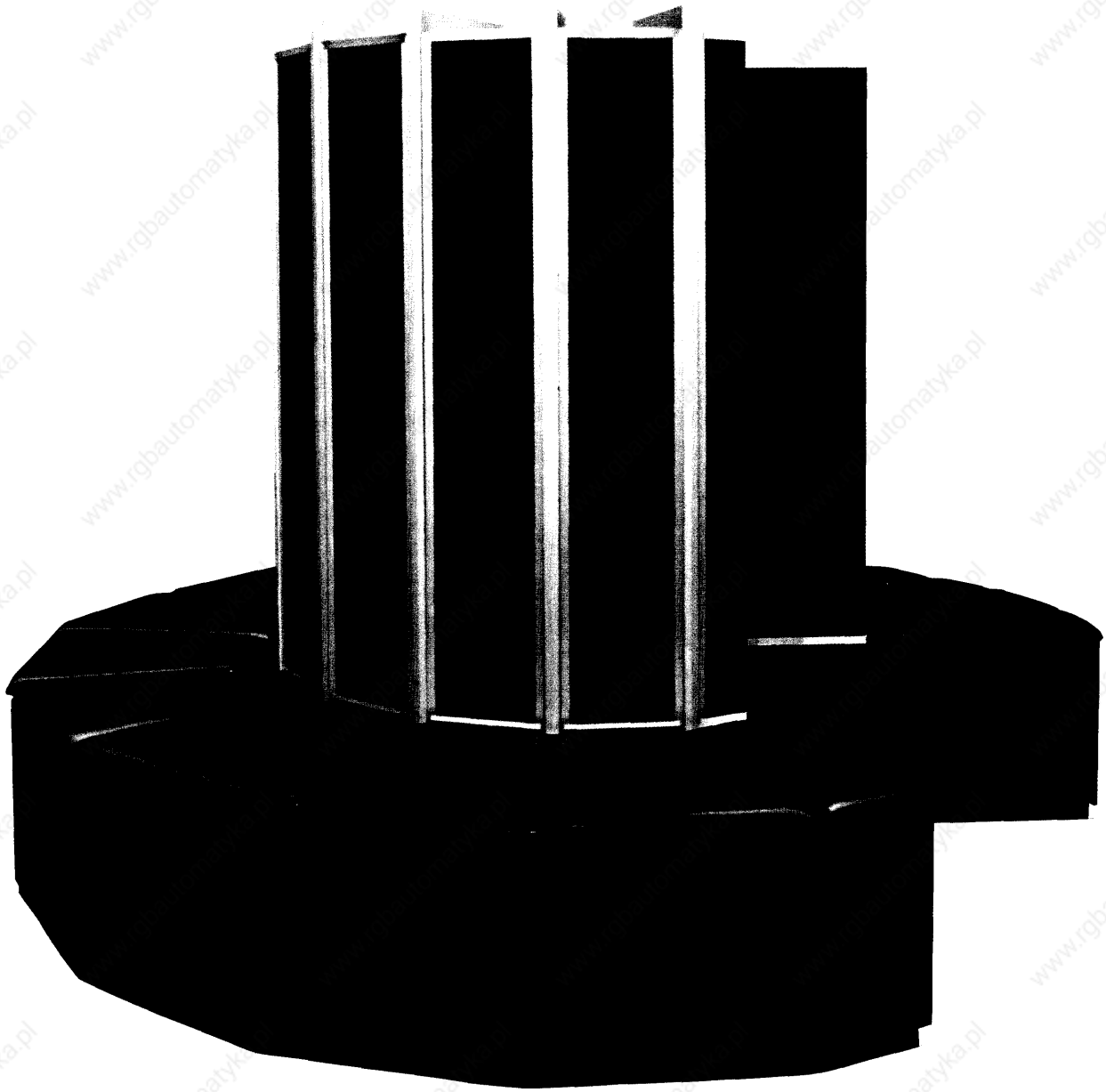
CRAY RESEARCH, INC.

CRAY-1® COMPUTER SYSTEM

OPERATING SYSTEM (COS) WORKBOOK

T-0103



FOR TRAINING PURPOSES ONLY

CRAY-1®

COMPUTER SYSTEM

OPERATING SYSTEM (COS)
WORKBOOK

T-0103

Copyright© 1979 by CRAY RESEARCH, INC. This manual or parts thereof may not be reproduced in any form without permission of CRAY RESEARCH, INC.

RECORD OF REVISION

CRAY RESEARCH, INC.

PUBLICATION NUMBER T-0103

Each time this manual is revised and reprinted, all changes issued against the previous version in the form of change packets are incorporated into the new version and the new version is assigned an alphabetic level. Between reprints, changes may be issued against the current version in the form of change packets. Each change packet is assigned a numeric designator, starting with 01 for the first change packet of each revision level.

Every page changed by a reprint or by a change packet has the revision level and change packet number in the lower righthand corner. Changes to part of a page are noted by a change bar along the margin of the page. A change bar in the margin opposite the page number indicates that the entire page is new; a dot in the same place indicates that information has been moved from one page to another, but has not otherwise changed.

Requests for copies of Cray Research, Inc. publications and comments about these publications should be directed to:

CRAY RESEARCH, INC.,
1440 Northland Drive,
Mendota Heights, Minnesota 55120

Revision	Description
	December, 1979 - Original printing.
A	June, 1980 - obsoletes other revision.
B	January, 1981 - obsoletes other revisions.
C	September, 1981 - obsoletes other revisions. Supports 1.10 release.

OVERVIEW OF CRAY-1 OPERATING SYSTEM (COS)

CRAY-1 OPERATING SYSTEM (COS)

- MULTIPROGRAMMING OF USER APPLICATIONS
- SCHEDULING OF APPLICATIONS BY PRIORITY (JOB CLASS)
- MANAGES DISK AND TAPE RESOURCES
- MANAGES FRONT-END COMMUNICATIONS
- MANAGES FILE MAINTENANCE
- MANAGES PROGRAM MAINTENANCE
- CAPABLE OF MODIFICATION AT STARTUP

COS HARDWARE ELEMENTS

- MCU

  STARTUP AND RECOVERY OF COS
  FRONT-END COMPUTER SYSTEM

- MASS STORAGE SUBSYSTEM

  UTILITIES
  DATA
  LIBRARIES
  COPY OF COS (OPTIONAL)
  COS OVERLAYS (OPTIONAL)
  COPIES OF CSP (OPTIONAL)

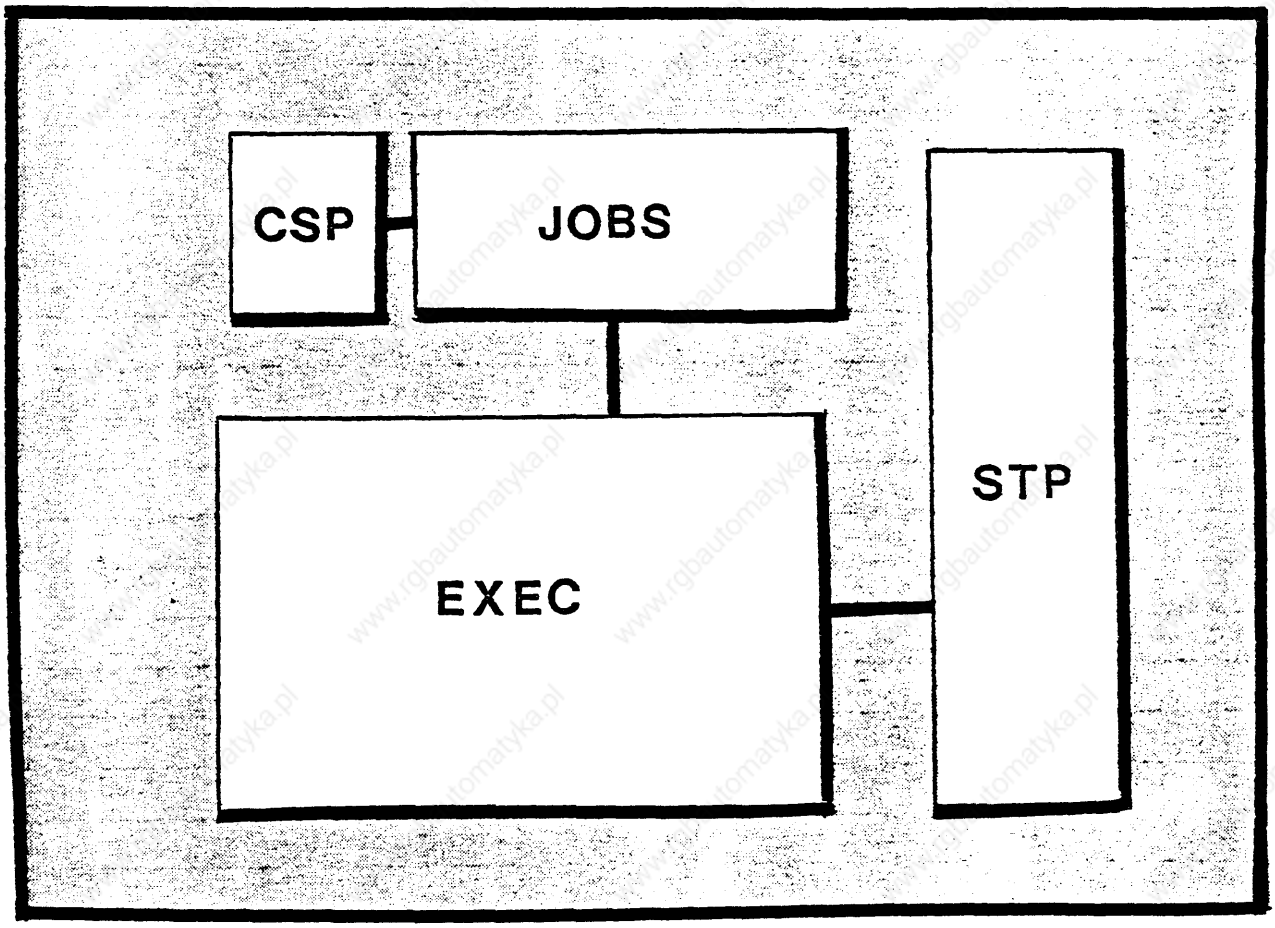
- CRAY-1 COMPUTER

  COS RESIDENT
  USER APPLICATIONS EXECUTION AREAS
  USER STATION BUFFERS

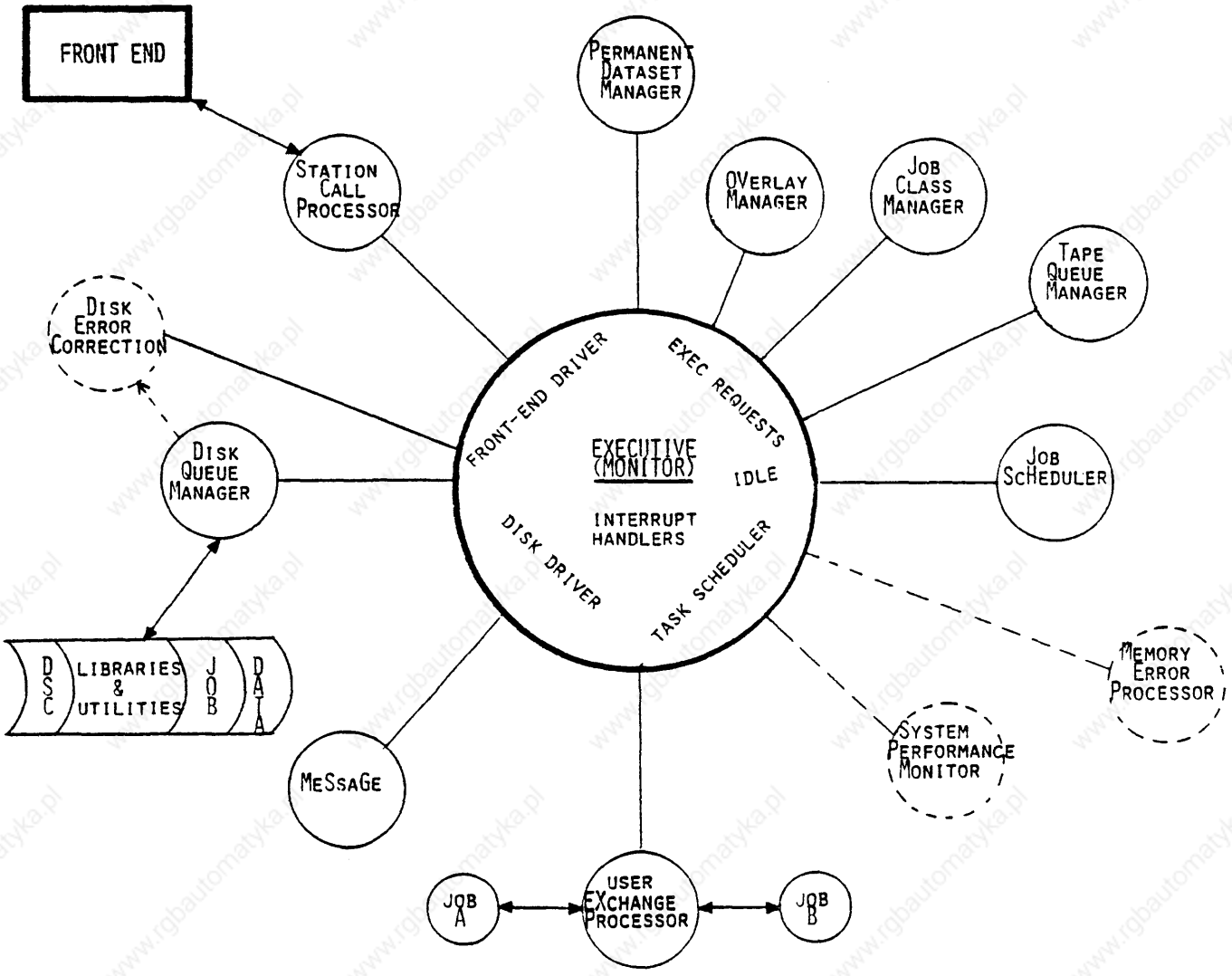
COS SOFTWARE ELEMENTS

- RESIDENT SYSTEM PROGRAMS

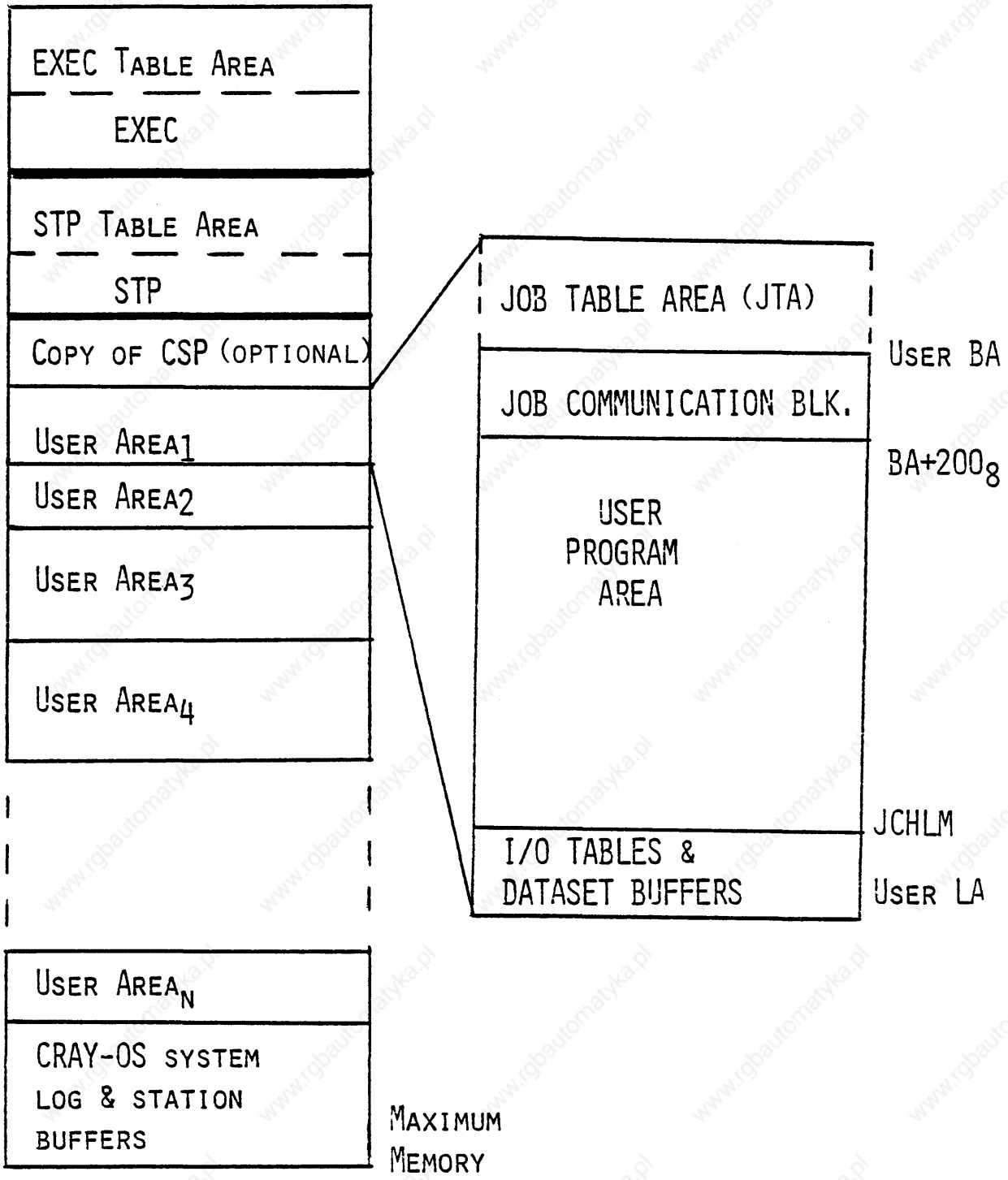
  EXECUTIVE (EXEC)
  SYSTEM TASK PROCESSOR (STP)
  CONTROL STATEMENT PROCESSOR (CSP) - OPTIONAL



- NON-RESIDENT PROGRAMS
  - LIBRARIES
  - SYSTEM UTILITIES
  - USER APPLICATIONS


FRONT END
STATION CALL PROCESSOR
PERMANENT DATASET MANAGER
OVERLAY MANAGER
JOB CLASS MANAGER
TAPE QUEUE MANAGER
DISK ERROR CORRECTION
DISK QUEUE MANAGER
FRONT-END DRIVER
EXEC REQUESTS
EXECUTIVE (MONITOR)
IDLE
INTERRUPT HANDLERS
DISK DRIVER
TASK SCHEDULER
JOB SCHEDULER
DSC
LIBRARIES & UTILITIES
JOB
DATA
MESSAGE
MEMORY ERROR PROCESSOR
SYSTEM PERFORMANCE MONITOR
JOB A
USER EXCHANGE PROCESSOR
JOB B

SYSTEM MEMORY ASSIGNMENTS

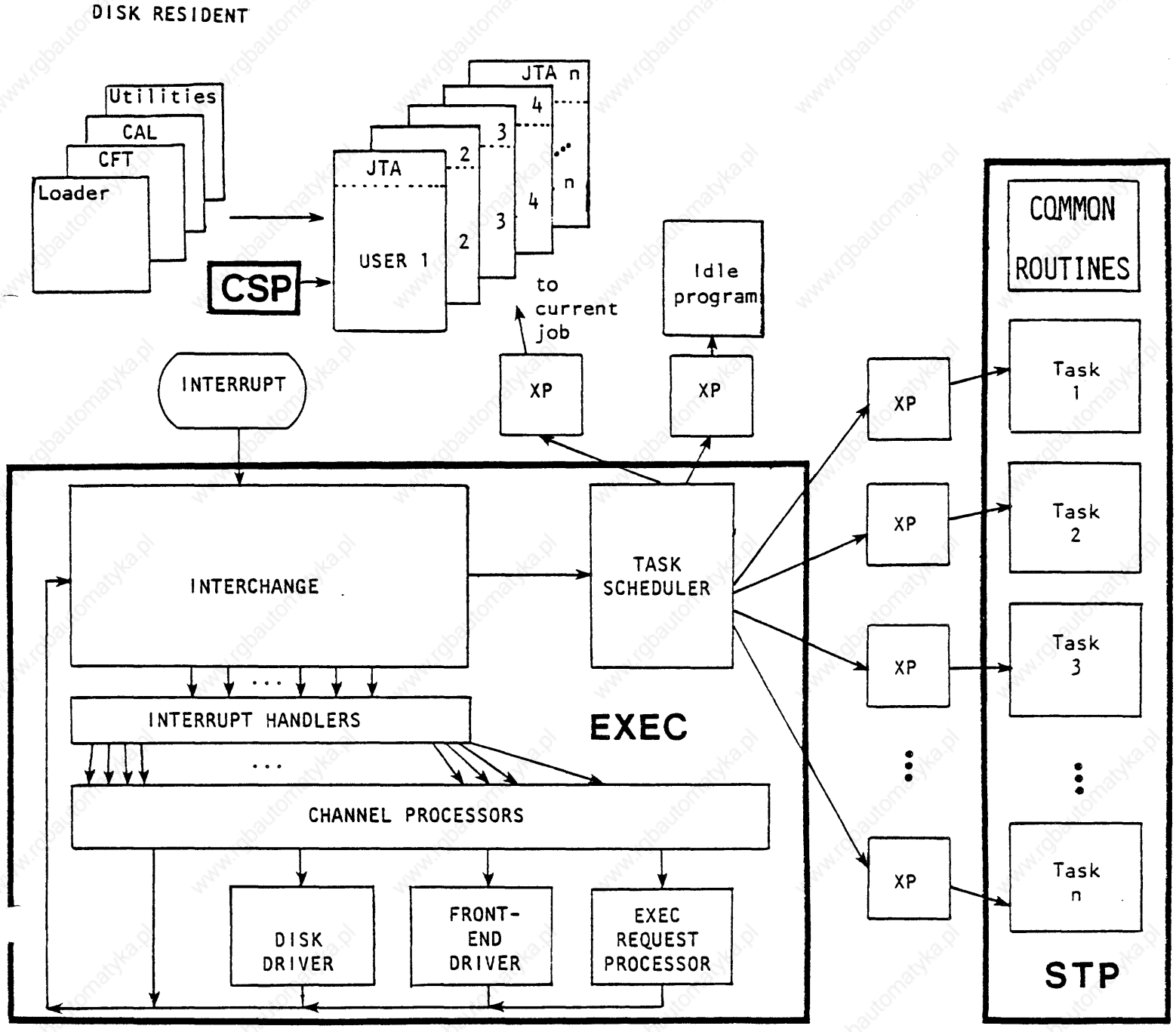


SYSTEM EXECUTIVE

www.rgbautomatyka.pl

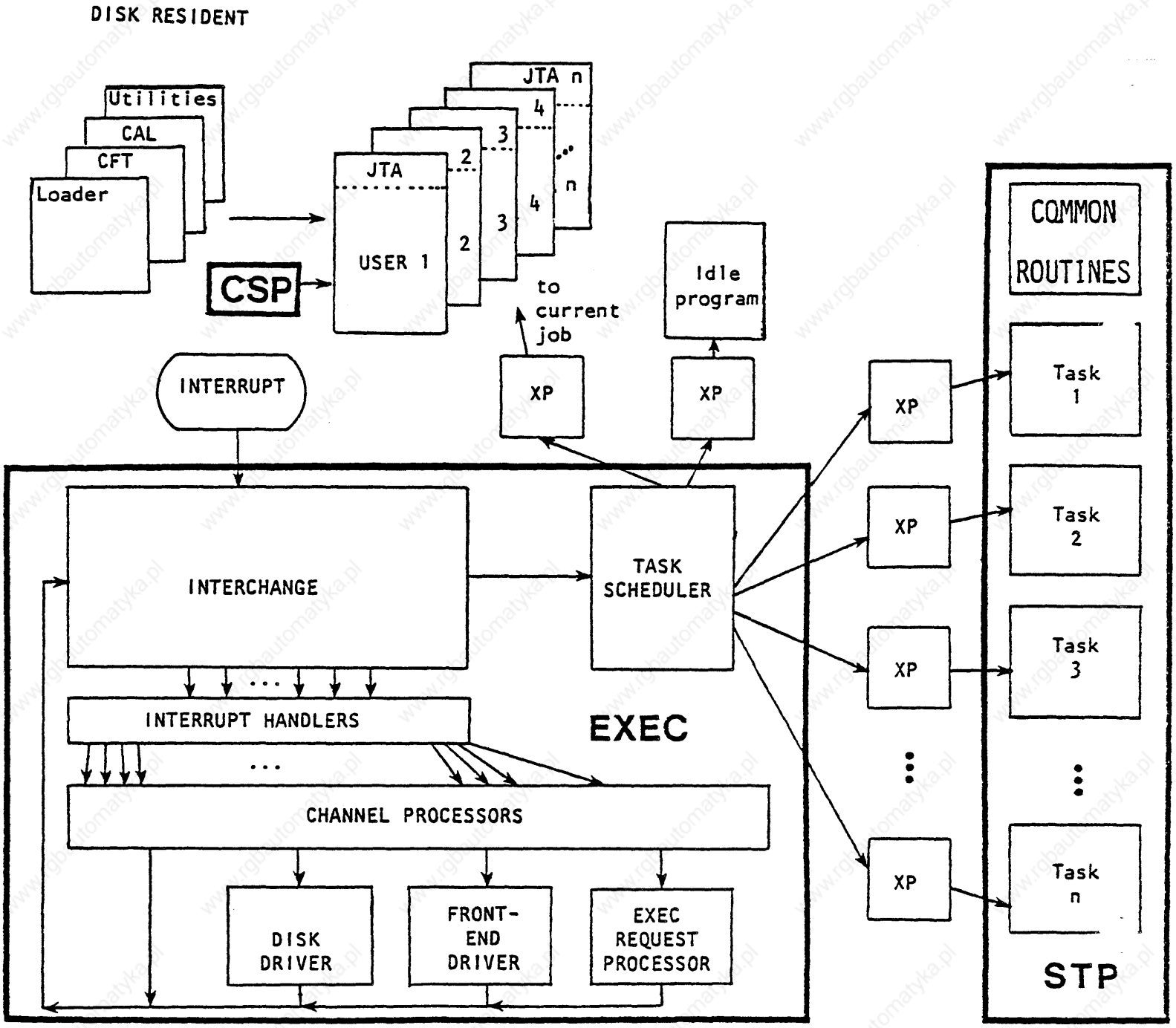
SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE
- CAN ACCESS ALL OF MEMORY (BA=0,LA=I@MEM)



SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE
- CAN ACCESS ALL OF MEMORY

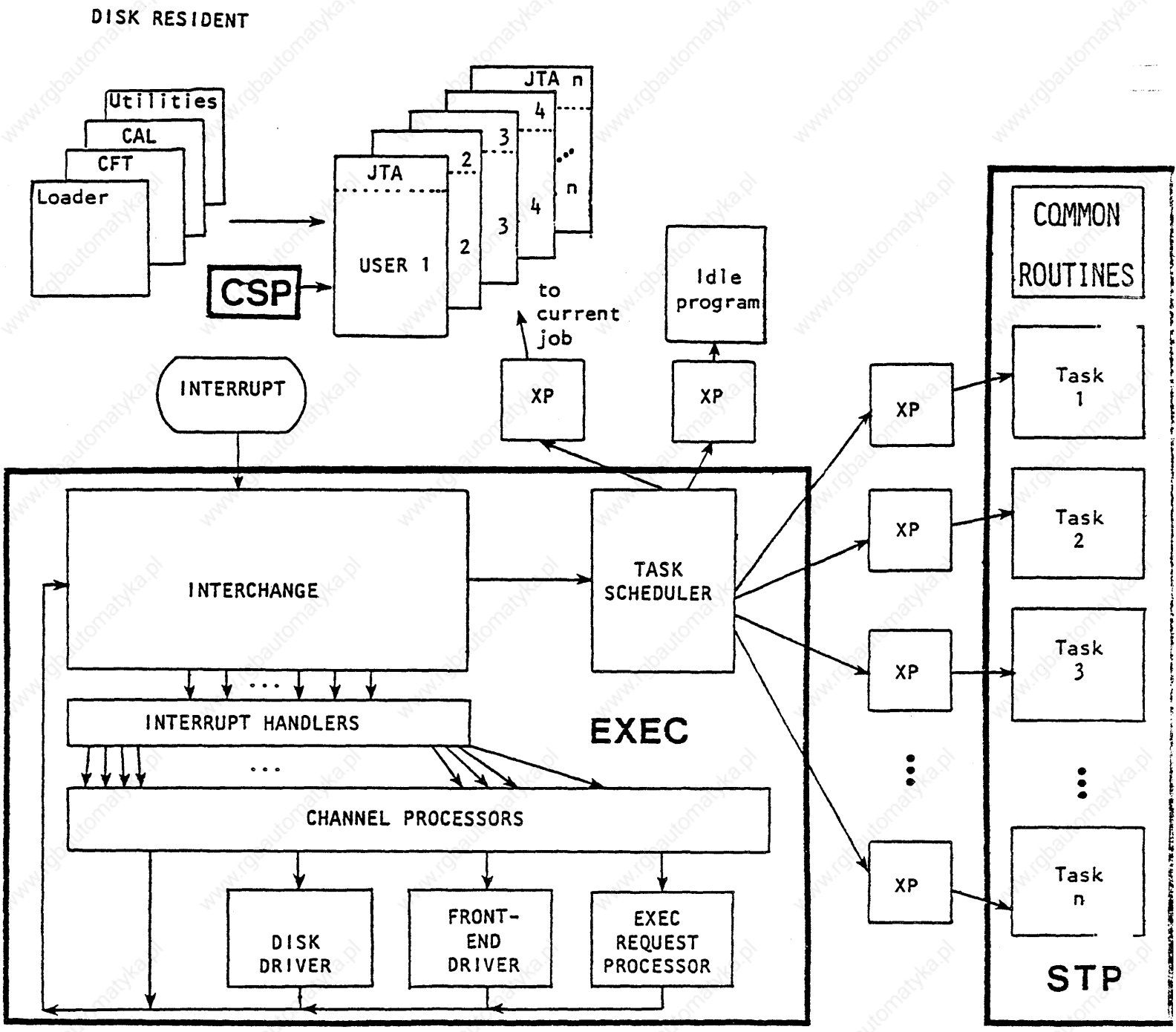


FUNCTIONS OF EXEC

- INTERCHANGE ANALYSIS
- INTERRUPT HANDLERS
- CHANNEL MANAGEMENT
- TASK SCHEDULER
- EXECUTIVE REQUEST PROCESSOR
- DISK DRIVER
- FRONT-END DRIVER

SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE Non Interruptable
- CAN ACCESS ALL OF MEMORY



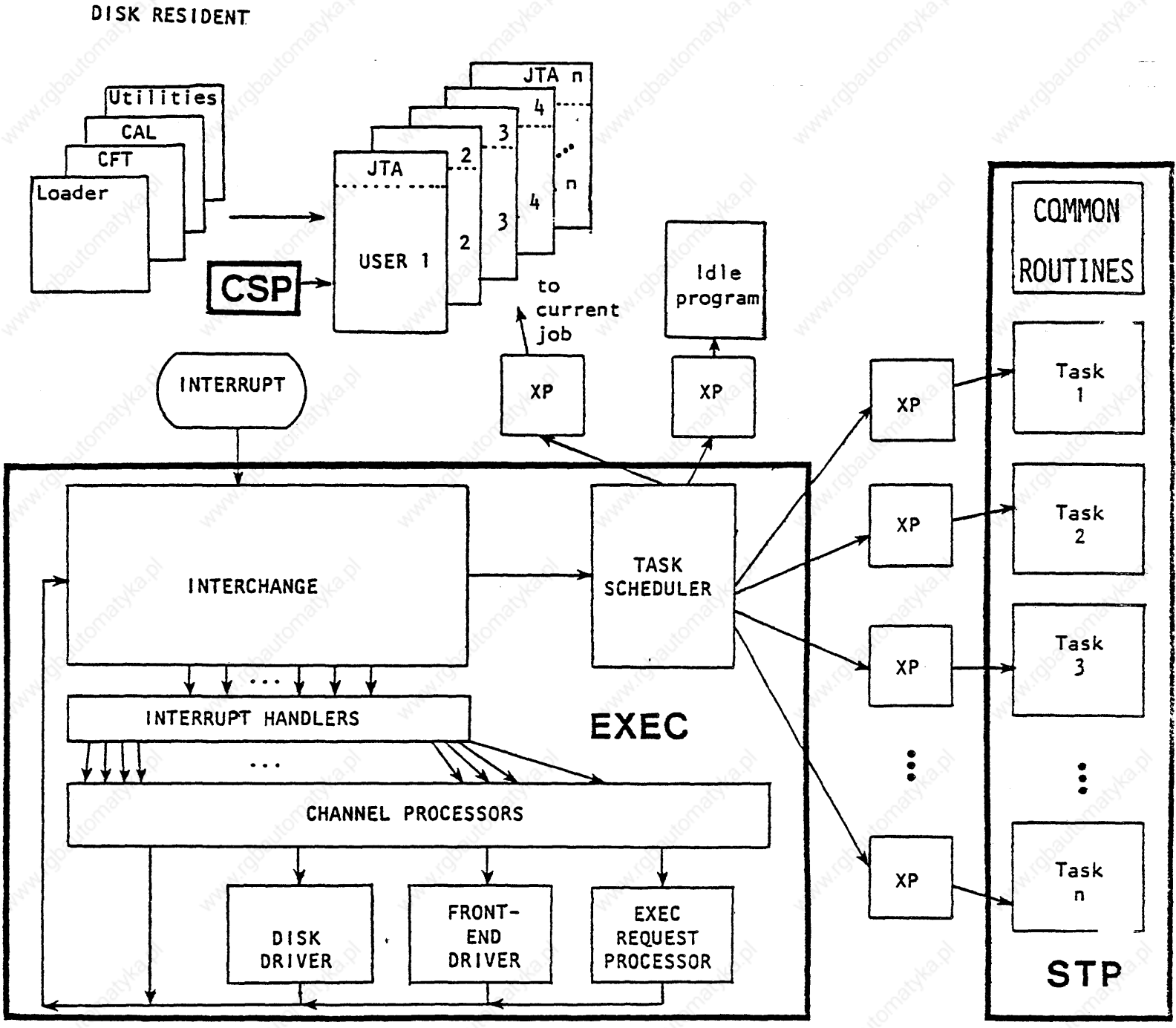
INTERCHANGE

- SAVES CURRENT RTC
- UPDATES ACCRUED CPU TIME
- CALLS I/O HANDLER IF I/O INTERRUPT
- SENSES FOR LOST HARDWARE INTERRUPTS OF PREVIOUS DISK I/O INSTRUCTIONS
- CALLS INTERRUPT HANDLERS FOR:

  CONSOLE / I/O interrupt
  REAL TIME CLOCK
  ERROR EXIT
  NORMAL EXIT
  MEMORY ERROR EXIT
  [illegible] clock
  floating point err
  Prog range err
  Operand range err

SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE
- CAN ACCESS ALL OF MEMORY



INTERRUPT HANDLERS

- EXECUTE ONE OF THE FOLLOWING INTERRUPT HANDLERS:

IOI	I/O INTERRUPT HANDLER
CII	CONSOLE INTERRUPT HANDLER
RTI	REAL-TIME INTERRUPT HANDLER
NEI	NORMAL EXIT INTERRUPT HANDLER
EEI	ERROR EXIT INTERRUPT HANDLER
MEI	MEMORY ERROR INTERRUPT HANDLER
PCI	Programmable Clock Int. Handler.

Not Used

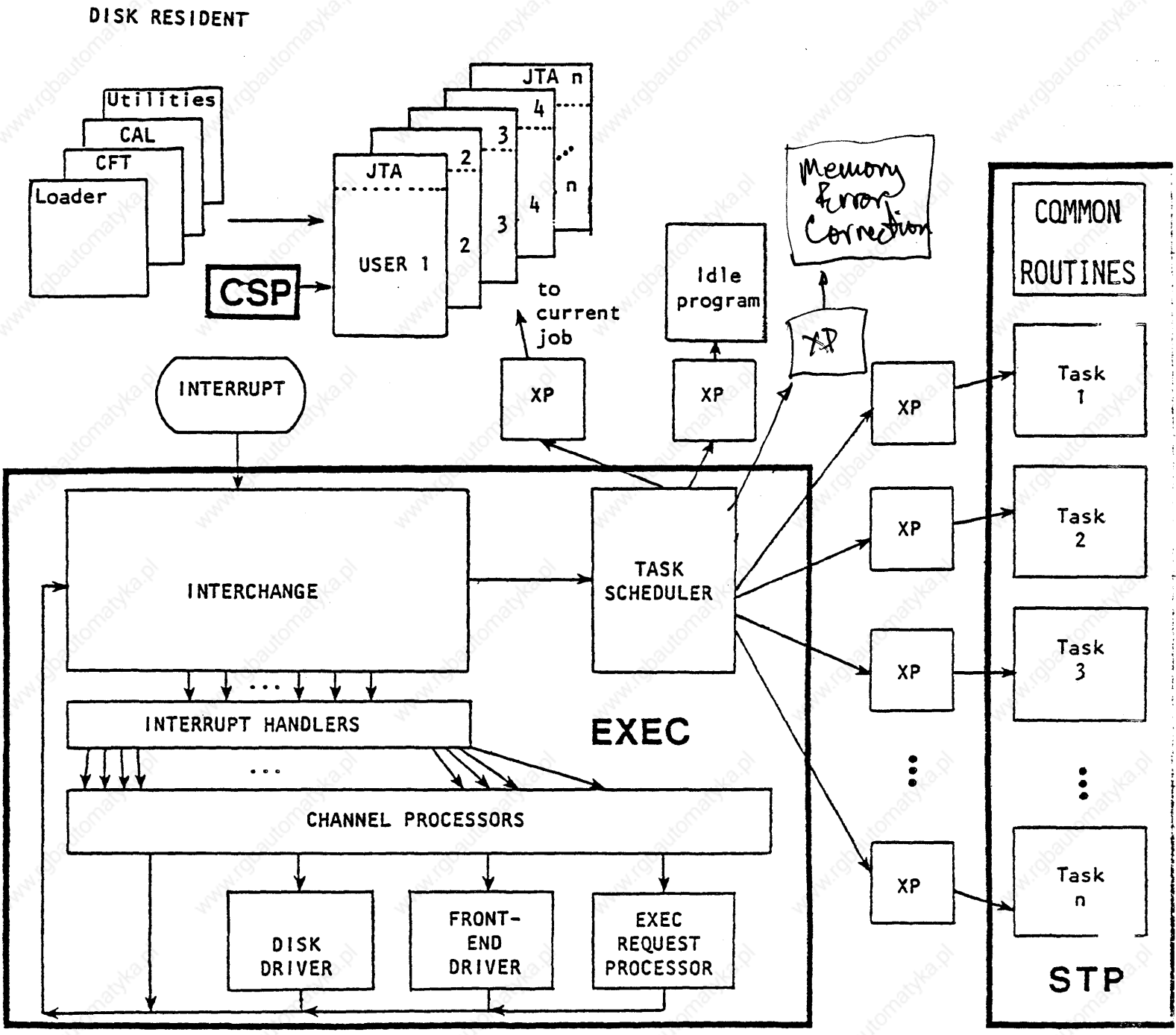
- CLEAR THE INTERRUPT FLAG IN THE EXCHANGE PACKAGE OF THE INTERRUPTED PROGRAM

- DETERMINE THE CHANNEL ON WHICH THE INTERRUPT OCCURED

- BRANCH TO THE APPROPRIATE CHANNEL PROCESSOR

SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE
- CAN ACCESS ALL OF MEMORY



CHANNEL PROCESSORS

- CHANNEL PAIRS ARE ASSIGNED NUMBERS AS FOLLOWS:

0	CONSOLE
1	~~MCU CLOCK~~
2	12 I/O CHANNELS
4	
6	
.	
.	
.	
24	
26	NORMAL EXIT PSEUDO CHANNEL
28	ERROR EXIT PSEUDO CHANNEL
30	PROGRAMMABLE CLOCK PSEUDO CHANNEL
32	MEMORY ERROR PSEUDO CHANNEL

- CHANNEL PROCESSOR TABLE (CHT) HAS ENTRIES FOR BOTH THE INPUT AND OUTPUT SIDES OF EACH CHANNEL

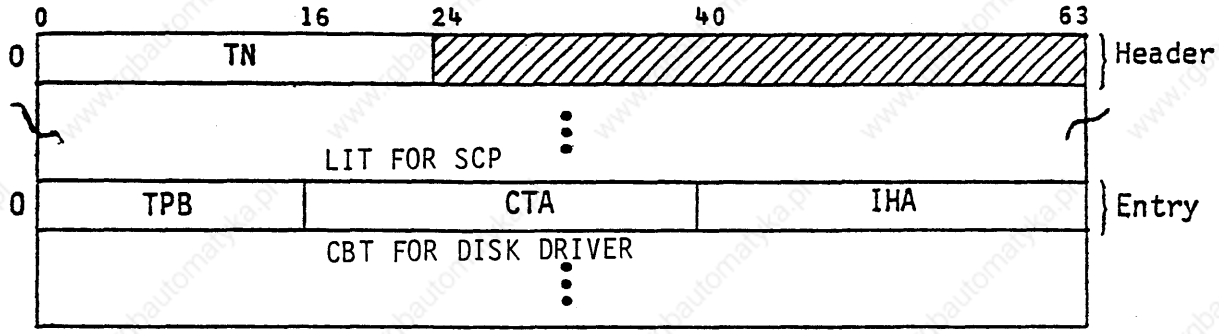
- CHT POINTS TO THE INTERRUPT PROCESSOR

- CHT POINTS TO A PARAMETER AREA FOR EACH CHANNEL PROCESSOR

- CHT POINTS TO CONTROL TABLE ADDRESS

CHANNEL TABLE - CHT

The Channel Table resides in EXEC memory and contains information for use by the interrupt handlers. There is one entry for each channel, physical or pseudo.



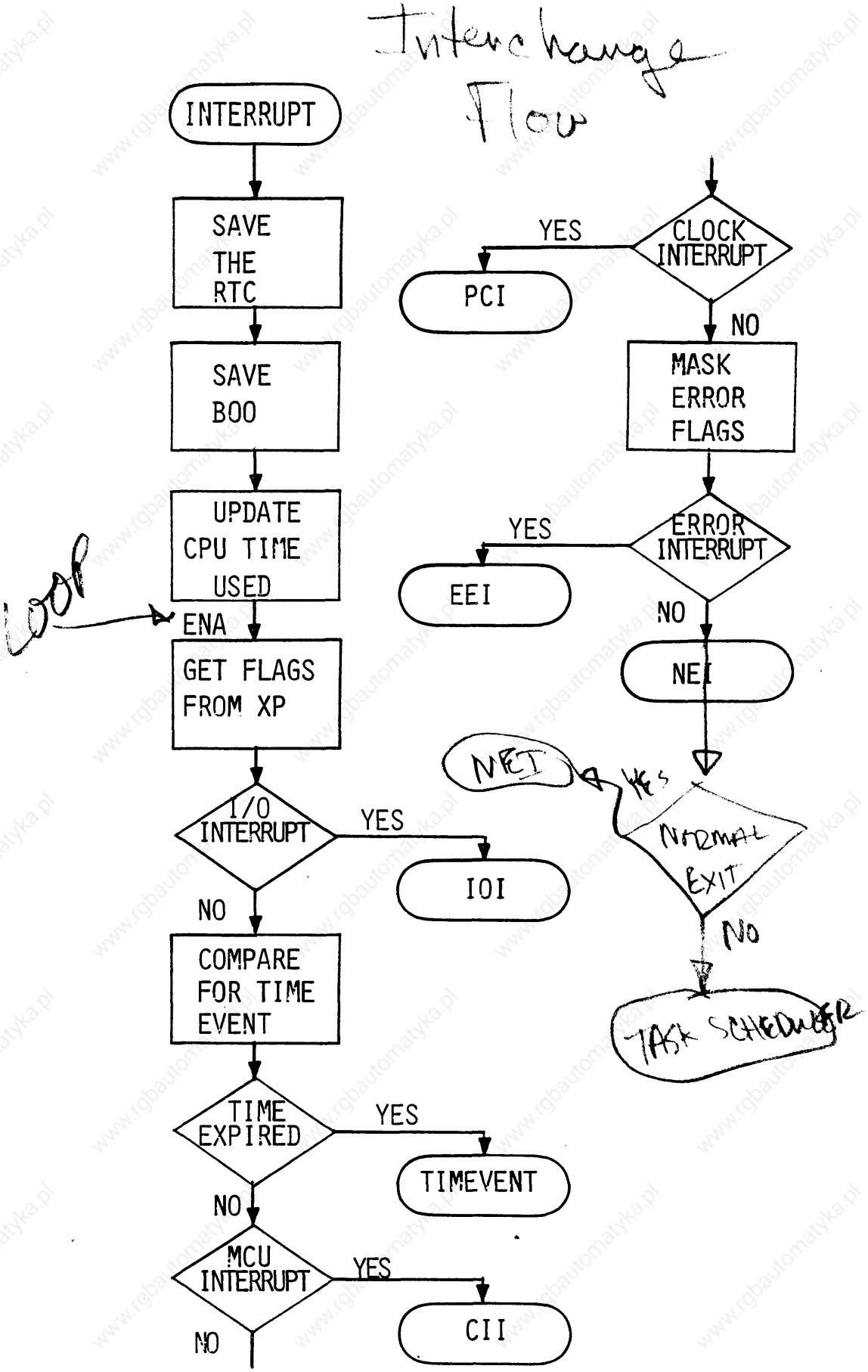
HEADER

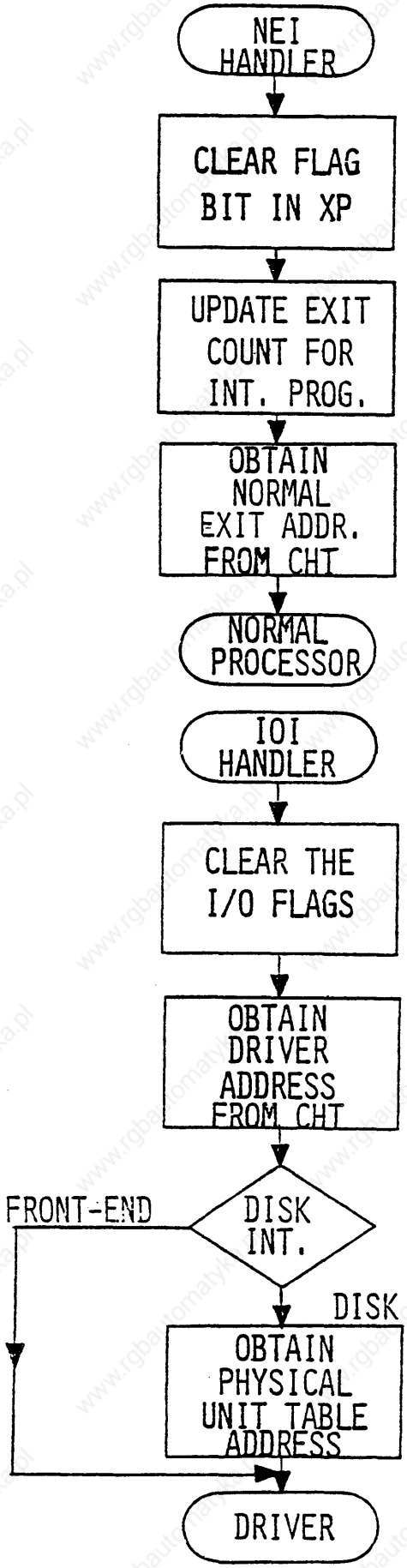
Field	Word	Bits	Description
CHTN	0	0-23	Table name; "CHT" in ASCII

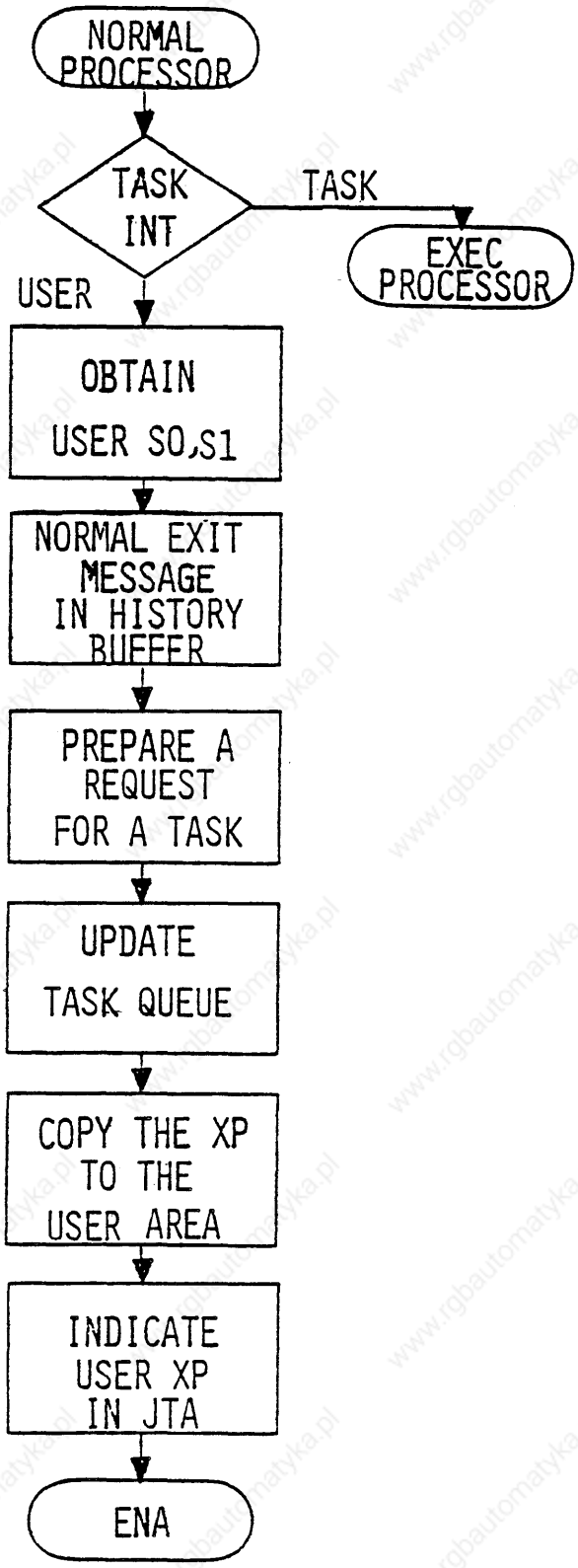
ENTRY

Field	Word	Bits	Description
CHTPB	0	0-15	Address of task parameter word
CHCTA	0	16-39	Control table address
CHIHA	0	40-63	Interrupt handler address

CHANNEL PROCESSOR TABLE


Interchange Flow
INTERRUPT
SAVE THE RTC
SAVE BOO
UPDATE CPU TIME USED
Loop
ENA
GET FLAGS FROM XP
I/O INTERRUPT
YES
IOI
NO
COMPARE FOR TIME EVENT
TIME EXPIRED
YES
TIMEVENT
NO
MCU INTERRUPT
YES
CII
NO
CLOCK INTERRUPT
YES
PCI
NO
MASK ERROR FLAGS
ERROR INTERRUPT
YES
EEI
NO
NEI
NEI
Yes
Normal EXIT
No
TASK SCHEDULER


NEI HANDLER
CLEAR FLAG BIT IN XP
UPDATE EXIT COUNT FOR INT. PROG.
OBTAIN NORMAL EXIT ADDR. FROM CHT
NORMAL PROCESSOR
IOI HANDLER
CLEAR THE I/O FLAGS
OBTAIN DRIVER ADDRESS FROM CHT
FRONT-END
DISK INT.
DISK
OBTAIN PHYSICAL UNIT TABLE ADDRESS
DRIVER


NORMAL PROCESSOR
TASK INT
TASK
EXEC PROCESSOR
USER
OBTAIN USER S0,S1
NORMAL EXIT MESSAGE IN HISTORY BUFFER
PREPARE A REQUEST FOR A TASK
UPDATE TASK QUEUE
COPY THE XP TO THE USER AREA
INDICATE USER XP IN JTA
ENA

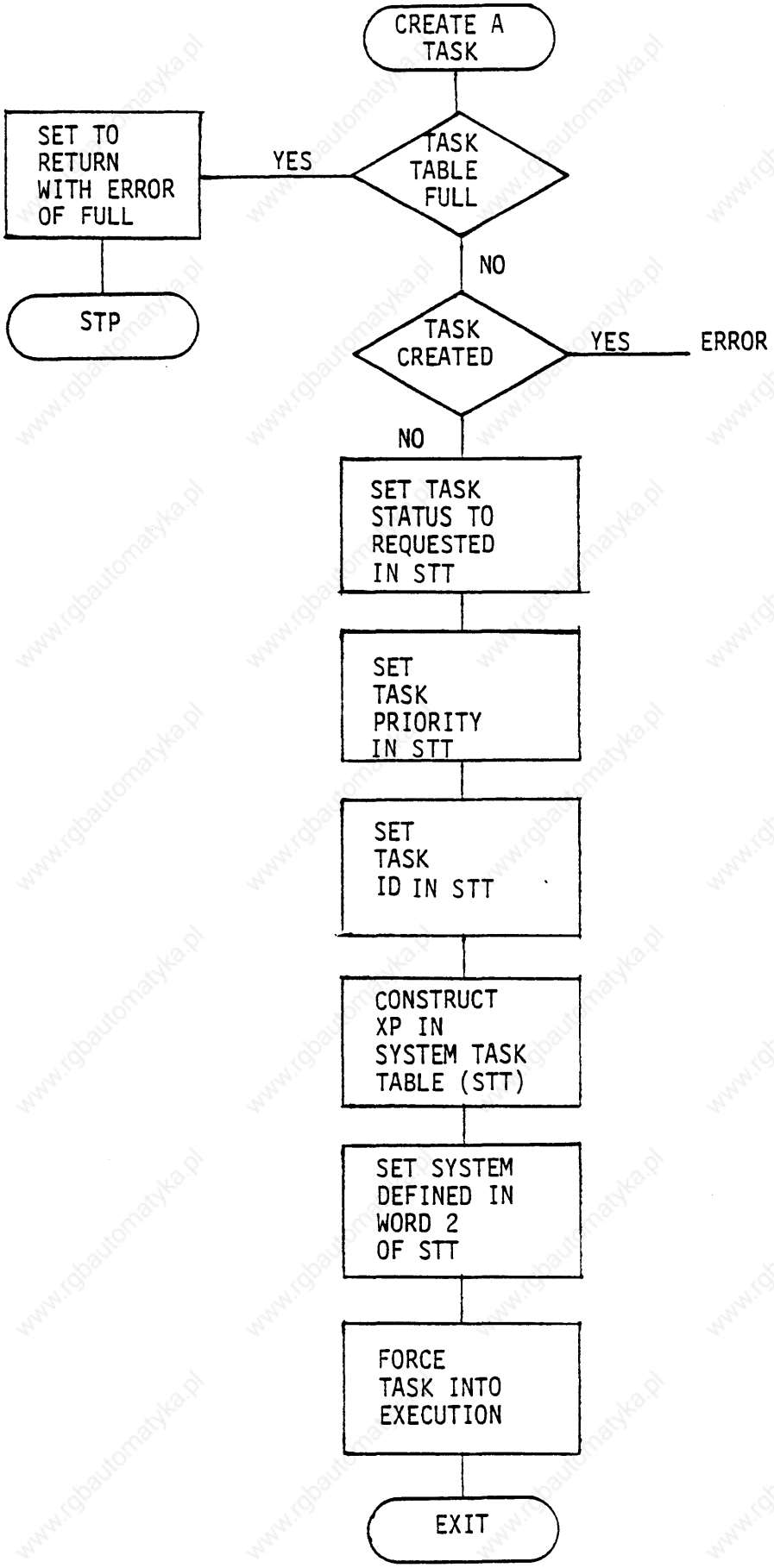
TASK REQUESTS

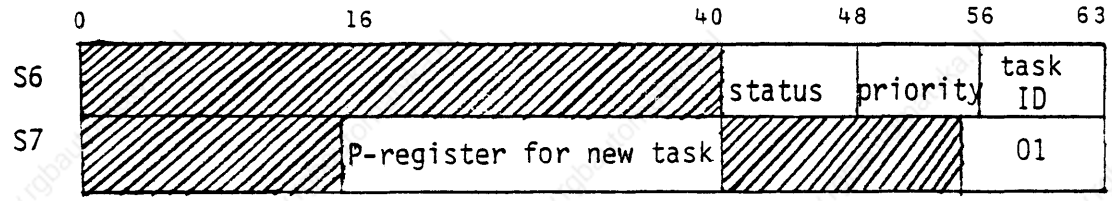
- THE EXECUTIVE REQUEST PROCESSOR IS INITIATED BY THE NORMAL EXIT CHANNEL PROCESSOR.

