INTERAKTIV PROGRAMMERING VID FLERA ANVÄNDARE

ALLAN BJÖRK

RE-160 Juni 1975
Inst. för Reglerteknik
Lunds Tekniska Högskola

INTERAKTIV PROGRAMMERING VID FLERA ANVÄNDARE.

A BJÖRCK

HANDLEDARE: J WIESLANDER

INNEHÅLL

Sammanfattning.

Målet med examensarbetet var att göra RSX tme-sharingvänligt. För att uppnå detta har vissa modifieringar och tillägg gjorts:

Ett nytt systemdirektiv - SHARE - som förklarar att ett program är klart att placeras på skivminnet - att swappas.

Tre nya programtillstånd - status sju, åtta och nio - vilka talar om huruvida ett program är klart att swappas, håller på att swappas eller är swappat.

Programmen TS.... och TSLOAD. TS.... är ett kommunikationsprogram mellan systemet och terminalerna. Det ser till att program swappas om det är nödvändigt. TSLOAD laddar in swappade program och återställer vissa register.

Abstract.

The purpose of this paper was to make certain changes in the real time monitor RSX to make time-sharing possible.

The following features were added:

A new system directive - SHARE - which declares a task ready to be swapped.

Three new programstatus - status seven, eight and nine. They indicate whether the task is ready to be swapped, it is being swapped and it has been swapped respectively.

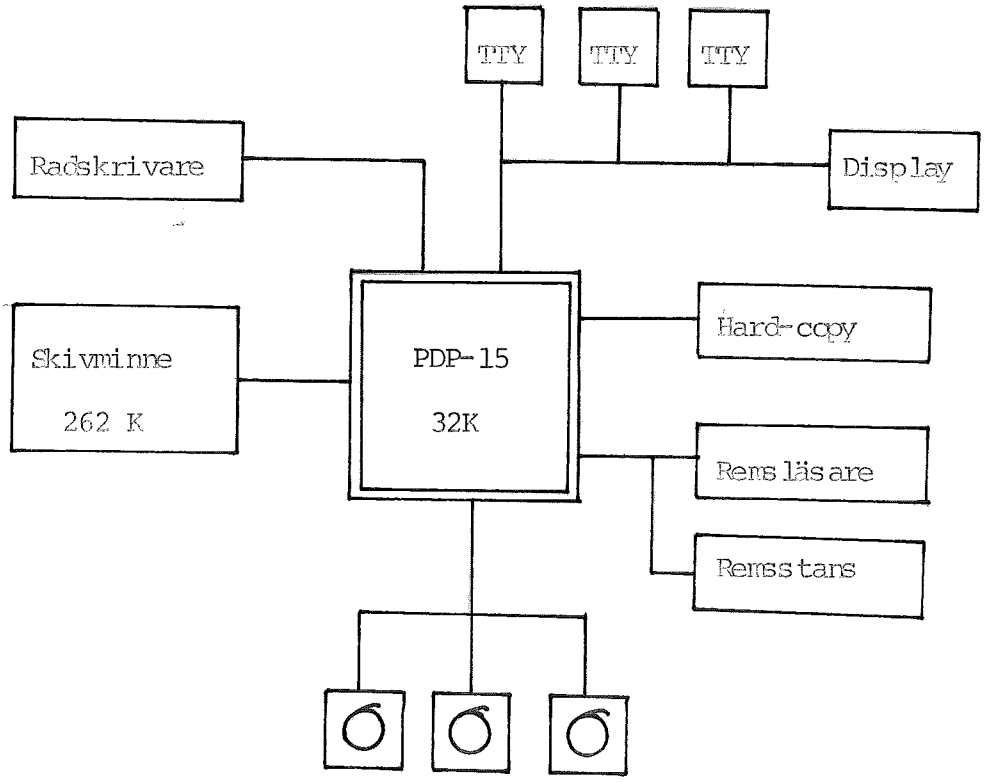
The programs TS.... and TSLOAD. TS.... is a link between the system and the users. If necessary it can make programs being swapped. TSLOAD loads swapped programs into core and restores certain registers.

TIMESHARING

Med timesharing avses i examensarbetet att ett program, som är inaktivt p g a att det ligger och väntar en yttre enhet, skall kunna swappas om ett annat program behöver ha access till kärnminnet.
Varje användare har egna filer på massminnet. Dessa skall inte nås av någon annan användare.

2. Hårdvara.

Examensarbetet utfördes på institutionens dator - en PDP-15. Den har 18 bitars ordlängd och 32K kärnminne, vilket kan byggas ut till 128K.
Datorn är utrustad med ett skivminne på 262K.
Radskrivare, teletyper, display, hardcopy, bandstationer, remsläsare och remsstans är anslutna till systemet.



3. RSX - Real Time System Executive.

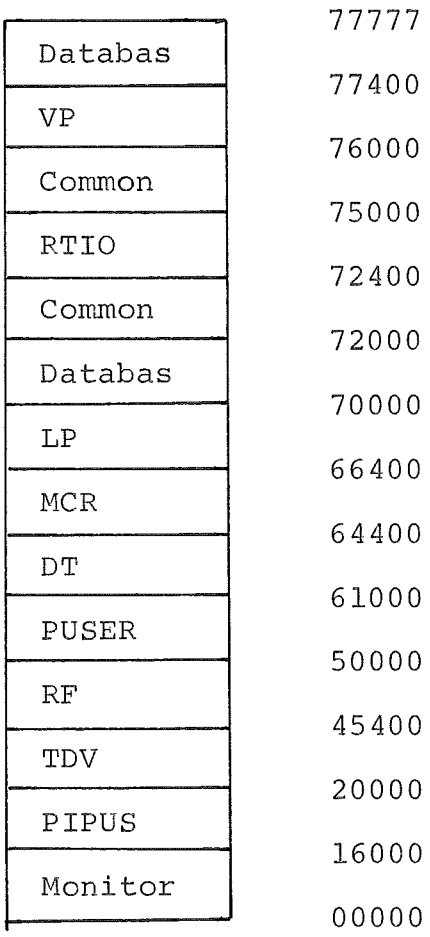
RSX är ett reelltidssystem, som ger möjligheter att utveckla, installera och exekvera program.

3.1 Kärnminnet under RSX.

Kärnminnet är under RSX uppdelat på följande sätt:

1. Avdelningar för program.
2. Common areor.
3. Informationslistor.
4. Exekutiven.

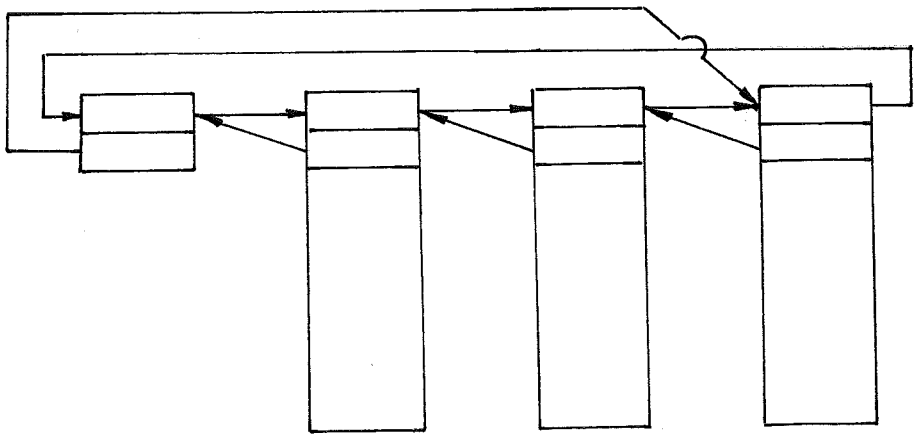
Alla program - användar- som systemprogram - är gjorda för att exekvera i speciella delar av kärnminnet s k avdelningar. Dessas storlek och placering preciseras vid uppstarten av systemet och är sedan fixerade.
Den nuvarande uppdelningen visas i figuren nedan.



Uppdelning av kärnminnet under RSX.

Avdelningarna VP, LP och DT innehåller drivrutiner för displayen, radskrivaren och bandstationerna.
I RF finns en filhanteringsrutin för skivminnet.
PIPUS, PUSER och TDV är avdelningar, som är avsedda för användaren. I TDV exekverar alla programutvecklingsrutinerna och det är för program, som exekverar i denna avdelning, "timesharingen" i examensarbetet är implementerad.

För att få information om det aktuella läget i systemet använder RSX dubbelt länkade listor.
Dessa består av listhuvuden, som är två intilliggande celler med fixa addresser, och vektorer eller noder vilka består av tio konsekutiva celler på godtycklig plats i kärnminnet.
De två första cellerna är pekare. Den första cellen innehåller adressen till nästa nod i listan och den andra adressen till närmast föregående nod.
En uppdatering av en lista sker lätt. Endast genom att ändra pekarna utesluts noder ur resp sätts in i en lista.



Lista i RSX.

De viktigaste listorna i RSX är:

STL - System Task List - i vilken det finns en nod för varje program, som förts in i systemet, med information om programmet.

ATL - Active Task List - är en lista över de aktiva programmen i systemet. Varje program representeras med en nod. Programmen är införda i prioritetsordning.
I noden finns följande information om programmen:

Programnamn.
Programmets prioritet.
PBDL nod adressen (Se nedan).
STL nod adressen.
Programstatusen.
Startadressen för programmet.

Statusen - programtillståndet - talar om för monitorn vad den skall göra när den träffar på programmet:

Status 1 Programmet finns på skivminnet och väntar på att dess avdelning skall bli ledig.

Status 2 Avdelningen ledig - starta inläsning från skivminnet.

Status 3 Inläsning pågår.

Status 4 Programmet klart att startas.

Status 5 Exekvering pågår.

Status 6 Exekveringen uppskjuten tills vidare.

