Richard Harper's Private 64/20 z/OS Assembly Language Development Platform

Vendor Developers
Do It With a DUCT

Version 1.4

Preface

Welcome to my world.

What I'm about to show you in the pages that follow is my Personal 64/20 z/OS Assembly Language Development Platform. The 64 in the name refers to having full support for 64-bit z/Architecture. And the 20 represents the 20-Bit displacement instruction set with the 'Y' and 'G' instructions.

Those 20-bit instructions are awesomely powerful in that they allow for direct addressability of one megabyte of storage. So, if we limit our coding to these instructions we can have 256 base registers for the price of one. To support this reality, this system includes the SSY macros to be used as a replacement for the SS instruction set, which cannot be converted to 20-bit instructions due to the limitations of the hardware. It would require 8-byte instructions and the hardware won't support that, because the PSW cannot represent an 8-byte instruction in the ILC, the Instruction Length Code.

There are many features to this system, including Stacking DSA for your below-the-line storage, Stacking ATB for your above-the-bar storage, and label level tracing is built into the system, which makes debugging and maintenance a breeze. These traces support multiple TCBs with a separate SYSOUT dataset being dynamically allocated for each task in the address space. And full support for programs running in ARMODE is also provided, that allows the display of data in other address spaces or dataspaces.

So, it's a pretty robust system. And it's the only true 64/20 Development Platform in the world. I know, because I had to write GETMAIN and FREEMAIN for above-the-bar storage, and you can't have a truly effective 64/20 system without that. In ten years IBM has failed to produce that result, so I had to do it myself. It's part of this system.

I hope you will find it as useful as I do.

Now, if you wish to get a ten minute overview of how this system works and what it produces, you might want to start with Appendix C. In that Appendix I provide three programs that use these traces, along with the trace output that is produced. As they say, 'a picture is worth a thousand words'. So that's why I included them in this document, so you don't have to install the system to understand what I have here.

These are three of the 30+ programs you will find in the /TST directory of the .zip file. And they demonstrate some of the capabilities of this system. The first shows traces for programs running in 64-bit AMODE and the ABEND diagnostics. The second shows support for programs running in ARMODE. And the third shows that I can get these same traces within a user SVC.

So, if you want a quick overview, I suggest you visit Appendix C.

Contents

Introduction

A long time ago I was working for Phoenix Software Company developing an SDSF replacement for CONDOR which was in itself a replacement for ISPF and similar in functionality to ROSCOE, though better, I've used both of these systems. I know.

I had a cool window office on the tenth floor that looked down on Century Boulevard in Los Angeles just outside the Airport, LAX. I remember working there and looking out my window when one day a bunch of heavy equipment rolled onto this empty lot next door and began to dig a hole in preparation for putting up a new building. It seems like they worked on the hole for months. They dug down something like fifty feet for about a whole city block. And then they built some forms and brought in the cement trucks, poured perhaps thirty foundation stone cubes ten feet on a side, at least. Then they filled in the dirt around the stones, packed it down and laid down a cement slab around the whole place forty feet below ground level. They were building a parking garage. And then they slowly went up one floor at a time, until they finally reached the surface.

The foundation for that building which was to be twenty stories high took those people six or eight months to build. But at that point, they stopped pouring concrete and started laying down a steel framework. It was amazing to see, both in how long it took to lay the foundation, but then also how fast that building went up once the foundation had been laid and they switched to steel. It seemed that this building grew another floor every few days. It's amazing to see what can be done when you build a solid foundation.

Now, the point of this little parable is that this is a universal truth that applies equally well to a Software Company, as it does to a Construction Crew. When you build a solid foundation, you can write code very, *very* quickly.

The problem I've seen in the six major software development companies that I've worked for, is that they will rarely spend the money on infrastructure. In my mind, they're way too focused on the next quarter's profits, and building out the infrastructure costs money up front and does not provide an immediate payoff.

So, the propensity in many companies is to throw something together and push their people hard to get a deliverable out the door, as quickly as possible. And so, we often end up with hundreds of thousands of lines of spaghetti code that is fragile, breaks easily, has a high PE ratio, and often times fails to deliver for the customer. And I assert that this is because the management failed to authorize the expense of building out a proper foundation. So, they end up paying the cost of the foundation many times over on the back end, as the cost of maintenance.

Another cost for failing to build a proper foundation, is that the life of the product is shortened dramatically. Because in a software company many different people will work on a given product and they come and go over time, and as fixes go into the product, written by people who don't really understand all the reasons and why-for's, the code inevitably become less and less comprehensible. This is just the natural progression of a software product. So, at some point it will become totally incomprehensible, as demonstrated by the PE ratio, and will eventually either have to be retired or rewritten.

However, with a solid foundation that provides a robust diagnostic capability and a good design that highlights maintainability over performance, the life of the product can be extended significantly, perhaps even two or three times longer.

I've been around the block a few times and have seen this happen many times. It's not always true, though. Boole & Babbage did a really great job on BBI3, the foundation for CMF, but it took five people five years to do it. And David Hempstead and I did a fine job replacing the foundation for DMS getting rid of the MI interface and putting in a tracing facility similar to this, but we had to do it on the sly, under the guise of providing SMPE support.

So, this is what I like to do. I'm an Infrastructural Architect, it's what I do. And I think I have built a pretty solid foundation here. I also think I've gotten pretty good at it by now. This is the third time I've written a system like this. And this is my best work.