- THE REQUEST IS PASSED TO EXEC IN REGISTERS S6 AND S7

- EXECUTIVE REQUESTS ARE:

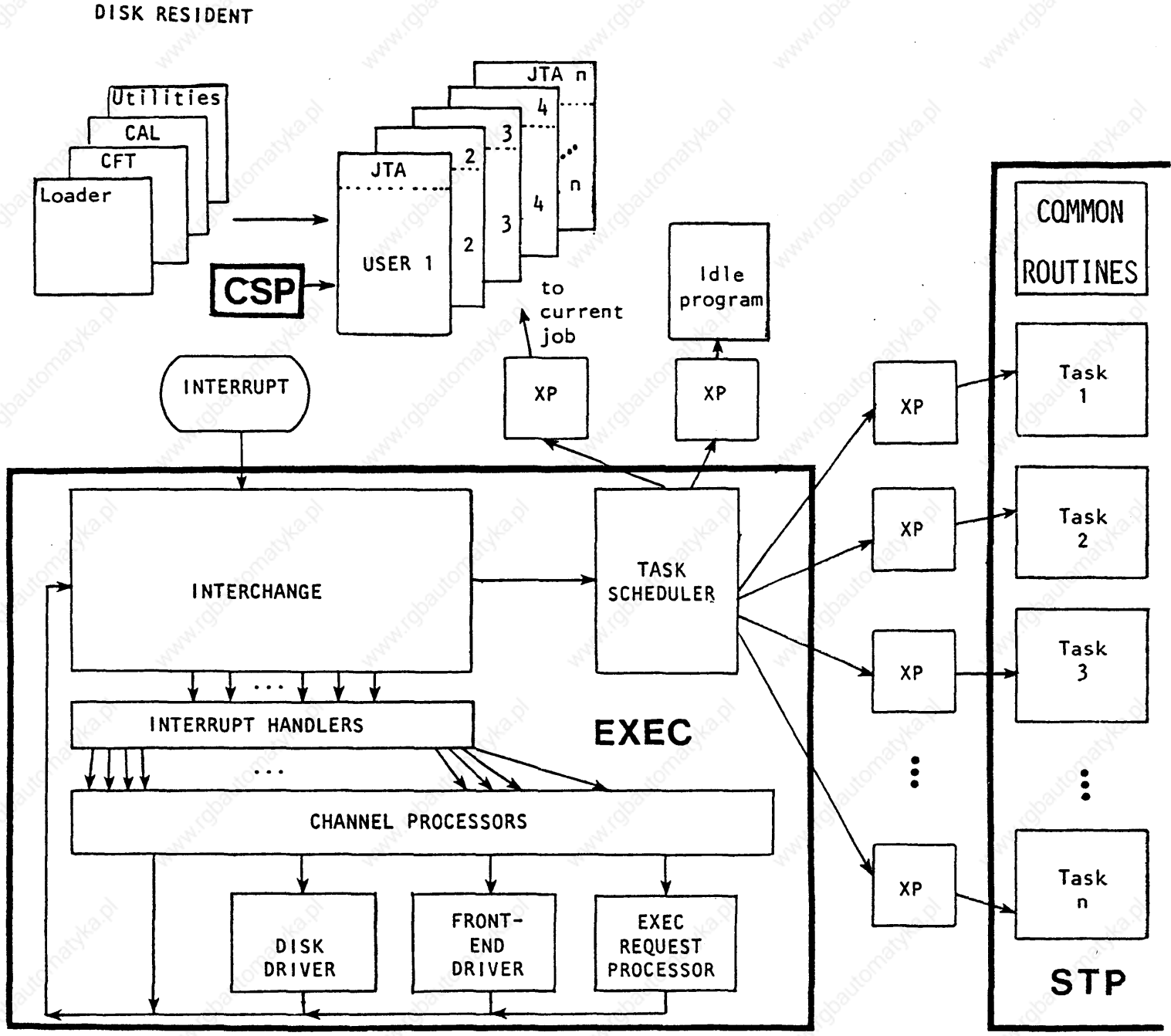
  CREATE A TASK
  READY A TASK
  SELF SUSPEND TASK
  ASSIGN CHANNEL
  STATION I/O REQUEST
  DISK BLOCK I/O REQUEST
  READY TASK AND SUSPEND SELF
  GET TIME AND DATE
  CONNECT USER JOB TO CPU
  DISCONNECT USER JOB FROM CPU
  POST A MESSAGE IN HISTORY BUFFER
  SET MEMORY SIZE
  START/STOP OPERATING SYSTEM
  DISPLAY MEMORY/EXCHANGE PACKAGE
  ENTER MEMORY/EXCHANGE PACKAGE
  SET/CLEAR SYSTEM BREAKPOINT


CREATE A TASK
TASK TABLE FULL
YES
SET TO RETURN WITH ERROR OF FULL
STP
NO
TASK CREATED
YES
ERROR
NO
SET TASK STATUS TO REQUESTED IN STT
SET TASK PRIORITY IN STT
SET TASK ID IN STT
CONSTRUCT XP IN SYSTEM TASK TABLE (STT)
SET SYSTEM DEFINED IN WORD 2 OF STT
FORCE TASK INTO EXECUTION
EXIT


0
16
40
48
56
63
S6
S7
status
priority
task ID
P-register for new task
01

SYSTEM EXECUTIVE PROGRAM (EXEC)

- CONTROL CENTER FOR COS
- EXECUTES IN MONITOR MODE
- CAN ACCESS ALL OF MEMORY

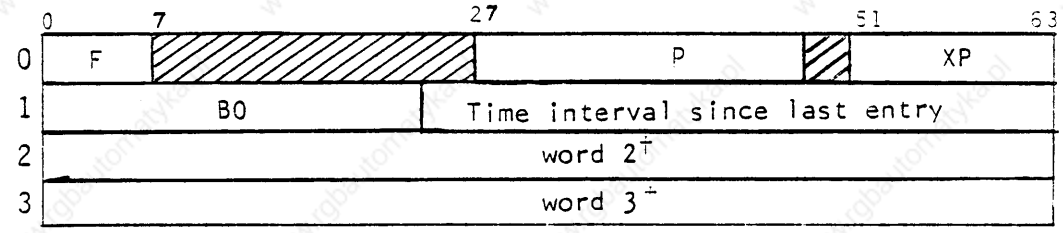


EXEC HISTORY TRACE

- CIRCULAR BUFFER BEGINS AT LOCATION DBF
- LOCATION DBFP POINTS TO NEXT BUFFER ADDRESS
- EACH TRACE MESSAGE 4 WORDS IN LENGTH
- CIRCULAR BUFFER HOLDS THE 1024 MOST CURRENT MESSAGES.
- FUNCTIONS MAY BE COLLECTIVELY ENABLED/DISABLED

  THROUGH SETTING OF LOCATION DBUGM AND ITS ASSOCIATED FUNCTION TABLE.
- SETTING LOCATION DBUGM NONZERO ENABLES ALL FUNCTIONS
- SETTING DBUGM+FUNCTION NUMBER ENABLES THE FUNCTION

```
                          *
                          *      HISTORY BUFFER FUNCTION FLAGS
                          *
0405142300000000000001  DBUGM    CON     'ALL'L+1        UNIVERSAL ENABLE FLAG
0445172220000000000000          CON     'IOI'L          I/O INTERRUPT
0525162120000000000000          CON     'UNE'L          USER NORMAL EXIT
0515162120000000000000          CON     'SNE'L          STP NORMAL EXIT
0425162120000000000000          CON     'ENE'L          EXEC NORMAL EXIT
0511242220000000000000          CON     'RTI'L          REAL TIME INTERRUPT
0541202060000000000000          CON     'XPC'L          COPY USER XP TO JTA
0465032520000000000000          CON     'MCU'L          MCU COMMAND
0515032204500000000000          CON     'SCHJ'L         SCHEDULE USER JOB
0421112365100000000000          CON     'DIOR'L         DISK DRIVER REQUEST
0445242325140000000000          CON     'ITMS'L         INTER-TASK MESSAGE
0425052220000000000000          CON     'EEI'L          ERROR EXIT
                  17            BSSZ    DBUGM+DBUGN-W.*
0000000000000000000000  XCHGBP   CON     0               XCHG PACKAGE TRACE CURRENT POSITION
0000000000000000000000  DBFP     CON     0               DEBUG BUFFER CURRENT POSITION
```



Field	Word	Bits	Description
F	0	0-6	Function number
P	0	27-50	Program register of interrupted exchange package
XP	0	51-63	Exchange package address

FUNCTION		P-REG	SAVED B0	XP	INTERVAL	WORDS 2 AND 3				
011	DQM	0001660A	0000001B	01500	0000000000000472	011440000202271000070220			@	
011	DQM	0001660A	0000001B	01500	0000000000000276	000001001111100000145000		@		
001	I/O	0001660A	0377760A	01500	0000000000003050	000005000002303000067053		+		
001	I/O	0001660A	0377760A	01500	0000000000000673	000005000002303000067173				
001	I/O	0001660A	0377760A	01500	0000000000000501	000004000002303000067212				
001	I/O	0001660A	0377760A	01500	0000000000002340	000005000002303000067656				
001	I/O	0001660A	0377760A	01500	0000000000000714	000005000002303000066274				
001	I/O	0001660A	0377760A	01500	0000000000000612	000004000002303000066325				
001	I/O	0001660A	0377760A	01500	0000000000000774	000005000002303000067755				
001	I/O	0001660A	0377760A	01500	0000000000001122	000005000002303000070167				
010	SCP	0001660A	00000000A	01500	0000000000000777	000000000000000000000000	001300126500077777 1500		+P	@
004	ENE	0001660A	0001461C	01500	0000000000000364	000000000000000000000000	001300126500077777 1500		+P	@
001	I/O	0001660C	0377760A	01500	0000000000203235	000004000002303000070220				
011	DQM	0001660C	0000001B	01500	0000000000000472	011440000202271000070220			@	
011	DQM	0001660C	0000001B	01500	0000000000000276	000001001111100000145000		@		
001	I/O	0001660C	0377760A	01500	0000000000003050	000005000002303000067053		+		
001	I/O	0001660C	0377760A	01500	0000000000000673	000005000002303000067173				
001	I/O	0001660C	0377760A	01500	0000000000000501	000004000002303000067212				
001	I/O	0001660C	0377760A	01500	0000000000002340	000005000002303000067656				
001	I/O	0001660C	0377760A	01500	0000000000000714	000005000002303000066274				
001	I/O	0001660C	0377760A	01500	0000000000000612	000004000002303000066325				
001	I/O	0001660C	0377760A	01500	0000000000000774	000005000002303000067755				
001	I/O	0001660C	0377760A	01500	0000000000001122	000005000002303000070167				
010	SCP	0001660C	00000000A	01500	0000000000000777	000000000000000000000000	001300126500077777 1500		+P	@
004	ENE	0001660C	0001461C	01500	0000000000000364	000000000000000000000000	001300126500077777 1500		+P	@
001	I/O	0001652C	0377760A	01500	0000000000203270	000004000002303000070220				
011	DQM	0001652C	0000001B	01500	0000000000000472	011440000202271000070220			@	
011	DQM	0001652C	0000001B	01500	0000000000000276	000001001111100000145000		@		
001	I/O	0001652C	0377760A	01500	0000000000003050	000005000002303000067053		+		
001	I/O	0001652C	0377760A	01500	0000000000000673	000005000002303000067173				
001	I/O	0001652C	0377760A	01500	0000000000000501	000004000002303000067212				
005	RTI	0001652C	00000000C	01500	0000000000002025	100000000000000000000000	001300126500077777 1500		+P	@
004	ENE	0101636D	0101676A	02060	0000000000001345	000000000000000000001650	001300126500077777 1500		+P	@
005	RTI	0050606B	0102432B	02060	0000000000006456	000000000000000000001650	001300126500077777 1500		+P	@
003	SNE	0050606B	0102432B	02060	0000000000000753	000000020202020000000714	000000000000000001212020			
012	ITM	0050606B	0102432B	02060	0000000000000250	100000000030307000033333	000000000000000000036610	0 6		=

2.23

	FUNCTION	P-REG	SAVED B0	XP	INTERVAL	WORDS 2 AND 3				
004	ENE	0050606D	0102432B	02060	0000000000000623	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0050647D	0102432B	02060	0000000000001564	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0050647D	0102432B	02060	0000000000000753	00000000000000000000006	00000000000000000000002			
004	ENE	0050650B	0102432B	02060	0000000000001632	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0050606B	0102512D	02060	0000000000004210	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0050606B	0102512D	02060	0000000000000753	00000002020202000000704	00000000000000001212020			
012	ITM	0050606B	0102512D	02060	0000000000000250	10000000001050000104075	00000000000000000000000	@ =		
004	ENE	0050606D	0102512D	02060	0000000000000623	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0051004A	0102165C	02060	0000000000003561	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0051004A	0102165C	02060	0000000000000753	04000300000000000000000	00000000000000001212020	@		
012	ITM	0051004A	0102165C	02060	0000000000000250	04000000000000000000007	00073475200540000004424	@		
004	ENE	0051004C	0102165C	02060	0000000000000623	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0051005C	0102165C	02060	0000000000000733	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0051005C	0102165C	02060	0000000000000753	00000000000000000000006	00000000000000000000002			
004	ENE	0051006A	0102165C	02060	0000000000001630	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0103051A	0101676A	02060	0000000000003430	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0103051A	0101676A	02060	0000000000000753	00000000030107400045310	00000000054262000002011	< J	X	
011	DQM	0103051A	2646214A	02060	0000000000000366	00000202070400000323200	00000000054262000002011		X	
001	I/O	0103051C	0377760A	02060	0000000000001715	00000500000230300067656	01000000000000407777777			
001	I/O	0103051C	0377760A	02060	0000000000000714	00000500000230300066274	01000000000000407777777			
004	ENE	0103051C	0101676A	02060	0000000000001012	00000000000000000001650	00130012650007777771500		+P	@
001	I/O	0103051C	0377760A	02060	0000000000000707	00000400000230300066325	01000000000000407777777			
001	I/O	0103051C	0377760A	02060	0000000000000774	00000500000230300067755	01000000000000400020777			!
001	I/O	0103051C	0377760A	02060	0000000000001156	00000500000230300070337	01000000000000400020777			!
005	RTI	0103051C	0377760A	02060	0000000000000640	00000300000000000001650	00130012650007777771500		+P	@
004	ENE	0103051C	0101676A	02060	0000000000000747	00000000000000000001650	00130012650007777771500		+P	@
005	RTI	0101636B	0101676A	02060	0000000000001120	00000000000000000001650	00130012650007777771500		+P	@
003	SNE	0101636B	0101676A	02060	0000000000001010	00000000000000000000005	00000000000000000000003			
004	ENE	0074442D	0074450B	02100	0000000000001414	00000000000000000001657	00130012650007777771500		+P	@
001	I/O	0050437C	0377760A	02100	0000000000006010	00000400000230300070370	01000000000000400020777			!
005	RTI	0050437C	0377760A	02100	0000000000000753	00000300000000000001657	00130012650007777771500		+P	@
004	ENE	0050437C	0050335B	02100	0000000000000747	00000000000000000001657	00130012650007777771500		+P	@
005	RTI	0050412A	0075103B	02100	0000000000004246	00000000000000000001657	00130012650007777771500		+P	@
013	ERR	0050412A	0075103B	02100	0000000000000670	05152425000000000000000	00000200000000000000000	STT		

2.25

DISK DRIVER (RO11)

- DOES THE PHYSICAL I/O ON DISKS
- REQUESTS TO RO11 QUEUED VIA THE REQUEST TABLE (RQT)
- UPDATES THE DATASET PARAMETER TABLE (DSP)
- EXAMINES THE DISK RESERVATION TABLE (DRT) FOR USEABLE DISK SPACE
- PERFORMS SERVO OFFSET AND DATA STROBE FUNCTIONS FOR ERROR RECOVERY

COMMON SUBROUTINES

SYSTEM TASK PROCESSOR

- DEFINITION - CONSISTS OF TABLES, COMMON SUBROUTINES, TASKS, AND I/O ROUTINES.

- COMMON SUBROUTINES
  - RE-ENTRANT ROUTINES
  - USED BY TASKS

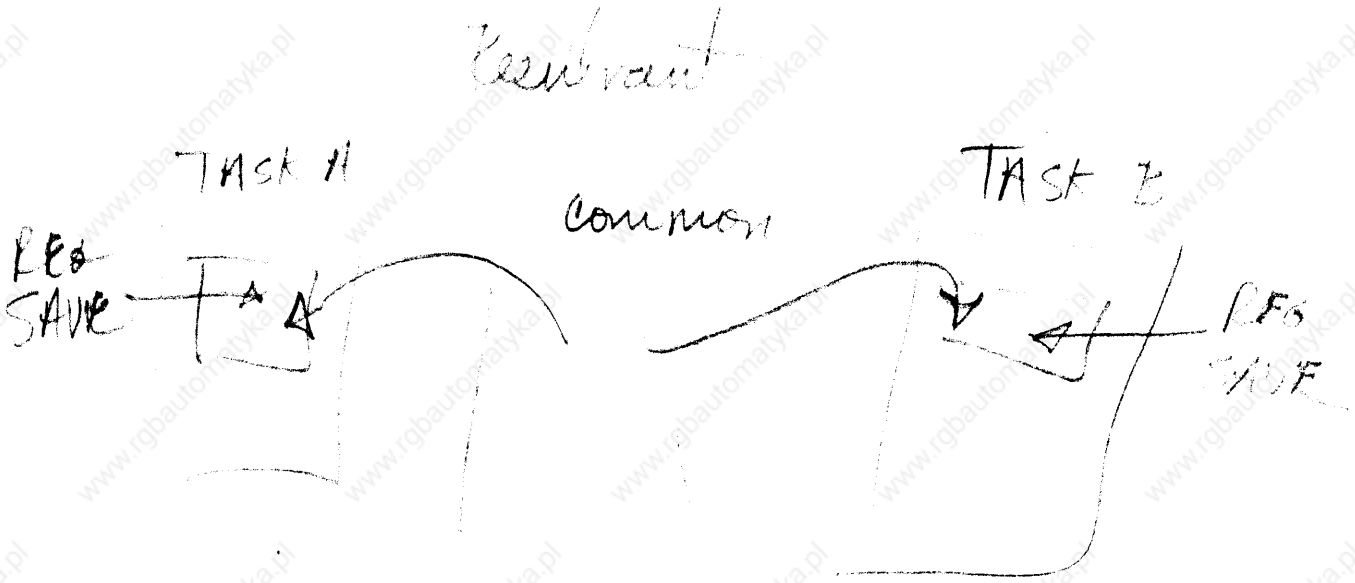
- TASKS
  - PERFORMS A SPECIFIC OPERATION
  - CAN BE CALLED BY OTHER TASKS
  - HAVE THEIR OWN XP's
  - HAVE THEIR OWN ID NUMBERS ($0\text{-}35_8$).
  - HAVE THEIR OWN PRIORITIES ($0\text{-}377_8$).
  - BA AND LA THE SAME FOR ALL TASKS (BA=B@STP, LA=I@MEM)
  - COMMUNICATE WITH EXEC, EACH OTHER AND WITH USER JOBS.

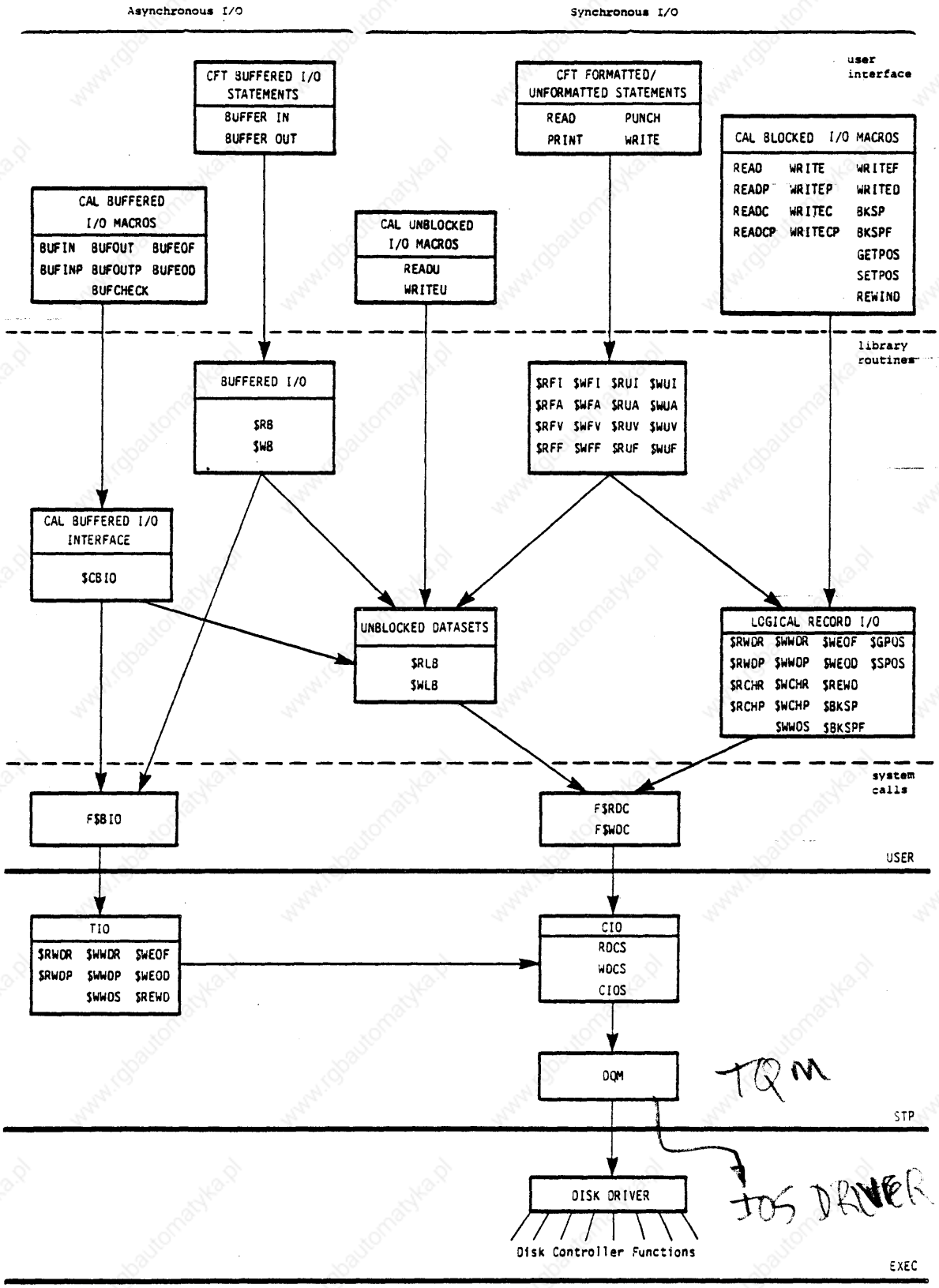
- TABLES
  - USED BY STP AND EXEC

STP COMMON ROUTINES

- USED BY TASKS TO PERFORM:

  TASK LOGICAL INPUT/OUTPUT (TIO)
  CIRCULAR INPUT/OUTPUT (CIO)
  MEMORY MANAGEMENT
  ITEM CHAINING/UNCHAINING
  TASK TO TASK COMMUNICATION





Overview of COS I/O

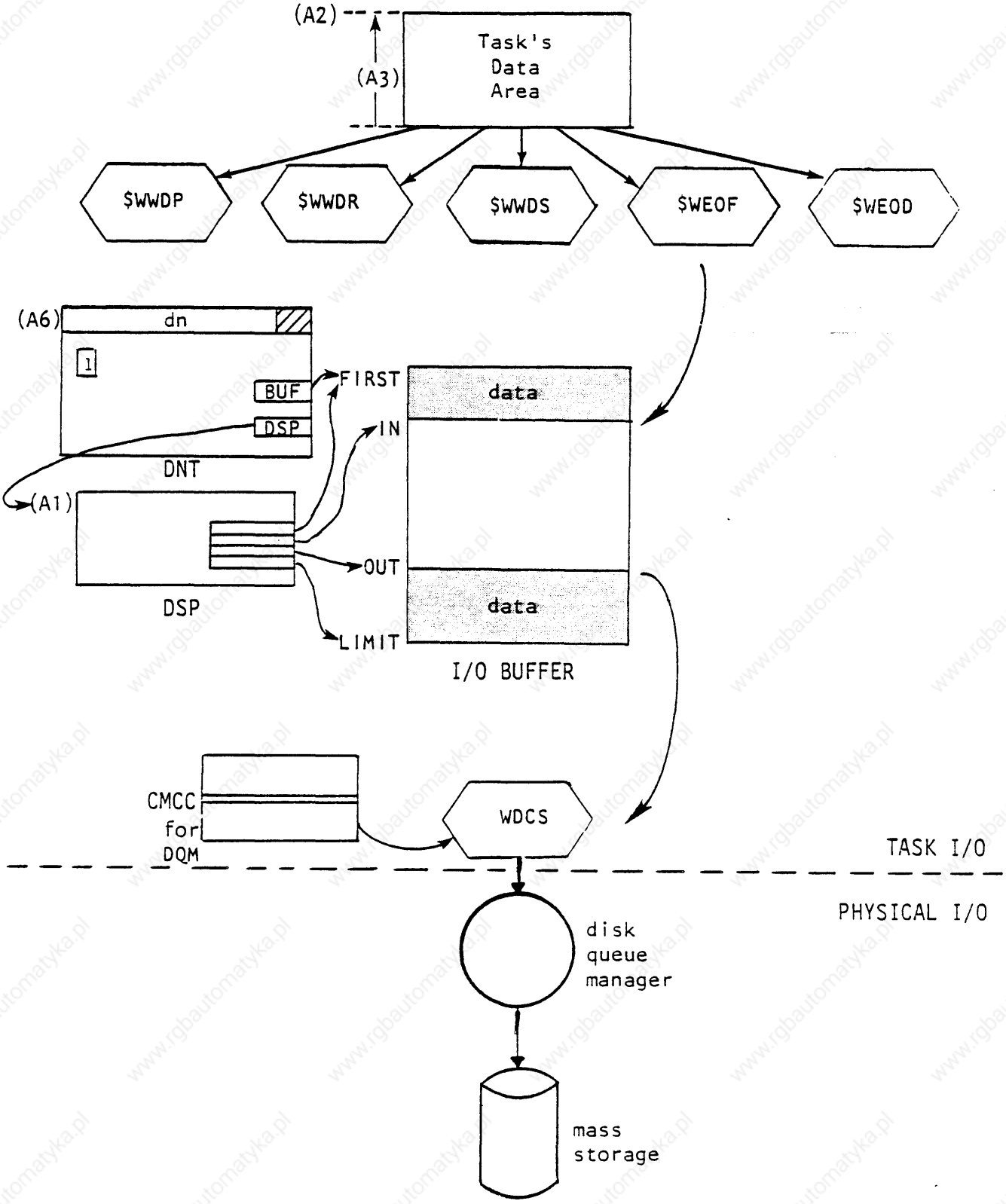
TASK LOGICAL I/O (TIO)

- ALLOWS A SYSTEM PROGRAMMER TO DO LOGICAL I/O AT THE TASK LEVEL.

- TIO ROUTINES ARE:
  - $RWDP/$RWDR-READ WORDS PARTIAL/FULL RECORD
  - $WWDP/$WWDR-WRITE WORDS PARTIAL/FULL RECORD
  - $WEOF-WRITE END OF FILE
  - $WEOD-WRITE END OF DATA
  - $REWD-REWIND A DATASET
  - $WWDS-WRITE WORDS--UNUSED BIT COUNT

- TASKS CALL TIO BY PLACING REQUIRED PARAMETERS IN 'A' REGISTERS AND EXECUTING A RETURN JUMP TO THE ROUTINE.

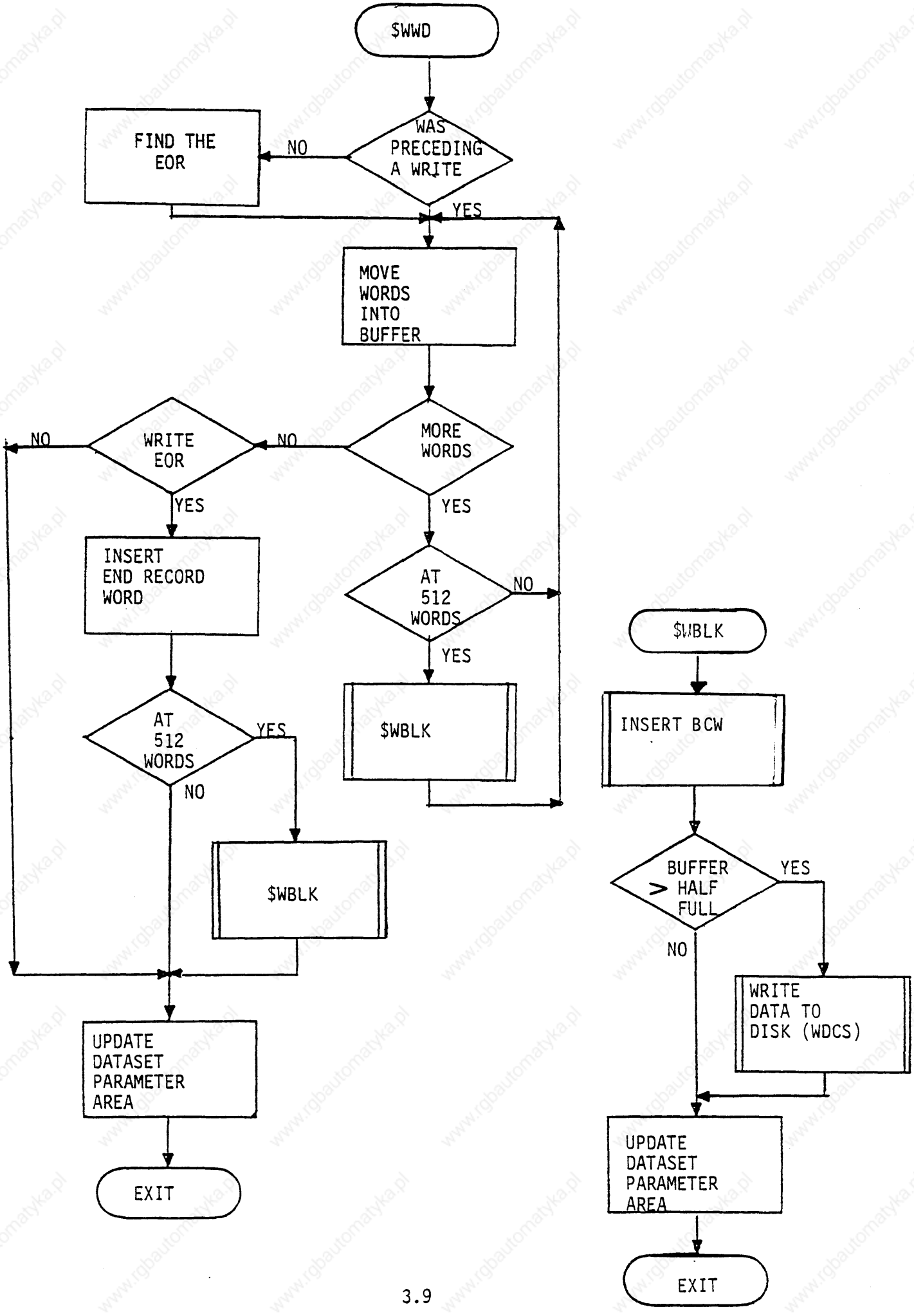
- 1 TIO ROUTINE ($WWDP/$WWDR) WILL BE EXAMINED HERE. REFER TO SM0040 FOR REMAINDER OF TIO ROUTINES.



TIO logical write

- $WWDP/$WWDR WRITES THE NUMBER OF WORDS SPECIFIED INTO THE I/O BUFFER. $WWD IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

ENTRY CONDITIONS:	(A1)	Address of DSP
	(A2)	FWA of task's data area
	(A3)	Word count If count is 0, no data is transferred
	(A6)	Address of DNT
	(A7)	Address of JXT (=0 if not job related)
RETURN CONDITIONS:	(A0)	Status
		<0 TIO error
		=0 Logical I/O complete


$WWD
WAS PRECEDING A WRITE
NO
FIND THE EOR
YES
MOVE WORDS INTO BUFFER
MORE WORDS
NO
YES
WRITE EOR
NO
YES
INSERT END RECORD WORD
AT 512 WORDS
NO
YES
$WBLK
AT 512 WORDS
YES
NO
$WBLK
UPDATE DATASET PARAMETER AREA
EXIT
$WBLK
INSERT BCW
BUFFER > HALF FULL
YES
NO
WRITE DATA TO DISK (WDCS)
UPDATE DATASET PARAMETER AREA
EXIT

```
*    WRITE WORDS                                                    S.16418
*        ENTRY:                                                     S.16419
*                    A1  @FCB                                       S.16420
*                    A2  FWA                                        S.16421
*                    A3  COUNT                                      S.16422
*                    A6  DNT ADDRESS                                S.16423
*                    A7  JXT ADDRESS, =0 IF NOT JOB RELATED         S.16424
*        EXIT:                                                      S.16425
*                    A1  @FCB                                       S.16426
*                    A2  FWA                                        S.16427
*                    A3  COUNT                                      S.16428
*                    A6  DNT ADDRESS                                S.16429
*                    A7  JXT ADDRESS                                S.16430
*        MODIFIES:                                                  S.16431
*                    A0,4,5    S0,1,2,3,4,5,6,7                     S.16432
*                    DPTM WORDS IN DSP                              S.16433
*    WRITE WORDS, PARTIAL MODE                                      S.16434
$WWDP    =          *                                              S.16435
         S6         0                                              S.16436
         J          WW01                                           S.16437
*    WRITE WORDS, RECORD MODE                                       S.16438
$WWDR    =          *                                              S.16439
         S6         >1                                             S.16440
WW01     =          *                                              S.16441
         S1         A6              DNT ADDRESS                    S.16442
         S2         A7              JXT ADDRESS                    S.  443
         S1         S1<D'24                                        S.  444
         S1         S1!S2                                          S.16445
         W@DPTM+5,A1 S1                                            S.16446
WW02     =          *                                              S.16447
         A7         B0                                             S.16448
         S1         A7              B.ZA                           S.16449
         S2         A2              B.ZB                           S.16450
         S1         S1<D'24                                        S.16451
         S1         S1!S2                                          S.16452
         W@DPTM+2,A1 S1                                            S.16453
         R          TDSP            CALCULATE A7 = DSP BASE        S.16454
         S4         5,A1            PW                             S.16455
         S3         2,A1            IN                             S.16456
         A0         S4                                             S.16458
         S2         <30                                            S.16459
         JAN        WW03            NOT FIRST WRITE                S.16460
         S4         S3!S4&S2        SET POWA                       S.16461
         S2         <1                                             S.16462
         A4         S3                                             S.16463
         S3         S3+S2           IN + 1                         S.16464
         S2         164                                            S.16465
         S2         S2<71                                          S.16466
         A4         A4+A7                                          S.16467
         0,A4       A0              CLEAR FIRST BCW                S.16468
         A4         A4-A7                                          S.16469
         S4         S2!S4           INSERT RFW BITS                S.16470
WW03     =          *                                              S.16471
         S0         S4<4                                           S.16472
         S2         >4                                             S.  473
         JSM        WW04            PRECEDING WRITE                S.16474
         R          WB30            PROCESS WRITE AFTER READ       S.16475
         JAN        WW22            ERROR                          S.16476
```

```
        R       TDSP        CALCULATE A7 = DSP BASE ADDRESS
WW04    =       *
        S1      1,A1        FIRST
        S0      S4
        S4      S6!S4&S2    CLEAR RFD BITS
        S2      0
        JSP     WW05        NOT BOR
        S0      S0<2
        S2      <17
        JSP     WW05        NOT BOR
        S0      S0<1
        S2      <43
        JSM     WW39        WRITE AFTER EOD
WW05    =       *
        S2      S2<30
        A3      A2+A3       LWA + 1
        S4      #S2&S4      CONDITIONAL CLEAR BFI, BRI
        S1      S1-S3       F - I
        S2      <11
        A4      S3          I
        S1      S2&S1
        S1      S3+S1
        A5      S1          NCWA
        W@DPTM,A1 S4        T.ZA
WW10    =       *
        S7      W@DPBIO,A1
        S6      A1
        A1      A2-A3
        A6      A4-A5
        S1      A1
        A0      A1-A6
        S2      <6
        A1      S6
        JAP     WW11        COUNT FIRST
        S1      A6
WW11    =       *
        S0      S1
        S2      #S1&S2
        A6      S2
        A6      A6+1        FIRST SEGMENT LENGTH
        A7      A4+A7
        JSZ     WW15        ZERO LENGTH
        S0      S7
        ERRIF   S@DPBIO,NE,0
        JSM     WWVMV       USE VECTOR MOVE IF BUFFERED I/O
        S0      0,A2
WW14    =       *
        A6      A6-1
        A2      A2+1
        A0      A6-1        DONE?
        A7      A7+1
        A4      A4+1
        -1,A7   S0
        S0      0,A2
        JAP     WW14        NO
        A6      D'64        MOVE IN 64 WORD CHUNKS
WW15    =       *
        A0      A4-A5
        JAM     WW16        NOT AT BCW
        A7      A7-A4
        R       $WBLK       PROCESS BCW
        JAN     WW22        ERROR
        J       WW10
```

```
          A0        A2-A3                                          S.16540
          S0        0,A2                                           S.16541
          JAM       WW14            NOT AT EOC                     S.16542
          A7        A7-A4                                          S.16543
          S0        W@DPTM,A1       T.ZA                           S.16544
*     END OF COUNT                                                 S.16545
          S4        W@DPTM,A1       PW (T.ZA)                      S.16546
          JSP       WW21            PARTIAL MODE                   S   547
*     RECORD MODE                                                  S.16548
          S1        W@DPTM,A1       T.ZA                           S.16549
          S2        <43                                            S.16550
          S1        S1>17                                          S.16551
          S2        S2<11                                          S.16552
          A6        S4              @ PCW                          S.16553
          S1        S2&S1           BFI, BRI                       S.16554
          S2        >4                                             S.16555
          S1        S4!S1&S2        INSERT MODE BITS               S.16556
          A7        A7+A4                                          S.16557
          0,A7      S1              STORE RCW                      S.16558
          A7        A7-A4                                          S.16559
          A7        A7+A6                                          S.16560
          S6        0,A7            PCW                            S.16561
          A2        A6+1                                           S.16562
          S1        A4                                             S.16563
          A2        A4-A2           FWI                            S.16564
          S2        <30                                            S.16565
          S4        S1!S4&S2        UPDATE PCWA                    S.16566
          A4        A4+1                                           S.16567
          S1        A2                                             S.16568
          A0        A4-A5                                          S.16569
          S6        S6!S1           INSERT FWI                     S.16570
          0,A7      S6                                             S.16571
          A7        A7-A6                                          S.16572
          JAM       WW21            NOT AT BCW                     S.16573
          R         WB01            WRITE BLOCK                    S.16574
          JAN       WW22            ERROR                          S   575
*     PARTIAL MODE                                                 S.  576
WW21      =         *                                              S.16577
          S1        A4                                             S.16578
          5,A1      S4              UPDATE PW                      S.16579
          S2        <30                                            S.16580
          S3        S1!S3&S2        INSERT NWA                     S.16581
          S2        <60                                            S.16582
          S3        S2&S3           CLEAR BP                       S.16583
          2,A1      S3              UPDATE IN                      S.16584
          A2        W@DPTM+2,A1     B.ZB                           S.16585
          A3        A3-A2           COUNT                          S.16586
          A0        0               COMPLETE FLAG                  S.16587
          J         WW22                                           S002.1447
WW39      =         *                                              S.16592
          S1        W@DPERR,A1                                     S.16593
          S3        >1                                             S.16594
          S3        S3>S@DPERW      WRITE PAST EOD                 S.16595
          S1        S1!S3           SET ERROR FLAG                 S.16596
          W@DPERR,A1 S1                                            S.16597
WW22      =         *                                              S002.1448
          S1        W@DPTM+2,A1                                    S002.1449
          S1        S1>D'24                                        S002.1450
*                                                                  S.16599
*         S1        48/,24/B0                                      S.16599
*         A1        DSP ADDRESS                                    S.16600
*                                                                  S.16601
WW99      =         *                                              S.16602
          A7        S1                                             S.16603
          B0        A7                                             S   604
          S1        W@DPTM+5,A1                                    S.16605
          A7        S1              NXT ADDRESS                    S.16606
          S1        S1>D'24                                        S.16607
          A6        S1              DNT ADDRESS                    S.16608
          J         B0                                             S.16609
```

```
*         WWVMV     VECTOR MOVE FOR WRITE RECORD                                S.16611
*                                                                               S.16612
*         A2        SOURCE ADDRESS                                              S.16613
*         A4        OUT                                                         S.16614
*         A7        DESTINATION ADDRESS                                         S.16615
*         A6        SHORT VECTOR LENGTH (NOT ZERO)                              S.16616
*         S1        NEGATIVE NUMBER OF WORDS TO BE MOVED                        S.16617
WWVMV     S2        A3              SAVE A3                                     S.16618
          A0        BIOV0SV         V0 SAVE AREA FOR BUFFERED I/O               S.16619
          A3        ZS0             D'64                                        S.16620
          VL        A3                                                          S.16621
          ,A0,1     V0              SAVE V0                                     S.16622
                                                                                S.16623
          A3        S1                                                          S.16624
          A3        -A3                                                         S.16625
          A4        A4+A3           INCREMENT OUT                               S.16626
                                                                                S.16627
WWVMV1    A0        A2              SOURCE ADDRESS                              S.16628
          VL        A6                                                          S.16629
          V0        ,A0,1                                                       S.16630
          A0        A7              DESTINATION ADDRESS                         S.16631
          A3        A3-A6           WORDS LEFT TO MOVE                          S.16632
          A2        A2+A6                                                       S.16633
          A7        A7+A6                                                       S.16634
          A6        ZS0             D'64                                        S.16635
          ,A0,1     V0                                                          S.16636
          A0        -A3                                                         S.16637
          JAM       WWVMV1          LOOP UNTIL ALL WORDS MOVED                  S.16638
                                                                                S.16639
          VL        A5                                                          S.16640
          A0        BIOV0SV                                                     S.16641
          V0        ,A0,1           RESTORE V0                                  S.16642
          A3        S2              RESTORE A3                                  S.16643
          J         WW15                                                        S.16644
                                                                                S.16645
**********************************************************************************S.16646
**                                                                              S.16647
* NAME    TDSP                                                                  S.16648
*         CALCULATE DSP POINTER BASE                                            S.16649
*         = 0 IF JXT ADDRESS =0                                                 S.16650
*         = JTA ADDRESS IF DNJTF =1                                             S.16651
*         = USER BA IF DNJTF =0                                                 S.16652
* ENTRY   TDSP:                                                                 S003.737
*         A1        DSP ADDRESS                                                 S003.738
*                                                                               S003.739
*         TDSP1:                                                                S003.740
*         S7        16/0,24/DNT ADDRESS,24/JXT ADDRESS                          S003.741
*                                                                               S003.742
* EXIT    A1        DSP ADDRESS (TDSP ENTRY ONLY)                               S003.743
*         A7        BASE OF DSP POINTERS                                        S.16656
*                                                                               S.16657
*         REGISTERS MODIFIED -                                                  S.16658
*                   A0,A7  S0,S7                                                S.16659
*                                                                               S.16660
**********************************************************************************S.16661
TDSP      S7        W@DPTM+5,A1                                                 S.16662
TDSP1     =         *               ENTRY W/ S7 = JXT ADDRESS                   S.16663
          A0        S7              JXT ADDRESS                                 S.16664
          A7        0                                                           S.16665
          JAZ       TDSP9           NOT JOB RELATED                             S.16666
          A7        S7              JXT ADDRESS                                 S.16667
          A7        W@JXJTA,A7      JTA ADDRESS                                 S.16668
          S7        S7>D'24                                                     S.16669
          TDSPA,0   A6              SAVE A6                                     S.16670
          A6        S7              DNT ADDRESS                                 S.16671
          S0        W@DNJTF,A6                                                  S.16672
          S0        S0<S@DNJTF                                                  S.16673
          A6        L@JTA                                                       S.16674
          JSM       TDSP8           IN JTA                                      S.16675
          A7        A6+A7           USER BA                                     S.16676
TDSP8     A6        TDSPA,0                                                     S.16677
TDSP9     J         B0              RETURN                                      S.16678
TDSPA     CON       0                                                           S.16679
```

CIRCULAR I/O

- PERFORMS PHYSICAL I/O ON A DATASET

  ACCESSIBLE TO TASKS THROUGH TIO AND DIRECT CALLS.