PBDL - Partition Block Description List - är en lista över alla avdelningarna i kärnminnet.
I PBDL-noden finns det en registerräddningsarea, som används vid programavbrott. Dessutom lagras vissa variabler, som är unika för varje program, i noden. Dessa måste räddas om programmet swappas då det nya programmets variabler är annorlunda.

PDVL - Physical DeVice List - är en lista över de fysiska enheterna i systemet.
I PDVL-noden finns ett listhuvud för de I/O-operationer enheten skall göra.

POOL - innehåller alla outnyttjade noder.

3.2 Relltidsmonitorn RSX.

RSX har fyra huvudbeståndsdelar:

1. Programutvecklingsrutiner.
2. Operatörskommunikation.
3. I/O-rutiner.
4. Monitorn eller exekutiven.

Programutvecklingen sköts av ett kärnminnesresident program - TDV - i exekutiven. Det kallar på kommandon från användaren in editorer, kompilatorer och andra nödvändiga rutiner.

Operatörskommunikationen sköts av ett interaktivt program - MCR.

Monitorn övervakar exekveringen av program, sköter kärnminnes- och skivminnesallokering resp deallokering och I/O-köande.

Vad systemet skall göra avgör monitorn genom att undersöka den aktiva listan m a p programmens status. Undersökningen startar från "toppen" så att program med hög prioritet behandlas först.
När ett program är i status ett undersöks om avdelningen i kärnminnet där programmet skall exekvera är ledig.
Om avdelningen är upptagen får programmet vänta och nästa program i listan undersöks.
Är avdelningen ledig sätts statusen till två. Det innebär att inläsning av programmet från skivminnet till kärnminnet startas och statusen sätts till tre.
När inläsningen är klar sätts statusen till fyra , vilket betyder att programmet är klart att startas av monitorn.
Status fem innebär att programmet håller på och exekverar. Om ett programavbrott kommer skall vissa register räddas.
Vid status sex är exekveringen uppskjuten tills vidare.

3.3 I/O-operationer.

När ett program skall göra en I/O-operation sker detta på en logisk enhet. Den fysiska enheten är sedan associerad till den logiska.
Detta gör systemet flexibelt. Om en yttre enhet faller ifrån associeras bara en ny enhet till den logiska och operationen kan fortsätta.

Varje fysisk enhet har en drivrutin. I denna finns en kö - request-kön - för de operationer enheten skall utföra.

När ett program skall göra en I/O-operation på en logisk enhet placeras en nod, vilken innehåller de data som behövs för operationens utförande, i request-kön hos den drivrutin som är associerad till den logiska enheten.
Requestkön undersöks var gång monitorns undersökning av den aktiva listan når till drivrutinen.
Då tas den första noden ur kön och operationen utförs.
När denna är klar så initieras nästa tills kön är tom.

3.4 Kommunikation program-monitor.

För att kommunicera med monitorn använder programmen direktiv, som utförs genom s k CAL-instruktioner.
CAL är en instruktion, som överför kontrollen till monitorn via en fix adress i kärnminnet.
I CAL-instruktionen ingår en adress till ett "CAL CPB" - CAL Parameter Block - som innehåller de variabler monitorn behöver för att utföra direktivet.
När ett program gör ett monitoranrop med en CAL-instruktion måste det rädda innehållet i de hådvaruregister, som kan behövas efter anropet. Inga register får nämligen förväntas ha samma värden efter monitoranropet som det hade innan.

4. "TSX - Time Share Executive".

RSX har i examensarbetet modifierats så att det tillåter att program swappas. En allmän beskrivning av det nuvarande systemet och proceduren att swappa ett program följer nedan samt sedan en närmare beskrivning av de ingående delarna.

4.1 "Varför swappa program?".

Ett program befinner sig normalt på skivminnet tills det kallas in. När det kallats in stannar det i kärnminnet tills det har exekverat färdigt. Swapping är inte möjligt.

Den enda effektivitetshöjande faciliteten i det nuvarande RSX är multiprogrammering d v s flera program kan samtidigt ligga i kärnminnet i olika avdelningar. När ett program väntar på någon yttre enhet kan ett annat exekvera under tiden.
Detta gör att programutvecklingsfunktioner, vilka vi främst haft i åtanke för att införa en time-sharing facilitet, skulle kunna exekveras parallellt i olika avdelningar.
I det nuvarande RSX kan inte olika användare göra detta oberoende av varandra. Kommandorader till TDV (se avsnitt 3.2) kan bara ges från en enhet - TDV-teletypen.
Ett sådant arrangemang är dessutom kärnminneskrävande. Det går ut över andra avdelningar och common block, som får utgå för att ge plats åt den extra avdelningen. I det nuvarande systemet går det därför inte att tillåta en sådan kärnminnesdisposition.

Nackdelarna försvinner om man tillåter att program swappas t ex när de väntar på någon I/O-operation. Programmen kommer bara att utnyttja en gemensam area i kärnminnet.

Exempel: Tiden att swappa avdelningen TDV och ladda in den igen är av storleksordningen 1 sekund. Om en operatör skall skriva en rad på en teletype tar detta några sekunder. Behöver han dessutom tänka innan han skriver inses vilka vinster, som kan göras.

4.2 "Swapping".

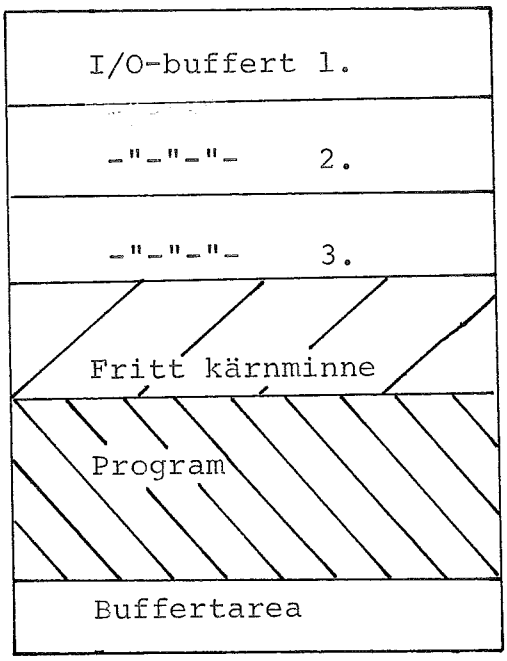
"Swapping" innebär att ett program kan placeras på ett massminne när ett annat program vill ha access till kärnminnet.

För att ett program skall få swappas måste vissa vilkor vara uppfyllda:

1. Programmet skall vara i status sju.
2. Det finns ett annat program, som vill ha access till avdelningen TDV.
3. En räddningsarea skall finnas på skivminnet.

Att programmet är i status sju betyder att systemet förklarat att det inte finns några hinder för att programmet swappas.
Status sju sätts av ett systemdirektiv - SHARE.

Om ett program swappas eller ej avgörs när den aktiva listan (ATL) undersöks av monitorn. Finns det ett program i status sju ser den efter om något swappat program vill in. Det görs genom att monitorn undersöker TSQ - en speciell lista över alla swappade program - om det finns något som är klart att komma in.
Finns inget kontrollerar monitorn om något nytt program, som exekverar i TDV, vill in. Så är fallet om en speciell variabel kallad RNTSEV är satt.
När ett annat program vill in swappas det inneliggande programmet. Detta görs genom att en nod över programmet placeras i TSQ. I noden räddas en del data ur PBDL-noden, som kommer att förändras när ett nytt program kommer in.
Därefter reserveras en i förväg iordningställd räddningsarea på skivminnet för programmet genom att den nod, som representerar arean i DPQ, tas ur DPQ. DPQ är en lista över tillgängliga räddningsareor.
Sedan startas överföringen av avdelningen TDV från kärnminnet till skivminnet. Avdelningen som helhet förs alltså ut på skivminnet oberoende av det inneliggande programmets storlek. Anledningen till detta är hur en avdelning utnyttjas. Se figuren nedan.



Swappar man hela avdelningen kommer automatiskt eventuella buffertareor med. Inga kontroller eller extra överföringar behöver göras.
När överföringen av det inneliggande programmet är klar undersöks ATL från toppen för att programmet med högst prioritet skall komma in.
Swappade program laddas in av ett speciellt program i monitorn - TSLOAD. Det startas av monitorn när den finner ett swappat program, vars I/O-operationer är klara.

4.3.1 TSQ - Time Share Queue.

TSQ är en lista över alla swappade program.
Program förs in i listan när monitorn hittar ett program i status sju och ett annat program som vill in.
Programmet TSLOAD tar bort program ur listan sedan de laddats in i kärnminnet.
Varje program representeras av en nod. I noden räddas vissa data ur programmets "partition block":

1. Programstorlek. x/
2. Antal pågående I/O-operationer.
3. Virtuell programstorlek.
4. Antal I/O-buffrar.
5. JEA-registret.

Dessutom lagras ATL- och DP-nodsadresserna i noden.
Några andra register behöver inte räddas då programmet för att vara i status sju måste ha gjort ett monitoranrop. Vid ett sådant kan inga register förväntas vara densamma utan programmet får rädda de de register det behöver.

Listning se appendix A.

4.3.2 DPQ - Disk Parameter Queue.

När ett program skall swappas behövs det utrymme på skivminnet. Att göra detta var gång ett program skall ut är en tidsödande operation. Det enda monitorn kan komma att göra är att allokera skivminne om det är stor input-output.
Därför har DPQ införts. Det är en lista med färdigställda räddningsareor.
Dessa kan antingen allokeras vid uppstarten av systemet eller också kan en räddningsarea iordningställas varje gång request görs på ett program. I det senare fallet får deallokering ske vid exit.
När monitorn tänker swappa ett program tas en nod ur DPQ, vilket reserverar den räddningsarean för programmet, som skall swappas. Adressen till DPQ-noden räddas i TSQ-noden.
När det swappade programmet laddats in av TSLOAD sätts noden åter in i DPQ.
Varje räddningsarea representeras av en nod med följande innehåll:

1. Sidan på skivminnet.
2. Adressen på skivminnet.
3. Reserverat för kärnminnesadressen.
4. Storleken på programmet (= TDV's storlek: 25400 oktalt).

