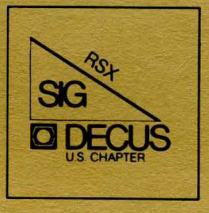
DECUS

The DeVIAS Letter



The RSX Multi-Tasker

August 1983

Printed in U.S.A.

The

DeVIAS Letter

and

RSX Multi-Tasker

This first Subscription Service issue of the combined IAS and RSX newsletters features the IAS Newsletter.

Please remember that the newsletter editors are eager to hear your comments and receive your articles. Articles may be submitted to the editors directly or sent to:

DECUS Publications
DECUS
Digital Equipment Corporation
One Iron Way, MR02-1/C11
Marlboro, MA 01752

DECUS

The DeVIAS Letter

The Newsletter of the IAS SIG

August 1983

Issue No. 15

IN THIS ISSUE

(Next Issue)
Whatever you send me!!

From the Editor

The Delaware Valley IAS Local User Group has evolved into the U.S. DECUS IAS SIG. This change in status was brought about by the diligent efforts of our Chairman, Bob Curley and the people he employs to obtain his ends.

As you may have noticed, the newsletters are not too frequent. The major problem is material. In a magnanimous effort to encourage submissions, I have reduced the standards to nil. I'll accept anything (text, pictures, wiring diagrams, whatever you send). I'll publish whatever DECUS will let me.

A new feature of the newsletter is the DeVIAS Question and Answer form (at the end of the newsletter).

If you have a problem or a solution to someone else's problem, write it down and it will appear in the next Letter.

The success of the SIG is closely tied to our activity in the symposia and the newsletter. If you want to see IAS continue to grow and improve, please participate (if only by a short note).

Contributions

The DeVIAS Letter needs contributions in order to continue as an effective medium for exchange of information regarding IAS.

Contributions may be submitted in any form you wish. Originals on 8½ x 11 paper are preferred. However, even photocopies of relevant match-book covers would be appreciated.

Send all contributions to:

Ontario Hydro
700 University Avenue
Toronto, Ontario
CANADA, M5G 1X6

Attn: John W. Drummond
Mail Stop - M4E5

Department of Radiation Therapy
University of Pennsylvania
Room 410
133 South 36th Street
Philadelphia, Pennsylvania 19104
11 June 1983

Dear DeVIAS Members,

I have been told that there is now an IAS SIG! This is to say there is a lot that has been happening behind the scenes and between the sparse issues of the DeVIAS Letter. Eight of us petitioned DECUS U.S. to become a Special Interest Group, separate from the RSX SIG. Last Winter I was invited to the RSX/IAS SIG Executive Steering Committee Woods Meeting in Marlboro, Massachusetts to present my arguments to the only SIG who would be most affected by our request. The RSX/IAS SIG Executive Steering Committee was very receptive to my presentation and voted, unanimously, to support our petition to the Executive Board. Since that time we applied and the application circulated to the SIG Chairmen for their comment. At the St. Louis Symposium I was invited to the SIG Chairmen's breakfast meeting on Thursday to present our case. Again, the reception was very supportive. Legare Coleman, the RSX/IAS SIG Chairman, was especially vocal in our support.

Ray French, who has been an outstanding friend throughout this whole process, called last Friday to say that the Executive Board had approved our petition to form a separate SIG. The petitioners are Bob Stodola of the Institute for Cancer Research, John Drummond of the Ontario Hydro, Tim Mahaney of the Naval Air Propulsion Center, Ron Fussell of the Air Force Intelligence Service, George Wells of General Electric, John Jenkinson of Mostek and I.

While this is wonderful news to many of us there are still a few problems that we must solve together. But I, personally, feel renewed after St. Louis - It was a fantastic time to be an IAS Supporter!

First, and foremost: The "Retirement of IAS" was Retired! Digital announced that they would support IAS beyond the previously announced limit of June 1985. No date for retirement was announced. While there were few specific examples, there were undercurrents of huge optimism. Rumors and suggestions of new hardware support, layered product enhancements and new functionality in IAS itself. I expect that the Las Vegas Symposium will be the stage for specifics. We have been asked for a wish list - who would have believed it two years ago when Chuck Turley announced in Miami that Digital would stop supporting IAS this month. So, send me your wishes for IAS. What have you been wanting? What have you, in desperation, implemented yourself and can share with all of us in an official release? Dream on paper and send them to me for forwarding to the DEVELOPMENT TEAM.

If there is one person to whom we should send our thanks, it is Tim Leisman, the Product Manager of IAS. He is an outstanding champion for users in distress. Those of you who have attended DECUS Symposia and DeVIAS Meetings have seen a lot of him. You who can not go to these meetings, he is a power house of energy and has a strong sense of RIGHT. It is Right that Digital continue supporting the users that have invested heavily in IAS, so he made it happen. For you who are glad that IAS is revitalized, send Tim a "Thank You Note".

It was really an IAS place, St. Louis. Thanks again to Tim Leisman, we had some new IAS buttons to help us identify each other. Carol Chorlton and John Brady, DeVIAS Members from Reading, England, and previous IAS Developers were there. They are currently working on non-DECnet inter-computer communication protocols, still in Reading. David Haigh, who used to teach IAS, was there in his present capacity as Product Manager of RSTS.

The principal effect of becoming a SIG is that the DeVIAS Letter now becomes a SIG newsletter and raised out of the oblivion of a LUG Newsletter that somehow got published by DECUS. But, we must fill it. John Drummond, now a SIG Newsletter Editor, assures me that we will take anything that is usable and interesting. Forget the RUNOFF format, just make it readable. We'll try to get it to you in a more timely fashion, you give us stuff to fill it.

I must thank all of you. You have supported IAS through DeVIAS by being a member and contributing in many ways. You have made a difference. Each of you who made up the membership in 35 states of the U.S., 5 provinces of Canada and the other 17 countries have made the past couple of years unforgettable. Thanks,

Sincerely,
Bob Curley

RFC/

UNIVERSITY of PENNSYLVANIA

SCHOOL OF MEDICINE

DEPARTMENT OF RADIATION THERAPY

Computer Facility

Robert F. Curley, Director
Robert E. Wallace

Hospital of the University of Pennsylvani
3400 Spruce Street
Philadelphia, Pennsylvania 19104
(215) 662-3083

26 May 1983

George Hamma

Executive Board

U.S. Chapter

DECUS

Dear Mr. Hamma,

I have enclosed, at your request: (a)a proposed set of operating procedures, which are slightly modified from the list that I enclosed with my letter of 5 April 1983 to Mr. Powderly. (b) A list of the proposed steering committee members and their positions. (c) A list of the activities that I propose the SIG accomplish in the first year.

Thank you for helping,

Sincerely,

Robert F. Curley

IAS SIG Steering Committee Note

From: Bob Curley

To: John Drummond
Ron Fussell
Ken Grualnik
John Jenkinson
Tim Leisman
Tim Mahaney
Bob Stodola
George Wells

6 June 1983

As some of you know the DECUS Executive Board has approved our application to form the IAS SIG. I haven't been notified officially, but Ray French called the Friday of DECUS Week with the good news.

On Thursday (of DECUS Week) I was invited to the "SIG Chairman's Breakfast" to present our case to the SIG Chairmen for their approval. The food was bad, but our reception was cordial. They said nice things about DeVIAS and our track record. They recommended that the Executive board approve our application. The discenting votes were on the principle that there are already too many SIG's and there should be no more until the OD Task Force settles the organizational structure of DECUS. Fortunately, that opinion is held by a minority of the SIG Chairmen.

Following that meeting, George Hamma asked for a few more details. John Jenkinson and I got them together in time for presentation to the Board on Friday. The documents are enclosed. As you will notice, there are some minor changes to the Operating Principles. Also, there are actual assignments next to some of the Steering Committee members. Ther rest of you will be approached shortly, but I put titles in line with the Operating Principles.

I'd like to hear from anyone who feels interested in some of the things that need to get done this year. Especially if you feel like doing it or getting it done.

I wish you a happy summer.

Activities proposed for the first year:

1. Regular Newsletters. There should be ten issues of the "DeVIAS Letter" in the first year, one issue per month, except the months containing the U.S. Chapter Symposia. The emphasis of the newsletter should be timeliness of the contents as well as the number of pages per issue.

2. U.S. Chapter Symposia Sessions. We will continue to provide IAS related sessions and sponsor additional activities as desirable. To this end, we plan to cooperate and support the on-going activities of the RSX SIG Symposia Coordinator.

3. Library. We will continue to maintain and enhance an IAS Program Library, programs known to run under IAS. Again, we plan to cooperate with the RSX SIG and participate within their "tape copy" organization.

4. Telephone Advice. We plan to enhance the present group of IAS "advice givers" to answer user-level question and suggest directions for SIG members with questions.

5. Black Book. We wish to organize the present site-profile book so that it is more available to the members.

6. Keep in Touch. We will organize and maintain a mechanism to continue serving the members of the Delaware Valley IAS LUG who are not eligible to be members of the U.S. Chapter by reason of geography.

Proposed Steering Committee:

Chairman	Robert F. Curley Department of Radiation Therapy University of Pennsylvania Philadelphia, PA
Newsletter Editor	John W. Drummond Ontario Hydro Toronto, Ontario
Symposia Coordinator (acting)	John Jenkinson Mostek Corporation Carrollton, Texas
Librarian	Robert K. Stodola Institute for Cancer Research Fox Chase Cancer Center Philadelphia, PA
Membership	George R. Wells General Electric Company Louisville, KY
At-large	Ronald B. Fussell Headquarters Air Force Intelligence Service Department of the Air Force Bolling AFB, DC
At-large	Thomas Mahaney Naval Air Propulsion Center Department of the Navy Trenton, NJ
At-large	Kenneth I. Guralnik EG&G Energy Measurements Group Las Vegas, NV

OPERATING PROCEDURES

Article I
Name

1.0 The name of the organization is the IAS Special Interest Group (IAS SIG).

Article II
Purpose

2.0 The IAS SIG is established under the bylaws of the DECUS/U.S. Chapter to:

1. Provide a forum for users of the IAS operating system to exchange ideas, programs, and any other items of common interest.

2. Provide feedback to Digital Equipment Corporation (DEC) on all matters concerning the IAS operating system, related software products, services, policies, and all DEC manufactured computers, peripheral equipment, and other products and services.

Article III
Membership

3.0 Membership requirements:

1. Any DECUS member using or interested in the IAS operating system or its related products, equipment or services is qualified to be a member.

2. Any person qualified to be a member will be accepted as a member upon submitting a request to the DECUS U.S. Chapter.

3.1 Rights of members:

1. Members shall be eligible to participate in SIG activities and be members of the Steering Committee.

2. Ten or more members of the IAS SIG may, by written petition, bring a motion before a meeting of the SIG Steering Committee.

Article IV
Steering Committee

4.0 General

1. The IAS SIG shall be administered by the Steering Committee.

2. The Steering Committee shall consist of four officers and up to four at-large members and up to four past officer members.