  CIO ROUTINES ARE:

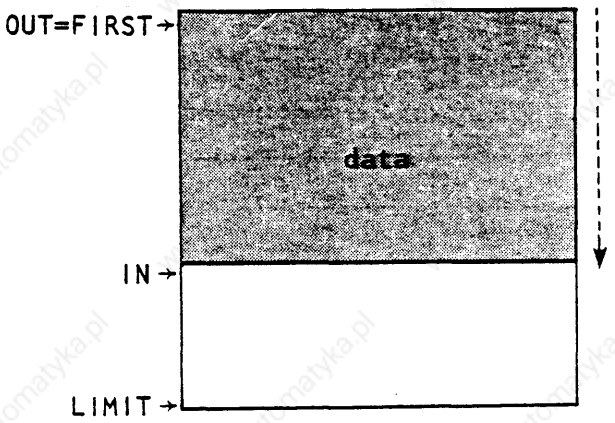
  RDCS-READ CIRCULAR REQUEST

  WDCS-WRITE CIRCULAR REQUEST

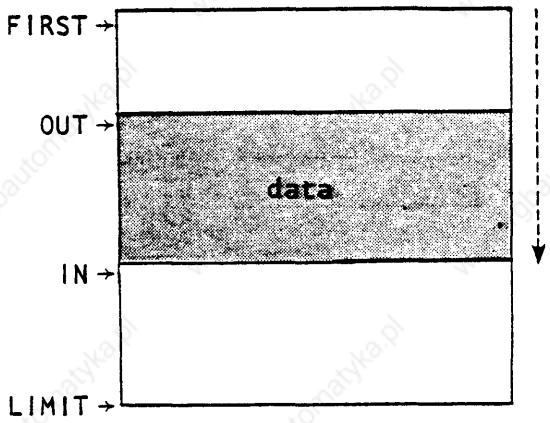
- TASKS CALL CIO BY PLACING REQUIRED PARAMETERS IN 'A' REGISTERS AND EXECUTING A RETURN JUMP TO THE ROUTINE.

- CIO READS/WRITES 512 WORD BLOCKS. THE CALLER HAS THE RESPONSIBILITY OF MAINTAINING THE BUFFER IN/OUT POINTER IN THE DSP. AS SHOWN IN THE PREVIOUS $WWD FLOW DIAGRAM.

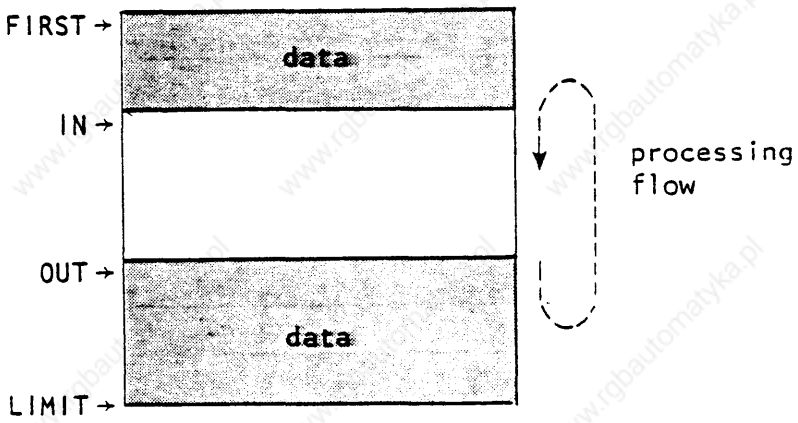
- THE CALLER SENSES COMPLETION OF PHYSICAL I/O BY CALLING GETREPLY. IF A REPLY IS FOUND THE CALLER SHOULD CALL ROUTINE REPCIO WITH S1 AND S2 INTACT FROM GETREPLY.



A. Filling the buffer



B. Emptying the buffer

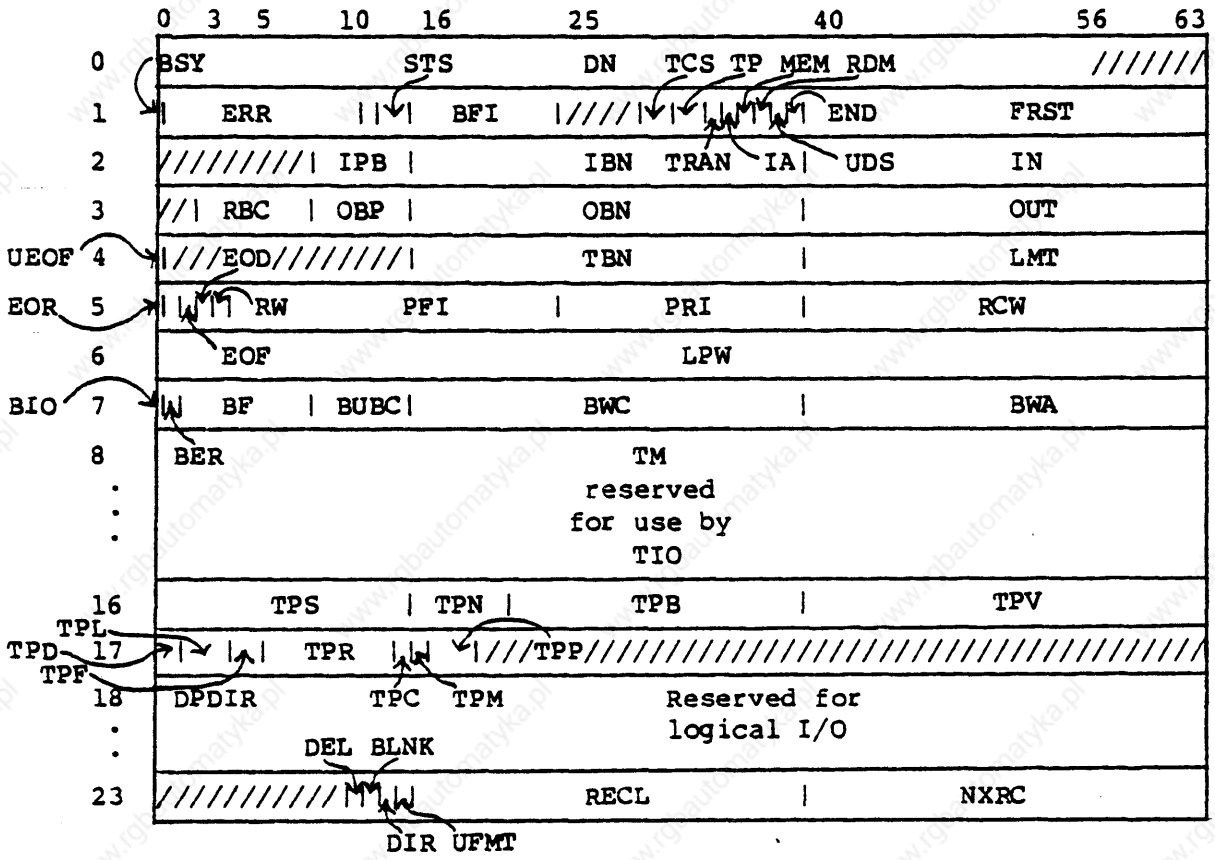


C. Concurrently filling
and emptying the buffer

Physical I/O

DATASET PARAMETER AREA - DSP

Logical I/O requires the presence of a DSP for the dataset in the user's field. Refer to publication SR-0011 for details of DSP use.



Dataset Parameter Area (DSP)

MEMORY MANAGEMENT

- PROVIDES TEMPORARY MEMORY AREAS FOR TASKS (AREAS IN STP)
- MEMORY AREAS ARE OF VARIABLE-SIZE
- MEMORY POOLS ARE USED

MEMORY MANAGEMENT ROUTINES ARE:

MEMAL-ALLOCATES A TASK MEMORY AREA

MEMDE-DEALLOCATES A TASK MEMORY AREA

- MEMAL ALLOCATES A VARIABLE-SIZE MEMORY AREA TO A TASK. MEMAL IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING:

INPUT REGISTERS: (A6) = Number of memory pool from which to allocate

(A7) = Number of words desired

OUTPUT REGISTERS: (A6) = Status:

0 Good status
1 Invalid memory pool number
2 Invalid number of words requested
3 Memory not available

(A7) = Address of first usable word of memory to be allocated

```
         EJECT
*************************************************************************
**
*
*NAME    MEMAL
*
*PURPOSE MEMAL WILL ALLOCATE A VARIABLE SIZE AREA OF MEMORY FROM A
*        MEMORY POOL.
*
*ENTRY   A6 = NUMBER OF THE MEMORY POOL TO ALLOCATE FROM
*        A7 = NUMBER OF WORDS REQUIRED
*
*EXIT    A6 = STATUS
*                  0 = GOOD RETURN
*                  1 = INVALID POOL  NUMBER
*                  2 = INVALID NUMBER OF WORDS REQUESTED
*                  3 = MEMORY NOT AVAILABE
*        A7 = ADDRESS OF FIRST USEABLE WORD ALLOCATED--MEANINGFUL
*             ONLY IF A6 = 0
*
*************************************************************************
MISTAKE1 =         1
MISTAKE2 =         2
MISTAKE3 =         3
MEMAL    =         *
         A5        POOLTBL        BASE ADDRESS OF POOL TABLE
         GET,S5    S6&S7,PTMAX,A5 MAX POOL NUMBER
         A4        S5
         A0        A4-A6
         JAM       MEMER1         JUMP IF POOL NUMBER GREATER THAN MAX
         A5        A5+A6          ADDRESS OF POOL WORD
         S0        0,A5
         JSZ       MEMER1
         A0        A7
         JAZ       MEMER2         JUMP IF ZERO WORDS REQUESTED
*
*        THE TOTAL SIZE TO BE ALLOCATED IS  THE REQUESTED SIZE PLUS
*        THE SIZE OF THE HEADER AND TRAILER
*
         A4        MPHT
         A7        A7+A4          TOTAL SIZE REQUIRED
*
*        VERIFY THAT THE REQUESTED SIZE IS NOT LARGER THAN THE POOL
*
         GET,S1    S6&S7,PTSIZE,A5
         A4        S1
         A0        A4-A7
         JAM       MEMER2
*
         GET,S1    S6&S7,PTBASE,A5
         A4        S1             BASE ADDRESS OF MEMORY POOL
*
*        LOCK OUT INTERRUPTS UNTIL THE SPACE HAS BEEN ALLOCATED
*
         S1        1
         STRLK,A0  S1
MEMAL10  =         *
         GET,S1    S6&S7,MPID,A4  VALIDATE THE HEADER WORD
         S2        IDWRD,A6
         S0        S1-S2
         ERRSN
         GET,S0    S6&S7,MPST,A4
         JSN       MEMAL50        JUMP IF AREA IN USE
         GET,S1    S6&S7,MPSIZE,A4  SIZE OF AVAILABLE AREA
         A1        S1
         A0        A1-A7
         JAN       MEMAL30        REQUEST SIZE UNEQUAL TO AVAILABLE
```

```
*       AT THIS POINT THE SIZE OF THE REQUEST IS EQUAL TO THE                   A0255.1000
*       AVAILIABLE AREA                                                         A0255.1001
*                                                                               A0255.1002
        S1      1                                                               A0255.1003
        PUT,S1  S6&S7,MPST,A4  SET IN USE INDICATOR                             A0255.1004
        A2      A4+A1                                                           A0255.1005
        A2      A2-1           ADDRESS OF TRAILER WORD                          A0255.1006
        PUT,S1  S6&S7,MPST,A2                                                   A0255.1007
*                                                                               A0255.1008
*       RELEASE INTERRUPT LOCKOUT                                               A0255.1009
*                                                                               A0255.1010
        S1      0                                                               A0255.1011
        UNLOCK                                                                  C0983.29
        A7      A4+1           RETURN ADDRESS FIRST USABLE WORD                 A0255.1012
*                                                                               A0255.1013
*       CLEAR THE MEMORY ALLOCATED WITHOUT DESTROYING THE HEADER                A0255.1014
*       AND TRAILER                                                             A0255.1015
*                                                                               A0255.1016
        A6      A7                                                              A0255.1017
MEMAL20 =       *                                                               A0255.1018
        0,A6    S1                                                              A0255.1019
        A6      A6+1                                                            A0255.1020
        A0      A6-A2                                                           A0255.1021
        JAN     MEMAL20                                                         A0255.1022
*                                                                               A0255.1023
*       RETURN TO REQUESTOR WITH GOOD STATUS                                    A0255.1024
*                                                                               A0255.1025
        A6      0                                                               A0255.1026
        J       B00                                                             A0255.1027
*                                                                               A0255.1028
*       HERE AN AVAILABLE MEMORY AREA HAS BEEN LOCATED.  IT IS UNEQUAL          A0255.1029
*       IN SIZE TO THE REQUEST                                                  A0255.1030
*                                                                               A0255.1031
MEMAL30 =       *                                                               A0255.1032
        JAM     MEMAL50        AVAILABLE IS LESS THAN REQUEST                   A0255.1033
*                                                                               A0255.1034
*       REPLACE THE EXISTING HEADER WORD AND CREATE A NEW TRAILER               A0255.1035
*                                                                               A0255.1036
        A2      A4+A7                                                           A0255.1037
        A2      A2-1           TRAILER ADDRESS                                  A0255.1038
        S1      0                                                               A0255.1039
        0,A2    S1             CLEAR OUT ANY GARBAGE                            A0255.1040
        S1      A7                                                              A0255.1041
        PUT,S1  S6&S7,MPSIZE,A4                                                 A0255.1042
        PUT,S1  S6&S7,MPSIZE,A2  SIZE OF ALLOCATED AREA                         A0255.1043
        S1      1                                                               A0255.1044
        PUT,S1  S6&S7,MPST,A4                                                   A0255.1045
        PUT,S1  S6&S7,MPST,A2  IN USE INDICATOR                                 A0255.1046
        S2      IDWRD,A6                                                        A0255.1047
        PUT,S2  S6&S7,MPID,A4                                                   A0255.1048
        PUT,S2  S6&S7,MPID,A2  POOL ID                                          A0255.1049
*                                                                               A0255.1050
*       CREATE A NEW HEADER AND TRAILER FOR THAT SPACE WHICH IS                 A0255.1051
*       STILL AVAILABLE                                                         A0255.1052
*                                                                               A0255.1053
        A2      A2+1           ADDRESS OF NEW HEADER                            A0255.1054
        A1      A1-A7          SIZE OF NEW AREA                                 A0255.1055
        A6      A2+A1                                                           A0255.1056
        A6      A6-1           ADDRESS OF NEW TRAILER                           A0255.1057
        S6      0                                                               A0255.1058
        0,A2    S6             CLEAR NEW HEADER                                 A0255.1059
        S1      A1                                                              A0255.1060
        PUT,S1  S6&S7,MPSIZE,A2                                                 A0255.1061
        PUT,S1  S6&S7,MPSIZE,A6                                                 A0255.1062
        S1      0                                                               A0255.1063
        PUT,S1  S6&S7,MPST,A2                                                   A0255.1064
        PUT,S1  S6&S7,MPST,A6                                                   A0255.1065
        PUT,S2  S6&S7,MPID,A2                                                   A0255.1066
        PUT,S2  S6&S7,MPID,A6                                                   A0255.1067
        UNLOCK                                                                  C0983.30
```

```
*                                                                    A0255.1071
*       CLEAR THE ALLOCATED AREA                                     A0255.1072
*                                                                    A0255.1073
        A2      A2-1                                                 A0255.1074
        A7      A4+1                                                 A0255.1075
        A4      A4+1                                                 A0255.1076
MEMAL40 =       *                                                    A0255.1077
        0,A4    S1                                                   A0255.1078
        A4      A4+1                                                 A0255.1079
        A0      A2-A4                                                A0255.1080
        JAN     MEMAL40                                              A0255.1081
        A6      0                                                    A0255.1082
        J       B00             RETURN GOOD STATUS                   A0255.1083
MEMAL50 =       *                                                    A0255.1084
        GET,S1  SS&S7,MPSIZE,A4  SIZE OF CURRENT AREA                A0255.1085
        A2      S1                                                   A0255.1086
        A4      A4+A2           ADDRESS NEXT HEADER WORD             A0255.1087
        GET,S1  SS&S7,PTBASE,A5  POOL BASE ADDRESS                   A0255.1088
        GET,S2  SS&S7,PTSIZE,A5  POOL SIZE                           A0255.1089
        S1      S1+S2           END ADDRESS OF POOL                  A0255.1090
        A1      S1                                                   A0255.1091
        A0      A1-A4                                                A0255.1092
        JAN     MEMAL10         JUMP IN ENTRIE POOL NOT YET  SEARCHED A0255.1093
*                                                                    A0255.1094
*       THE MEMORY  POOL DOES NOT CONTAIN ENOUGH AVAILABLE CONSECUTIVE A0255.1095
*       SPACE TO FILL THE REQUEST                                    A0255.1096
*                                                                    A0255.1097
        UNLOCK                  RELEASE INTERRUPT LOCK-OUT           C0763.1
        A6      MISTAKE3                                             A0255.1098
        J       B00                                                  A0255.1099
MEMER1  =       *                                                    A0255.1100
        A6      MISTAKE1                                             A0255.1101
        J       B00                                                  A0255.1102
MEMER2  =       *                                                    A0255.1103
        A6      MISTAKE2                                             A0255.1104
        J       B00                                                  A0255.1105
```

- MEMDE RETURNS (DEALLOCATES) MEMORY TO THE MEMORY POOL FOR REALLOCATION. MEMDE IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING:

INPUT REGISTERS:
(A6) = Memory pool number
(A7) = Address of first usable word of memory to be deallocated

OUTPUT REGISTERS:
(A6) = Status:
0 Good return
1 Invalid address
2 Area not currently allocated
3 Invalid pool number

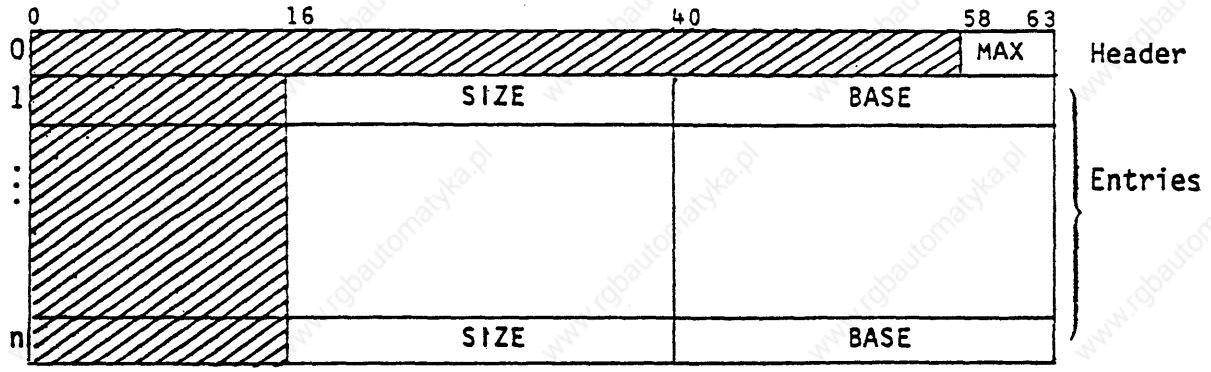
(A7) = Address of memory released; meaningful only if status is 0.

```
          EJECT
*************************************************************************
**
*
*NAME     MEMDE
*
*PURPOSE MEMDE RETURNS MEMORY TO THE POOL COMBINING  IT WITH AVAILABLE
*         ADJACENT AREAS TO HELP PREVENT FRAGMENTATION
*
*ENTRY    A6 = POOL NUMBER MEMORY SHOULD BE RETURNED TO
*         A7 = ADDRESS OF THE MEMORY TO BE RELEASED
*
*EXIT     A6 = STATUS
*                   0 = GOOD RETURN
*                   1 = INVALID HEADER WORD
*                   2 = AREA NOT CURRENTLY ALLLOCATED
*                   3 = INVALID POOL NUMBER
*         A7 = ADDRESS OF AREA RELEASED--MEANINGFUL ONLY IF GOOD STATUS
*
**
*************************************************************************
DEERR1    =         1
DEERR2    =         2
DEERR3    =         3
MEMDE     =         *
          A1        A7-1
          GET,S1    S6&S7,MPID,A1
          S2        IDWRD,A6
          S0        S2-S1
          JSN       DEER1           JUMP IF INVALID HEADER WORD
          GET,S0    S6&S7,MPST,A1
          JSZ       DEER2           JUMP IF PREVIOUSLY RELEASED
          A5        POOLTBL         BASE ADDRESS OF POOL TABLE
          GET,S1    S5&S6,PTMAX,A5 MAX VALID POOL NUMBER
          S2        A6
          S0        S1-S2
          JSM       DEER3           JUMP IF INVALID POOL
          A5        A5+A6           ADDRESS OF POOL WORD
          S0        0,A5
          JSZ       DEER3           JUMP IF INVALID POOL
          GET,S1    S6&S7,PTBASE,A5
          GET,S2    S6&S7,PTSIZE,A5  POOL SIZE
          S2        S2+S1           END ADDRESS OF POOL
          S3        A1              ADDRESS OF AREA TO RELEASE
          S0        S3-S1
          JSM       DEER3           JUMP IF ADDRESS OUTSIDE POOL RANGE
          S0        S2-S3
          JSM       DEER3           JUMP IF ADDRESS OUTSIDE POOL RANGE
*
*         THE VALIDATION IS COMPLETE.  NOW ACTUALLY RELEASE THE SPACE
*
          S6        1
          STPLK,A0  S6              LOCK OUT INTERRUPTS
          GET,S4    S6&S7,MPSIZE,A1  SIZE OF AREA
          S5        S3+S4
          A2        S5
          A2        A2-1            TRAILER ADDRESS
          S6        0
          PUT,S6    S0&S7,MPST,A1  MARK IT AVAILABLE
          PUT,S6    S0&S7,MPST,A2
          S0        S5-S2
          JSP       MEMDE10         JUMP IF LAST AREA IN POOL
          A4        S5              ADDRESS OF NEXT HEADER
          GET,S0    S6&S7,MPST,A4
          JSN       MEMDE10         JUMP IF NEXT AREA NOT AVIALABLE
```

```
*                                                                    A0255.1171
*       THE FOLLOWING MEMORY AREA IS AVAILABLE. COMBINE THE TWO INTO A0255.1172
*       ONE LARGE  AREA                                              A0255.1173
*                                                                    A0255.   4
        GET,S5    S6&S7,MPSIZE,A4 SIZE OF FOLLOWING AREA             A0255.1175
        S4        S5+S4           COMBINED SIZE                      A0255.1176
        A5        S4                                                 A0255.1177
        A2        A1+A5                                              A0255.1178
        A2        A2-1            TRAILER ADRESS                     A0255.1179
MEMDE10 =         *                                                  A0255.1180
        S0        S1-S3                                              A0255.1181
        JSZ       MEMDE20         JUMP IF AREA IS FIRST IN POOL      A0255.1182
        A4        A1-1            ADDRESS OF PRECEEDING TRAILER      A0255.1183
        GET,S0    S6&S7,MPST,A4                                      A0255.1184
        JSN       MEMDE20         JUMP IF PRECEEDING AREA IN USE     A0255.1185
*                                                                    A0255.1186
*       THE PRECEEDING AREA IS AVAILABLE. COMBINE THE TWO INTO       A0255.1187
*       ONE LARGE AREA                                               A0255.1188
*                                                                    A0255.1189
        GET,S5    S6&S7,MPSIZE,A4   SIZE OF PRECEEDING AREA          A0255.1190
        S4        S4+S5           COMBINED SIZE                      A0255.1191
        A4        S5                                                 A0255.1192
        A1        A1-A4           BASE ADDRESS OF LARGEST AREA       A0255.1193
*                                                                    A0255.1194
*       CREATE NEW HEADER AND TRAILER WORDS                          A0255.1195
*                                                                    A0255.1196
MEMDE20 =         *                                                  A0255.1197
        PUT,S4    S6&S7,MPSIZE,A1                                    A0255.1198
        PUT,S4    S6&S7,MPSIZE,A2                                    A0255.1199
        A6        0               GOOD STATUS                        A0255.1200
        UNLOCK                                                       C0983.31
        J         B00                                                A0255.   2
DEER1   =         *                                                  A0255.1203
        A6        DEERR1                                             A0255.1204
        J         B00                                                A0255.1205
DEER2   =         *                                                  A0255.1206
        A6        DEERR2                                             A0255.1207
        J         B00                                                A0255.1208
DEER3   =         *                                                  A0255.1209
        A6        DEERR3                                             A0255.1210
        J         B00                                                A0255.1211
```

POOL TABLE - POOLTBL

The Pool Table is an STP-resident table used for memory pool management.



Pool Table

HEADER

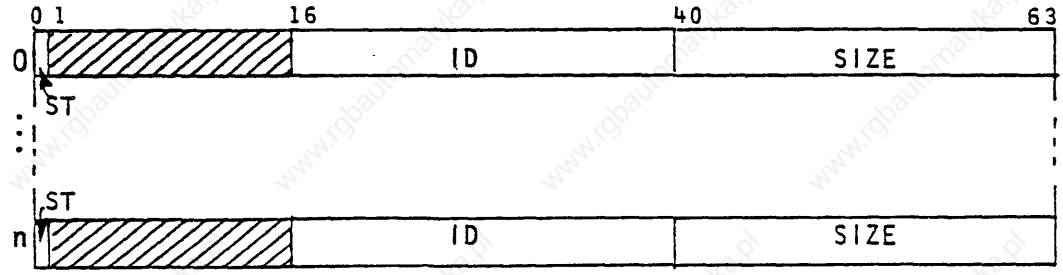
Field	Word	Bits	Description
PTMAX	0	58-63	Maximum valid memory pool number in system

ENTRY

Field	Word	Bits	Description
PTSIZE	0	16-39	Size of the memory pool
PTBASE	0	40-63	Base address of the memory pool

MEMORY POOL - POOLI....n

Memory pool areas are surrounded by header and trailer words that control the allocation and deallocation of the areas.



Memory Pool

Field	Word	Bits	Description
MPST	0,n,etc.	0	Status of the memory area: 0 Available 1 In use
MPID	0,n,etc.	16-39	Memory pool identification: 01010101 Pool 1 0x0x0x0x Pool x
MPSIZE	0,n,etc.	40-63	Size of the memory pool

ITEM CHAINING/UNCHAINING

- PROVIDES MEANS FOR TASKS TO LINK DATA
- AMOUNT OF DATA TO LINK IS DEFINED BY THE TASKS
- MAY BE USED TO LINK REGISTER DATA OR POOL DATA
- DATA IS CONSIDERED AN ITEM

- CHAIN/CHAINF PLACE AN ITEM ON A CHAIN. CHAIN WILL PLACE AN ITEM ON THE END WHEREAS CHAINF WILL PLACE AN ITEM ON THE FRONT OF A CHAIN. CHAIN/CHAINF ARE CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

INPUT REGISTERS:	(A6) = Address of chain control word
	(A7) = Address of the item to be chained
OUTPUT REGISTERS:	(A6) = Unchanged from input
	(A7) = Unchanged from input

```
          EJECT                                                                 A0255.1212
**********************************************************************          A0255.1213
**                                                                              A0255.1214
*                                                                               A0255.1215
*NAME     CHAIN                                                                A0255.1216
*                                                                               A0255.1217
*PURPOSE CHAIN WILL ADD AN ITEM TO AN EXISTING QUEUE.  ITEMS ARE                A0255.1218
*        ALWAYS ADDED AT THE END OF THE QUEUE.                                  A0255.1219
*                                                                               A0255.1220
*ENTRY   A6 = ADDRESS OF CHAIN CONTROL WORD                                     A0255.1221
*        A7 = ADDRESS OF THE ITEM TO ADD TO THE CHAIN                           A0255.1222
*                                                                               A0255.1223
*EXIT    A6 = UNCHANGED                                                         A0255.1224
*        A7 = UNCHANGED                                                         A0255.1225
*                                                                               A0255.1226
**                                                                              A0255.1227
**********************************************************************          A0255.1228
CHAIN    =          *                                                          A0255.1229
         S1         0                                                          A0255.1230
         PUT,S1     SB&S7,CIFL,A7                                              A0255.1231
         PUT,S1     SB&S7,CIBL,A7  JUST TO BE SAFE                             A0255.1232
         S1         A6                                                         A0255.1233
         PUT,S1     SB&S7,CICC,A7  SAVE CHAIN CONTROL ADDRESS                  A0255.1234
         GET,S1     SB&S7,CCTAIL,A6                                            A0255.1235
         S2         A7                                                         A0255.1236
         PUT,S2     SB&S7,CCTAIL,A6   NEW TAIL ADDRESS                         A0255.1237
         PUT,S1     SB&S7,CIBL,A7  BACKWAR LINK = OLD TAIL                     A0255.1238
         S0         S1                                                         A0255.1239
         JSN        CHAIN10                                                    A0255.1240
*                                                                               A0255.1241
*        THE CHAIN WAS EMPTY.  THIS IS THE ONLY ITEM ON QUEUE                   A0255.1242
*                                                                               A0255.1243
         PUT,S2     SB&S7,CCHEAD,A6  HEAD = TAIL                               A0255.1244
         J          CHAIN20                                                    A0255.1245
CHAIN10  =          *                                                          A0255.1246
         A5         S1                                                         A0255.1247
         PUT,S2     SB&S7,CIFL,A5  FORWARD LINK OLD TAIL = NEW TAIL            A0255.1248
CHAIN20  =          *                                                          A0255.1249
         J          B00                                                        A0255.1250
```

```
**********************************************************************  B0492.124
**                                                                      B0492.125
*NAME    CHAINF                                                        B0492.126
*                                                                      B0492.127
*PURPOSE CHAINF WILL ADD AN ITEM TO THE FRONT OF AN EXISTING QUEUE.    B0492.128
*                                                                      B0492.129
*        ENTRY     A6 = ADDRESS OF CHAIN CONTROL WORD                  B0492.130
*                  A7 = ADDRESS OF THE ITEM TO BE ADDED                B0492.131
*                                                                      B0492.132
*EXIT                                                                  B0492.133
*                  A6 = UNCHANGED                                      B0492.134
*                  A7 = UNCHANGED                                      B0492.135
*                                                                      B0492.136
*REGISTERS MODIFIED - S0,S1,S2,S6,S7,A5                               B0492.137
**                                                                     B0492.138
**********************************************************************  B0492.139
CHAINF   =         *                                                   B0492.140
         S1        1                                                   B0492.141
         STPLK,0   S1              LOCK OUT INTERRUPTS                 B0492.142
         GET.S2    S6&S7,CCHEAD,A6                                     B0492.143
         S0        S2                                                  B0492.144
         S1        A7                                                  B0492.145
         JSZ       CHAINF1                                             B0492.146
         SPUT,S1   S6&S7,CCHEAD,A6                                     B0492.147
         A5        S2                                                  B0492.148
         PUT,S1    S6&S7,CIBL,A5                                       B0492.149
         J         CHAINF2                                             B0492.150
CHAINF1  =         *                                                   B0492.151
         SET,S1    S6&S7,CCHEAD                                        B0492.152
         SET,S1    S6&S7,CCTAIL                                        B0492.153
         W@CCHEAD,A6 S6                                                B0492.154
CHAINF2  =         *                                                   B0492.155
         S1        A6                                                  B0492.156
         PUT,S1    S6&S7,CICC,A7  CHAIN CONTROL POINTER                B0492.157
         S6        W@CIFL,A7                                           B0492.158
         SET,S2    S6&S7,CIFL                                          B0492.159
         S1        0                                                   B0492.160
         SET,S1    S6&S7,CIBL                                          B0492.161
         W@CIFL,A7 S6                                                  B0492.162
         UNLOCK                                                        C0983.32
         J         B0                                                  B0492.164
         ERRIF     W@CIFL,NE,W@CIBL                                    B0492.165
         ERRIF     W@CCHEAD,NE,W@CCTAIL                                B0492.166
```

- UNCHAIN REMOVES AN ITEM FROM ANYWHERE ON THE CHAIN. THE CALLER MUST UPDATE THE COUNT OF THE NUMBER OF ITEMS REMAINING ON THE CHAIN. UNCHAIN IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

INPUT REGISTER:	(A7) = Address of item to be unchained
OUTPUT REGISTER:	(A7) = Unchanged from input

```
          EJECT                                                                 A0255.1251
**********************************************************************          A0255.1252
**                                                                              A0255.1253
*                                                                               A0255.1254
*NAME     UNCHAIN                                                              A0255.1255
*                                                                               A0255.1256
*PURPOSE UNCHAIN WILL REMOVE AN ITEM FROM AN EXISTING QUEUE. THE ITEM           A0255.1257
*        TO BE REMOVED MAY OCCUPY ANY POSITION IN THE CHAIN.                    A0255.1258
*                                                                               A0255.1259
*ENTRY   A7 = ADDRESS OF ITEM TO BE REMOVED                                     A0255.1260
*                                                                               A0255.1261
*EXIT    A0 = STATUS                                                           A0255.1262
*                 0 = GOOD RETURN                                              A0255.1263
*                 1 = ITEM IS NOT ON A CHAIN                                   A0255.1264
*        A7 = ADDRESS OF ITEM REMOVED                                           A0255.1265
*                                                                               A0255.1266
**                                                                              A0255.1267
**********************************************************************          A0255.1268
UNCHER1  =        1                                                            A0255.1269
UNCHAIN  =        *                                                            A0255.1270
         GET,S1   S6&S7,CICC,A7  CHAIN CONTROL WORD ADDRESS                    A0255.1271
         S0       S1                                                           A0255.1272
         A6       S1                                                           A0255.1273
         A0       UNCHER1                                                      A0255.1274
         JSZ      UNCH50         JUMP IF ITEM NOT ON A CHAIN                   A0255.1275
         S1       1                                                            A0255.1276
         STPLK,A0 S1             LOCK OUT INTERRUPTS                          A0255.1277
         GET,S1   S6&S7,CCHEAD,A6  HEAD OF CHAIN                              A0255.1278
         S2       A7             ADDRESS OF ITEM TO REMOVE                    A0255.1279
         GET,S3   S6&S7,CIFL,A7  FORWARD LINK OF ITEM                          A0255.1280
         GET,S4   S6&S7,CIBL,A7  BACKWARD LINK OF ITEM                         A0255.1281
         S0       S1-S2                                                        A0255.1282
         JSN      UNCH10         JUMP IF ITEM NOT HEAD OF CHAIN                A0255.1283
         PUT,S3   S6&S7,CCHEAD,A6  ITEM FOWARD LINK = NEW HEAD                 A0255.1284
UNCH10   =        *                                                            A0255.1285
         GET,S1   S6&S7,CCTAIL,A6  TAIL OF CHIAN                              A0255.1286
         S0       S1-S2                                                        A0255.1287
         JSN      UNCH20         JUMP IF ITEM NOT TAIL OF CHIAN                A0255.1288
         PUT,S4   S6&S7,CCTAIL,A6  ITEM BACKWARD LINK = NEW TAIL               A0255.1289
UNCH20   =        *                                                            A0255.1290
         S0       S3                                                           A0255.1291
         JSZ      UNCH30         JUMP IF FOWAD LINK IS ZERO                    A0255.1292
         A2       S3                                                           A0255.1293
         PUT,S4   S6&S7,CIBL,A2  NEW BACKWARD LINK FOR FOLLOWING ITEM          A0255.1294
UNCH30   =        *                                                            A0255.1295
         S0       S4                                                           A0255.1296
         JSZ      UNCH40         JUMP IF BACKWARD LINK IS ZERO                 A0255.1297
         A2       S4                                                           A0255.1298
         PUT,S3   S6&S7,CIFL,A2  NEW FORWARD LINK FOR PRECEEDING ITEM          A0255.1299
UNCH40   =        *                                                            A0255.1300
         UNLOCK                                                                C0983.33
         PUT,S1   S6&S7,CICC,A7  CLEAR CHAIN CONTROL                           A0255.1303
         A0       0              GOOD STATUS                                   C0983.34
UNCH50   =        *                                                            A0255.1305
         J        B00                                                          A0255.1306
```

CHAIN CONTROL - CC

Intertask communication requires chain control words in the format defined.

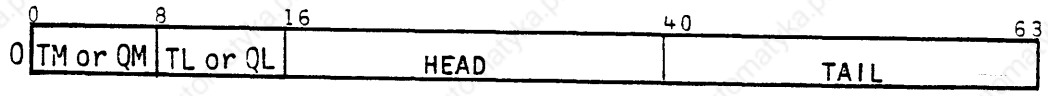
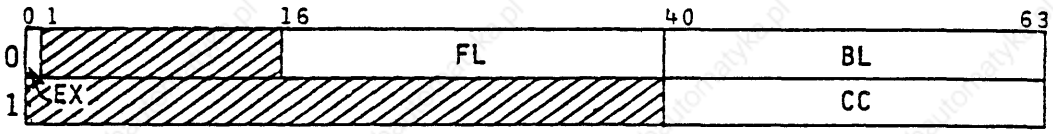


Figure 1.CC-1. Chain Control Word

Field	Word	Bits	Description
CCTM	0	0-7	Maximum number of items to be queued to a particular task
CCQM	0	0-7	Maximum number of items to be queued from one task to another
CCTL	0	8-15	Number of items queued to a particular task
CCQL	0	8-15	Number of items to be queued from one task to another
CCHEAD	0	16-39	Address of first item on the chain
CCTAIL	0	40-63	Address of last item on the chain

CHAIN ITEM - CI

Any item queued using the STP common routines CHAIN and UNCHAIN must reserve the first two words of the item to be used by the common routines.



Chain Item

Field	Word	Bits	Description
CIEX	0	0	This bit, if set, indicates that the item is in execution.
CIFL	0	16-39	Forward link; address of next item on the chain
CIBL	0	40-63	Backward link; address of the preceding item on the chain
CICC	1	40-63	Address of the chain control word for this item

TASK TO TASK COMMUNICATION

- THERE ARE 2 AREAS FOR INTERTASK COMMUNICATION

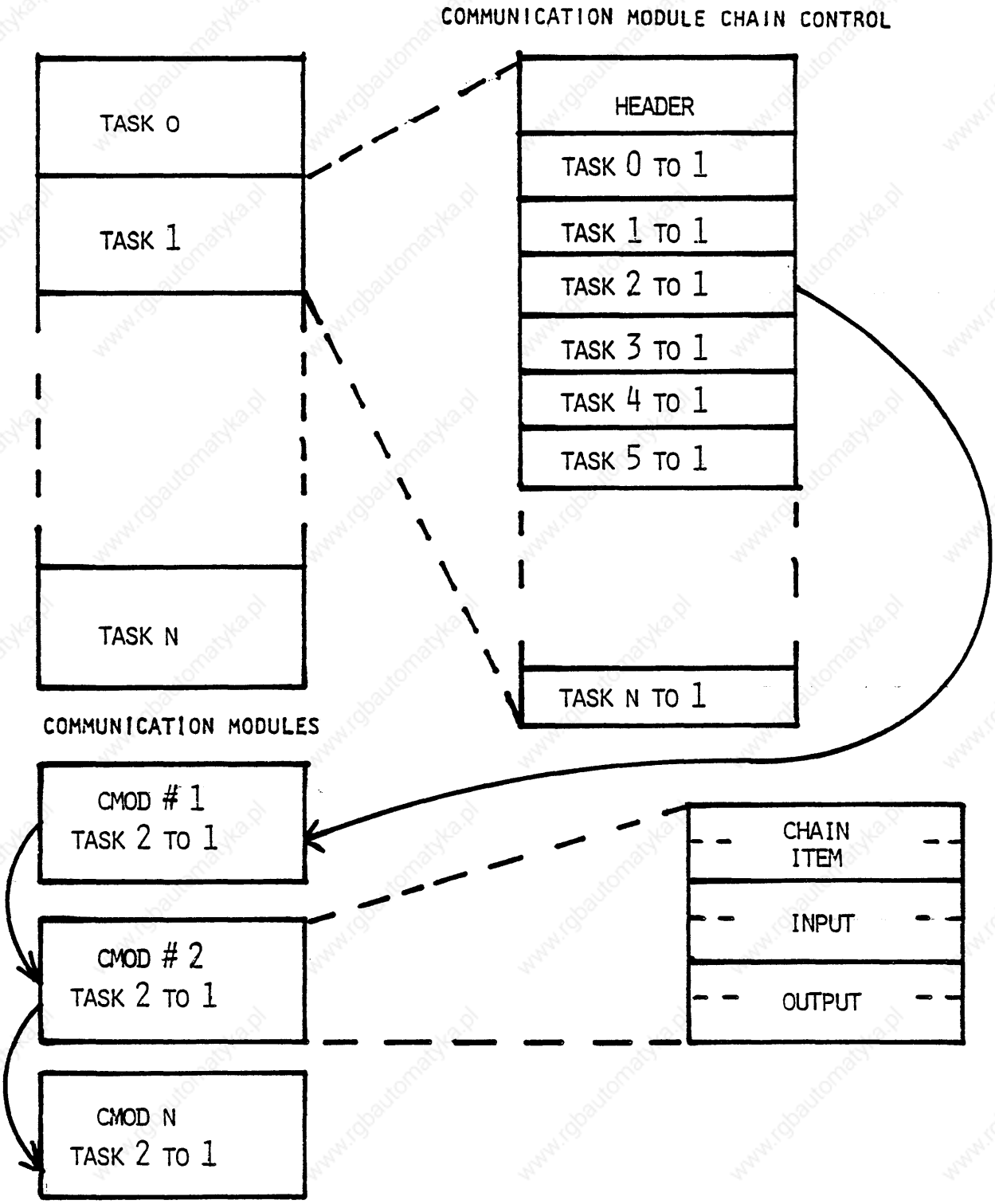
1. COMMUNICATION MODULE CHAIN CONTROL (CMCC).

   CONTIGUOUS AREA
   ENTRY FOR EACH POSSIBLE TASK COMBINATION
   ARRANGED IN TASK NUMBER SEQUENCE
   POINT TO THE COMMUNICATION MODULES (CMOD's)

2. COMMUNICATION MODULE (CMOD)

   ALLOCATED AS NEEDED FROM A POOL
   ALL TASK REQUESTS ARE THROUGH A CMOD.
   ALL TASK REPLIES ARE THROUGH A CMOD.
   - 2 WORDS FOR SYSTEM CONTROL
   - 2 WORDS AS TASK INPUT REGISTERS
   - 2 WORDS AS TASK OUTPUT REGISTERS

- TASKS PLACE REQUESTS IN THE INPUT WORDS OF A CMOD.
- TASKS RECEIVE REPLIES IN THE OUTPUT WORDS OF THEIR CMOD
- FORMAT OF A REQUEST IS DEFINED BY THE CALLED TASK


COMMUNICATION MODULE CHAIN CONTROL
TASK 0
TASK 1
TASK N
HEADER
TASK 0 TO 1
TASK 1 TO 1
TASK 2 TO 1
TASK 3 TO 1
TASK 4 TO 1
TASK 5 TO 1
TASK N TO 1
COMMUNICATION MODULES
CMOD # 1
TASK 2 TO 1
CMOD # 2
TASK 2 TO 1
CMOD N
TASK 2 TO 1
CHAIN ITEM
INPUT
OUTPUT

- A TASKS CALLS EXEC TO ACTIVATE ANOTHER TASK.

- THE TASK SCHEDULER IN EXEC EXAMINES THE SYSTEM TASK TABLE TO DETERMINE THE HIGHEST PRIORITY TASK READY TO EXECUTE.

- THE RE-ENTRANT ROUTINES;
  PUTREQ
  GETREQ
  PUTREPLY
  GETREPLY
  TSKREQ

  ARE USED FOR INTERTASK COMMUNICATION.

- THE REQUEST FOR INTERTASK COMMUNICATION IS PASSED IN REGISTERS S1 AND S2.

- PUTREQ PLACES THE REQUEST IN THE INPUT REGISTERS OF A CMOD AND LINKS THE CMOD TO THE APPROPRIATE CMCC. PUTREQ IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

INPUT REGISTERS:

(A1) = "Throw-away" indicator. If (A1) is positive, control is not returned to caller until request is queued. If (A1) is negative, control returns with no action taken if the request cannot be queued without suspending the caller.

(A2) = Requested task's ID

(S1) = INPUT+0
(S2) = INPUT+1
} request

OUTPUT REGISTERS: None

```
          EJECT
*************************************************************************
**
*
*NAME     PUTREQ
*
*PURPOSE PUTREQ QUEUES A REQUEST FOR PROCESSING TO ANOTHER TASK.
*         ONCE THE REQUEST HAS BEEN CHAINED PUTREQ RETURNS TO THE
*         THE ORIGINATOR
*
*ENTRY    A1 = THROW AWAY INDICATOR
*                   A1 = NEGATIVE THROW AWAY REQUEST INSTEAD OF SUSP
*                   A1 = POSITIVE SUSPEND UNTIL REQUEST QUEUED
*         A2 = DESTINATION TASK ID
*         S1 = INPUT+0
*         S2 = INPUT+1
*
*EXIT
*
**
*************************************************************************
PUTREQ    =         *
          A6        LE@SAVER
          A7        CTID,A0
          A6        A6*A7
          A7        SAVER
          A6        A6+A7          ADDRESS OF SAVE AREA FOR ORIGIN TASK
          SAVEA1,A6 A1
          SAVEA2,A6 A2
          SAVEA3,A6 A3
          SAVEA4,A6 A4
          SAVEA5,A6 A5
          SAVES0,A6 S0
          SAVES1,A6 S1
          SAVES2,A6 S2
          SAVES3,A6 S3
          SAVES4,A6 S4
          SAVES5,A6 S5
          SAVES6,A6 S6
          SAVES7,A6 S7
          A7        B00
          SAVERT,A6 A7
          S6        0
          SAVEA6,A6 S6
          SAVEA7,A6 S6
          A3        A6
          S0        S1
          ERRSZ                    ERROR IF INPUT&1 = 0
PUTREQ10  =         *
          A6        CMPOOL         MEMORY POOL NUMBER
          A7        CMSIZE         CMOD SIZE
          R         MEMAL          ALLOCATE A CMOD
          A5        SAVERT,A3      RESTORE B00 FOR TRACE ENTRIES
          B00       A5
          A0        A6
          JAZ       PUTREQ20       JUMP IF GOOD STATUS RETURNED
          A7        MISTAKE3
          A0        A6-A7
          ERRAN
```

```
*                                                                    A0255.436
*       MEMAL RETURNED A THREE STATUS--THERE IS NO MEMORY AVAILABLE. A0255.437
*       SUSPEND AND TRY AGAIN LATER                                  A0255.438
*                                                                    A0255.439
        A0      SAVEA1,A3                                            A0255.440
        JAM     PUTREQ50        THROW AWAY THE REQUEST               A0255.4
        S7      SUSP                                                 A0255.
        EX                                                           A0255.443
        R       ERROR0                                               A0255.444
        J       PUTREQ10                                             A0255.445
PUTREQ20 =      *                                                    A0255.446
        S1      SAVES1,A3                                            A0255.447
        S2      SAVES2,A3                                            A0255.448
        CMIN0,A7 S1                                                  A0255.449
        CMIN1,A7 S2             PUT THE MESSAGE IN THE CMOD          A0255.450
        POST    12,S1,S2                                             A0255.451
*                                                                    A0255.452
*       CALCULATE ADDRESS OF CHAIN CONTROL HEADER                    A0255.453
*       CHAIN CONTROL HEADER = DESTINATION TASK ID * LENGTH OF CHAIN  A0255.454
*                              CONTROL ENTRY + CHAIN CONTROL BASE    A0255.455
*                                                                    A0255.456
        A6      SAVEA2,A3                                            A0255.457
        A5      LE@CMCC                                              A0255.458
        A5      A5*A6                                                A0255.459
        A6      CMCC                                                 A0255.460
        A5      A5+A6           CHAIN CONTROL HEADER ADDRESS         A0255.461
*                                                                    A0255.462
*       CALCULATE THE CHAIN CONTROL WORD ADDRESS                     A0255.463
*       CHAIN CONTROL = CHAIN CONTROL HEADER + LENGTH OF HEADER +     A0255.464
*                       CURRENT TASK ID                              A0255.465
*                                                                    A0255.466
        A6      LH@CMCC                                              A0255.467
        A6      A6+A5                                                A0255.468
        A4      CTID,A0                                              A0255.4
        A6      A6+A4           CHAIN CONTROL ADDRESS                A0255.4
PUTREQ21 =      *                                                    A0255.471
        A1      1                                                    A0255.472
        STPLK,A0 A1             LOCK OUT INTERRUPTS                  A0255.473
        GET,S1  SS&S7,CCTL,A5                                        A0255.474
        GET,S2  SS&S7,CCTM,A5                                        A0255.475
        S0      S2                                                   A0255.476
        ERRSZ                   ERROR IF QUEUE MAX IS ZERO           A0255.477
        S0      S1-S2                                                A0255.478
        JSZ     PUTREQ22        JUMP IF QUEUE AT MAX                 A0255.479
        S4      1                                                    A0255.480
        S1      S4+S1                                                A0255.481
        GET,S2  S6&S7,CCQL,A6                                        A0255.482
        GET,S3  S6&S7,CCQM,A6                                        A0255.483
        S0      S3                                                   A0255.484
        ERRSZ                                                        A0255.485
        S0      S2-S3                                                A0255.486
        JSZ     PUTREQ22        JUMP IF QUEUE AT MAX                 A0255.487
        S2      S2+S4                                                A0255.488
*       UPDATE THE QUEUE COUNTS                                      A0255.489
        PUT,S1  SS&S7,CCTL,A5                                        A0255.490
        PUT,S2  SS&S7,CCQL,A6                                        A0255.491
        J       PUTREQ25                                             A0255.492
*                                                                    A0255.493
*       THE QUEUE IS AT ITS MAX.  EITHER SUSPEND OR THROW THE REQUEST A0255.494
*       AWAY                                                         A0255.495
*                                                                    A0255.496
PUTREQ22 =      *                                                    A0255.497
        A1        0                                                  A0255.4
        STPLK,A0  A1            RELEASE INTERRUPT LOCSOUT            A0255.4
        A0      SAVEA1,A3                                            A0255.500
        JAM     PUTREQ23        JUMP TO THOW THE REQUEST AWAY        A0255.501
        S7      SUSP                                                 A0255.502
        EX                                                           A0255.503
        R       ERROR0                                               A0255.504
        J       PUTREQ21                                             A0255.505
```

```
*                                                                    A0255.506
*       THROW THE REQUEST AWAY                                       A0255.507
*                                                                    A0255.508
PUTREQ23 =      *                                                    A0255.509
        A6      CMPOOL                                               A0255.510
        R       MEMDE           RELEASE THE CMOD                     A0255.511
        A5      SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.512
        B00     A5                                                   A0255.513
        J       PUTREQ50                                             A0255.514
PUTREQ25 =      *                                                    A0255.515
        R       CHAIN                                                A0255.516
        A5      SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.517
        B00     A5                                                   A0255.518
        A1      0                                                    A0255.519
        STPLK,A0 A1             RELEASE INTERRUPT LOCK-OUT           A0255.520
PUTREQ40 =      *                                                    A0255.521
        S6      SAVEA2,A3                                            A0255.522
        S7      RTSK            READY REQUESTED TASK                 A0255.523
        EX                                                           A0255.524
        R       ERROR0                                               A0255.525
*                                                                    A0255.526
*       RESTORE THE REGISTERS AND EXIT                               A0255.527
*                                                                    A0255.528
PUTREQ50 =      *                                                    A0255.529
        A6      A3                                                   A0255.530
        A0      SAVEA0,A6                                            A0255.531
        A1      SAVEA1,A6                                            A0255.532
        A2      SAVEA2,A6                                            A0255.533
        A3      SAVEA3,A6                                            A0255.534
        A4      SAVEA4,A6                                            A0255.535
        A5      SAVEA5,A6                                            A0255.536
        S0      SAVES0,A6                                            A0255.537
        S1      SAVES1,A6                                            A0255.538
        S2      SAVES2,A6                                            A0255.539
        S3      SAVES3,A6                                            A0255.540
        S4      SAVES4,A6                                            A0255.541
        S5      SAVES5,A6                                            A0255.542
        S6      SAVES6,A6                                            A0255.543
        S7      SAVES7,A6                                            A0255.544
        A7      SAVERT,A6                                            A0255.545
        B00     A7                                                   A0255.546
        A7      SAVEA7,A6                                            A0255.547
        A6      SAVEA6,A6                                            A0255.548
        J       B00                                                  A0255.549
```

- GETREQ SEARCHES FOR AN ACTIVE REQUEST FOR THE CALLER. GETREQ IS CALLED VIA A RETURN JUMP AND REPLIES WITH THE FOLLOWING:

INPUT REGISTERS: None

OUTPUT REGISTERS:

(A0) = "Found" indicator. If (A0) = 0, no outstanding requests exist. If (A0) ≠ 0, a request is being returned.