Dessutom används ett ord för att rädda ett register som int fått plats i TSQ-noden.

Listning se appendix B.

4.3.3 SHARE direktivet.

Direktivet SHARE placerar ett program i status sju d v s förklarar att det är klart att placeras på skivminnet om ett annat program skulle vilja använda avdelningen TDV.

SHARE finns i två former. I det första fallet har programmet självt förklarat sig villigt att stiga åt sidan - "selfissued directive" - genom en inprogrammerad instruktion. Det är främst avsett för beräkningsprogram med låg prioritet. När ett program med hög prioritet gör "share" på sig själv kan det förorsaka att det bara åker in och ut så fort något annat program vill ha access till TDV.
I det andra fallet är det ett systemprogram som tillåts köra ut ett inneliggande program genom en SHARE-instruktion. Att sätta status sju i detta fall tillåts endast om det inneliggande programmet väntar på en yttre enhet. Detta för att garantera att programmet inte har några anspråk på innehållet i hårdvaruregistrena. När ett program väntar på en yttre enhet är nämligen programmets sist utförda instruktion ett monitoranrop. Behövs något register efter anropet skall det vara räddati programmet.
Att ett program är fixerat i kärnminnet är inget hinder för att swappa det. Systemet utför inga tester om detta.

Listning se appendix C.

4.3.4 Status sju.

Om ett program är i status sju är det klart att swappas. Då monitorn vid undersökningen av den aktiva listan (ATL) finner ett program i status sju kontrollerar den om något annat program behöver avdelningen genom att undersöka dels TSQ för att se om något swappat program är klart dels om "time-sharing"-händelsevariabeln är skild från noll, vilket indikerar att ett program blivit efterfrågat och vill in.

Skulle det finnas någon som vill in swappas programmet, en del register i programmets "partition block" räddas och en nod över programmet placeras i TSQ. Avdelningen förklaras ledig och den aktiva listan undersöks från toppen.

Några andra register än de ovan nämda räddas inte då programmet för att vara i status sju måste ha gjort ett monitoranrop och därmed avsagt sig rätten till registerinnehållena.

Listning se appendix D.

4.3.5 Status åtta.

När ett program är i status åtta håller det på att placeras på skivminnet. Monitorn testar om detta är klart och sätter i så fall programstatusen till nio.

Listning se appendix E.

4.3.6 Status nio.

Programmet är swappat. Monitorn undersöker om avdelningen är ledig och om programmet är klart att komma in i igen. I så fall startas programmet TSLOAD (se nedan), vilket återför programmet till kärnminnet.

Listning se appendix F.

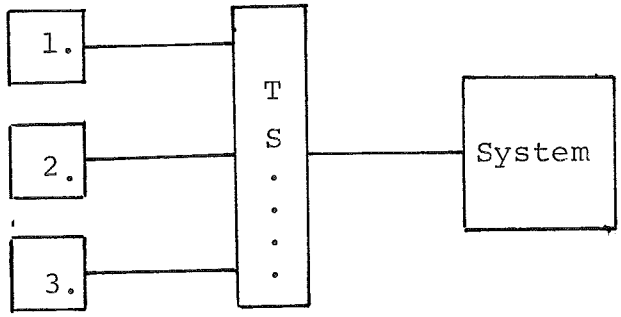
4.3.7 TSLOAD.

TSLOAD är ett kärnminnesresident program, vilket har en egen avdelning i exekutiven. Det hämtar in swappade program och återställer de i TSQ-noden räddade registrena i programmets PBDL-nod.
Den använda räddningsarean på skivminnet förklaras ledig genom att dess nod åter sätts in i DPQ.
TSLOAD startas av monitorn när den finner ett swappat program, vilket är klart att komma in.

Listning se appendix G.

4.3.8 TS.... .

Programmet TS.... sköter input/output-operationerna mellan teletyperna och systemet. TS.... kallas i fortsättningen TS av skrivtekniska skäl.



När ett program vill skriva eller läsa på en teletype köas operationen i en kö - request-kön - hos TS istället för som tidigare hos enheten själv.
Om ett program vill skriva överförs programmets buffert till en buffert i TS. TS ger sedan klartecken till programmet, som uppfattar att operationen är klar. Det kan fortsätta att exekvera.
TS skriver sedan på rätt teletype. Detta görs genom att request-kön undersöks och operationerna identifieras. Dessa överförs sedan till användarköer - varje användare har en kö för sina I/O-operationer.
Från dessa initieras sedan operationerna en och en.Var gång en operation är klar undersöks dessa köer av TS om det finns någon ny operation, som skall startas.

Vill ett program göra en läs-operation får det besked att vänta. TS läser in teletypebufferten till en buffertarea i TS. Därefter kontrollerar TS att programmet inte är swappat. Är så fallet förklaras programmet klart att komma in. När programmet är i kärnminnet överförs TS-bufferten till programmets buffert.

Om för många skrivoperationer eller en läsoperation köats för en användare gör TS "share" på programmet ifråga och väntar på att den operationen skall bli klar.

Sedan överförs inga operationer från request-kön till användarkön förrän TS dels blivit klar med operationerna ifråga och dels kontrollerat att programmet finns i kärnminnet. Detta är nödvändigt då ett annat program kanske ligger i avdelningen. Det kan då få felaktiga data i sina celler.

Listning se appendix H.

5. Uttestning.

En fullständig testning av det nya systemet har inte varit möjlig då de gamla systemprogrammen inte är gjorda så att de accepterar swapping av användarprogram.

Den testning som gjordes skedde i två etapper. I den första testades direktivet SHARE i "selfissued" form och de nya delarna i monitorn, som sköter status sju, åtta och nio för att se om vi överhuvudtaget var på rätt väg.
Det hela fungerade utmärkt. Körningar med upp till fem program "samtidigt" exekverande i TDV gjordes.
Begränsningen för antalet program tycks bara ligga i hur många räddningsareor man är beredd att släppa till.

I den andra etappen hade TS tillkommit och tilläts göra "SHARE" på inneliggande program. Editorer och några andra stora program swappades bl a och det fungerade bra.
Ett försök att köra två editorer parallellt i TDV misslyckades p g a att "Disk file handlern" inte var anpassad till swappingförfarandet. Den uppfattade det som om editorerna arbetade med samma fil på skivminnet, vilket inte är tillåtet.
I framtiden kommer varje användare att ha egna skivminnesutrymmen. Dessa svårigheter försvinner då.

Det hela tycks alltså kunna fungera med en del kompletteringar i gamla systemprogram.

Appendix A och B.

```
*** FORMAT & CONVENTION DESCRIPTIONS

    /

    /       TSQ - TIME SHARE QUEUE
    /       TSQ IS A DIRECTORY OF SHARED TASKS WHICH ARE SWAPPED
    /       TO DISK
    /       STATUS SEVEN INSERTS NODES IF AN OTHER TASK NEEDS THE
    /       PARTITION.
    /       TSLOAD REMOVES NODES FROM DEQUE.
    /       TSQ IS SCANNED FROM STATUS SEVEN AND FROM TSLOAD BY TSQSCN
    /       FOR TASKS READY TO RESUME EXECUTION.
    /
    /       EACH ENTRY IS A DEQUE NODE OF THE FOLLOWING FORMAT:
    /
    TS.FP=0     / FORWARD LINKAGE
    TS.BP=1     / BACKWARD LINKAGE
    TS.DP=2     / DISK PARAMETER NODE ADDRESS
    TS.AP=3     / ATL NODE ADDRESS
    TS.TP=4     / TASK PRIORITY
    TS.EV=5     / EV ADDRESS
    TS.PT=6     / COUNT OF PENDING TRANSFERS
    TS.VS=7     / VIRTUAL SIZE
    TS.BP=10    / BUFFER POINTER
    TS.JE=11    / JEA REGISTER
    /
    /       DPQ - DISK PARAMETER      QUEUE
    /       WHEN A TASK  IS RUN ON TS-BASIS AN AREA IS ALLOCATED ON
    /       DISK TO SAVE THE TASK IF IT HAS TO BE SWAPPED.
    /       THE NODE IS USED AS CONTROL TABLE BY STATUS SEVEN
    /       AND TSLOAD.
    /       NODE FORMAT:
    /            FORWARD LINKAGE
    /            BACKWARD LINKAGE
    /            DISK PLATTER NUMBER
    /            DISK STARTING ADDRESS
    /            USED FOR CORE STARTING ADDRESS
    /            WORD COUNT = PARTITION SIZE
    /            TASK SIZE SAVE (TEMPORARILY?)
    /            UNUSED
    /            UNUSED
    /            UNUSED
    /
              .EJECT
```

APPENDIX C.

