COMMAND EXECUTIVE

MAIN PROCEDURE OF EVERY PROCESS AT AN ITP WORK STATION

CONTROLS THE EXECUTION OF TRANSACTIONS

PLUS MUCH MORE

COMMAND EXECUTIVE FUNCTIONS

- WORK STATION AND PROCESS INITIALIZATION
- CONTROLS THE EXECUTION OF TRANSACTIONS
- CONTROLS THE MULTIPLEXING OF USERS (TENANTS) ACROSS PROCESSES
- TPR FUNCTIONS:
  - .CANCL
  - .FREE
  - .TRMTN
  - ETC.
- COBOL SEND/RECEIVE
- TPR MEMORY MANAGEMENT
- FAULT HANDLING
- PROCESS WRAPUP
- HONEYWELL SUPPLIED TPR's

COMMAND EXECUTIVE COMPONENTS

SHARED CODE:	
INITIALIZATION	2000
ALL OTHER	7000
TOTAL	9000
INITIALIZATION TPR (TP1H)	2700
GMAP RUN TIME	2400
TOTAL INITIALIZATION TPR	5100
TPR's	
TP-BIO	1200
TP-INI	500
TP-ABT	1000
TP-LOF	600
TP-TWS	500
TP-AWS	500
TOTAL TPR's	4300