(A2) = ID of task that generated the request.

(S1) = INPUT+0 } request

(S2) = INPUT+1 }

```
          EJECT
*************************************************************************
**
*
*NAME     GETREQ
*
*PURPOSE GETREQ SEARCHES FOR AN OUTSTANDING REQUEST TO THE TASK.
*         THE HIGHEST PRIORITY REQUEST WILL BE RETURNED TO THE
*         ORIGINATING TASK.  PRIORITY CORESPONDS TO TASK ID
*
*ENTRY
*EXIT     A0 = FIND/NO FIND INDICATOR
*                  0 = NO FIND
*                  1 = FIND
*         A2 = TASK ID OF THE REQUEST LOCATED
*         S1 = INPUT+0
*         S2 = INPUT+1
*
**
*************************************************************************
GETREQ    =         *
          A6        LE@SAVER
          A7        CTID,A0
          A6        A6*A7
          A7        SAVER
          A6        A6+A7          ADDRESS OF SAVE AREA FOR ORIGIN TASK
*
*         SAVE THOSE REGISTERS NOT ALREADY DESTROYED
*
          SAVEA1,A6 A1
          SAVEA2,A6 A2
          SAVEA3,A6 A3
          SAVEA4,A6 A4
          SAVEA5,A6 A5
          SAVES0,A6 S0
          SAVES1,A6 S1
          SAVES2,A6 S2
          SAVES3,A6 S3
          SAVES4,A6 S4
          SAVES5,A6 S5
          SAVES6,A6 S6
          SAVES7,A6 S7
          A7        B00
          SAVERT,A6 A7
          S6        0
          SAVEA6,A6 S6
          SAVEA7,A6 S6
          SAVEA0,A6 S6             ASSUME NO FIND WILL OCCUR
          A3        A6
*
*
*         CALCULATE THE ADDRESS OF CHAIN CONTROL HEADER
*         CHAIN CONTROL HEADER = CURRENT TASK ID * CHAIN CONTROL LENGTH
*                                OF ENTRY + BASE OF CHAIN CONTROL
*
          A1        CTID,A0
          A2        LE@CMCC
          A1        A1*A2
          A2        CMCC
          A1        A1+A2          CHAIN CONTROL HEADER ADDRESS
          GET,S0    S6&S7,CCTL,A1
          JSZ       GETREQ30       JUMP IF THERE IS NOTHING ON QUEUE
```

```
*                                                                    A0255.612
*       SEARCH EACH CHAIN CONTROL WORD FOR AN OUTSTANDING REQUEST    A0255.613
*                                                                    A0255.614
        A2      LH@CMCC                                              A0255.615
        A1      A1+A2                                                A0255.616
        A2      0               WILL CONTAIN TASK ID OF FIND ON EXIT A0255.617
        A4      M4XTN                                                A0255.
GETREQ10 =      *                                                    A0255.619
        GET,S0  S6&S7,CCQL,A1                                        A0255.620
        JSN     GETREQ20        JUMP IF ITEMS ON CHAIN               A0255.621
GETREQ15 =      *                                                    A0255.622
        A1      A1+1            THIS CHAIN IS EMPTY GO TO THE NEXT   A0255.623
        A2      A2+1                                                 A0255.624
        A0      A4-A2                                                A0255.625
        JAN     GETREQ10        JUMP IF ALL CHAINS NOT EXAMINED      A0255.626
        J       GETREQ30        THERE ARE NO REQUESTS FOR THE TASK   A0255.627
*                                                                    A0255.628
*       A CHAIN WITH ITEMS HAS BEEN LOCATED                          A0255.629
*                                                                    A0255.630
GETREQ20 =      *                                                    A0255.631
        GET,S1  S6&S7,CCHEAD,A1                                      A0255.632
        GET,S2  S6&S7,CCTAIL,A1                                      A0255.633
GETREQ22 =      *                                                    A0255.634
        A5      S1              CMOD ADDRESS                         A0255.635
        GET,S0  S6&S7,CIEX,A5                                        A0255.636
        JSZ     GETREQ25        JUMP IF REQUEST IS NOT IN EXECUTION  A0255.637
*                                                                    A0255.638
*       THE TASK IS EXECUTING THIS ITEM                              A0255.639
*                                                                    A0255.640
        S0      S1-S2                                                A0255.641
        JSZ     GETREQ15        JUMP IF ALL ITEMS IN EXECUTION       A0255.642
        GET,S1  S6&S7,CIFL,A5   ADDRESS OF NEXT CMOD                 A0255.643
        J       GETREQ22                                             A0255.644
*                                                                    A0255.645
*       A REQUEST TO PROCESS HAS BEEN LOCATED                        A0255.
*                                                                    A0255.
GETREQ25 =      *                                                    A0255.648
        SAVEA2,A3 A2            SAVE TASK ID OF REQUEST              A0255.649
        S1      1                                                    A0255.650
        SAVEA0,A3 S1            SET FIND INDICATOR                   A0255.651
        PUT,S1  S6&S7,CIEX,A5   SET EXECUTE INDICATOR                A0255.652
        S1      CMIN0,A5                                             A0255.653
        S2      CMIN1,A5                                             A0255.654
        S0      S1                                                   B0492.9
        ERRSZ                   ERROR IF INPUT&1 = 0                 B0492.10
        SAVES1,A3 S1                                                 A0255.655
        SAVES2,A3 S2                                                 A0255.656
*                                                                    A0255.657
*       RESTORE REGISTERS AND EXIT.                                  A0255.658
*                                                                    A0255.659
GETREQ30 =      *                                                    A0255.660
        A6      A3                                                   A0255.661
        A0      SAVEA0,A6                                            A0255.662
        A1      SAVEA1,A6                                            A0255.663
        A2      SAVEA2,A6                                            A0255.664
        A3      SAVEA3,A6                                            A0255.665
        A4      SAVEA4,A6                                            A0255.666
        A5      SAVEA5,A6                                            A0255.667
        S0      SAVES0,A6                                            A0255.668
        S1      SAVES1,A6                                            A0255.669
        S2      SAVES2,A6                                            A0255.670
        S3      SAVES3,A6                                            A0255.671
        S4      SAVES4,A6                                            A0255.672
        S5      SAVES5,A6                                            A0255.
        S6      SAVES6,A6                                            A0255.674
        S7      SAVES7,A6                                            A0255.675
        A7      SAVERT,A6                                            A0255.676
        B00     A7                                                   A0255.677
        A7      SAVEA7,A6                                            A0255.678
        A6      SAVEA6,A5                                            A0255.679
        J       B00                                                  A0255.680
```

- PUTREPLY PLACES A REPLY IN THE APPROPRIATE CMOD WHEN A TASK COMPLETES PROCESSING. PUTREPLY IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

INPUT REGISTERS: (A2) = ID of task to receive the reply.
(S1) = OUTPUT+0
(S2) = OUTPUT+1
} Reply

OUTPUT REGISTERS: None

```
          EJECT
*****************************************************************************
**
*
*NAME     PUTREPLY
*
*PURPOSE PUTREPLY SAVES OUTPUT IN THE APPROPRIATE CMOD WHEN A TASK
*         COMPLETES PROCESSING
*
*ENTRY    A2 = ID OF THE TASK TO RECEIVE THE REPLY
*         S1 = OUTPUT+0
*         S2 = OUTPUT+1
*
*EXIT
*
**
*****************************************************************************
PUTREPLY =          *
          A6        LE@SAVER
          A7        CTID,A0          ORIGINATING TASK ID
          A6        A6*A7
          A7        SAVER
          A6        A6+A7            ADDRESS OF SAVE AREA FOR ORIGIN TASK
*
*         SAVE THE REGISTERS CLEARING THOSE ALREADY DESTROYED.
*
          SAVEA0,A6 A0
          SAVEA1,A6 A1
          SAVEA2,A6 A2
          SAVEA3,A6 A3
          SAVEA4,A6 A4
          SAVEA5,A6 A5
          SAVES0,A6 S0
          SAVES1,A6 S1
          SAVES2,A6 S2
          SAVES3,A6 S3
          SAVES4,A6 S4
          SAVES5,A6 S5
          SAVES6,A6 S6
          SAVES7,A6 S7
          A7        B00
          SAVERT,A6 A7
          S6        0
          SAVEA7,A6 S6
          SAVEA6,A6 S6
          A3        A6
          S0        S2
          ERRSZ                      ERROR IF 2ND WORD OF REPLY IS ZERO
*
*         CALCULATE CHAIN CONTROL HEADER AND CHAIN CONTROL WORD ADR
*         CHAIN CONTROL HEADER = BASE OF CMOD CHAIN CONTROL + (LENGTH
*                                OF AN ENTRY * CURRENT TASK ID)
*         CHAIN CONTROL WORD = CHAIN CONTROL HEADER + LENGTH OF HEADER
*                              + RECEIVING TASK ID
*
          A1        CTID,A0
          A6        LE@CMCC
          A6        A6*A1
          A1        CMCC
          A6        A6+A1            CHAIN CONTROL HEADER
          A7        LH@CMCC
          A7        A7+A6
          A7        A2+A7            CHAIN CONTROL WORD
          GET,S0    S6&S7,CCQL,A7
          ERRSZ
          GET,S5    S6&S7,CCHEAD,A7
          GET,S4    S6&S7,CCTAIL,A7
```

```
PUTREP10 =        *                                                    A0255.746
         A5       S5              CMOD ADDRESS                         A0255.747
         S0       CMOUT1,A5                                            A0255.748
         JSZ      PUTREP20        JUMP IF CMOD HAS NO REPLY            A0255.749
*                                                                      A0255.750
*        A REPLY EXISTS IN THIS CMOD. GO TO THE NEXT CMOD IF THERE IS  A0255.751
*        ONE                                                           A0255.752
*                                                                      A0255.753
         S0       S4-S5                                               A0255.754
         ERRSZ                                                        A0255.755
         GET,S5   S6&S7,CIFL,A5   NEXT CMOD ADDRESS                   A0255.756
         J        PUTREP10                                            A0255.757
*                                                                      A0255.758
*        AT THIS POINT WE HAVE FOUND A CMOD FOR THE REPLY              A0255.759
*                                                                      A0255.760
PUTREP20 =        *                                                   A0255.761
         A1       1                                                   A0255.762
         STPLK,A0 A1              SET INTERRUPT LOCK-OUT              A0255.763
         CMOUT0,A5 S1                                                 A0255.764
         CMOUT1,A5 S2                                                A0255.765
*                                                                      A0255.766
*        DECREMENT THE NUMBER OF ITEMS ON QUEUE                        A0255.767
*                                                                      A0255.768
         GET,S1   S6&S7,CCQL,A7                                       A0255.769
         S4       1                                                   A0255.770
         S1       S1-S4                                               A0255.771
         PUT,S1   S6&S7,CCQL,A7                                       A0255.772
         GET,S1   S6&S7,CCTL,A6                                       A0255.773
         S1       S1-S4                                               A0255.774
         PUT,S1   S6&S7,CCQL,A6                                       A0255.775
         A1       0                                                   A0255.776
         STPLK,A0 A1                                                  A0255.777
*                                                                      A0255.778
*        READY THE RECEIVING TASK                                      A0255.779
*                                                                      A0255.780
         S1       CMOUT0,A5                                           A0255.781
         POST     12,S1,S2                                            A0255.782
         S6       A2                                                  A0255.783
         S7       RTSK                                                A0255.784
         EX                                                           A0255.785
         R        ERROR0                                              A0255.786
*                                                                      A0255.787
*        RESTORE THE REGISTERS AND EXIT                                A0255.788
*                                                                      A0255.789
         A6       A3                                                  A0255.790
         A0       SAVEA0,A6                                           A0255.791
         A1       SAVEA1,A6                                           A0255.792
         A2       SAVEA2,A6                                           A0255.793
         A3       SAVEA3,A6                                           A0255.794
         A4       SAVEA4,A6                                           A0255.795
         A5       SAVEA5,A6                                           A0255.796
         S0       SAVES0,A6                                           A0255.797
         S1       SAVES1,A6                                           A0255.798
         S2       SAVES2,A6                                           A0255.799
         S3       SAVES3,A6                                           A0255.800
         S4       SAVES4,A6                                           A0255.801
         S5       SAVES5,A6                                           A0255.802
         S6       SAVES6,A6                                           A0255.803
         S7       SAVES7,A6                                           A0255.804
         A7       SAVERT,A6                                           A0255.805
         B00      A7                                                  A0255.806
         A7       SAVEA7,A6                                           A0255.807
         A6       SAVEA6,A6                                           A0255.808
         J        B00                                                 A0255.809
```

- GETREPLY SEARCHES FOR A REPLY TO THE CALLING TASK. GETREPLY ALSO RELEASES THE APPROPRIATE CMOD WHEN A REPLY IS FOUND. GETREPLY IS CALLED VIA A RETURN JUMP AND REPLIES WITH THE FOLLOWING:

INPUT REGISTERS: None

OUTPUT REGISTERS:

(A0) = Find indicator. If (A0) = 0, no reply was located; if (A0) ≠ 0, a reply is being returned to the caller.

(A2) = ID of replying task

(S1) = OUTPUT+0 } Reply

(S2) = OUTPUT+1 }

```
         EJECT
*****************************************************************************
**
*
*NAME    GETREPLY
*
*PURPOSE GETREPLY LOCATES AND RETURNS THE HIGHEST PRIORITY REPLY TO
*        A TASK.  THE PRIORITY STRUCTURE CORRESPONDS TO THE TASK ID
*
*ENTRY
*
*EXIT    A0 = FIND/NO FIND INDICATOR
*                 0 = NO FIND
*                 1 = FIND
*        A2 = ID OF TASK WHERE REPLY FOUND
*        S1 = OUTPUT+0
*        S2 = OUTPUT+1
*
**
*****************************************************************************
GETREPLY =       *
         A6      LE@SAVER
         A7      CTID,A0
         A6      A6*A7
         A7      SAVER
         A6      A6+A7          ADDRESS OF SAVE AREA FOR ORIGIN TASK
*
*        SAVE THE REGISTERS CLEARING THOSE ALREADY DESTROYED
*
         SAVEA1,A6 A1
         SAVEA2,A6 A2
         SAVEA3,A6 A3
         SAVEA4,A6 A4
         SAVEA5,A6 A5
         SAVES0,A6 S0
         SAVES1,A6 S1
         SAVES2,A6 S2
         SAVES3,A6 S3
         SAVES4,A6 S4
         SAVES5,A6 S5
         SAVES6,A6 S6
         SAVES7,A6 S7
         A7      B00
         SAVERT,A6 A7
         S6      0
         SAVEA7,A6 S6
         SAVEA0,A6 S6            ASSUME NO FIND WILL OCCUR
         SAVEA6,A6 S6
         A3      A6
*
*        CALCULATE CHAIN CONTROL ADDRESS
*        CHAIN CONTROL = CURRENT TASK ID + CHAIN CONTROL BASE + LENGTH
*                        OF CHAIN CONTROL HEADER
*
         A6      CMCC
         A1      CTID,A0
         A6      A6+A1
         A5      LH@CMCC
         A6      A6+A5
         A2      0
         A4      MAXTN
         A5      LE@CMCC
```

```
GETREP10 =        *                                                    A0255.872
         GET,S1   S5&S7,CCHEAD,A6                                      A0255.873
         S0       S1                                                   A0255.874
         JSZ      GETREP20                                             A0255.875
         A7       S1              ADDRESS OF THE CMOD                  A0255.876
         S0       CMOUT1,A7                                            A0255.
         JSN      GETREP30        JUMP IF A REPLY EXISTS               A0255.
*                                                                      A0255.879
*        THERE IS NO REPLY FROM THIS TASK.  UPDATE TO NEXT TASK        A0255.880
*                                                                      A0255.881
GETREP20 =        *                                                    A0255.882
         A6       A6+A5           NEXT CHAIN CONTROL                   A0255.883
         A2       A2+1            NEXT TASK ID                         A0255.884
         A0       A4-A2                                                A0255.885
         JAN      GETREP10        JUMP IF ALL CHAINS NOT EXAMINED      A0255.886
*                                                                      A0255.887
*        AT THIS POINT THE SEARCH IS COMPLETE WITH NO REPLY FOUND      A0255.888
*                                                                      A0255.889
         J        GETREP40                                             A0255.890
GETREP30 =        *                                                    A0255.891
         SAVEA2,A3 A2             SAVE TASK ID OF FIND                 A0255.892
         A0       1                                                    A0255.893
         SAVEA0,A3 A0                                                  A0255.894
         S1       CMOUT0,A7                                           A0255.895
         S2       CMOUT1,A7                                           A0255.896
         SAVES1,A3 S1                                                  A0255.897
         SAVES2,A3 S2             MOVE THE REPLY OUT OF THE CMOD       A0255.898
         R        UNCHAIN         REMOVE IT FROM THE CHAIN             A0255.899
         A5       SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.900
         B00      A5                                                   A0255.901
         ERRAN                                                         A0255.902
         A6       CMPOOL          DEALLOCATE THE MEMORY                A0255.903
         R        MEMDE                                                A0255.904
         A5       SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.
         B00      A5                                                   A0255.
         A0       A6                                                   A0255.907
         ERRAN                                                         A0255.908
*                                                                      A0255.909
*        RESTORE THE REGISTERS AND EXIT                                A0255.910
*                                                                      A0255.911
GETREP40 =        *                                                    A0255.912
         A6       A3                                                   A0255.913
         A0       SAVEA0,A6                                            A0255.914
         A1       SAVEA1,A6                                            A0255.915
         A2       SAVEA2,A6                                            A0255.916
         A3       SAVEA3,A6                                            A0255.917
         A4       SAVEA4,A6                                            A0255.918
         A5       SAVEA5,A6                                            A0255.919
         S0       SAVES0,A6                                            A0255.920
         S1       SAVES1,A6                                            A0255.921
         S2       SAVES2,A6                                            A0255.922
         S3       SAVES3,A6                                            A0255.923
         S4       SAVES4,A6                                            A0255.924
         S5       SAVES5,A6                                            A0255.925
         S6       SAVES6,A6                                            A0255.926
         S7       SAVES7,A6                                            A0255.927
         A7       SAVERT,A6                                            A0255.928
         B00      A7                                                   A0255.929
         A7       SAVEA7,A6                                            A0255.930
         A6       SAVEA6,A6                                            A0255.931
         J        B00                                                  A0255.932
```

- TSKREQ QUEUES A REQUEST TO ANOTHER TASK.
- TSKREQ IS CALLED VIA A RETURN JUMP WITH THE CALLER PROVIDING THE FOLLOWING:

INPUT REGISTERS:	(A2) = ID OF REQUESTED TASK	
	(S1) = INPUT+0	} REQUEST
	(S2) = INPUT+1	
OUTPUT REGISTERS:	(S1) = OUTPUT+0	} REPLY
	(S2) = OUTPUT+1	

- ONCE THE REQUEST HAS BEEN PROCESSED, THE CALLER MAY EXAMINE ITS S1,S2 REGISTERS FOR A REPLY. CONVENTIONALLY, S1=ZERO WHEN THERE IS NO ERROR, OTHERWISE S1=ERR CODE. S2=THE CALLING TASKS INPUT+0 REGISTER (S1) INFORMATION.

```
          EJECT                                                                S003.79
*****************************************************************************
**
*NAME     TSKREQ
*
*PURPOSE  TSKREQ QUEUES A REQUEST FOR PROCESSING TO ANOTHER TASK. THE
*         ORIGINATING TASK IS SUSPENDED UNTIL THE DESTINATION TASK HAS
*         COMPLETED PROCESSING.
*
*ENTRY    A2 = DESTINATION TASK ID
*         S1 = INPUT+0
*         S2 = INPUT+1
*
*EXIT     S1 = OUTPUT+0
*         S2 = OUTPUT+1
*
**
*****************************************************************************
TSKREQ    =         *
          A3        LE@SAVER
          A4        CTID,A0
          A3        A3*A4
          A4        SAVER
          A3        A3+A4          ADDRESS OF SAVE AREA FOR ORIGIN TASK
*
*         SAVE THE REGISTERS AND CLEAR OUT THOSE THAT HAVE ALREADY
*         BEEN DESTROYED
*
          SAVEA0,A3 A0
          SAVEA1,A3 A1
          SAVEA2,A3 A2
          SAVEA5,A3 A5
          SAVEA6,A3 A6
          SAVEA7,A3 A7
          SAVES0,A3 S0
          SAVES1,A3 S1
          SAVES2,A3 S2
          SAVES3,A3 S3
          SAVES4,A3 S4
          SAVES5,A3 S5
          SAVES6,A3 S6
          SAVES7,A3 S7
          S6        0
          SAVEA3,A3 S6
          SAVEA4,A3 S6
          A4        B00
          SAVERT,A3 A4             SAVE THE RETURN ADDRESS
          S0        S1
          ERRSZ                    ERROR IF INPUT&1 = 0
TSKREQ10  =         *
          A6        CMPOOL         MEMORY POOL NUMBER
          A7        CMSIZE         NUMBER OF WORD REQUIRED
          R         MEMAL          ALLOCATE A CMOD
          A5        SAVERT,A3      RESTORE B00 FOR TRACE ENTRIES
          B00       A5
*
*         CHECK STATUS RETURNED FROM MEMAL
*
          A0        A6
          JAZ       TSKREQ20       JUMP IF GOOD RETURN
          A7        3
          A0        A6-A7
          ERRAN
```

```
*                                                                       A0255.251
*       AT THIS POINT THERE IS NO MEMORY AVAILABLE.  SUSPEND AND        A0255.252
*       TRY AGAIN LATER                                                 A0255.253
*                                                                       A0255.254
        S7      SUSP                                                    A0255.255
        EX                                                              A0255.256
        R       ERROR0                                                  A0255.257
        J       TSKREQ10                                                A0255.258
*                                                                       A0255.259
*       A CMOD HAS BEEN ALLOCATED  PUT THE INPUT MESSAGE IN THE CMOD    A0255.260
*                                                                       A0255.261
TSKREQ20 =      *                                                       A0255.262
        S1      SAVES1,A3                                               A0255.263
        S2      SAVES2,A3                                               A0255.264
        CMIN0,A7 S1                                                     A0255.265
        CMIN1,A7 S2                                                     A0255.266
        POST    12,S1,S2                                                A0255.267
*                                                                       A0255.268
*       CALCULATE THE ADDRESS OF THE CHAIN CONTROL HEADER               A0255.269
*       CHAIN CONTROL HEADER = DESTINATION TASK ID * LENGTH OF CHAIN    A0255.270
*                              CONTROL ENTRY + BASE OF CHAIN CONTROL    A0255.271
*                                                                       A0255.272
        A5      SAVEA2,A3                                               A0255.273
        A4      LE@CMCC                                                 A0255.274
        A5      A4*A5                                                   A0255.275
        A4      CMCC                                                    A0255.276
        A5      A4+A5           CHAIN CONTROL HEADER ADDRESS            A0255.277
*                                                                       A0255.278
*       CALCULATE CHAIN CONTROL WORD ADDRESS                            A0255.279
*       CHAIN CONTROL WORD = CHAIN CONTROL HEADER + LENGTH OF CHAIN     A0255.280
*                            CONTROL HEADER + CURRENT TASK ID           A0255.281
        A6      LH@CMCC                                                 A0255.282
        A6      A6+A5                                                   A0255.283
        A4      CTID,A0                                                 A0255.284
        A6      A6+A4           CHAIN CONTROL WORD ADDRESS              A0255.285
TSKREQ30 =      *                                                       A0255.286
        A1      1                                                       A0255.287
        STPLK,A0 A1             LOCK OUT INTERRUPTS                     A0255.288
        GET,S1  S6&S7,CCTL,A5                                           A0255.289
        GET,S2  S6&S7,CCTM,A5                                           A0255.290
        S0      S2                                                      A0255.291
        ERRSZ                   ERROR IF QUEUE MAX IS ZERO              A0255.292
        S0      S1-S2                                                   A0255.293
        JSZ     TSKREQ70        JUMP IF QUEUE AT LIMIT                  A0255.294
        S4      1                                                       A0255.295
        S1      S1+S4                                                   A0255.296
        GET,S2  S6&S7,CCQL,A6                                           A0255.297
        GET,S3  S6&S7,CCQM,A6                                           A0255.298
        S0      S3                                                      A0255.299
        ERRSZ                                                           A0255.300
        S0      S2-S3                                                   A0255.301
        JSZ     TSKREQ70        JUMP IF QUEUE AT LIMIT                  A0255.302
        S2      S2+S4                                                   A0255.303
        PUT,S1  S6&S7,CCTL,A5   UPDATE QUEUE COUNTS                     A0255.304
        PUT,S2  S6&S7,CCQL,A6                                           A0255.305
        R       CHAIN                                                   A0255.306
        A5      SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES           A0255.307
        B00     A5                                                      A0255.308
        A1      0                                                       A0255.309
        STPLK,A0 A1             ENABLE INTERRUPTS                       A0255.310
```

```
*                                                                    A0255.311
*       THE REQUEST HAS BEEN QUEUED TO THE DESTINATION TASK READY THE A0255.312
*       TASK AND SUSPEND UNTIL A REPLY IS RECEIVED                   A0255.313
*                                                                    A0255.314
TSKREQ40 =      *                                                    A0255.315
        S6      SAVEA2,A3                                            A0255.316
        S7      RTSS                                                 A0255.317
        EX                                                           A0255.318
        R       ERROR0                                               A0255.319
TSKREQ50 =      *                                                    A0255.320
        S1      CMOUT0,A7                                            A0255.321
        S2      CMOUT1,A7                                            A0255.322
        S0      S2                                                   A0255.323
        JSN     TSKREQ60                                             A0255.324
*                                                                    A0255.325
*       THE DESTINATION TASK DID NOT REACTIVATE ME                   A0255.326
*                                                                    A0255.327
        S7      SUSP                                                 A0255.328
        EX                                                           A0255.329
        R       ERROR0                                               A0255.330
        J       TSKREQ50                                             A0255.331
TSKREQ60 =      *                                                    A0255.332
        SAVES1,A3 S1                                                 A0255.333
        SAVES2,A3 S2                                                 A0255.334
        POST    12,S1,S2                                             A0255.335
        R       UNCHAIN         UNCHAIN THE CMOD                     A0255.336
        A5      SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.337
        B00     A5                                                   A0255.338
        ERRAN                                                        A0255.339
        A6      CMPOOL          RELEASE THE CMOD MEMORY              A0255.340
        R       MEMDE                                                A0255.341
        A5      SAVERT,A3       RESTORE B00 FOR TRACE ENTRIES        A0255.342
        B00     A5                                                   A0255.343
        A0      A5              STATUS CHECK                         A0255.344
        ERRAN                                                        A0255.345
*                                                                    A0255.346
*       RESTORE THE REGISTERS AND EXIT                               A0255.347
*                                                                    A0255.348
        A0      SAVEA0,A3                                            A0255.349
        A1      SAVEA1,A3                                            A0255.350
        A2      SAVEA2,A3                                            A0255.351
        A5      SAVEA5,A3                                            A0255.352
        A6      SAVEA6,A3                                            A0255.353
        A7      SAVEA7,A3                                            A0255.354
        S0      SAVES0,A3                                            A0255.355
        S1      SAVES1,A3                                            A0255.356
        S2      SAVES2,A3                                            A0255.357
        S3      SAVES3,A3                                            A0255.358
        S4      SAVES4,A3                                            A0255.359
        S5      SAVES5,A3                                            A0255.360
        S6      SAVES6,A3                                            A0255.361
        S7      SAVES7,A3                                            A0255.362
        A4      SAVERT,A3                                            A0255.363
        B00     A4                                                   A0255.364
        A4      SAVEA4,A3                                            A0255.365
        A3      SAVEA3,A3                                            A0255.366
        J       B00                                                  A0255.367
TSKREQ70 =      *                                                    A0255.368
*                                                                    A0255.369
*       THE QUEUE FOR THE DESTINATION TASK IS AT ITS LIMIT THE       A0255.370
*       REQUESTOR WILL HAVE TO WAIT                                  A0255.371
*                                                                    A0255.372
        A1      0                                                    A0255.373
        STPLK,A0 A1             RELEASE INTERRUPT LOCKOUT            A0255.374
        S7      SUSP                                                 A0255.375
        EX                                                           A0255.376
        R       ERROR0                                               A0255.377
        J       TSKREQ30                                             A0255.378
```

USER TO STP CALLS

- ADVANCE JOB
- ABORT JOB
- GET CURRENT DATE
- GET CURRENT TIME
- ENTER LOGFILE MESSAGE
- DATASET RECALL
- TERMINATE JOB
- SET SENSE SWITCH
- OPEN DATASET
- REQUEST MEMORY
- CLOSE DATASET
- CREATE DNT
- SET EXCHANGE PACKAGE MODE
- GET NEXT CONTROL STATEMENT
- LOAD BINARY DATASET
- RETURN DATASET
- PERMANENT DATASET MANAGEMENT REQUEST
- READ DISC CIRCULAR
- WRITE DISC CIRCULAR
- GET SYSTEM REVISION NUMBERS
- DISPOSE DATASET
- GET CURRENT JULIAN DATE
- RETURN ACCUMULATED CPU TIME
- RETURN ACCOUNTING INFORMATION
- SET P REGISTER & SUSPEND USER
- CLEAR SENSE SWITCH
- TEST SENSE SWITCH
- DELAY JOB

- A USER COMMUNICATES WITH STP THROUGH MACRO CALLS.

STARTUP (Z)

www.rgbautomatyka.pl

TASK PROCESSORS

- CURRENT TASKS ARE:

  CRAY-OS STARTUP

  STATION CALL PROCESSOR

  DISK QUEUE MANAGER

  PERMANENT DATASET MANAGER

  JOB CLASS MANAGER

  JOB SCHEDULER

  EXCHANGE PACKAGE PROCESSOR

  MESSAGE PROCESSOR

  MEMORY ERROR PROCESSOR

  DISK ERROR CORRECTION

  SYSTEM PERFORMANCE MONITOR

  OVERLAY MANAGER

  TAPE QUEUE MANAGER

CRAY-OS STARTUP

- LOADS CRAY-OS OPERATING SYSTEM (COS) INTO MEMORY.
- BEGINS EXECUTION OF THE OPERATING SYSTEM.
- GENERATES OR RECOVERS SYSTEM TABLES.
- THREE METHODS OF STARTUP.

INSTALL-INITIATION OF COS THE FIRST TIME

DEADSTART-CONTINUATION OF COS FOLLOWING A NORMAL SYSTEM SHUTDOWN

RESTART-CONTINUATION OF COS FOLLOWING A SYSTEM INTERRUPTION.

INSTALL

- LOADS THE CRAY-1 OPERATING SYSTEM (COS) INTO CRAY-1 MEMORY.
- CRAY-1 MASS STORAGE IS INITIALIZED FOR THE VERY FIRST TIME.

  A DEVICE LABEL (DVL) IS WRITTEN ON EACH DISK UNIT.

  SPACE IS ZEROED AND RESERVED ON THE MASTER DEVICE SUFFICIENT TO HOLD CRAY-1 MEMORY SIZE.

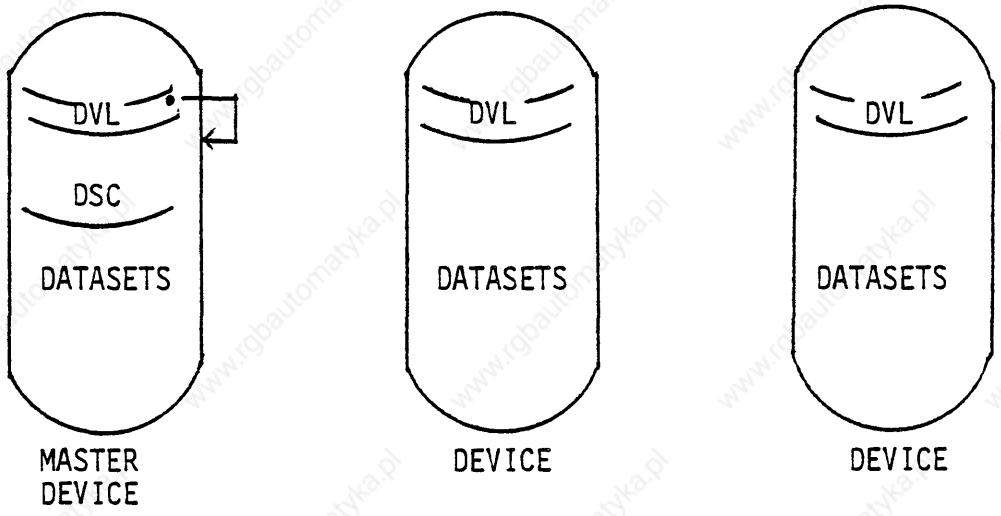
  A ROLL JOB INDEX DATASET IS INITIALIZED ($ROLL).

- SYSTEM TABLES ARE INITIALIZED FOR THE VERY FIRST TIME.

  REFLECT HOW MUCH USEABLE DISK SPACE IS AVAILABLE (DRT).

  CREATES A DISK DATASET CATALOG (DSC) AND WRITES THE DSC TO THE MASTER DEVICE.

  MAKES ENTRIES IN THE DSC FOR $ROLL



Mass Storage Organization

SYSTEM DIRECTORY INITIALIZATION

- THE SYSTEM DIRECTORY (SDR) MUST BE INITIALIZED AFTER ANY INSTALL (AUTOMATIC IN DEADSTART AND RESTART).

- INITIALIZATION OCCURS THROUGH SUBMISSION OF A HIGH PRIORITY JOB.

- THE JOB's PRIORITY SHOULD ENABLE IT TO EXECUTE BEFORE ANY OTHER JOB IN THE INPUT QUEUE.

- ONCE A DATASET NAME IS ENTERED IN THE SDR IT IS ACCESSABLE TO OTHER JOBS.

- AN 'ACCESS' CONTROL STATEMENT WITH PARAMETER 'ENTER' IS SPECIFIED FOR EACH DATASET BEING ENTERED IN THE SDR.

SAMPLE JOB:

```
JOB,JN=SYSDIR,T=2,M=20,P=15.
*.
*.  SDR initialization job
*.
ACCESS,ENTER,DN=AUDIT.
ACCESS,ENTER,DN=BUILD.
ACCESS,ENTER,DN=CAL.
ACCESS,ENTER,DN=CFT.
ACCESS,ENTER,DN=COMPARE.
ACCESS,ENTER,DN=COPYD.
ACCESS,ENTER,DN=COPYF.
ACCESS,ENTER,DN=COPYR.
ACCESS,ENTER,DN=DSDUMP.
ACCESS,ENTER,DN=DUMP.
ACCESS,ENTER,DN=EXTRACT.
ACCESS,ENTER,DN=FDUMP.
ACCESS,ENTER,DN=LDR.
ACCESS,ENTER,DN=PDSDUMP.
ACCESS,ENTER,DN=PDSLOAD.
ACCESS,ENTER,DN=SKIPD.
ACCESS,ENTER,DN=SKIPF.
ACCESS,ENTER,DN=SKIPR.
ACCESS,ENTER,DN=UNB.
ACCESS,ENTER,DN=UPDATE.
ACCESS,ENTER,DN=WRITEDS.
(EOF)
```

DEADSTART

- CONTINUES THE CRAY-1 OPERATING SYSTEM (COS) FOLLOWING A NORMAL SYSTEM SHUTDOWN.

- DELETES DSC ENTRIES FOR INPUT AND OUTPUT DATASETS. (SDT)

- PRESERVES DSC ENTRIES FOR PERMANENT DATASETS.

- PRESERVES DISK SPACE OCCUPIED BY THE SYSTEM DUMP.

- COPIES SYSTEM DUMP TO ANOTHER AREA IF USED AND MAKES IT A PERMANENT DATASET.

- REBUILDS THE SYSTEM DIRECTORY FROM DISK IF DESIRED.

RESTART

- CONTINUES THE CRAY-1 OPERATING SYSTEM (COS) FOLLOWING AN ABNORMAL SYSTEM INTERRUPTION.
- PRESERVES DSC ENTRIES FOR INPUT AND OUTPUT DATASETS.
- PRESERVES DSC ENTRIES FOR PERMANENT DATASETS.
- PRESERVES DISK SPACE OCCUPIED BY THE SYSTEM DUMP.
- COPIES SYSTEM DUMP TO ANOTHER AREA IF USED AND MAKES IT A PERMANENT DATASET.
- PRESERVES ROLLED JOBS AND ASSOCIATED DATASETS IF REQUIRED.
- REBUILDS THE SYSTEM DIRECTORY FROM DISK IF DESIRED.

RECOVERY OF THE SYSTEM DIRECTORY DURING A RESTART

- AN ACCESS OF THE $SDR PERMANENT DATASET IS REQUESTED.
- IF $SDR DOES NOT EXIST THEN THE SYSTEM CREATES $SDR AS A PERMANENT DATASET.
- IF $SDR EXISTS A DATSET NAME TABLE (DNT) IS CREATED AND THE DATASET IS ACCESSED.
- IF THE *SDR COMMAND IS IN THE RESTART PARAMETER FILE THE PREVIOUS $SDR DATASET IS NOT RECOVERED AND A NEW EDITION OF $SDR IS CREATED AS A PERMANENT DATASET.
- ONCE INITIALIZATION OF THE $SDR DATASET IS COMPLETE, ADDITIONAL ENTRIES MAY BE ADDED TO THE SYSTEM DIRECTORY AND THE $SDR DATASET.

RECOVERY OF ROLLED JOBS DURING A RESTART

- A VALIDATION OF THE ROLLED JOB INDEX TABLE ENTRY FOR A JOB ENSURES THE JOB IS NOT MARKED IRRECOVERABLE AND THE JOB RESIDES ON AN AVAILABLE DEVICE.

- THE JOBS JOB TABLE AREA (JTA) IS READ INTO MEMORY AND THE JOBS DATS ARE VERIFIED AS ARE ALL DATS WITHIN THE SYSTEM.

- IF A JOB'S DAT IS INVALID AND THE JOB IS RERUNNABLE THE SYSTEM CLEARS THE INDEX ENTRY AND THE JOB REMAINS ON THE INPUT QUEUE IN THE SYSTEM DATASET TABLE (SDT).

- IF ALL DATS FOR A JOB ARE VALID THE SYSTEM SETS THE DEVICE RESERVATION TABLE (DRT) ENTRIES FOR THE DATS.

DATASET ALLOCATION TABLE (DAT) VERIFICATION

- A DATASET ALLOCATION TABLE (DAT) MUST RESIDE ENTIRELY IN THE SYSTEM FOR SYSTEM DATASETS, AND WITHIN A JOBS JOB TABLE AREA (JTA) FOR A JOB.

- THE DAT MUST POINT TO THE CORRECT JOB EXECUTION TABLE (JXT) ENTRY FOR A JOB (LOCAL) DATASET.

- THE DEVICES USED BY THE DATASETS MUST BE AVAILABLE.

- FOR A DAT THE LOGICAL TRACK ADDRESSES MUST NOT HAVE THE CORRESPONDING BIT SET IN THE DEVICE RESERVATION TABLE (DRT).

- WHEN THE LAST LOGICAL TRACK ADDRESS IS VERIFIED AGAINST THE DRT BIT THE REMAINING ADDRESS COUNT MUST BE ZERO.

- DAT VERIFICATION OCCURS IN TWO PASSES

  FIRST PASS SCANS FOR ERRORS AS ABOVE
  SECOND PASS SETS THE DRT BITS

- PERMANENT DATASETS ARE FURTHER VERIFIED AGAINST THE DISK RESIDENT DATASET CATALOG (DSC) AND EACH DATASET DAT IS COMPARED AGAINST THE DISK COPY.

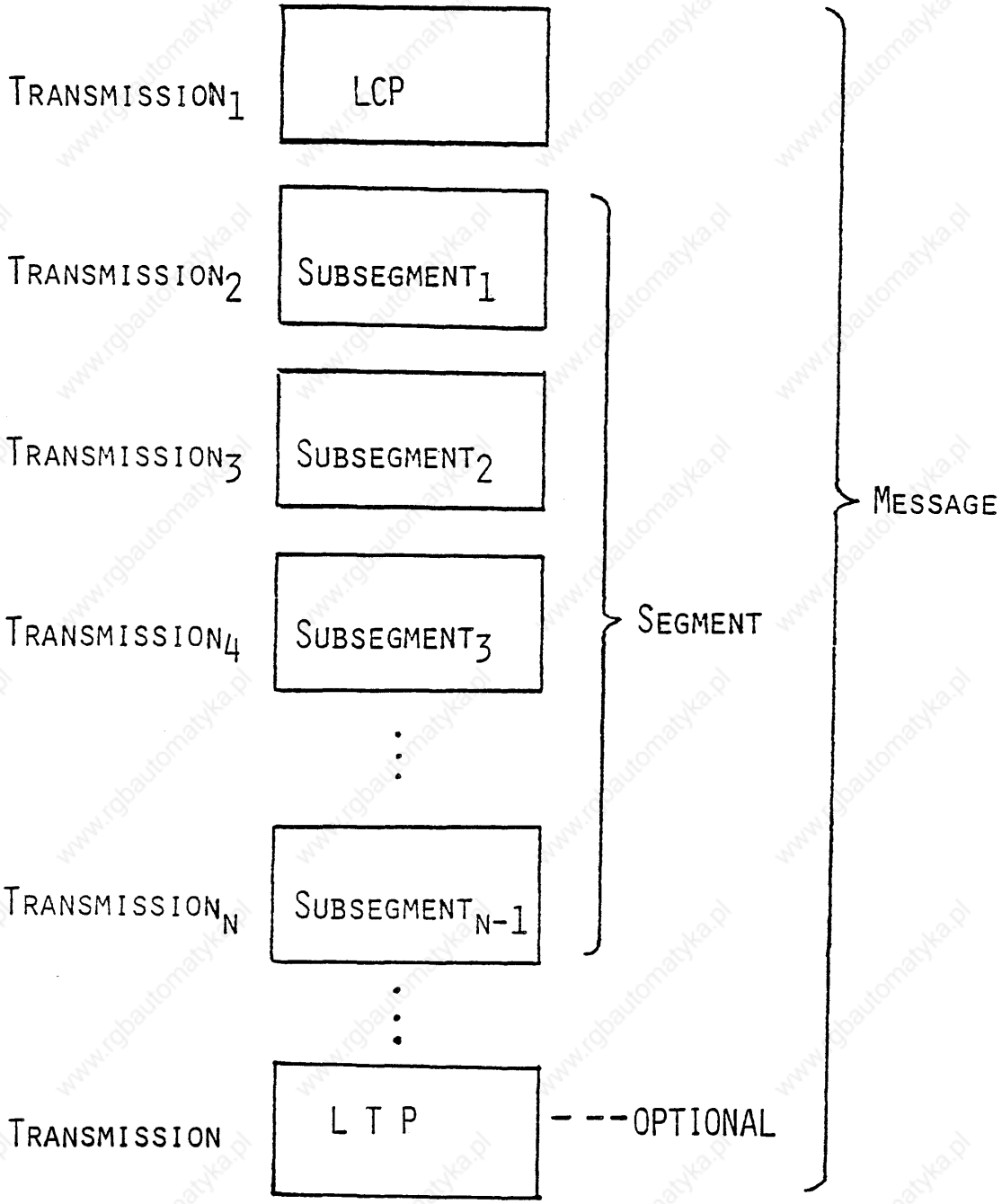
STATION CALL PROCESSOR (SCP)

STATION CALL PROCESSOR (SCP)

- PROVIDES COMMUNICATIONS WITH FRONT-END SYSTEMS.
- MANAGES I/O TRANSMISSION POOLS CREATED AT STARTUP.
- PROVIDES FOR OPERATOR GUIDANCE.
- CHANNEL DISCIPLINE IS TWO WAY ALTERNATE.
- INDEPENDENT OF FRONT-END TYPE.
- MULTIPLEXES STREAMS (8 INPUT AND 8 OUTPUT).

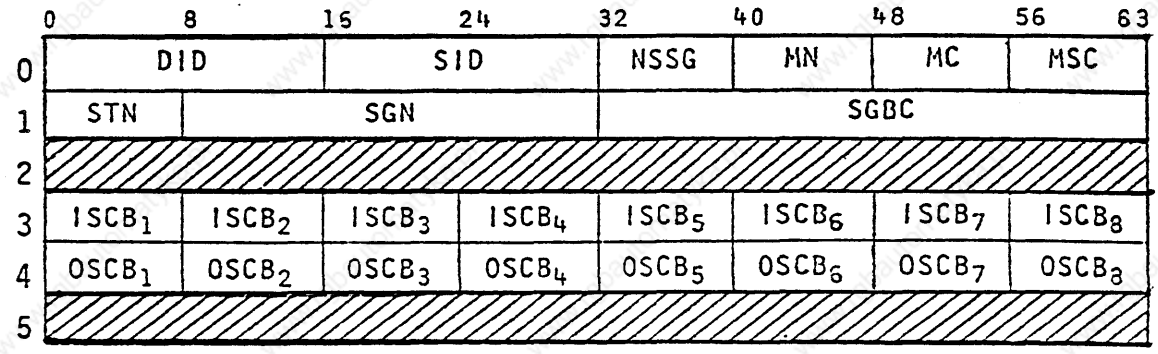
GENERAL INTERFACE PROTOCOL

- EACH MESSAGE IS HEADED BY A LINK CONTROL PACKAGE
- SUBSEGMENT SIZE VARIES WITH FRONT-END



LINK CONTROL PACKAGE

- EACH LCP CONSISTS OF SIX 64-BIT WORDS
- LCP CONTAINS:
  - SOURCE MAINFRAME ID (SID)
  - DESTINATION MAINFRAME ID (DID)
  - NO. OF SUBSEGMENTS (NSSG)
  - MESSAGE NUMBER (MN)
  - MESSAGE CODE (MC)
  - MESSAGE SUB CODE (MSC)
  - STREAM NO. (STN)
  - SEGMENT NUMBER (SGN)
  - SEGMENT LENGTH (SGBC)
  - STREAM CONTROL BYTES (ISCB, OSCB)



Message codes

Code	Function	Sender Front-end	Sender CRAY-1	Segment	Stream Required
001	Logon	x		x	
002	Relog	x††			
003	Logoff	x			
004	Start		x	x	
005	Restart		x		
006	Dataset header	x	x	x	x
007	Dataset segment	x	x	x	x
011	Control	x	x		
012	Message error	x	x		
013	Dataset transfer request	x††	x	x	
014	Dataset transfer reply	x	x	x	
015	Enter logfile request	x††		x	
016	Enter logfile reply		x	x	
020	Logfile information request	x††		x	
021	Job status request	x††		x	
022	System status request	x††		x	
023	Dataset status request	x††		x	
024	Link status request	x††		x	
025	Mass storage status request	x††		x	
026	Operator function request	x††		x	
027	Debug function request	x††		x	
030	Logfile information reply		x	x	
031	Job status reply		x	x	
032	System status reply		x	x	
033	Dataset status reply		x	x	
034	Link status reply		x	x	
035	Mass storage status reply		x	x	
036	Operator function reply		x	x	
037	Debug function reply		x	x	
040	Diagnostic echo request	x††	x	x	
041	Diagnostic echo reply	x††	x	x	

†† Optional; front end not required to send

STREAM CONTROL BYTES

- 8 INPUT AND 8 OUTPUT STREAMS

- ALTHOUGH EACH MESSAGE IS ASSIGNED TO ONLY ONE STREAM, THE LCP MUST CARRY STREAM CONTROL BYTES FOR ALL 16 STREAMS.