*** 'SHARE DIRECTIVE

```
          .TITLE *** 'SHARE DIRECTIVE
    /
    / THIS DIRECTIVE SETS STATUS TO SEVEN I.E. THE TASK IS
    / READY TO BE SWAPPED TO DISK IF A TASK WITH HIGHER
    / PRIORITY NEEDS THE PARTITION.
    / A FIVE WORD CAL PARAMETER BLOCK IS USED:
    /
    /   CPB    (0)    CAL FUNCTION CODE = (36)
    /          (1)    EVENT VARIABLE ADDRESS
    /          (2)    NAME WORD FIRST HALF
    /          (3)    NAME WORD SECOND HALF
    /          (4)    TSLOADING EVENT VARIABLE
    /
    / IF THE DIRECTIVE IS REJECTED, THE EVENT VARIABLE (IF SPECIFIED)
    / IS SET TO ONE OF THE FOLLOWING NEGATIVE VALUES:
    /
    /         -202    TASK NOT ACTIVE
    /         -203    DIRECTIVE NOT TASKISSUED
    /         -333    THE TASK, THAT IS TO BE SHARED,
    /                 IS NOT IN STATUS THREE.
    /
    / IF THE DIRECTIVE IS ACCEPTED THE TASK STATUS IS SET TO SEVEN
    / AND THE EVENT VARIABLE (IF SPECIFIED) IS SET TO +1.
    / IF THE DIRECTIVE IS SELFISSUED THE ADDRESS OF A POSITIV EVENT
    / VARIABLE IS DEPOSITED IN THE ATKL NODE TO INDICATE THAT THE
    / TASK IS READY TO RESUME EXECUTION IMMEDIATLY.
    /
    / IF CAL NOT SELFISSUED IS THE TSLOADING EV ADDRESS STORED IN THE ATKL
    / NODE. IN THIS CASE THE TASK MUST BE IN STATUS THREE TO BE PUT IN
    / STATUS SEVEN.
    /
    /
    SH.     LAW     -5              / CHECK RANGE
            JMS     CPBRX
            LAC     X11             / CAL TASKISSUED?
            SZA                     /
            JMP     CX203           / NO - EXIT.
            LAC     2,X             / YES - CAL SELFISSUED(I.E.
            SZA                     / NAME WORDS ARE ZERO)?
            JMP     SH.2
            LAC     3,X
            SZA
            JMP     SH.2            / NO.
            LAC     1,X             / YES -
            SNA                     / EV?
            JMP     SH.1            / NO EV
            PAX                     / EV SPECIFIED - SET IT +1
            CLA.IAC
            DAC     0,X
    SH.1    LAC     CURTSK          / SET AN ADDRESS OF A POSITIV
            PAX                     / EV IN ATL NODE TO INDICATE
            LAC     (SH.EV)         / THAT THE TASK IS READY TO
            DAC     A,EV,X          / RESUME EXECUTION.
```

```
*** 'SHARE DIRECTIVE

            .SET6                   / .SET6 TO SCAN ATL FROM TOP
 *G         LAC     (401000)
 *G         ISA
            LAC     (7)             / SET STATUS SEVEN AND JUMP
            JMP     WSCC            / WAIT,WAITFOR&SUSPEND COMMON CODE
    /
    / CAL NOT SELFISSUED, CHECK IF THE TASK IS ACTIVE AND IF IT IS IN
    / STATUS THREE.
    /
    SH.2    LAC     X10             / SCAN ATL FOR TASK NAME
            AAC     +2
            DAC     R2              / NAME DOUBLE WORD ADDRESS
            LAC     (ATKL)          /
            DAC     R1              / DEQUE LIST HEAD
            JMS     SNAM            / NAME FOUND?
            JMP     CX202           / NO - EXIT
            PAX                     / YES - SAVE ATL NODE ADDRESS IN XR
            LAC     A.TS,X
            AND     (377777)        / TASK IN STATUS THREE
            AAC     -3
            SZA
            JMP     CX333
            LAC     A.TS,X          / YES - SET STATUS SEVEN
            AAC     +4
            DAC     A.TS,X
            LAC     X10             / GET TSLOADING EVENT VARIABLE ADDRESS
            AAC     +4              / (IF SPECIFIED)  AND STORE IN ATL NODE.
            DAC     R2
            LAC*    R2
            SZA                     /SPECIFIED?
            DAC     A.EV,X          /YES
            JMP     CXSUC           / SET EV +1 IF SPECIFIED AND EXIT
                                    / CAL SERVICE ROUTINE.
    SH.EV   1
            .LST
            .IFUND  L.SER
            .NOLST
            .LST
            .IFUND L.S789
            .NOLST
            .ENDC
            .EJECT
```

APPENDIX D.

```
*** SIGNIFICANT EVENT RECOGNITION

    /
    / STATUS SEVEN - A CORERESIDENT TASK IS READY TO BE SWAPPED.
    / ENTRY AT API-7 WITH ATL NODE ADDRESS IN XR
    /
    S7      .RTL6                   /RAISE TO LEVEL 6
 *G
 *G         LAC     (400002)
 *G         ISA
            PXL                     /SAVE ATL NODE ADDRESS
            JMS     TSQSCN          /SCAN TSQ FOR TASK READY TO START,
                                    /(R6,X10,XR&AC ALTERED).
            SKP                    / NO TASK READY
            JMP     S7,A           / TASK IN TSQ READY
            LAC     RNTSEV         / HAS ANY TASK RESIDING IN TDV DONE REQUEST.
            SZA
            JMP     S7,A           / YES - REQUEST10
            PLX
            DBK
            LAW     -1             / NOT REQUEST -
            TAD     A,EV,X         / TEST OWN EV IF I/O IS READY
            DAC     X10
            LAC*    X10
            SNA
            JMP     M2             / NOT READY - CONTINUE ATL SCAN
            JMP     S3,B           / READY - GO TO STATUS FOUR
    /
    / A TASK NEEDS ITS PARTITION, SET UP A TSQ NODE AND
    / PUT STATUS SEVEN TASK ON DISK
    /
    S7,A    LAC     (DPQ)          / LIST HEAD ADDRESS FOR PICK
            DAC     R1
            JMS     PICK           / PICK A NODE (R2,R6,XR&AC ALTERED).
            HLT                    / SHOULDN'T HAPPEN - STOP.
            DAC     R4             / SAVE NODE ADDRESS
    S7,B    JMS     PENP           / PICK A NODE FROM POOL (R1,R6,XR&AC ALTERED).
            JMP     S7,B           / POOL EMPTY - STAY AT STATUS SEVEN
            DAC     R2             / SAVE ADDRESS FOR SPRI
            IAC
            DAC     X10            / SET UP TO FILL NODE
            LAC     R4             / SET DPN ADDRESS IN TSQ NODE
            DAC*    X10
            PLX
            PXA                    / ATL NODE ADDRESS
            DAC*     X10
            LAC     A,TP,X         / TASK PRIORITY
            DAC*    X10
            LAC     A,EV,X         / SAVE EVENT VARIABLE ADDRESS IN TSQ NODE.
            DAC*    X10
            LAC     R2             / SET POINTER TO TSQ NODE IN A,EV
            DAC     A,EV,X
    /
    / SAVE PARTITION BLOCK DATA
    /
```

```
*** SIGNIFICANT EVENT RECOGNITION

            LAC      A,PB,X        / ACCESS TO PARTITION BLOCK
            PAX
            LAC     P,TP,X         / COUNT OF PENDING TRANSFERS
            DAC*    X10
            LAC     P,VS,X         / VIRTUAL SIZE
            DAC*    X10
            LAC     P,BP,X         / BUFFER POINTER
            DAC*    X10
            LAC     P,JE,X         / JEA REGISTER
            DAC*    X10
            LAC     R4             / TASK SIZE IS SAVED IN DP NODE.
            AAC     +5
            DAC     X10
            LAC     P,TS,X
            DAC*    X10
            LAC     (TSQ)          / LIST HEAD ADDRESS
            DAC     R1
            JMS     SPRI           / INSERT NODE (R1,R2,R3,R6,XR&AC ALTERED).
    /
    / SET UP TO FILL DISK CONTROL NODE AND CONTROL TABLE
    /
    S7.C    JMS     PENP           / PICK NODE FROM  POOL (R1,R6,XR&AC ALTERED).
            JMP     S7.C           / POOL EMPTY - STAY AT STATUS SEVEN
            DAC     R2             / SAVE ADDRESS FOR SPRI
            IAC
            DAC     X10
            PLX                    / ATL NODE ADDRESS TO XR
            LAC     A,SN,X         / STL NODE ADDRESS
            DAC*    X10
            DZM*    X10            / EXEC MODE INDICATION
            LAC     A,TP,X         / TASK PRIORITY
            DAC*    X10
            LAC     (31)           / FUNTION CODE - PUT
            DAC*    X10
            LAC     A,PB,X
            PAX
            AAC     P,EV           / EV ADDRESS
            DAC*    X10
            DZM     P,EV,X         / CLEAR DISK PUT EV
    /
            LAC     R4             / DP NODE ADDRESS TO AC.
            AAC     +2             / CONTROL TABLE IS IN DP NODE.
            DAC*    X10
            IAC                    / SET CORE STARTING ADDRESS
            DAC    X10
            LAC     P,BA,X         / CORE STARTING ADDRESS
            DAC*    X10
            LAC     (DSKRQ)        / LIST HEAD ADDRESS
            DAC     R1
            JMS     SPRI           / INSERT NODE (R1,R2,R3,R6,XR&AC ALTERED).
    /
            ,INH
            LAC     DSKTG          / SET DISK TRIGGER
```

*** SIGNIFICANT EVENT RECOGNITION

```
        AND     (377777)
        TAD     (400000)
        .ENB
        DAC     DSKTG
        PLX                     / ATL NODE ADDRESS TO XR
        LAC     (400010)        / SET STATUS EIGHT
        DAC     A.TS,X
        DBK                     / DEBREAK TO API-7
        JMP     M6              / A SIGNIFICANT EVENT HAS JUST
                                / OCCURRED- SCAN ATL FROM TOP
        .EJECT
```

```
*** SIGNIFICANT EVENT RECOGNITION

    /
    / TSQSCN - TSQ SCAN. SCAN TSQ FOR TASK READY TO
    / RESTART.
    / CALLING SEQUENCE:
    /   JMS TSQSCN
    / EXIT CONDITIONS:
    /   RETURN AT JMS+1 IF NO TASK
    /   RETURN AT JMS+2 WITH TSQ NODE ADDRESS IN XR
    /   IF TASK FOUND
    / REGISTERS ALTERED:
    /     R6, X10, XR, AC
    /
    TSQSCN  0
            LAC     TSQSCN          /SAVE RETURN ADDRESS
            DAC     R6
            LAC     (TSQ)         / LIST HEAD ADDRESS
            PAX
    SCN1.   LAC     T,FP,X        / ACCESS TO NEXT NODE
            SAD     (TSQ)         / END OF LIST
            JMP*    R6            / YES- NO TASK READY
                                  / RETURN AT JMS+1
            PAX                   / NO
            LAW     -1            / EVENT VARIABLE SET
            TAD     TS.EV,X
            DAC     X10
            LAC*    X10
            SNA                   / TASK READY TO RESTART?
            JMP     SCN1.         / NO - CONTINUE SCAN
            ISZ     R6            / YES - RETURN AT JMS+2
            JMP*    R6
    /
            .LST
            .IFUND L.SER
            .NOLST
            .LST
            .IFUND  L.RER
```

