area is recommended.

In order for the TPS to effectively communicate with the TAR area and with the other TPS(s) that are required to process the transaction, a knowledge of the data associated with the transaction and its relative position within the TAR scratch area is necessary. The area(s) may have been mapped at systems design time or may come about after a meeting of the TPS programmers concerned.

TCS stages the first part of each message (or the entire message) in core and then transfers the message or parts of it in response to user requests to a TAR scratch area. In order to efficiently secure parts of messages through TCS, the TPS writer should familiarize himself with the size of the particular input queue buffers.

The medium provided by TCS to the TPS to communicate with the TCS routines is a EXRN * 70000 instruction. Prior the execution of this instruction the A and B7 are loaded (as necessary) with the indicative information regarding the service request. All information regarding the formatting of the service requests is included in the section entitled <u>User Functions</u>. The general simplified form of a service request is:

```
ENT*A*FUNCTION CODE
ENT*B7*ADDRESS OF PACKET (RELATIVE TO LL)
EXRN*70000
```

3.5 TPS LAYOUT

Transaction Processing Segments (TPS) are made of two sections both of which exist for every activated TPS.

The first section of a TPS is the control area. Its existence as an entity is dependent on the TPS's TYPE. The control section for a TYPE 1 TPS is a 100_8 word area immediately before the second section. In a TYPE 2 TPS the control section will be contained within the TAR control area. Pertinent TPS control information such as name/version and the backward/forward TPS chain links are contained in this area.

The organization of the second section is not only dependent on the TYPE of TPS but also the mode. It is the processing segment. For the re entrant TYPE 1 TPS the second section is a straight forward Omega load element. When activated the entire element is brought into core. A non re entrant TYPE 1 TPS consists of an Omega generated segment control area followed by the areas established to hold the primary and secondary load elements. The Omega generated segment control contains a jump to the starting address of the processing segment and necessary segment descriptors to load and activate the different segments as required.

The second section of a TYPE 2 TPS is similar in some respects to the non re entrant TYPE 1 TPS. It too has an Omega generated segment control area and areas established to hold the primary and secondary load elements of the processing segment. The difference lies in the fact that (1) this type TPS must provide a COMMUNICATION AREA. This area is placed immediately following the segment control area (i.e. the first of the user written programs - see Fig. 2). (2) Scratch areas for storage(s) of intermediate results and control information are established by the TPS writer as he sees fit. The only restriction is that the scratch areas must lie within the bounds of the TPS.

3.6 FILE HANDLER

As stated previously the File Handler is a collection of routines that enable the user and the TCS itself to manipulate mass storage files. It also provides the necessary controls so that the manipulation of the files is done in an orderly fashion. The File Handler controls the allocation and de-allocation of scratch and sequential files to the user and TCS. Control is exercised by the File Handler over the files so that recovery procedures can be implemented. In order to do the above the File Handler maintains two files of its own.

Omega uses one and two letter mnemonics file code to identify files under its control. All file codes used in the TCS are of the two character form, from AA to YZ. The greater part of the file codes, from AA down to some limit, are used to identify mass storage files. Users of TCS request files by specifying a user number/file number. The File Handler maintains a directory in order to control the assignment of the file codes for the users requests. These file codes are then communicated only by TCS to Omega to use the particular files.

The File Handler insures that the updating of individual records is done in a serial manner and locks out the record from simultaneous updates. It provides the ability to create an image of a record both before and after it has been updated on an audit trail. This gives the system the ability to recreate its files or pieces of them if they have been harmed.

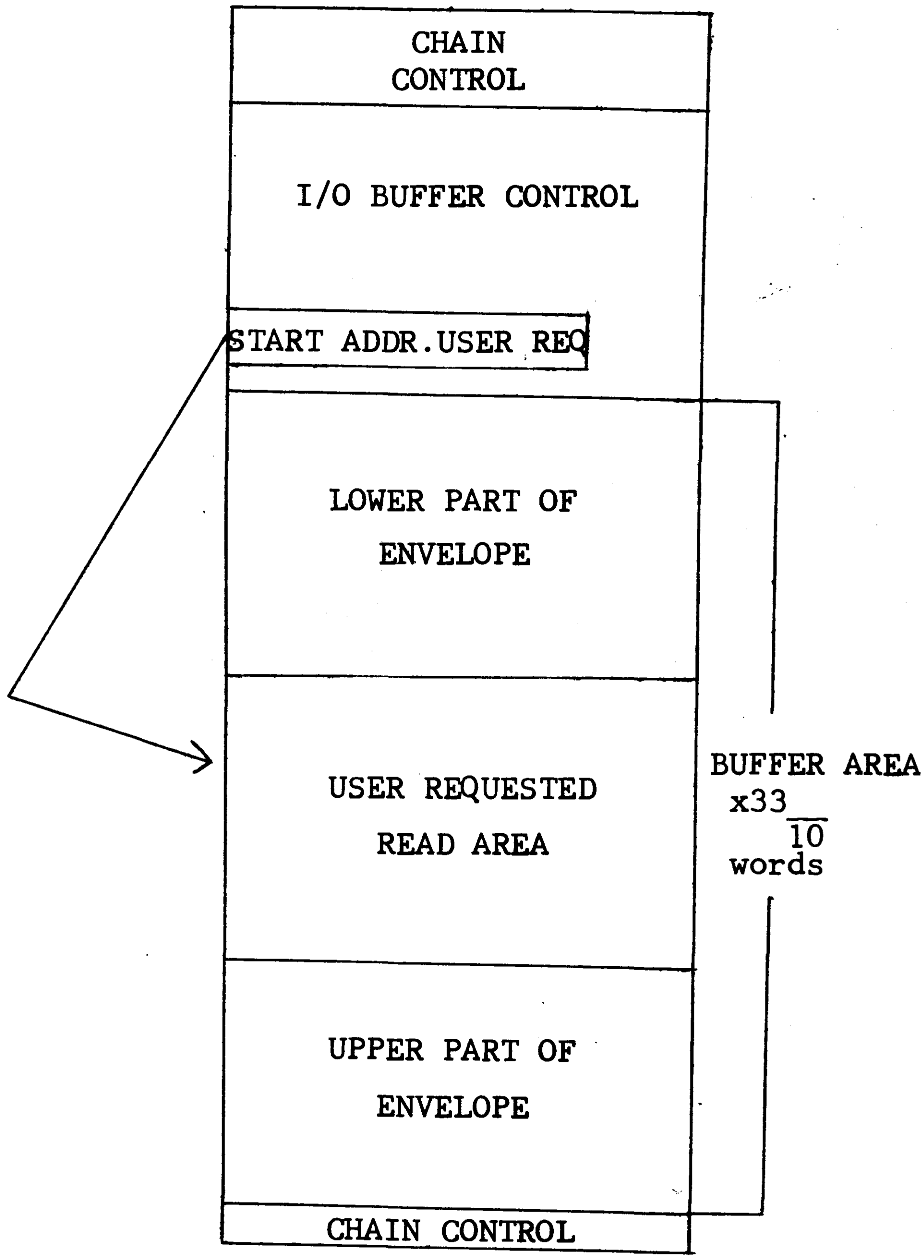
Users having a duplex system require the writing of records to two mediums. The File Handler does the dual write if required and optionally alternates the reading of records between the dual systems.