Octal Code	Mnemonic	Request/Response	Sender	Receiver
00	IDL	Idle	x	x
01	RTS	Request to send	x	
02	PTR	Preparing to receive		x
03	SND	Sending	x	
04	RCV	Receiving		x
05	SUS	Suspend		x
06	END	End dataset	x	
07	SVG	Saving dataset		x
10	SVD	Dataset saved		x
11	PPN	Postpone	x	x
12	CAN	Cancel	x	x
13	MCL	Master clear	x	x

RECEIVER SCB RESPONSE

SENDER SCB SENT	IDL	PTR	RCV	SUS	SVG	SVD	PPN	CAN
IDL	N							
RTS		N	C	C			A	
SND			N	N			A	A
END					N	C	A	A
PPN	C							
CAN	C							

N = Normal receiver SCB response

C = Normal receiver SCB response which requires change in sender SCB

A = Abnormal receiver SCB response

SENDER SCB RESPONSE

RECEIVER SCB SENT	IDL	RTS	SND	END	PPN	CAN
IDL	N	C				
PTR		N				
RCV			N	C	A	A
SUS			N	C	A	A
SVG				N		
SVD	C					
PPN	C					
CAN	C					

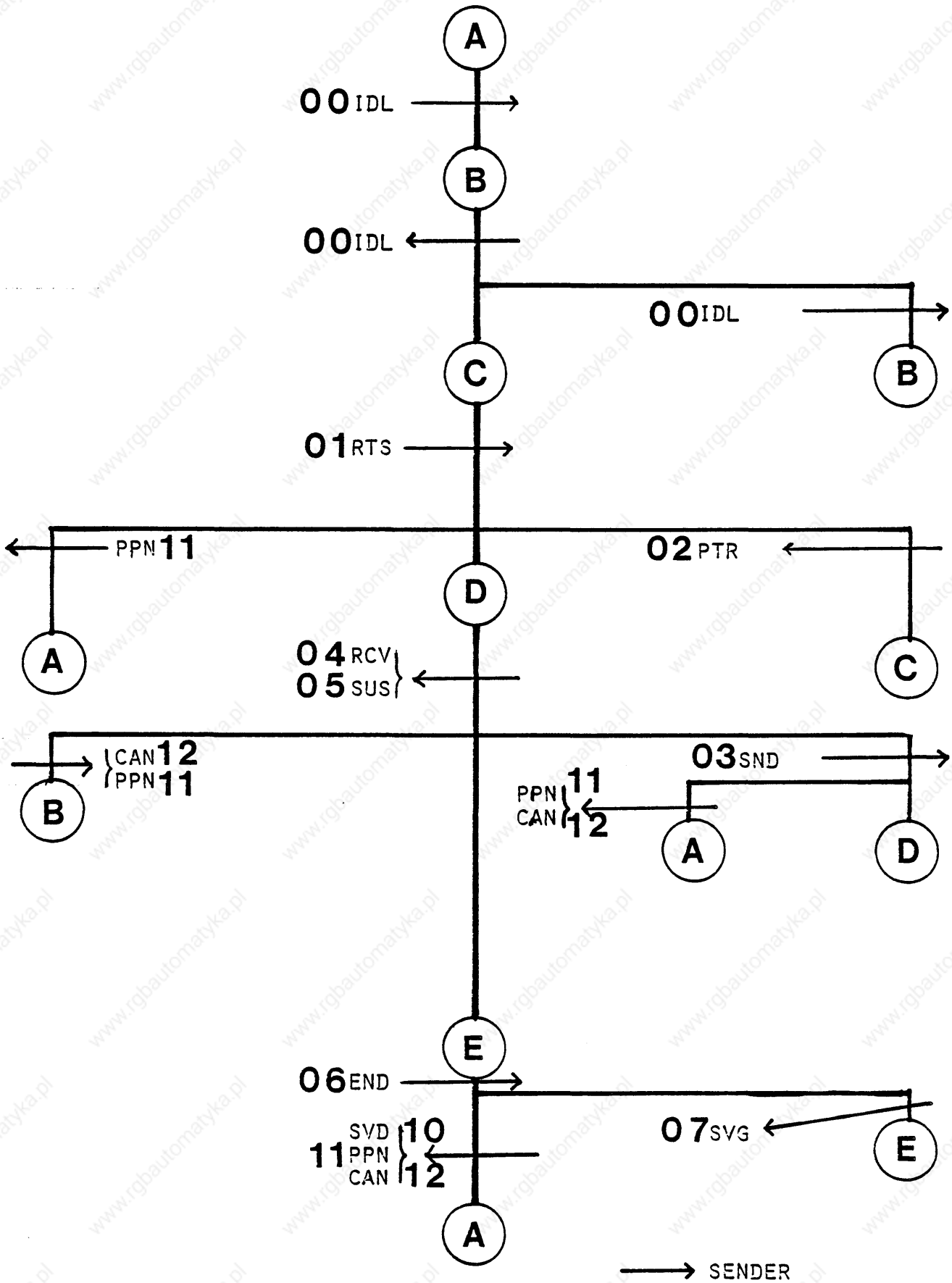
N = Normal sender SCB response

C = Normal sender SCB response which requires change in receiver SCB

A = Abnormal sender SCB response

BASIC STREAM FLOW

- FRONT-END IS LOGGED ON
- COMMUNICATIONS IN AN IDLE STATE
- FRONT-END SENDS RTS(01) TO THE CRAY-1
- CRAY-1 SENDS RCV (04) TO THE FRONT END.
- FRONT-END SENDS SND (03) TO THE CRAY-1 ALONG WITH THE JOB DATASET
- CRAY-1 SENDS RCV (04) TO THE FRONT-END WHILE DECODING THE MESSAGE AND SAVING THE JOB DATASET
- FRONT-END SENDS END (06) TO THE CRAY-1 UNTIL CRAY-1 HAS SAVED THE DATASET.
- CRAY-1 SENDS SVD (10) TO THE FRONT-END ONCE DATASET HAS BEEN SAVED.
- FRONT-END AND CRAY-1 THEN KEEP COMMUNICATIONS OPEN BY ALTERNATELY SENDING AND RECEIVING IDL(00).



13

NOTE: A MCL SCB IS A LEGAL REQUEST OR RESPONSE AT ANY TIME. THE ONLY LEGAL REPLY TO MCL IS IDL.

STREAM CONTROL BYTE FLOW

SCP JOB STREAM FLOW

- EXAMINES THE DATASET HEADER MESSAGE AND DETERMINES THE DATASET IS A JOB DATASET.

- BUILDS DISK BUFFERS FROM STATION BUFFERS AND WHEN BUFFERS ARE FULL HAS THEM WRITTEN TO DISK.

- 'CRACKS' THE JOB CARD ASSIGNING DEFAULTS WHEN APPROPRIATE.

- MAKES APPROPRIATE ENTRIES IN THE JOB QUEUE (SYSTEM DATASET TABLE (SDT)), SUCH AS:

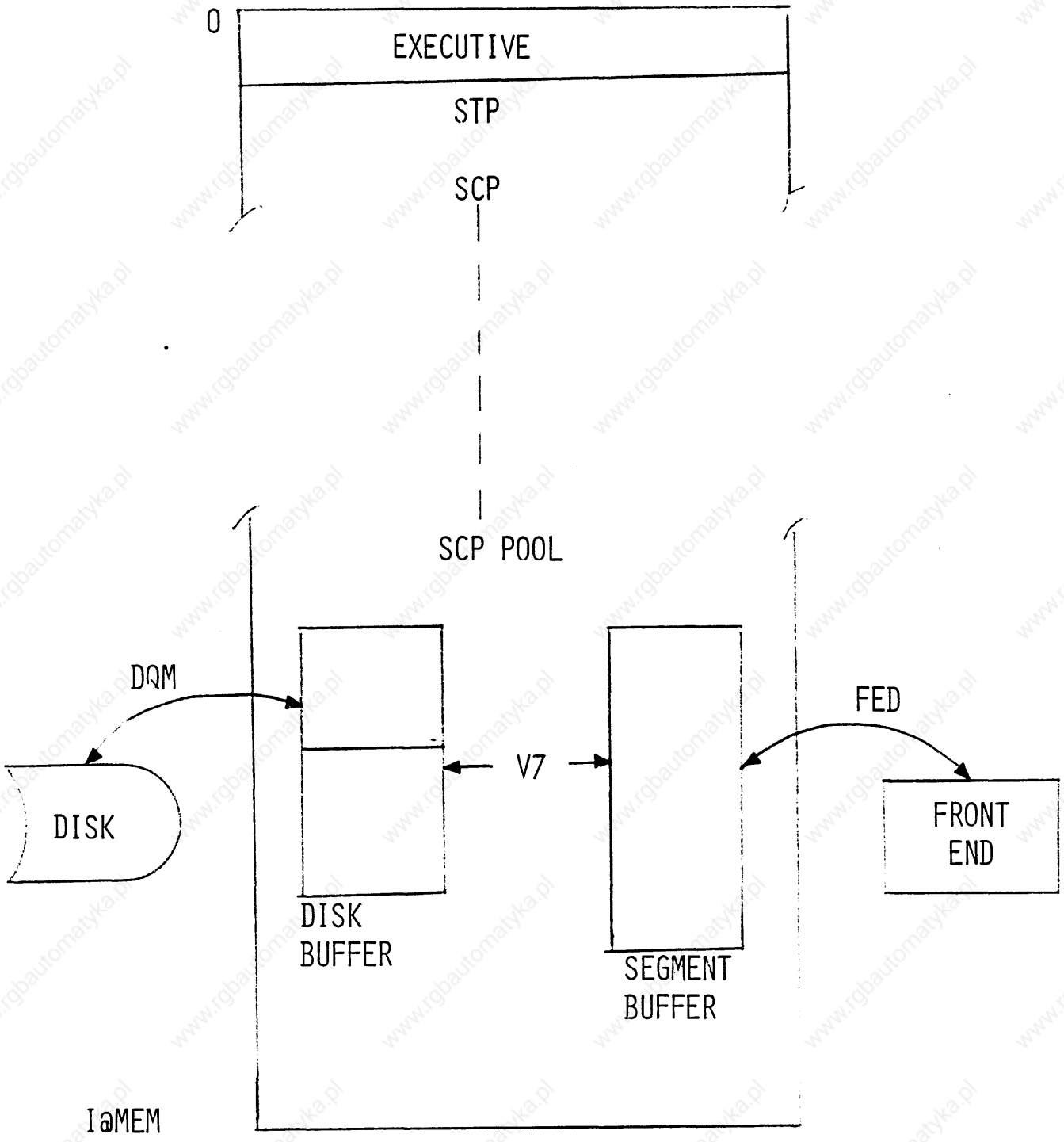
  DATASET NAME

  JOB NAME AND PRIORITY

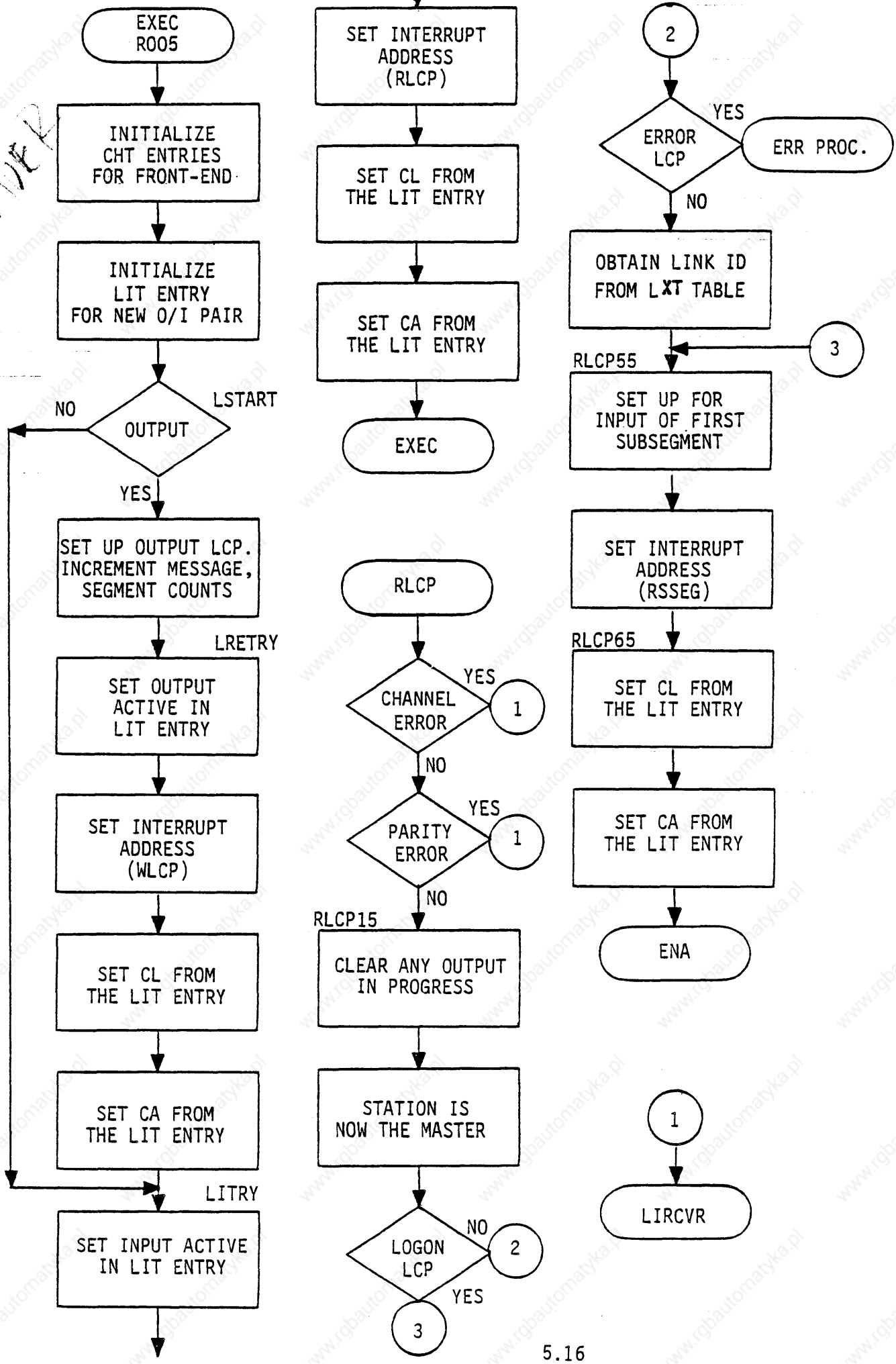
  SOURCE AND DESTINATION ID

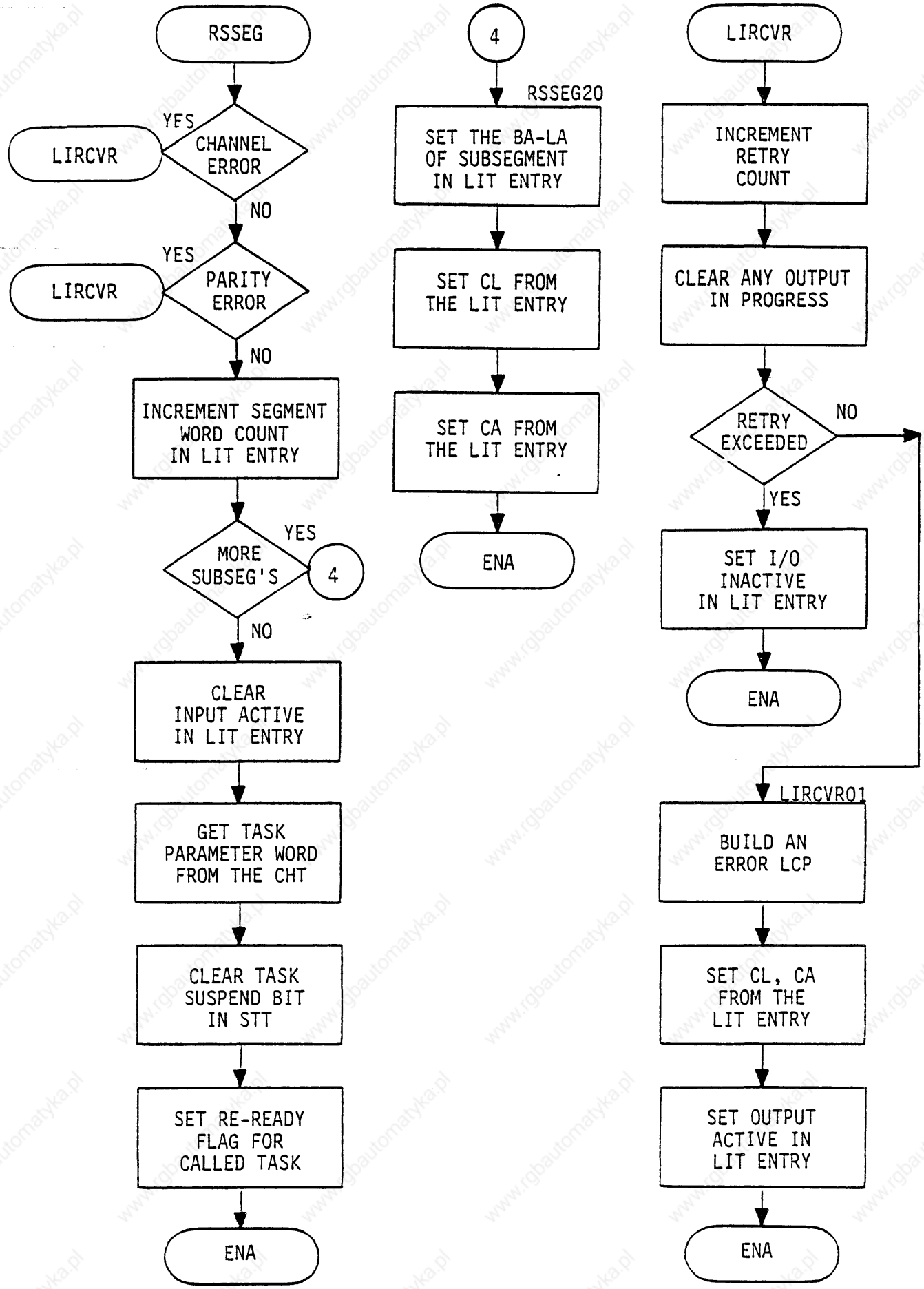
  DISPOSITION CODE SET TO JOB DATASET (IN)

- HAS THE DATASET MADE PERMANENT FOR THE LIFE OF THE JOB. (DSC)



SCP DATA TRANSFER


EXEC
R005
INITIALIZE
CHT ENTRIES
FOR FRONT-END
INITIALIZE
LIT ENTRY
FOR NEW O/I PAIR
NO
OUTPUT
LSTART
YES
SET UP OUTPUT LCP.
INCREMENT MESSAGE,
SEGMENT COUNTS
LRETRY
SET OUTPUT
ACTIVE IN
LIT ENTRY
SET INTERRUPT
ADDRESS
(WLCP)
SET CL FROM
THE LIT ENTRY
SET CA FROM
THE LIT ENTRY
LITRY
SET INPUT ACTIVE
IN LIT ENTRY
SET INTERRUPT
ADDRESS
(RLCP)
SET CL FROM
THE LIT ENTRY
SET CA FROM
THE LIT ENTRY
EXEC
RLCP
CHANNEL
ERROR
YES
1
NO
PARITY
ERROR
YES
1
NO
RLCP15
CLEAR ANY OUTPUT
IN PROGRESS
STATION IS
NOW THE MASTER
LOGON
LCP
NO
2
YES
3
2
ERROR
LCP
YES
ERR PROC.
NO
OBTAIN LINK ID
FROM LXT TABLE
3
RLCP55
SET UP FOR
INPUT OF FIRST
SUBSEGMENT
SET INTERRUPT
ADDRESS
(RSSEG)
RLCP65
SET CL FROM
THE LIT ENTRY
SET CA FROM
THE LIT ENTRY
ENA
1
LIRCVR


RSSEG
YES
CHANNEL ERROR
LIRCVR
NO
YES
PARITY ERROR
LIRCVR
NO
INCREMENT SEGMENT WORD COUNT IN LIT ENTRY
YES
MORE SUBSEG'S
4
NO
CLEAR INPUT ACTIVE IN LIT ENTRY
GET TASK PARAMETER WORD FROM THE CHT
CLEAR TASK SUSPEND BIT IN STT
SET RE-READY FLAG FOR CALLED TASK
ENA
4
RSSEG20
SET THE BA-LA OF SUBSEGMENT IN LIT ENTRY
SET CL FROM THE LIT ENTRY
SET CA FROM THE LIT ENTRY
ENA
LIRCVR
INCREMENT RETRY COUNT
CLEAR ANY OUTPUT IN PROGRESS
RETRY EXCEEDED
NO
YES
SET I/O INACTIVE IN LIT ENTRY
ENA
LIRCVR01
BUILD AN ERROR LCP
SET CL, CA FROM THE LIT ENTRY
SET OUTPUT ACTIVE IN LIT ENTRY
ENA

DISK QUEUE MANAGER (DQM)

DQM REQUESTS

— PRE ALLOCATE SPACE — MODES
STATIC
DYNAMIC

— DATA TRANSFER

— DE ALLOCATE SPACE

SIZE IN TRACKS

ONLY SIZE IS TRACK

DEVICE ALLOCATION

BY LOGICAL DEVICE NAME

BY [illegible] NAME

DISK QUEUE MANAGER (DQM)

- MANAGES ALLOCATION/DEALLOCATION OF MASS STORAGE (DISKS)
- MANAGES MASS STORAGE REQUEST QUEUES
- MANAGES MASS STORAGE CHANNELS, CONTROLLERS AND DISK UNITS.
- MAXIMUM OF 11 DISK CONTROLLERS (44 DSU's)
- UPDATES THE DEVICE RESERVATION TABLE (DRT) FOR EACH DSU.
- CALLED BY EXEC (UPON TRANSFER OF AN I/O SEGMENT) OR BY ANOTHER TASK (FOR DISK REQUEST).

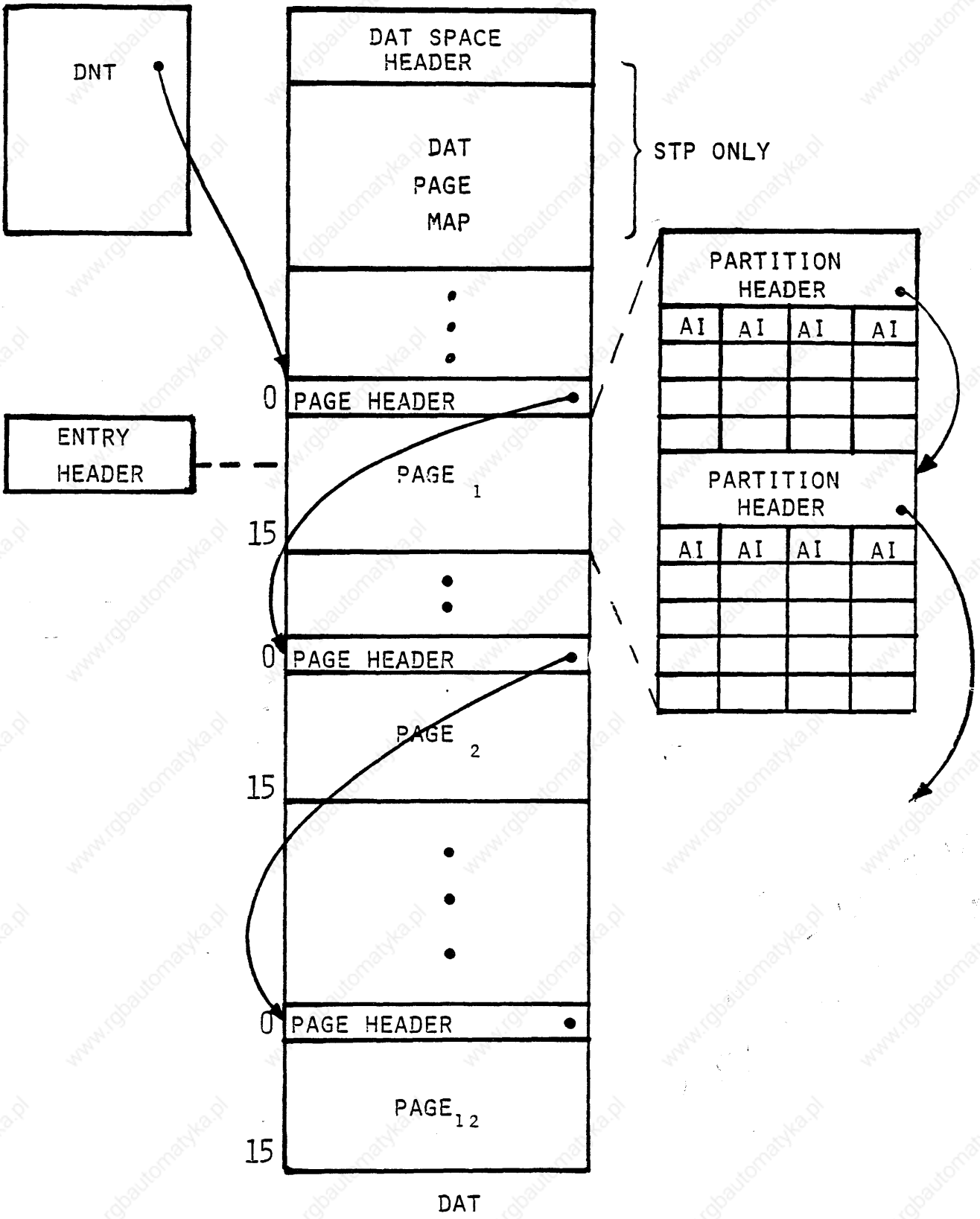
DATASET ALLOCATION TABLE - DAT

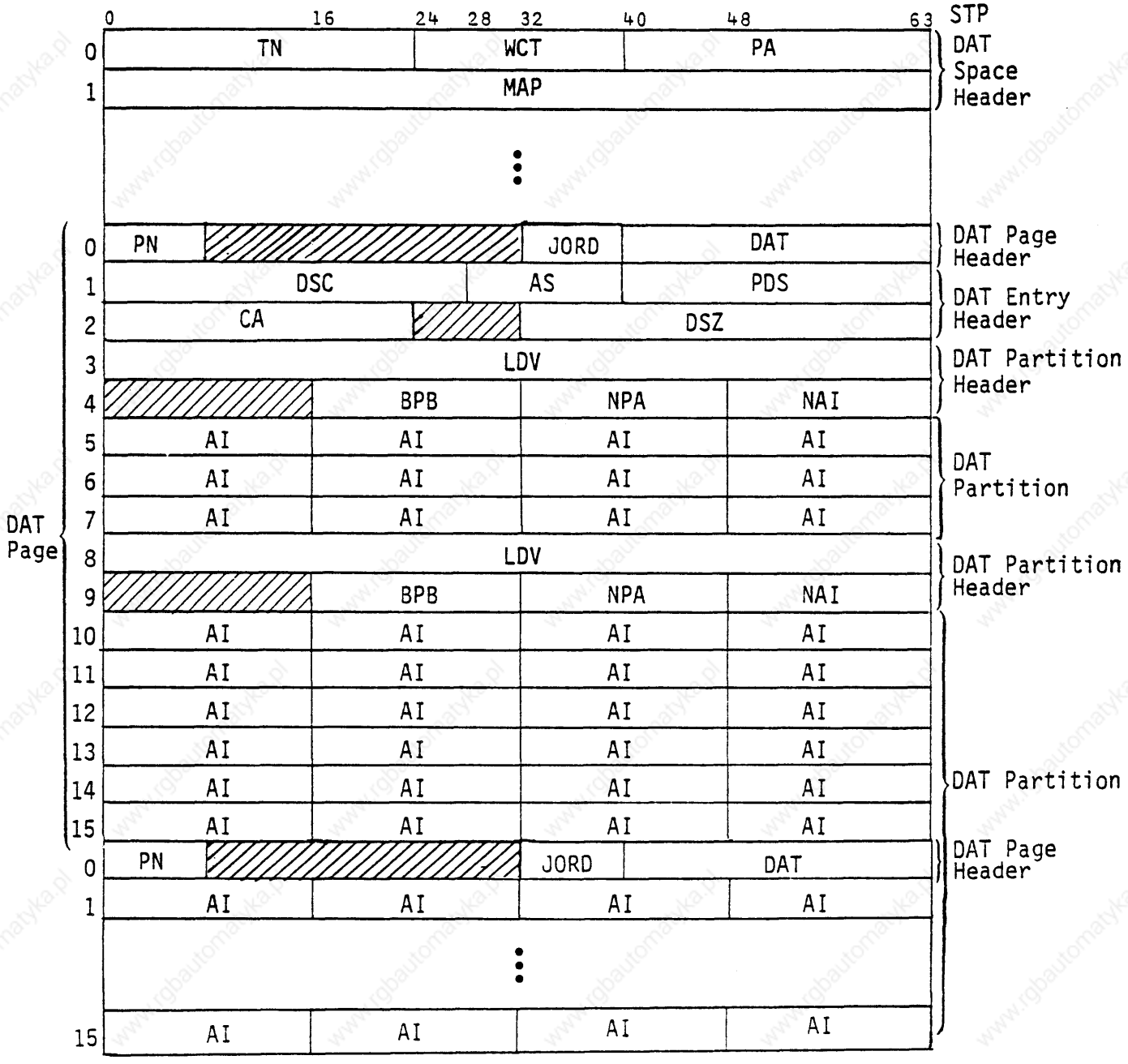
A DAT exists for each dataset in the system. A DAT defines where a dataset logically resides on mass storage, i.e., on which logical device or devices, and what portions of each device.

The DAT page and entry header contains general dataset information.

The DAT partition header contains general information concerning a particular partition of the DAT. A partition represents a portion of a single logical device. Each allocation index (AI) in a partition is a bit number in the respective Device Reservation Table (DRT).

The DAT is composed of as many 16-word DAT pages as necessary to represent the mass storage occupied by the dataset. Additional DAT pages continue from the point at which the last DAT left off.

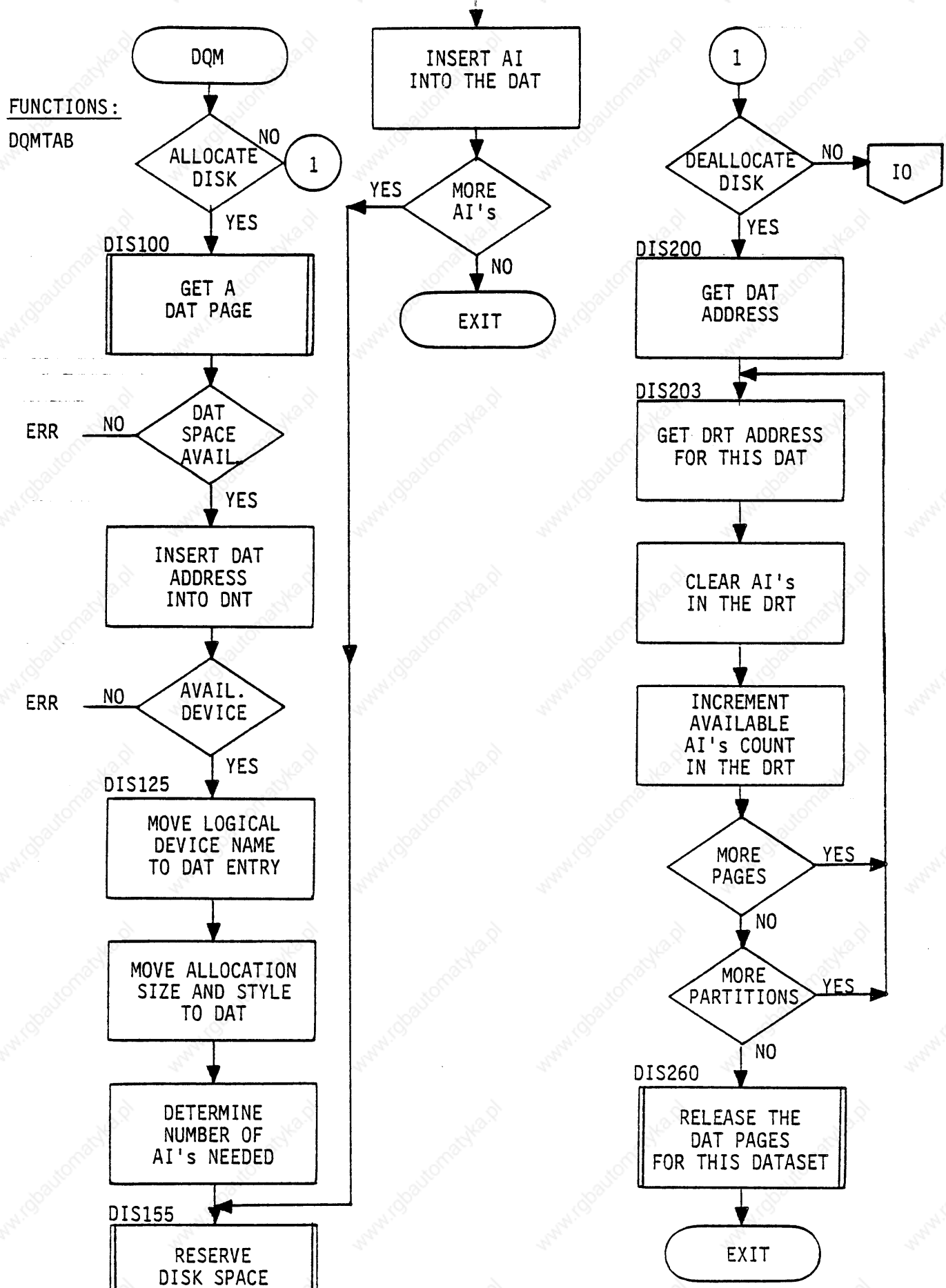

DNT
DAT SPACE HEADER
DAT PAGE MAP
STP ONLY
PARTITION HEADER
AI AI AI AI
ENTRY HEADER
0
PAGE HEADER
PAGE 1
15
PARTITION HEADER
AI AI AI AI
0
PAGE HEADER
PAGE 2
15
0
PAGE HEADER
PAGE 12
15
DAT

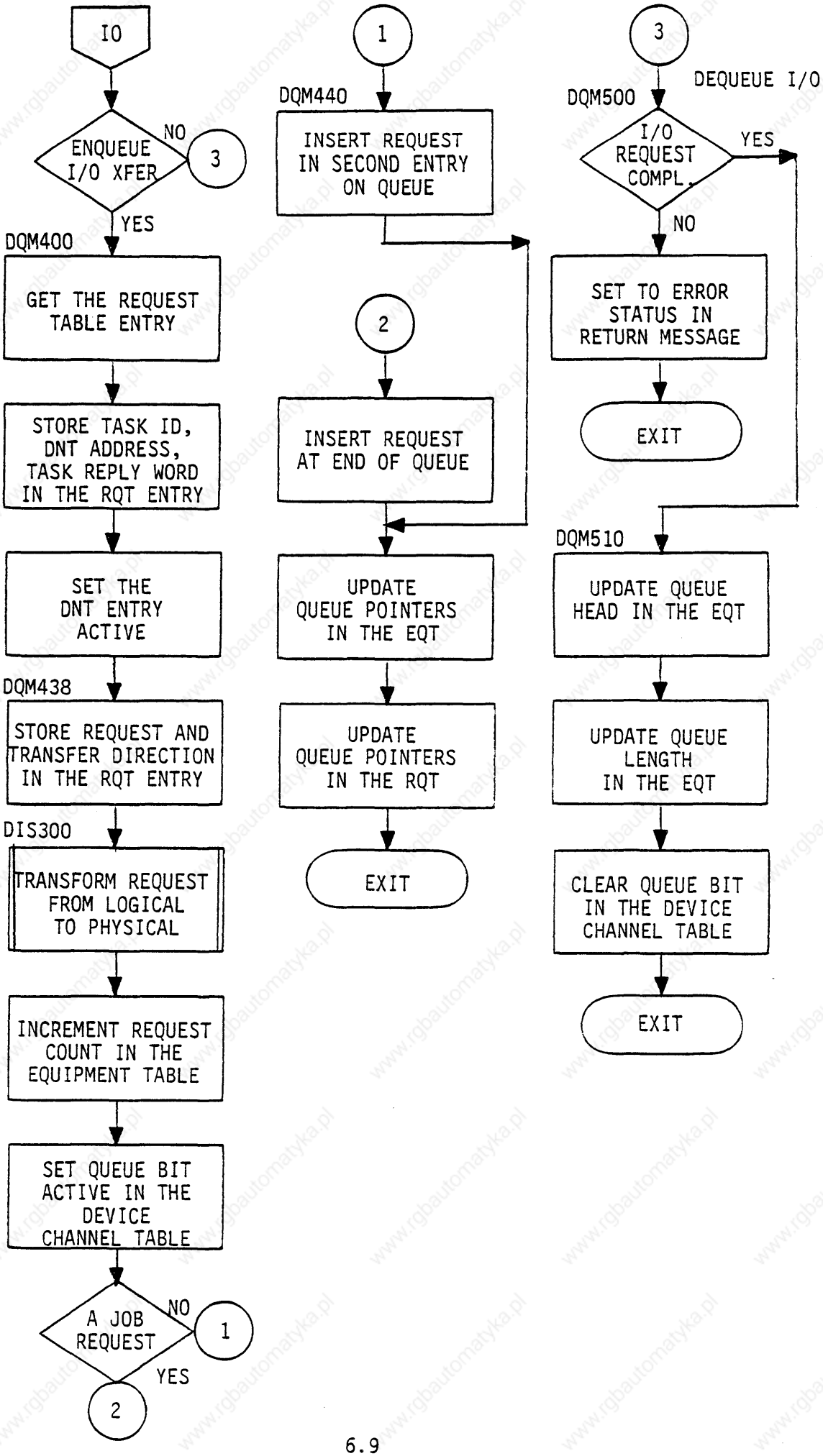


Dataset Allocation Table (DAT)

```
        DATASET ALLOCATION TABLE AREA
0020372 0421012500000000000412 1777773777774000000000                                                  DAT
0020374 0000000000000000000000 0000000000000000000000 0000000000000000000000

0020400 0004000000000000000000 0000000000000100000000 0000001620000001223000 0421041323047113231060   '                9   & DD-19-20
0020404 0000000001100000000045 0000020000140001000005 0000060000340002000011 0000120000600003200016       %
0020410 0000170001000004200022 0000230001200005200026 0000270001400006200032 0000330001600007200036
0020414 0000370002000010200042 0000430002200011200046 0000470000000000000000 0000000000000000000000     ! " # $ % & '

0020420 0004000000000000020440 0000000000000100000000 0000000000000004026000 0421041323047113231060     !                , DD-19-20
0020424 0000000001100000000163 0000500002440012400053 0000540002640013400057 0000600003040014400063       ( ) * + , - . / 0 1 2 3
0020430 0000640003240015400067 0000700003440016400073 0000740003640017400077 0001000004040020400103 4 5 6 7 8 9 : ; < = > ? @ A B C
0020434 0001040004240021400107 0001100004440022400113 0001140004640023400117 0001200005040024400123 D E F G H I J K L M N O P Q R S

0020440 0010000000000000020460 0001240005240025400127 0001300005440026400133 0001340005640027400137     !0 T U V W X Y Z [ \ ] ^ _
0020444 0001400006040030400143 0001440006240031400147 0001500006440032400153 0001540006640033400157 `
0020450 0001600007040034400163 0001640007240035400167 0001700007440036400173 0001740007640037400177
0020454 0002000010040040400203 0002040010240041400207 0002100010440042400213 0002140010640043400217

0020460 0014000000000000000000 0002200011040044400223 0002240011240045400227 0002300011440046400000
    464 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
    470 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
    474 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000

0020500 0004000000000000000000 0000013340000100000000 0000000520000000001000 0421041323047113231060                     DD-19-20
0020504 0000000001100000000001 0000010000000000000000 0000000000000000000000 0000000000000000000000
0020510 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0020514 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000

0020520 0004000000000000020540 0000021604000100000000 0000004300000003720000 0421041323047113231060     !`  8@             DD-19-20
0020524 0000000001100000000160 0023010114100470202352 0024040145000624603171 0031730151540702203416                P S
0020530 0034210161100704603424 0034250161400713203456 0034700163501012004102 0041640207241035604334           - . 8 : ( B
0020534 0043760222341174605002 0050060243341216005071 0064240321541507006437 0064430322201511206446   '      7 8 9        # $ % &

0020540 0010000000000000020560 0064470322401512206453 0064540322641513406457 0064600323041514406463     !  ' ( ) + , - . / 0 1 2 3
0020544 0064640323241515406477 0065020324141521006505 0065060327341741407607 0076100370441742407613 4 5 6 ? B C D E F
0020550 0076140370641743407617 0076200371101744607624 0076250371341746007631 0076320371541747007635
0020554 0076360371741750007641 0076420372141751007645 0076460372341752007651 0076520372541753007655

0020560 0014000000000000000000 0076560372741754007661 0076620373141755007666 0000000000000000000000
              LOCATIONS 00020564 THROUGH 00020573 CONTAIN 0000000000000000000000
0020574 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000

0020600 0004000000000000000000 0000001064000100000000 0000000520000000001000 0421041323047113231463         #@             DD-19-33
0020604 0000000001100000000001 0012010000000000000000 0000000000000000000000 0000000000000000000000
```

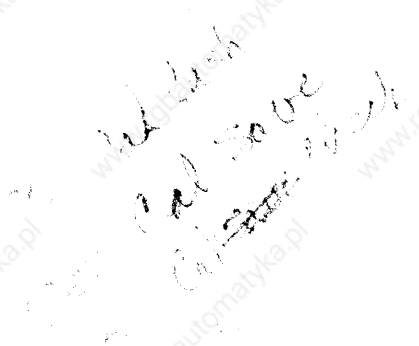
6.7


FUNCTIONS:
DQMTAB
DQM
ALLOCATE DISK
NO
1
YES
DIS100
GET A DAT PAGE
ERR
NO
DAT SPACE AVAIL.
YES
INSERT DAT ADDRESS INTO DNT
ERR
NO
AVAIL. DEVICE
YES
DIS125
MOVE LOGICAL DEVICE NAME TO DAT ENTRY
MOVE ALLOCATION SIZE AND STYLE TO DAT
DETERMINE NUMBER OF AI's NEEDED
DIS155
RESERVE DISK SPACE IN THE DRT
INSERT AI INTO THE DAT
YES
MORE AI's
NO
EXIT
1
DEALLOCATE DISK
NO
IO
YES
DIS200
GET DAT ADDRESS
DIS203
GET DRT ADDRESS FOR THIS DAT
CLEAR AI's IN THE DRT
INCREMENT AVAILABLE AI's COUNT IN THE DRT
MORE PAGES
YES
NO
MORE PARTITIONS
YES
NO
DIS260
RELEASE THE DAT PAGES FOR THIS DATASET
EXIT


IO
ENQUEUE I/O XFER
NO
3
YES
DQM400
GET THE REQUEST TABLE ENTRY
STORE TASK ID, DNT ADDRESS, TASK REPLY WORD IN THE RQT ENTRY
SET THE DNT ENTRY ACTIVE
DQM438
STORE REQUEST AND TRANSFER DIRECTION IN THE RQT ENTRY
DIS300
TRANSFORM REQUEST FROM LOGICAL TO PHYSICAL
INCREMENT REQUEST COUNT IN THE EQUIPMENT TABLE
SET QUEUE BIT ACTIVE IN THE DEVICE CHANNEL TABLE
A JOB REQUEST
NO
1
YES
2
1
DQM440
INSERT REQUEST IN SECOND ENTRY ON QUEUE
2
INSERT REQUEST AT END OF QUEUE
UPDATE QUEUE POINTERS IN THE EQT
UPDATE QUEUE POINTERS IN THE RQT
EXIT
3
DEQUEUE I/O
DQM500
I/O REQUEST COMPL.
YES
NO
SET TO ERROR STATUS IN RETURN MESSAGE
EXIT
DQM510
UPDATE QUEUE HEAD IN THE EQT
UPDATE QUEUE LENGTH IN THE EQT
CLEAR QUEUE BIT IN THE DEVICE CHANNEL TABLE
EXIT

PERMANENT DATASET MANAGER (PDM)

www.rgbautomatyka.pl



PERMANENT DATASET MANAGER (PDM)

- PROVIDES FOR CREATING, ACCESSING, DELETING, MAINTAINING, AND AUDITING PERMANENT DATASETS.

- PERMANENT DATASETS MAY EXIST FOR USERS OR THE SYSTEM.

- USER PERMANENT DATASETS ARE CREATED UPON REQUEST BY A USER JOB. (SAVE)

- SYSTEM PERMANENT DATASETS ARE CREATED FOR SPOOLED INPUT AND OUTPUT DATASETS.

- A SYSTEM PERMANENT DATASET IS CREATED FOR JOB DATASETS ARRIVING FROM A FRONT-END. (SDT)

- A PERMANENT JOB DATASET IS DELETED UPON COMPLETION OF THE JOB.