INITIALIZATION

ITP WORKSTATION PROCESSES ARE SPAWNED AUTOMATICALLY BY TENANT MANAGEMENT

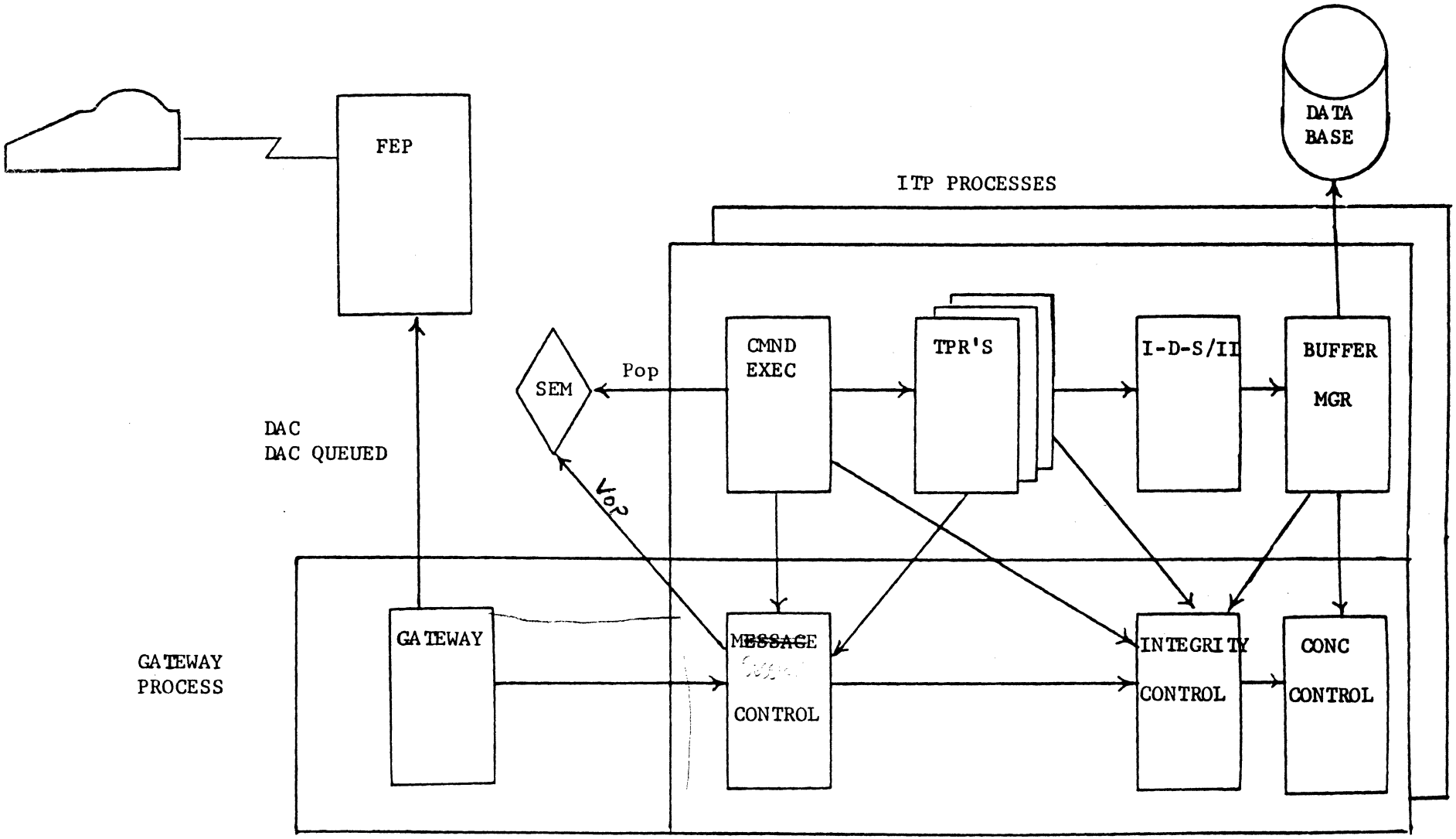
THE FIRST PROCESS STARTED AT THE WORKSTATION IS CALLED THE INSTIGATOR

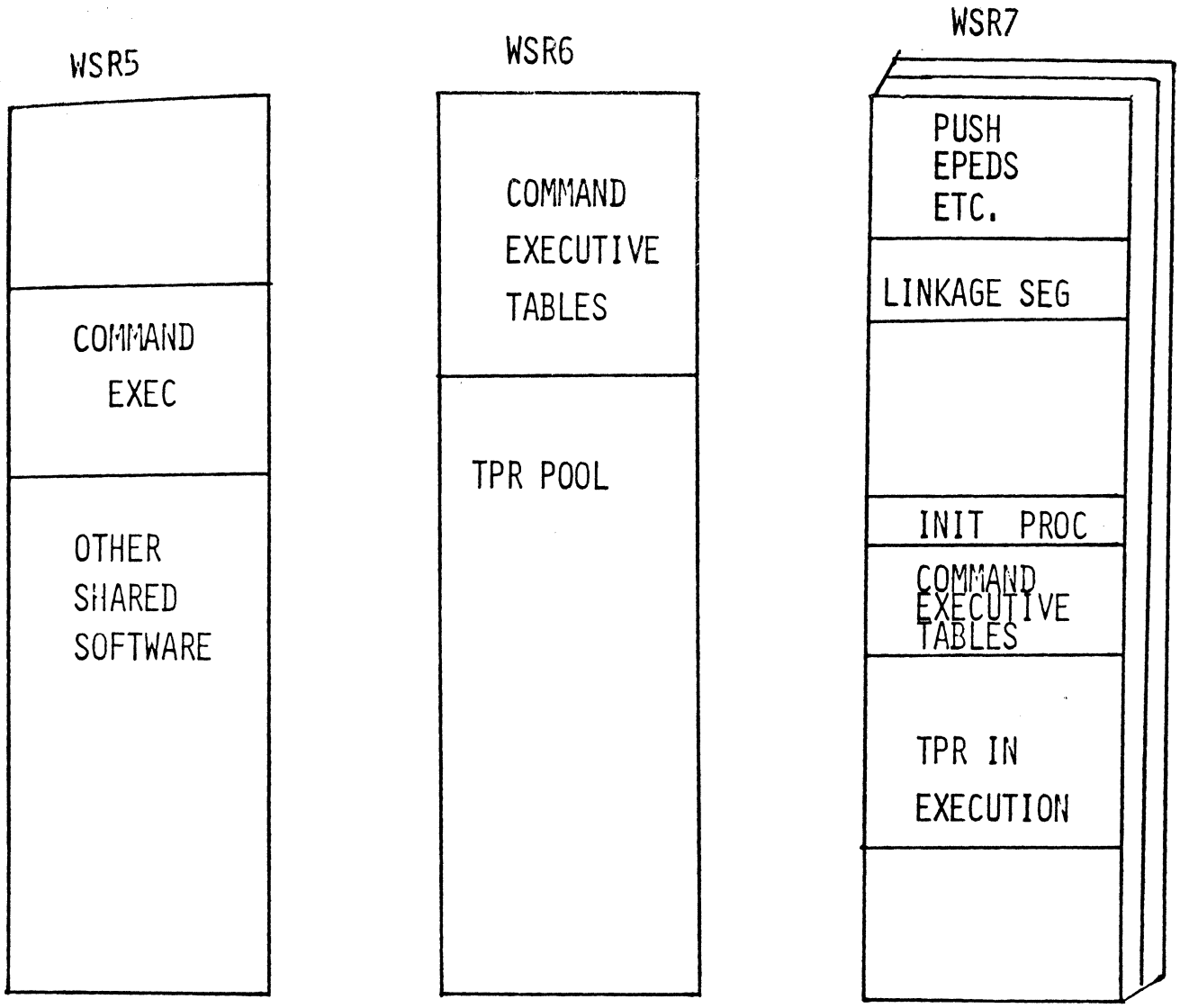
THE INSTIGATOR IS SPAWNED WHEN THE FIRST USER LOGS ON TO AN ITP WORKSTATION

WORKSTATION MANAGEMENT INITIALIZES SOME TABLES FOR THE CX WHEN THE WORKSTATION IS STARTED (THE INSTIGATOR IS STARTED).

THESE TABLES PROVIDE GATING AND TEMPORARY STORAGE FOR THE CX

TRANSACTION FLOW




WSR5
COMMAND
EXEC
OTHER
SHARED
SOFTWARE
WSR6
COMMAND
EXECUTIVE
TABLES
TPR POOL
WSR7
PUSH
EPEDS
ETC.
LINKAGE SEG
INIT PROC
COMMAND
EXECUTIVE
TABLES
TPR IN
EXECUTION

INITIALIZATION CONTINUED

THE INSTIGATOR INITIALIZES THE WORKSTATION STRUCTURES AS FOLLOWS:

- BOOTSTRAP VERSIONS OF THE CX TABLES ARE BUILT
- INITIALIZATION TPR IS LOADED
- ITP EXTENSION OF THE WORKSTATION DATA BASE IS READ
- GMAP RUN TIME ROUTINES ARE CALLED TO CREATE AND AND INITIALIZE SEGMENTS

THE INSTIGATOR AND EVERY OTHER PROCESS INITIALIZES THE PROCESS LOCAL STRUCTURES AS FOLLOWS:

- THE CX LINKAGE SEGMENT IS INITIALIZED
- PROCESS LOCAL SEGMENTS ARE CREATED AND INITIALIZED
- THE READY TPR(s) ARE EXECUTED TO:
  - READY THE AREAS FOR EACH SUBSCHEMA
  - INITIALIZE GLOBAL STORAGE

CONTROL IS PASSED TO THE MAIN PROCEDURE OF CX

TENANT MULTIPLEXING

TENANT CREATION

TENANT MANAGEMENT CREATES A TENANT WHEN A NEW USER LOGS ON THE ITP WORKSTATION

TENANT MANAGEMENT NOTIFIES THE ITP WORKSTATION THAT A NEW USER HAS LOGGED ON THROUGH THE WORKSTATION SEMAPHORE

ONE OF THE ITP PROCESSES RESPONDS TO THE NOTIFICATION:
THE CX MAPS THE TENANT TO THE PROCESS

THE LOG-ON TPR (TP-INI) IS EXECUTED:

- A SESSION CONTROL ACCEPT IS ISSUED
- VALIDITY CHECKING IS PERFORMED
- SECURITY PARAMETERS ARE ESTABLISHED FOR THE TENANT
- AN LID IS ESTABLISHED

TENANT IS UNMAPPED FROM THE PROCESS

TENANT MULTIPLEXING

TRANSACTION CONTROL

TENANT MANAGEMENT NOTIFIES THE ITP WORKSTATION WHEN
AN END QUARANTINE UNIT HAS BEEN RECEIVED BY
SESSION CONTROL

ONE OF THE ITP PROCESSES RESPONDS TO THE NOTIFICATION:
THE CX MAPS THE TENANT TO THE PROCESS
A PRERECEIVE OF THE FIRST RECORD IS DONE
A SCAN IS MADE TO LOCATE THE COMMAND
THE COMMAND IS INTERPRETED
IMPLICIT RECEIVE IS DONE IF REQUIRED
VIRTUAL RESOURCES ARE ALLOCATED IF REQUIRED
GLOBAL STORAGE IS ALLOCATED IF REQUIRED
THE INITIAL TPR OF THE TRANSACTION IS LOADED
AND EXECUTED
THE TPR IS UNLOADED
IMPLICIT SEND IS DONE IF REQUIRED
ONE OF FIVE ACTIONS CAN NOW TAKE PLACE:

1. TRANSACTION TERMINATION (COMMITMENT)
2. TRANSACTION ABORT
3. CONVERSATION WITH COMMITMENT
4. DEFERRED COMMITMENT (CHECKPOINT)
5. TPR CHAINING