User requests for scratch and sequential files are processed through the File Handler. Seperate file codes are dedicated at system generation time for assignment to these types of files. The File Handler uses a scratch file to store before-looks of accessed files for each transaction in process. The TCS also uses a scratch file to perform a TAR roll-out.

Assignment and allocation of files for the audit trail and cyclic dump are made by the associated TCS routines.

Requests for user mass storage records transfer these records to/from core buffers from/to mass storage. These buffers are not directly accessible to the user since they lie outside his PLR limits. The File Handler performs the transfer of data as required from/to the user work area to/from the buffers providing numerous advantages in field protection and timing.

Due to the physical characteristics of the mass storage files the area to be read may not correspond directly to the area requested by the user. If the file is not core resident and Before-Looks have not been negated, the area to be read must be made synchronous with the sector organization of mass storage. The logical start and end address must be modulo 33. This means that the actual number of words to be read (and possibly written later) in this case must be computed from these two addresses, and the relative start address of the record as specified by the user must be computed relative to the actual starting address of the read. This expanded record is referred to as the "envelope" and is treated as the record to be processed. With the use of the expanded record area (modulo 33 starting and ending addresses) it is possible to update a record area (read, update, write) with two I/O requests to the mass storage system in all cases. See figure on next page depicting an I/O buffer housing an "enveloped" read area. The number of words in the envelope portion(s) can in the worst case be sixty-four (64) words. This would occur when the user specified read area begins at a logical address of n plus 32 words and ends at a logical address of n plus 1 word where n is some multiple of 33 words.



An I/O Buffer showing the layout of an "enveloped" user read request. The locklist entry contains the entire buffer area until this I/O Buffer has been written or released. Then, the locked out area is set to the user requested area only.

FIGURE 4

The File Handler provides two types of buffers, normal I/O buffer and Core Resident Buffer, to house the user's data. Unless the core resident type is requested by the user, the I/O Buffer type is used.

The I/O buffers are assigned and released as described in the particular user functions that make use of them. The I/O buffers being used for a particular transaction are chained to each other and to the TAR of the transaction through forward and backward links. Figure 5 depicts three (3) I/O buffers chained to a TAR.

For portions of certain high activity files it is desirable that a buffer be retained in core. For all file operations this buffer corresponds to the mass storage and is automatically written back or read as required. This type buffer is called Core Resident. This type buffer gives the user the ability to access a large number of contiguous record areas with a minimum number of reads/writes to the mass storage system(s).

To insure that the updating of an individual record by multiple transactions is done in a serial manner the File Handler maintains a locklist of the areas of the file(s) that are being updated. As long as an area appears on this list no transaction other than the one presently updating the area can have access to the record(s) within the area for updating purposes. Areas of the File(s) are added to and deleted from this list as the proper functions are initiated.

A situation may arise where a stalemate exists. This may be caused by one of two conditions.

Suppose Transaction 1 has locked out record A and is now attempting to read record B. At the same time transaction 2 is attempting to read record A and had previously locked out record B. Neither transaction will be successful. In order to break this stalemate one of the transactions will be aborted.

The same sort of condition may exist when a number of transactions are vying for free core, the amount of which is insufficient to fill all the requests. If after an interval of time (installation parameter) a request for core remains unfilled, the transaction associated with the request will be aborted.

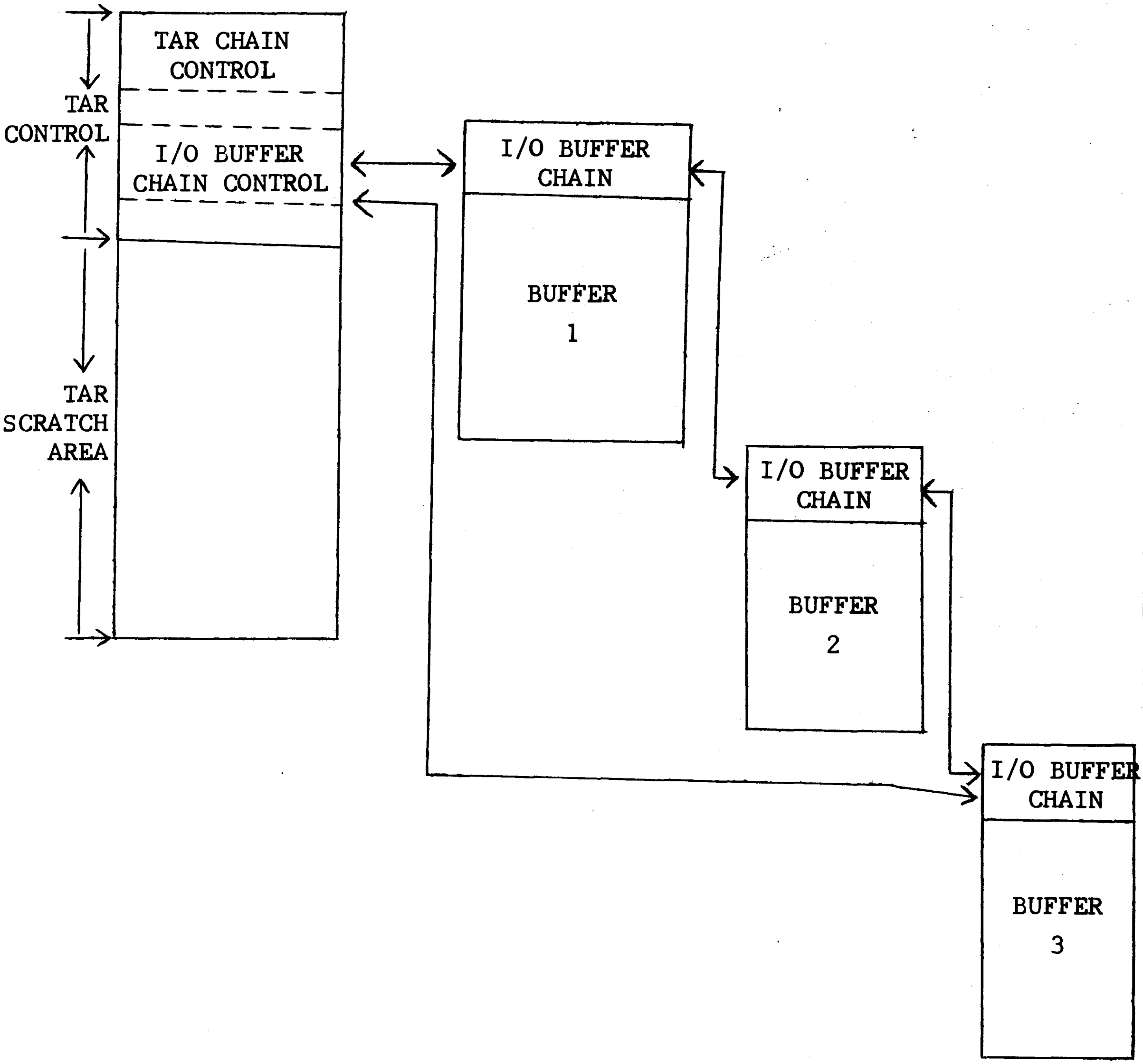


FIGURE 5

A Transaction Area (TAR) with
3 I/O Buffers Chained to it.