When I left Serena, I was replaced by a Level II person, who was promoted into that position to maintain the code that I had written. Oops, I guess I had written it too well, as my code had a diagnostic facility similar to this one, which made my code easy to maintain by someone much cheaper than I.

It's never pleasant to get laid off, and I'm sure that many people reading this have had that experience. But the worst thing about being laid off from Serena was that I lost my tools. I had written a *wonderful* Label Level Tracing Facility, as I called it then, and it made me twice as productive as any other developer in the world. And I had to leave it behind, because I did it as "work for hire" and they owned it, not I. It was like being a carpenter and having someone take away your level and square. I was devastated at the loss of my tools.

So, when I left Serena I picked up a copy of Hercules, and I spent nearly six months between jobs writing this replacement system. I started out with the Assembler F, because that was the only thing available on MVS 3.8J. But then I found the Tachyon Legacy Assembler, which was free and is basically a High Level Assembler that will run on MVS 3.8J, a.k.a. MVS/SP. Thank you, David Bond. I will be forever in your debt.

So, it's mine and I own it. It's protected by the GNU General Public License. It was written by me, for me, and to the benefit of me. And never again will I be left without my tools.

My Expectations for This System?

1) A 50-60 Percent Reduction in the Development Costs for New Code

This is because most of the basic work that all new programs require has already been done for you and is provided in a robust series of macros. You don't have to worry about reentrancy, you have stacking DSA and ATB storage, all the program entry and exit stuff is done for you, and you have the traces built into the system which should cut debugging time in half or more, depending upon the competence level of the developer writing the code.

2) A 50-60 Percent Reduction in the Cost of Maintenance of the Code

This is because the tracing facility is built right into the system and is integral, they're always there and available to easily diagnose any new problems discovered in the code over the life of the product.

When diagnosing problems, it's always best if you can recreate the issue in-house. In that case you can add new #TRACE macros at any point in the code to see what's happening at any given point. And new trace points can be added by simply reassembling and relinking the system. Then run the test again.

However, there are always times when only the customer can recreate the problem. But since the tracing SVC can be installed at the customer's site, these diagnostics can be run at their site also. Often times, the program entry and exit trace points, with the subroutine entry and exit points and the #TRACE macros that are dormant within the system and activated only by the existence of the TRCPRINT DD, will be sufficient to diagnose an issue at the customer's site.

If you need to go a little deeper, you can also do as I have done. Recompile a program with new trace points, ship the object module to the customer via e-mail, ask them to relink and recreate the test. Resolving customer issues can happen very quickly with this method.

3) A 200-300 Percent in Increase the Longevity of the Code.

Now, of course, this depends upon the competence level of the developers maintaining the code.

I was talking to another Developer some time ago about the IF/THEN/ELSE constructs that I use to write code these days where I depend upon ASMMSP. He said that he had seen such code with seven or eight levels of indention and thought these constructs were disgusting. My reply was that "You can write spaghetti code in any language."

I guess the point here is that if you have good people maintaining the code, the code should live much longer with a foundation such as this that provides a robust diagnostic facility built into the system but dormant until required for diagnosis.

So, Why Am I Distributing This Now?

This is my personal Development Platform, written of me, by me, and for the benefit of me. So why would I just give it away? I have two reasons for doing so.

1) Because I'm looking for another job.

I was most recently working for IBM doing defect support for JES2 level II. The work wasn't bad and I found some very interesting bugs during that time, one that hadn't been found in twenty years. But they hired me as a contractor, without medical or dental insurance, three vacation days a year and at a salary that was effectively a 40% cut from what I was making at Compuware. And I want my next job to be something better.

Given the job market in this country at present, I need some way to get my work under the nose of the Hiring Manager. This is one impressive piece of work and if I can get it in the hands of the right person, well, as they say: "By there works, shall ye know them."

So, if you happen to receive a copy of this platform and you're not a Hiring Manager, please pass it along to your Boss with your comments.

2) To Gain a Little Respect from My Peers

This is an incredibly powerful system for the development of Assembly Language programs and systems. There are not very many people on this planet that know about this stuff or work at this level. And by that I mean people who write system level code in Assembler for z/OS. So, we're a pretty elite group, those of us who can do this work at this level. And anyone who takes an interest in this platform is likely to be a peer or close to it.

I'm sure that once you begin to play with this system and see what can be done with it, how fast you can produce the result, and how easy it is to debug your code, you will be impressed and appreciative.

Now, this code is subject to the GNU General Public license. You are free to give it away to anyone in the world that you wish. It's "Free Software" and I'd like to see it in the hands of every Assembler Developer on this planet, every Systems Programmer and every college student in every college in the world that teaches this stuff; z/OS Assembler. This pretty much means the colleges in Prague, Moscow, Bangalore, and Peking. It seems like they don't teach this stuff in America anymore, and that's a great sadness for me. Anyway, please do. Give it away, far and wide to anyone who wants or has the ability to make use of it, or can understand what I have here.

What You Can and Cannot Do With This System

You have to understand that this system, as distributed is subject to the GNU-GPL license, which means that it cannot be used to write or develop commercial code, which is to say code intended for sale or production use by any company. You can't do that, the GNU-GPL precludes that, which is why I distributed it this way.

What you can't do.

This is a macro based system, and as such anything you write using this platform will inherently include code that I have written which is subject to GNU-GPL and this makes anything that you write using this platform also subject to GNU-GPL. And what that means is that anything you write using this platform cannot be sold, it must be given away for free, as Free Software. Well, that might not be quite accurate. You can sell it, but anyone you sell it to can give it away for free to anyone, because it's Free Software by definition. Understand?