3. There will be a non-voting, ex officio member of the IAS SIG Steering Committee appointed by the RSX SIG to be the RSX liason with the IAS SIG.

4. Any member of the IAS SIG may be on the Steering Committee and the Steering Committee shall be composed solely of members.

5. The Chairman may act independently on all matters, and shall inform and consult with the Steering Committee as (s)he sees fit. A majority vote of the remaining members shall be required to override decisions of the chairman.

4.1 Steering Committee Officers

1. The Steering Committee Officers shall serve for two years, and be elected by the steering committee.

2. The officers are the Chairman, the Newsletter Editor, the Symposia Coordinator, and the Program Librarian.

4.2 At-large Members

1. The chairman may appoint up to two at-large Members of the Steering Committee.

2. Up to two at large members may be appointed by a majority vote of the steering committee.

4.3 Past Officer Members

To assist in the transfer of responsibilities to new officers, past officers, when eligible, will remain as members of the Steering Committee for one year.

4.4 Duties of the Chairman

1. The Chairman is the chief executive officer of the SIG, and shall chair all steering committee meetings and be responsible for directing all activities of the SIG between meetings of the steering committee. The Chairman is subject to the review of the Steering Committee, or recall by vote of no confidence of the members.

2. The Chairman shall appoint one of the steering committee members to be the liason with the RSX SIG.

4.5 Duties of the Newsletter Editor

1. The Newletter Editor shall edit and publish the SIG newsletter.

2. In the event that the position of Chairman becomes vacant, the Newletter Editor shall temporarily assume all duties of the Chairman except that of Steering Committee appointments until a permanent Chairman is elected by remaining members of the steering committee.

4.6 Duties of the Symposia coordinator

1. The Symposia coordinator is responsible for the planning and scheduling of IAS sessions at DECUS Symposia.

2. In the event that both the position of Chairman and Newsletter Editor become vacant, the Symposia coordinator shall temporarily assume all duties of the Chairman except that of Steering Committee appointments until a permanent Chairman is elected.

4.7 Duties of the Program librarian

1. The program librarian shall be responsible for the collection and distributions of IAS specific software via the DECUS tape copy program.

2. In the event that the positions of Chairman, Newsletter Editor, and the Symposia coordinator become vacant, the Program librarian shall temporarily assume all duties of the Chairman except that of Steering Committee appointments until a permanent Chairman is found.

4.8 Vacancy in Office

1. Should the Chairman vacate his(her) office by resignation, disability, or ineligibility or impeachment, a new Chairman shall be elected by a majority vote of the remaining members of the steering committee.

2. Should any other officer vacate his(her) office by resignation, disability, or ineligibility or impeachment, the Chairman shall appoint a replacement.

Article V
Elections

5.0 Election of officers

Officers shall be elected by the steering committee at the first steering committee in every even numbered year, but not less than two years from the recognition of the SIG by DECUS.

5.1 Appointment of at-large members

1. The two at-large members appointed by the chairman are intended to assist the chairman in discharging his or her duties. They will serve until discharged by the current chairman.

2. The two at-large members appointed by vote of the steering committee shall be elected by the officers at the first steering committee meeting in every odd numbered year, but not less than two years from the recognition of the SIG by DECUS.

5.2 Past Officer members

Outgoing officers, unless ineligible or elected as an officer or appointed as an at-large member, or impeached from office shall become a past officer member of the steering committee for one year.

5.3 Impeachment of Officers

In accordance with Article III, the Steering Committee will accept any motion to remove an officer of the IAS SIG. The motion will be presented in the next Newsletter along with the comments of the remaining Steering Committee members and a request that members file a vote on the motion within 30 days. Should a majority of respondents comprising at least

1/4th of the membership at the time of the Newsletters distribution agree to the removal, the officer is impeached, and must be replaced by election by the members, as described below.

5.4 Nominations

Should an officer be impeached, or all four Steering Committee officer positions become simultaneously vacant, nominations for that(those) position(s) will be accepted by the Newsletter Editor, or the person designated to function in that capacity. The nominees will be contacted, and shall accept by filing a brief statement in their behalf to be published in the next Newsletter. All members may return the ballot published in that copy of the Newsletter. The nominee receiving the most votes will be elected and take office immediately.

Article VI
Meetings

6.0 General meetings

Business meetings shall be scheduled at the Spring and Fall DECUS U.S. Chapter symposia.

6.1 Steering Committee meetings

The Steering Committee shall meet by phone prior to each general meeting, or at the Chairman's request, and shall also meet at each DECUS U.S. Chapter symposia.

Article VII
Amendments

7.0 Amendments to these operating procedures shall be made by a majority vote of the Steering Committee. An amendment will be proposed at one meeting, and voted on in a future meeting, with the proposed amendment published in the Newsletter in the interim.

This document will attempt to describe the problem that occurs when trying to use some DMA I/O devices.

A brief lesson in DMA addressing will be required before getting to the heart of the problem.

Most unibus devices using DMA mode of operation contain several hardware registers. Among these H/W registers is a memory address register (MAR) that holds the lo-order 16 bits of the memory address into which the data starts going, and another register that contains the 2 hi-order address bits (MA16, MA17). When the DMA transfer is started, the device places the full 18-bit address on the bus. For 18-bit addressing machines (i.e., PDP-11/45), this is fine. But for 22-bit addressing machines (i.e., PDP-11/44), the computer interprets the 18-bit address slightly different. The upper 5 bits of the address references 1 of 32 unibus mapping registers (UMR) that contains the 22-bit base address for the transfer, and the lo-order 13 bits of the address is added to the base address to yield a full 22-bit memory address. This is the memory addressing sequence performed by the different computers.

Now a lesson in the functions of device handlers. When a request for input from a DMA device is made from a user program, that request is passed to the device handler. A call to a system subroutine to determine the physical address of the user's buffer is made. Returned to the device handler is a two word value. One word contains the lo-order 16 bits of the physical address and the other word contains the 2 hi-order bits of the physical address giving an 18-bit address. If this device is on a 22-bit addressing machine, the system subroutine selects an unused UMR and loads a 22-bit base address into it. The 18-bit address returned reflects which UMR is to be used as well as an offset from the 22-bit base address. (Remember the upper 5 bits reference a UMR and the lo-order 13 bits are an offset to the base address specified in the UMR). The device handler now loads the MAR and MA16, MA17 as returned by the system subroutine and starts the transfer. On 18 bit machines, the 18-bit address returned was actually the physical address and the data is transferred to the right physical memory locations. On 22-bit machines, the 18-bit address references a UMR which contains the 22-bit base address and adds the lo-order 13 bits to the base to reference a 22-bit physical memory address and the transfer is into the correct physical memory locations.

NOW...
Why is there a problem with some DMA I/O devices? It's simple...and occurs when MA16 and MA17 are hardwired, not programmable. When the system subroutine selects a UMR where the upper 2 bits are different than what the I/O device is hardwired for, the UMR selected when the 18-bit address is output by the device, and the UMR preset by the system subroutine will not be the same, causing the data to be transferred somewhere in memory (unpredictable as to where), causing terrible things like crashes, CPU hangings, etc.

Ken Guralnik
EG&G Energy Measurements Group
Las Vegas, NV

digital

26 January 1983

Mr. Robert Curley
IAS LUG Chairman
University of Pennsylvania
3400 Spruce Street
Philadelphia, PA 19104

Dear Bob:

It has been nine (9) months since support for IAS was moved from Reading, England to Maynard, MA. I am pleased to say the transition could not have gone smoother.

The IAS Development Organization is now firmly established and going about their business of providing quality support for IAS. I am pleased to announce that just last week another IAS engineer was brought on board.

Our IAS development organization is now larger than the Reading organization and we have not yet reached our planned staffing level.

Aside from these tid-bits of IAS news I would like to thank you and DEVIAS for your support of our efforts.

Regards,

Timothy M. Leisman
IAS Product Manager

TML/cap

DIGITAL EQUIPMENT CORPORATION, 40 OLD BOLTON ROAD, STOW, MASSACHUSETTS 01775

digital

15 June 1983

Mr. Bob Curley
P.O. Box 322
Flourtown, Pennsylvania 19031

SUBJECT: SPRING DECUS 83 IAS ANNOUNCEMENTS

The following IAS announcements were made at DECUS recently held in St. Louis:

1. IAS supported licenses are no longer a requirement to purchase the product first time. First-time customers are only required to purchase a DZ (unsupported license) license and an H-Kit (binaries and documentation).

2. The previously announced retirement date of June, 1985, has been retracted. IAS support will continue as currently available until further notice. We may modify support but we do not intend to completely discontinue support. One year's notice will be given to customer prior to any changes in services/support offered.

3. BASIC and DECsupport will continue to be made available where currently available.

4. Phone support has been enhanced by having the IAS DEV. Group provide direct backup to the Support Center.

5. The price of IAS unsupported license price has been reduced over 50% effective 3 July 1983.

6. Autopatch service will now be distributed to all IAS SMS customers effective with the release of Autopatch E which began shipping late May, 1983.

7. Future releases of IAS and many of its layered products can be expected. "What" and "when" are futures which were not discussed.

I believe this list summarizes the important points.

Regards.

Sincerely yours,

Tim Leisman

Tim Leisman

TL:t

DIGITAL EQUIPMENT CORPORATION, 40 OLD BOLTON ROAD, STOW, MASSACHUSETTS 01775
(617) 897-5111

What I'm going to try and do is just give you a rough idea of how the IS EXEC looks from the inside and the major parts to it, the overall flow, and some idea of why it looks the way it looks. All of the charts that I am going to show that show memory layouts will have high memory on the top and low memory on the bottom. It seems to be the current standard. We go through various phases inside the company and every now and then do a 180 shift and confuse everybody including ourselves. The EXEC, like everything else, is constrained to live within a 32K virtual address space and is split up into eight 4K segments numbered zero through seven. Each one corresponds to an act page register which is what the memory management unit uses to translate from virtual addresses into the physical addresses. I am not really going to go into the details of how that works but if anybody doesn't know and is interested, one of the hardware handbooks will tell you in a fairly adequate fashion. We have to communicate with the license and some of the registers that are within the processor memory management unit and since the top 4K of the physical address space is devoted to those registers, we use the top APR of the virtual address space to map to them which gives us the external page setup there. I use the term external page rather than IP page because it talks to more that just the IO stuff, but you can also get memory

management registers and other things like that. Not all of the code that is executive function is actually within the EXEC I&S. The best example is within SCOM, but you don't need it all the time so there is no point in keeping it always in the EXEC and thus causing the EXEC to be bigger, but you have to have some way so that the EXEC, in these privilege tasks, can communicate with each other so we have a data base called SCOM that contains some routines to manipulate the data, to do some synchronization of various functions as well as the data itself. We always use APR6 to map SCOM at the top of that 4K chunk we got the routines that I mentioned earlier, some tables that are, actually for the most part, just random data items such as the current task, decisive memory and things like that and then some variable lists and the variable area where we can get more memory if we need to increase the size of the list. That latter two combinations are known as the neutral. If necessary, and you do specify this, this in system phase l of the SCOM directed if you can get the SCOM to be more than 4K up to l2K in which case you would use APR5 and 4 in that order to map SCOM. I have SCOM in parenthesis in item 5 and 4 since they don't always map to SCOM. If you don't have SCOM that is over 4K in size or over 8K in size depending upon whether you just use 6 and 5, you don't get the unused APRs for anything. They are just dedicated to that. The privilege tasks, although they use the user APRs and not the ... APRs, have their APRs setup exactly the same way as EXEC does.