APPENDIX E.

```
*** SIGNIFICANT EVENT RECOGNITION

     /
     / STATUS EIGHT - A TASK IS SWAPPING TO DISK
     / ENTRY AT API-7 WITH ATL NODE ADDRESS IN XR
     /
     S8        .INH
               PXL                    / SAVE ATL NODE ADDRESS IN LR
               LAC     A.PB,X
               PAX                    / ACCESS TO PARTITION BLOCK
               LAC     P.EV,X
               SNA                    / TASK ON DISK
               JMP     M5             / NO - CONTINUE ATL SCAN
     /
     / TASK ON DISK. FLAG THE PARTITION FREE. SET STATUS NINE
     / AND SCAN ATL FROM TOP.
     /
               DZM     P,FW,X         / YES - SET PARTITION FREE
               PLX                    / SET STATUS NINE
               LAC     (400011)
               DAC     A.TS,X
               .ENB
               JMP     M6             / A SIGNIFICANT EVENT - SCAN ATL FROM TOP
     /
               .EJECT
```

APPENDIX F.

```
*** SIGNIFICANT EVENT RECOGNITION

    /
    / STATUS NINE - TASK HAS BEEN SWAPPED.
    / ENTRY AT API-7 WITH ATL NODE ADDRESS IN XR
    /  TEST IF I/O IS READY
    /
    S9      .INH
            PXL                     / SAVE ATL NODE ADDRESS.
            LAC     A.EV,X          / EV ADDRESS IS IN TSQ NODE.
            PAX                      / TSQ NODE ADDRESS
            LAW     -1
            TAD     TS.EV,X
            DAC     X10
            LAC*    X10
            SNA                      / I/O READY
            JMP     M5               / NO - CONTINUE ATL SCAN
    /
    / I/O IS READY, TEST IF PARTITION IS FREE.
    /
            PLX
            LAC     A.PB,X           / ACCESS TO PARTITION BLOCK
            PAX
            LAC     P.FW,X           /
            SPA                      / PARTITION OCCUPIED?
            JMP     M5               / YES - CONTINUE ATL SCAN
            XOR     (400000)         / NO - FLAG PARTITION OCCUPIED
            DAC     P.FW,X
            CLC
            .ENB
            DAC     TSLTG           / AND START TSLOAD
            JMP     M6
            .EJECT
```

APPENDIX G.

*** 'TSLOAD' - TIME SHARE LOAD TASK

```
          .TITLE *** 'TSLOAD' - TIME SHARE LOAD TASK
    /
    / TSLOAD IS TRIGGERED WHEN A TASK IN TSQ IS READY TO
    / RESUME EXECUTION AND THE PARTITION IS FREE OR THE TASK
    / OCCUPYING IT HAS BEEN SWAPPED TO DISK.
    /  TSLOAD SCANS TSQ FOR A TASK READY TO RESUME EXECUTION. IF A TASK
    / IS READY IT IS LOADED INTO ITS PARTITION.
    /
    / THIS TASK IS ALWAYS CORE RESIDENT AND RESIDES IN  PARTITION
    / THAT IS NOT AVAILABLE FOR OTHER TASKS.
    / IT CONTAINS A PARTITION BLOCK THAT IS ONLY USED AS A REGISTER SAVE
    / AREA.
    / IT IS NOT A PART OF THE PARTITION BLOCK DESCIPTION LIST NOR IS THE
    / FLAGS WORD EVER CHECKED OR ALTERED.
    / THE TASK IS ALWAYS IN STATUS FOUR I.E. READY TO BE RUN.
    / WHEN THE TASK IS IDLE IT WILL PERFORM WAITFOR ON THE TRIGGER EVENT
    / VARIABLE.
    / THE TASK IS NEVER REQUESTED, NOR DOES IT EXIT, THEREFOR THERE
    / IS NO STL ENTRY FOR IT.
    /
    /
    WFTSL    CAL     WFTSLT          / WAIT FOR TRIGGER
             DZM     TSLTG           / CLEAR THE TRIGGER EV
             JMS     TSQSCN          / SCAN TSQ FOR TASK READY TO
                                     / RESUME EXECUTION.
             HLT                     / NO TASK READY - STOP
             PXL                     / TASK READY - SAVE TSQ NODE ADDRESS
             PXA
             DAC     SAVTSN
             LAC     (DSKGET)        / SET CT ADDRESS IN CPB
             AAC     +3
             DAC     R3
             LAC     TS.DP,X         / CT IN DISK PARAMETER NODE
             DAC     R4              / SAVE DPN ADDRESS TEMPORARILY
             AAC     +2
             DAC*    R3
             LAC     TS.AP,X         / ACCESS TO PARTITION BLOCK
             PAX
             LAC     A.PB,X
             PAX
    /
    /        RESTORE PARTITION BLOCK
    /
             PLA                     / ACCESS TO SAVED DATA
             AAC     +5              / IN TSQ NODE
             DAC     X10
             LAC*    X10             / COUNT OF PENDING TRANSFERS
             DAC     P.TP,X
             LAC*    X10             / VIRTUAL SIZE
             DAC     P.VS,X
             LAC*    X10             / BUFFER POINTER
             DAC     P,BP,X
             LAC*    X10             /  JEA REGISTER
```

```
*** 'TSLOAD' - TIME SHARE LOAD TASK

            DAC     P,JE,X
            LAC     R4                  / TASK SIZE
            AAC     +6
            DAC     R4
            LAC*    R4
            DAC     P,TS,X
    /
            DZM     DKEV                / CLEAR DISK GET EV
            CAL     DSKGET              / GET TASK FROM DISK
            CAL     WFDKEV
    /
    /       RETURN DP NODE TO DPQ.
    /
            LAC     SAVTSN              / RESTORE TSQ NODE ADDRESS
            PAL
            LAC     (DPQ)              / LISTHEAD ADDRESS
            DAC      R1
            PLX
            LAC     TS,DP,X             / NODE ADDRESS
            DAC     R2
            JMS     NADD                / ADD NODE TO DEQUE (R2,R6,XR&AC ALTERED).
    /
    /       RETURN TSQ NODE TO POOL
    /
            PLX
            LAC     TS,AP,X
            PAL                         / SAVE ATL NODE ADDRESS IN LR
            LAC     1,X                 / ADDRESS OF  PRECEEDING
            DAC     R1                  / NODE
            LAC     TS,EV,X             / RESTORE EV ADDRESS
            PLX
            DAC     A,EV,X
            JMS     PICK                / PICK THE NODE (R2,R6,XR&AC ALTERED).
            HLT                         / SHOULDN'T HAPPEN - STOP.
            DAC     R2                  / SAVE ADDRESS FOR NADD
            LAC     (POOL)              / IST HEAD ADDRESS
            DAC     R1
            JMS     NADD                 / ADD NODE TO POOL (R2,R6,XR&AC ALTERED).
            PLX
            LAC     DKEV
            SMA                         / DISK READ OK?
            JMP     TSL.2               / YES
            ISZ     SE.AD               / NO - INCREMENT COUNT OF TASK LOAD ABORTS
            LAC     (RETX)              / DUE TO DISK READ ERRORS.
            DAC     A,RA,X
    TSL.2   LAC     (4)                 / SET STATUS FOUR
            DAC     A,TS,X
            LAC     A,EV,X              /SET TSLOADING EV = +1
            AAC     -1
            DAC     X10
            CLA!IAC
            DAC*    X10
            JMP     WFTSL                /PROCESS STATUS FOUR
```

```
*** 'TSLOAD' - TIME SHARE LOAD TASK

    /
    /
    /
    /       VARIABLES
    /
    DKEV    0                       / TSLOAD DISK EV
    SAVTSN  0                       / TSQ NODE ADDRESS
    /
    /       CAL PARAMETER BLOCKS
    /
    DSKGET  3000                    / DISK GET CPB - FUNCTION CODE
            DKEV
            1                       / LUN
            0                       / CT ADDRESS
    WFDKEV  20                      / WAITFOR DKEV
            DKEV
    WFTSLT  20                      /
            TSLTG                   / EVENT VARIABLE ADDRESS
    /
    /       ACTIVE TASK LIST NODE
    /
    TSLOAD  SFG                     / FORWARD LINKAGE
            IORD                    / BACKWARD LINKAGE
            ,SIXBT "TSL"            / TASK NAME
            ,SIXBT "OAD"            / TASK NAME
            10                      / TASK PRIORITY
            TSLIC-P.IC
            0                       / STL NODE ADDRESS - NONE
            4                       / TASK STATUS
            WFTSL                   / RESUMPTION ADDRESS
            0                       / EV ADDRESS
    /
    /       "PARTITION BLOCK" - REGISTER SAVE AREA
    /
    TSLIC   SHPB
*G
*G          0
*G          DBA
*G          JMS     SAVE
            ,REPT   PBIB
            0
*R
*R
*R
*R
*R
*R
*R
*R
*R
*R
*R
*R
*R
```