But even more than that, as I understand the GNU-GPL license, nothing that you write using this platform can be linked-in or included into any commercial application, because if it is, the entire application becomes Free Software. So, for that reason it would be very unwise to do so.

Another thing that you can't do, well it's not that you can't do, but it would be very unwise to do, is to write any production code with this system. You see, anything written with the GNU-GPL version of this code is by definition "Free Software." So, if you're working for a bank and write something cool like an Investment Portfolio System, which I did once long ago, and use this system to do it? Well, it's free software. So any employee can take that code and give it away to any competitor for free. Understand?

So, you really don't want use this system for production code. In that event the company needs to purchase a Commercial License from me or my representative, which we'll talk about later.

What you can do.

Now that said, there are a lot of reasons to use this platform. Any code that is not for sale or for production use is not a problem. You can do whatever you want as long as you're not intending to sell what you write with this platform. So, let's talk about what you can do.

You can write any kind of test program that you like, and we all need to write test programs to test out our production code. I'm distributing some 30+ such programs with this system in the /TST library.

When I was working with JES2, that group had two people where there whole job was to write test programs to test out new interfaces placed into the component, such as SAPI, Extended Status, and the like. Such testers can certainly use this system to develop these kinds of programs, since they were never intended for sale in the first place. And this is true for any company and any developer anywhere. Yes, it's Free Software, but who cares, it's a bloody test program.

And this is a sanctioned use for the GNU-GPL version of this system. You, as the developer will likely find yourself many times more productive than the developer sitting in the cubical next to you, if you learn this system and use it. And in todays job market that can only work to your advantage, as it will benefit you and your company.

The other really important use for this system is for your own education or edification. We all have to remain current in our knowledge and understanding of the z/OS Operating System. It's our job and continuing education is a part of that job. Every few years IBM comes out with a new version of POPs (Principals of Operations), and every time a new version comes out you need to read it, to see what's new or what has changed. They're constantly adding new instructions.

But this is a system that allows you to test out new instructions where you can use the #TRACE macro to get a clear and distinct understanding of how these new instructions work. But it's not just new instructions. If you haven't done an MGCR, write a test program that uses it. If you haven't written an SRB, do so. If you haven't written a PC, write one, and use this system to do it. You will learn more quickly for it. Ongoing education is important, and something you need to do committedly if you're working in this field. So, this is another legitimate and sanctioned use of this system.

The Importance of Creative Control

I was at IBM in Poughkeepsie last year 2009, and this is the site where they actually make the z/OS Z10 computer. It's not just software people that work at this site. It has manufacturing and hardware development people too, as well as, all kinds of support and administrative staff on that campus. It's a big campus.

Anyway, I met this man one day who was new to the company. He was a physicist working on this thing he called nano-meter technology, which I guess had to do with making the chips smaller and smaller. This is not my field. We had a conversation where he was carping about deadlines and all the meetings he was forced to attend, when what he really wanted to do was just sit at his desk and work. I looked at him and said, "That's the problem with IBM. Nobody has creative control." He looked at me a little shocked, tilted his head in thought and then replied, "You know, you're absolutely right."

I had seen this in the software area where I worked, but it turned out to be true in the hardware part of the company too. IBM does everything by committee which is about as efficient as the U.S. Congress, and this is why it costs IBM three or four times as much money to develop anything, as it does in a Vendor Shop.

Now realistically, few people get the opportunity to have Creative Control in their careers. Every company that you're likely to work for has their Policies and Procedures, and you have to work within their rules. But some companies are more stringent than others. When I was working for Phoenix Software International, I had no restrictions. There were only two Developers in the shop, me and Fred. When working at Compuware Strobe, we gave lip service to ISO9000, someone would look over someone else's code and sign off on it, so the paperwork was there for government contracts. But at IBM for JES2 they took ISO9000 very seriously, every change to JES2 was reviewed by seven or more people. And that's probably appropriate for JES2, since it is such a mission critical product. You don't want to mess it up. But even so, they have had a lot of PE's on JES2, especially in the SAPI interface.

I have been lucky in my career. In most shops in which I have worked I have had Creative Control, and that folks is more important than I can begin to express. So, I'm going to tell you three stories where I had such control and how that affected my performance.

Please understand that part of the reason I'm shipping this code is to get myself a new job, so this whole document is really an extension of my resume. And so, for this reason, telling war stories is not inappropriate here. I can tell a hundred war stories, but these three stories are intended help to illustrate the point that I wish to make, the importance of having Creative Control.

Phoenix Software - 1983

I remember when I went through the interview for this job. I had always wanted to write Operating Systems and Compilers since I first learned to program in High School, and this was my first opportunity to work at that level for real. I had been a Sysprog in the Marines and worked on the retrofit of HASP/NJI to allow it to run on the S/360. But I had never been a "Developer." I remember going home after the interview, understanding that he needed someone to write a replacement for SDSF, and I was excited, *extremely* excited.

So, I got bold. I called up Fred Hochette after I arrived home and said to him, "Fred I really, really, *really* want this job." He laughed over the phone and then with a voice of sanity said, "I get that you really want the job, but I have two more people to interview. I'll let you know by Friday." Cool, that extra push may very well have gotten me the opportunity. I started the next Monday.

Fred didn't really believe in me at first. In his mind, I was a kid at 24. So he said to me, "Go write something, anything, I want to see what you can do."