The EXEC is obviously going to want to get to more than just the same 32K physical memory all the time; so what is done is that we use APR3 to map to whatever area of memory we want to get to. Where APR3 points is best explained by where it really shouldn't be pointing and that's where any of the other APRs point because there is really no need for it. Other than that, the way to really tell where APR3 should be pointing is to find out exactly where you are going at that particular point. If you are going over a crash dump and you happen to find APR3 pointing at any particular point, it is not necessarily any cause for panick unless you know what the EXEC is doing.

That leaves us three other APRs at 4K/APR and that is 12K which is the whole EXEC. We now have a problem because the EXEC is larger than 12K, but that is really not a problem because we can split the EXEC up into three different chunks: one that is no more than 8K in size and the other two that is no more than 4K in size, and the 4K chunks are pretty much independent of one another. You really don't go bouncing back and forth between them. You are in one and do something there and then go to the other one and spend a considerable amount of time there. What we can do is build the EXEC as if it were a memory resident overlaid task and have the EXEC actually do the overlaying and use the APR 0 and 1 to map to the route and APR2 to map to whichever the two overlaps we care about at that particular point and time. APR 0 and 1 should remain fixed once the system has been loaded but APR2 will go back and forth between

them. This is basically the way EXEC is laid out in virtual memory and the gross organization of the EXEC if you will.

Now I am going to go into a little more detail on how it looks on the inside. The best way to start is start taking a look at the overlay structure in a bit more detail, and on the top we've got this standard sort of tree structure with the route which is known as EXEC and the two overlays known as DIR and ASX. Near the route, you have low memory with all the trap and interrupt factors. Some code that handles all of the traps that can occur be they trap instructions and segment false and etc. There are keys for the clock service and some utility routines. Utility routines are chosen, for the most part, on one of two criteria and either they have to be in the route, or they don't have to be in the route, but there is room to put them there so we do. Overlay DIR contains code specific for directives. In other words, a code to handle any particular directive and until you apply the first patch for the Executive, it came out with 3.1. It also contains the code figures of which particular directive that you want, but when we came up with the first patch, we found out the DIR was getting rather full and there wouldn't be much patch space so the Directive Dispatcher got moved from DIR down into the route where there was enough room for it. ASX contains the code that is specified to scheduling with both the realtime scheduling and the time-sharing scheduling. The time-sharing scheduler is not always

present, but the realtime is; it is contained in EXEC EM05 and most of the entry points in EM05 start off with the three letters ASX. Why they start off with those letters I have no idea, but they do and that is why this overlay got the name of ASX. The other two names are fairly obvious.

Let's now take a look at how the EXEC is actually physically played out. The bootstrap is always in the very bottom of memory and with version 3.1, they are 500 octal locations in size exactly. The EXEC sits some place else usually in every system that I've actually seen it comes right after the bootstrap; although it is not constrained to come right after. The EXEC itself--the EXEC route is first, then comes DIR, then comes ASX and then comes SCOM after. I got ASX and DIR in parenthesis because it gives the answer to one question and it gets asked a lot. Why is there the huge gap between the EXEC and SCOM in my demo display? The reason is that the demo is kind of dumb and doesn't understand that the EXEC is overlay so we will show the memory that is occupied by EXEC but it won't show it for DIR or ASX. Rest assured that memory is not wasted, and there is really something useful there. At most, what you are wasting is simply what it takes to round up to a 32 word blocks at each overlay and SCOM can start on a 32 word block. Now is also a good time to bring out one ... that is in the system. What happens when you get a trap or an interlock? If your memory management unit is turned on, all traps and interrupts go through vectors which are located in kernal virtual

space, and thus most all of the traps and interrupts that can occur while the system is running will go through upper location within the segment EXEC, however, if your memory management unit is turned off, the addresses are interpreted as physical addresses and not kernal virtual addresses and there is one place were this becomes important, and I found out about this the hard way by tripping over it, i.e., when you get the power up. When the power fails, memory management is on so, of course, you will trap through segment EXEC but when the power comes back, memory management unit is turned off. You have to go through a physical location which turns out to be inside the bootstrap which is one of the reasons why the bootstrap has to set in in low memory but the EXEC is free from it. There is a minimal power recovering code in there, namely, just enough to turn memory management back on and jump into the rest of the power upcode within EXEC. If you were to put a hault in the beginning of the power upcode within EXEC, you would find, much to your amazement if you had taken a look at how the hardware is suppose to work, that upon power recovery, you have your memory management unit turned on actually because you have already executed the code in boot and not because your hardware has done something funny.

Now you have enough of an idea of what the major parts of the EXEC are, and now we can go into some of the modules and the best part to start with that is on the naming conventions.

Most all of the modules within the EXEC follow the following naming conventions. Those that start with EM stand for executive module. Those that start with ES are executive subroutines, TS is specific to time-sharing, DM is specific to a particular directive, and MP is specific to memory carry.

I want to go into some of the modules in a bit more detail. Almost all of the modules that start with EM are within the route. There are four exceptions. The first exception is EXEC Module 0 which contains a lot of data structure definitions, registered definitions and external page and the like. If you are familiar with the layout of the llM EXEC, EXEC Module 0 is the structural equivalent although not put together in the same way of the EXEC Module library where you have the definitions for TCBs and so forth. It contains a little bit of code which is for the ATl scan in the definition of the task states and the address of what routine to go to to handle the particular task state. The next one is EXEC Module l which is like llMs local or module contains the local or vector, kernal staff and some interior areas and also defies the base of tables that are used for VTL scan, the other end EXEC Module 0. EXEC Module 2 is basically a collection of routines for existing from an interlock or trap and also one other routine for processing the end of a direct if quiets there, I'll get to it in a little while. EXEC Module 3 has the power fail and as much as the power upcode is within the EXEC itself; remember there has been

other beginning parts to power up and stuff down the bootstrap. EXEC Module 4 covers all of the codes specified for initially handling a trap of any sort that can occur and any traps that are distinct from interrupts. The interrupt routines go through other places. And also, the code that will turn the traps into SSTs and almost all what it takes to actually effect an SST. The rest of it for actually transferring control back to the task is within EXEC Module 2. EXEC Module 6 contains the clock interrupt code and the clock servicing code. It is broken up into two parts, because you want to l) keep the time that you spend in the clock interrupt routine to an absolute minimum so all we do is go in and increment a counter. Then when we go through a Tech Module 2 to try an exit from an interrupt we find out if there are any clock ticks that occur. If they have occurred, then we go through the other part of the Tech Module 6 which actually updates the time dependent data base. We go through and increment a tick at a time rather than incrementing by the number of ticks that actually occur. The reason for it is if the system is working well and you don't have it heavily overloaded and there is not something wrong with the hardware, you should have no more than three ticks pending at anytime, but that is rather unlikely. Two really isn't all that common. You usually have only one; so if we only go through and do increments, it is a lot faster than doing adds and most of the time you aren't going to have to make another pass through it. We want to try and make this code as efficient as we can since you are going to be going through it rather often.

EXEC Module 7 contains the stuff that is declaring ASTs and exactly how that works I am not going into, but basically when you have to say yes somebody should get an AST because routines within EXEC Module 7 that are used to actually handle that. EXEC Module 8 contains floating point exceptions interrupt or traps. First, this sounds as if it is duplicating code that was back in EXEC Module 4 with the other trap handlers but it is not really. What this handles is the floating point processor trap which is asynchronous. If you come bombing in even after the task has entered the EXEC and so you can't really effect the SST at that particular point. What we do is set a flag and the same work where we keep track of the clock ticks that says we have a floating point processor interrupt and then we exit from that interrupt when we are going through EXEC Module 2 and checking for clock ticks we also checking to see if we've gotten or press for significant event or floating point processor interrupt and it is at that point that we will then go into the common code in EXEC Module 4 to actually effect the SST if the user task has actually setup for an SST. The EXEC Module 9 I will skip for a minute. EXEC Module 10 is the one which handles the directive dispatching and directive return and it is this one that use to be in DIR and got moved down into the route. Then there is EMNULL which is the NULL task which doesn't run any user mode but runs from colonel mode so that we can switch back and forth between the NULL task and the rest of the EXEC very quickly. EXEC Module 9 is not any

place. The reason for it is back before the EXEC had to be overlaid a whole collection of subroutines could be put in one module which was then one monolithic EXEC but when we have to overlay it, we couldn't fit everyone within the route so it was done as EXEC Module 9 was split up into a collection of routines all starting off with ES something and their names are fairly descriptive of what sorts of routines are in there. EST PARR, for example, is the executive subroutines for handling T type partitions so on and so forth. Two others that are worth mentioning whose difference may not be readily apparent are ESSFL for the swap file manipulations or routine and ES SWAP, swapping and checkpointing routines. The first one for actually manipulating the swap file itself--allocating would be allocating space and then filling in actual parameters for the physical location on disc of the swap file when you are doing IO2 or ... Whereas ES SWAP contains the code that actually says we want to do some swapping or checkpointing. From the point of view of actually moving the task out of memory and bringing it back in, there is no difference between swapping and checkpointing. It is all handled by common code. The difference is in deciding whether or not you want to swap or you want to checkpoint; that is why that code is all together. The other two executive subroutine models which might if not at first be apparent why there are two of them are ES PLAS for PLAS directives and ESTIOR which are for general subroutines. We have it split up so if you don't ask for PLAS you don't have to have the code in there

for these utility subroutines that are only needed for the PLAS directives. The other EXEC Module I left out which isn't in the route, is EXEC Module 5 the master ATL5 which is in segment ASX.

Then we come to a collection of routines that start off with TS for time-sharing modules, data base, utility routines, and then of course, TS SCHED which is the time-sharing scheduler itself which resides in overlay ASX when it is around. If you have a realtime system you don't have TS SCHED. If you have a multiuser or time-sharing you do, of course, have TS SCHED. Then the various modules that start off with DM which handle a parti- cular directive or related class of directives. For example, there is one module to handle the ABORT directive and one module to handle both an ABLE and DISABLE checkpointing and then the various memory ... routines including when you don't want any particular memory parity at all. Then there are a few other odd balls for trace and ODT. I am not quite sure whether or not ODTs still really work. John Harper, who did a lot of work to bring about IS Version 3 claims in some comments within the terminal, that he could never get it to work. That is sort of like Cutler claiming there is a part to DMS that he could never get to work.