4. GLOSSARY

Audit Trail	- A path left by TCS while it is in operation. This path contains information from which any transaction can be recreated, a file can be recreated or the system restarted. It contains record images, before and after changes, messages, statistics and control information.
Base Address	- An address which serves as the base, initial, index or starting point for an area of core or file. Records or fields, within such an area are specified relative to the base address and accessed by adding to the base.
Chaining	- A system of linking records together into groups by the use of control fields in each record which contains the address of the next record in the series or chain.
Communication Area	- A 10 word (octal) scratch area used for transferring parameters and status between the TCS functions and the user. Its location is dependent on the associated TPS's type. A Type 1 TPS's communication area begins at the base of the TAR scratch area (LL). With a Type 2 TPS the area begins at the user start address plus 1. Words 0, 1 are used to return parameters from the service functions. Word 2 contains the function number. Words 3-7 are available to the function.
Concurrency	- The ability of several activities to share the computer at the same time. Omega switches the CPU from one activity (e.g. TAR, transaction) to another so each gets a chance to process.
CWORD0,1,2,..,7	- Words 0-7 of the Communication Area.
Duplex File	- A file for which two copies exist and are maintained.
Nonreentrant	- A nonreentrant routine is one which cannot be entered by a transaction (or activity) until the transaction currently using it is through with it. A nonreentrant routine may be self-initializing in that it may be entered repeatedly without retrieving an initialized copy from the drum.

Program Lock-in Register - This register defines the upper (UL) and lower (LL) memory address limits which can be accessed. The lower limit (LL) may be used as the relative index for operands.

Reentrant - A characteristic of a routine. A reentrant routine is one which does not modify or overlay itself. It is merely code. All data and scratch areas are referenced through index registers. Thus, many concurrent transactions (activities) can be executing the routine in the same time frame (in a multi-processor, simultaneously).

Relative Index Register (RIR) - Used as a relative index for operand and/or instruction addresses, it provides the ability to load worker programs anywhere within memory without modification to these programs.

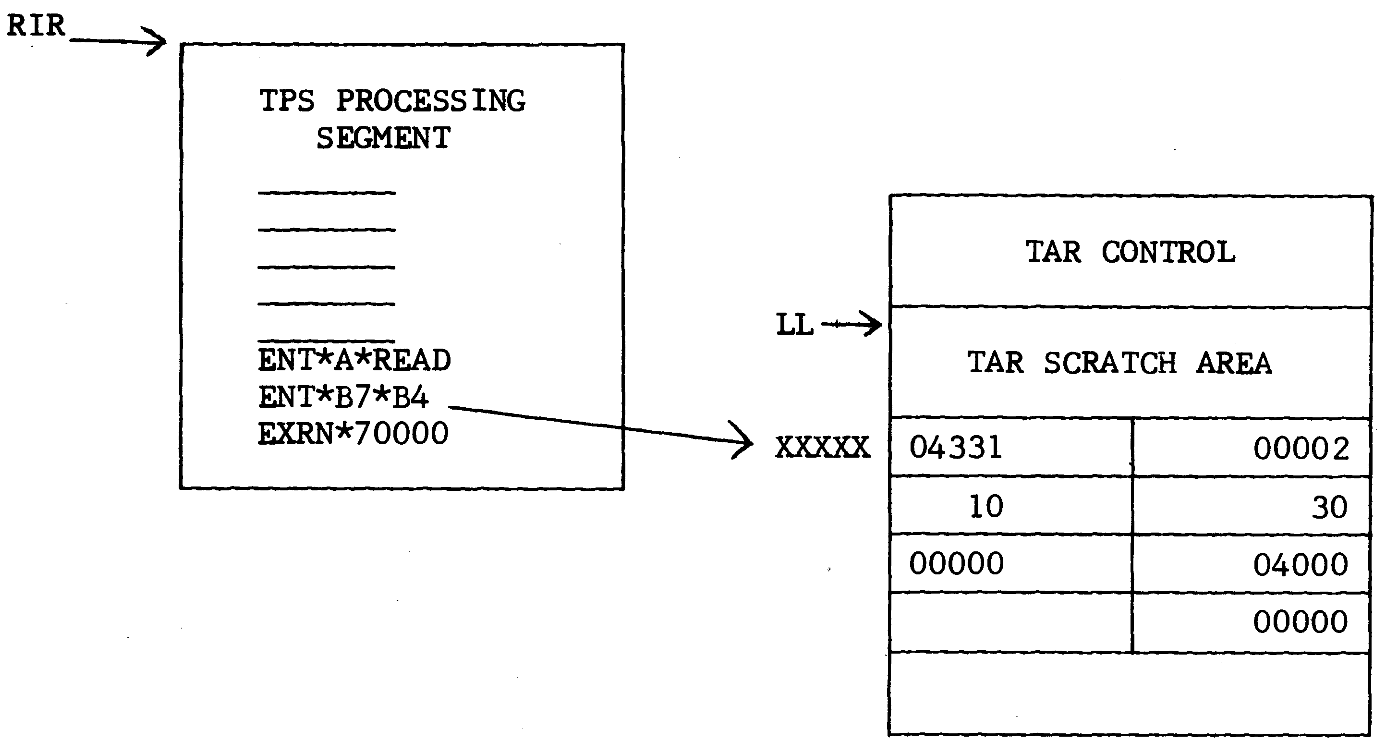
TAR - A Transaction Area is an area of core established for processing a message. Each message has its own TAR and there are many TAR's in core at the same time. A TAR consists of a control portion (100 octal words) and a scratch portion. With a Type 2 TPS, the TPS itself is loaded within to scratch portion of a TAR. The TAR is used for control by TCS and for holding the message, intermediate results, current I/O data, etc. by a TPS.

TPS - Transaction Processing Segment. A program segment used to perform a logical unit of processing, e.g. a message is processed by one or more TPS's. TPS's may be reentrant or nonreentrant and may or may not require separate transaction area. By definition, reentrant TPS's do require a separate TAR.

5. TCS SERVICE REQUESTS FOR USER FUNCTIONS

Users will communicate with the Transaction Control System (TCS) by executing an EXRN*70000 call. The particular function desired is specified by a code in the A register. The address of the packet containing the particular service function's parameters is in the B7 register (if a packet is specified for the service request). Since the user will have to have access to the packets it is necessary that the packet area(s) be in the TAR scratch area for TYPE 1 TPS(s). (This packet may be anywhere in the TYPE 2 TPS). The formats for the packets are defined under the individual functions.

A function call could look like this:



PLEASE NOTE: All parameters conveyed are considered to be OCTAL, (except where the packet is Sequential Assignment Request).

The normal return is with CWORDO positive. In some cases an error code may be returned; CWORDO will be negative in the event of an error. Any parameters resulting from the function requested will be in CWORDO and CWORD1.

5.1 READ

The READ service request:

1. Requests the assignment of an I/O buffer from free core.
2. Transfers the data from a specified area of a particular file into the I/O buffer.
3. Performs record lockout.
4. Takes before looks of the specified file.
5. If the file specified is core resident see the section on core resident reads.

Options available:

1. The file may be locked out in its entirety, an absolute lockout.
2. The record lockout be inhibited.
3. The after look be eliminated.
4. The file will have a core resident buffer - it will be a core resident file. Does not override a file lockout.
5. Read area without regard to locklist.