APPENDIX H.

```
 .TITLE TS..
/
/ THIS TASK IS ALWAYS CORE RESIDENT AND RESIDES IN A PARTITION
/ THAT IS NOT AVAILABLE FOR OTHER TASKS.
/ IT CONTAINS A PARTITION BLOCK THAT IS ONLY USED AS A REGISTER SAVE
/ AREA.
/ IT IS NOT A PART OF THE PARTITION BLOCK DESCRIPTION LIST NOR IS THE
/ FLAGS WORD EVER CHECKED OR ALTERED.
/ THE TASK IS ALWAYS IN STATUS FOUR I.E. READY TO BE RUN.
/ WHEN THE TASK IS IDLE IT WILL PERFORM WAIT.
/ THE TASK IS NEVER REQUESTED, NOR DOES IT EXIT, THEREFOR THERE-
/ IS NO STL ENTRY FOR IT.
/
/ THIS TASK IS A 'SWITCHBOARD' BETWEEN THE TELETYPES AND THE SYSTEM
/ UNDER TSX.
/ IT HAS A PHYSICAL DEVICE LIST NODE WITH A REQUEST QUEUE, WHICH IS
/ SCANNED EACH TIME TS.. TRIGGER EVENT VARIABLE IS NON-ZERO.
/ THE NODES IN THE REQUEST QUEUE ARE PASSED TO USER QUEUES WITHIN THE
/ TASK( TS.. EXPECTS TO FIND THE USER IN BITS 5 - 7 OF THE PRIORITY
/ WORD.).
/ THEESE QUEUES ARE SCANNED EACH TIME THE SCAN OF ACTIVE TASK LIST
/ REACHES TS.. . FROM THE USER QUEUES NODES ARE PICKED AND THE
/ OPERATIONS ARE INITIATED BY TS.. .
/ WHEN TS.. DISCOVERS A READ REQUEST OR THAT TOO MANY WRITE REQUESTS
/ HAVE BEEN QUEUED FOR ONE USER IT TRIES TO SHARE THAT TASK I.E. PUT
/ THE TASK IN STATUS SEVEN SO THE TASK CAN BE SWAPPED IF NESSESSARY.
/
/ IF THE SHARE OPERATION FAILS TS.. PERFORMS WAIT AND MAKES A  NEW
/ ATTEMPT AT THE NEXT SIGNIFICANT EVENT.
/
/
/ LEGAL I/O FUNCTIONS:
/                          READ
/                          WRITE
/                          HINF
/
/ ATTACH AND DETACH IS IGNORED BECAUSE TS.. MANAGES I/O FOR
/ SEVERAL TTY'S.
/ ABORT IS NOT IMPLEMENTED.
/
X14=14      /AUTOINCREMENT REGISTER 14
X15=15      /    " - "        "-"    15
R1=101      /REENTRANT REGISTER 1
R2=102      /  " - "     " -"    2
NADD=107    /ADD NODE ROUTINE ENTRY POINT
PENP=115    /PICK NODE FROM POOL ROUTINE ENTRY POINT
PICK=120    /PICK NODE ROUTINE ENTRY POINT
SNAM=123    /NAME SCAN ROUTINE ENTRY POINT
SPRI=126    /
SAVE=131    /SAVE REGISTER ROUTINE ENTRY POINT
POOL=240    /LIST HEAD FOR POOL OF EMPTY NODES
ATKL=244    / "     "     "  ACTIVE TASK LIST
PDVL=252    / "     "     "  PHYSICAL DEVICE LIST
IOCD=345    /DECLARE I/O COMPLETED ROUTINE ENTRY POINT
P.IC=14
PBIB=24
PBFP=6
        .DEC
LENG=34     /BUFFER LENGTH
NUMBUF=12   /NUMBER OF BUFFERS
/
/*****************************************************************************
/
```

```
/ TO GET THE APPROPRIATE NUMBER OF USERS DEFINE NUS EQUAL TO
/ THAT NUMBER.
/
        .IFUND NUS
NUS=8       /NUMBER OF USERS
        .ENDC
/
/*************************************************************************
/
        .OCT
.INH=705522 /INHIBIT INTERRUPTS
.ENB=705521 /ENABLE     " - "
IDX=ISZ
        .ABSP NLD
        .LOC      105     /SET INITIATION CODE
        .DSA      TSINIT  /ENTRY POINT IN R5
        .LOC 14000
/
TS..    CAL       (05)     /WAIT
        LAC       TSLR     /SETUP LR & XR TO SCAN USER  QUEUES.
        PAL
        CLX
TSA.1   PXA
        DAC       TSUS     /SAVE USER
        LAC       TSOLOP,X / ANY OLD OPERATION TO SERVICE?
        SNA
        JMP       TSINOP   /NO - ANY NEW ONES TO INITIATE?
        DAC       TSNAD    /YES - SAVE NODE ADDRESS
        AAC       +5       /GET OPERATION CODE
        DAC       TSTEMP
        LAC*      TSTEMP
        AND       (777)
        DAC       TSOPCD
/
        LAC       TSEV2,X  /IS TASK BEING LOADED INTO CORE?
        SZA
        JMP       TSA.3    /YES - LOADING READY?
        LAC       TSEV1,X  /NO - OPERATION DONE?
        SNA
        JMP       TSA.EX   /NO - TRY NEXT USER
        SPA                /OPERATION OK?
        HLT                /NO
        LAC       TSOPCD
        SAD       (026)    /OPERATION WAS READ?
        SKP                /YES
        SAD       (360)    /NO - WAS IT SHARE?
        SKP                /YES
        JMP       TSA.2    /NO
/
/ LAST OPERATION READ OR SHARE, NO MORE NODES TO SERVICE, START
/ TSLOADING IF THE TASK IS SWAPPED.
/
        LAC       TSNAD    /GET TASK STATUS TO DETERMINE WHETHER THE TASK
        AAC       +2       /HAS BEEN SWAPPED OR NOT
        DAC       TSTEMP
        LAC*      TSTEMP
        AAC       +2       /NAME WORDS ADDRESS
        DAC*      (R2)
        LAC       (ATKL)
        DAC*      (R1)
        JMS*      (SNAM)   /SCAN ATL FOR THIS TASK.
        HLT                /SHOULD NOT HAPPEN.
        DAC       TSTEMP   /SAVE ATL NODE ADDRESS TEMPORARILY
        JMS       TSRXR    /RESTORE XR & LR.
```

```
        LAC     TSTEMP    /
        AAC     +7        /STATUS WORD ADDRESS
        DAC     TSTEMP
        AAC     +2
        DAC     TSSHRD,X  /SAVE EV WORD ADDRESS
        LAC*    TSTEMP
        SMA               /TASK LOADING FLAG IS SET IF STATUS 8 OR NINE
        JMP     TSA.3A    /THE TASK HAS NOT BEEN SWAPPED.
        LAW     -1        /TASK HAS BEEN SWAPPED, INITIATE TSLOADING.
        DAC     TSEV2,X
        JMP     TSA.EX
/
/ THE OPERATION IS WRITE( ONLY WRITE,READ&SHARE IS QUEUED), THE
/ BUFFER IS RETURNED AND THE BUFFER COUNT IS DECREMENTED.
/
TSA.2   LAC     TSNAD     /GET BUFFER ADDRESS
        AAC     +4
        DAC     TSTEMP
        LAC*    TSTEMP
        DAC*    (R1)
        JMS     RETBF     /RETURN BUFFER
        JMS     TSRXR     /RESTORE XR & LR
        LAC     NBF,X     /DECREMENT BUFFER COUNT
        AAC     -1
        DAC     NBF,X
        JMP     TSA.5     /RETURN NODE TO POOL AND INITIATE
                          /NEXT OPERATION.
/
/       TSLOADING
/
TSA.3   SAD     (-1)      /TSLOADING DONE?
        JMP     TSA.EX    /NO
TSA.3A  LAC     TSOPCD    /YES TASK IN CORE . IF OPERATION
                          /IS READ TRANSFER TS.. BUF TO USER BUF.
        SAD     (360)     /READ?
        JMP     TSA.4     /NO
/
        LAC     TSNAD     /GET TS.. BUFFER ADDRESS
        AAC     +4
        DAC     TSTEMP
        LAC*    TSTEMP
        DAC*    (R1)      /SAVE BUFFER ADDRESS FOR RETBF
        DAC*    (X14)
        LAC     TSNAD     /GET USER BUFFER ADDRESS
        AAC     +10
        DAC     TSTEMP
        LAC*    TSTEMP
        AAC     -1
        DAC*    (X15)
        IDX     TSTEMP    /SET BUFFER SIZE FOR TSXFER
        LAC*    TSTEMP
        DAC     SIZE
        JMS     TSXFER    /TRANSFER BUFFER
        JMS     RETBF     /RETURN BUFFER(R1,LR,XR&AC ALTERED)
        JMS     TSRXR     /RESTORE XR & LR
        LAC     NBF,X     /DECREMENT BUFFER COUNT
        AAC     -1
        DAC     NBF,X
/
TSA.4   LAC     TSSHRD,X  /RESTORE EV ADDRESS IN ATL NODE
        DAC     TSTEMP
        LAC     TSNAD     /SET USER EV
        AAC     +6
        DAC     TSNAD
```

```
        LAC*    TSNAD
        AAC     -1
        DAC*    (X14)
        DAC*    TSTEMP
        LAC     TSEV1,X
        DAC*    X14
        DZM     TSEV2,X   /TS NOT LOADING
        DZM     TSSHRD,X   /TASK NOT SHARED
        LAC     TSDQ       /ANY NODES IN REQ QUEUE?
        SAD     (TSDQ)
        SKP                /NO
        IDX     TSTGEV     /YES - SET TRIGGER TO ENSURE REQ. QUEUE SCAN
TSA.5   LAC     TSUS      /RETURN NODE TO POOL
        TAD     TSUS
        TAD     (TTQ)
        DAC*    (R1)
        JMS*    (PICK)     /PICK NODE FROM USER QUEUE
        HLT                /SHOULD NOT HAPPEN
        DAC*    (R2)
        LAC     (POOL)
        DAC*    (R1)
        JMS*    (NADD)  /ADD NODE TO POOL(R2,R6,XR&AC ALTERED)