TENANT MULTIPLEXING

TRANSACTION ABORT

TWO KINDS OF ABORT: FATAL AND RESTARTABLE

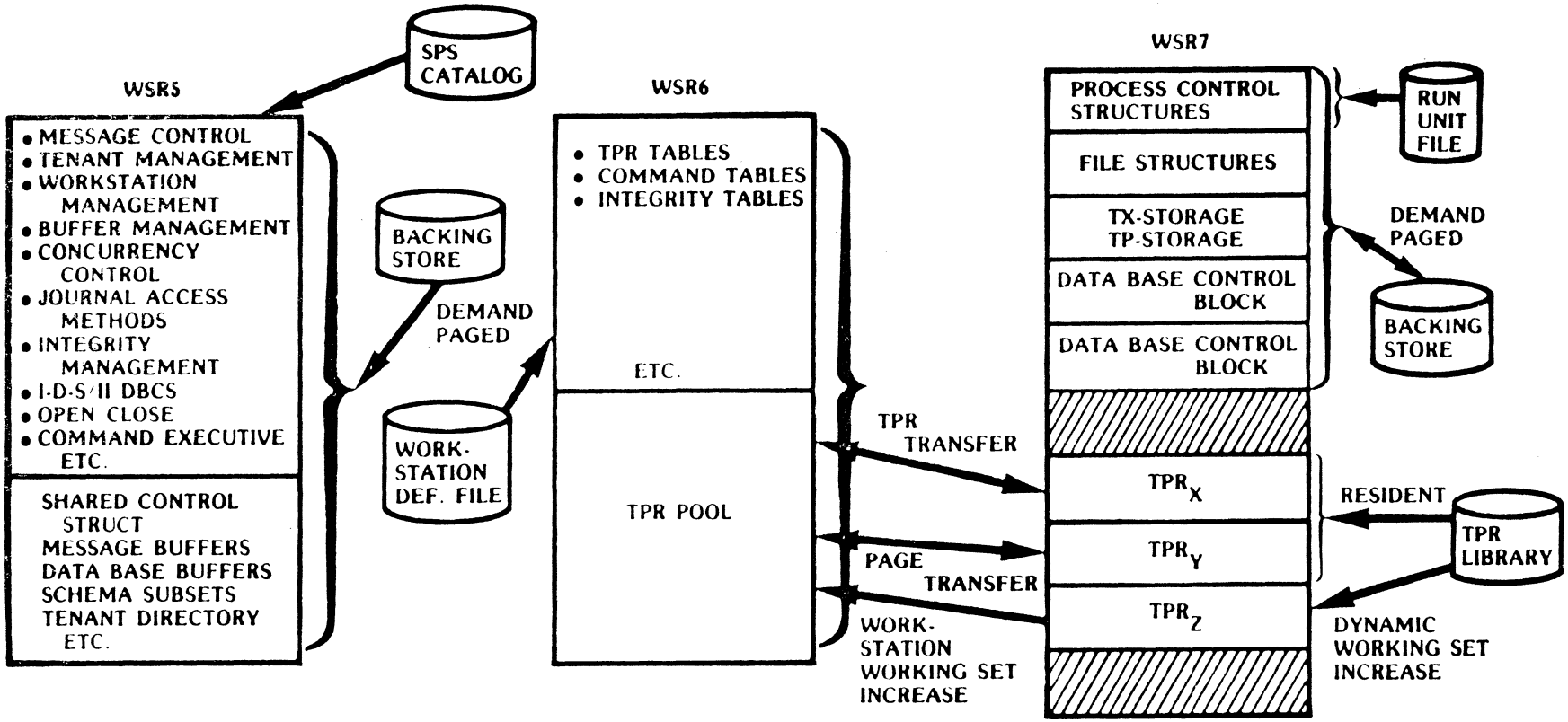
FATAL ABORT:

- ROLLBACK IS DONE
- ON-ABORT TPR IS LOADED AND EXECUTED
- COMMITMENT IS DONE
- TENANT IS UNMAPPED FROM THE PROCESS

RESTARTABLE ABORT:

- ROLLBACK IS DONE
- REQUEUING OF LAST INPUT MESSAGE IS REQUESTED
- AN ATTEMPT IS MADE TO UNMAP TENANT:
  - SUCCESSFUL IF INPUT NOT REQUEUED
  - TENANT STAYS MAPPED IF INPUT IS REQUEUED

PHYSICAL LAYOUT



TPR FUNCTIONS

TWO KINDS:

IMMEDIATE

DEFERRED - EXECUTED WHEN TPR TERMINATES

IMMEDIATE FUNCTIONS:

.UNLKV - UNLOCK VIRTUAL RESOURCES

.ABORT - FATAL ABORT

.CHANC - CHANGE GLOBAL STORAGE

.DEFNG - DEFINE GLOBAL STORAGE

DEFERRED FUNCTIONS:

.FREE - DEFERRED COMMITMENT (CHECKPOINT)

.FREER - CONVERSATION WITH

.UNLKV - UNLOCK VIRTUAL RESOURCES

.CANCL - CANCEL TPR

.RBORT - RESTARTABLE ABORT

.TRMTN - TERMINATE TENANT

FAULT HANDLING

BASIC MECHANISM IS EPED's

WHEN A FAULT OCCURS, CX GETS CALLED

IF FAULT OCCURED BEHIND WSR7, A CHECK IS MADE
IF TPR HAS A VALID FAULT VECTOR ENTRY.

IF SO, CONTROL IS TRANSFERRED TO THE TPR
FOR FAULT HANDLING

IF NOT, THE TRANSACTION IS ABORTED.

IF FAULT OCCURED BEHIND ANOTHER WSR, THE PROCESS IS
ABORTED.

IMPLICIT/EXPLICIT SEND AND RECEIVE

	SEND		RECEIVE	
	INPUTS	OUTPUTS	INPUTS	OUTPUTS
PARAM	o ADVANCING IND. o ADVANCING CT. o SEND IND.(EXP)	o STATUS	o ENCLOSURE	o STATUS
CD	o DEST.CT. o TEXT LNGTH. o LID o SUBID	o STATUS KEY		o LID o SUBID o TEXT LNGTH o END KEY
BUFFER	o BUFFER			o BUFFER
TP-STORAGE	o SEND IND.(IMP)	o ABORT CODE o ITP RET CODE		o ABORT CODE o ITP RET CODE

IMPLICIT/EXPLICIT SEND AND RECEIVE

	SEND	RECEIVE
IMPLICIT	o COMMAND TERMINATION CONVERSATION W/COMMIT DEFERRED COMMIT TPR-CHAINING	o NEW TX RESPONSE FROM CONVERSATION
	o IFF OUTPUT BUFFER SIZE ≠ Ø AND SEND INDICATOR ≠ Ø	o IFF INPUT (REPLY) BUFFER SIZE ≠ Ø
EXPLICIT	o CLIMB TO CX DOMAIN	o CLIMB TO CX DOMAIN
	o COBOL "SEND" VERB A RUNTIME ROUTINE MAPS "SEND" TO A CLIMB	o COBOL "RECEIVE" VERB A RUNTIME ROUTINE MAPS "RECEIVE" TO A CLIMB
IMPLICIT AND EXPLICIT	o CX PERFORMS VIA COMMON ROUTINES	o CX PERFORMS VIA COMMON ROUTINES

IMPLICIT/EXPLICIT SEND AND RECEIVE

SEND INDICATORS

		IMPLICIT	EXPLICIT
PORTION	0	NO	YES
ESI	1	YES	YES
EMI	2	YES	YES
EGI	3	YES	YES
RFU	4	NO	NO
PURGE	5	NOT YET	NO

EDIT/NO EDIT MODE

- EDIT MODE IMPLIES THE AFTER ADVANCING AND BEFORE ADVANCING CHARACTERS ARE GENERATED BY THE CX SEND ROUTINES
- DEFAULT IS EDIT MODE
- ENTERING AND EXITING EDIT MODE IS CONTROLLED BY TYPES OF SEND INDICATORS USED:
  - SEND PORTION — LEAVE EDIT MODE
  - SEND SEGMENT — ENTER EDIT MODE
  - OTHERS — NO CHANGE IN MODE
- TO BE ADDED: EDIT MODE SPECIFICATION BY TX TYPE

PROCESS WRAPUP

CLEANUP OF A TRANSACTION INTERRUPTED BY A PROCESS ABORT IS DONE IN WRAPUP

THE ROLLBACK OF THE TRANSACTION IS DONE BY INTEGRITY MANAGEMENT DURING THE FIRST ROLLCALL BY SHARED SOFTWARE TERMINATION

THEN PROCESS WRAPUP IS PAID TO THE COMMAND EXECUTIVE IF A TPR WAS IN EXECUTION AT THE TIME OF THE ABORT, IT IS UNLOADED.