"Hmm, well that's interesting. Let me think about this for a moment." I replied, and retreated to my brand new desk. I sat there and thought for about an hour. If I could write anything I want, what would it be? How could I impress Fred Hochette? When I was sixteen years old I had written my first disassembler. It was for the PDP/11 a DEC machine we had in High School. I didn't want to write throw-away code, but something that might be useful in the future. So, that's what I decided to do, when I walked back into Fred's office.

"Fred," I said, "I think I'd like to write a disassembler."

"A disassembler, eh?" He said, obviously impressed.

"Yes. If I'm going to write something, I want to do something that might be useful in the future. And with IBM going more and more towards OCO, a disassembler might be very useful tool for the future."

"Okay," he said, thoughtfully. "Let's see what you can do."

Two weeks later I had about five thousand lines of code and a pretty good online interactive disassembler running under CONDOR. It wasn't quite complete, as it hadn't been fully debugged, but it was pretty damn good.

At that point Fred came into my office and said, "Okay. I see that you can write code. I have another job for you." Then he proceeded to tell me that he needed a replacement for SDSF that would run under CONDOR in MVS. This was the real job I was hired on to do.

CONDOR was a product that was originally written to run under DOS. You can think of it as MFT running in an MVS address space. He was losing customers that were moving to MVS and he wanted to keep them. So, he needed an MVS version of CONDOR, and the biggest missing piece was SDSF.

CONDOR is incredible and still available today. It can handle 6,000 users for about the same CPU resources as it takes to handle 100 TSO users. It's an awesome product and provides an incredible savings in software charges for most companies. It provides everything that any average programmer needs to do their job. And while Sysprogs need ISPF, most programmers don't. CONDOR works just fine for the average programmer. And it has a much better editor than ISPF.

Anyway, he showed me what he had done, using the PSO interface to JES2 and asked me to take a look at it. I spent the rest of that afternoon reading the manuals about PSO, and looking over Fred's code. PSO was a piece of crap, and I realized we would never be able to replicate the full functionality of SDSF using that interface. I went home and slept on it.

The next morning I went into Fred's office to deliver the bad news. I said, "Fred, this project you're asking me to do is bigger than a bread box. There is no way in hell that this PSO interface is going to give us what we need. We're going to have to go directly to the Spool Packs, and dig into JES2's address space, if we're going to do this job. It's a piece of work and it's going to take some time."

"Can you do it?" He inquired.

"Yes, I can do it. But it will take time." I replied, emphatically.

He looked at me then quizzically, "Okay, how long is it going to take?"
I cringed, fearful of having to deliver the bad news. "Six months, maybe a year." I said, it was my best estimate, but I was worried about how that information might be received. I had only been on the job for like two weeks.
He leaned back in his chair, put his thumb and forefinger to his chin, as if thinking, then replied, "Six months, maybe a year, uh?"
"Probably a year, Fred." I sighed, in response.
"Probably a year." He nodded. Then he sat upright, put both arms on the desk, looked me square in the eye, and said, "Okay, go do it."
I was flabbergasted. In that moment, Fred had given me the one thing every developer wants more than anything else in the world; Carte Blanch Authority and Creative Control. So, I got to work. Six months later I had produced a 1.0 version. Then we sent it out to the customer base and it broke in many different ways. Oops. They were on XA and ESA and I had developed this on SP. And of course, customers always do things you would never expect. But within a year though, it was pretty stable. I got my job done.
If this had been IBM it would have taken five people five years to produce the same result. I was able to do it, one person in one year, because I had Creative Control.

Sterling Software - 1996

Sterling Williams was a man that created a business plan based on mergers and acquisitions. His first acquisition was Advanced Data Systems which he acquired for one million dollars, because the company could not make payroll. This company had a product at that time called DMS the only real competitor for IBM's HSM. In 1996, I was co-architect for that product with David Hempstead, then called SAMS:Disk.
Sterling was a great businessman, no question about it. He took a one million dollar investment and turned it into a business that generated six hundred million dollars per year, and he did it in six years, before selling out to Computer Associates. I didn't like the man, but he knew how to run a software company, for which I am grateful.
In the Fall of 1996, Sterling had just acquired another company and, as they always did during a new acquisition, they had a layoff to trim the fluff. The result in our LAB was that we lost our Team Lead for SAMS:Disk. At the same time we, the developers had a requirement to make SAMS:Disk SMPE installable. This was not an easy project. The product had not been completely reassembled for the last ten years, and we were shipping an IEBCOPY version at that time.
The manager of the LAB, a boy named Norm Patriquin, called David and I into his office and said something like "I don't care what you have to do, just get it done. And I want it done by the end of the year." He was talking about SMPE installability. At that moment he gave us Carte Blanch Authority, and Creative Control. So, we picked up the ball and ran with it. And boy did we run.
This was late October. David and I both knew that we were never going get this kind of opportunity again, so we took maximum advantage of it. David had a whole binder full of all the enhancements that he would like to see implemented into the system if he ever got the chance. Well, this was our chance.

The product was in deep trouble. It had not been reassembled in many years and in order to produce an SMPE installable version, we needed to have Object Modules which we did not have at that point.

So, the next day I set out to reassemble the entire system. It was fifteen-hundred programs and over a million lines of code. So, I created a PROC and broke it down to 25 programs per job and shoved them out the door. It took some six hours to reassemble everything. Well, everything we could. We discovered that there were about 30 programs for which we did not have the source. That first attempt showed that we had about 70 other programs that would not assemble, due mostly to macro changes and blowing base registers.