Packet:

Ø	USER #	FILE #
1	OPTIONS	# OF WORDS
2	LOGICAL ADDRESS	
3	UNUSED	CORE RESIDENT BUFFER SIZE

User #/File # : The user number and the file number used to identify the particular file to be read. For scratch files the user number is 77777 and the file number is the mnemonic that has been assigned by the scratch file allocator.

Options:

1 - Lockout File
2 - No Record Lockout
10 - No After Look
20 - Core Resident File
40 - Ignore Locklist, Read

NOTE: These various options can be combined as octal numbers to provide the options desired.

# of Words	The number of words to be read from the file.
Logical Address	The relative address in the file at which the reading is to begin. The same as the logical address in Omega.
Buffer Size	The size of the core resident buffer to be established to contain the file. If this option is not selected, this parameter should be zero (0).
Done Address	Communication Word 0 (CWORDO) will contain the buffer number used to reference the requested area.

The following conditions will cause a transaction abort:

1. User number or file number equal to zero.
2. User number or file number cannot be found.
3. File number is odd.
4. Core resident buffer size is equal to zero. When option 20 is used.
5. In the case of a scratch file, if the scratch file mnemonic has not been assigned for this TAR.
6. Unable to read the requested area of the file.
7. Number of words to be read is equal to zero.
8. Omega assign for the file cannot be fulfilled.

5.2 WRITE

The WRITE service request:

1. Transfers the specified data buffer back to mass storage.
2. Generates the after look record.
3. Releases the I/O buffer previously assigned for this area.
4. If the file specified is core resident see the section on core resident reads.

Options available:

1. Release the record lock and I/O buffer area - no data transfer will occur and no after look will be taken.
2. Release the file lock, all other functions of the WRITE service request will be performed.

NOTE: A WRITE to an area cannot be given unless you previously READ it with lock.

Packet:

Ø	OPTIONS	I/O BUFFER NO.

Options — [redacted]
2 - Release the file lock
4 - Take after-look on duplex file

I/O Buffer No. — The number of the I/O Buffer that was created for the read and returned to Communication Word 0.

Done Address — Communications Word 0 (CWORDO) will contain the I/O buffer number.

The following conditions will cause a transaction abort:

1. An attempt to write in an I/O buffer that was not preceeded by a READ with lock of that area.
2. Incorrect I/O buffer number.

5.3 GET

The GET service request:

1. Transfers a specified amount of data from an I/O buffer to a specified location in the TAR.

Packet:

Ø	# OF WORDS	TAR LOCATION
1	BUFFER NO.	WORD NO.

# of Words — The number of words to be transferred from the I/O buffer to the TAR.

TAR Location — The location in the TAR to which the transfer will start.

Buffer No. — The number of the I/O Buffer from which the data is to be obtained.

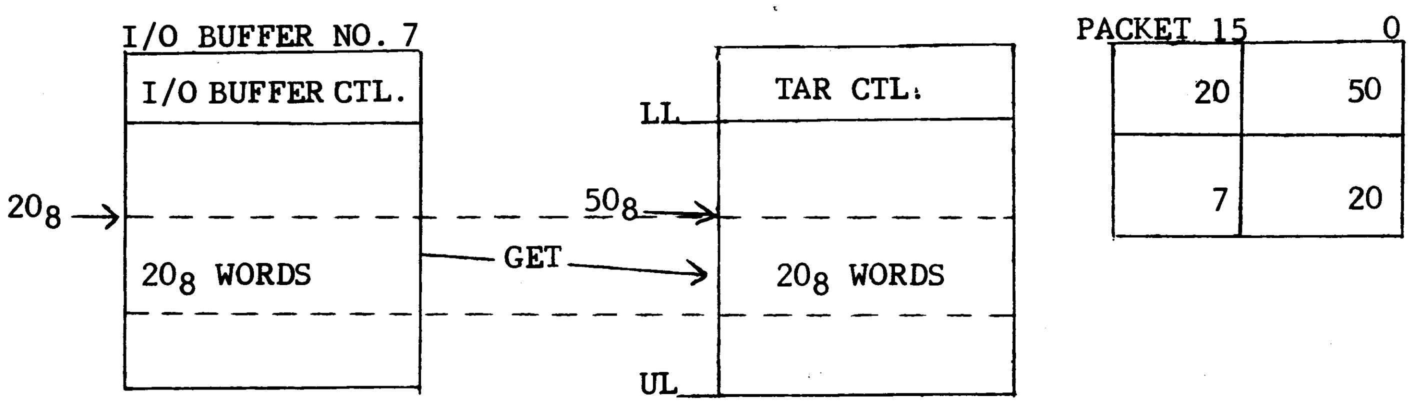
Word No. — The word number, relative to zero, in the buffer from which the transfer will start.

Done Address — Normal return.

The following conditions will cause a transaction abort:

1. The transfer would move data into an area outside the TAR.
2. The number of words to be transferred is larger than the data record requested.
3. The I/O buffer number cannot be found.

Example:



5.4 PUT

The PUT service request:

1. Transfers a specified amount of data from a location in the TAR to an area in a particular I/O buffer.

Packet:

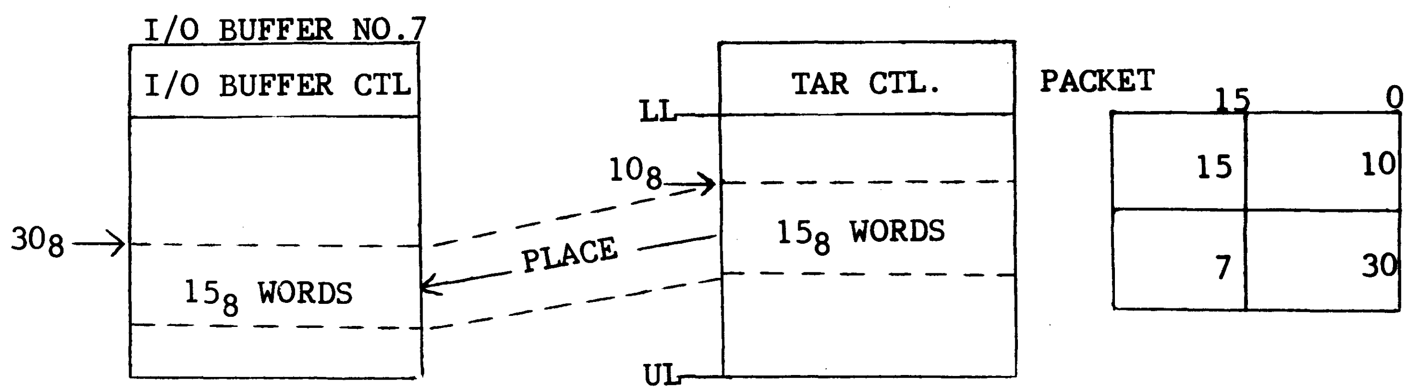
Ø	# OF WORDS	TAR LOCATION
1	BUFFER NO.	WORD NO.

# of Words	The number of words to be transferred from the TAR to the I/O Buffer.
TAR Location	The location of the TAR at which the transfer is to start.
Buffer No.	The number of the I/O Buffer into which the data is to be transferred.
Word No.	The word number, relative to zero, in the buffer at which the transfer will start.
Done Address	Normal return.