Now we have a description of the basic way out of the EXEC with the basic players. Now what I am going to try and do is give

some idea of the overall flow through the EXEC and take you through the handling of a particular directive request. The next place to start is with the ATL. The only data structure I'm really going to mention in any detail. The ATL is in one sense the master list within the system. In a sense you can say that the way the system works is that you have a list of entries that describe all of the tasks that you would like to run. The way the Executive decides which one to run is by having a collec- tion of routines which will make particular modifications to the ATL entries depending on what happens, it's dumb waiting for it, it has an event flag set, etc., and then EXEC Module 5 which will go through and scan the ATL and actually say such and such a transition has occurred and I now know that I can or cannot do such and such a thing with this particular task. In particular, what EXEC Module 5 is trying to do it trying to find a task which is runnable. Unless the system gets scrubbed, it will always succeed. While there may not always be a task that is in state run which means you got someone who can use the run in user mode the IDL task should always be around in state IDL. Uses a different state so that the ATl scan knows that when it goes to set up to do a contact switch to IDL we setup to contact switch to something that is not a real task and it is going to run with a colonel mode. We want to try and cut down on the time it takes to switch back and forth between the NULL and the rest of the system. The NULL task should always be there; it should always be runnable

so the ATL scan should never fail to find at least something to which it can give control. If the WAIT set module 5 works as it goes through and scans the ATLs looking at entry by entry its major item of concern in the task state ... up TS which is a byte and has a numeric value in it unlike llM which uses bits to try and indicate the task state; we just use a binary value which we can use as an index. The task state which has and all the ... start with TS dot and three letters describing it. For example, the IDL task is always in state TS.IDL and is what we used to get through to figure out what to do. While most programs should not concern themselves with the actual value of the task state when you started to consider EXEC Module 5 and a few other parts of the system or you don't need to know the actual binary values themselves, it is important to understand that there is an ordering in a relationship between some of the values not simply because it is used as an index into a jump table. I mean, you can just reorder the entries in the jump table and everything would be fine and that part would be corrupted for there are a few other places where the EXEC will not actually play some tricks but do something fancy. For example, when a task is into a wait for state, it will always start out in WAIT FOR 0; it doesn't matter what group of event flags you are waiting for. It also will go into WAIT FOR 0 to begin with if you are doing a stop for event flag. Then what the EXEC does is it finds out which particular group that you are interested in and adds that group number to the task state

which will, because the way the task states are ordered, transform it into the appropriate WAIT FOR state and actually they stop for and then adds to the task and the difference between WAIT FOR 0 and STOP FOR 0 which will transform the appropriate WAIT FOR state into its corresponding STOP FOR state. If you are interested in going through EXEC Module 5 and seeing what is going on, it is a good idea to first take a look at that part of EXEC Module 0 where it lists all the task states defining them and pay attention to those particular task states where it says the ordering is important and it will help you understand some of the funny tricks that it will pull later on. There are also some other parts of the system, for example, with I) rundown where it will just do an increment in the task state and if you will take a look at the ordering once again you will notice it is in this particular state, and the increment will get it into this next particular state. This is one place where you will have to know more than just the neumonics for them to really understand what is going on.

Now, we have enough to give a brief outline of what happens when you issue a direct and this gives you a fairly good idea of the overall flow through the system. In the issue directive which means you are going to issue an EMT. EMT is a trap so we go into EXEC Module 4. We take a look and find out if the trap is issued from user mode but if it is issued from KERL mode that was fatal. But it wasn't, it was issued from user mode

and we find out whether or not it is an EMT 377 so we know whether or not it is a SST or it is a directive. In this case, it is a directive. Now we map to the directive overlay and then jump to the Directive Dispatcher. This is a little premature in that we now have the Directive Dispatcher in the route but it really isn't going to be any waste of time to do it. All that is done to map to it is simply to haul the appropriate value out of a table within the EXEC and stuff it into KERL APR2. We don't have to save and restore KERL APR2 as far as the EXEC itself is concerned because we know what the other value should be and we really don't do a lot of fancy flipping back and forth be- tween them. You just know you are going into this overlay so I map to it. Once you get into the Directive Dispatcher it will go through and try and find out what particular directive you are talking about. If it is not a valid directive or it's a valid directive but you don't have the privilege for it, it will return to directive status, exactly how, I will mention in a little bit. Let's say it is a valid directive and in this case, a WAIT FOR IT, we will then go to the appropriate direc- tive module NPIR and in this case, it would be DMSET which handles all the WAIT FOR and a few other related things like decline significant events. It is here that we go and take a look and find out if the event flag is legal or not and other things like that are really unique to the particular directive in question. If this is all good, we will then put the task into the WAIT FOR state. You asked,

what happens if one of the event flags in WAIT FOR is already set. At this point, we don't care. It helps the overall flow of the EXEC to really not worry at this point but to worry a little bit later on whether or not the WAIT FOR or certain other similar conditions have actually been satisfied. We just put it into the state WAIT FOR and then we issue directive status. That is used by using the trap instruction. We use it because the low byte can contain a value, and specifically, we put in there the value for the Directive Status--success or failure. The trap gets you back to EXEC Module 4 which finds out that you are returning Directive Status and so you go back and do EXEC Module 10 where you got the return directive status code. In other words, you go through and find out if it is a trap and you haul off the Directive Status, find out where the Directive Statphysically in memory of the appropriate value and do any other updating you need. Then you jump down to EXEC Module 2 to do the indirect processing and all this really is, is a place within the route so that you can now do remapping where you remap not from DIR but to ASX and then jump into EXEC Module 5 to go scanning the ATL. In this case, you will start scanning from the task which was last in control. Since in this particular case it is in WAIT FOR state, M05 will then say that I should go to the routine that handles the WAIT FOR state and now if the task has actually got an event flag set which would satisfy the WAIT FOR, then it will get set back into run. If the somebody isn't satisfied, M05 will continue

scanning until it finds someone it can run, possibly the IDL task. Once it is finished doing that, it will then go down and do EXEC Module 2 for the common EXIT, and it is here that we decide to find out if there are any clock ticks that have occurred. If there has been a floating point interrupt, it is interrupt rather than trapped because the trap we usually refer to the synchronous 40 floating point whereas, the interrupt from the asynchronous floating point processor. If there has been clock ticks, we will go to EXEC Module 6. We will do the updating of the time dependent data base, and then back to EXEC Module 2 to find out if any of the flags such as floating point interrupt or repress for signficant event had been set. If they have been, we will then go back to EXEC Module 5 and start scanning the ATL all over again from the top if necessary. Eventually you will get to the point where the significant system event recognition flag, which is what contains the count of the clock ticks and these other flags, is zero. Now you can actually go and give control to whatever task has been selected. I've got the tasks in different colors because you aren't necessarily going to go back to the same task that issued the directive of that particular point. If you do switch from one task to another, it is EXEC Module 5 when it finds someone who is runnable that will say is it the same task that was running and if it is, fine, and if it is not, I've got to go and save the old context and get the new context set up so that if EXEC Module 2 decides to actually return to a task, it has what it

wants and where it wants it. That is the basic flow through the Executive. The one part that is left out is how do we handle time-sharing and time-sharing. From one point of view, it is just another realtime task so that the particular flow for scanning the ATL and so forth also applies for time-sharing. It is how we get the time-sharing task in question onto the UTL which is what separates time-sharing from realtime. Realtime tasks always stay in the ATL whereas time-sharing tasks appear and disappear from the ATL.

Time-sharing scheduling for the most part is TS SCHEDS FUNCTION. Its corresponding list to the ATL for its purposes is the UTL (user task list). The user task list consists of a linkage the UTL nodes which are contiguous extensions to the related to the ATL node for the time-sharing tasks in question *What you are saying now, does this also apply to a multiuser ...?

*Yes, there is no difference in anything that I am talking about between the time-sharing and the multiuser system. There is some strange piece called TCP which is what makes the difference between time-sharing and multiuser. The time-sharing tasks will have a longer ATL node because they have the UTL stuck on the end. The UTLs are organized in lists which have listheads, one for each level that you setup during ... What TS SCHED does is it goes through these UTLs in a particular way; I'm not going to say exactly how because it is written up

fairly in both the System Manager Guide and the Performance and Tuning Guide with exactly the algorithms that it uses for figuring out who it is suppose to grab. It goes through these levels starting from the highest level which has the lowest number, down to the lowest level trying to find somebody that it considers runnable. If it finds somebody that is runnable, he then goes and puts it on the ATL. He doesn't actually put it onto the ATL but what he does is he moves it. All the time-sharing tasks are always on the ATL; it is just when they are not the current one that they are linked after the NULL task. From EM05 point of view, the NULL task is always runnable, it will never scan down and notice any of these tasks. EM05 point of view the time-sharing tasks go on and go off of the ATL, but from the point of view of TS SCHED, they are always somewhere on th ATL. There will even be either priority l or priority 0 depending on whether or not the time-sharing scheduler has actually seen them or whether it has never seen them before. If you are trying to find out exactly what is going on with yours and you take a look at a crash dump ... out of a ATL and you find some task that has got a priority l and some have a priority 0, the difference is that the first time-sharing scheduler is seen and done something whereas the others is yet to see. They are on the new task list. Once it is found, the ATL node for the particular task that it wants to run it puts it in the ATL after a node that really doesn't correspond to any real task. It has a task state of TSl and

when the EXEC Module 5 encounters this node for TS1 it goes into TS SCHED in one particular entry point where TS SHED knows it has to do something about scheduling. The ATL node for the task that you want to run is linked after it and then the node for TS2 which is basically the same as the one for TS1. The real difference is that you will go into TS SCHED in a slightly different place. The difference in it is that you will get to TS2 on .... task voluntarily gives up. In other words, it issues a directive or something along the ... or something like that and the ATL scan goes from here down to here so that when the time-sharing scheduler is entered at that particular entry point for TS2 it knows that somebody has given up ahead of their quantum expiration and so I have to go through and do a little bit different rescan of my list to find out who I want. Once we have all of this in here now, we just consider the ATL from here down to here as if it were a strictly realtime system. We just start scanning down the list from wherever we happen to be or a significant event has been declared from the top which will always be the entry for the system disc camera, and just scan down until we find someone who is runnable. At the very least, we will encounter the NULL job and transfer control of that. What the time-sharing scheduler will also do in addition to this is set the timer and if the timer expires a signficant event gets declared so we are going to start the ATL scan from here and we will encounter TS1 again. The time-sharing scheduler remembers that it has someone on the ATLs so

it does things a little bit differently. Namely, it will take this guy out of here and link him down here and go looking for somebody else. If it can find them and it links the new one up there, and if it can find them it will also remove the TS2 node. If you have the time-sharing system with the time-sharing scheduler active you will always have the TSl node in there but if it is not a time-sharing or multiuser system or you don't have the time-sharing scheduler active the TS node is not around.

LE:mcm:7l50A:0l/06/83A

Larry W. Ebinger,
Sandia National Laboratory,
Division 7121,
Post Office Box 5800,
Albuquerque,
NM 87185

digital

February 24, 1983

Mr. Bob Curley
P.O. Box 322
Flourtown, PA 19031

Dear Bob:

As you are aware, Digital's recommendation regarding the cleaning of certain disk packs has changed since our previous correspondence in June, 1982 (see attached letter).