Doc. contains DATS
(SDS header) Page

ECW
Page header
SDS header

Entry DATS
63 words

7/page

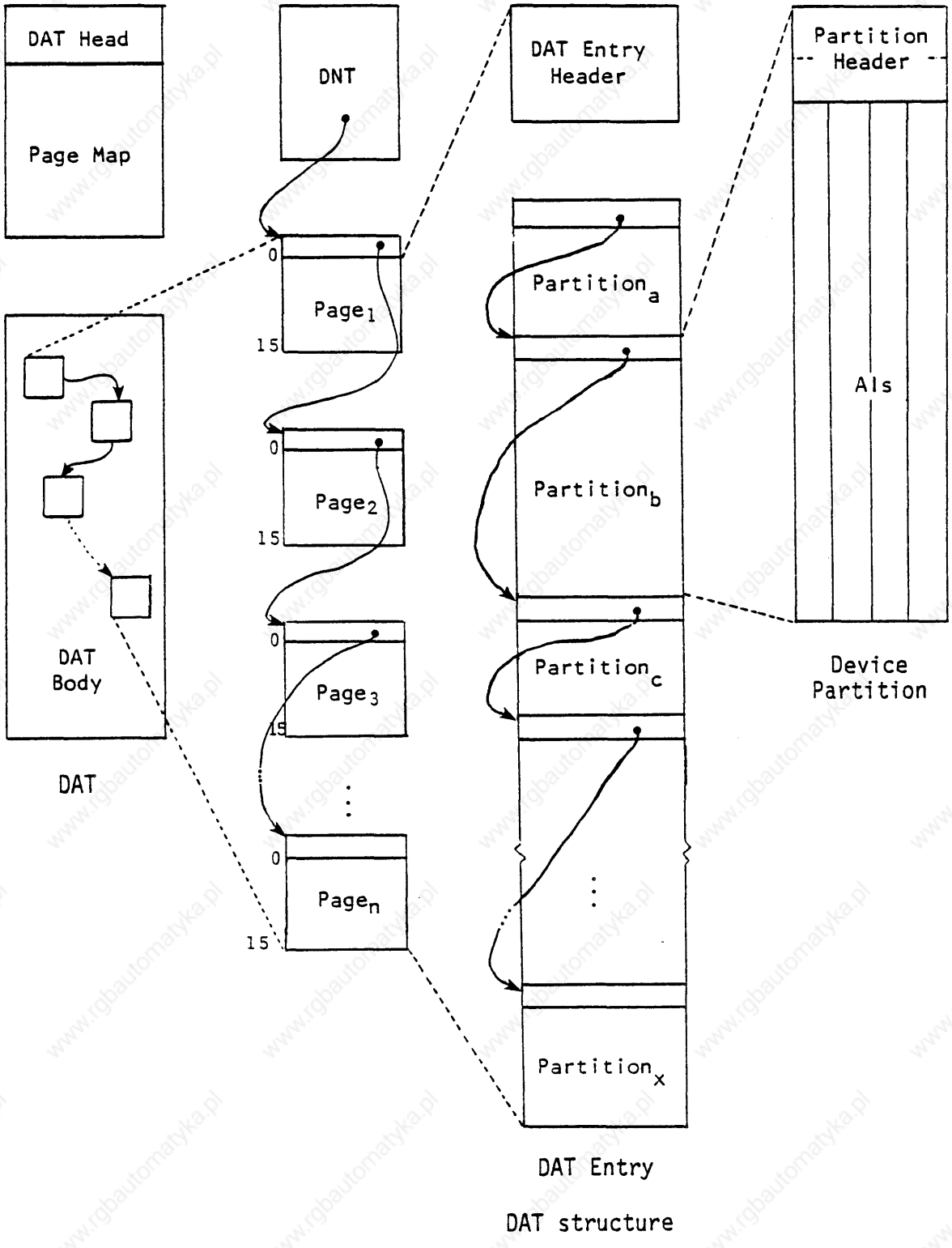
ASFS HASH info
[illegible] AROll
No tree structure

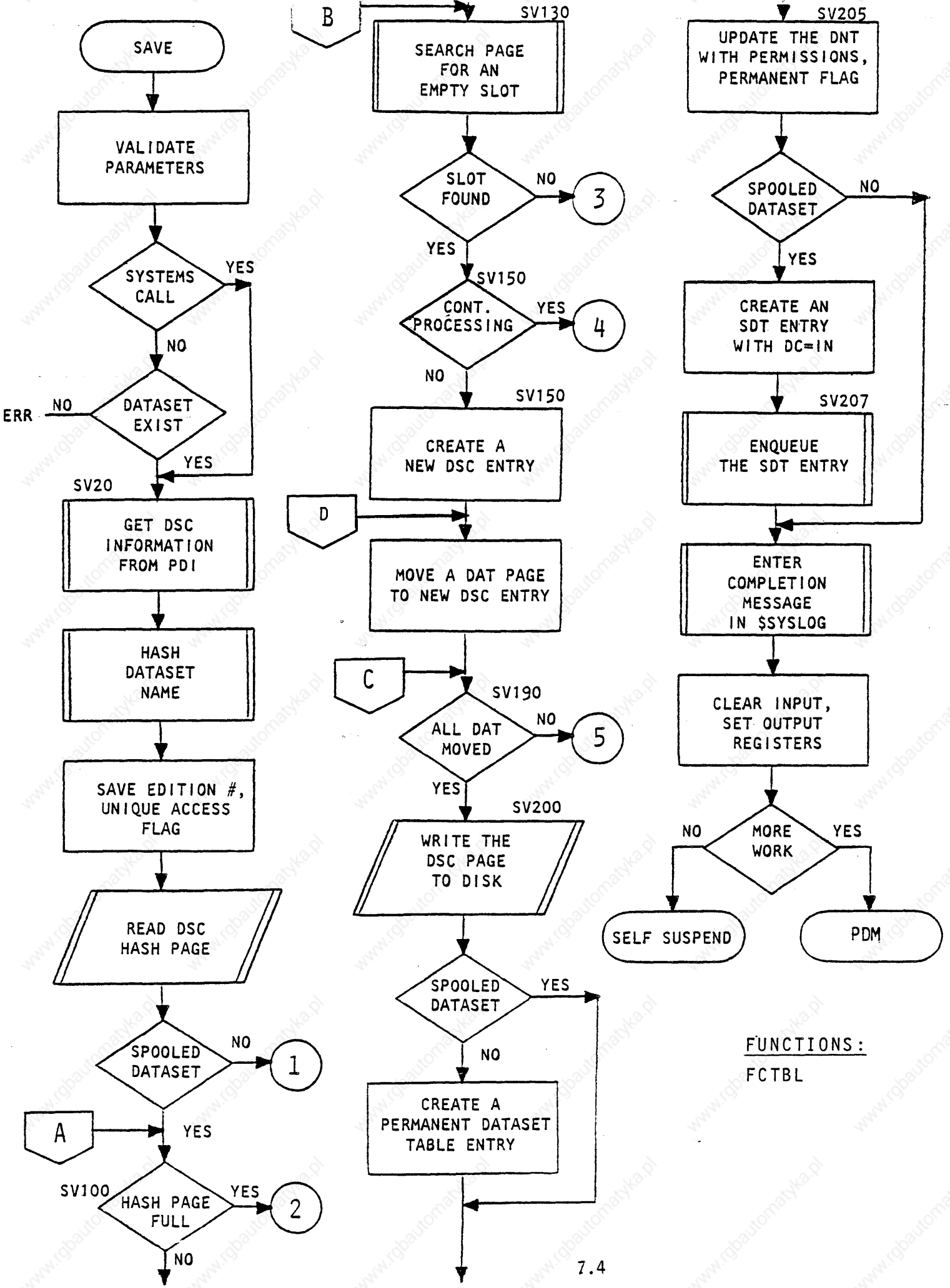
Media Descr. label contains
[illegible] => Max Doc 15 M[illegible]

[illegible] UNEMARK [illegible] DR (NEW STRUCT)

PDT PDS PDD [illegible] REGISTER

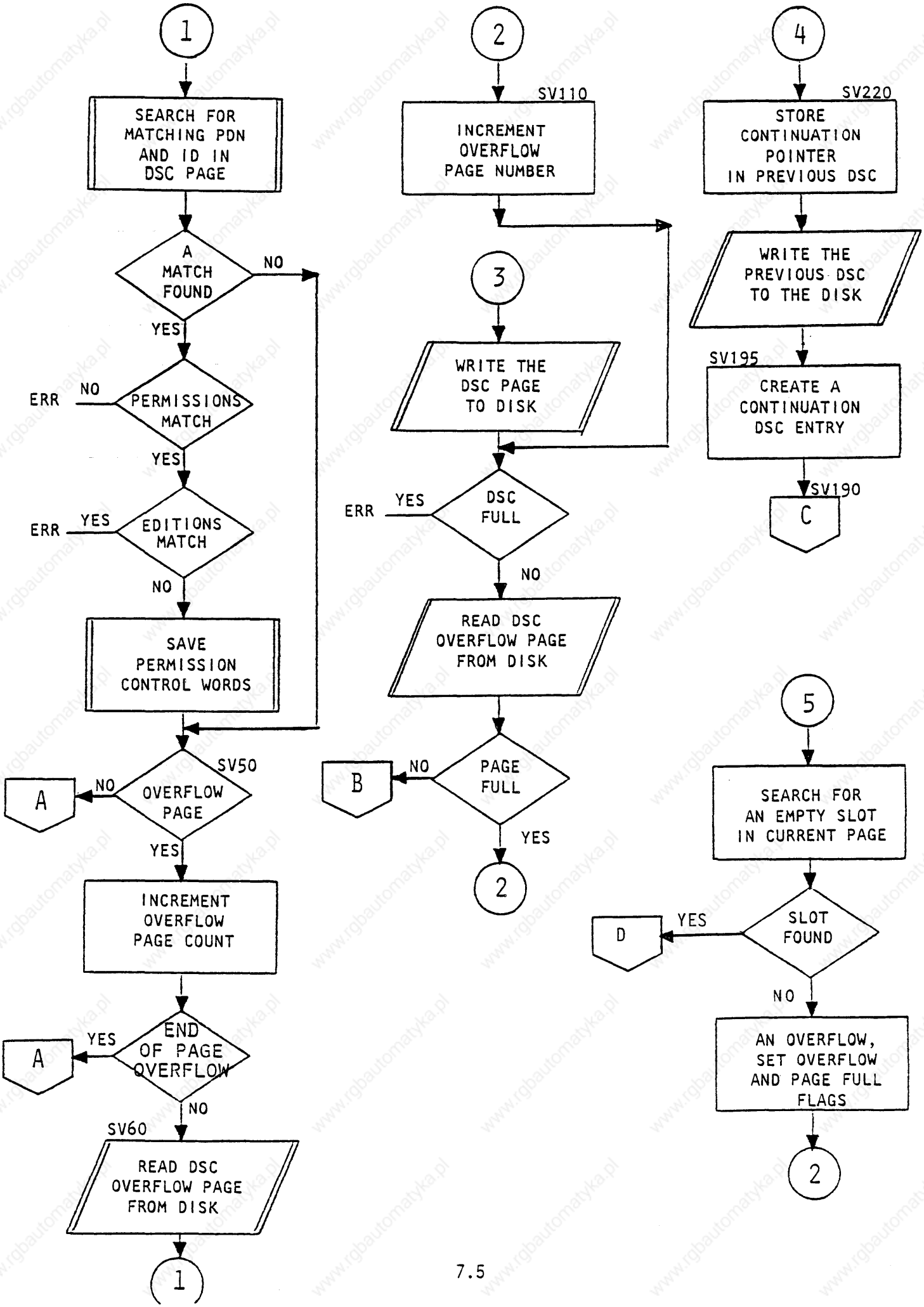
www.rgbautomatyka.pl

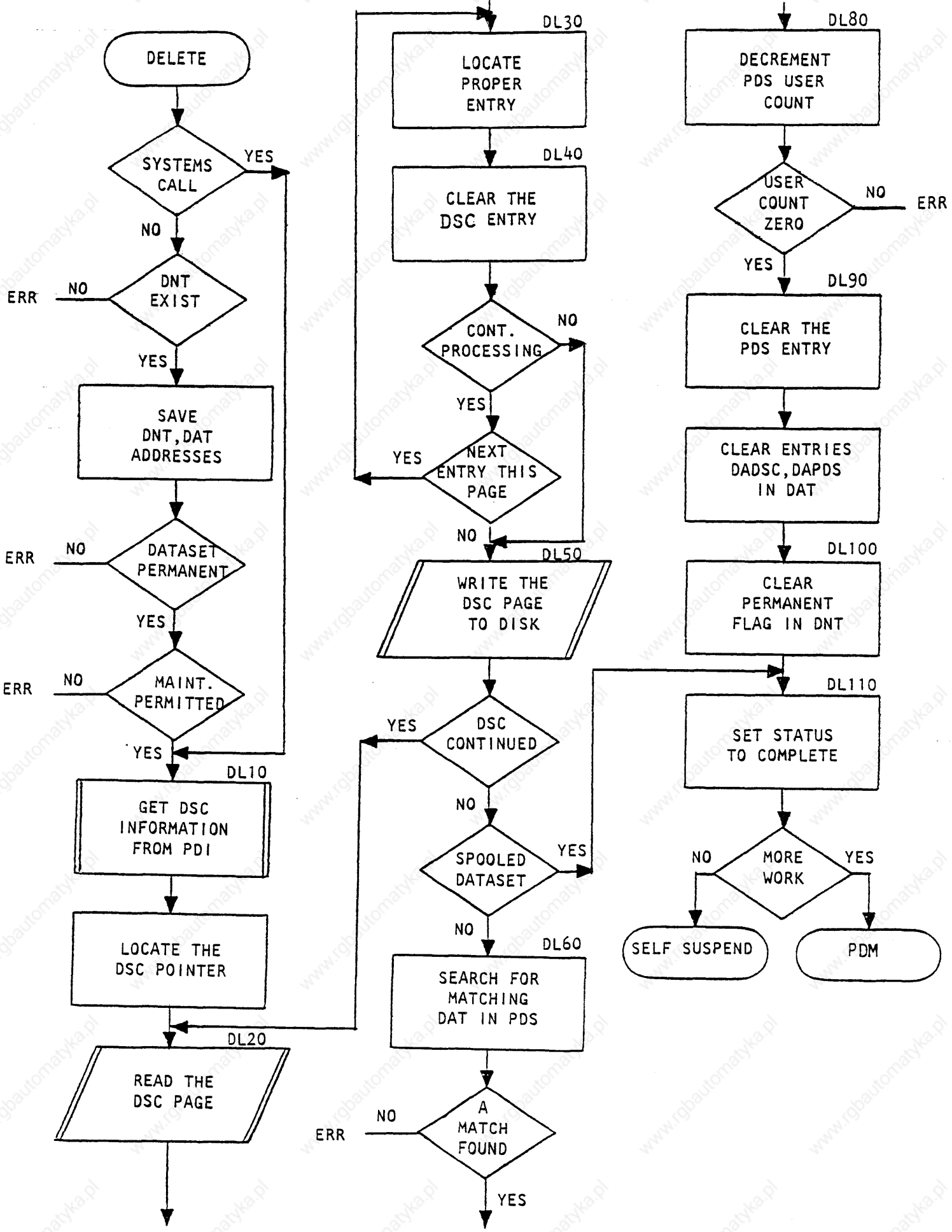

DAT Head
Page Map
DAT
Body
DAT
DNT
0
15
Page1
Page2
Page3
Pagen
DAT Entry
Header
Partitiona
Partitionb
Partitionc
Partitionx
DAT Entry
Partition
Header
AIs
Device
Partition
DAT structure



FUNCTIONS:

FCTBL


1
SEARCH FOR MATCHING PDN AND ID IN DSC PAGE
A MATCH FOUND
NO
YES
ERR
NO
PERMISSIONS MATCH
YES
ERR
YES
EDITIONS MATCH
NO
SAVE PERMISSION CONTROL WORDS
SV50
A
NO
OVERFLOW PAGE
YES
INCREMENT OVERFLOW PAGE COUNT
A
YES
END OF PAGE OVERFLOW
NO
SV60
READ DSC OVERFLOW PAGE FROM DISK
1
2
SV110
INCREMENT OVERFLOW PAGE NUMBER
3
WRITE THE DSC PAGE TO DISK
ERR
YES
DSC FULL
NO
READ DSC OVERFLOW PAGE FROM DISK
B
NO
PAGE FULL
YES
2
4
SV220
STORE CONTINUATION POINTER IN PREVIOUS DSC
WRITE THE PREVIOUS DSC TO THE DISK
SV195
CREATE A CONTINUATION DSC ENTRY
SV190
C
5
SEARCH FOR AN EMPTY SLOT IN CURRENT PAGE
D
YES
SLOT FOUND
NO
AN OVERFLOW, SET OVERFLOW AND PAGE FULL FLAGS
2


DELETE
SYSTEMS CALL
YES
NO
ERR
NO
DNT EXIST
YES
SAVE DNT,DAT ADDRESSES
ERR
NO
DATASET PERMANENT
YES
ERR
NO
MAINT. PERMITTED
YES
DL10
GET DSC INFORMATION FROM PDI
LOCATE THE DSC POINTER
DL20
READ THE DSC PAGE
DL30
LOCATE PROPER ENTRY
DL40
CLEAR THE DSC ENTRY
CONT. PROCESSING
NO
YES
YES
NEXT ENTRY THIS PAGE
NO
DL50
WRITE THE DSC PAGE TO DISK
YES
DSC CONTINUED
NO
SPOOLED DATASET
YES
NO
DL60
SEARCH FOR MATCHING DAT IN PDS
ERR
NO
A MATCH FOUND
YES
DL80
DECREMENT PDS USER COUNT
USER COUNT ZERO
NO
ERR
YES
DL90
CLEAR THE PDS ENTRY
CLEAR ENTRIES DADSC,DAPDS IN DAT
DL100
CLEAR PERMANENT FLAG IN DNT
DL110
SET STATUS TO COMPLETE
NO
MORE WORK
YES
SELF SUSPEND
PDM

JOB SCHEDULER (JSH)

JOB SCHEDULER (JSH)

- INITIATES JOB SCHEDULING
- MANAGES USER AREA MEMORY RESOURCES. (JTA-LA)
- MANAGES USER JOB SWAPPING.
- MANAGES ROLL-IN AND ROLL-OUT OF USER JOBS.
- INITIATED THROUGH CALLS BY OTHER TASKS.

JOB SCHEDULING

- JSH SELECTS JOBS FROM THE SYSTEM DATASET TABLE (DC=IN).
- JOBS ARE ENTERED ON THE JOB EXECUTION TABLE (JXT) BY CLASS, PRIORITY AND TIME OF SUBMISSION.
- A PRIORITY OF 0-15 MAY BE SPECIFIED ON THE JOB CARD.
- THERE CAN BE UP TO 63 ENTRIES ON THE JXT.
- JOBS ON THE JXT CONTEND FOR MEMORY BASED ON PRIORITY AND JOB SIZE. (EXCLUDING PRIORITY 0 JOBS).
- ONCE A WAITING JOB BECOMES THE HIGHEST MEMORY PRIORITY IT IS MOVED FROM THE DISK TO MEMORY.

USER JOB SWAPPING

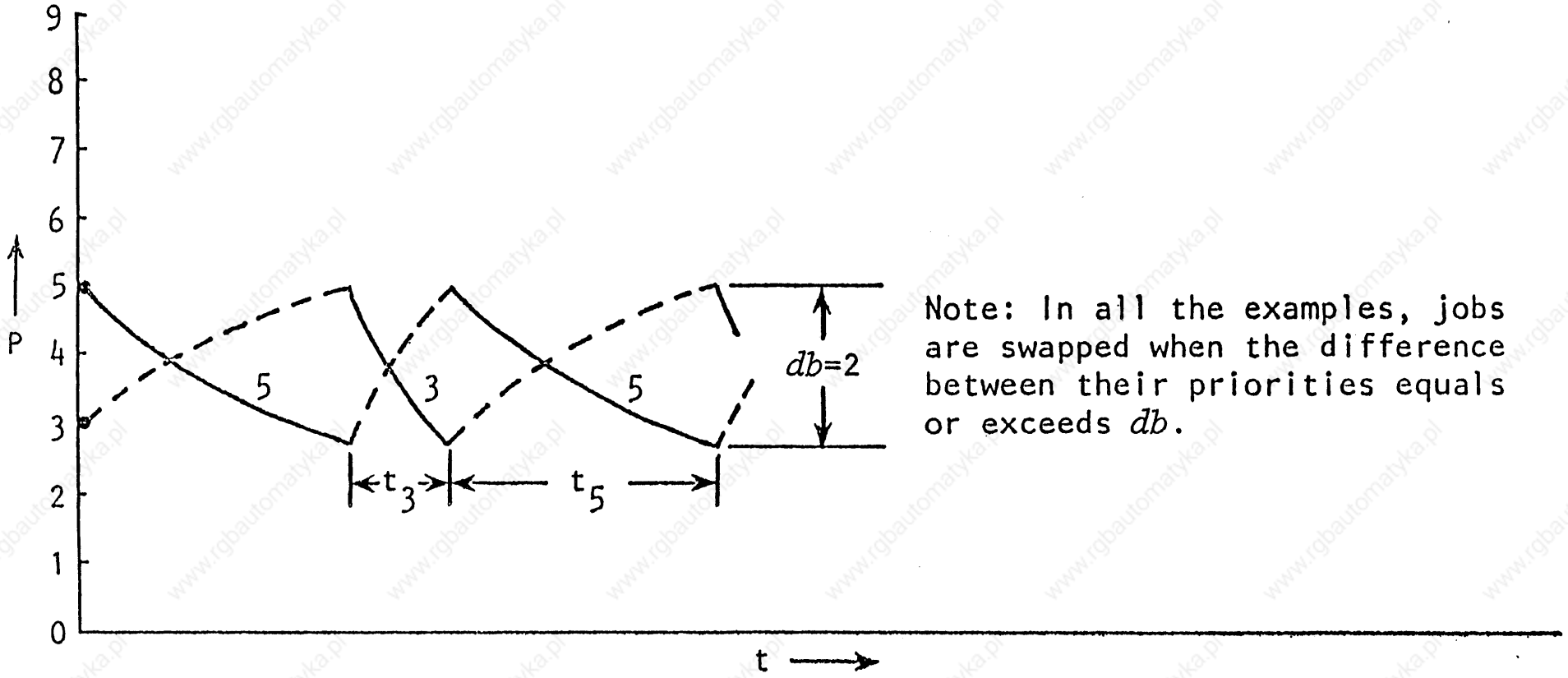
- ONCE IN MEMORY A JOB MUST CONTEND WITH OTHER JOBS IN MEMORY FOR CPU TIME.
- CPU PRIORITY FALLS WHEN A JOB IS CONNECTED TO THE CPU.
- CPU PRIORITIES ARE RECOMPUTED AFTER EACH SCHEDULING INTERVAL.
- CPU PRIORITY IS GIVEN TO JOBS THAT ARE STREAMING (DOING I/O).
- THE ALGORITHM FOR COMPUTING CPU PRIORITY IS ADJUSTABLE.

ROLL-OUT AND ROLL-IN OF JOBS

- MEMORY PRIORITY RISES WHILE A JOB IS WAITING FOR MEMORY
- MEMORY PRIORITY FALLS WHEN A JOB IS IN MEMORY AND EXECUTING
- HIGHER PRIORITY JOBS MAY PRE-EMPT MEMORY FROM LOWER PRIORITY JOBS, THEN:

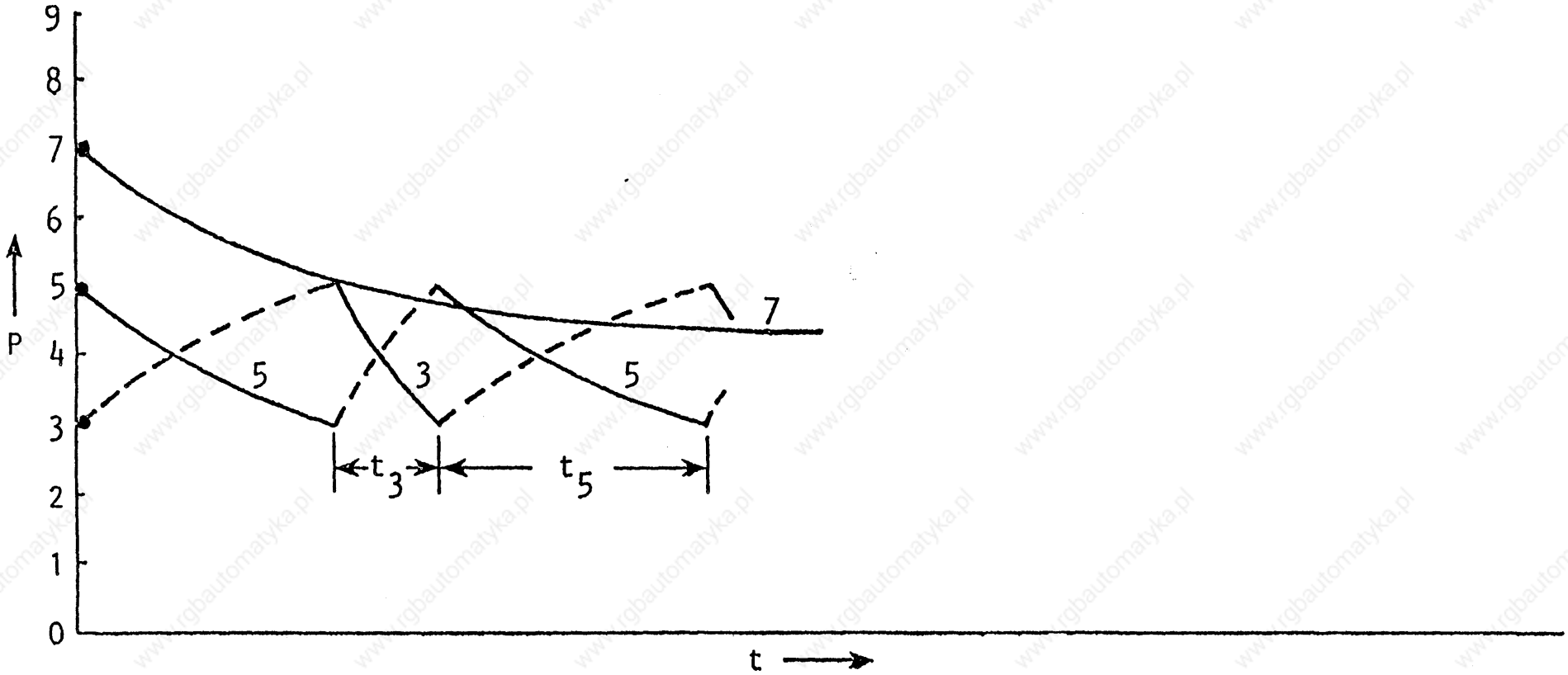
  THE LOWER PRIORITY JOB IS WRITTEN TO DISK (ROLLED-OUT).
  HIGHER PRIORITY JOB USES THE VACATED MEMORY.
  MEMORY MAY HAVE TO BE 'COMPACTED' SO HIGH PRIORITY JOB HAS A CONTIGUOUS AREA.

- ROLLED-OUT JOBS PRIORITY IS RISING SO EVENTUALLY IT WILL BE WRITTEN BACK TO MEMORY (ROLLED-IN).
- THE ALGORITHM FOR COMPUTING MEMORY PRIORITY IS ADJUSTABLE.
- ROLLED JOBS MAY BE CONTINUED OR RERUN AFTER A RESTART



Note: In all the examples, jobs are swapped when the difference between their priorities equals or exceeds *db*.

Showing two jobs which are too large to share memory. Their initial priorities (from the job cards) are 3 and 5. Note that the higher-priority job runs first and consistently enjoys a longer stay in memory, because its priority has asymptotes at P=2 and 8 (as opposed to 0 and 6 for the other job.)



Showing three jobs, only two of which can share memory at any one time. Except for the presence of the priority-7 job, this graph is identical to figure 3.8-2. The priority-7 job can never be forced out of memory because its priority never will be even as much as one unit below P=5, which is the maximum attainable by either of the other two jobs. (If its initial priority were 6 instead of 7, it still would never be forced out by the other two jobs.)

```
          SUBTITLE  'JOB SCHEDULER  (ROLLJOB)'
*****************************************************************************
**
*                          ----------------
*                          SUBROUTINE ROLLJOB
*                          ----------------
*
*  PURPOSE:
*           TO MAKE A REQUEST OF THE DISK QUEUE MANAGER, EITHER TO COPY
*           A JOB OUT ONTO ITS ROLLOUT DATASET OR TO READ THE JOB'S IMAGE
*           BACK INTO MEMORY.
*
*  ENTRY:
*           A4 = JXT-ENTRY ADDRESS.
*           A5 = JTA ADDRESS.
*           DNP (PROCESSING DIRECTION IN THE ROLLFILE'S DNT) IS ALREADY
*                SET -- TO 0 IF ROLLING IN, OR TO 1 IF ROLLING OUT.
*
*  EXIT:
*           I/O IS IN PROGRESS.
*
*  REGISTERS:
*           (A0-A2), (A6-A7), (S0-S2), (S6-S7) ARE DESTROYED
**
*****************************************************************************

ROLLJOB  =        *
         A7       W@JXDNT,A4
         A6       W@JXCJS,A4
         S1       A5              SET UP THE DNT:
         PUT,S1   S6&S7,DNBUF,A7    BUFFER ADDRESS = JTA ADDRESS,
         S2       A6                NUMBER OF BLOCKS = JOB SIZE/512.
         S2       S2>D'9
         PUT,S2   S6&S7,DNNBK,A7
         S1       A4              SUBMIT THE I/O REQUEST:
         S1       S1<D'40           LEFT-ADJUST THE JXT ADDRESS.
         S2       A7
         S1       S1!S2             INSERT THE DNT ADDRESS.
         A1       JSHID,0
         A2       DQMID,0
         S2       TRANSFER
         J        PUTREQ          LET PUTREQ RETURN TO THE CALLER.
```

USER AREA MANAGEMENT

- PROVIDES FOR ROLL-IN, ROLL-OUT RESOURCES
- PROVIDES FOR INITIAL JOB ENTRY TO MEMORY
- PROVIDES DYNAMIC ALLOCATION OF MEMORY TO USER JOBS
- PROVIDES DYNAMIC DEALLOCATION OF MEMORY TO USER JOBS
- USES A TABLE (MST) FOR DETERMINING USEABLE MEMORY SPACE
- USES A 'FIRST-FIT' METHOD FOR GAINING MORE MEMORY
- 'FIRST-FIT' METHOD:

  ALLOCATES MEMORY (BY BLOCK) BEGINNING AT LOW END OF MEMORY

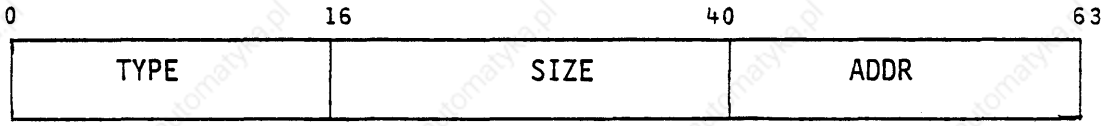
  MOVES JOBS (IF REQUIRED) BEGINNING AT HIGH END OF MEMORY

  ENCOURAGES LARGE BLOCKS OF FREE MEMORY AT THE HIGH END OF MEMORY

```
        SUBTITLE  'JOB SCHEDULER  (LIB@INIT)'
*****************************************************************************
**
*                          ---------------------
*                          SUBROUTINE LIB@INIT
*                          ---------------------
*
*  PURPOSE:
*           TO FIND THE MST ENTRY FOR AN ALLOCATED SEGMENT WHOSE ADDRESS
*           IS KNOWN, OR TO FIND THE MST ENTRY FOR THE FIRST ALLOCATED
*           SEGMENT ABOVE A GIVEN ADDRESS.
*
*  ENTRY:
*           A3 = ADDRESS OF AN ALLOCATED SEGMENT WHICH IS TO BE FREED,
*              OR ANY VALUE LESS THAN THAT ADDRESS AS LONG AS IT IS
*              GREATER THAN THE ADDRESS OF ANY LOWER ALLOCATED SEGMENT.
*
*  EXIT:
*           S0 = 0 IF NO ALLOCATED SEGMENT EXISTS WITH AN ADDRESS
*              THAT IS EQUAL TO OR GREATER THAN (A3).
*           A2 = (IF S0 IS NONZERO) THE OFFSET OF THE DESIRED MST ENTRY.
*           S2 = (IF S0 IS NONZERO) THE MST ENTRY ITSELF.
*
*  REGISTERS:
*           (A0-A2), (S0-S2) ARE DESTROYED.
**
*****************************************************************************

LIB@INIT =        *
         A2       -1              LET A2=0 TO FETCH THE FIRST MST ENTRY.
LIB1     =        *
         A2       A2+1            ADVANCE THE MST OFFSET.
         S2       B@MST,A2        FETCH AN MST ENTRY.
         S0       S2
         JSZ      LIB2            RETURN S0=0 IF NO MSADDR IS >= A3.
         S0       S2>D'64-N@MSTYPE
         A1       S2              A1 = MSADDR.
         A0       A1-A3           COMPARE MSADDR TO THE CALLER'S ADDRESS.
         JSZ      LIB1            RELOOP IF THIS SEGMENT ISN'T ALLOCATED.
         JAM      LIB1            LOOP UNTIL THE DESIRED SEGMENT IS FOUND.
LIB2     =        *
         J        B0              RETURN.
```

MEMORY SEGMENT TABLE - MST

The MST in STP memory contains a one-word entry for each segment of memory that has been allocated by the Job Scheduler plus additional entries that describe free segments. MST entries are stored in ascending order according to the beginning address of the segment (MSADDR). Any free space between two allocated segments is consolidated into a single entry. The last entry in the table is always followed by a zero word. To provide for the case where every allocated segment is surrounded by a free segment, the MST must have twice as many words in it as the maximum number of allocated segments, plus two more.



Memory Segment Table (MST) entry

Field	Word	Bits	Description
MSTYPE	0	0-15	Contains 0 if the segment is free; otherwise, it contains the JXT ordinal of the job to which the segment is allocated.
MSSIZE	0	16-39	Number of words in the segment. This is always a multiple of 1000_8.
MSADDR	0	40-63	STP-relative address of the first word in the segment. This is always a multiple of 1000_8.

```
        MEMORY SEGMENT ALLOCATION TABLE
0040753 000001001450000116000
0040754 000002001450000263000 000000012100000430000 000000000000000000000 000000000000000000000
            LOCATIONS 00040760 THROUGH 00041053 CONTAIN 000000000000000000000
0041054 000000000000000000000
```

```
        SUBTITLE  'JOB SCHEDULER  (MOVEMEM)'
**********************************************************************
**
*                          ------------------
*                          SUBROUTINE MOVEMEM
*                          ------------------
*
*  PURPOSE:
*           TO MOVE A BLOCK OF WORDS IN MEMORY.
*
*  ENTRY:
*           A3 = ADDRESS OF THE DESTINATION AREA.
*           A5 = ADDRESS OF THE FIRST WORD TO BE MOVED.
*           A6 = NUMBER OF WORDS IN THE DESTINATION AREA.
*           A7 = NUMBER OF WORDS TO BE MOVED.
*
*  EXIT:
*           THE NUMBER OF WORDS GIVEN BY MIN (A6,A7) IS MOVED FROM (A5)
*           TO (A3).  A2 AND S2 ARE SET UP FOR A SUBSEQUENT 'ERASEMEM'
*           CALL (WHICH WILL HAVE NO EFFECT IF S2 <= 0).
*
*  REGISTERS:
*           (V0), (B10-B11), (A0-A2), (S0-S3) ARE DESTROYED.
*           (T10,T11) ARE DESTROYED WHILE THE POST CALL IS PRESENT.
*
*  METHOD:
*           IF A3=A5, NO MOVING IS DONE, BUT A2 AND S2 ARE SET FOR
*           ERASEMEM ANYWAY.  IF A3<A5, MOVING COMMENCES WITH THE
*           LOWER ADDRESSES;  IF A3>A5, IT BEGINS AT THE HIGHER ONES.
*           64 ELEMENTS ARE MOVED AT A TIME, UNLESS THE STORE WOULD
*           OVERFLOW THE DESTINATION AREA --- IN WHICH CASE A FINAL
*           PASS IS MADE WITH A REDUCED VL.
**
**********************************************************************

MOVEMEM  =       *
         B10     A3
         B11     A5
         A0      A6-A7          TOTAL MOVE LENGTH (IN S3) = MIN (A6,A7).
         A1      A6-A7
         S3      A7
         JAP     MVM1
         S3      A6
MVM1     =       *
         S2      A1             SET UP S2 AND A2 FOR A SUBSEQUENT CALL TO
         A2      A3+A7            ERASEMEM (WILL BE A NO-OP IF S2<=0)

         A0      A3-A5          COMPARE THE TWO GIVEN ADDRESSES.
         S1      <6
         A1      ZS0            INITIAL VECTOR LENGTH = 64.
         S0      #S1&S3
         JAZ     MVM9           DO NOTHING IF A3=A5.
         JSM     MVM9           DO NOTHING IF S3<0.

         T10     S2             BUILD ARGUMENTS FOR A POST CALL.
         T11     S7
         S1      A5             FROM-ADDRESS
         S1      S1<D'24
         S2      A3             TO-ADDRESS
         S2      S2<D'24
         S7      A7             FROM-LENGTH
         S1      S1|S7
         S7      A6             TO-LENGTH
         S2      S2|S7
         S7      'MV'R          IDENTIFIER
         S7      S7<D'48
         S1      S7|S1
         POST    32,S1,S2       POST THE MOVE ARGUMENTS
         S2      T10              AND RESTORE REGISTERS.
         S7      T11
```

```
                                                                        S002.1303
        JSZ     MVM4            DO ONLY THE FINAL PASS IF S3<64.          S.20531
        JAP     MVM6            MOVE FROM THE TOP DOWN IF A3>A5.          S.20532
                                                                          S.20533
MVM2    =       *                                                         S.20534
        VL      A1              SET VL TO S4 (OR, ON FINAL PASS, TO RESIDUE).  S.20535
        S1      A1                                                        S.20536
                                                                          S.20537
MVM3    =       *                                                         S.20538
        A0      A5              SET ADDRESS FOR VECTOR LOAD.              S.20539
        S3      S3-S1           S3 = WORDS STILL TO BE MOVED AT LOOP'S END.  S.20540
        A5      A5+A1           ADVANCE VECTOR-LOAD POINTER.              S.20541
        V0      ,A0,1           LOAD                                      S.20542
        A0      A3              SET ADDRESS FOR VECTOR STORE.             S 20543
        S0      S3-S1           S0 WILL BE NEGATIVE WHEN S3<64 (OR, ON FINAL  S.20544
        A3      A3+A1             PASS, WHEN S3=0).                       S.20545
        ,A0,1   V0              STORE.                                    S.20546
        JSP     MVM3            RELOOP WHILE S3 >= 64.                    S.20547
                                                                          S.20548
MVM4    =       *                                                         S.20549
        S0      S3              0 <= S3 < 64:  WE ARE DONE IF S3=0.       S.20550
        A1      S3                                                        S.20551
        JSN     MVM2            IF S3>0, GO BACK TO MAKE A FINAL PASS.    S.20552
                                                                          S.20553
        J       MVM9            ELSE RETURN.                            S002.1304
                                                                          S.20556
*       HERE IF MOVING FROM THE TOP DOWN INSTEAD OF FROM THE BOTTOM UP.   S.20557
                                                                          S.20558
MVM6    =       *                                                         S.20559
        A1      S3              START A5 AND A3 AT THEIR HIGHEST VALUES.  S.20560
        A5      A5+A1                                                     S.20561
        A3      A3+A1                                                     S.20562
        A1      ZS0             RESTORE A1=64.                            S.20563
                                                                          S.20564
MVM7    =       *                                                         S.20565
        VL      A1              SET VL TO S4 (OR, ON FINAL PASS, TO RESIDUE).  S.20566
        S1      A1                                                        S.20567
                                                                          S.20568
MVM8    =       *                                                         S.20569
        A0      A5-A1           SET ADDRESS FOR VECTOR LOAD.              S.20570
        S3      S3-S1           S3 = WORDS STILL TO BE MOVED AT LOOP'S END.  S.20571
        A5      A5-A1           BACK UP VECTOR-LOAD POINTER.              S.20572
        V0      ,A0,1           LOAD.                                     S.20573
        A0      A3-A1           SET ADDRESS FOR VECTOR STORE.             S.20574
        S0      S3-S1           S0 WILL BE NEGATIVE WHEN S3<64 (OR, ON FINAL  S.20575
        A3      A3-A1             PASS, WHEN S3=0).                       S.20576
        ,A0,1   V0              STORE.                                    S.20577
        JSP     MVM8            RELOOP WHILE S3 >= 64.                    S.20578
                                                                          S.20579
        S0      S3              0 <= S3 < 64:  WE ARE DONE IF S3=0.       S.20580
        A1      S3                                                        S.20581
        JSN     MVM7            IF S3>0, GO BACK TO MAKE A FINAL PASS.    S.20582
                                                                          S.20583
MVM9    =       *                                                       S002.1305
        A3      B10                                                     S002.1306
        A5      B11                                                     S002.1307
        J       B0              RETURN.                                   S.20584
```

JOB SCHEDULER JOB FLOW

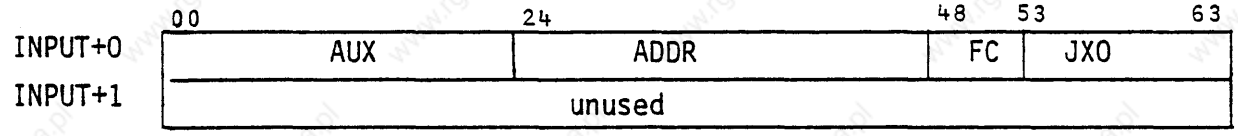
- JOB DATASET ENTERS THE SYSTEM
- STATION CALL PROCESSOR (SCP) HAS THE JOB SAVED ON DISK
- SCP HAS AN ENTRY MADE IN THE SYSTEM DATASET TABLE (SDT), AND DATASET CATALOG (DSC)
- JOB IS PLACED IN JOB EXECUTION TABLE (JXT) ACCORDING TO CLASS/ PRIORITY AND TIME OF SUBMISSION
- ONCE ON THE JXT THE JOB CONTENDS FOR MEMORY
- JOB MEMORY IS ERASED WHEN JOB INITIATED (I@ERASE)
- JOB IN MEMORY CONTENDS FOR CPU TIME
- JOB IN MEMORY MAY BE ROLLED OUT
- WHEN A JOB COMPLETES, ITS OUTPUT DATASETS ARE PLACED IN THE OUTPUT QUEUE IN SDT AND MADE PERMANENT
- WHEN FRONT-END ACKNOWLEDGES RECEIPT OF A DATASET THE DATASET CATALOG ENTRY IS REMOVED

Calling Sequence

JSH can be invoked from any other task by calling either TSKREQ or PUTREQ with the following instruction sequence:

Location	Result	Operand
	A1	calling task's ID
	A2	JSHID,0
	S1	function code (already shifted)
	S2	job's offset in JXT
	S1	S1!S2
[	S2	address if any]
[	S2	S2<D'16
[	S1	S1!S2]
[	S2	auxiliary information if any]
[	S2	S2<D'40
[	S1	S1!S2]
	R	TSKREQ or PUTREQ

Input register format:



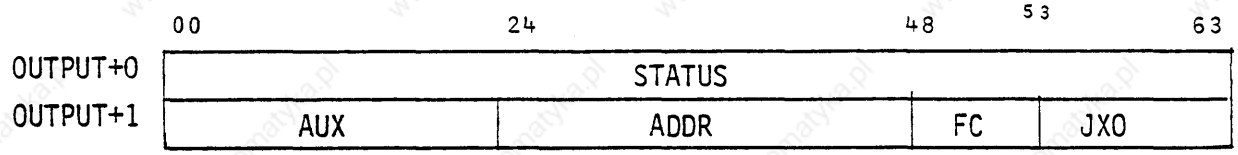
AUX Auxiliary information; unused by JSH. (Any value the caller places in INPUT+0 is returned verbatim in OUTPUT+1.)

ADDR Word address relative to the beginning of STP of an additional word or list of words if needed to fully specify the call.

FC Function code; use equated labels of the form J$XXXX, selected from table

JXO JXT offset for the job in question.

Output register format:



STATUS Status of requested function.

0 Requested function completely accomplished.

≠0 Error or system is unable to fulfill request completely.

PARTIAL JOB SCHEDULER FUNCTIONS

FUNCTIONS:
JSREQTAB

Function Code	Input Parameters	Function
—no input required—		Fill up JXT with jobs from SDT.
J$ABORT	JXO,CODE	Abort a job.
J$DELETE	JXO	Release all space allocated to a job.
J$RERUN	JXO	Same as J$DELETE but reinitiate a job.
J$ALLOC	JXO,ADDR	Allocate or release memory for a job.
J$IOSUSP	JXO	Suspend a job until an I/O request is done.
J$IODONE	JXO	Resume an I/O suspended job.
J$DELAY	JXO,ADDR	Suspend a job for a given time.
J$DELAYK	JXO,ADDR	Same as J$DELAY but keep the job in memory.
J$AWAIT	JXO,ADDR	Suspend a job until a given event occurs.
J$SUSP	JXO	Suspend a job momentarily; system initiated.
J$SUSPK	JXO	Same as J$SUSP but keep the job in memory.
J$REMK	JXO	Lift the "keep in memory" restriction.
J$RESUME	JXO	End momentary suspension.
J$STOP	JXO	Suspend a job indefinitely; operator action.
J$STPALL	JXO	Suspend all jobs indefinitely; operator action.
J$START	JXO	End an indefinite suspension for a job; operator action.
J$STRALL	JXO	End an indefinite suspension for all jobs; operator action.
J$CLEAR	JXO	Force the end of a job's suspension.
J$INDEX	JXO	Mark the job irrecoverable.
J$SHTDWN	JXO	Idle down job activity in preparation for a system interruption.
J$RCVR	JXO	Lift the suspension from jobs suspended by a J$SHTDWN or system interruption.

JOB STATUS AND STATE CHANGES

The status field

The 23-bit status field (JXSTAT) in each job's JXT entry is described in table 3.8-2. The bits labeled Q, R, X, I, U, L, S, O, and M are mutually exclusive. Generally, whichever one of them is set determines the job's state. The other bits modify the job's state.

If all of bits 3 through 22 are zero, the job is said to be waiting to be connected to the CPU (state W).

Table 3.8-2. Status bit assignments

Bit position in JXSTAT	Bit name	Corresponding job state	Interpretation (when bit is set)
0	K	*S*	Keep this job in memory; don't roll it out.
1	A	*any*	Abort pending; reason given in JXEPC.
2	H	*O*	Holding operator or shutdown suspension until RN is set.
3	O	*O*	Suspended (indefinitely) by operator.
4	S	*S*	Suspended (momentarily) by system.
5	T	*S*	Suspended until a given time elapses.
6	E	*S*	Suspended until a given event occurs.
7	M	*M*	Memory allocation is pending.
8	Q	*Q*	Queued up; waiting to be initiated.
9	R	*R*	Rolled out. The M bit may also be set.
10	X	*X*	Executing.
11	I	*I*	Dormant pending recall on I/O completion.
12	C	*any*	Rerun request in process.
13	D	*any*	Delete request in progress.
14	U	*U*	Unloading from memory to roll file.
15	L	*L*	Loading into a new memory area.
16	P	*U or L*	Unload or load initiation is pending.
17	Y	*M, Q or R*	Waiting for memory liberation.
18	Z	*M, Q or R*	Waiting for memory compaction.
19	B	*S*	Suspended (indefinitely) by recovery.
20	V	*I*	Waiting on INDEX write completion.
21	F	*I*	Waiting on rollfile write completion.
22	N	*M, Q or R*	Not in memory.

CPU SWAPPING

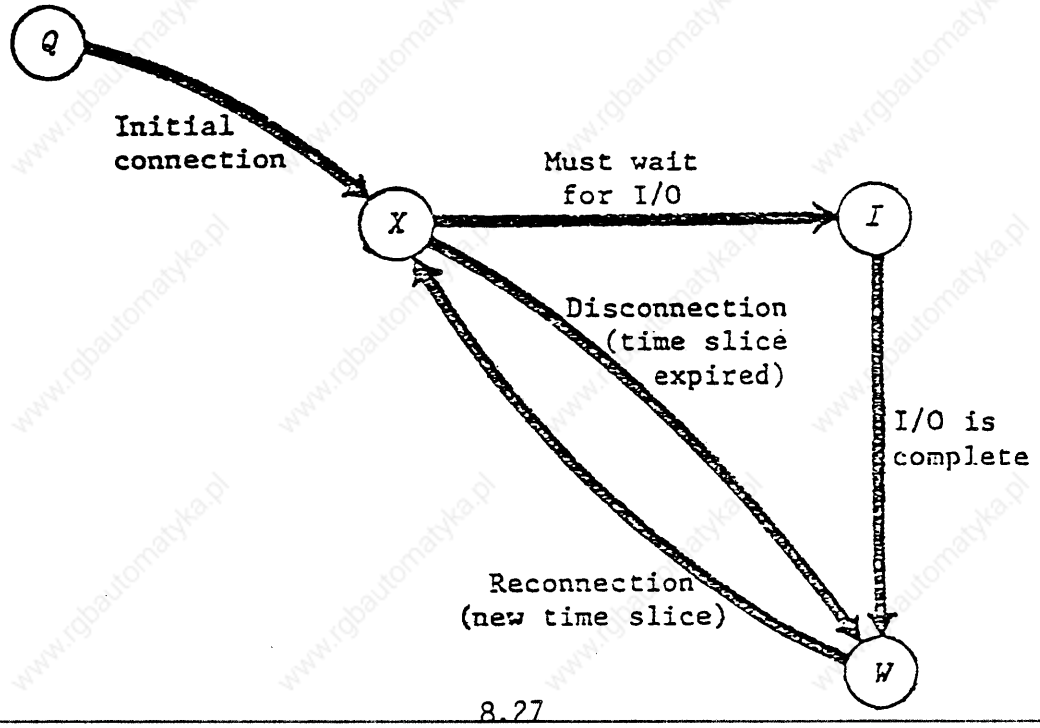
Q ⟶ X — A JOB IS QUEUED IN THE JOB EXECUTION TABLE (JXT) WAITING MEMORY. THE JOB IS BROUGHT INTO MEMORY AND BEGINS EXECUTION

X ⟶ I — THE JOB IS SUSPENDED PENDING COMPLETION OF I/O. THE CPU IS AVAILABLE FOR USE BY ANOTHER JOB.

I ⟶ W — THE I/O COMPLETES. THE JOB MAY WAIT FOR MORE CPU TIME.

W ⟶ X — JSH RECONNECTS THE WAITING JOB BECAUSE OF ITS HIGH PRIORITY

X ⟶ W — THE JOB'S TIME SLICE EXPIRES. THE CPU IS AVAILABLE FOR USE BY ANOTHER JOB.



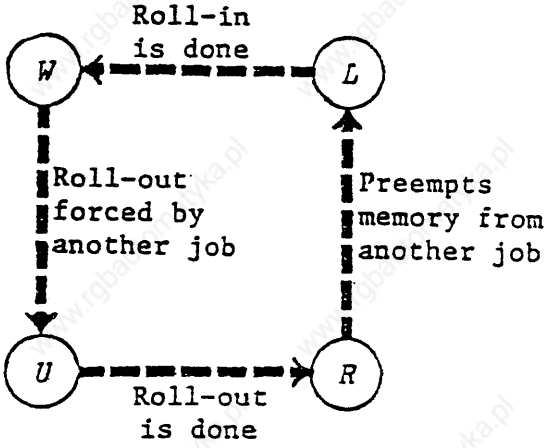
MEMORY SWAPPING

W→U I/O IS INITIATED FOR ROLL-OUT OF THE WAITING JOB.

U→R THE JOB IS ROLLED-OUT. THE JOB'S MEMORY IS AVAILABLE.

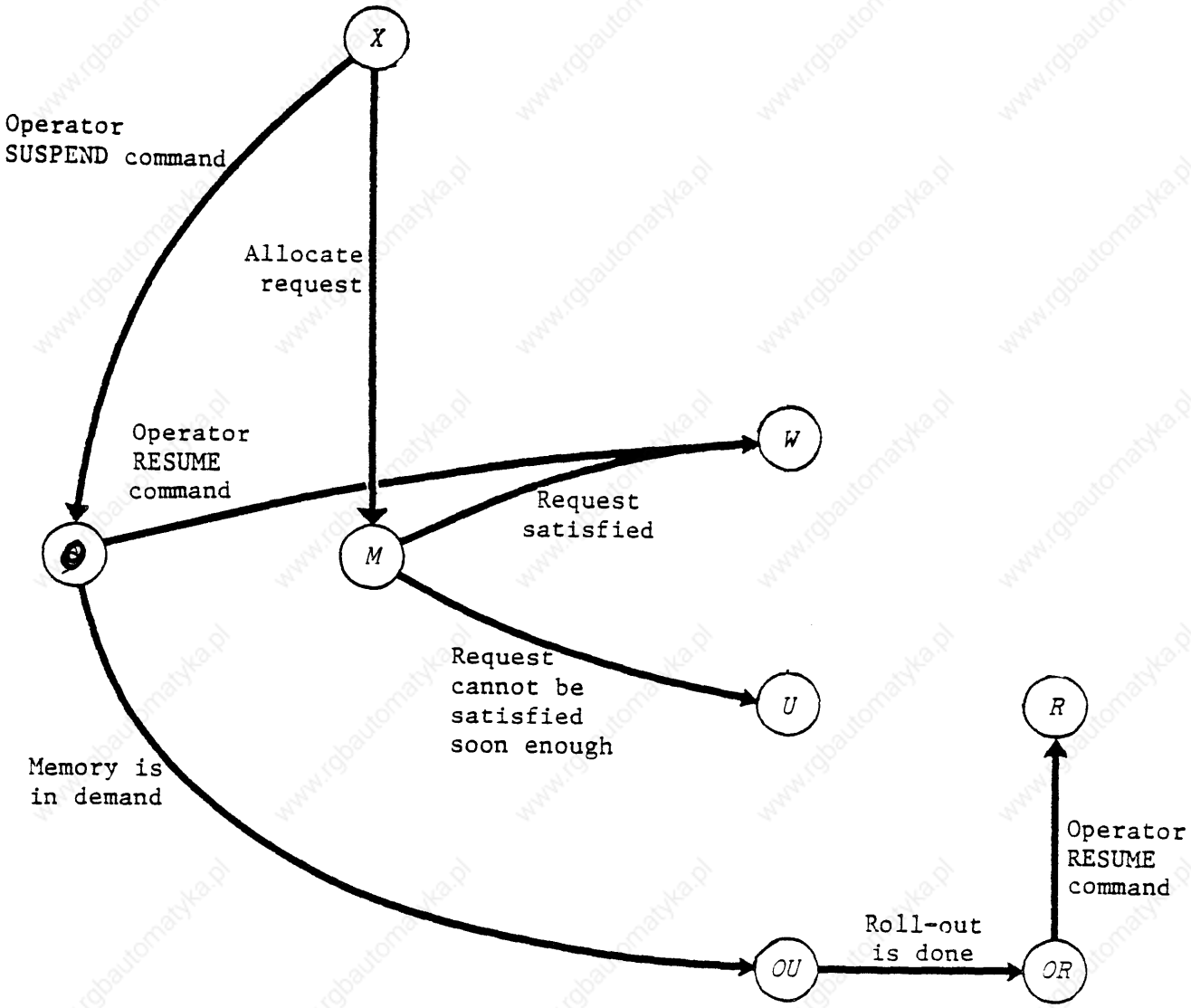
R→L I/O IS INITIATED FOR ROLL-IN OF THE JOB. MEMORY HAS BEEN ALLOCATED.