The following conditions will cause a transaction abort:

1. The transfer would move data from an area outside the TAR.
2. The number of words to be transferred is larger than the data record requested.
3. The I/O buffer number cannot be found.

Example:



5.5 COPY

The COPY service request:

1. Transfers one area of data from one location of an I/O buffer to another location of a different I/O buffer.

Packet:

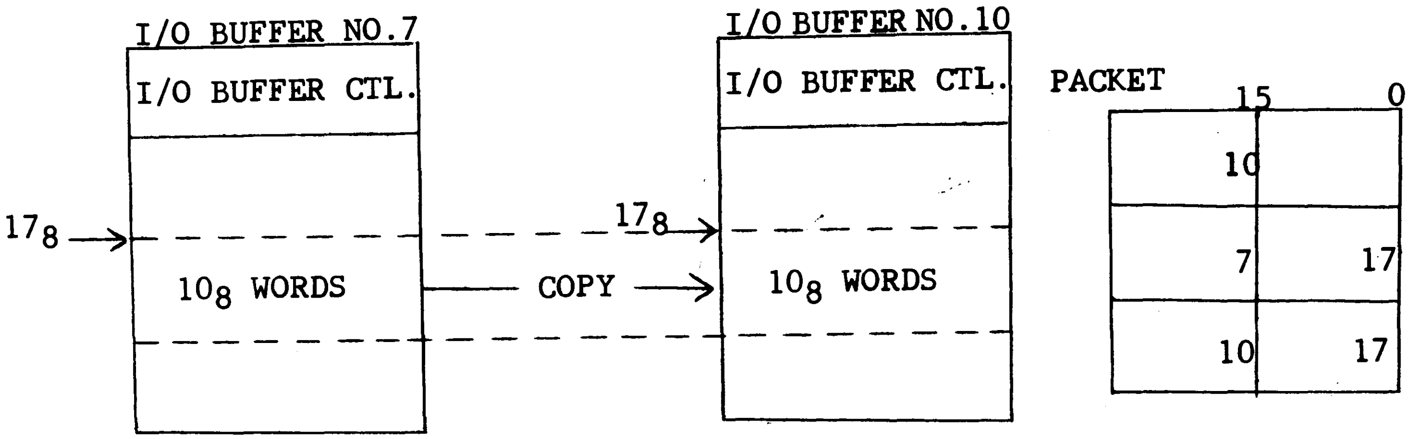
Ø	# OF WORDS	NOT USED
1	BUFFER NO. A	WORD NO. A
2	BUFFER NO. B	WORD NO. B

# of Words	The number of words of data to be transferred from one I/O Buffer to another.
Buffer No. A	The I/O Buffer No. from which the data is to be obtained.
Word No. A	The word number, relative to zero, in the buffer from which to transfer will start.
Buffer No. B	The I/O Buffer No. into which the data is to be transferred.
Word No. B	The word number, relative to zero, in the buffer at which the transfer will start.
Done Address	Normal return.

The following conditions will cause a transaction abort:

1. The number of words to be transferred will exceed one of the data record limits.
2. One of the specified buffer numbers cannot be found.

Example:


I/O BUFFER NO.7
I/O BUFFER CTL.
17_8
10_8 WORDS
COPY
17_8
I/O BUFFER NO.10
I/O BUFFER CTL.
10_8 WORDS
PACKET
15
0
10
7
17
10
17

5.6 SWAP

The SWAP service request:

1. Interchanges the specified areas of two data records.

Packet:

Ø	# OF WORDS	NOT USED
1	BUFFER NO. A	WORD NO. A
2	BUFFER NO. B	WORD NO. B

# of Words — The number of words of data to be transferred from one I/O Buffer to another.

Buffer No. A — The I/O Buffer No. from which the data is to be obtained.

Buffer No. B — The I/O Buffer No. into which the data is to be transferred.

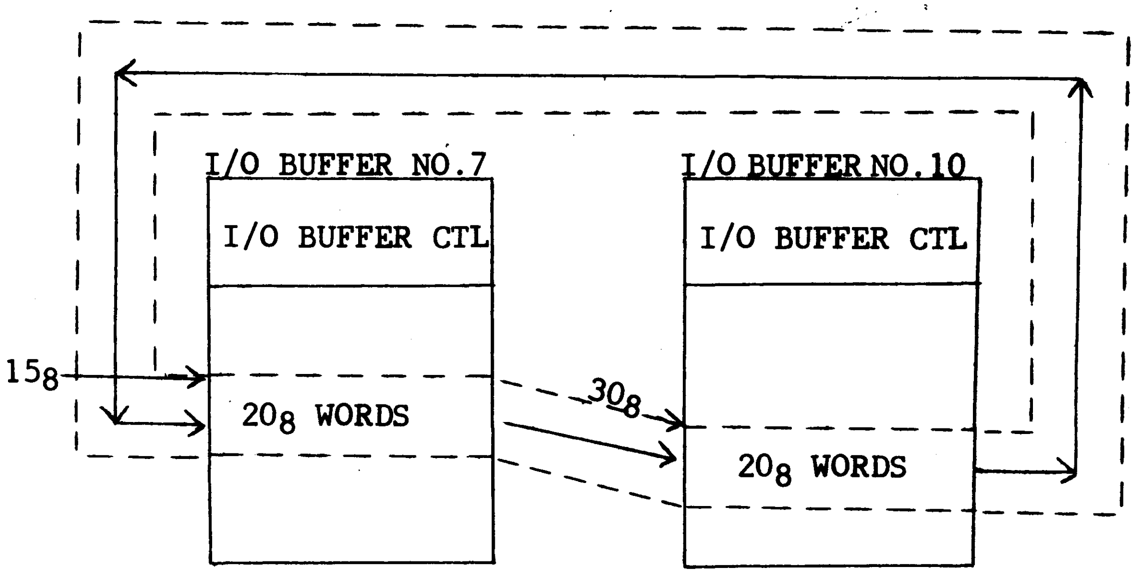
Word No. B — The word number, relative to zero, in the buffer at which the transfer will start.

Done Address — Normal return.

The following conditions will cause a transaction abort:

1. The number of words to be transferred will exceed one of the data record limits.
2. One of the specified buffer numbers cannot be found.

Example:



PACKET

15	0
20	
7	15
10	30

5.7 ACQUIRE SCRATCH FILE

The ACQUIRE SCRATCH FILE service request:

1. Acquires a scratch file of the requested size for the duration of the transaction.
2. Before looks and after looks of this file are not generated.
3. Assigns the file in requested modules of 1024_{10} words.

Note: All references to the acquired scratch file is by a user number/file number of 77777/nnnnn where nnnnn is the file mnemonic assigned by the TCS. At termination the scratch file is made available to other transactions.

Packet:

Ø	UNUSED	N MODULES

N Modules — The number of 1024 decimal word modules desired for this file.

Done Address — Communications Word 0 (CWORDO) will contain 77777 in the upper and the file mnemonic of the file that has been assigned by TCS in the lower.

The following conditions will cause a transaction abort:

1. If after a fixed delay period the request cannot be fulfilled.

5.8 WRITE SCRATCH

The WRITE SCRATCH service request:

1. Writes the entire contents of an I/O buffer to a specified allocated scratch file.
2. Releases the I/O buffer after the write.
3. Before looks and after looks are not generated.