During the past few months Digital has done extensive studies on certain disk packs with regard to pack cleaning and we found that, if done properly, it enhances data reliability and the life of the pack. As a result, we are offering a pack cleaning service for RL01, RL02, RK05, RK06 and RK07 disk cartridges that embodies our recent findings.

For other packs, the previous philosophy is still in effect but we expect tests to be completed on those disks in the near future. Therefore, there is a high probability that this philosophy will change as well.

For the packs that we currently recommend cleaning, Digital advises the use of a cleaner that utilizes soft foam-type sponges and a freon-based liquid cleaner to clean disk surfaces. We feel this method is more effective because:

1) The liquid remains on the cleaning head both top and bottom;

2) sponges have squeegie effect in removing liquid from surfaces as the disk rotates during cleaning; and

3) the freon-based cleaner evaporates rapidly with less chance of any cleaning fluid left on disk surface or cartridge's interior surface after cleaning.

Under Digital's Media Mainatenancae Service, specialists will inspect and clean the disk media using the above method twice a year at the time of scheduled Preventive Maintenance visits. The cost of this service is $2 per pack per month.

DIGITAL EQUIPMENT CORPORATION, 1740 WALTON ROAD, BLUE BELL, PENNSYLVANIA 19422
TEL: (215) 825 - 4200 TWX: (215) 825-4200

February 24, 1982

I have enclosed a brochure on Media Maintenance which will provide you with some additional information on the service.

As always, please feel free to contact me if I can assist you in any way.

Sincerely,

DIGITAL EQUIPMENT CORPORATION

Sandy

Sandy Russell
Marketing Specialist

SR:nm
Enclosure

The IAS Working Group has shifted its focus. After the DEC announcment that IAS had reached its origional design functionality with Version 3.1, a mad scramble took place with users quickly converting to other operating systems, forming this working group, taking surveys to convince DEC that they had made the wrong decision, leaving the profession, or contemplating suicide. The dust has now settled and we now see that DEC is slowly pulling RSX-11 M Plus to where IAS was years ago, RSX-11M is also approching its design functionality, and all the main focus within DEC appears to be VAX/VMS and the Personal. DEVIAS and the IAS working group have accomplished a major goal in getting continued software support thru June 1985, so we now need to shift our focus. Self-help is now the goal of the IAS working group. We need to know:

1. What you have done that you can share with the other IAS users.
2. What you are doing or plan on doing that you could share.
3. What you would be willing to do for the benefit of other users.
4. What you would like to have done by someone with the resources.

We need ideas - lotsa ideas. Maby a menu for the working group. I have lots of things I would like to do for our IAS sites, but my priorities are set by my sites. If I knew that one or several other sites wanted the samething I could shift my priorities. Also, other people have ideas that I have never thought of that I would like to implement. Multilpy this by the number of active IAS sites and you have a great deal of potential. I beleive IAS is the best operating system for the mid to large scale PDP-11. Just look at the record - IAS has reached its design goal in just 3.1 versions. RSX-11 M is up to version 4.0, RXS-11 M+ is up to version 2.0, RSTS is around version 7, RT I've lost track of, and VMS is soon to have version 3.1 and IAS can still do things all of these other systems can't. DEC is pulling RSX 11M+ and VMS up to where IAS is, but if they had chosen to continue development work on IAS ... . I counted 12 items on the RSX/IAS menu that were current features of IAS. If you have any inputs on these or any other topics that may be addressed by the IAS working group, please contact me either by mail or at the Symposium.

John Jenkinson
MOSTEK Corporation
1215 West Crosby Road
MS 32
Carrollton, Texas 75006

IAS

- IAS resurrected
- Autopatch "E" now being shipped
- Phase 3 DECNET support to be implemented
- All utilities for IAS will be maintained by the IAS software group (after July '83)
- DBMS-11 will not be supported after Dec./83 for IAS
- IAS may be implemented on the Q-bus hardware
- Fortran 77 V5 has no IAS support yet
- Looking into dual porting disks (2 ACPs)
- Software support based back in U.S.

Pricing Policy

- 50% reduction in unsupported license

Syed Mir
Ontario Hydro

June 23,1983

Dear Mr Curley,

This is a note to request your assistance, through the DEC Users Group, in locating a communications handler to run under the IAS operating system that can talk to DEC's multi-channel, synchronous communications card -DV11.

Thomas Ford
Veda Inc.
2 Three Notch Rd.
Lexington Park,
MD 20653
(301)862-2100

System Communications Area Node Pool

Problems and Possible Solutions

On the IAS Operating System

SCOMM NODE POOL PROBLEMS ON IAS

Section Ø -- Introduction

The following pages will provide on overview of the IAS System Communications Area (SCOMM). This will include a discussion of the memory layout of SCOMM and a brief description of the data structures maintained by the system in SCOMM.

The second half of this overview will concentrate on increasing the number of nodes available in SCOMM for user task execution. Reducing the number of nodes needed by the system as well as increasing the total number of nodes available will be discussed.

The overview has been divided into the following sections:

Section Ø -- Introduction
Section 1 -- Memory Layout of The System Comunications Area
Section 2 -- Data Blocks Maintained by IAS in SCOMM
Section 3 -- Reducing The Number of Nodes Used by the Operating System
Section 4 -- Increasing the Total Number of Nodes Available
Section A -- An Example of Increasing the Total Number of Nodes Available

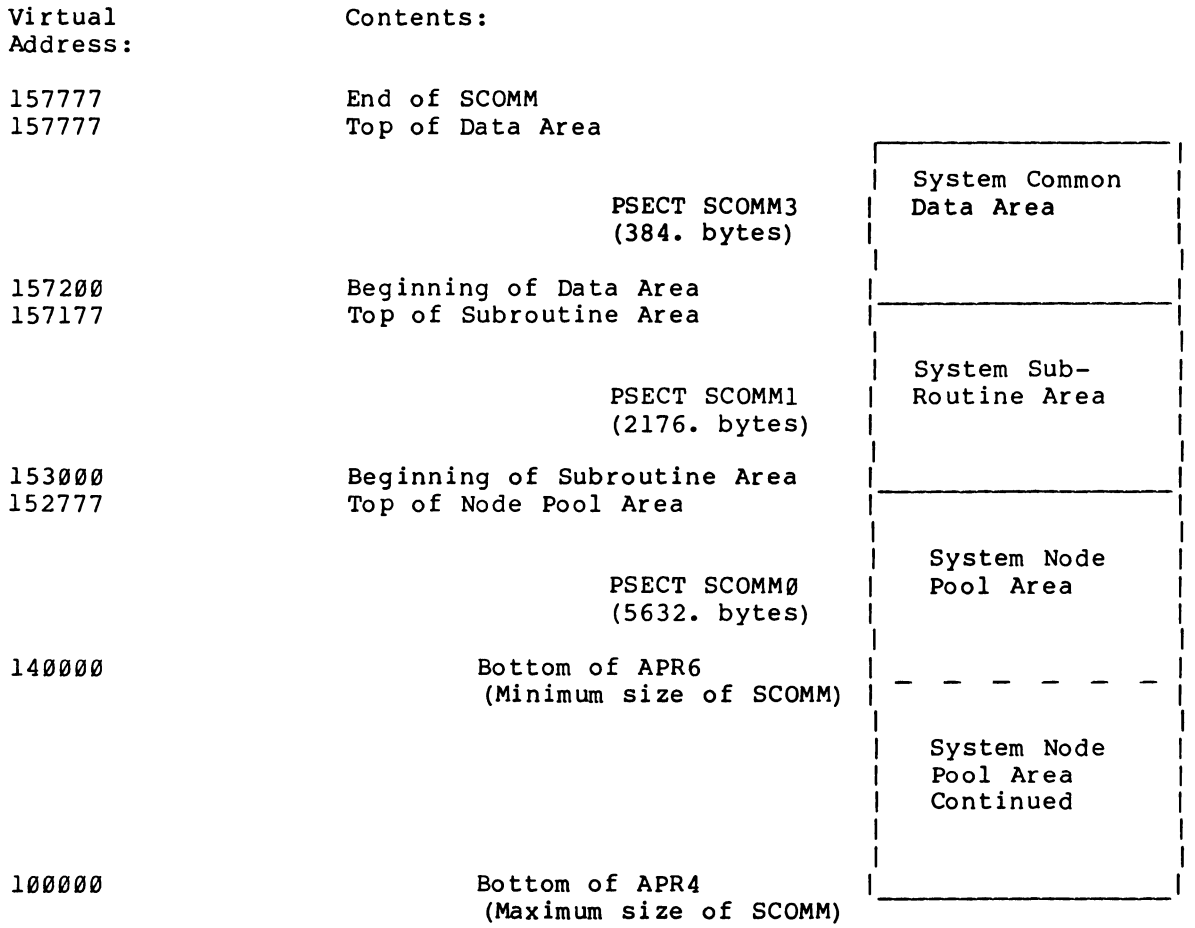
DECUS U.S. Symposium

Fall 1982

SCOMM NODE POOL PROBLEMS ON IAS

Section 1 -- Memory Layout of The System Comunications Area

SCOMM is organized into five Program sections (PSECTS). These sections are placed in memory by sysgen phase 1 (SGN1) as shown below. The addresses are based on IAS V3.1 as released.



1. PSECT SCOMM3 includes 16. bytes of fill to round in up to a multiple of 64. bytes in size.

2. PSECT SCOMM1 includes 116. bytes of patch space.

3. The minimum size of the System Node Pool Area is 5632. bytes or 352. nodes and the maximum is 22016. bytes or 1376. nodes.

Section 2 -- Data Blocks Maintained by IAS in SCOMM

The IAS Executive spends much of its time processing data contained in lists (sometimes called queues) within the Node Pool Area of SCOMM. A list consists of blocks of one or more contiguous nodes linked together. The address of the first block in a list is stored in a fixed word in the SCOMM Data Area (PSECT SCOMM3). A second word in the Data Area hold the address of the last block within the list .and within each block holds the address of the previous block in the list. Two words in each block are used to store the address of the next and the previous blocks (elements) within the list. Some lists use only one word to link the blocks together. Generally the address of the first element in such lists is held within an element of another list. And some lists do not use these link word(s). Instead all of the blocks are allocated contiguously in the Node Pool Area when the system is generated.

The major lists maintained by the IAS system are:

Asynchronous System trap Queue

Purpose:

Each ASQ block describes an Asynschronous System Trap (AST) to be executed for a task.

Size: Minimum - 1 node often another block (IRQ for example) is used as an ASQ block

Active Task List

Purpose:

Each active task has an ATL block which describes its current status, owner, I/O counts and other information needed by the EXEC for scheduling.

Size: Non-timesharing task - 3 nodes
Timesharing task - 4 nodes

ClocK Queue

Purpose:

Each function scheduled to be performed at some time in the future has a CKQ block allocated for it.

Size: 2 nodes

Global Common Directory

Purpose:

A GCD block is allocated for each Sharable Global Area (SGA) installed in the system, each region created by the CRRG$ directive and the Read-only (pure) area of installed tasks.