THE PROPER ON-ABORT TPR IS EXECUTED

HONEYWELL SUPPLIED TPR's

TP-BIO - CONTROLS BIBO EXECUTION

TP-INI - LOG-ON TPR

TP-ABT - DEFAULT ON-ABORT TPR

TP-LOF - LOG OFF TPR ($BYE)

TP-TWS - TERMINATE WORKSTATION ($TERM)

TP-AWS - ABORT WORKSTATION ($ABRT)

TENANT MANAGEMENT

- WHAT IS TENANT MANAGEMENT
- REVIEW OF WORK STATIONS, TENANTS, AND OTHER STUFF
- MAJOR FUNCTIONS
- ALL FUNCTIONS
- AUTOMATIC SPAWNING AND TERMINATION OF PROCESSES
- INTERNAL STRUCTURES

1-31-80
BGT

WHAT IS TENANT MANAGEMENT

- MONITOR CONTROLLING CREATION, DELETION AND STATE CHANGES OF TENANTS.
- SINGLE DOMAIN OF ITP SHARED SOFTWARE
- PACKAGING:
  - SINGLE MODULE (i.e. single object deck)
  - part of SC 1.0 subcatalog
  - 2 to 2.5 K
  - GMAP

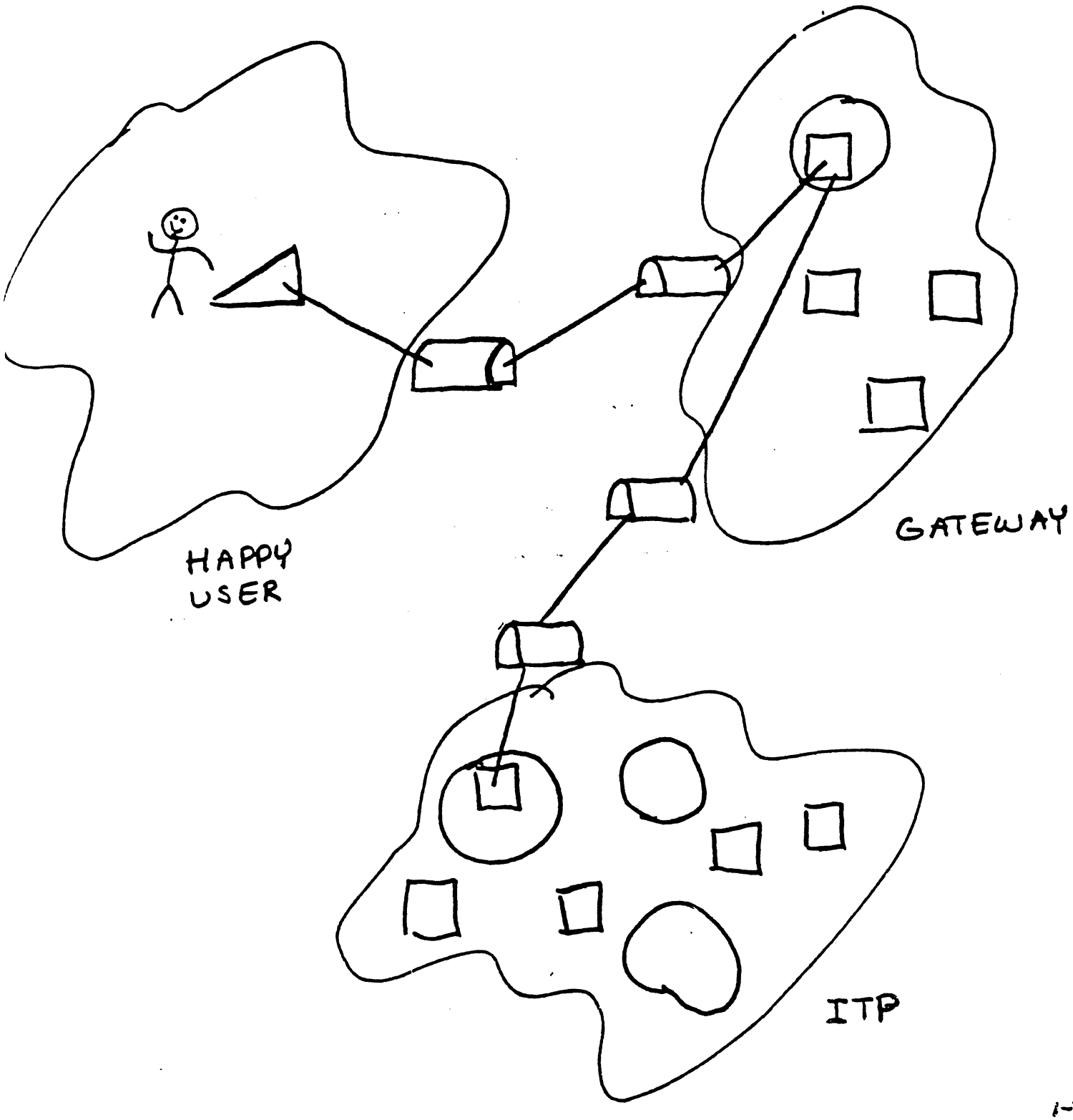
REVIEW OF WORK STATIONS, TENANTS, AND OTHER STUFF

- VIEWS OF WORK STATIONS
  - N IDENTICAL PROCESSES
  - POINT OF COMMUNICATION
  - SET OF CAPABILITIES
- TENANT IS LOGICAL REPRESENTATION OF A USER
  - MANY TENANTS MULTIPLEXED ACROSS N PROCESSES OF A WORK STATION
  - COMMUNICATION BETWEEN TENANTS
  - LOGICALLY ATTACHED TO A WORK STATION
- TYPES OF WORK STATIONS

TYPE	EXAMPLE	# PROCESSES	# TENANTS
1	Batch	1	1
2	Gateway	1	M
3	New Timesharing	N	N
4	ITP	N	M

- MESSAGE GROUPS
- WS SEMAPHORE
  - POP
  - V-OP

REVIEW OF WORK STATIONS, TENANTS, AND OTHER STUFF (CONT'D)



MAJOR FUNCTIONS

- ASSIGN TENANT
  - CREATION OF TENANTS
  - EXPLICIT OR IMPLICIT
- MAP TENANT
  - WS PROCESS WANTS TO DO WORK ON BEHALF OF A TENANT
  - TENANT ID INTO PCB
  - WS SEMAPHORE CONTROL
  - ONLY TYPES 2 + 4
- UNMAP TENANT
  - WS PROCESS IS DONE DOING WORK ON BEHALF OF A TENANT
  - TENANT ID IN PCB ZEROED
  - MARKING
- NOTIFY TENANT
  - V-OP on WS SEMAPHORE
  - IMPLICIT TENANT CREATION
  - AUTOMATIC PROCESS SPAWNING

ALL FUNCTIONS

- ASSIGN TENANT
- DELETE TENANT
  - MIGHT DELETE WS
- MAP TENANT
- NOTIFY TENANT
- P-OP
  - AUTO SPAWN
  - PMME FINISH
- ENABLE WORK STATION
- DISABLE WORK STATION
- T-OP
- UNMAP TENANT
- UNCONDITIONAL UNMAP TENANT

AUTOMATIC SPAWNING AND TERMINATION OF PROCESSES

- CAN'T MIX WITH MANUAL SPAWN
- ONLY FOR TYPE 4 WS
- SPAWN RULES:

  CURRENT < MIN

  MIN ≤ CURRENT < NORM
  AND NO PROCESSES WAITING

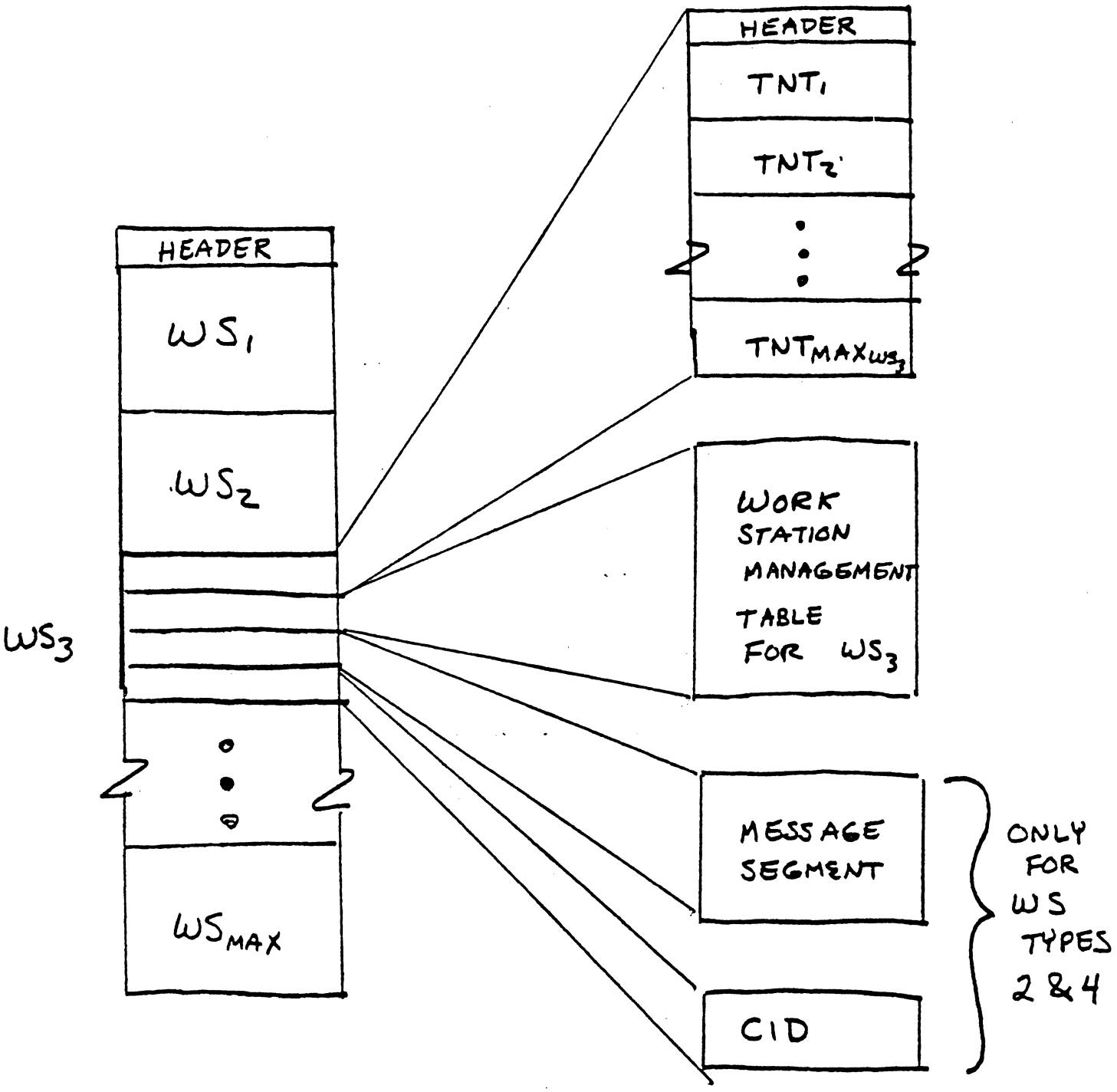
  NORM ≤ CURRENT < MAX
  AND # MESSAGES QUEUED
  ON WS SEMAPHORE > THRESHOLD

- TERMINATION RULES:

  MIN < CURRENT < NORM
  AND ALL OTHER PROCESSES WAITING

  NORM ≤ CURRENT ≤ MAX
  AND ANY PROCESSES WAITING

INTERNAL STRUCTURES



1-31-80
BGT

- SOME HISTORY
- ITP OBJECTIVES
- ITP ARCHITECTURE
  - DSA POINT OF VIEW
  - NSA/GCOS 8 POINT OF VIEW
- WHERE ARE WE GOING ?

SOME HISTORY

- TPE
  - + integrated with OS
  - − program # limitation
  - − no sharing of database management
  - − FMS concurrency/recovery limitations
  - − no conversation facility
  - − no program linking by name
- 1Q 1974 _ TDS
  - + good concurrency/recovery
  - + conversation facility
  - + dynamic program linking by name
  - + shared database manager
  - + can handle thousands of terminals
  - − everything ≠ from batch
  - − only BCD / COBOL 68
  - − hard to use (differed I/O's)
  - − limited to one processor
  - − limited to 192 K of memory
  - − difficult TX/batch concurrency
- 2Q 1977 _ DMS IV TP
  - + same as TDS
  - + ASCII / COBOL 74
  - + SAME IDS-II as batch
  - + easier to use than TDS (no differed I/O's)
  - − no BCD / COBOL 68
  - − limited to one processor
  - − limited to 192 K of memory

- 78 TPE-II
  - \+ integrated with OS
  - \+ fast loading/unloading of programs
  - \+ reuse of processes
  - − FMS concurrency/recovery limitations
  - − no sharing of database management
  - − no conversation capability

- 1Q77 first signature of DSA protocols
- 2Q77 Top-Down definition of ITP
- 4Q77 Start ITP implementation on ACOS, new from Japan
- 1Q78 4VX/ITP Conceptual Design Review: Go
- 4Q78 Version I2 running on 4V
- 2Q79 Version I3 running on early 4VX
- 4Q79 Version I4 running on 4VX

ITP OBJECTIVES (CDR)

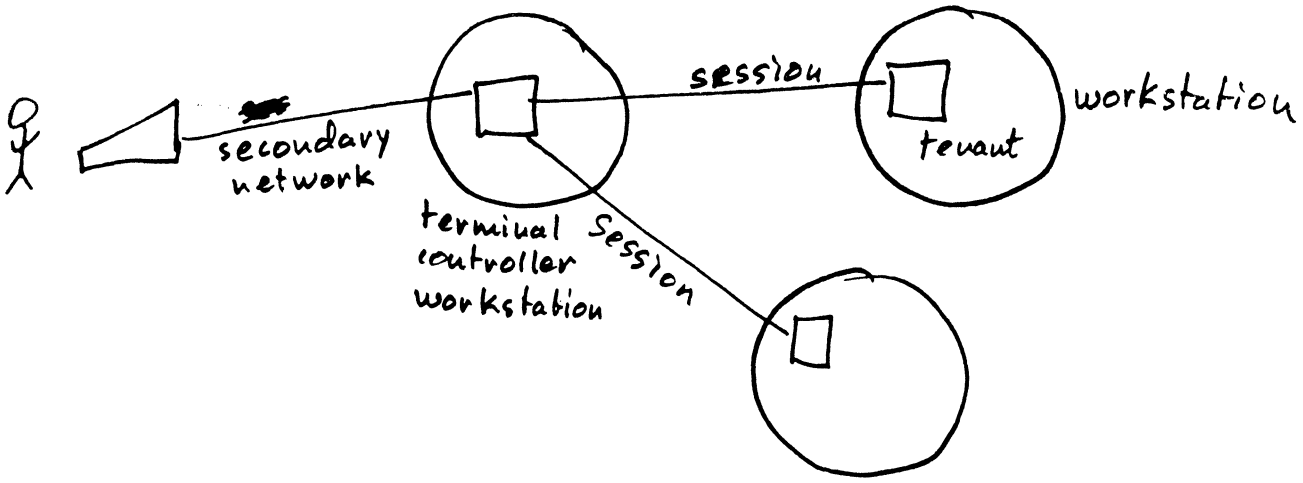
- **ELIMINATE PROBLEMS OF TPE/TPE-II/TDS/DMIV-TP**
  - INTEGRATED TO OS : multiprocessor, no memory limit
  - NEW CONCURRENCY/RECOVERY SYSTEM WIDE
  - SHARED DATABASE MANAGEMENT