        JMS     TSRXR    /RESTORE LR & XR
        DZM     TSOLOP,X / CLEAR OLD OP EV
/
TSINOP  LAC     TSUS      /MORE OPERATIONS TO INITIATE?
        TAD     TSUS
        TAD     (TTQ)
        DAC     TSTEMP
        LAC*    TSTEMP
        SAD     TSTEMP
        JMP     TSA.EX    /NO - QUEUE EMPTY, TRY NEXT USER.
        DAC     TSNAD     /YES - SAVE NODE ADDRESS
        TAD     TSXADJ
        PAX
        LAC     5,X       /GET OP CODE
        AND     (777)
        SAD     (360)     /SHARE?
        SKP               /YES
        JMP     TSA.6     /NO
        JMS     TSRXR     /RESTORE XR
        CLA IAC           /SHARE IS THE LAST NODE, SHARE IS DONE.
        DAC     TSEV1,X   /INDICATE THIS BY SETTING TSEV1
        LAC     TSNAD     /SET OLD OPERATION EV
        DAC     TSOLOP,X
        JMP     TSA.1     /START TSLOADING
TSA.6   DAC     TSOPCD
        ALSS    6
        DAC     TSCAL     /SET OP CODE IN CAL CPB
        LAC     (TSEV1)   /SET EV ADDRESS IN CPB
        TAD     TSUS
        DAC     TSCAL+1
        DZM*    TSCAL+1
        LAC     TSLUN     /GET LUN
        TAD     TSUS
        DAC     TSCAL+2
        LAC     7,X       /SET DATA MODE
        DAC     TSCAL+3
        LAC     4,X        /SET BUFFER ADDRESS
        AAC     +1
        DAC     TSCAL+4
        CAL     TSCAL
        JMS     TSRXR      /RESTORE XR
        LAC     TSNAD      /INDICATE THAT AN OPERATION IS BEING DONE
```

```
        DAC     TSOLOP,X
/
TSA.EX  AXS     1           /ARE ALL USERS SERVICED?
        JMP     TSA.1       /NO
        LAC     TSTGEV      /YES - ARE THERE ANY NEW NODES?
        SNA
        JMP     TS..        /NO. WAIT FOR NEXT SIGN. EVENT
        DZM     TSTGEV      /YES. CLEAR TRIGGER
        LAC     (TSDQ)      /REQUEST QUEUE ADDRESS
        DAC     TSNP
TSLOOP  LAC*    TSNP        /ANY NODE?
        SAD     (TSDQ)
        JMP     TS..+1      /NO - INITIATE OPERATIONS
        DAC     TSNAD       /YES - SAVE ADDRESS
        AAC     +5          /GET USER FROM PRIORITY WORD IN
        DAC     TSTEMP      /THE REQUEST NODE
        LAC*    TSTEMP
        LRS     10
        AND     (7)
        DAC     TSUS        /SAVE USER
        PAX

        LAC     TSSHRD,X    /IS THE TASK SHARED?
        SNA
        JMP     TSB.2       /NO
        LAC     TSNAD       /YES. TRY NEXT NODE.
        DAC     TSNP
        JMP     TSLOOP
TSB.2   LAC     TSNP          /PICK NODE FROM REQUEST QUEUE
        DAC*    (R1)
        JMS*    (PICK)        /PICK THE NODE
        HLT                   /SHOULD NOT HAPPEN
        DAC     TSNAD         /SAVE NODE ADDRESS
        DAC*    (R2)
        TAD     TSXADJ
        PAX
        LAC     5,X           /GET OP.-CODE
        AND     (777)
        DAC     TSOPCD
        SAD     (026)       /READ?
        SKP                 /YES
        SAD     (027)       /NO - WRITE?
        JMP     TSGBF       /YES
/
/ ITS NOT ALLOWED TO ATTACH TS., BECAUSE TS., MANAGES I/O FOR SEVERAL
/ USERS. ATTACH AND DETACH IS IGNORED.
/
        LAC     6,X         /SET USER EV NON-0
        AAC     -1
        DAC*    (X14)
        LAC     HINFEV
        DAC*    X14
        JMS*    (IOCD)      /DECREMENT TRANSFERS PENDING COUNT(R5,
        LAC     (POOL)      /XR&AC ALTERED),RETURN NODE TO POOL.
        DAC*    (R1)
        JMS*    (NADD)
        JMP     TSLOOP      /NEXT REQUEST IF ANY
/
/ READ OR WRITE.
/
TSGBF   JMS     GETBF       /GET A BUFFER (XR,AC&X15 ALTERED)
        JMP     TSNOBF      /NO BUFFER AVAILABLE - WAIT FOR NEXT
                            /SIGNIFICANT EVENT.A BUFFER MAY BE
                            /AVAILABLE THEN.
```

```
        JMS*    (IOCD)          /DECREMENT TRANSFERS PENDING  COUNT
        LAC     TSNAD           /BUFFER ADDRESS IN X15 AND AC.
        AAC     +4              /SET BUFFER ADDRESS IN REQUEST NODE
        DAC     TSTEMP
        LAC*    (X15)
        DAC*    TSTEMP
        JMS     TSQND           /QUEUE A REQ NODE(R2,R6,XR,AC&TSTEMP ALTERED)
        JMS     TSRXR           /RESTORE XR
        IDX     NBF,X           /INCREMENT BUFFER COUNT
        LAC     TSOPCD
        SAD     (026)           /READ?
        JMP     TSSH            /YES - SHARE
        LAC     TSNAD           /NO - WRITE
        AAC     +10
        DAC     TSTEMP
        LAC*    TSTEMP          /GET USER BUFFER
        AAC     -1
        DAC*    (X14)
        LAC     (LENG)          /SET BUFFER SIZE FOR TSXFER
        DAC     SIZE
        JMS     TSXFER          /TRANSFER BUFFER
        LAC     NBF,X           /TOO MANY BUFFERS?
        AAC     -4
        SMA
        JMP     TSQSH           /YES - SHARE
        LAC     TSNAD           /NO - SET USER EV NON-0 AND TRY NEXT
        AAC     +6              /NODE.
        DAC     TSTEMP
        LAC*    TSTEMP
        AAC     -1
        DAC*    (X14)         /SET USER EV NON-0
        CLA!IAC
        DAC*    X14
        JMP     TSLOOP
/
/ TOO MANY WRITE REQUESTS. QUEUE A SHARE NODE.
/
TSQSH   JMS*    (PENP)          /PICK A NODE FROM POOL
        JMP     TSNND           /POOL EMPTY - WAIT
        DAC*    (R2)            /SAVE NODE ADDRESS FOR TSQND
        TAD     TSXADJ
        PAX
        LAC     TSNAD           /SAVE STL NODE ADDRESS IN SHARE NODE
        AAC     +2
        DAC     TSTEMP
        LAC*    TSTEMP
        DAC     2,X
        LAC     TSNAD           /SAVE EV ADDRESS IN SHARE NODE
        AAC     +6
        DAC     TSTEMP
        LAC*    TSTEMP          /
        DAC     6,X
        LAC     (360)           /SET OP.-CODE
        DAC     5,X
        JMS     TSQND           /QUEUE A REQ NODE
        JMS     TSRXR           /RESTORE XR
/
/ READ OR TOO MANY WRITE REQUESTS. SET UP SHARE CPB AND TRY TO
/ SHARE THE TASK.
/
TSSH    LAC     (TSSHRD)        /SET EV ADDRESS IN CPB
        TAD     TSUS
        DAC     TSSHR+1
        DZM     TSSHRD,X        /CLEAR EVENT VARIABLE
```

```
        LAC     TSNAD
        AAC     +2
        DAC     TSTEMP
        LAC*    TSTEMP          /STL NODE ADDRESS
        AAC     +2
        DAC     TSTEMP          /SET TASK NAME IN CPB
        LAC*    TSTEMP
        DAC     TSSHR+2
        ISZ     TSTEMP
        LAC*    TSTEMP
        DAC     TSSHR+3
        LAC     (TSEV2)         /SET TSLOADING EV IN CPB
        TAD     TSUS
        DAC     TSSHR+4
        DZM     TSEV2,X         /CLEAR EVENT VARIABLE
SHARE   CAL     TSSHR
        JMS     TSRXR           /RESTORE XR
        LAC     TSSHRD,X        /SHARE OK?
        SMA
        JMP     TSLOOP          /YES. NO MORE NODES ARE ALLOWED TO BE QUEUED
                                /FOR THIS USER.
        CAL     (05)            /NO TRY AFTER NEXT SIGNIFICANT  EVENT
        JMP     SHARE
/
TSNOBF  LAC     TSNP            /NO BUFFER AVAILABLE, WAIT.
        DAC*    (R1)            /A BUFFER MAY BE FREE AFTER NEXT SIGN. EVENT.
        JMS*    (NADD)
        IDX     TSTGEV          /SET TRIGGER TO ENSURE THAT REQUEST QUEUE
                                /SCAN WILL BE CONTINUED.
        JMP     TS..
/
TSNND   CAL     (05)            /POOL EMPTY, WAIT FOR A NODE.
        JMP     TSQSH
/
        .EJECT
/
/ TSRXR -- RESTORE LR&XR SUBROUTINE
/
TSRXR   0
        LAC     TSUS
        PAX
        LAC     TSLR
        PAL
        JMP*    TSRXR
/
/
/ TSXFER -- TRANSFER BUFFER SUBROUTINE
/
/ ENTRY CONDITIONS:  X14 CONTAINS ADDRESS-1 FOR THE BUFFER