Note: The only types of files that may be specified for this service request are those that were assigned by the ACQUIRE SCRATCH FILE service request.

Packet:

Ø	BUFFER NO.	UNUSED
1	USER NO.	FILE NO.
2	LOGICAL ADDRESS	

Buffer No. — The number of the I/O Buffer that was created for the read and returned to Communication Word 0. The entire I/O Buffer will be written.

User #/File # — The User number must be 77777 and the file # is the mnemonic that has been assigned by the scratch file allocator.

Logical Address — The relative address in the file at which the writing is to begin. The same as the logical address in Omega.

Done Address — Normal Return.

The following conditions will cause a transaction abort:

1. The I/O buffer is not valid.
2. The user number is not 77777.
3. The file mnemonic has not been assigned to this TAR by the ACQUIRE SCRATCH FILE service request.
4. The logical address in the scratch file is not valid.
5. The logical address plus the buffer size exceeds the boundry of the scratch file.
6. Omega has failed to do the write successfully.

5.9 ACQUIRE SEQUENTIAL

The ACQUIRE SEQUENTIAL service request:

1. Allocates a sequential file, if available, to the user.
2. Inserts the file code of the file into the original call.
3. Assigns the file codes as required by the service request.

Note: A fixed group of file codes will be assigned at TCS generation time for this service request.

Packet: Similar to that generated by SPURT.

```
#ASGΔOptionsΔPeripheral Name,
        ZZ,Estimate,File ID
```

A TPS should not execute an #ASG directly to Omega. It should deliver the packet to TCS. Refer to Omega for description of operands, except that a dummy file code of ZZ is used. In the TAR, the format might look as follows:

Ø	#	A	S	G	Δ
1	O	H	R	S	W
2	U	Δ	T	A	P
3	E	,	Z	Z	,
4	,	M	A	S	T
5	E	R	-	F	I
6	L	E	-	2	1

Done Address: Normal Return.
Communication Word Ø (CWORDØ) will contain file code assigned.

Note: File code assigned should be saved for subsequent use in releasing file.

The following conditions will cause a transaction abort:

1. A file code is not available for allocation.
2. Omega error return.

5.10 RELEASE SEQUENTIAL FILE

The RELEASE SEQUENTIAL FILE service request:

1. Releases the mnemonic assigned to a sequential file.
2. Makes the file code available to the system.

Packet: Similar to that generated by SPURT

#FREEΔOptionsΔZZ,File ID,Duration

A TPS should not issue a FREE request directly to Omega. It should deliver the packet to TCS. Refer to Omega for description of the operands, except that a dummy file code ZZ is used and must be changed by the TPS to the file code assigned by TCS for the related Acquire Sequential. In the TAR, the format might look as follows:

Ø	#	F	R	E	E
1	Δ	P	R	Δ	Z
2	Z	,	M	A	S
3	T	E	R	-	F
4	I	L	E	-	2
5	1	,	5	Δ	Δ

Done Address: Normal return.

The following condition will cause a transaction abort:

1. Invalid file code.

5.11 ACQUIRE BUFFER

The ACQUIRE BUFFER service request:

1. Assigns an I/O buffer of the requested size to the user.

Note: It may be used as a write scratch area as well as just as a working area.

Packet:

Ø	NOT USED	# OF WORDS

# of Words — The size of the I/O Buffer that is desired.

Done Address — Communications Word 0 (CWORDO) will contain the I/O Buffer number assigned.

The following conditions will cause a transaction abort:

1. The core required for the I/O buffer is larger than the limit.

5.12 RELEASE BUFFER

The RELEASE BUFFER service request:

1. Releases an I/O Buffer assigned to a transaction.

Note: If the I/O Buffer has previously been READ with lock the lock must be released prior to the releasing of the I/O Buffer.

Packet:

Ø	~~NOT USED~~ Options	BUFFER NO.

Buffer No. The number of the I/O Buffer to be released.

Done Address Normal return.

The following conditions will cause a transaction abort:

1. The buffer number is invalid.
~~2. The previous read with lock of this I/O buffer has not been released.~~

5.13 EXTENDED FILE

The EXTENDED FILE service request:

1. Extends the size of a mass storage file presently registered with TCS and Omega.

Packet:

Ø	USER #	FILE #
1	NOT USED	# OF WORDS

User #/File # — The User #/File # for the file to be expanded.

# of Words — The number of words by which the file is to be expanded.

Done Address — Normal Return.

The following conditions will cause a transaction abort:

1. User number or file number cannot be found in the TCS directory.

5.14 READ MESSAGE (READM)

The READ MESSAGE service request:

1. Transfers a specified number of words of an input message starting at word 6 of the Input Queue Buffer to a specified location in the TAR.

Note: The message may be partially or completely contained in the Input Queue Buffer.

Packet:

Ø	# OF WORDS	TAR LOCATION

/# of Words — The number of words of the input message that is to be transferred.

TAR LOCATION — The location in the TAR to which the transfer will start.

Done Address — Communications Word 0 (CWORDO) upper will contain the number of words transferred. Communications Word 0 (CWORDO) lower will be negative if the end of message has been reached.

The following conditions will cause a transaction abort:

1. The TAR location plus the number of words specified is outside the limits of the TAR.
2. The TAR location is outside the limits of the TAR.
3. A READM request is given and there are no more words of the input message available (end of message has been reached).

5.15 WRITE MESSAGE (WRITEM)

The WRITE MESSAGE service request:

1. Sets up a control block indicating the destination of the message and saves a few words of the message.
2. If the first part of a message; sets up the control block and passes the request to Omega.
3. If the continuation of a message; passes the request directly to Omega.

Packet:

Ø	CTM/UNIT I.D.	# OF WORDS
1	NOT USED	TAR LOCATION
2	OPTIONS	

CTM/UNIT I.D. — As required by Omega.

# of Words — The number of words of the output message that is to be transferred.

TAR Location — The location in the TAR from which the transfer will start.

Options — 1 - Initial message segment.
2 - Continuation of the message.

Done Address — Normal Return.

The following conditions will cause a transaction abort:

1. The TAR location plus the number of words specified is outside the limits of the TAR.
2. The TAR location specified is outside the limits of the TAR.

5.16 ABORT 1

The ABORT 1 service requests:

1. Performs a join to insure the completion of any parallel I/O.
2. If the EOM of an input message has not been reached, reads the message until the EOM condition is found.
3. Releases any file allocated to roll out this TAR.
4. Releases all the locked out records for this transaction.
5. Restores the files to their initial condition from the "before look" scratch file.
6. De-allocates from the TAR all I/O buffers.
7. Sends a message to each CTM/unit ID to which output messages were queued that the transaction was aborted and identifying the message by the first part.
8. All control blocks for the output messages are de-allocated.
9. Any scratch files allocated are released.
10. Releases any files locked out by the transaction.
11. If logical switch A or C is on, sends the TAR statistics to the primary output.
12. Sends an error message to the primary output and console.
13. Dumps the contents of the control and working storage sections of the TAR I/O buffers, etc. on the primary output.
14. The TAR and its storage are released.
15. The TPS is released.
16. The activity is terminated.