- **COMPATIBLE WITH DMIV-TP**
  - TPR's OBJECT COMPATIBLE
  - NO DATABASE CHANGES
  - NO TERMINAL OPERATOR CHANGES

- **BASE OF DISTRIBUTED PROCESSING (DSA)**
  - WORKSTATION ARCHITECTURE
  - SESSION CONTROL LAYER
  - GATEWAY TO EXISTING NETWORK

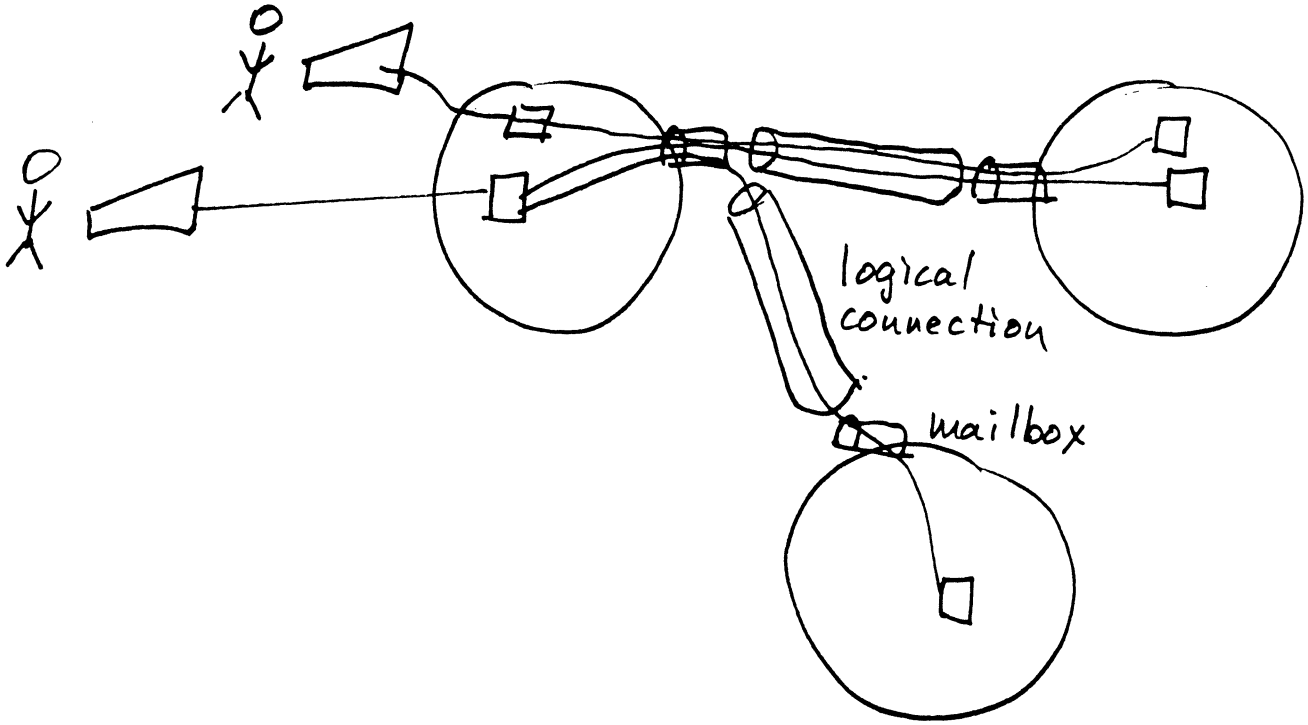
- **STEPPING STONE FOR GCOS 8 EVOLUTION**
  - FULL USAGE OF NSA CAPABILITY:
    DOMAIN ISOLATION, VIRTUAL MEMORY, SLAVE MODE
  - ALL ITP FUNCTIONS (EXCEPT EXECUTIVE)
    MUST BE USABLE IN ALL DIMENSIONS

ITP ARCHITECTURE - DSA POINT OF VIEW



- The tenant is the "agent" of the user
- All these agents communicate via "sessions"
- tenants are located in "workstations"
- workstations are indivisible, but relocatable

ITP ARCHITECTORE _ DSA POINT OF VIEW

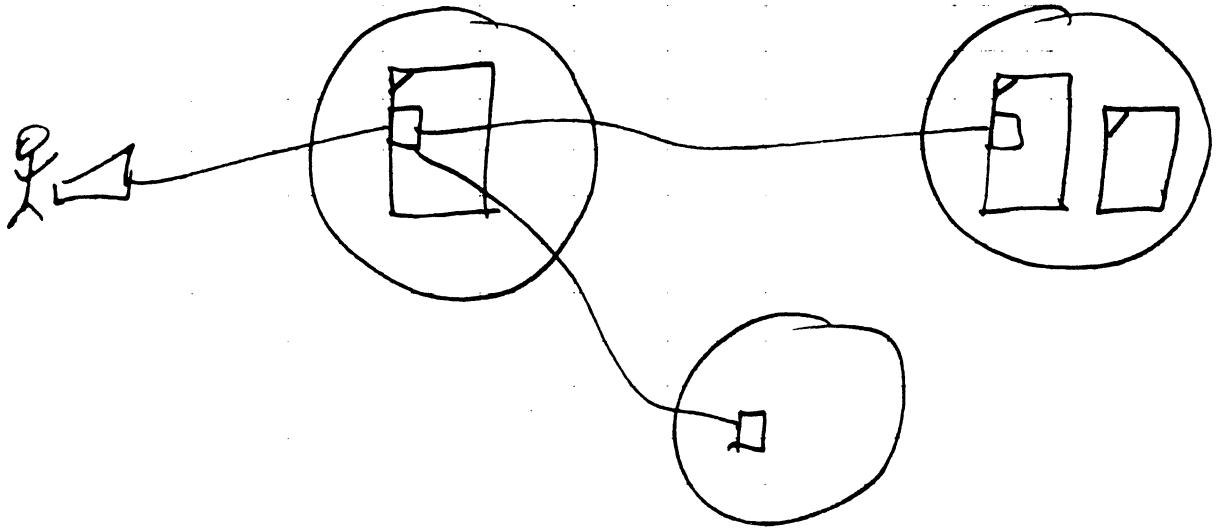


- workstations are addressed by

  < node code, mailbox name, mailbox extension

- many sessions using identical protocol options may share a "logical connection"

ITP ARCHITECTURE - DSA POINT OF VIEW



- To do useful work a tenant must be "mapped" to a process
- many more tenants than processes may exist
- unmapping may be meaningless for some types of workstations.