One of the major changes was to remove this thing called MI which stood for Module Interface. At that time in DMS every program was a discreet program. Every time you would call a program you would call MI first, and it would pass control to that program, loading it if necessary. It had a rudimentary tracing capability built into it, which was why it was used up to that point.

We replaced it with a system similar to what I have here, though not nearly as robust. David called it PET for Program Event Tracing. Then we statically linked the system, avoiding physical loads of individual programs, which in the end gave us about a 30% increase in performance; unintended consequences.

I disassembled all the programs for which we had lost the source, and fixed all the programs that would not assemble. David wrote the #CALL macro which replaced all the IBM CALL macros throughout the system. We changed the program entry and exit macros, and installed the PET tracing facility and provided much better ABEND diagnostics.

We were working our asses off that fall, sixteen hours a day, seven days a week. This was our one shot, and we were committed to making the best of it. In the end David and I made 50,000 lines of code changes in 90 days. And that's a hell of a performance, even for me. We changed every single program in the system, all fifteen-hundred of them.

In January after the Holidays, another Developer looked at what we were doing and freaked out. We were making too many code changes, but we were also doing our job. She went to the Boss and complained.

So, David and I got called on the carpet. But the changes we made survived. In the end we were vindicated. The next June we went to a User's Conference, and somebody, I believe it was from CitiCorp stood up in a room of at least 200 customers and said that this was the most stable version of DMS that he had seen in the last fifteen years. We got a standing ovation and I glowed in that moment.

We should have been honored, have gotten a slap on the back and a nice big fat bonus for having worked our asses off, and for such a great performance. But you know what we got? We got to keep our jobs. Sometimes a job well done has to be its own reward.

Even so, it's still amazing what can be done when one has Creative Control, and no one looking over your shoulder. In that situation, you just do it, because you can.

Serena Software - 1999

I was working for Boole & Babbage in San Jose California, after I left Sterling. I had this really cool project and I was Chief Architect. I was tasked to produce this result called CMF for TCP/IP, a TCP/IP monitor.

I was happy, but my wife was not. She hated San Jose. We had a two bedroom townhouse that cost $2,200.00 per month, which I thought was ridicules for an apartment where your windows looked into another person's window 30 feet away. And she called the roadways "rivers of steel" which was not an unrealistic characterization.

Anyway, about nine months into the project, a friend who I had worked with at Sterling called me up. I don't know how he found me.

But he called me up at Boole. At the time he was the Team Lead for this product called ChangeMan for Serena Software. And he said "Richard, I need you. I have a problem with the Audit component of ChangeMan and I know you can fix it." He knew my mettle. So, I said to him, "I have this great job with Boole. I'm Chief Architect. I love what I'm doing here. I don't need another job." To which he replied, "What would it take to get you to come on board?"

So, I thought for a moment, because I really liked this man. "Well, my woman hates San Jose. If I could work from home, I might just be open to the idea, though I'd have to run it by her." Then he said, "If you can work in the office for six months and get to know the people, you can live anywhere you want in the country." I sighed, I had been hooked. "Okay, I'll let you know tomorrow. I have to talk this over with my wife." I talked it over with her that night, and she was all for it. So, I made what might have been a bad career decision.

You see, writing CMF for TCP/IP was a really incredible project dealing with routers and bridges and hubs, oh my. Fixing Audit for ChangeMan, well that was really application assembler, a project of no real challenge whatsoever. But working remote got my wife off my back and allowed us to move to Friday Harbor in Washington, and then later to Maui Hawaii for some six years. It was pretty cool, and really an easy job.

But the point I'm trying to make is about the value of having Creative Control. I knew this man and he knew me and my capabilities from our time at Sterling. So, I told him before taking the job, that if he would just keep Management off my back, I would solve this problem for him, which in the end I did. And he kept his part of that bargain, too.

So, that's how I took a 20,000 line spaghetti code program with a 30% PE ratio and turned it into a 50,000 line 53 program system which was easily maintainable by the Level II person who eventually replaced me at half my salary.

It's so important to have Creative Control, with no one looking over your shoulder telling you how to write it. That's when you can do your best work.

Point made.

The Commercial License

I recently obtained a UID for a z/OS 1.11 System. So, I can produce a Commercial version of this product, and I am doing so as we speak.

For a Commercial version, I need to add value and filtering is the most important piece, though there are several enhancements I intend to make described later in this document. Filtering will allow you to designate what programs you want to trace. It's not needed for a single program. But it is needed if you have a fifty or hundred program system. You don't want to be inundated by a bunch of trace data for programs that you don't need and are not interested in seeing.

Now, I'm not a businessman and I'm not a marketing person. I'm a Developer, I write the code. And it would be best if I could locate a sponsoring software company to market this product. I just love to write the code.

But until I can find a company that would like to sell this product and negotiate a deal, I'll have to do that myself. So, if you or your company would like to purchase a Commercial License that would allow you to write commercial, or production code using this system, not subject to the GNU-GPL, please contact me. My resume is in the /DOC directory and it contains contact information such as my e-mail address and phone number.

I think this product as it stands is worth $100,000 per copy or per site, though I'm negotiable on that point, especially for the first few customers. It would take five times that amount for a Vendor shop to develop something comparable to this system and that's assuming that they have Developers good enough to do it. And they're not many of us on this planet.

So, if you or your company is interested in a Commercial License, e-mail me and let me know.

Also, if you work for a Vendor Shop selling software, that might be interested in taking over the marketing responsibilities for this product, let's talk.