/                        THAT IS GOING TO BE TRANSFERED.
/                    X15 THE SAME FOR THE BUFFER TO WHICH THE
/                        TRANSFER IS DONE.
/ REGISTERS ALTERED: X14,X15,AC
/
TSXFER  0
        LAC     SIZE            /2'S COMPLEMENT OF THE BUFFER SIZE
        TCA
        DAC     SIZE
        LAC*    X14             /TRANSFER BUFFER
        DAC*    X15
        ISZ     SIZE            /END OF BUFFER?
        JMP     .-3             /NO - CONTINUE TRANSFER
        JMP*    TSXFER          /YES - RETURN AT JMS+1
/
```

```
SIZE    LENG
/
/  SUBROUTINE GETBF - RESERVES AN I/O-BUFFER FOR A TASK.
/
/ CALLING SEQUENCE: JMS GETBF
/
/ RETURN: JMS+1 IF NO BUFFER AVAILABLE
/         JMS+2 IF A BUFFER IS AVAILABLE WITH BUFFER ADDRESS
/         IN X15 AND AC.
/
/ REGISTERS ALTERED: X15 , XR , AC & LR
/
GETBF   0
        LAC     (NUMBUF)    /NUMBER OF BUFFERS
        PAL
        CLX
GTB.1   .INH
        LAC     BFADSS,X    /THIS BUFFER AVAILABLE?
        DAC     TSTEMP
        LAC*    TSTEMP
        SNA
        JMP     GTB.2       /NO
        .ENB                /YES
        DZM*    TSTEMP      /MARK IT UNAVAILABLE
        LAC     BFADSS,X    /AND RETURN ITS ADDRESS IN X15
        DAC*    (X15)
        IDX     GETBF
        JMP*    GETBF       /RETURN AT JMS+2
/
GTB.2   ,ENB
        AXS     1           /INDEX TO NEXT
        JMP     GTB.1       /TRY THIS ONE
        JMP*    GETBF       /NO BUFFER AVAILABLE - RETURN
                            /AT JMS+1
/
/ SUBROUTINE RETBF - RETURNS A BUFFER (MAKES IT AVAILABLE).
/
/ CALLING SEQUENCE:  R1  BUFFER ADDRESS
/                    JMS RETBF
/
/ RETURN:   JMS+1 UNCONDITIONALLY
/
/ REGISTERS ALTERED: XR , AC & LR
/
RETBF   0
        LAC     (NUMBUF)    /NUMBER OF BUFFERS
        PAL
        CLX
        LAC*    (R1)
RTB.1   SAD     BFADSS,X    /IS THIS THE BUFFER?
        JMP     RTB.2       /YES
        AXS     1           /NO
        JMP     RTB.1
        HLT,SKP             /SHOULD NEVER HAPPEN
RTB.2   LAC     BFADSS,X    /MAKE BUFFER AVAILABLE
        DAC     TSTEMP
        DAC*    TSTEMP
        JMP*    RETBF
/
/ SUBROUTINE TSQND - QUEUE A NODE
/
/ CALLING SEQUENCE:   TSUS -- USER
/                     JMS     TSQND
/
```

```
/ REGISTERS ALTERED: R2, R6, XR, AC & TSTEMP
/
/ NADD IS USED TO ENSURE THAT WRITE&READ OPERATIONS APPEAR IN
/THE RIGHT SEQUENCE. IF SPRI IS USED HIGH-PRIORITY TASKS
/MAY GET THEIR NODES IN FRONT OF LOW-PRIORITY TASK'S.
/
TSQND   0
        LAC     (TTQ)
        TAD     TSUS
        TAD     TSUS        /INSERT NODE AT THE END OF THE QUEUE
        DAC     TSTEMP
        IDX     TSTEMP
        LAC*    TSTEMP
        DAC*    (R1)
        JMS*    (NADD)
        JMP*    TSQND
        .EJECT
/
/ DATA REGISTERS.
/
/ VARIABLES
/
        .DEC
TSLUN   27
        .OCT
HINFEV  300701 /
TSLR    NUS    / LR SAVE REGISTER
TSNAD   0      / NODE ADDRESS
TSNP    0      / NODE POINTER
TSOPCD  0      / OPERATION CODE
TSTGEV  0      / TS.. TRIGGER EVENT VARIABLE.
TSTEMP  0      / TEMPORARY REGISTER
TSUS    0      / USER
TSXADJ  .&700006777777+1  /INDEX REGISTER ADJ FACTOR - TWO'S
                          /COMPLEMENT OD THE PAGE BITS OF THE PAGE
                          /IN WHICH THE CODE RESIDES -- CANSELS OUT
                          /THE PAGE BITS ADDED BY THE HARDWARE ON
                          /INDEX MEMORY INSTRUCTIONS.
        .REPT NUS
NBF     0              / NUMBER OF BUFFERS/USER.
/
TTQ     .              /USER OPERATION QUEUES.
        .-1
        .
        .-1
        .
        .-1
        .
        .-1
        .
        .-1
        .
        .-1
        .
        .-1
        .
        .-1
/
        .REPT NUS
TSEV1   0               / EV USED WHEN TS.. PERFORMS READ & WRITE.
/
/
/ TSLOADING EVENT VARIABLE. THE EV ADDRESS IS USED WHEN TS.... TRIES
/ TO SHARE TASK. THE ADDRESS IS STORED BY THE SHARE DIRECTIVE IN THE ATL
```

```
/ NODE. TS,... SETS THE EVENT VARIABLE NON-0 WHEN ALL OPERATIONS ARE
/ COMPLETED. THE TASK IS THEN READY TO BE BROUGHT IN BY TSLOAD.
/
        .REPT NUS
TSEV2   0               /TSLOADING EVENT VARIABLE
/
/
/ STATUS REGISTER.
/ TSOLOP CONTAINS THE I/O-NODE ADDRESS IF TS,... PERFORMS AN OPERATION
/ FOR THIS USER ELSE IT IS 0.
/
        .REPT NUS
TSOLOP  0               /TS.. STATUS REGISTER
/
/ SHARE EVENT VARIABLE. USED BY TS.. WHEN IT SHARES A TASK.
/ IT IS TESTED BY TS.. IF NON-0. IF SO THE TASK HAS BEEN SHARED AND
/ NO MORE OPERATION MAY BE QUEUED FOR THIS USER.
/
                        /0 IF THIS USER QUEUE IS IDLE
                        /NON-0 IF NOT IDLE
        .REPT NUS
TSSHRD  0               /SHARED EVENT VARIABLE
/
/       CAL PARAMETER BLOCKS.
/
TSCAL   0       /OPERATION CODE
        0       /EV ADDRESS
        0       /LUN
        0       /I/O DATA MODE
        0       /STARTING ADDRESS
        LENG    /BUFFER SIZE
/
TSSHR   36      / SHARE OP CODE
        0       / EVENT VARIABLE ADDRESS
        0       / TASK NAME
        0       /  "    "
        0       / TSLOADING EVENT VARIABLE
/
/       BUFFER AREA.
/
/ THE BUFFER LENGTH IS 34+1 (DEC). THE FIRST WORD OF THE LINE
/ BUFFER IS USED AS AN AVAILABLE INDICATOR:
/         0   THE BUFFER IS NOT AVAILABLE
/         NON-0 THE BUFFER IS AVAILABLE
/
        .REPT NUMBUF 43
BFADSS  BFSTRT      //BUFFER ADDRESS
/
BFSTRT=.
/
        .BLOCK LENG+1*NUMBUF-LENG-1
/
TSINIT  0               /BUFFER AREA. THE CODE IS USED AT INITIALIZATION.
        LAC     (ATKL) /TS,. PDVL AND ATKL NODES ARE INSERTD IN
        DAC*    (R1)    / THEIR SYSTEM LISTS.
        LAC     (TS)
        DAC*    (R2)
        JMS*    (SPRI)
        LAC     (PDVL)
        AAC     +1
        DAC     TSTEMP
        LAC*    TSTEMP
        DAC*    (R1)
        LAC     (TS00)
```

```
        DAC*    (R2)
        JMS*    (NADD)
        LAC     (NUMBUF-1) /BUFFERS ARE SET AVAILABLE.
        PAL
        CLX
        LAC     BFADSS,X
        DAC     TSTEMP
        CLA!IAC
        DAC*    TSTEMP
        AXS     1
        JMP     .-5
        JMP*    TSINIT
/
        .BLOCK  LENG+1+TSINIT-. /REMAINDER OF 34 (DEC) WORD BUFFER
/
/
/       PDVL NODE FOR TS..
/
/       THE PDVL NODE NAME FOR TS.. HAS TO BE "TTA" WITH UNIT
/       7 UNTIL THE THE REA MCR-FUNCTION HAS BEEN CHANGED SO IT
/       CAN ACCEPT "TSA".
/
TSDD    0               /FORWARD LINKAGE
         0              /BACKWARD LINKAGE
        .SIXBT "TTA"    /DEV. NAME 1ST HALF
        0               / "     "   2ND  "
        0               / "   ATTACH FLAG
        7               / UNIT NUMBER
TSDQ    .               / REQUEST QUEUE
        .-1             /   "        "
        TSTGEV          / EVENT VARIABLE ADDRESS
        0               / UNUSED
/
/       ACTIVE TASK LIST NODE
/
TS      0               / FORWARD LINKAGE
        0               / BACKWARD LINKAGE
        .SIXBT "TSA"    / TASK NAME
        0               /  "      "
        10              / TASK PRIORITY
        TSIC-P.IC       / PARTITION BLOCK ADDRESS
        0               / STL NODE ADDRESS
        4               / TASK STATUS
        TS..            / RESUMPTION ADDRESS
        0               / EVENT VARIABLE ADDRESS
/
/       TS.. SHORT PARTITION BLOCK
/
TSIC    0
        DBA
        JMS*    (SAVE)
        .REPT PBIB
        0
        SKP
        .REPT PBFP
        0
/
        .END TS..
```