5.17 CLEAN-UP

The CLEAN-UP Service Request:

1. Validates that the associated TAR is not presently Rolled-OUT.
2. Validates that the input message has been completely read.
3. Determines if transaction was a "test"; if so, performs an abort.
4. Releases any of the following if they exist for this TAR:

   a) Output message control blocks.
   b) I/O buffers.
   c) Locklist entries.
   d) Before-Look file.
   e) Scratch file(s).
   f) Files locked out.

5. Deallocates the TPS area if possible.
6. Updates statistics kept by the system.
7. Sends statistics to primary output (dependent on logical switch settings).
8. Deallocates the TAR area.
9. Writes message on the audit trail noting successful completion of transaction.
10. Terminates the activity.

The following will cause a transaction abort:

1. If the message has not been completely read from the Input Queue.

5.18 TAR EXPANSION

The TAR EXPANSION Service Request:

1. Verifies number of expansion words and, if necessary, rounds this amount upward to some multiple of 100_8.
2. Secures new enlarged area TAR for requesting transaction and performs necessary housekeeping.

Packet:

Ø	NOT USED	# WORDS

Words — Number of additional words required. This number is rounded upwards to a multiple of 100_8 words.

Done Address — Normal return.

The following conditions will cause a transaction abort:

1. Number of expansion words invalid.

5.19 TAR CONTRACTION

The TAR CONTRACTION service request:

1. Verifies number of contraction words and, if necessary, rounds this amount upwards to a multiple of 100_8.
2. Releases specified amount of core from the TAR to free core.

Packet:

Ø	NOT USED	# WORDS

Words — Number of words to reduce TAR area by.

Done Address — Normal Return.

5.20 TPS JUMP

The TPS JUMP service request:

1. Verifies that the requesting TPS is a TYPE 1 TPS.
2. Verifies that the requested TPS is a TYPE 1 TPS.
3. The requested TPS is called and activated.

Packet:

Ø	NAME
1	NAME
2	VERSION

Name — Name of desired TPS as registered with Omega.

Version — Version of desired TPS as registered with Omega.

The following conditions will cause a transaction abort:

1. Requesting or requested TPS is not a TYPE 1 TPS.
2. Requested TPS can not be found.

5.21 TIMER TABLE CHANGE

The TIMER TABLE CHANGE service request:

1. Changes the time of day and priority value for the particular entry in the TIMER TABLE.

2. Rearranges entries in TIMER TABLE in proper sequence based on the Time of Day.

Packet:

Word		
Ø	TIME OF DAY	
1		
2	STARTER VALUE	PRIORITY

Time of Day — New Time of Day to be entered in Timer Table.

Starter Value — Value used to initiate the TPS.

Priority — Rated importance of entry.

Done Address — Normal Return.

The following condition will cause a transaction abort:

1. The starter value specified cannot be found in any entry on the TIMER TABLE.

6. OMEGA SERVICE REQUESTS UNDER TCS

Since TCS is a task running under Omega, a TPS can not use certain task oriented Omega service requests.

6.1 Forbidden Omega Service Requests:

#COR
RETURN
ABORT
ERROR
RETURN1
PUSH
POP
QREF1
REGCT
DSWITCH
DRETURN
ERRADD
OPPADD
UNSOL
MADD
MREL
FCHAIN
RCHAIN
MEMADD
MEMREL
TESTFOF
TESTFUF

6.2 Not recommended Omega Service Request - these can be used, but require conventions to prevent concurrent transactions under TCS from conflicting. Improper use can hang up transaction processing. Their use is discouraged.

#PRAM
#CALL
#ASG
#FREE
#SWITCH
#DEL
XOFF
XON
XTEST

REGQ
QREF
FORK
JOIN
(REGQ, QREF, FORK, JOIN: The new activities can't use any TCS service requests)
TCORE
SET15
SET17
TFC
TIMEL

6.3 Omega Service Requests that can be freely used by TPS's:

#START
#MSG
#LOG
#DUMP
LOAD
LOADA
FETCH
SEND
RECEIVE
(SEND, RECEIVE: Must use unique ID number)
DELAY
TIMED
DATIM
MSG
MOUNT
DISMOUNT
CHANGE

7. TCS SIMULATOR

The TCS Simulator is designed to permit testing of TPS's prior to the implementation of the Transaction Control System. The functions available in the TCS system are implemented to a degree in the Simulator.

The user is allowed to set up his own files, test data, and perform all the functions permitted. In addition, controlled dumps can be taken at any point to aid in debugging.

Formatting is the same for the Simulator as for TCS and service requests are generated by macros. Therefore, whenever a TPS is written and debugged, it can be run under TCS without any change except to use a different set of macros.

Each test is run under the Simulator as a separate JOB. One or several TPS's may be loaded and tested, but these will all be grouped in one JOB deck.

The Simulator uses certain control cards which are peculiar to it alone. These cards follow a # GO card and are followed by a # END card. The Simulator operates as a task and the control cards are read from the I/O cooperative system of OMEGA.

The control cards are:

- START - Defines the start of a test and sets up a TAR
- FILE - Sets up a file by User number/File number
- DATA - Defines records or data to be loaded into a file
- MSG - Specifies message header
- MD - Defines body of a message
- TAR - Defines data and/or values to be placed into a TAR
- DUMP - Defines TAR or file areas to be printed
- TPS - Specifies a TPS to load
- GO - Directs the simulator to execute the currently loaded TPS

Each of these control cards are described in detail.

7.1 TCS SERVICE REQUESTS

The functions (service requests) implemented in the Simulator are as follows:

READ - Requests the assignment of an I/O buffer from Free Core. Transfers the data from a specified area of a designated file into the I/O buffer.

WRITE - Transfers the specified data back to mass storage, releases the I/O buffer assigned for this area.

GET - Transfers a specified number of words from an I/O buffer to a specific location in the TAR.

PUT - Transfers a specified number of words from a location in the TAR to an area in a designated I/O buffer.

COPY - Transfers a specified number of words from a designated I/O buffer location to some other designated I/O buffer location.

ACQUIRE SCRATCH - Acquires a scratch file of size requested. File is assigned in modules of 1024 words. Before and after looks are not generated, and the file is released when the transaction terminates.

WRITE SCRATCH - Writes contents of I/O buffer to the designated scratch file. I/O buffer is released after write. Before and after looks are not generated. Only files assigned by the ACQUIRE SCRATCH may be specified in this request.

ACQUIRE SEQUENTIAL - Allocates a magnetic tape file if one is available. If non available, the transaction aborts. The selected (allocated) file code is placed in the original call and assigns file codes required by the service request.

RELEASE SEQUENTIAL - Releases the mnemonic assigned to the designated file and makes the file code available to the system.

ACQUIRE BUFFER - Assigns an I/O buffer of the requested size to the requester.

RELEASE BUFFER - Releases an I/O buffer assigned to a transaction.

EXTEND FILE - Extends the size of a mass storage file currently registered with TCS and OMEGA.

SWAP - Exchanges a specified number of words between two I/O buffers.

READM - Reads a specified number of words from the input message into the TAR.

WRITEM - Takes a message or part of a message from the TAR and prints it.

ABORT1 - Aborts the current test transaction and dumps core areas.

CLEAN-UP - Releases the files and core used by the current test transaction.

TPS JUMP - Not implemented as such, causes the next control card to be executed, thus chaining of TPS's is done through card control.