Size: 3 nodes

Interrupt Service Node

Purpose:

The ISR node is used to pass control to the device handler task's Interrupt Service Routine when a device interrupt occurs.

Size: 2 nodes

I/O Request Queue

Purpose:

Each currently outstanding I/O request is described by an IRQ block.

Size: Task I/O (QIO) requests - 3 nodes Executive I/O requests - 4 nodes

Memory Usage List

Purpose:

An MUL block exists for each allocated segment in a Timesharing (T) type partition.

Size: 1 node

Physical Unit Directory

Purpose:

A PUD exists for each device specified during sysgen and contains the information needed when accessing the device.

Size: 4 nodes

Swap File List

Purpose:

An SFL block describes each currently available swap file.

SCOMM NODE POOL PROBLEMS ON IAS

Size: 2 nodes

Send/Receive Queue

Purpose:

An SRQ block is created for each Send or Send by Reference directive issued by a task. A QIO request which is an ACP function also causes an SRQ to be created and queued to the ACP task.

Size: Minimum 3 nodes - Send directive (data dependent)
3 nodes - Send by Reference directive

System Task Directory

Purpose:

Each task installed in the system is described by an STD block. In addition an "Alpha Table" is created during sysgen which provides for a list of pointers to the STD entries.

Size: STD - 2 nodes
Alpha Table - specified during sysgen

Spawn Task List

Purpose:

An STL block is created for each task spawned by the SPWN$ directive. Part of the block contains the command line passed to the task and is deallocated when the task issues the GMCR$ directive.

Size: 6 nodes - Including command line

Task Partition Directory

Purpose:

The TPD block describes a partition in the system. Sysgen phase one creates a TPD block for each partition defined at sysgen time.

Size: 2 nodes

User Task List

Purpose:

A UTL block exists for each active Timesharing task. It is used to schedule the timesharing tasks.

Size: 2 nodes

Section 3 -- Reducing The Number of Nodes Used by the Operating System

Reducing the number of nodes used by the operating system generally involves reducing the number of elements in one or more of the system maintained lists. For example installing fewer tasks results in fewer STD blocks and a savings of 2 nodes per task not installed. This is not without side effects, however. Many installed tasks respond to MCR level commands and receive their command lines without prompting because they are installed. This also affects PDS users since PDS commands are usually translated into an MCR type command. However Tasks which are only rarely invoked could be removed with minor loss of functionality.

Reducing the number of elements in other lists has a similar effect. System functionality or throughput suffers.

It is possible to modify the operating system and thereby reduce the effect of removing elements from the system lists. These modifications range from minor patches to major re-writes of various components within the system. For example:

1. The timesharing pseudo device PI could "auto-install" a task if needed before spawning that task. Since it is possible for more than one user to request the same task at the same time the executive would have to be modified to remove the task when the last copy exits. Spawns to a task installed in this way would be allowed. The INS and REM tasks would also have to properly handle such tasks.

   This sort of install when needed, remove when not need could be generalized to include any task found in the Default UIC (DUIC) specified during sysgen phase 1. In this way installed tasks would include those with an STD entry and those found on device LB: in the DUIC.

   The additional overhead and possible reduced throughput resulting from this modification would have to be evaluated by each system manager.

2. The PUD entries for such pseudo devices as SY:, LB:, and TI: could be eliminated. Two words in the SCOMM data area would serve as the redirect pointers for devices SY: and LB: respectively. The SCOMM subroutines FDEV (Find DEVice) and REDT (follow the REDirecT chain for a device) would be enhanced to handle these 3 names correctly.

   The effect of this modification on current privileged tasks is not yet known.

3. When a IRQ block is queued to a device handler, the handler examines the function requested. If the function is an ACP function (Enter a name in a directory, read file attributes, etc) a message is sent to the ACP task informing it of the request. This use of an SRQ entry could be eliminated by queueing the IRQ directly to the ACP.

Section 4 -- Increasing the Total Number of Nodes Available

Although reducing the number of elements in system lists is one way to increase the number of nodes available for user tasks, there is a finite limit on the size of the SCOMM node pool area and on the number of nodes which can be made available.

1. By extending the size of the node pool area significant numbers of free nodes can be created. Due to the virtual addressing limits of the PDP-11 the entire SCOMM area cannot exceed 12k words in size. However, some of the lists contained within SCOMM are normally used only by the IAS executive. These lists could be moved into a second node pool area accessable only to the executive. This results in more nodes within the SCOMM area available for general use.

   This Executive COMMon area or ECOMM would be used for such lists as the SRQ, STL, CKQ, SFL, and Interrupt Service Nodes. When needed, ECOMM would be mapped through kernal APR3. The SCOMM node pool area would be used in the event of an allocation failure in ECOMM. A block in a list with an APR3 address (60000-77777) would be found in ECOMM, otherwise it is in SCOMM. An example of this technique as applied to SRQ entries is shown in Section A

2. The size of the SCOMM subroutine area can be reduced by approximately 70 nodes (1120. bytes) if certain of the subroutines are incorporated into the executive itself. An entry point for each displaced routine would remain in the SCOMM area. This entry point would PUSH the address of the called routine onto the user's stack and execute an EMT 376 instruction. The executive then calls the routine in kernal mode. Control is finally returned directly to the caller. Routines called in kernal mode are accessed directly.

SCOMM NODE POOL PROBLEMS ON IAS

Section A -- An Example of Increasing the Total Number of Nodes Available

Node Pool Extension for Send/Receive Directives

The following set of patches which will permit a user controlled partition to act as a node pool for the exclusive use of the Send and Receive Directives. Pool space for all other users remains in SCOMM and the effective dynamic memory size can be increased by up to 4k words. The patch will provide most benefit to IAS sites with a job mix that requires heavy use of send directives. The patches are fairly straightforward but do take up some space in the executive root and of course up to 4k words of memory. The run time overhead involved is not very substantial.

The method used to provide the extension is to use a 4k word user controlled partition (which must be called SENPAR) as an alternative node pool. It must be first in the TPD hence the first partition to be declared to sysgen phase one (SGN1). SOME unused parts of the TPD entry are reused to hold extra parameters. Whenever the executive has cause to access the Send and Receive Queue (SRQ) the scratch APR (i.e. APR3) is temporarily remapped to the partition. This of course also applies to any privileged task that accesses the SRQ (notably ...REM). The executive first tries to pick nodes from the partition as needed. If this fails for any reason (the partition doesn't exist or all of it's nodes have been allocated) nodes are picked from SCOMM as usual.

The patches are applied to three executive modules - ESMISC, ESDIR and DMSAR. In addition two new executive modules, ESPOOL and ESRNMS are included. These two modules contain most of the new code. SGN1 is modified to check the first partition for the correct name and size and to generate the modified TPD entry. Anyone considering using the patch should check make sure they are running no privileged tasks which modify the SRQ or make use of the four spare words in each TPD entry. Examination of the patches to REM demonstrate how a privileged task could be modified to work with the patch, should the need arise.

If the first partition specified to SGN1 is not SENPAR, the patched executive, REM and SGN1 tasks will function exactly as they do in an unpatched system. Thus it is possible to use the patched tasks to generate a system with or without the new feature

To apply the patches make copies of the affected tasks/files i.e.

```
[11,1]SGN1.TSK          [11,1]REM.TSK
[11,17]EXEC.TSK         [11,17]EXECUTIVE.STB
[11,17]SYSGEN.CMD *
```

* or whatever you call this file.

Modify [11,17]SYSGEN.CMD to include the new partition, e.g.

```
^*PAR=SENPAR,,4K,U\*
```

Copy the contents of this directory into a privileged directory (group number less than 11) on your system disk and set your current UIC to the privileged

directory. This is to insure PIP has write access to the needed directories on the system disk.

The installation WILL NOT delete or overwrite any existing files on your system disk. Only the OLB files will be modified (updated).

Check that your system disk has the required libraries present else copy them from the object distribution medium . The libraries are:

```
[11,15]EXEC.OLB;1     and     [11,15]EXEC.OLB;2,
[11,17]SG1LIB.OLB;1   and     [11,17]SG1LIB.OLB;2 *,
[11,13]LIB13.OLB;1    and      [11,13]LIB13.OLB;2.
```

* This file may not have been created by an article when this patch is applied. If so create it from version one.

Similarly the following command files should be available:

[11,17]SG1BLD.CMD, [11,15]EXECBLD.CMD, and [11,13]REMTKB.CMD.

The patch installation command file (NEWEXEC.CMD) will move all of the files from your current UIC to the appropriate UIC's as part of the installation process. The command file will also use SLP to create a new version of your executive ODL file ([11,15]EXEC.ODL) to reference the new modules. The command file also assumes that the ;1 versions of the OLB files are the unpatched versions and ;2 are the patched ones following the general conventions assumed in IAS Software Dispatch Articles

Perform a system generation using the modified SYSGEN.CMD. During SYSGEN phase one a fatal error may be given:

```
^*SG1 -- *FATAL* - NO SPACE FOR POOL\*
```

This error could be caused by bad SYSGEN planning; however it has been "reused" by the SGN1 modifications to indicate SENPAR is greater than 4K words.

The executive patch files:

[11,15]DMSAR.PAT

```
        .TITLE  DMSAR
        .IDENT  /V3.1SR/
        .PSECT  EXEC
.BLK.=.
.=.BLK.+20
        CALL    PDMS00
.=.BLK.+30
        CALL    PDMS01
        NOP
.=.BLK.+306
        CALL    PDMS02
.=.BLK.+420
        JMP     PDMS03
        NOP
        NOP
.=.BLK.+442
        CALL    PDMS05
.=.BLK.+450
        CALL    PDMS04
.=.BLK.+1050
        CALL    PDMS05

        .PSECT  ZZPAT

PDMS00:
        CALL    ..MPSP
        CALL    .NDSCH
        RETURN
PDMS01:
        BIS     #AF.RA,A.TF(R4)
        CALL    ..RMSP
        RETURN
PDMS02:
        CALL    ..MPSP
        CALL    ..PNMS
        RETURN
PDMS03:
        BISB    #EV.SE,.SERFG
        CALL    ..RMSP
        RETURN
PDMS04:
        CALL    ..MPSP
        CMP     R2,#4
        RETURN
PDMS05:
        CALL    ..RNMS
        CALL    ..RMSP
        RETURN

        .END
```

[11,15]ESDIR.PAT

```
        .TITLE  ESDIR
        .IDENT  /V3.1SR/
        .PSECT  EXEC
.BLK.=.
.=.BLK.+320
        CALL    PESDØ1
.=.BLK.+326
        CALL    PESDØ2
.=.BLK.+354
        CALL    PESDØ3
.=.BLK.+426
        CALL    PESDØ4
.=.BLK.+454
        CALL    PESDØ5

        .PSECT  ZZPAT

PESDØ1:
        CALL    ..MPSP
        CALL    .NDSCH
        RETURN
PESDØ2:
        CALL    ..RMSP
        MOVB    A.TS(R4),R1
        RETURN
PESDØ3:
        CALL    ..RMSP
        MOV     #.RQTI-A.TI,RØ
        RETURN
PESDØ4:
        CALL    ..MPSP
        CALL    ..NDEL
        RETURN
PESDØ5:
        TST     R1
        BMI     1Ø$
        CALL    ..RNMS
        BR      2Ø$
1Ø$:
        CALL    ..RNTV
2Ø$:
        CALL    ..RMSP
        RETURN

        .END
```