ITP ARCHITETURE – DSA POINT OF VIEW

- There are many types of workstation
- TO each workstation corresponds:
  - a description of the WS entities via Workstation Control Language
  - a main procedure associated with each workstation process:
    "The command executive"
- Workstation types in 4VX:
  - Gateway
  - ITP
  - BIBO
  - Administrative

LTP ARCHITECTURE - NSA/GCOS POINT OF VIEW

- EACH TPR RUNS AS A SUBROUTINE UNDER LTS OWN PROCESS
- MANY IDENTICAL PROCESSES AT A WORKSTATION
- PROCESSES ARE UNMAPPED AT CONVERSATIONS AND END OF TRANSACTIONS, THEY ARE REUSE FOR OTHER TRANSACTIONS
- UP TO 477 PROCESSES
- TRANSACTION PROCESSING IS ONE OR SEVERAL WORKSTATIONS
- POOL OF TPR'S KEPT IN REAL MEMORY
- SHARED DATABASE MANAGER
- NEW RECOVERY/RESTART/CONCURRENCY CONTROL

ITP ARCHITECTURE - NSA/GCOS 8 POINT OF VIEW

- WORKSTATION = SET OF IDENTICAL PROCESSES:
- WORKSTATION MANAGEMENT ADDED TO OS
  PROCESS INITIATION/TERMINATION
- MULTI PROCESS WORKSTATION HAVE THEIR OWN WSQ
- 2 HARDCORE MODULES, 23 SLAVE DOMAINS
- ITP ONLY USER OF DEMAND PAGING AND MULTISEGMENT ENVIRONMENT
-
- TEMPORARY DYNAMIC LINKER/LOADER FOR SHARED SOFTWARE PENDING DEFINITION OF SHARED RUN TIME ENVIRONMENT.
- ALL ITP SOFTWARE IS OPTIONAL

WHERE ARE WE GOING?

2Q80 SR 1000 GENERAL RELEASE

4Q80 SR 1100 CONTROLLED RELEASE

+ regressions items

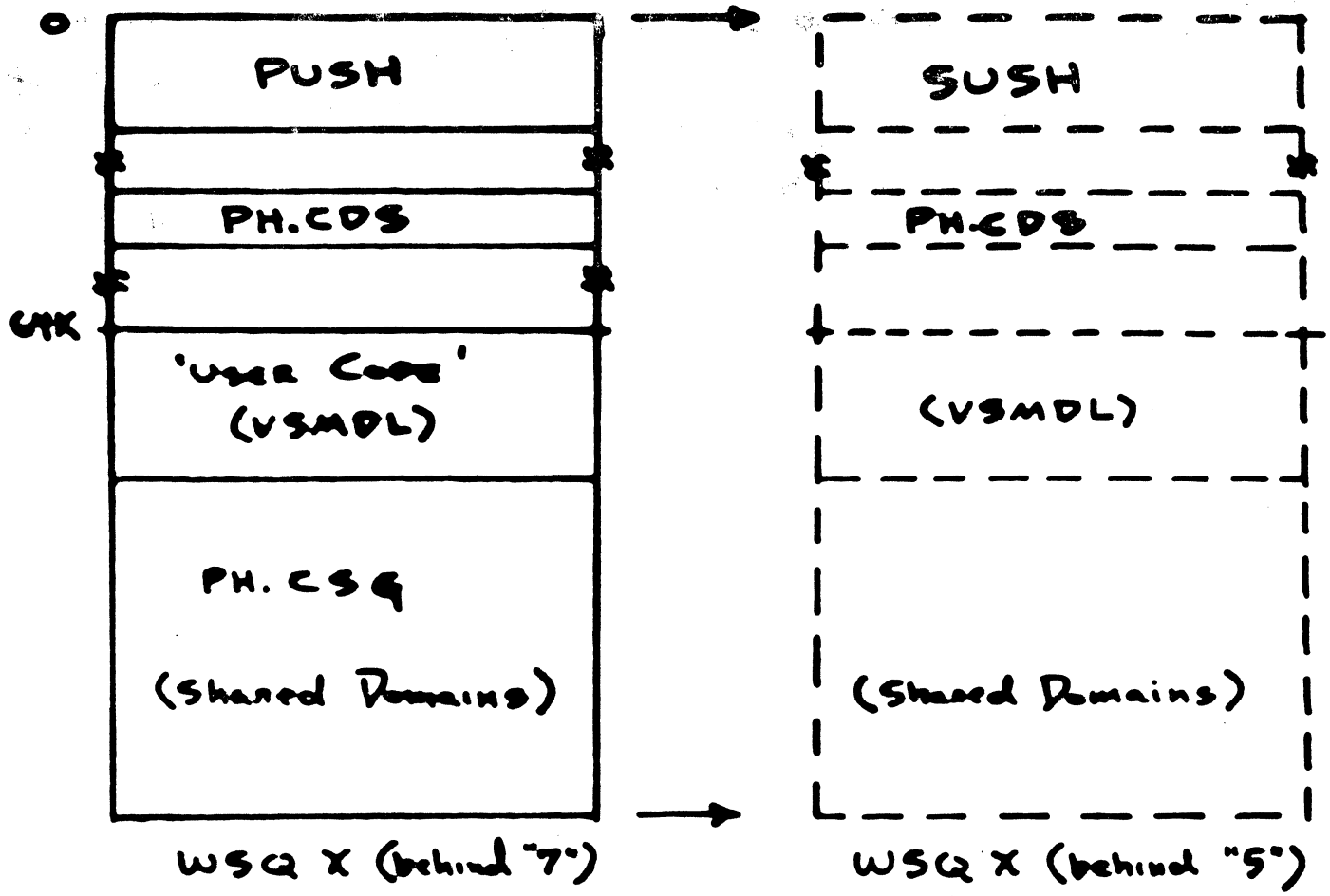
2Q81 SR 2000 CONTROLLED RELEASE

+ Forms support
+ UNCP connection
+ DUAL DATA ENVIRONMENT

2Q82 SR 3000 CONTROLLED RELEASE

+ MESSAGE QUEUING
+ TX SCHEDULING
+ NATIVE TPR's

Loading Shared Software



- VSMDL a native mode process
  - paging of shared software
  - shared structures, i.e., PATS
  - commonality of process/shared structure

- Major steps in loading shared software
  - loads WSR 5 with contents of WSR 7
  - loads each shared domain into PH.CSQ and climbs to initialization entry point of domain
  - name WSQ using name from WSQDF

Domain Initialization

What's required by a domain at run time ?

1- For it to be climbed to, it must have provided entry descriptor(s) to the system enabling access to it.