More detail on the Service Requests is contained in Section 5 of the User's Manual.

7.2 CARD LAYOUTS FOR THE TCS SIMULATOR

A. The card format is similar to SPURT except it starts in column 1, columns 1-5 contain the card name and the operands start in column 6.

```
1   5 6              80
CTYPE  VO*V1*V2 .... *VN
```

B. If continuation of the operands onto the succeeding card is desired, a "#" is punched in column 80. The operands on the next card begin in column 6.

C. If a "@" follows the last operand, notes can be added.

D. The DATA, MD, and TAR cards are used to input data. The data operands in these cards can have the following formats:

Numbers - will be converted to octal and stored in 1 or 2 words as required. A number followed by a "D" will be converted to octal. Examples:

```
      +1234567
12345671234567
      -5555558D
     999999999D
```

Floating Point Numbers - $\pm$nnnn.nn E$\pm$ee will be converted to a two word floating point number.

Internal Decimal Constants - will be stored as a two word Fieldata Encoded Decimal Number.

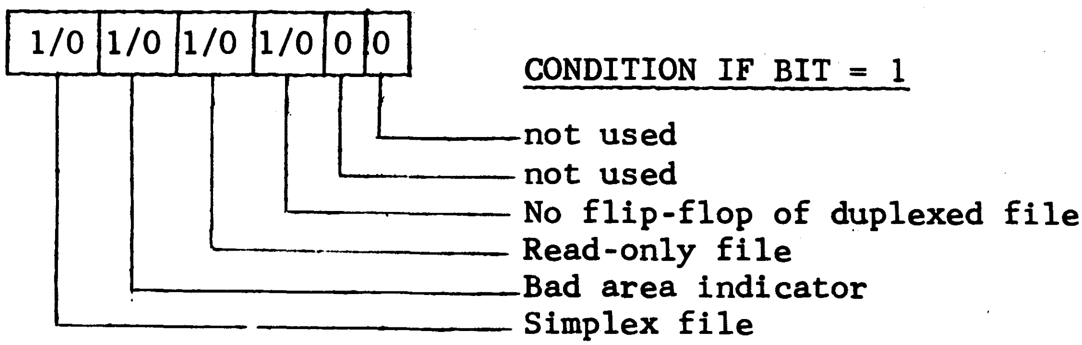
Alphanumeric Data - any operand which includes an alphanumeric character except a code as the last character or as a floating point number will be stored as Fieldata in as many words as required, left justified.

7.2.1 FILE CARD

FILE V_0* V_1* V_2* V_3* V_4* V_5@

V_0 will be the User number/File number in the format U/F.
V_1 will be the peripheral type (standard Omega).
V_2 will be the size of the file (# words octal or decimal).
V_3 will be the Control Flags.*
V_4 will be optional - C to clear file area, blank to leave area alone.
V_5 will be optional - J to hold file for entire job.

* Control Flags - two octal digits whose bits have the following meaning:



7.2.2 DATA CARD (DATA FOR FILE)

D   V0* V1* V2* V3 ... Vn@
(or)
DATA V0* V1* V2* V3 ... Vn@

V0 - User number/File number in format U/F.
V1 - The logical address of the file where the data (V3-Vn) is to start loading. If blank, the data will be loaded following the preceeding "D" card, no other cards may intervene.
V2 - Number of words contained in V3 (or 0 indicating no verification).
V3-Vn - Data operands to be loaded into the file.

7.2.3 MESSAGE HEADER CARD

MSG V0*V1*V2*V3@

V0 - The CTM/ID
V1 - Time of Message (in the form HH:MM)
V2 - Starter value
V3 - Message number

7.2.4 MESSAGE DATA CARD

MD V0*V1*V2@

V0 - Message number (same as V3 above)
V1 - Optional - can be used as sequence number, if desired, to keep messages of multiple cards in order or can be blank "**".
V2 - Message text using alphanumeric operands.

All MD cards must be grouped consecutively in the deck.

7.2.5 START CARD

START V0*V1*V2@

V0 - Name/Version of the first TPS to be loaded and executed.
V1 - TPS type (1 or 2).
V2 - If V1 is type 1, the size of the TAR scratch area.

The START card denotes the beginning of a test. When aborts, etc. occur the simulator reads ahead to the next START card.

7.2.6 TAR CARD

TAR V0*V1*V2* ...*Vn@

V0 - This is the relative address of the TAR working storage area where V1 will be placed (relative to LL).
V1-Vn - Data operands to be placed in TAR working storage starting at location specified by V0.

7.2.7 TPS CARD

TPS   V0@

V0 - The name/version of the TPS to be loaded.

7.2.8 GO CARD

GO@

No operands. Activates the currently loaded TPS.

7.2.9 DUMP CARD

DUMP   V0*V1*V2@

V0 - User number/File number in the format U/F.*
V1 - Number of words to be dumped.
V2 - Beginning address of area to be dumped.

* If a complete TAR dump is required, V0 will be "TAR". V1 and V2 will not be required in this case.

7.3 PREPARING A TEST DECK

Each test run under the TCS Simulator is a separate JOB. This JOB may actually cause the loading and testing of many TPS's and files, but all are grouped together in one JOB deck.

The TCS Simulator uses certain control cards which it alone recognizes and processes. These cards will follow the #GO card and a #END card will follow them. As can be seen, the TCS Simulator operates as a task and reads the control cards from the I/O co-operative system of OMEGA.

The following is an example of a JOB deck to compile two TPS's, load two files, test the first TPS, load another file, and test the other TPS:

```
#JOB
#SPURT YZR TPS1/RB
      Spurt coding for TPS1
#END
#SPURT YZR TPS2/RB
      Spurt coding for TPS2
#END
#ASG H TAPE,AA,TCS-LIB
#IN RX AA,TCSIM/ABS            put simulator in JOB library
#IN X AA,SRINT/RB              put serv. req. interface in JOB lib.
#FREE AA
#LOAD LY TPS1/RB,TPS1/ABS
#LOAD LY TPS2/RB,TPS2/ABS
#GO ABCM TCSIM/ABS
MSG    142*13:20*1*1234
MD     1234*1*HEADER,TO ADDRESS,FROM ADDRESS
MD     1234*2*CUST XXX,SHIP TO,500 #1432 WIDGETS
FILE   1/9D*DRUM*20000D*40*C
DATA   1/9D*200*1432D*WIDGETS*25D*35D*0*0@
...
DATA   1/9D**1732D*DODADS*75D*95D*0*0@
FILE   1/10D*FAST*1000000*40*C
....
D      1/10D*0104*0*0*DEPT01-04TOTALS@
START TPS1/ABS*1*300
TAR    177*20
DUMP   TAR
DUMP   1/9D*20000D*0
DUMP   1/10D*1000D*0
```

(CONTINUED)

```
TPS    TPS1/ABS
GO
DUMP   TAR
DUMP   1/9D*20000D*0
DUMP   1/10D*1000D*0
FILE   1/11D*DRUM*1000D*00*C
DATA   1/11D*0*25*26D*12345I*256.78E+3@
...
DATA   1/11D*775*EFGH*26*27D*12346I*256.79E-4
TAR    100*EFGH
TPS    TPS2/ABS
GO
DUMP   1/10D*500D*100D
DUMP   1/11D*500D*100D
#END
```