New Versions of This Code

There will be future versions of this code. There are still bugs in the system, though not too many, but no code is bug free. It's just the nature of code. But I'm committed to fixing whatever bugs I find. And those fixes will be ported down to the GNU-GPL version.

There are also certain enhancements that I feel are appropriate to port down to the GNU-GPL version, like #DD for example, or perhaps the #TPUT or the #COMPUTE macro. So, there will be new versions. I'd like to produce a new version every year, if only for the purpose of keeping up interest.

So, my intent, while I haven't done it yet, is to provide a web site where you can get the latest version of this code. That web site will be located at: www.TheWarriorsPath.com/Code/.

The Warriors Path is a web site that I have already created and is available. It's not about code, it's really about metaphysics. But it is a place where I can put the code on the web for you to download. If you're interested in metaphysics, you're welcome to check out that site.

This directory will be a place where you can report bugs, I mean I can't really find them all myself and I could use your help. If it's a real bug, I'll fix it. But it will also be also a place where you can suggest enhancements, and if they're not too difficult, I may very well put them in. It will also be a place where you can register and let me know that you're using this system. At least that's what I envision. I haven't done it yet, but it should be available by Fall 2010.

Installation Instructions

This system is distributed in source as a .zip file with five directories. These directories contain the following files:

\DOC	Contains documentation files.
\JCL	Contains the JCL necessary to install the system.
\SRC	Contains the source code for programs.
\MAC	Contains the macro library.
\TST	Contains a number of test programs

The files in these libraries will need to be uploaded via whatever method you choose from the PC and into a mainframe PDS. Your terminal emulator should be able to provide this functionality. So, I'll leave that to you.

The steps necessary to install this system on a z/OS mainframe computer are as follows.

Step 1: Catalog an Alias for an HLQ for the Mainframe Libraries

RDHDPCAT is a member in the JCL directory that contains JCL to catalog an alias into a user catalog. You will need to determine which user catalog you would like to use. The suggested HLQ to use is RDH, but you can use whatever you like.

Step 2: Allocate the Required Libraries on the Mainframe

The next thing is to allocate the required PDS libraries. There is a member in the JCL directory that will do this for you, and it is: RDHDPALO. You can allocate the libraries yourself manually, or you can upload this member and run it to allocate the PDS libraries.

Step 3: Upload All Source Members to the Mainframe PDS Libraries.

Use your favorite upload facility to upload the members of the distribution directories into their corresponding libraries on the mainframe.

/JCL	to	hlq.$TLS.JCL
/SRC	to	hlq.$TLS.SRC
/MAC	to	hlq.$TLS.MAC
/TST	to	hlq.$TLS.TST

Step 4: Assemble the Programs in the System

There is a member in the JCL library called RDHDPASM that will assemble all the programs supporting the system. It uses an inline procedure to perform this task. If the HLQ you have chosen is not RDH then you will need to change the HLQ parameter in this PROC. Change the JOBCARD as needed and submit the job. The following programs operate in support of this system:

RDHTRACE	-	This is the main tracing program.
RDHTRACG	-	This is a glue code program for the BASR implementation.
RDHTTAPI	-	This is the installer for the TRAP2 processor.
RDHTRSVC	-	This is the tracing SVC.
RDHINSTL	-	This is the SVC dynamic installation program.
RDHMSGS	-	This is a program supporting #MSG, #WTO and #WTOR.
RDHGATB	-	This program provides support #GETATB and #FREATB.
RDHGASVC	-	This is the SVC implementation for RDHGATB.

Note: This system requires SYS1.SASMMAC2 from the HLASM Toolkit to assemble properly. The JCL in RDHDPASM and the compile procedures RASMCL and RASMCLG will need to be customized if you used a different name for that library, or if you used a HLQ other than RDH.

Step 5: APF Authorize the LOD Library.

The system load library requires APF authorization to dynamically install the tracing SVC on a running system. Additionally, the BASR and TRAP2 implementations require that the programs using these features also be authorized, though the SVC implementation does not. The following operator commands can be used to dynamically APF authorize the library:

From System Console:
```
   SETPROG APF,ADD,DSNAME=hlq.$TLS.LOD,VOLUME=volser
   D PROG,APF,DSNAME=hlq.$TLS.LOD
```

Step 6: Install the Tracing SVC

Part of the assembly process above was to assemble the dynamic SVC install program RDHINSTL. There is a jobstream in the JCL library, RDHDPINS that when run will install the tracing SVC on the system. I like to use SVC 166 but you can use whatever you like.

Step 6: Install the #GETATB/#FREATB SVC

Part of the assembly process above was to assemble the dynamic SVC install program RDHINSTL. There is a jobstream in the JCL library, RDHDPATB that when run will install this SVC on the system which provides support for GETMAIN and FREEMAIN of above-the-bar storage in increments of less than one megabyte. I use SVC 167, if you wish to use another SVC number you must modify these two macros to indicate the SVC being used.

Step 8: Run a Few Programs from hlq.$TLS.TST to verify proper installation.

The final step is to run a few test programs from the hlq.$TLS.TST library to verify the installation went properly. These are the test programs that I use for regression testing and they provide a good overview of what one can do with a development platform such as this.

Users Guide

I give you a Sword and a Shield.
Use them wisely.