[11,15]ESMISC.PAT

```
        .TITLE  ESMISC
        .IDENT  /V3.1SR/
        .PSECT  EXEC
.BLK.=.
.=.BLK.+1422
```

```
        CALL    PAT01
.=.BLK.+1512
        CALL    ..RNMS
.=.BLK.+1520
        CALL    PAT02
.=.BLK.+2050
.NDSCH:

        .PSECT  ZZPAT
PAT01:
        MOV     A.TD(R0),R2
        CALL    ..MPSP
        RETURN
PAT02:
        CALL    ..RMSP
        MOV     #.RRQLH,R1
        RETURN

        .END
```

[311,15]ESPOOL.MAC

```
        .TITLE  ESPOOL -- RESIDENT EXECUTIVE
        .SBTTL  ROUTINES FOR SEND/RECEIVE POOL
        .IDENT  /3.1SR/
        .PSECT  EXEC
;
;
;
; COPYRIGHT (C) 1982
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
; EQUIPMENT CORPORATION.
;
;
;
;       AUTHOR: IAS Development Team
;       DATE:   13-Jul-82
;
;
        .MACRO  .INH0
        MOV     PS.EXP,-(SP)
        BIS     #140,PS.EXP
        .ENDM
;
.SBTTL  INTERNAL DATA AND MACROS
;
;
PSVPDR::.WORD   0               ; SAVED PDR3 VALUE
PSVPAR::.WORD   0               ; SAVED APR3 VALUE
MPFLAG::.WORD   0               ; SET IF SEND/RECEIVE POOL IN USE
;
.SBTTL  ..MPSP - MaP to alternative Send Partition
```

```
;+
; This routine is called to preserve the existing
; APR 3 contents and swap them for values determined
; in SYSGEN Phase 1. If the first partition is not
; SENPAR no mapping is done.
;
;       Output conditions
;               MPFLAG  Set if mapping done
;               CC-C    Set if SENPAR not in place, not mapped
;               Old PDR,PAR values saved in PSVPDR,PSVPAR if mapping done
;-

..MPSP::
        CALL    MAPCHK          ; MAPPING REQUIRED ?
        BCS     10$             ; CS IF NO
        MOV     R0,-(SP)        ; SAVE R0
        MOV     KP.DR3,PSVPDR   ; SAVE APR 3
        MOV     KP.AR3,PSVPAR
        MOV     .TPDBA,R0       ; GET ADDRESS OF TPD ENTRY
        MOV     T.CF(R0),KP.DR3 ; AND GET NEW  APR 3 VALUES
        MOV     T.CB(R0),KP.AR3
        MOV     (SP)+,R0
10$:
        RETURN
;
.SBTTL  ..RMSP - Return saved Mapping from Send Pool
;+
; The saved APR 3 values are restored.
;-

..RMSP::
        TST     MPFLAG          ; REMAPPING REQUIRED ?
        BEQ     10$             ; EQ IF NO
        MOV     PSVPDR,KP.DR3   ; RESTORE PDR
        MOV     PSVPAR,KP.AR3   ; AND PAR
10$:
        RETURN
;
.SBTTL  ..PNMS - Pick Node from Mapped Send receive pool
;
; This routine is based on SCOMM routine ..PENV and is
; called and returns parameters exactly the same. It
; checks to see if the new pool partition has been
; initialized and then  selects its node pool listhead
; accordingly. I.E. .POLLH for normal operation and the
; listhead in the TPD for the SEND/RECEIVE pool. Routine
; ..MPSP must have been called first to map the node pool.
;
;       Calling Sequence
;
;               R1      STD address of requesting task
;
;               R3      Number of 8 word blocks required
;
;       Return Conditions
```

```
;
;               R1      Address of node picked
;
;               CC-C    Set if no node picked, else clear
;
;-

..PNMS::
        MOV     R4,-(SP)
        MOV     R3,-(SP)
        MOVB    S.AV(R1),R4     ; GET ACTIVE VERSION COUNT
        BNE     5$
        INC     R4              ; NEVER LESS THAN ONE
5$:     CLR     R3              ; GET LIMIT PER VERSION
        BISB    S.PV(R1),R3
        MUL     R4,R3
        SUB     (SP),R3
        CMP     R3,S.PU(R1)     ; CHECK IF LIMIT EXCEEDED
        SEC
        BLT     50$             ; RETURN IF NO SPACE
        MOV     (SP),R3         ; RESET R3 TO NODES DESIRED
        ASH     #4,R3           ; CONVERT SIZE TO BYTES
        .INHØ
        TST     MPFLAG          ; HAS SEND/RECEIVE POOL BEEN MAPPED?
        BEQ     1Ø$             ; EQ IF NO
        MOV     .TPDBA,R4       ; GET ADDRESS OF SEND/REC TPD
        ADD     #T.RF,R4        ; POINT TO LISTHEAD
        CALL    ..RQCB          ; TRY TO GET NODE
        BCC     3Ø$             ; NODE FOUND ? , CONTINUE
1Ø$:    MOV     #.POLLH,R4      ; TRY AGAIN FROM SCOMM
        CALL    ..RQCB
        BCC     3Ø$
        CLR     R4              ; NO - THEN CLEAR R4
3Ø$:    CALL    ..ENBØ
        TST     R4
        BNE     4Ø$
        SEC                     ; SET ERROR RETURN
        BR      5Ø$             ;
4Ø$:
        ADD     (SP),S.PU(R1)   ; INCREASE USAGE COUNT
        MOV     R1,N.AW(R4)     ; SET UP NODE ACCOUNTING WORD
        MOV     R4,R1           ; POINT R1 AT PICKED NODE
5Ø$:    MOV     (SP)+,R3
        MOV     (SP)+,R4
        RETURN
;
.SBTTL  MAPCHK - MAPping CHecK routine
;+ This routine is called to determine whether the alternative
;  node pool is in place so that ..MPSP can map to it or not
;  as required.
;
;       Output conditions
;
;               CC-C    Set if send/receive node pool not in place
;               CC-C    Clear if send/receive node pool there
```

```
;               MPFLAG  Incremented if send/receive pool in place
;
;-
MAPCHK:
        TST     MPFLAG          ; ALREADY USED SEND/REC POOL?
        BNE     30$             ;++CC-CLEARED++ NE IF YES
        MOV     R4,-(SP)        ; SAVE R4
        MOV     .TPDBA,R4       ; GET ADDRESS OF FIRST
        CMP     T.PN(R4),PARNAM ; TPD ENTRY, CHECK ITS NAME
        BNE     10$
        CMP     T.PN+2(R4),PARNAM+2
        BNE     10$
        TST     T.RF(R4)        ; CHECK LISTHEAD NON ZERO
        BEQ     10$
        INC     MPFLAG          ; SAY SEND/REC POOL IN PLACE
        CLC                     ; POOL IN USE
        BR      20$
10$:    SEC                     ; SAY POOL NOT BEING USED
20$:    MOV     (SP)+,R4        ;
30$:    RETURN
;
PARNAM: .RAD50  /SENPAR/        ; PARTITION NAME
        .END
```

[311,15]ESRNMS.MAC

```
        .TITLE  ESRNMS  -- RESIDENT EXECUTIVE
        .SBTTL  ..RNMS  -- RETURN NODE FOR SEND/RECEIVE POOL OPTION
        .IDENT  /3.1SR/
        .PSECT  EXEC
;
;
;
; COPYRIGHT (C) 1982
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
; EQUIPMENT CORPORATION.
;
;
;
;       AUTHOR: IAS Development Team
;       DATE:   13-Jul-82
;
;
        .MACRO  .INH0
        MOV     PS.EXP,-(SP)
        BIS     #140,PS.EXP
        .ENDM
;
.SBTTL  ..RNMS  Return node to send/receive partition
;
;       Based on whether the node address is in SCOM or in
;       the send/receive partition pool this routine returns
```

```
;       the node to the correct pool
;
;       It is based on SCOMM routine ..RNTV
;       and its parameter interface is identical
;
;-

..RNMS::
        MOV     N.AW(R1),-(SP)   ; DO NODE  ACCOUNTING
        ADD     #S.PU,(SP)       ; POINT TO POOL UTILIZATION COUNT
        SUB     R3,@(SP)+        ; DECREASE COUNT BY LENGTH OF NODE
        MOV     R4,-(SP)
        MOV     R3,-(SP)
        CMP     #100000,R1       ; IS NODE IN SCOM?
        BLOS    10$              ; LOS IF YES
        MOV     .TPDBA,R4        ; ADDRESS OF TPD ENTRY FOR POOL
        ADD     #T.RF,R4         ; POINT TO LISTHEAD FOR POOL
        BR      20$
10$:    MOV     #.POLLH,R4       ; LISTHEAD FOR SCOM POOL
20$:
        ASH     #4,R3
        .INH0
        CALL    ..RLCB
        CALL    ..ENB0
        MOV     (SP)+,R3
        MOV     (SP)+,R4
        RETURN
        .END
```

The patches to SGN1:

[11,17]PARGN.PAT

```
        .TITLE  PARGN
        .IDENT  /V3.1SR/
;+
; This patch will initialize the partition "SENPAR"
; as a pool and then write it to disk as all  zeroes.
; The  first two  words are set to indicate:
;
; a) That the first block is the last in the pool
; b) This block fills the whole pool.
;
;-
SN.DPL=4000
.BLK.=.
.=.BLK.+70
        CALL    PAT01
        NOP
.PSECT  ZZPAT
PAT01:
        CALL    SPGEN
        BITB    #TF.UC,T$FW(R4)
        RETURN
SPGEN:
        CMP     (R4),#$LHPAR    ; MUST BE FIRST IN TPD
        BNE     40$
        BIT     #SN.DPL,$SYSFW  ;SHOULD IT BE THERE?
        BEQ     40$             ; IF EQ, NO EVERYTHING IS OK
        MOV     R4,-(SP)        ; PRESERVE CONTEXT
        MOV     R3,-(SP)
        MOV     R2,-(SP)
        MOV     R1,-(SP)
        MOV     R0,-(SP)
        MOV     T$BA(R4),R2     ; LOAD BASE ADDRESS
        CALL    $POSWT          ; POSITION WRITE POINTER
        CALL    $CLRBF          ; CLEAR OUTPUT BUFFER
        MOV     #$LNBUF,R1      ; LOAD BUFFER ADDRESS
        MOV     T$PZ(R4),R0     ; LOAD PARTITION SIZE
        ASH     #6,R0           ; CONVERT TO BYTES
        CLR     (R1)            ; SET BLOCK TO LAST
        MOV     R0,2(R1)        ; SET SIZE TO POOL SIZE
        MOV     #1000,R2
10$:
        CMP     R2,R0           ; SET TO REMAINDER
        BLOS    20$
        MOV     R0,R2
20$:
        CALL    $FILWT          ; WRITE BUFFER
        CALL    $CLRBF
        SUB     R2,R0
        BNE     20$
30$:
```

```
        MOV     (SP)+,RØ        ; RESTORE CONTEXT
        MOV     (SP)+,R1
        MOV     (SP)+,R2
        MOV     (SP)+,R3
        MOV     (SP)+,R4
40$:    RETURN
        .END
```