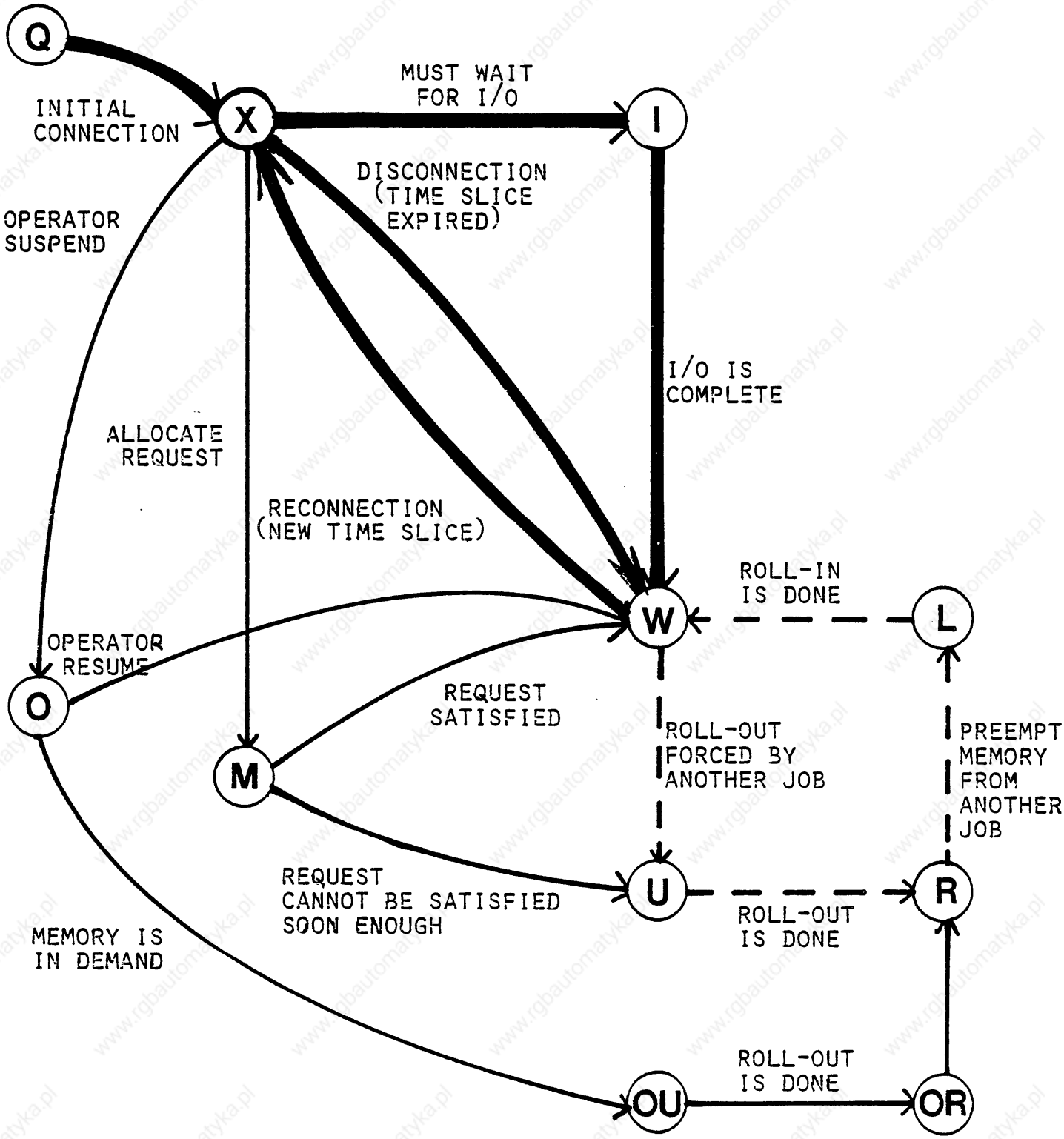
L→W THE JOB IS ROLLED-IN. THE JOB MAY WAIT FOR CPU TIME.



JOB SUSPENSION

	X → M	THE EXECUTING JOB MAKES A REQUEST FOR MORE MEMORY. IF REQUEST CANNOT BE SATISFIED THE JOB MAY BE ROLLED-OUT.
	M → W	THE ALLOCATION REQUEST WAS SATISFIED. THE JOB WAS NOT ROLLED OUT SO NOW MAY WAIT FOR CPU TIME
	M → U	I/O IS INITIATED FOR ROLL-OUT OF THE JOB REQUESTING MORE MEMORY
X → O	X → S	THE EXECUTING JOB REQUESTS SUSPENSION. JOB IS DISCONNECTED FROM CPU AND LIABLE TO BE ROLLED OUT.
O → OU	S → SU	I/O IS INITIATED FOR ROLL-OUT OF THE SUSPENDED JOB.
OU → OR	SU → SR	THE SUSPENDED JOB IS ROLLED-OUT. THE JOB'S MEMORY IS AVAILABLE.
OR → R	SR → R	THE SUSPENDED JOB IS REACTIVATED.
O → W	S → W	THE SUSPENDED JOB IN MEMORY IS REACTIVATED.


X
Operator
SUSPEND command
Allocate
request
Operator
RESUME
command
W
Request
satisfied
M
Request
cannot be
satisfied
soon enough
U
R
Memory is
in demand
Operator
RESUME
command
Roll-out
is done
OU
OR


Q
INITIAL CONNECTION
X
MUST WAIT FOR I/O
I
DISCONNECTION (TIME SLICE EXPIRED)
OPERATOR SUSPEND
I/O IS COMPLETE
ALLOCATE REQUEST
RECONNECTION (NEW TIME SLICE)
ROLL-IN IS DONE
W
L
OPERATOR RESUME
O
REQUEST SATISFIED
ROLL-OUT FORCED BY ANOTHER JOB
PREEMPT MEMORY FROM ANOTHER JOB
M
U
R
REQUEST CANNOT BE SATISFIED SOON ENOUGH
ROLL-OUT IS DONE
MEMORY IS IN DEMAND
ROLL-OUT IS DONE
OU
OR

USER EXCHANGE PROCESSOR (EXP)

www.rgbautomatyka.pl

USER EXCHANGE PROCESSOR (EXP)

- PROCESSES ALL USER NORMAL ACTION REQUESTS.
- PROCESSES ALL USER ERROR EXITS.
- PROCESSES JSH REQUESTS TO INITIATE OR ABORT A USER JOB.
- ALL REQUESTS MADE THROUGH THE EXCHANGE PROCESSOR REQUEST WORD (JTEP) IN A USER JTA. JTEP IS WORD 67_8.

Field	Bits	Description
JTEPX	0-1	User exit 2 Normal exit 1 Error exit or execution error 0 Not user exit
JTEPC	2	Continuation flag
JTEPJ	3	Job Scheduler request flag
JTEPM	4	JTA expansion request flag
JTEPF	7-15	Exchange package flags
JTEPP	16-39	P register for errors
JTEPA	40-63	EXP Continuation address

USER NORMAL EXIT

- USER ISSUES A SYSTEM ACTION REQUEST WHICH SETS S REGISTERS AND ISSUES AN EX INSTRUCTION

  S0-DESIRED CALL TABLE OFFSET (FUNCTION CODE)

  S1-CONVENTION -- AN ADDRESS OF A USER TABLE (OPTIONAL)

  S2-FURTHER OPTIONAL ARGUMENT

- WHEN THE REQUEST IS COMPLETE THE USER'S S0-REGISTER IS SET

  S0-ZERO IF REQUEST COMPLETED NORMALLY

  S0-ERROR CODE OR JOB ABORTS

- EXP EXAMINES THE USERS JXT ENTRIES. A JXT ENTRY POINTS TO THE USERS JTA

- THE USERS JTA CONTAINS A COPY OF THE EXCHANGE PACKAGE

- EXP EXAMINES JTEP (WORD 67_8) AND THE USERS S0 REGISTER TO DETERMINE REQUEST

```
        *       NORMAL EXCHANGE CALLS AND PROCESSING ADDRESSES                          S.1182
        *                                                                               S.1183
        CALL    =       W.*                                                             S.1184
                CON     0*X24+ADV       00 ADVANCE PROGRAM                              S.1185
                CON     0*X24+ABT       01 ABORT JOB                                    S.1186
                CON     1*X24+DAT       02 GET DATE - MM/DD/YY                          S.1187
                CON     1*X24+TIM       03 GET TIME - HH:MM:SS                          S.1188
                CON     D'8*X24+MSG     04 PUT MESSAGE INTO USERS DAYFILE               S.1189
                CON     2*X24+RCL       05 SUSPEND IF I/O ACTIVE                        S.1190
                CON     0*X24+TRM       06 TERMINATE JOB                                S.1191
                CON     1*X24+SSW       07 SET SENSE SWITCHES                           S.1192
                CON     2*X24+OPN       10 OPEN DATASET                                 S.1193
                CON     1*X24+MEM       11 MEMORY REQUEST                               S.1194
                CON     1*X24+LBN       12 GET LAST BLOCK NUMBER                        S.1195
                CON     2*X24+CLS       13 CLOSE DATASET                                S.1196
                CON     D'8*X24+DNT     14 CREATE/SENSE DNT                             S.1197
                CON     1*X24+MDE       15 SET MODE                                     S.1198
                CON     D'8*X24+GNS     16 GET NEXT CONTROL CARD                        S.1199
                CON     0*X24+EXU       17 EXECUTE AN OBJECT CODE FILE                  S.1200
                CON     D'8*X24+RLS     20 RELEASE DATASET                              S.1201
                CON     0*X24+PDM       21 PERMANENT DATASET MANAGER REQUEST            S.1202
                CON     D'16*X24+RDC    22 READ DISK CIRCULAR                           S.1203
                CON     D'16*X24+WDC    23 WRITE DISK CIRCULAR                          S.1204
                CON     2*X24+GRN       24 GET SYSTEM REVISION LEVELS                   S.1205
                CON     D'16*X24+DPS    25 DISPOSE DATASET                              S.1206
                CON     1*X24+JDA       26 GET JULIAN DATE - YYDDD                      S.1207
                CON     1*X24+JTI       27 ACCUMULATE JOB CPU TIME                      S.1208
                CON     D'8*X24+ACT     30 GET ACCOUNTING INFORMATION FROM JTA          S.1209
                CON     0*X24+SPS       31 SET P AND SUSPEND USER                       S.1210
                CON     1*X24+CSW       32 CLEAR SENSE SWITCH                           S.1211
                CON     1*X24+TSW       33 TEST SENSE SWITCH                            S.1212
                VWD     D'40/LE@DSP,D'24/BIO  034 BUFFERED I/O REQUEST                  S.1213
                CON     1*X24+DLY       35 DELAY JOB X NUMBER OF MILLISECONDS           S.1214
                CON     0*X24+AQR       36. ACQUIRE DATASET FROM FRONT END          C0805.239
                CON     0*X24+NRN       37 DISABLE NO-RERUN CHECK                   E1327.317
                CON     0*X24+RRN       40 OVERRIDE NOT-RERUNNABLE FLAG             E1327.318
40      CALMAX  =       W.*-CALL-1      CALL LIMIT                                      S.1215
41      CALMAX1 =       CALMAX+1                                                    D1032.52
                                                                                        S.1216
```

```
        EXCHANGE PROCESSOR CALL TABLE
0045175 0000000000000000313162 0000000000000000313632 0000000000000100321101
0045200 0000000000000100327712 0000000000001000325512 0000000000000200327377 0000000000000000324366
0045204 0000000000000100327661 0000000000000200325763 0000000000000100325364 0000000000000100324250
0045210 0000000000000200320401 0000000000001000321754 0000000000000100324314 0000000000001000323667
0045214 0000000000000000323067 0000000000001000327437 0000000000000000326641 0000000000002000330115
0045220 0000000000002000330120 0000000000000200324022 0000000000002000322540 0000000000000100324036
0045224 0000000000000100324230 0000000000001000313106 0000000000000000327624 0000000000000100321050
0045230 0000000000000100327725 0000000000002000317724 0000000000000100321730 0000000000000000316730
0045234 0000000000000000325734 0000000000000000327544
```

```
0136000 0411252465440000000000 0000000000063551645625 0000000000023663324313 0000000000000073314630 BUSY          K
0136004 0000000000000000000055 0000000000000000000007 0000000000000000000000 0000000000000000000000          -
              LOCATIONS 00136010 THROUGH 00136017 CONTAIN 0000000000000000000000
0136020 0000000000104100000212 0000000014500000000000 0000000030301600000000 0000001500000000000000     !                    4
0136024 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000213
0136030 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0136034 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000001
0136040 1777777777777777777777 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00136044 THROUGH 00136057 CONTAIN 0000000000000000000000
0136060 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000063551645625                    K
0136064 0000000000000000000000 0425322255254201321403 0000000000000000324715 0000000000000000000000        EZJ
0136070 0000000000000000000000 0000000000000000000000 0000000000000000000000 0421072104340000036026                DGDG  <
0136074 0475202125110125047522 0400615063140064344400 0400505667546174000000 0000040000000000000000 OPERATOR@1 3    @(  1
0136100 0000000000000000001020 0000000000000000000213 0000000000000000000000 0000000000000000000000
              LOCATIONS 00136104 THROUGH 00137317 CONTAIN 0000000000000000000000
0137320 0221032460000000000001 0000020000000000002000 0000000000000100002000 0000000000000000002010 $CS
0137324 0000000000000000003000 1000000000000000002011 0000000000000000000000 0000400000000000000000
0137330 1000000000000000002007 0000000000000000000000 0000000033651400117660 0000000000000000000000                L
0137334 0000000000000000352545 0000000012240600036026 0020000000700500007017 1000000000000000000000          <
0137340 0221142364340000000001 0000010000000000003000 0000000000000000003152 0000000000000000003000 $LOG
0137344 0000000000000000004000 1040000000000000003151 0000000000000000000000 0000000000000000000000
0137350 1040000000000000003140 0000000000000000000000 0000000035636600000712 0000000000000000000000        `
0137354 0000000000000000000000 0000000012242400036026 0000000000000000000000 0000000000000000000000          <
0137360 0461042440000000000000 0400000000000000000200 0000000000001100000200 0000000000001000010450 LDR     @                    (
0137364 0000000000000000011200 1000000100001000010450 0000000000000000000000 0000000000000000000000          (
0137370 0000000100001000010447 0000000000000000000000 0000000033630700125200 0000000000000000000000        ,
0137374 0000000000000000352615 0000000011742000036026 0062000001515000017414 0221202044700000000000          <        $PBN
0137400 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00137404 THROUGH 00137417 CONTAIN 0000000000000000000000
0137420 0461042440000000000000 0000002464140100021140 0000040000000500132200 1000200000000000000000 LDR       SC  "`
0137424 0000000035213400001360 0000000000000000000000 0000000100000001100004 0000000000000000000011     \
0137430 0000000001100000116100 0000000000000000000000 0000000000000000000000 0000000000000000000000       @
0137434 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0137440 0000000000000000343413 0000000000000000116000 0000000000000000036026 0000000000000000333175                  <
0137444 0000000000000000004520 0000000000000000122000 0000000000000000004000 0000000000000077777777       P
0137450 0000000000000000000000 0000000000000000000000 0007070260011751604001 0000000000000000000001                O
0137454 0000010000000000000000 0377777777777777700000 0000000000000000000000 0000000000000000333175        ?
0137460 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00137464 THROUGH 00137657 CONTAIN 0000000000000000000000
0137660 0461042442700000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000 LDR.
              LOCATIONS 00137664 THROUGH 00137673 CONTAIN 0000000000000000000000
0137674 0000000001451600136000 0000030012200000121600 0000060000000000121564 0000000000000000000000     N
```

```
          LOCATIONS 00137700 THROUGH 00137737 CONTAIN 000000000000000000000
0137740 000000000000000340546 000000000000000000000 000000000000000000000 000000000000000004002
0137744 000000000000000122520 000000000000000242660 000000000000000000000 000000000000000000000        P      E
0137750 000000000000000000000 000000000000000000000 000000000000000000000 000000000000000000000
0137754 000000000000000130572 000000000000000000000 000000000000000000000 000000000000000000000
          LOCATIONS 00137760 THROUGH 00137777 CONTAIN 000000000000000000000
0140000 000000000000000000002 045117204241122343650 2 052523262261241220000 102000000000000000001          JOB(JN=BUSY,T)
0140004 041506250270000000000 104000000000000000001 046104244270000000000 104000000000000000001 CFT.              LDR.
0140010 042530222520560000000 103000000000000000001 042530222520560000000 103000000000000000000 EXIT.             EXIT.
0140014 160000000000000000001 015444222200751003000 0 101000000000000000002 020060140304010044440          #I = 0          001  I
0140020 036440142000000000000 105000000000000000002 015444216475242362006 0 030061000000000000000 = 1             #GOTO 001
0140024 106000000000000000001 015444212471040000000 0 103000000000000000000 160000000000000000000          #END
0140030 170000000000000000000 000000000000000000000 000000000000000000000 000000000000000000000
          LOCATIONS 00140034 THROUGH 00140777 CONTAIN 000000000000000000000
0141000 000000000000000000016 020040100200401002004 0 310611643046016431062 020040100200401003005 6                21:10:22      0.
0141004 030060140300401002004 0 041523240200401002004 0 020040100200401002004 0 041522202544551422004 0 0000    CSP             CRAY-1
  1010 051505244445012302646 1 031040100200401002004 0 041522222200551004650 5 047104236521011004410 5 SERIAL-12       CRI - MENDOTA HE
  1014 044507220521231302011 5 044516234270401002004 0 030066136310611363347 1 100000000000000000006 IGHTS, MINN.     06/21/79
  1020 020040100200401002004 0 310611643046016431062 020040100200401003005 6 030060140300401002004 0          21:10:22      0.0000
  1024 041523240200401002004 0 020040100200401002004 0 100000000000000000016 020040100200401002004 0 CSP
0141030 310611643046016431062 020040100200401003005 6 030060140300401002004 0 041523240200401002004 0 21:10:22      0.0000    CSP
0141034 020040100200401002004 0 041522202544551422004 0 047520212511012504451 6 043440246545232504251 5          CRAY-1  OPERATING SYSTEM
0141040 020040100200401002004 0 041517246200611343006 6 020040202515232124650 2 046131100421012504244 0          COS 1.06  ASSEMBLY DATE
0141044 030066136310601363347 1 100000000000000000006 020040100200401002004 0 310611643046016431062 06/20/79                  21:10:22
0141050 020040100200401003005 6 030060140300401002004 0 041523240200401002004 0 020040100200401002004 0       0.0000    CSP
0141054 100000000000000000006 020040100200401002004 0 310611643046016431062 020040100200401003005 6                21:10:22      0
0141060 030060140300401002004 0 041523240200401002004 0 020040100200401002004 0 100000000000000000007 0000    CSP
0141064 020040100200401002004 0 310611643046016431062 020040100200401003005 6 030060140300401002004 0          21:10:22      0.0000
0141070 041523240200401002004 0 045117204241122343650 2 052523262261241222004 0 100000000000000000006 CSP      JOB(JN=BUSY,T)
0141074 020040100200401002004 0 310611643046016431062 020040100200401003005 6 030060140310401002004 0          21:10:22      0.0002
0141100 041523240200401002004 0 041506250270401002004 0 100000000000000000013 020040100200401002004 0 CSP      CFT.
0141104 031061164304601643106 2 020040100200401003005 6 030060140314401002004 0 052523212510401002004 0 21:10:22      0.0003    USER
0141110 043124140300641002644 0 041506250201262125112 3 044517234200551002004 0 030066136310601363347 1 FT004 - CFT VERSION -    06/20/79
0141114 020123206441052105251 4 042522100200401002004 0 100000000000000000012 020040100200401002004 0  SCHEDULER
0141120 031061164304601643106 2 020040100200401003005 6 030060142324401002004 0 052523212510401002004 0 21:10:22      0.0015    USER
0141124 043124140300611002644 0 041517232501112304244 0 052111232424401722004 0 020040140270601403046 2 FT001 - COMPILE TIME =     0.0012
0141130 020123212415172344212 3 100000000000000000006 020040100200401002004 0 310611643046016431062  SECONDS                  21:10:22
0141134 020040100200401003005 6 030060142324401002004 0 041523240200401002004 0 046104244270401002004 0       0.0015    CSP      LDR.
0141140 100000000000000000010 020040100200401002004 0 310611643046016431062 020040100200401003005 6                21:10:22      0.
0141144 030060144324401002004 0 052523212510401002004 0 461041403046010026440 041105216445161004253 0 0025    USER     LD010 - BEGIN EX
0141150 042503252521112364704 0 100000000000000000000 000000000000000000000 000000000000000000000 ECUTION
          LOCATIONS 00141154 THROUGH 00141777 CONTAIN 000000000000000000000
```

```
0142000 0017740000000000007000 0000000000000000000000 0221042124112521600000 0000000000000000000000                $DEBUG
0142004 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142010 0000000000000000000000 0000000000000000000000 0411252465440000000000 0441172524700000000001                BUSY    HOUN
0142014 0300020040000000116000 0004160130000000145000 0212462231431100145000 0735670000000000036067 0              " I            <7
0142020 0000000000057133222455 0000000000000000000055 0040000012200000004000 0000000000000000000000       %-       -
0142024 0000000000000000000000 0000001411317731664600 0461041403046010026440 0411052164451610042530          0     LD010 - BEGIN EX
0142030 0425032525211123647040 0000000000000000000000 0000000000000000000000 0000000000000000000000 ECUTION
0142034 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142040 0000000000000000000000 0000000000002763711340 0000000000000000000000 0425322255167321422126                          EZJ  F$V
0142044 0425322255254201230216 0425322255253643523024 0425322255167434626471 0377676123163236734000 EZJ   0 EZJ ^   EZJ   -9?  3
0142050 0000000000000000000000 0000000000000000000000 0000000000000000007000 0511172304604021047124                          ROLL DNT
0142054 0700002464140000021420 0000000000000000000000 0000000000000000000145 0000000000000000000000   SC  #
              LOCATIONS 00142060 THROUGH 00142073 CONTAIN 0000000000000000000000
0142074 0004000000000000000000 0000000000000100000000 0000000640000000145000 0421041323047113231463                          DD-19-33
0142100 0000000001100000000006 0041050204301034204162 0041640207300000000000 0000000000000000000000          E F
              LOCATIONS 00142104 THROUGH 00142403 CONTAIN 0000000000000000000000
0142404 1000000707070700000016 0000000000000000004423 0221032460000000000001 0400002224700100020600                $CS     @ IN  !
0142410 0000010000000000120000 1001600000000000000000 0100000035222400001320 0000000000000000000000
0142414 0000000100000000100001 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142420 0000000000000000000000 1000000707070700000016 1000000707070700000016 0000000000440500004441                                !
0142424 0221142364340000000001 0200002405100177773321 0000000000000000000000 0001600000000000000063 $LOG      PR                  3
0142430 0100000000000000000000 0000000000000000000000 0000000000000000000001 0000000000000000000000
0142434 0000000000000000000063 0000000000000000000000 0000000000000000000000 1000000707070700000016       3
0142440 1000000707070700000016 0000000000442300004501 0221112340000000000000 0610002464140100020600                A$IN       SC  !
0142444 0000010000000000253000 1000200000000000000000 0000000000000000134640 0000000000000000000000       V
0142450 0000000100000000100004 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142454 0000000000000000000000 1000000707070700000016 1000000707070700000022 0004000000000100000000
0142460 0000000000000100000000 0000000560000000000000 0421041323047113231463 0000000001100000000003                DD-19-33
0142464 0037030174240761600000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142470 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142474 0000000000000000000000 0000000000000000000000 0000000000000000000000 1000000707070700000022
0142500 1000000707070700000016 0000000000444100004517 0221172525200000000000 0600002405100100000000              !  O$OUT    ' PR
0142504 0000000000000000000000 0001600000000000000000 0000000000000000000000 0000000000000000000000
0142510 0000000000000000000004 0000000000000000000000 0000000000000000003720 0000000000000000000000
0142514 0000000000000000000000 1000000707070700000016 1000000707070700000016 0000000000450100004535                              A  I
0142520 0221022304200000000000 0000002464140177773203 0000010000000000247000 0001600000000000000000 $BLD      SC          N
0142524 0000000000000000124660 0000000000000000000000 0000000200000000000004 0000000000000000000001
0142530 0000000000000000116100 0000000000000000000000 0000000000000000000000 1000000707070700000016       @
0142534 1000000707070700000016 0000000000451700000000 0221042124112521600000 0700012464140177773225          O   $DEBUG    SC
0142540 0000010000000000243000 0001600000000000000000 0000000000000000124700 0000000000000000000000       F
0142544 0000000100000000100004 0000000000000000000001 0000000000000000116100 0000000000000000000000                        @
0142550 0000000000000000000000 1000000707070700000016 1000000707070700000022 0004000000000100000000
```

9.8

```
0142554 0000000000000100000000 0000000520000000001000 042104132304711323146 3 0000000001100000000001                DD-19-33
0142560 0037100000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142564 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0142570 0000000000000000000000 0000000000000000000000 0000000000000000000000 1000000707070700000022
0142574 1000000707070700000022 0004000000000100000000 0000000000000100000000 0000000520000000001000
0142600 042104132304711323146 3 0000000001100000000001 0037110000000000000000 0000000000000000000000 DD-19-33
            LOCATIONS 00142604 THROUGH 00142613 CONTAIN 000000000000000000000000
0142614 0000000000000000000000 1000000707070700000022 0000000707070700002162 0000000000000000000000
            LOCATIONS 00142620 THROUGH 00144773 CONTAIN 000000000000000000000000
0144774 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000707070700002162
```

```
      **** DMEM,FWA=303000,L=7000.
```

9.9

```
0303000 0415222025151000000000 0000000000000000072535 0000000000000217262754 0000000000000006340325 CRASH          ]     =
0303004 0000000000000000000004 0000000000000000000003 0000000000000000000000 0000000000000000000000
              LOCATIONS 00303010 THROUGH 00303017 CONTAIN 0000000000000000000000
0303020 0000000000133300000000 0000000031200000000021 0000000045001600000001 0000001502100000000000                    P      4"
0303024 0000000000000000000000 0000000000000000000657 0000000000000000000020 0000000000000000000006
0303030 0000000000000000000017 0000000000000000000020 0000000000000000000000 0000000000000000000000
0303034 0200000000000000000000 0000000000000000000017 0000000000000000000040 0000000000000000000176
0303040 0002000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00303044 THROUGH 00303057 CONTAIN 0000000000000000000000
0303060 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000066657
0303064 0000000000000000000000 0000000000000000000000 0000000000000000324715 0000000000000000343636
0303070 0000000000000000000000 0000000000000000000001 0000000000000000000001 0421072104340000036102                    DGDG  <B
0303074 0475202125110125047522 0400375321405400000000 0400455051050044000000 0000040000000000000000 OPERATOR@   ,   @%  @
0303100 0000000000000000004561 0000000000000000000200 0000000000000000001074 0000000000000000000000                    <
              LOCATIONS 00303104 THROUGH 00303167 CONTAIN 0000000000000000000000
0303170 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000001616
0303174 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00303200 THROUGH 00303277 CONTAIN 0000000000000000000000
    300 0425162504252200000000 0124120644252200000000 0060122504252200000000 0000122504252200000000 ENTER      ER     TER     TER
    304 0135162504252200000000 0111162504252200000000 0041162504252200000000 0100372504252200000000  NTER    NTER    NTER     TER
    310 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00303314 THROUGH 00303377 CONTAIN 0000000000000000000000
0303400 0421160000000000000000 0501042340000000000000 0445040000000000000000 0425040000000000000000 DN      PDN     ID      ED
0303404 0510000000000000000000 0534000000000000000000 0464000000000000000000 0525210000000000000000 R       W       M       UQ
0303410 0425162504252200000000 0060140160602502400000 0074140420000101447000 0114060360002200042400 ENTER                N       E
0303414 0114060421003524600000 0114060500603505600000 0114050364252324600000 0110020520012324600000      S             ESS      SS
0303420 0110240241042006600000 0000000000000000000000 0000000000000000000000 0000000000000000000000
              LOCATIONS 00303424 THROUGH 00303477 CONTAIN 0000000000000000000000
0303500 1000000000000000000000 0000000000000000000000 0000000000000000000000 0300000000000000000000                    0
0303504 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0303510 0000000000000000000000 0300000000000000000000 0000000000000000000000 0000000000000000000000        0
              LOCATIONS 00303514 THROUGH 00303577 CONTAIN 0000000000000000000000
0303600 1000000000000000000000 1000000000000000000000 1000000000000000000000 1000000000000000000000
0303604 1000000000000000000000 1000000000000000000000 1000000000000000000000 0000000000000000000001
0303610 0000000000000000000001 1000000000000000000000 0000000000000000000001 0000000000000000000000
              LOCATIONS 00303614 THROUGH 00303677 CONTAIN 0000000000000000000000
0303700 0000000000000000000660 0000000000000000000660 0000000000000000000662 0000000000000000001230
0303704 0000000000000000000673 0000000000000000000674 0000000000000000000675 0000000000000000001231
0303710 0000000000000000001232 0000000000000000000651 0000000000000000001642 0000000000000000000000
              LOCATIONS 00303714 THROUGH 00303777 CONTAIN 0000000000000000000000
0304000 0000000000000000000657 0000000000000000000660 0000000000000000000662 0000000000000000001230
0304004 0000000000000000000673 0000000000000000000674 0000000000000000000675 0000000000000000001231
```

```
0304010 000000000000000001232 000000000000000000650 000000000000000001642 000000000000000000000
        LOCATIONS 00304014 THROUGH 00304077 CONTAIN 000000000000000000000
0304100 000000000000000000657 000000000000000000661 000000000000000000662 000000000000000001230
0304104 000000000000000000673 000000000000000000674 000000000000000000675 000000000000000001231
0304110 000000000000000001232 000000000000000000650 000000000000000001642 000000000000000000000
        LOCATIONS 00304114 THROUGH 00304177 CONTAIN 000000000000000000000
0304200 040523246445072340000 000000000000000000017 042116000000000000000 000000000000000000007 ASSIGN          DN
0304204 022111234000000000000 000000000000000000017 040400000000000000000 000000000000000000007 $IN .          A
0304210 043124140324000000000 000000000000000000017 046115000000000000000 000000000000000000007 FT05           LM
0304214 030000000000000000000 000000000000000000037 000000000000000000000 000000000000000000000 0
        LOCATIONS 00304220 THROUGH 00304317 CONTAIN 000000000000000000000
0304320 022103246000000000001 000003000000000002000 000000000000010002000 000000000000000002012 $CS
0304324 000000000000000003000 100000000000000002013 000000000000000000000 000020000000000000000
0304330 100000000000000002011 000000000000000000000 000000003365140026466O 000000000000000000000                L
0304334 000000000000000352545 000000002674060003610Z 002000000070050000701Z 100000000000000000000          <B
0304340 022114236434000000001 000001000000000003000 000000000000000003130 000000000000000003000 $LOG                X
0304344 000000000000000004000 104000000000000003127 000000000000000000000 000000200000000000000              W
0304350 104000000000000003120 000000000000000000000 000000003563660000071Z 000000000000000000000         P
0304354 000000000000000000000 000000002674240003610Z 000000000000000000000 000000000000000000000          <B
0304360 041517240545272000000 100000000000000000200 000000000000000000200 177777777777700000200 COPYR
0304364 000000000000000001200 000000000000000000000 000000000000000000000 000000000000000000000
0304370 000000000000000000000 000000000000000000000 000000003360200000720O 000000000000000000000
0304374 000000000000000352545 177777777777777521200 000200000072000000721O 000000000000000000000
        LOCATIONS 00304400 THROUGH 00304437 CONTAIN 000000000000000000000
0304440 000000000000000000000 000000000000000263000 000000000000000036102 000000000000000272000                <B
0304444 000000000000000004520 000000000000000335713 000000000000000000010 000000000000000000005         P
0304450 000000000000000000000 041517240545220000000 041517240545220000000 000000000000000000000           COPYR   COPYR
0304454 000000000000001400177700 177777777600000000000 000000000000000177700 000000000000001400177700
0304460 000000000000000000000 000000000000000000000 000000000000000000000 000000000000000000000
        LOCATIONS 00304464 THROUGH 00304657 CONTAIN 000000000000000000000
0304660 041517240545221340000 000000000000000000000 000000000000000000000 000000000000000000000 COPYR.
        LOCATIONS 00304664 THROUGH 00304673 CONTAIN 000000000000000000000
0304674 000000000064750013600O 000002001260000012560O 000004000000000012557O 000000000000000000000      =
        LOCATIONS 00304700 THROUGH 00304737 CONTAIN 000000000000000000000
0304740 000000000000000033425Z 000000000000000000000 000000000000000000000 000000000000000007643
0304744 000000000000000026744Z 000000000000000000000 000000000000000000000 000000000000000000000          "
        LOCATIONS 00304750 THROUGH 00304777 CONTAIN 000000000000000000000
0305000 000000000000000000002 045117204241122343650З 051101246440510000000O 103000000000000000003          JOB(JN=CRASH)
0305004 040503206425232462410Ч 047075206475202625105Ч 042516250425221220000O 102000000000000000001 ACCESS(DN=COPYR,ENTER)
0305010 041517240545221340000O 102000000000000000001 042530222520560000000O 103000000000000000001 COPYR.          EXIT.
0305014 042530222520560000000O 103000000000000000000 160000000000000000000 170000000000000000000 EXIT.
0305020 000000000000000000000 000000000000000000000 000000000000000000000 000000000000000000000
```

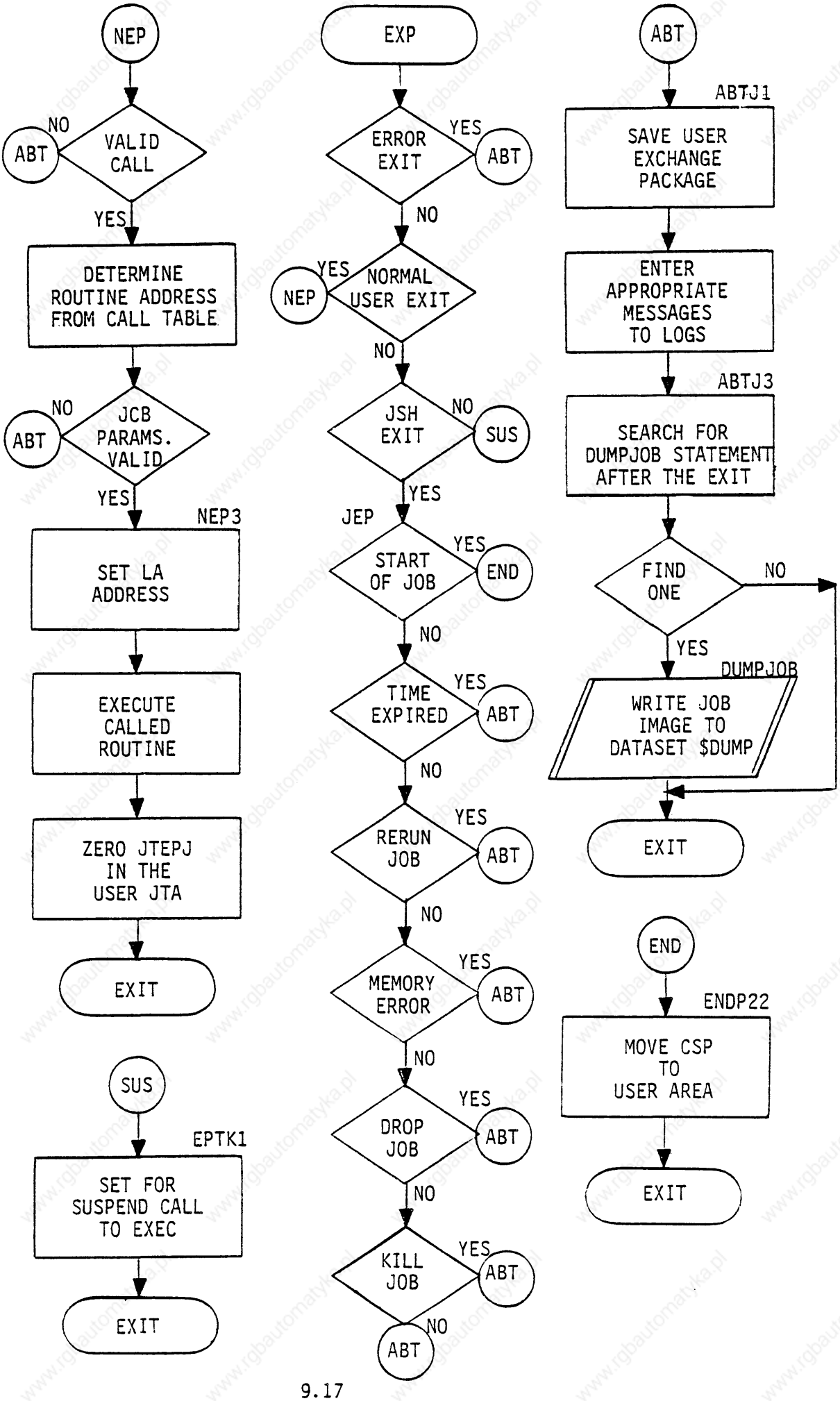
6.11

```
          LOCATIONS 00305024 THROUGH 00305777 CONTAIN 00000000000000000000
0306000 0000000000000000000016 0200401002004010020040 0310611643046216431062 0200401002004010030056                21:12:22      0.
0306004 0300601403004010020040 0415232402004010020040 0200401002004010020040 0415222025445514220040 0000    CSP             CRAY-1
0306010 0515052444450123026461 0310401002004010020040 0415222222005510046505 0471042365210110044105 SERIAL-12       CRI - MENDOTA HE
0306014 0445072205212313020115 0445162342704010020040 0300661363106113633471 1000000000000000000006 IGHTS, MINN.    06/21/79
0306020 0200401002004010020040 0310611643046216431062 0200401002004010030056 0300601403004010020040        21:12:22      0.0000
0306024 0415232402004010020040 0200401002004010020040 1000000000000000000016 0200401002004010020040 CSP
0306030 0310611643046216431062 0200401002004010030056 0300601403004010020040 0415232402004010020040 21:12:22      0.0000    CSP
0306034 0200401002004010020040 0415222025445514220040 0475202125110125044516 0434402465452325042515        CRAY-1  OPERATING SYSTEM
0306040 0200401002004010020040 0415172462006113430066 0200402025152321246502 0461311004210125042440        COS 1.06  ASSEMBLY DATE
0306044 0300661363106013633471 1000000000000000000006 0200401002004010020040 0310611643046216431062 06/20/79              21:12:22
0306050 0200401002004010030056 0300601403004010020040 0415232402004010020040 0200401002004010020040      0.0000    CSP
0306054 1000000000000000000006 0200401002004010020040 0310611643046216431062 0200401002004010030056                21:12:22      0.
0306060 0300601403004010020040 0415232402004010020040 0200401002004010020040 1000000000000000000007 0000    CSP
0306064 0200401002004010020040 0310611643046216431062 0200401002004010030056 0300601403004010020040        21:12:22      0.0000
0306070 0415232402004010020040 0451172042411223436503 0511012464405110020040 1000000000000000000010 CSP     JOB(JN=CRASH)
0306074 0200401002004010020040 0310611643046216431062 0200401002004010030056 0300601403104010020040        21:12:22      0.0002
0306100 0415232402004010020040 0405032064252324624104 0470752064752026251054 0425162504252212220040 CSP     ACCESS(DN=COPYR,ENTER)
0306104 1000000000000000000013 0200401002004010020040 0310611643046216431062 0200401002004010030056                21:12:22      0.
0306110 0300601403144010020040 0501042322004010020040 0501041403006110026440 0405032064252324620040 0003    PDM       PD001 - ACCESS
0306114 0415172405452210020040 0200401002004010020040 0425041723006014031440 0415172325011421252105 COPYR           ED=0003 COMPLETE.
0306120 1000000000000000000006 0200401002004010020040 0310611643046216431062 0200401002004010030056                21:12:22      0.
0306124 0300601403144010020040 0415232402004010020040 0415172405452213420040 1000000000000000000000 0003    CSP       COPYR.
0306130 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
          LOCATIONS 00306134 THROUGH 00306777 CONTAIN 00000000000000000000
0307000 0017770000000000007000 0000000000000000000000 0000000000000000000000 0000000000000000000000
          LOCATIONS 00307004 THROUGH 00307403 CONTAIN 00000000000000000000
0307404 1000000707070700000016 0000000000000000004423 0221032460000000000001 0400002224700200021400                $CS       @ IN  #
0307410 0000010000000000765000 1001600000000000000000 0100000035222400001320 0000000000000000000000
0307414 0000000100000000100001 0000000000000000000000 0000000000000000000000 0000000000000000000000
0307420 0000000000000000000000 1000000707070700000016 1000000707070700000016 0000000000440500004441                                !
0307424 0221142364340000000001 0200002405100277773321 0000000000000000000000 0001600000000000000063 $LOG      PR                      3
0307430 0100000000000000000000 0000000000000000000000 0000000000000000000001 0000000000000000000000
0307434 0000000000000000000063 0000000000000000000000 0000000000000000000000 1000000707070700000016        3
0307440 1000000707070700000016 0000000000442300004501 0221112340000000000000 0610002464140200021400                A$IN        SC  #
0307444 0000010000000000420000 1000200000000000000000 0000000000000000134640 0000000000000000000000
0307450 0000000100000000100004 0000000000000000000000 0000000000000000000000 0000000000000000000000
0307454 0000000000000000000000 1000000707070700000016 1000000707070700000022 0004000000000200000000
0307460 0000000000000100000000 0000000560000000000000 0421041323047113231463 0000000001160000000003                DD-19-33
0307464 0037200175040764400000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0307470 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000000000000000000
0307474 0000000000000000000000 0000000000000000000000 0000000000000000000000 1000000707070700000022
```

```
0307500 1000000707070700000016 0000000000444100004517 0221172525200000000000 0600002405100200000000              !  O$OUT     ' PR
0307504 0000000000000000000000 0001600000000000000000 0000000000000000000000 0000000000000000000000
0307510 0000000000000000000004 0000000000000000000000 0000000000000000003720 0000000000000000000000
0307514 0000000000000000000000 1000000707070700000016 1000000707070700000016 0000000000450100000000                    A
0307520 0415172405452200000000 1000002464140177773243 0000020000000000272200 1000200000000000000000 COPYR      SC
0307524 0040000000000000001360 0000000000000000000000 0000000100000000200004 0000000000000000000004
0307530 0000000000040000116100 0000000000000000000000 0000000000000000000000 1000000707070700000016         @
0307534 1000000707070700000022 0004000000000200000000 0000007424000100035405 0000000520000000004000                 @  ;
0307540 0421041323047113231463 0000000001100000000001 0020130000000000000000 0000000000000000000000 DD-19-33
            LOCATIONS 00307544 THROUGH 00307553 CONTAIN 000000000000000000000000
0307554 0000000000000000000000 1000000707070700000022 0000000707070700002222 0000000000000000000000
            LOCATIONS 00307560 THROUGH 00311773 CONTAIN 000000000000000000000000
0311774 0000000000000000000000 0000000000000000000000 0000000000000000000000 0000000707070700002222
```

USER ERROR EXIT

- A JOB EXECUTES AN ERROR EXIT (ERR) INSTRUCTION OR ENCOUNTERS A HARDWARE DETECTED ERROR
- EXP ISSUES APPROPRIATE ERROR MESSAGES
- EXP MAY INITIATE JOB ABORT OR REPRIEVE PROCESSING
- EXP MAY SEARCH THE REMAINING JOB CONTROL STATEMENTS FOR AN EXIT OR END-OF-FILE
- IF EXIT IS FOUND EXP EXAMINES THE NEXT STATEMENT
- IF THE NEXT STATEMENT IS 'DUMPJOB' A DATASET NAMED $DUMP IS WRITTEN TO DISK IN BINARY
- $DUMP IS A BINARY UNBLOCKED DATASET AND CONTAINS THE JTA AND ENTIRE USER FIELD (BA-LA) OF THE ABORTED JOB
- $DUMP MAY THEN BE USED BY THE 'DUMP' CONTROL STATEMENT TO OUTPUT VARIOUS AREAS OF THE ABORTED PROGRAM
- IF REPRIEVE PROCESSING, EXP TRANSFERS CONTROL BACK TO THE USER JOB AT THE ADDRESS SPECIFIED IN THE REPRIEVE MACRO


NEP
VALID CALL
NO
ABT
YES
DETERMINE ROUTINE ADDRESS FROM CALL TABLE
JCB PARAMS. VALID
NO
ABT
YES
NEP3
SET LA ADDRESS
EXECUTE CALLED ROUTINE
ZERO JTEPJ IN THE USER JTA
EXIT
SUS
EPTK1
SET FOR SUSPEND CALL TO EXEC
EXIT
EXP
ERROR EXIT
YES
ABT
NO
NORMAL USER EXIT
YES
NEP
NO
JSH EXIT
NO
SUS
YES
JEP
START OF JOB
YES
END
NO
TIME EXPIRED
YES
ABT
NO
RERUN JOB
YES
ABT
NO
MEMORY ERROR
YES
ABT
NO
DROP JOB
YES
ABT
NO
KILL JOB
YES
ABT
NO
ABT
ABT
ABTJ1
SAVE USER EXCHANGE PACKAGE
ENTER APPROPRIATE MESSAGES TO LOGS
ABTJ3
SEARCH FOR DUMPJOB STATEMENT AFTER THE EXIT
FIND ONE
NO
YES
DUMPJOB
WRITE JOB IMAGE TO DATASET $DUMP
EXIT
END
ENDP22
MOVE CSP TO USER AREA
EXIT

MESSAGE PROCESSOR (MSG)

MESSAGE PROCESSOR (MSG)

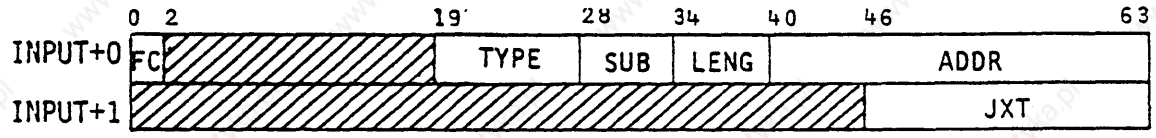
- WRITES MESSAGES IN THE SYSTEM AND USER LOG FILES.
- CAN BE CALLED BY OTHER TASKS AND USERS(THROUGH EXP).
- MSG USES TIO WHEN PERFORMING I/O.
- THE SYSTEM LOG CAN BE ANALYZED BY THE 'EXTRACT' PROGRAM.
- USERS CAN ENTER MESSAGES IN THE SYSTEM AND/OR USER LOG.

SYSTEM LOG PROCESSING

- THE SYSTEM LOG IS A PERMANENT DATASET NAMED $SYSTEMLOG
- THE SYSTEM LOG BUFFER AND DSP ARE ALLOCATED IN HIGH MEMORY.
- A TASK CALLS MSG THROUGH THE TSKREQ ROUTINE WITH THE FOLLOWING REGISTERS SET:

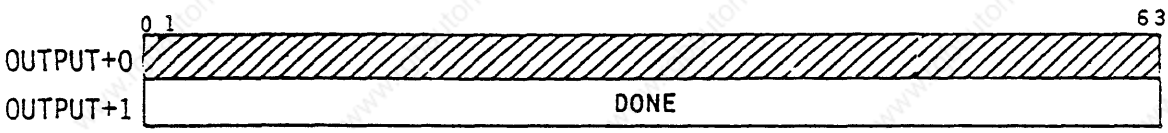
(A1) ID of task requesting message entry
(A2) MSGID,0
(S1) INPUT+0 } request
(S2) INPUT+1 }

Request format:



Field	Word	Bits	Description
FC	INPUT+0	0-1	Function code: 1 Write message in user's logfile only 2 Write message in SYSTEMLOG only 3 Write message in both user's logfile and in SYSTEMLOG
TYPE	INPUT+0	19-27	Major class of log record
SUB	INPUT+0	28-33	Subtype of log record
LENGTH	INPUT+0	34-39	Length in words of message. If length is 0, the message is a character string terminated by a zero byte in the low-order eight bits.
ADDR	INPUT+0	40-63	Starting address relative to STP of message to be written
JXT	INPUT+1	46-63	JXT address if message associated with job; otherwise 0

Reply format:



Field	Word	Bits	Description
DONE	OUTPUT+1	0-63	Characters 'MSG DONE' in ASCII

USER LOG PROCESSING

- THE USER LOG ($LOG) IS CREATED BY JSH FOR EACH USER JOB.
- THE USER LOG BUFFER IS IN THE JOB's JTA.
- THE USER CALLS MSG VIA THE 'MESSAGE' MACRO:

MESSAGE - ENTER MESSAGE IN LOGFILE

The printable ASCII message at the location specified in the macro call is entered in the job and system logfile. The message must be 1-80 characters terminated by a zero byte. A flag, *loc*, indicates the destination for the message.