Richard D. Harper
Author

Chapter 1 – Program Management Macro Reference

This is a macro based system that also has a few programs that operate in support of the macros to produce the traces and the ABEND diagnostics. In addition to the basic program entry and exit macros, which are the base of the system, there are several other macros that exist simply to make writing code easier. And we'll talk about those in a separate section, so as not to confuse the matter.

The primary macros that support the diagnostic traces are as follows:

PGMNTRY	– This is your Program Entry macro.
PGMEXIT	– This is the Program Exit macro.
SVCNTRY	– This is a program entry macro supporting SVC linkage.
SVCEXIT	– This is a program exit macro that supports SVC linkage.
SUBNTRY	– Provides services for subroutines within a program.
SUBEXIT	– Provides subroutine exit linkage
#TRACE	– Generates a Trace Point trap, the most important macro.
CONBEG	– Defines the beginning of the Constant Area.
CONEND	– Defines the end of the Constant Area.
DSABEG	– Defines the beginning of the Dynamic Storage Area (DSA).
DSAEND	– Defines the end of the Dynamic Storage Area (DSA).
ATBBEG	– Defines the beginning of Above-The-Bar Storage (ATB).
ATBEND	– Defines the end of the Above-The-Bar Storage Area (ATB).

These are the primary macros that set up the environment for the system, and we will discuss each of them in detail, and how they interrelate to one another.

1.1 PGMNTRY Macro

The PGMNTRY macro initializes the environment for basically everything else in the system. It does all the normal things that one would expect from a program entry macro, such as defining the RSECT, saving registers via BAKR, establishing base registers, obtaining storage for DSA, establishing the desired AMODE and the like.

But in addition to these standard functions for a program entry macro, this one also sets up the tracing environment, provides for stacking ATB storage, establishes an ESTAE that the traces can use to print abend diagnostics in the event of an abend, and provides some simple ESTAE support. It also does a few things to support 20-bit displacements, and base-registerless code.

One of the more revolutionary things that it does is to establish a LOCTR structure for the program. These are defined in a copybook called PGMGBLAS which can be found in the macro library. The global variables for the LOCTRs are setup in PGMNTRY, which can then be used by any macro in the system that copies in PGMGBLAS to carve out any storage that may be needed in any of these areas. These areas are defined like this:

```
         GBLC  &LCSCT                  SCT LOCTR - Main CSECT
         GBLC  &LCSC1                  SCT LOCTR - Middle 1 CSECT
         GBLC  &LCSC2                  SCT LOCTR - Middle 2 CSECT
         GBLC  &LCSPG                  SPG LOCTR - SUB Programs
.*
         GBLC  &LCCON                  CON LOCTR - For Constants
         GBLC  &LCCN1                  CON LOCTR 1 For Constants
         GBLC  &LCCN2                  CON LOCTR 2 For Constants
         GBLC  &LCTRC                  TRC LOCTR - For Trace Data
.*
         GBLC  &DSDSA                  DSA DSECT - For Dynamic Storage
         GBLC  &LCDSA                  DSA LOCTR - For Dynamic Storage
         GBLC  &LCDSB                  DSA LOCTR - For Middle 1 DSA
         GBLC  &LCDSC                  DSA LOCTR - For Middle 2 DSA
         GBLC  &LCDSD                  DSA LOCTR - For Middle 3 DSA
         GBLC  &LCDSE                  DSA LOCTR - For End of DSA
.*
         GBLC  &DSATB                  ATB DSECT - Above-The-Bar Storag
         GBLC  &LCATA                  ATB LOCTR - For Dynamic Storage
         GBLC  &LCATB                  ATB LOCTR - For Middle 1 ATB
         GBLC  &LCATC                  ATB LOCTR - For Middle 2 ATB
         GBLC  &LCATD                  ATB LOCTR - For Middle 3 ATB
         GBLC  &LCATE                  ATB LOCTR - For End of ATB
```

This LOCTR structure makes it possible to write macros that carve out their own storage in any of these areas, and allows us to eliminate the need for MF=E/L constructs when writing code generating macros. This is a BIG deal.

1.1.1 PGMNTRY Syntax

The syntax for the PGMNTRY macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
PGMNTRY	Macro name.
b	One or more blanks.

IMP=SVC	Implementation: BASR/SVC/TRAP2
TYPE=SUBR,	Type of Entry MAIN/SUBR
BASE=	Program Base Registers
CBAS=	Constant Base Registers
DBAS=,	Additional Dynamic Storage Base Registers
ABAS=,	Above-The-Bar Base Register
AMODE=31,	Default Amode 31/64
RMODE=ANY,	Default Rmode
ASCENV=P,	P/AR - Access Register Mode
ARCHLVL=2,	0/1/2 - Zos Macro Level=2
KEY=8,	Storage Key
SP=0,	Subpool
LEN=4,	Length for DSA Getmain
DSANAM=,	Short Name for DSA-If Desired
PARM=,	Input Parameters to Print
ESTAE=NO,	YES/NO - Activates DSARETRY
PRINT=NOGEN	Print Mode GEN/NOGEN
<u>Assembly Options</u>	
DREG=YES,	Generate ASMDREG
BRC=YES,	Generate IEABRC
OPSYN20=NO,	Generate 20-Bit OPSYNs
KILLBDD=NO,	YES/NO – Kill BDDD Instructions
<u>Tracing Options</u>	
TRACE=YES,	Allow Program Traces
BUFFDAT=NO,	YES/NO to Buffer Trace Data
BUFFMAX=1000,	Number of Lines to Buffer
MSG=,	Message to Print
REGS=ALL,	Registers to Print
DATA=,	Data to Print
TRCDD=TRCPRINT,	Trace DDNAME
TRCPGM=RDHTRACG	Trace Program Name - IMP=BASR

1.1.2 PGMNTRY Parameters

These parameters are described more fully below:

IMP=SVC Default: SVC
IMP=BASR
IMP=TRAP2

This parameter specifies the implementation that is to be used to invoke the traces. There are three different ways to implement the traces, and each has advantages and disadvantages, but they all end up in RDHTRACE, which is the trace point processor. The default is to use the SVC implementation, which is useful for Key(8) programs.