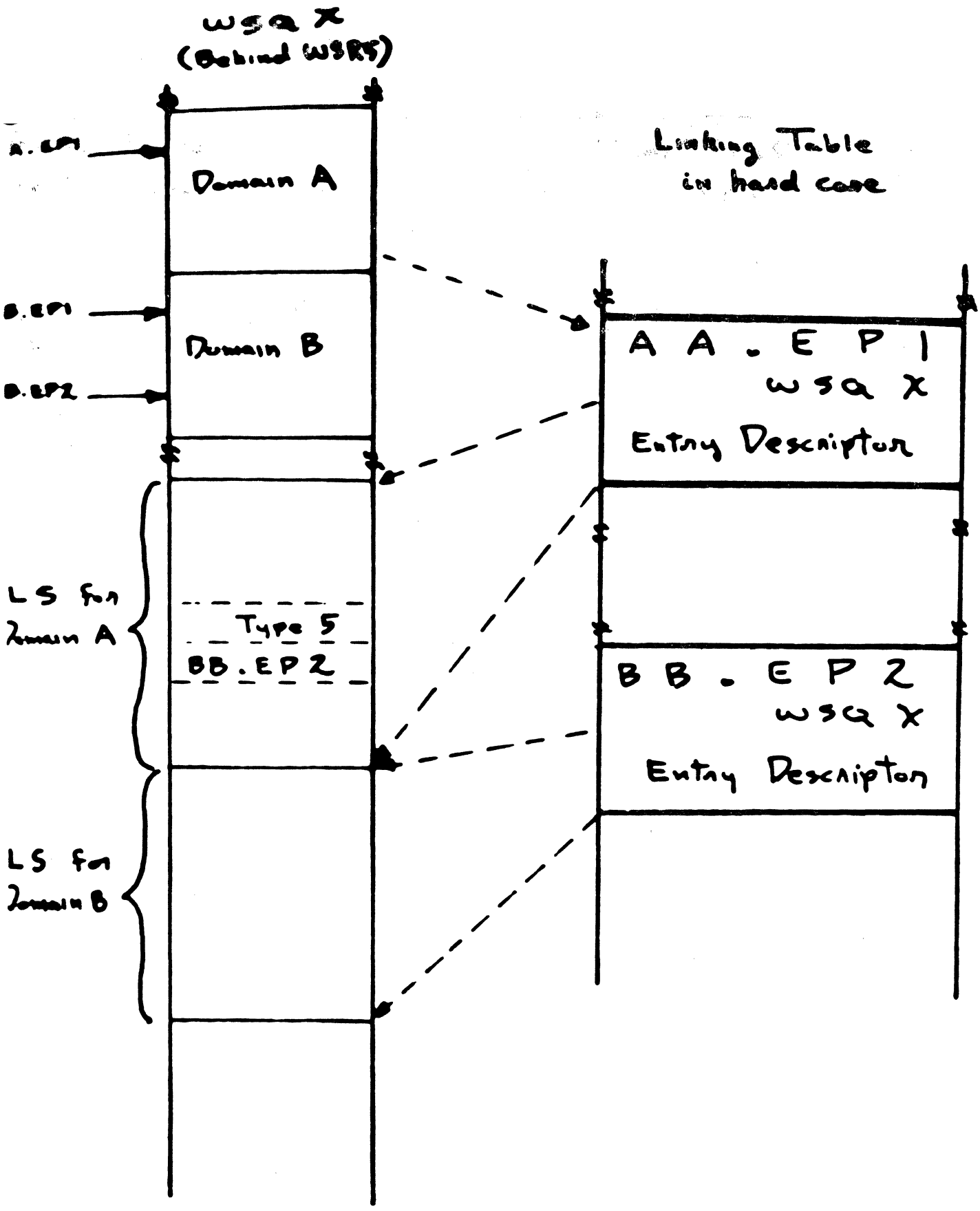
2- It must be operating with a linkage segment that:
   a- provides access to necessary structures
   b- contains entry descriptors that enable climbs to other domains that it must call

- The solution to requirements 1 and 2b involves the utilization of dynamic linking descriptors to define domain entry points (..EPX. macro)

- The remainder of the solution to requirements 1 and 2 is provided by utilizing the .EPX.M mac (or .PLS.M macro). Via these macros, the domain passes to MLINK:

   a- A list of 'type 5' descriptors identifyin its entry points (in BCI); and for each an associated entry descriptor.

   b- A copy of the linkage segment that it wishes to operate under when it is called.


wsq X
(Behind WSRS)
A.EP1
Domain A
B.EP1
Domain B
B.EP2
LS for
Domain A
Type 5
BB.EP2
LS for
Domain B
Linking Table
in hard core
A A . E P 1
wsq X
Entry Descriptor
B B . E P 2
wsq X
Entry Descripton

Steps in dynamic linking

A climb is attempted using type 5 descriptor, causing dynamic linking fault which passes control to MLINK in kernel core.

) MLINK retrieves faulting entry descriptor (type 5) and locates corresponding entry in Linking Table.

) Climb re executed using correct entry descriptor.

Misc:

- MLINK provides facility at system startup for placing dynamic linking descriptors to shared software into system linkage segment.
- Option on ..EPX. macro permits specification of whether faulting entry descriptor should be replaced.
- The $ SHRLIB card in process JCL enables linking to a specific set of shared software (in-house test environment)
- In the case of process local linkage segment- the linkage segment is built at the time of the dynamic linking fault.

Workstation Management

General Purpose:
To provide integration between the operating system process management functions and shared domain management of work stations.

Process Start

PALC
P&P4
P&P8

Process Executing

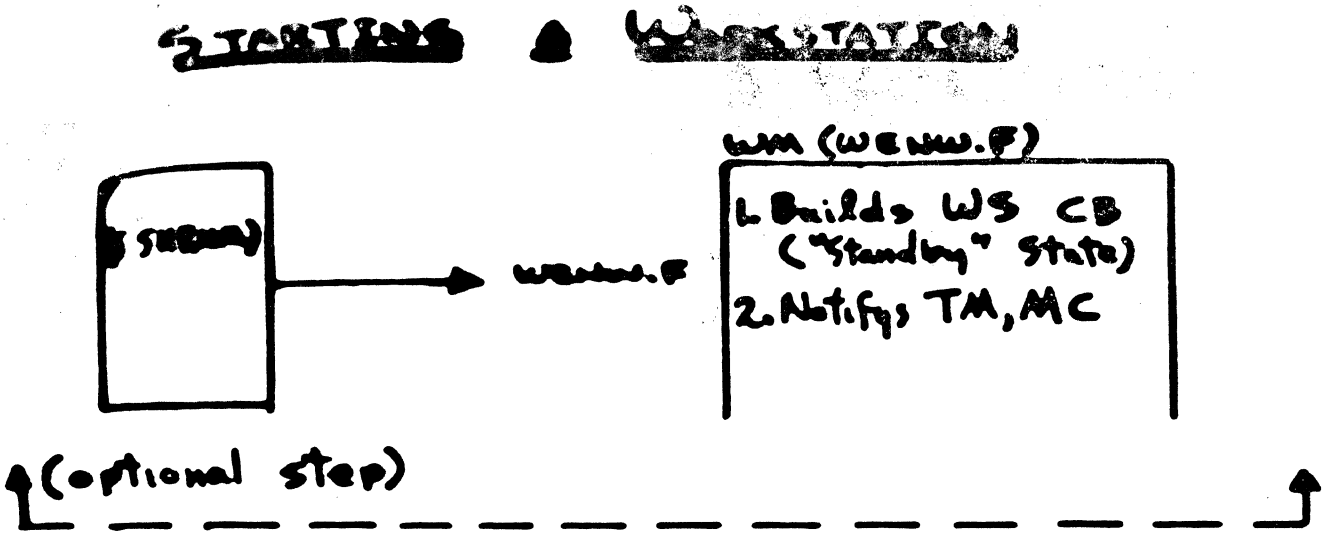
Process Termination

BRT2
BRT6

WS definition on data base

Active WS with one or more processes operating under it

WS exists only on data base definition



(optional step)

PALC

If $ WORKST → WACT.F

WM (WACT.F)

1. Determine if WS pnum enabled
2. "Gate" start function until WS is in "Active" state
3. Rss WS CB with state set to "Pre" or "Semi" Active

Returns
1. "Instigator"
2. "Continue"
3. Wait/Try again
4. Abort ($ WORKST error)

P&P8

If WORKST → WSTW.F

WM (WSTW.F)

If state = "Pre-Active" → Call WC ...
If state = "Semi-Active" →
  Act/Sta. WS if type 3/4
  Notify GLBDM re Buffer Pools
If state = Active →
  Notify IM
  Create/Map Tem. if WS type = 1/3

Returns
Continue
Bad Status → "LOG/WSN" error

TERMINATION of WORKSTATION

1. A WORKSTATION MAY BE "TERMINATED" OR "ABORTED" via calls to WTMW.F OR WABW.F .

   WTMW.F → WS state set to "quiescent"
   MC notified

   WABW.F → WS state set to "aborted"
   All active processes on WS aborted

2. Process Termination

   IF WS state = "Active" → # Active proc. decremented and snumb removed

   IF WS state = "Quiescent" or "Aborted" and last process terminates →
   WORK STATION reverts to defined state
   TM, MC, IM notified
   Buffer pools and all other structure released