Format:

Location	Result	Operand
	MESSAGE	*address,loc*

address A symbol or an A, S, or T register that contains the address.

loc Destination for message. May be any of the following:

- U User logfile only
- S System logfile only
- US User and system logfiles; default, if *loc* is blank.

MEMORY ERROR PROCESSOR (MEP)

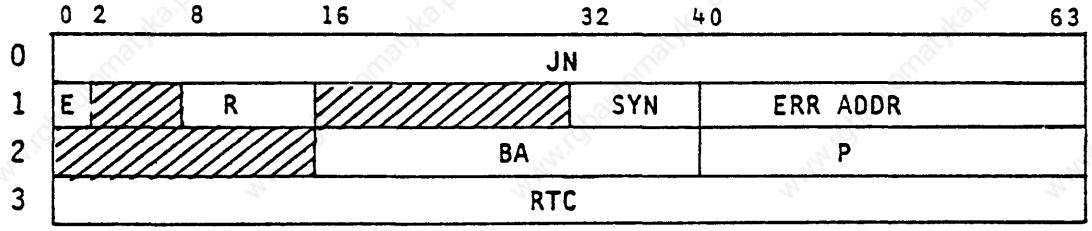
DISK ERROR CORRECTION (DEC)

www.rgbautomatyka.pl

MEMORY ERROR PROCESSOR (MEP)

- PURPOSE IS TO RELAY MEMORY ERROR INFORMATION FROM EXEC TO MSG.

- MESSAGES FROM EXEC TO MSG ARE IN THE FORMAT:



Field	Word	Bits	Significance
JN	0	0-63	Jobname or 'STP'
E	1	0,1	Error type (binary): 10 Uncorrectable memory error 01 Correctable memory error
R	1	8-15	Read mode: 0 Scalar 1 I/O 2 Vector 3 Fetch
SYN	1	32-39	Syndrome bits
ERR ADDR	1	40-63	Error address
BA	2	26-39	Base address
P	2	40-63	Program address
RTC	3	0-63	Real-time clock

DISK ERROR CORRECTION (DEC)

- ATTEMPTS CORRECTION OF A DATA ERROR FROM DISK.
- USES CYCLIC REDUNDANCY CHECKWORDS (CRC) FOR ERROR CORRECTION.
- DQM SCHEDULES DEC IF NORMAL RECOVERY PROCEDURES ARE UNSUCCESSFUL.
- DQM PASSES THE EQT ADDRESS WHICH CONTAINS THE VALID CHECKWORDS.
- DEC PROCESSES ONE DISK ERROR AT A TIME.
- REFER TO MASS STORAGE SUBSYSTEM REFERENCE MANUAL 2240630 FOR THE CRC ALGORITHM.

SYSTEM PERFORMANCE MONITOR

SYSTEM PERFORMANCE MONITOR

- MEASURES OVERALL SYSTEM PERFORMANCE
- INVOKED ON A TIME DELAY BASIS
- DELAY INTERVAL BETWEEN COLLECTIONS IS (I@SPMDLY)
- USES MESSAGE TASK TO WRITE SYSTEM TIMES TO THE SYSTEMLOG
- USES POOL 1 AS A BUFFER FOR THE SYSTEM TIMES
- MEASUREMENTS COLLECTED BY SPM INCLUDE

  CPU UTILIZATION
  NUMBER OF TASK READIES
  NUMBER OF EXEC REQUESTS BY TASK
  NUMBER OF EXEC REQUESTS BY TYPE
  MEMORY UTILIZATION
  CHANNEL INTERRUPT COUNT
  DISK CHANNEL UTILIZATION
  DISK UNIT UTILIZATION
  LINK UTILIZATION
  USER CALL USAGE
  JOB SCHEDULER STATISTICS
  JOB CLASS STATISTICS

```
11:12:02.9094       EXECUTIVE REQUESTS                TIME INTERVAL =    60012.18 MILLISECONDS
                          NUMBER OF TASK  0 REQUESTS =          0
                          NUMBER OF TASK  1 REQUESTS =        313
                          NUMBER OF TASK  2 REQUESTS =       1370
                          NUMBER OF TASK  3 REQUESTS =        266
                          NUMBER OF TASK  4 REQUESTS =          0
                          NUMBER OF TASK  5 REQUESTS =        715
                          NUMBER OF TASK  6 REQUESTS =        576
                          NUMBER OF TASK  7 REQUESTS =          0
                          NUMBER OF TASK  8 REQUESTS =         21
                          NUMBER OF TASK  9 REQUESTS =       2238
                          NUMBER OF TASK 10 REQUESTS =          0


11:12:02.9096       TASK UTILIZATION                  TIME INTERVAL =    60012.18 MILLISECONDS
                          NUMBER OF TASK  0 READIES =          0
                          NUMBER OF TASK  1 READIES =        129
                          NUMBER OF TASK  2 READIES =        488
                          NUMBER OF TASK  3 READIES =         86
                          NUMBER OF TASK  4 READIES =          0
                          NUMBER OF TASK  5 READIES =        297
                          NUMBER OF TASK  6 READIES =        156
                          NUMBER OF TASK  7 READIES =          0
                          NUMBER OF TASK  8 READIES =          3
                          NUMBER OF TASK  9 READIES =        410
                          NUMBER OF TASK 10 READIES =          0


11:12:02.9099       CPU UTILIZATION                   TIME INTERVAL =    60012.18 MILLISECONDS
                          USER TIME =        60134.40 MSEC    100.20%
                          IDLE TIME =            0.00 MSEC      0.00%
                          BLOCKED TIME =         0.00 MSEC      0.00%
                          EXEC TIME =          151.90 MSEC      0.25%
                          TASK  0 TIME =         0.00 MSEC      0.00%
                          TASK  1 TIME =        16.42 MSEC      0.03%
                          TASK  2 TIME =        25.84 MSEC      0.04%
                          TASK  3 TIME =         4.39 MSEC      0.01%
                          TASK  4 TIME =         0.00 MSEC      0.00%
                          TASK  5 TIME =        14.85 MSEC      0.02%
                          TASK  6 TIME =        17.61 MSEC      0.03%
                          TASK  7 TIME =         0.00 MSEC      0.00%
                          TASK  8 TIME =         0.41 MSEC      0.00%
                          TASK  9 TIME =        48.38 MSEC      0.08%
                          TASK 10 TIME =         0.00 MSEC      0.00%
                          TIME NOT ACCOUNTED FOR =    -402.02 MSEC     -0.67%

- - - - - - END OF EXTRACT REPORT          43892 RECORDS READ FROM $SYSLOG          49 RECORDS WRITTEN ON $OUT
```

Listing of all subtypes of SPM records

```
11:12:02.9087      CHANNEL INTERRUPT COUNTS        TIME INTERVAL =    60012.18 MILLISECONDS
                     CHANNEL  0 =       0            CHANNEL 17 =       0
                     CHANNEL  1 =       0            CHANNEL 18 =       0
                     CHANNEL  2 =     118            CHANNEL 19 =       0
                     CHANNEL  3 =     116            CHANNEL 20 =       0
                     CHANNEL  4 =     611            CHANNEL 21 =       0
                     CHANNEL  5 =     694            CHANNEL 22 =       0
                     CHANNEL  6 =    1819            CHANNEL 23 =       0
                     CHANNEL  7 =    1090            CHANNEL 24 =       0
                     CHANNEL  8 =       0            CHANNEL 25 =       0
                     CHANNEL  9 =       0            CHANNEL 26 =    5627
                     CHANNEL 10 =       0            CHANNEL 27 =       0
                     CHANNEL 11 =       0            CHANNEL 28 =       0
                     CHANNEL 12 =       0            CHANNEL 29 =       0
                     CHANNEL 13 =       0            CHANNEL 30 =     496
                     CHANNEL 14 =       0            CHANNEL 31 =       0
                     CHANNEL 15 =       0            CHANNEL 32 =       0
                     CHANNEL 16 =       0            CHANNEL 33 =       0

11:12:02.9090      EXEC CALL USAGE                 TIME INTERVAL =    60012.18 MILLISECONDS
                     NUMBER OF TYPE  0 CALLS =       0          NUMBER OF TYPE 17 CALLS =       0
                     NUMBER OF TYPE  1 CALLS =       0          NUMBER OF TYPE 18 CALLS =       0
                     NUMBER OF TYPE  2 CALLS =     905          NUMBER OF TYPE 19 CALLS =       0
                     NUMBER OF TYPE  3 CALLS =    1172          NUMBER OF TYPE 20 CALLS =       0
                     NUMBER OF TYPE  4 CALLS =       0          NUMBER OF TYPE 21 CALLS =       0
                     NUMBER OF TYPE  5 CALLS =     104          NUMBER OF TYPE 22 CALLS =       0
                     NUMBER OF TYPE  6 CALLS =     370          NUMBER OF TYPE 23 CALLS =       0
                     NUMBER OF TYPE  7 CALLS =       0          NUMBER OF TYPE 24 CALLS =       0
                     NUMBER OF TYPE  8 CALLS =      24          NUMBER OF TYPE 25 CALLS =       0
                     NUMBER OF TYPE  9 CALLS =     176          NUMBER OF TYPE 26 CALLS =       0
                     NUMBER OF TYPE 10 CALLS =      24          NUMBER OF TYPE 27 CALLS =       0
                     NUMBER OF TYPE 11 CALLS =       0          NUMBER OF TYPE 28 CALLS =       1
                     NUMBER OF TYPE 12 CALLS =       0          NUMBER OF TYPE 29 CALLS =       1
                     NUMBER OF TYPE 13 CALLS =     108          NUMBER OF TYPE 30 CALLS =       1
                     NUMBER OF TYPE 14 CALLS =     307          NUMBER OF TYPE 31 CALLS =       1
                     NUMBER OF TYPE 15 CALLS =     307          NUMBER OF TYPE 32 CALLS =       1
                     NUMBER OF TYPE 16 CALLS =    1997

11:12:02.9092      DISK CHANNEL UTILIZATION        TIME INTERVAL =    60012.18 MILLISECONDS
                              CHANNEL  0 =       0.00 MSEC     0.00%
                              CHANNEL  1 =       0.00 MSEC     0.00%
                              CHANNEL  2 =    2559.79 MSEC     4.27%
                              CHANNEL  3 =    1426.73 MSEC     2.38%
                              CHANNEL  4 =       0.00 MSEC     0.00%
                              CHANNEL  5 =       0.00 MSEC     0.00%
                              CHANNEL  6 =       0.00 MSEC     0.00%
                              CHANNEL  7 =       0.00 MSEC     0.00%
                              CHANNEL  8 =       0.00 MSEC     0.00%
                              CHANNEL  9 =       0.00 MSEC     0.00%
                              CHANNEL 10 =       0.00 MSEC     0.00%
                              CHANNEL 11 =       0.00 MSEC     0.00%
```

Listing of all subtypes of SPM records (continued)

08:54:24.8568 MEMORY UTILIZATION TIME INTERVAL = 300006.12 MILLISECONDS

EXECUTE MEMORY INTEGRAL	= 0.10355550E+02 MWS	8.944 % UTILIZATION
I/O WAIT MEMORY INTEGRAL	= 0.13730810E+02 MWS	11.860 % UTILIZATION
CPU WAIT MEMORY INTEGRAL	= 0.28333448E+02 MWS	24.473 % UTILIZATION
AVAILABLE MEMORY INTEGRAL	= 0.11577658E+03 MWS	

08:54:24.8570 DISK UTILIZATION TIME INTERVAL = 300006.12 MILLISECONDS

LDU	RATE	%SEEK	%TRANS	% TOTAL	# REQS	# ON-CYL	% FREE	% PERM	BLKS/REQ
DD-19-20	0.103857 MWDS/S	38.34	29.50	67.84	5348	773	6.76	0.00	11.38
DD-19-30	0.064985 MWDS/S	9.09	14.75	23.85	3476	290	12.26	0.00	10.95
DD-19-40	0.104673 MWDS/S	11.78	27.04	38.82	3203	376	23.22	0.00	19.12
DD-19-50	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-29-21	0.032098 MWDS/S	9.72	7.91	17.63	1539	266	56.03	0.00	12.22
DD-19-31	0.040775 MWDS/S	9.40	11.62	21.03	2426	289	32.09	0.00	9.85
DD-19-41	0.007004 MWDS/S	8.39	4.73	13.12	2188	713	20.15	0.00	1.88
DD-19-51	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-22	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-32	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-42	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-52	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-23	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-33	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-43	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
DD-19-53	0.000000 MWDS/S	0.00	0.00	0.00	0	0	100.00	0.00	0.00
	0.253392 MWDS/S						71.61	0.00	

08:54:24.8577 LINK UTILIZATION TIME INTERVAL = 300006.12 MILLISECONDS

LINK IDNT	TOTAL # MESSAGES	GROSS RATE (MBITS/S)	→OUTPUT DATA NET RATE← WORDS SENT	(MBITS/S)	←INPUT DATA NET RATE→ WORDS RCVD	(MBITS/S)
MS	152	0.001105	3299	0.000704	56	0.000012
VS	1372	0.014410	34197	0.007295	16885	0.003602
IC	2318	0.008560	4903	0.001046	206	0.000044
EB	0	0.000000	0	0.000000	0	0.000000
DG	796	0.003017	4486	0.000957	104	0.000022
JB	63	0.000220	264	0.000056	10	0.000002

Listing of all subtypes of SPM records (continued)

```
EXTRACT VERSION 1.10     COMPILED 05/30/81  RUN 06/04/81  11:06:21   ON SYSTEMLOG 06/04/81

08:54:24.8591       USER CALL USAGE                TIME INTERVAL =   300006.12 MILLISECONDS
                      NUMBER OF TYPE  0 CALLS =        0          NUMBER OF TYPE 25 CALLS =        0
                      NUMBER OF TYPE  1 CALLS =        1          NUMBER OF TYPE 26 CALLS =        0
                      NUMBER OF TYPE  2 CALLS =       92          NUMBER OF TYPE 27 CALLS =        8
                      NUMBER OF TYPE  3 CALLS =       83          NUMBER OF TYPE 28 CALLS =    13832
                      NUMBER OF TYPE  4 CALLS =      691          NUMBER OF TYPE 29 CALLS =        0
                      NUMBER OF TYPE  5 CALLS =     2720          NUMBER OF TYPE 30 CALLS =        0
                      NUMBER OF TYPE  6 CALLS =        7          NUMBER OF TYPE 31 CALLS =        0
                      NUMBER OF TYPE  7 CALLS =        0          NUMBER OF TYPE 32 CALLS =        0
                      NUMBER OF TYPE  8 CALLS =      656          NUMBER OF TYPE 33 CALLS =      778
                      NUMBER OF TYPE  9 CALLS =       26          NUMBER OF TYPE 34 CALLS =       21
                      NUMBER OF TYPE 10 CALLS =      178          NUMBER OF TYPE 35 CALLS =        0
                      NUMBER OF TYPE 11 CALLS =      593          NUMBER OF TYPE 36 CALLS =        0
                      NUMBER OF TYPE 12 CALLS =      480          NUMBER OF TYPE 37 CALLS =        1
                      NUMBER OF TYPE 13 CALLS =        0          NUMBER OF TYPE 38 CALLS =      204
                      NUMBER OF TYPE 14 CALLS =     2098          NUMBER OF TYPE 39 CALLS =        8
                      NUMBER OF TYPE 15 CALLS =        0          NUMBER OF TYPE 40 CALLS =        0
                      NUMBER OF TYPE 16 CALLS =       38          NUMBER OF TYPE 41 CALLS =        0
                      NUMBER OF TYPE 17 CALLS =       46          NUMBER OF TYPE 42 CALLS =       88
                      NUMBER OF TYPE 18 CALLS =     2338          NUMBER OF TYPE 43 CALLS =        0
                      NUMBER OF TYPE 19 CALLS =     3105          NUMBER OF TYPE 44 CALLS =        0
                      NUMBER OF TYPE 20 CALLS =      825          NUMBER OF TYPE 45 CALLS =      312
                      NUMBER OF TYPE 21 CALLS =        3          NUMBER OF TYPE 46 CALLS =      203
                      NUMBER OF TYPE 22 CALLS =        2          NUMBER OF TYPE 47 CALLS =        0
                      NUMBER OF TYPE 23 CALLS =   118773          NUMBER OF TYPE 48 CALLS =        0
                      NUMBER OF TYPE 24 CALLS =       11

08:49:24.8523       JOB CLASS STATISTICS           TIME INTERVAL =   300006.14 MILLISECONDS
                            CLASS NAME =                        OPERATR
                            NUMBER OF ACTIVE JXTS =                   0
                            NUMBER OF JOBS WAITING FOR JXTS =         0
                            NUMBER OF RESERVED JXTS =                 1
                            MAXIMUM NUMBER OF JXTS =                 94
                            STATUS =                                 ON

08:54:24.8586       JOB SCHEDULER STATISTICS       TIME INTERVAL =   300006.12 MILLISECONDS
                            NUMBER OF MEMORY COMPACTS =           5756
                            NUMBER OF ROLLS =                     1288
                            NUMBER OF EXPANDS =                     14
                            NUMBER OF REDUCES =                      2
                            NUMBER OF INITIATES =                    8
                            NUMBER OF TERMINATES =                   6
                            NUMBER OF SCHEDULING INTERVALS =        80
                            NUMBER OF INDEX WRITES =               212
                            CLASS STRUCTURE NAME =                 DAY
                            NUMBER OF JOBS IN SYSTEM =               38
                            NUMBER OF ACTIVE JXTS =                  22
                            MAXIMUM # JXTS  (LIMIT)                  30
                            NUMBER OF AVAILABLE POOL JXTS =           3
                            NUMBER OF DEFINED CLASSES =              14
                            NUMBER OF CLASSES WAITING FOR JXTS =  65535
```

Listing of all subtypes of SPM records (continued)

OVERLAY MANAGER

OVERLAY MANAGER (OVM)

- CONTROLS THE LOADING OF TASK OVERLAYS.

- EACH TASK USING OVM MUST RESERVE A MEMORY AREA SUFFICIENT TO HOLD ONE OR MORE DISK BLOCKS (512 WORDS) AS AN OVERLAY AREA.

- OVERLAYS MAY BE NESTED UP TO TEN DEEP, THAT IS OVERLAY X MAY CALL OVERLAY Y, ETC.

- AN OVERLAY MAY CALL ANOTHER OVERLAY WITH OR WITHOUT RETURN REQUESTED TO ITSELF.

- ONCE IN MEMORY AN OVERLAY MAY OR MAY NOT BE RELOADED FROM DISK DEPENDENT ON CHANGES IN INTERNAL CODING OR TABLES WITHIN THE OVERLAY.

- EACH CALLED OVERLAY OVERWRITES THE TASKS OVERLAY AREA WITH THE REQUESTED OVERLAY.

JOB CLASS MANAGER

JOB CLASS MANAGER (JCM)

- VALIDATES JOB CLASS QUALIFICATIONS AND ASSIGNS A JOB TO A CLASS.

- LINKS JOB INPUT QUEUE ENTRIES IN THE SDT BY CLASS RANK, JOB PRIORITY, AND TIME SUBMITTED.

- OPTIONALLY REPLACES EACH JOB STATEMENT PRIORITY WITH THE CLASS RANK PRIORITY.

- ASSIGNS THE NUMBER OF AVAILABLE JOB EXECUTION TABLE (JXT) ENTRIES FOR EACH SPECIFIED JOB CLASS.

- ONCE A JOB HAS A JOB EXECUTION TABLE (JXT) ENTRY, ITS CLASS ASSIGNMENT DOES NOT CHANGE UNLESS THE JOB IS RERUN.

- A USER MAY LOWER A JOB CLASS THROUGH A PARAMETER IN THE JOB CONTROL STATEMENT.

- ENABLED/DISABLED THROUGH OPERATOR COMMANDS.

TAPE QUEUE MANAGER

TASK TO BE PROVIDED

CONTROL STATEMENT PROCESSOR (CSP)

CONTROL STATEMENT PROCESSOR (CSP)

- BINARY COPY OF CSP RESIDES IN MEMORY IMMEDIATELY ABOVE STP OR OPTIONALLY ON DISK(S).

- WHEN A USER JOB IS SUBMITTED JSH CALLS EXP TO COPY CSP INTO THE USER FIELD (AT BA+200_8).

- CSP EXECUTES IN THE USER FIELD AND IS SUBJECT TO USER JOB STATES.

- CSP SHARES THE USER TABLES SUCH AS THE JTA, JCB, DSP's, ETC.

- CSP 'CRACKS' USER CONTROL STATEMENTS.

- CSP ENTERS MESSAGES INTO THE LOGS THROUGH THE MESSAGE MACRO.

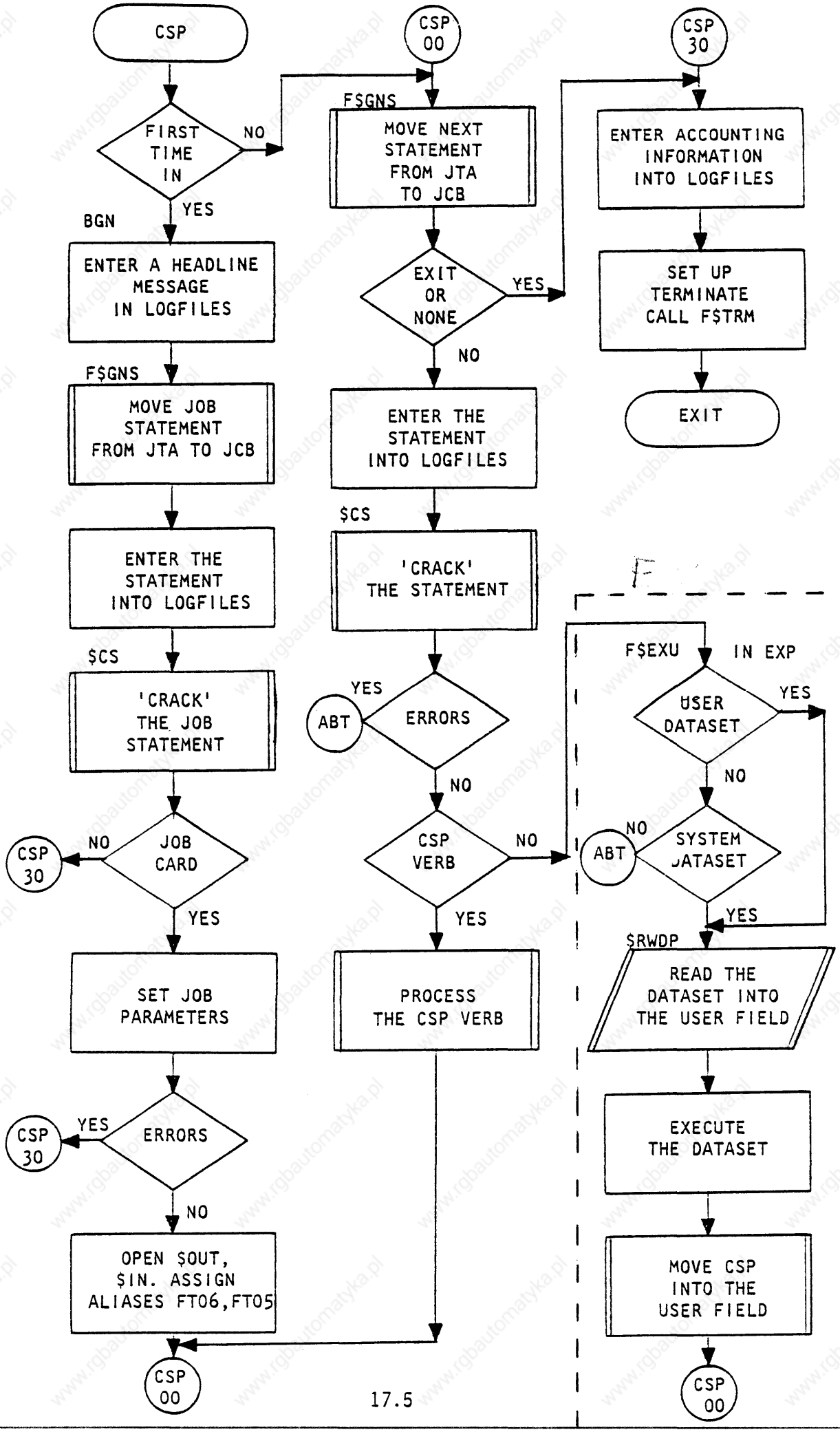
EXAMPLE

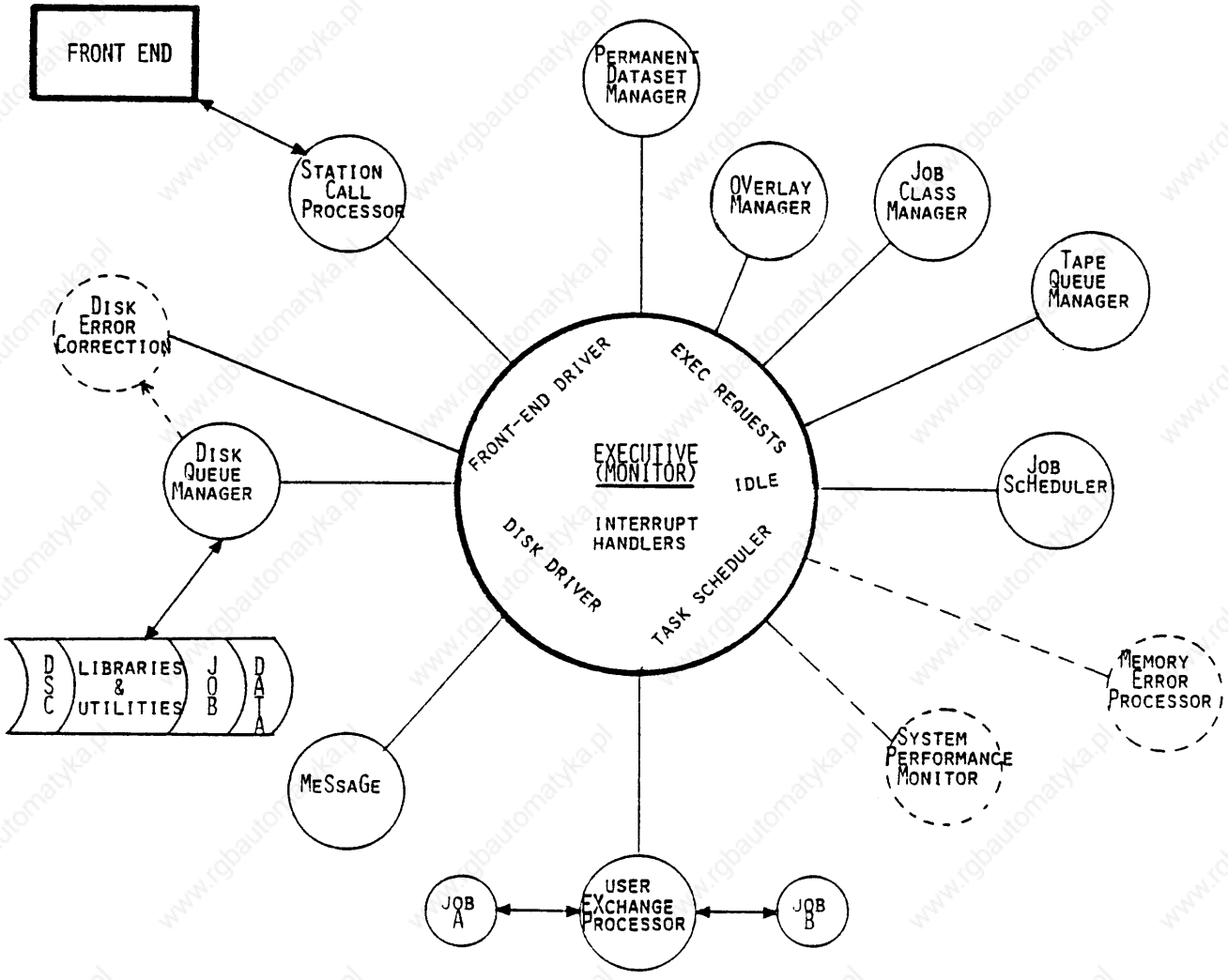
THE CONTROL STATEMENT

```
ASSIGN(DN=MYFILE,BS=25)
```

IS STORED IN THE JCB AS FOLLOWS:

0								
5	A	S	S	I	G	N	(	D
	N	=	M	Y	F	I	L	E
	,	B	S	=	2	5	)	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
				.				
				.				
				.				
15	0	0	0	0	0	0	0	0
16	A	S	S	I	G	N	0	0
	0	0	0	0	0	0	0	17_8
	D	N	0	0	0	0	0	0
	0	0	0	0	0	0	0	07
	M	Y	F	I	L	E	0	0
	0	0	0	0	0	0	0	17_8
	B	S	0	0	0	0	0	0
	0	0	0	0	0	0	0	07
	2	5	0	0	0	0	0	0
	0	0	0	0	0	0	0	37_8


CSP
FIRST TIME IN
NO
YES
BGN
ENTER A HEADLINE MESSAGE IN LOGFILES
F$GNS
MOVE JOB STATEMENT FROM JTA TO JCB
ENTER THE STATEMENT INTO LOGFILES
$CS
'CRACK' THE JOB STATEMENT
JOB CARD
NO
CSP 30
YES
SET JOB PARAMETERS
ERRORS
YES
CSP 30
NO
OPEN $OUT, $IN. ASSIGN ALIASES FT06,FT05
CSP 00
CSP 00
F$GNS
MOVE NEXT STATEMENT FROM JTA TO JCB
EXIT OR NONE
YES
NO
ENTER THE STATEMENT INTO LOGFILES
$CS
'CRACK' THE STATEMENT
ERRORS
YES
ABT
NO
CSP VERB
NO
YES
PROCESS THE CSP VERB
CSP 30
ENTER ACCOUNTING INFORMATION INTO LOGFILES
SET UP TERMINATE CALL F$TRM
EXIT
F$EXU
IN EXP
USER DATASET
YES
NO
SYSTEM DATASET
NO
ABT
YES
$RWDP
READ THE DATASET INTO THE USER FIELD
EXECUTE THE DATASET
MOVE CSP INTO THE USER FIELD
CSP 00


FRONT END
STATION CALL PROCESSOR
PERMANENT DATASET MANAGER
OVERLAY MANAGER
JOB CLASS MANAGER
TAPE QUEUE MANAGER
DISK ERROR CORRECTION
DISK QUEUE MANAGER
JOB SCHEDULER
FRONT-END DRIVER
EXEC REQUESTS
EXECUTIVE (MONITOR)
IDLE
INTERRUPT HANDLERS
DISK DRIVER
TASK SCHEDULER
DSC
LIBRARIES & UTILITIES
JOB DATA
MESSAGE
MEMORY ERROR PROCESSOR
SYSTEM PERFORMANCE MONITOR
JOB A
USER EXCHANGE PROCESSOR
JOB B

```
JOB,JN=EXAMPLE,P=3.
ACCESS,DN=TEMP,PDN=MASTERFILE,R=SECRET.
ASSIGN,DN=TEMP,BS=12.
COPYD,I=TEMP,O=COPY.
DISPOSE,DN=COPY,SDN=BACKUP,DC=ST,MF=A.
EXIT.
EOF.
```

1. JOB IS STAGED FROM FRONT-END TO CRAY-1.

2. STATION CALL PROCESSOR (SCP) RECEIVES JOB AND PLACES IT IN THE SYSTEM DATASET TABLE (SDT) WHICH IS THE INPUT QUEUE FOR JOB SCHEDULER. SCP ALSO SAVES JOB IN THE DATASET CATALOG (DSC).

3. JOB SCHEDULER PICKS THE JOB FOR EXECUTION FROM SDT AND THEN MAKES AN ENTRY FOR THE JOB IN THE JOB EXECUTION TABLE AND ALLOCATES MEMORY.

4. JOB SCHEDULER CONNECTS THE JOB TO CPU.

5. CONTROL STATEMENT PROCESSOR (CSP) CRACKS THE ACCESS CARD, CALLS THE EXCHANGE PROCESSOR (EXP) TO VERIFY PERMISSIONS TO ACCESS MASTERFILE.

6. CSP CRACKS THE ASSIGN CARD, CALLS EXP TO OPEN TEMP, AND CHANGE THE BUFFER SIZE TO 12 INSTEAD OF 4 BLOCKS.

7. CSP CRACKS THE COPYD CARD, CALLS EXP TO LOAD PROGRAM COPYD FROM THE SYSTEM DIRECTORY.

8. COPYD EXECUTES, I/O BOUND, STREAMING DATA FROM TEMP TO COPY.

9. JOB SCHEDULER RECOGNIZES THAT JOB'EXAMPLE' IS STREAMING AND CONNECTS EXAMPLE TO THE CPU WHENEVER A BLOCK IS AVAILABLE IN THE INPUT CIRCULAR BUFFER.

10. CSP CRACKS THE DISPOSE STATEMENT, CALLS EXP.

11. EXP WRITES AN ENTRY FOR BACKUP IN THE SDT OUTPUT QUEUE.

12. SCP STAGES BACKUP TO MAINFRAME A.

13. DEPENDING ON MAINFRAME A, SCP DELETES ENTRIES FOR BACKUP IN THE SDT AND DSC.

14. CSP CRACKS EXIT STATEMENT, CALLS EXP TO TERMINATE THE JOB.

15. EXP SEARCHES FOR $OUT, COPIES $LOG TO $OUT (EXTENDING $OUT), CHANGES $OUT's DATASET NAME TO JOBNAME.

16. JOB TERMINATION CHANGES ENTRIES IN THE DSC AND SDT OUTPUT QUEUE FOR DATASET EXAMPLE.

17. SCP STAGES EXAMPLE TO MAINFRAME A, ETC.

18. JOB TERMINATION RELEASES ALL LOCAL DATASETS, DELETES $CS, AND TERMINATES JOB.

SYSTEM INTERACTION

JOB ENTRY

1. THE STATION CALL PROCESSOR (SCP) TASK EXAMINES A DATASET HEADER MESSAGE AND DETERMINES A JOB DATASET IS ENTERING THE SYSTEM.

   SCP MOVES THE SEGMENT BUFFER TO THE DISK BUFFER EACH TIME THE SEGMENT BUFFER BECOMES FULL FROM INPUT OF THE SUBSEGMENTS.

   SCP REQUESTS THE JOB DATASET DISK BUFFER BE WRITTEN TO DISK BY THE DISK QUEUE MANAGER (DQM) TASK.

2. THE DISK QUEUE MANAGER (DQM) TASK ALLOCATES DISK SPACE FOR THE JOB DATASET ON THE INITIAL CALL, AND HAS THE DISK BUFFER WRITTEN TO DISK VIA THE DISK DRIVER.

3. UPON REQUEST FROM THE STATION CALL PROCESSOR (SCP) TASK, THE JOB CLASS MANAGER (JCM) TASK DETERMINES WHICH JOB CLASS QUEUE ON THE SYSTEM DATASET TABLE (SDT) THE JOB SHOULD BE ENTERED IN.

4. UPON REQUEST FROM THE STATION CALL PROCESSOR (SCP) TASK, THE PERMANENT DATASET MANAGER (PDM) TASK CREATES A PERMANENT DATASET STRUCTURE FOR THE JOB AND SETS THE DISPOSITION CODE (DC) EQUAL TO ASCII CHARACTERS IN.

5. THE JOB SCHEDULER (JSH) TASK SELECTS THE HIGHEST PRIORITY JOB FROM THE INPUT QUEUE WHOSE JOB CLASS HAS A JOB EXECUTION TABLE (JXT) ENTRY AVAILABLE.

   JSH CONSTRUCTS THE JOBS JXT ENTRY AND CHANGES THE SYSTEM DATASET TABLE (SDT) INPUT QUEUE ENTRY TO EXECUTE QUEUE AND SETS THE JOB STATE TO THE QUEUED (Q) STATE IN THE JXT.

INITIAL JOB PREPARATION

1. THE JOB SCHEDULER (JSH) TASK SELECTS A JOB FROM THE JOB EXECUTING TABLE (JXT) THAT HAS BEEN GIVEN ITS INITIAL MEMORY ALLOCATION.

2. THE JOB SCHEDULER (JSH) ZEROES OUT THE JOB TABLE AREA (JTA) AND JOB COMMUNICATION BLOCK (JCB) OF THE JOB.

3. THE JOB SCHEDULER (JSH) INITIALIZES THE JOB COMMUNICATION BLOCK (JCB) POINTERS SUCH AS HLM, FL, DSP, ETC. AND SAVES THE POINTER TO THE JCB IN THE JOB TABLE AREA (JTA).

4. THE JOB SCHEDULER CREATES DATASET NAME TABLE (DNT) ENTRIES FOR $CS, $LOG, $IN, AND $OUT IN THE USERS JOB TABLE AREA (JTA).

5. THE JOB SCHEDULER INITIALIZES THE JOBS ROLLFILE DATASET NAME TABLE (DNT) ENTRY AT THE END OF THE JOB EXECUTION TABLE (JXT) ENTRY.

6. THE JOB SCHEDULER INITIALIZES THE ROLL JOB INDEX (RJI) TABLE ENTRY FOR THE JOB FOR POSSIBLE RECOVERY.

7. THE JOB SCHEDULER (JSH) SETS THE JOBS STATE TO WAITING (W) IN THE JXT THUS ALLOWING THE JOB TO CONTEND FOR THE CPU ALONG WITH OTHER ACTIVE JOBS IN THE JXT.

JOB ROLLOUT

1. THE JOB SCHEDULER (JSH) TASK DETERMINES A NEED BY A JOB FOR USER MEMORY AND CREATES A REQUEST TO HAVE ANOTHER JOB IN MEMORY ROLLED-OUT.

   JSH SEARCHES THE JOB EXECUTION TABLE (JXT) FOR A CANDIDATE FOR JOB ROLL-OUT.

   JSH AFTER LOCATING A CANDIDATE MAY HAVE TO COMPACT MEMORY TO OBTAIN A LARGE ENOUGH FREE SEGMENT OF MEMORY.

   JSH SETS THE PROPER JOB STATES OF THE WAITING JOB AND THE JOB ABOUT TO BE ROLLED-OUT.

2. THE DISK QUEUE MANAGER (DQM) TASK ALLOCATES DISK SPACE IF NEEDED, AND HAS THE MEMORY RESIDENT JOB AREA WRITTEN TO DISK VIA THE DISK DRIVER.

3. THE JOB SCHEDULER (JSH) TASK UPDATES THE MEMORY SEGMENT TABLE (MST).

4. THE JOB SCHEDULER (JSH) TASK LIFTS THE SUSPENSION ON THE JOB WAITING FOR MEMORY LIBERATION.

ACCESS A PERMANENT DATASET

1. A JOB MAKES A REQUEST FOR PERMISSION TO USE A PERMANENT DATASET.

2. THE EXCHANGE PACKAGE PROCESSOR (EXP) TASK EXAMINES THE REQUEST FOR VALID PARAMETERS AND ISSUES A REQUEST TO THE PERMANENT DATASET MANAGER (PDM) TASK FOR ACCESS AND THE JOB SCHEDULER (JSH) TASK FOR JOB SUSPENSION.

3. THE PERMANENT DATASET MANAGER (PDM) TASK VALIDATES ACCESS TO THE SELECTED DATASET FOR THE USER JOB.

   PDM MANAGES THE NECESSARY SYSTEM TABLES SO THAT THE USER HAS THE DESIRED PERMISSIONS GRANTED.

   PDM CREATES A PERMANENT DATASET TABLE (PDS) ENTRY FOR THE ACTIVE DATASET.

   PDM CREATES A DATASET NAME TABLE (DNT) ENTRY FOR THE USER AND INSURES THE DATASET ALLOCATION TABLE (DAT) FOR THE DATASET IS MADE RESIDENT IN THE REQUESTING JOB AREA THUS MAKING THE DATASET LOCAL FOR THAT JOB.

4. THE JOB SCHEDULER (JSH) TASK SETS THE JOBS STATUS BIT TO SUSPK (K) THUS DISALLOWING THE JOB FROM ROLL-OUT AND MEMORY MOVEMENT UNTIL THE JOBS DATASET ALLOCATION TABLE (DAT) ENTRY IS READ INTO THE USER AREA FROM THE DISK DATASET CATALOG (DSC).

   JSH RESETS THE JOBS STATUS TO THE WAIT (W) STATE FOLLOWING COMPLETION OF GENERATION OF THE LOCAL DATASET BY THE PERMANENT DATASET MANAGER (PDM) TASK.

DATASET ACQUISITION

1. A JOB MAKES A REQUEST TO ACQUIRE A PERMANENT DATASET.

2. THE EXCHANGE PACKAGE PROCESSOR (EXP) TASK EXAMINES THE REQUEST FOR VALID PARAMETERS AND ISSUES A REQUEST TO THE PERMANENT DATASET MANAGER (PDM) TASK TO SEARCH FOR AN EXISTANT PERMANENT DATASET.

   IF THE DATASET ALREADY EXISTS IN THE SYSTEM, THE FLOW FOLLOWS ACCESS PROCESSING.

   IF THE DATASET DOES NOT EXIST WITHIN THE SYSTEM EXP ISSUES A REQUEST TO THE JOB SCHEDULER (JSH) TASK TO SUSPEND THE JOB AND EXP ENQUEUES THE DATASET NAME AND PLACES IT ON THE REQUEST QUEUE IN THE SYSTEM DATASET TABLE (SDT).

3. THE PERMANENT DATASET MANAGER (PDM) TASK SEARCHES THE DATASET CATALOG (DSC) FOR AN EXISTANT DATASET AND IF SO TO VERIFY THE PERMISSION WORDS.

   PDM MONITORS NECESSARY TABLES AND USER AREAS ENABLING THE DATASET TO BECOME LOCAL TO THE JOB THAT ISSUED THE ACQUIRE AS IN THE ACCESS FLOW.

4. IF THE JOB SCHEDULER (JSH) TASK WAS REQUESTED IT SUSPENDS THE JOB AWAITING THE DATASET TRANSFER FROM THE FRONT-END.

5. IF THE DATASET IS NOT CRAY RESIDENT THE STATION CALL PROCESSOR (SCP) TASK SELECTS THE REQUEST ENTRY FROM THE SYSTEM QUEUE AND COMMUNICATES WITH THE FRONT-END VIA THE FRONT-END DRIVER TO RETRIEVE THE NON-RESIDENT DATASET.

   SCP THEN COMMUNICATES WITH THE JOB SCHEDULER (JSH), THE PERMANENT DATASET MANAGER (PDM) AND THE DISK QUEUE MANAGER (DQM) TO INSURE PROPER SAVING OF THE DATASET AND JOB ADVANCEMENT.

LOCAL I/O

1. A JOB ISSUES A WRITE REQUEST THROUGH A SYSTEM MACRO CAUSING PHYSICAL I/O.

2. THE EXCHANGE PACKAGE PROCESSOR (EXP TASK EXAMINES THE REQUEST FOR VALID PARAMETERS AND FORWARDS THE REQUEST FOR I/O.

3. THE TASK I/O (TIO) ROUTINES TRANSFER DATA, INSERTING RCW's WHERE NEEDED, FROM THE JOB DATA AREA TO THE JOB DISK BUFFERS ON A WRITE.

   TIO DETERMINES THE DISK BUFFERS ARE GREATER THAN HALF FULL AND PERFORMS CIRCULAR I/O (CIO).

   CIRCULAR I/O (CIO) ISSUES A WRITE REQUEST OF THE DISK QUEUE MANAGER (DQM) TASK AND HAS THE JOB SUSPENDED, STATE I, BY THE JOB SCHEDULER (JSH).

4. THE DISK QUEUE MANAGER (DQM) TASK ALLOCATES DISK ON A WRITE IF NEEDED, AND PERFORMS THE I/O VIA THE DISK DRIVER.

JOB MEMORY REQUEST

1. A JOB MAKES A REQUEST TO ALLOCATE/DEALLOCATE MEMORY THROUGH AN RFL CONTROL STATEMENT.

2. THE EXCHANGE PACKAGE PROCESSOR (EXP) EXAMINES THE USERS JOB COMMUNICATION BLOCK (JCB) FOR VALID PARAMETERS SUCH AS HLM, FL, ETC.

   EXP REQUESTS THE JOB SCHEDULER (JSH) TASK FOR THE ACTUAL ALLOCATE/DEALLOCATE OF MEMORY.

3. THE JOB SCHEDULER (JSH) TASK UPON A DEALLOCATE REQUEST MUST UPDATE USER FIELDS SUCH AS LA, HLM, DSP POINTERS, ETC.

   JSH RETURNS THE RELEASED MEMORY TO THE SYSTEM BY UPDATING THE MEMORY SEGMENT TABLE (MST) AND FILLING THE RETURNED MEMORY WITH THE CONTENTS OF I@ERASE SHIFTED LEFT 24 BITS.

4. THE JOB SCHEDULER (JSH) TASK UPON AN ALLOCATE REQUEST MUST SEARCH THE MEMORY SEGMENT TABLE (MST) FOR FREE MEMORY.

5. IF FREE MEMORY IS FOUND ADJACENT TO THE USER JOB THE MEMORY MAY BE GIVEN TO THE USER BY UPDATING VARIOUS USER FIELDS SUCH AS LA, HLM, ETC. AND UPDATING THE MEMORY SEGMENT TABLE (MST).

6. IF THE JOB SCHEDULER (JSH) TASK IN ITS SEARCH OF THE MEMORY SEGMENT TABLE (MST) CANNOT FIND MEMORY ADJACENT TO THE USER JOB, MEMORY COMPACTION OR ROLL-OUT MAY HAVE TO BE USED.

   IF SUCH IS THE CASE THE REQUESTING JOB IS SET TO THE WAITING (W) STATE AND THE REQUESTING JOB ITSELF BECOMES SUBJECT TO ROLL-OUT.

JOB TERMINATION

1. A JOB TERMINATES WITH AN EXIT CONTROL STATEMENT INTERPRETED BY THE CONTROL STATEMENT PROCESSOR (CSP).

2. CSP POSTS TERMINATION MESSAGES TO THE LOGFILES VIA THE MESSAGE (MSG) TASK.

3. THE EXCHANGE PACKAGE PROCESSOR (EXP) TASK COPIES THE JOBS $LOG TO $OUT.

   EXP CHANGES $LOG TO SCRATCH AND RELEASES SCRATCH DATASETS AS WELL AS $IN, $CS.

4. THE PERMANENT DATASET MANAGER (PDM) CREATES A DATASET CATALOG ENTRY (DSC) FOR EACH JOB OUTPUT DATASET THUS MAKING THE DATASET PERMANENT.

5. EXP HAS CREATED ENTRIES FOR THE JOBS OUTPUT DATASETS IN THE SDT.

6. THE DISK QUEUE MANAGER (DQM) TASK ALLOCATES DISK FOR $OUT IF NEEDED, AND HAS THE BUFFER WRITTEN TO DISK VIA THE DISK DRIVER.

7. THE JOB SCHEDULER (JSH) TASK TERMINATES THE JOB BY RELEASING THE EXECUTING QUEUE ENTRY IN THE JOB EXECUTION TABLE (JXT).

   JSH RELEASES THE JOBS MEMORY AND RELEASES THE JOBS DATASET AND UPDATES THE MEMORY SEGMENT TABLE (MST).

8. THE STATION CALL PROCESSOR (SCP) TASK MANAGES SYSTEM MEMORY BUFFERS AND TRANSMITS THE JOBS OUTPUT DATASETS TO THE FRONT-END VIA THE FRONT-END DRIVER.

9. UPON RECEIPT OF EACH OUTPUT DATASET BY THE FRONT-END, THE STATION CALL PROCESSOR (SCP) REQUESTS THE PERMANENT DATASET MANAGER (PDM) TO DELETE THE DATASET CATALOG (DSC) ENTRY FOR THE DATASET AND DQM RELEASES THE DATASETS DISK SPACE.

Cray Research, Inc.
Corporate Addresses

CORPORATE HEADQUARTERS

1440 Northland Drive
Mendota Heights, MN 55120
Tel: 612-452-6650
TLX 2[illegible]3444

≈

THE CHIPPEWA FACILITIES

Manufacturing:
Highway 178 North
Chippewa Falls, WI 54729
Tel: 715-723-2221
TWX 910 0285 1699

Engineering:
Highway 178 North
Chippewa Falls, WI 54729
Tel: 715-723-5501

CRAY LABORATORIES

Cray Labs Headquarters
5311 Western Avenue
Boulder, CO 80301
Tel: 303-449-3351

Hallie Lab
P.O. Box 169
Chippewa Falls, WI 54729
Tel: 715-723-0266

SALES OFFICES

Eastern Regional Sales Office
10750 Columbia Pike, Suite 602
Silver Spring, MD 20901
Tel: 301-681-9626

≈

Central Regional Sales Office
5330 Manhattan Circle, Suite F
Boulder, CO 80303
Tel: 303-499-3055

Houston Sales Office
3121 Buffalo Speedway, Suite 400
Houston, TX 77098
Tel: 713-877-8053

Austin Sales Office
3415 Greystone, Suite 201
Austin, TX 78731
Tel: 512-345-7034

≈

Western Regional Sales Office
Sunset Office Plaza
1874 Holmes Street
Livermore, CA 94550
Tel: 415-447-0201

Los Angeles Sales Office
101 Continental Boulevard, Suite 456
El Segundo, CA 90245
Tel: 213-640-2351

Seattle Sales Office
536A Medical Dental Building
2728 Colby Avenue
Everett, WA 98201
Tel: 206-259-5075

INTERNATIONAL (SUBSIDIARIES)

Cray Research (UK) Limited
James Glaisher House
Grenville Place
Bracknell, UK
Tel: 44-344-21515
TLX: 848841

Cray Research GmbH
Wartburgplatz 7
8000 Munich 40
West Germany
Tel: 49-89-3630-76
TLX: 05213211

Cray Research Japan, Limited
Shin Aoyama Building, West 1661
1-1 Minami-Aoyama 1-chome
Minato-ku, Tokyo 107 Japan
Tel: 81(03)403-3471

CRAY RESEARCH, INC.