[11,17]TPDGN.PAT

```
        .TITLE  TPDGN
        .IDENT  /V3.1SR/
;+
; This patch to the TPD generation module of SYSGEN
; phase 1 will check if a partition "SENPAR" has
; been declared first in the partition list and, if so,
; that its size is less than or equal to 4 K words
; Four spare words in the entry are reused to hold:
;
; a) The pool listhead                    (T.RF)
; b) The dummy back link                  (T.RB)
; c) The PDR value to map the partition   (T.CF)
; d) The PAR value                        (T.CB)
;
;-

SN.DPL=4000

.BLK.=.
.=.BLK.+1422
        JMP     PATØ1
        .PSECT  ZZPAT1
PATØ1:  CLR     T$HZ(R4)

        CMP     (R4),#$LHPAR    ; CHECK THIS IS FIRST IN TPD
        BNE     30$
        CMP     T$PN(R4),PARNAM ; CHECK PARTITION NAME
        BNE     30$
        CMP     T$PN+2(R4),PARNAM+2
        BNE     30$
        TST     T$PZ(R4)        ; CHECK PARTITION SIZE
        BEQ     40$
        CMP     #128.,T$PZ
        BLO     40$
        BIS     #SN.DPL,$SYSFW  ; SAY WE HAVE SENPAR
        MOV     #60000,T.RF(R1) ; SETUP LISTHEAD , ON APR 3
        CLR     T.RB(R1)        ; CLEAR BACKWARD POINTER
        MOV     T.BA(R1),T.CB(R1)
        MOV     T.PZ(R1),T.CF(R1)
        DEC     T.CF(R1)
        SWAB    T.CF(R1)
        BIC     #100377,T.CF(R1)
        BIS     #000006,T.CF(R1)
30$:
        RETURN
```

```
40$:
        MOV     (PC)+,R1
        .BYTE   E$R12,S$V2      ; REUSE ERROR MESSAGE
                                ; "INSUFFICENT POOL"
        CALL    $ERMSG

PARNAM: .RAD50  /SENPAR/        ; PARTITION NAME

        .END
```

The patch to REMove:

[11,17]REMOV.PAT

```
        .TITLE  REMOV
        .IDENT  /V3.1SR/
        .ENABL  LC
;+
;       When a task is being REMoved from the STD , the entries in the
;       Send data and Send by Reference queues are flushed. This means
;       that the new partition must be mapped onto and the modified
;       node picking routines must be called. This patch is designed
;       to work whether the new pool is present or not
;
;-
.BLK.=.
.=.BLK.+10276
        CALL    PAT01
        CALL    PAT02
        NOP
        NOP
.=.BLK.+10342
        CALL    ..RNMS
.=.BLK.+10752
FLSRR:

        .PSECT  ZZPAT
MPFLAG: .WORD   0               ; MAPPING FLAG
PDRSAV: .WORD   0               ; OLD PDR VALUE
PARSAV: .WORD   0               ; OLD PAR VALUE
PAT01:
        MOV     S.RF(R2),R4     ; GET A RECEIVE NODE
        CLR     MPFLAG          ; ASSUME DON'T MAP SENPAR
        RETURN

PAT02:  CMP     @S.RB(R2),S.RF(R2) ; NULL LIST YET?
        BEQ     20$             ; IF YES, FINISH UP
        TST     MPFLAG          ; SENPAR MAPPED?
        BNE     10$             ; YES, RETURN
        CMP     #100000,R4      ; EXAMINE NODE ADDRESS , IF IT EXISTS
        BLOS    10$             ; JUMP IF NODE IS IN SCOMM
        MOV     .TPDBA,R1       ; LOAD R1 WITH ADRS. OF FIRST TPD ENTRY
        MOV     T.CF(R1),-(SP)  ; STACK PDR VALUE OF PARTITION
```

```
        MOV     T.CB(R1),-(SP)  ; STACK PAR VALUE
        CALL    ..SPD3          ; REMAP USER APR 3 TEMPORARILY
        MOV     (SP)+,PARSAV    ; SAVE PAR VALUE
        MOV     (SP)+,PDRSAV    ; SAVE PAR
        MOV     #1,MPFLAG       ; INDICATE THAT STACK HOLDS OLD APR VALUES
10$:
        RETURN                  ; GO BACK TO REMOV

20$:
        TST     MPFLAG          ; SEE IF MAPPING TOOK PLACE
        BEQ     30$             ; BRANCH IF IT DIDN'T
        CLR     MPFLAG          ; RESET MAPPING FLAG
        MOV     PDRSAV,-(SP)    ; PREPARE TO REMAP
        MOV     PARSAV,-(SP)
        CALL    ..SPD3          ; REMAP USING SAVED PARAMETERS
        CMP     (SP)+,(SP)+     ; TIDY UP THE STACK
30$:
        TST     (SP)+           ; POP THE RETURN ADDRESS
        JMP     .BLK.+10352     ; AND JUMP BACK

        .END
```

[11,15]EXECODL.COR

```
[11,15]EXEC.ODL/-AU=[11,15]EXEC.ODL
-/ESMISC:/,.
ESMISC: .FCTR   [11,1]5EXEC/LB:ESMISC:ESPOOL:ESRNMS
/
```

[11,13]REMTKBCMD.COR

```
[11,13]REMTKB.CMD/-AU=[11,13]REMTKB.CMD
-/[11,11]TKB/,.
[11,11]TKB/LB:SCVTR,[11,15]EXEC/LB:ESRNMS,[1,1]EXEC.STB/SS
/
```

The command files used to apply the patches:

NEWEXEC.CMD

```
;
; This command file will prepare the system and build the
; required new executive
;
;
; To have the capability to go back to the original system
; save the following files:
;
;       [11,17]EXEC.TSK
;       [11,17]EXECUTIVE.STB
;       [11,1]SGN1.TSK
;
; Patch and rebuild the new executive
;
@EXEC
;
; Patch and rebuild Sysgen Phase 1 task ( SGN1 )
;
@SGN1
;
; Patch and rebuild the priviliged task "REMove"
;
@REMOV
;
; Now perform a  system generation using the new SGN1  and
; EXEC tasks ensuring that if the partition SENPAR is used
; it is the first partition declared and is  4K  words  or
; less in size
;
;
```

EXEC.CMD

```
.IFNINS ...SLP INS [11,1]SLP
.IFNINS ...PAT INS [11,1]PAT
MAC [11,15]ESMISC.POB=[11,15]ESMISC.PAT
MAC [11,15]ESDIR.POB=[11,15]ESDIR.PAT
MAC [11,15]DMSAR.POB=[11,15]DMSAR.PAT
MAC [11,15]ESPOOL=[311,15]ESPOOL
MAC [11,15]ESRNMS=[311,15]ESRNMS
LBR [11,15]ESMISC.OBJ;1=[11,15]EXEC.OLB;1/EX:ESMISC
LBR [11,15]DMSAR.OBJ;1=[11,15]EXEC.OLB;1/EX:DMSAR
LBR [11,15]ESDIR.OBJ;1=[11,15]EXEC.OLB;1/EX:ESDIR
PAT [11,15]ESMISC.OBJ;2=[11,15]ESMISC.OBJ;1,ESMISC.POB
PAT [11,15]DMSAR.OBJ;2=[11,15]DMSAR.OBJ;1,DMSAR.POB
PAT [11,15]ESDIR.OBJ;2=[11,15]ESDIR.OBJ;1,ESDIR.POB
LBR [11,15]EXEC.OLB/RP/-EP=[11,15]ESMISC;2
LBR [11,15]EXEC.OLB/RP/-EP=[11,15]DMSAR;2
LBR [11,15]EXEC.OLB/RP/-EP=[11,15]ESDIR;2
LBR [11,15]EXEC.OLB/IN/-EP=[11,15]ESPOOL
LBR [11,15]EXEC.OLB/IN/-EP=[11,15]ESRNMS
```

SCOMM NODE POOL PROBLEMS ON IAS

```
SLP @[11,15]EXECODL.COR
TKB @[11,15]EXECBLD
```

SGN1.CMD

```
MAC [11,17]TPDGN.POB=[11,17]TPDGN.PAT
LBR [11,17]TPDGN.OBJ;1=[11,17]SG1LIB.OLB;1/EX:TPDGN
PAT [11,17]TPDGN.OBJ;2=[11,17]TPDGN;1,TPDGN.POB
LBR [11,17]SG1LIB;2/RP/-EP=[11,17]TPDGN;2
MAC [11,17]PARGN.POB=[11,17]PARGN.PAT
LBR [11,17]PARGN.OBJ;1=[11,17]SG1LIB.OLB;1/EX:PARGN
PAT [11,17]PARGN.OBJ;2=[11,17]PARGN;1,PARGN.POB
LBR [11,17]SG1LIB;2/RP/-EP=[11,17]PARGN;2
TKB @[11,17]SG1BLD
```

REMOV.CMD

```
SLP @[11,13]REMTKBCMD.COR
MAC [11,13]REMOV.POB=[11,13]REMOV.PAT
LBR [11,13]REMOV.OBJ;1=[11,13]LIB13.OLB;1/EX:REMOV
PAT [11,13]REMOV.OBJ;2=[11,13]REMOV.OBJ;1,REMOV.POB
LBR [11,13]LIB13.OLB;2/RP/-EP=[11,13]REMOV.OBJ;2
TKB @[11,13]REMTKB
```

DeVIAS Questions - Answers

Name: _____

Mailing
Address: _____
_____
_____

Phone
Number: ( ) _____-_____

Instructions

1) Complete relevant part of form
2) Mail to Editor, The DeVIAS Letter
3) Question and/or Answer will be published in next newsletter.

Date Submitted: _____/_____/_____

Question:

Answer:

DECUS SUBSCRIPTION SERVICE
DIGITAL EQUIPMENT COMPUTER USERS SOCIETY
ONE IRON WAY, MRO2-1/C11
MARLBORO, MASSACHUSETTS 01752

Bulk Rate
U.S. Postage
PAID
Fitchburg, MA
Permit No.
21

MOVING OR REPLACING A DELEGATE?

Please notify us immediately to guarantee continuing receipt of DECUS literature. Allow up to six weeks for change to take effect.

( ) Change of Address
( ) Delegate Replacement

DECUS Membership No.: _____

Name: _____

Company: _____

Address: _____

_____

State/Country: _____

Zip/Postal Code: _____

Mail to: DECUS - ATT: Subscription Service
One Iron Way, MRO2-1/C11
Marlboro, Massachusetts 01752 USA