The BASR implementation was put in for me because it makes it easier to run my tests when I'm in there making changes to the traces. The disadvantage with BASR is that the calling program must be in Supervisor State and Key(0).

The TRAP2 implementation is the most versatile in that it can be used in PC's, programs in cross-memory mode and within an SRB, though I don't have that support available as yet. But this will likely be the default in the future.

TYPE=SUBR Default: SUBR
TYPE=MAIN

The difference here is that TYPE=MAIN will always do a STORAGE OBTAIN to acquire its own DSA upstack separate from the caller. The TYPE=SUBR will use the upstack of the caller for his DSA if there is sufficient room available. If there is not sufficient room available the macro will obtain a separate area for upstack DSA to use.

BASE= (No Default)

This parameter is used to specify program base registers to be used to address program storage. A USING will also be generated that will provide addressability beginning at the label specified on the PGMNTRY macro. A maximum of four base registers are supported. But realistically in today's world you don't need any program base registers, which is why there is no default.

CBAS= (No Default)

This parameter is used to specify base registers for the constant area within the program for those programs that are written such that they do not require base registers for the source code. A USING will also be generated that will provide addressability beginning at the primary LOCTR for constants. A maximum of four base registers are supported.

DBAS= (No Default)

This parameter is used to specify additional base registers for the programs Dynamic Storage Area (DSA). A USING will also be generated that will provide addressability beginning at the primary LOCTR for constants. A maximum of four base registers are supported. R13 will always be the primary DBAS register, whether you use this stacking DSA or not is up to you, understanding that this storage will always be below-the-line.

ABAS= (No Default)

This parameter is used to specify base registers for the programs Above-The-Bar Storage Area (ATB). A USING will also be generated that will provide addressability beginning at the primary LOCTR for Above-The-Bar storage. A maximum of four base registers are supported. This parameter defines your stacking ATB storage. This storage is always above-the-bar and should only be used for 64-bit programs.

AMODE=31, Default: 31

This parameter is used to specify the addressing mode for the program and may be specified as either 31 or 64. I've made a conscious choice to not support amode 24-bit programs with this system.

RMODE=ANY Default: ANY

This parameter specifies the residency mode for the program. This can be 24, 31, or ANY.

ASCENV=P Default: P – for Primary Mode
ASCENV=AR

This parameter is available to indicate whether the program is to run in Primary Mode or in Access Register Mode. Home Mode and Secondary Mode are not supported. And that was a conscious decision, by me.

ARCHLVL=2 Default: 2

This parameter allows the user to indicate which architecture level is being used for this assembly. Some IBM macros generate different code based on the level of the machine. An ARCHLVL of 2 indicates z/OS. The options are 0/1/2 and have the same meanings as on the SYSSTATE macro which is generated internally.

KEY=8 Default: 8 unless IMP=BASR, then its 0.

This parameter allows the user to specify the storage key to be used to acquire DSA and ATB storage. Usually key(8) and SP(0) is fine, but certain programs need to run in different keys, like JES2 Exits that need to run in Key(1).

SP=0 Default: 0 unless IMP=BASR then its 240.

This parameter is there to provide the user the ability to override the default subpool for obtaining storage for DSA and ATB storage. And SP(240) is a special subpool for Supervisor State programs that really ends up as subpool zero. It's an IBM anomaly.

LEN=4 Default: 4

This parameter is specified as a decimal number that represents the length of the DSA Upstack to OBTAIN. The default is 4K, and it should be specified as a decimal number in a multiple of 4K. LEN=4 means 4K.

The default is set thus for TYPE=SUBR programs, where they should be getting their DSA from the calling program. If there is not enough room in that preceding DSA, the program will obtain its own DSA of 4K. Realistically, a TYPE=MAIN program should specify a more realistic number like LEN=32, so that the subprograms called can use the stacking DSA from the main program without having to obtain and thereafter free their own storage.

DSANAM= (No Default)

The generated primary DSECT name for DSA for a given CSECT in the assembly will always be the CSECT name with an '_DSA' appended. This is necessary to assure a unique DSA DSECT name for all invocations of the PGMNTRY Macro. However, this causes a problem when one wishes to use hard coded DSA offsets as in DSAWORKD-DSA(R3). It means that one would have to code: DSAWORKD-CSECTNAM_DSA(R3) which is a little verbose. So, I added this parameter so I could provide a short name that will become the name of the DSA DSECT for a given invocation of PGMNTRY. It must be unique for a given CSECT within a single assembly. We suggest: DSA, DSB, DSC, DSD, etc.

It's really only important or even useful when you have several different CSECTs or4 PGMNTRY macros in a single assembly. But in that situation, it can be helpful.

PARM= (No Default)

This parameter represents a future desire, and while it is in the macro is not yet functional. The idea is that when a program is called and we know the parameters that we're expecting as input pointed to off R1, then we can print the parameters passed in at program entry to the trace datasets.

ESTAE=NO Default: NO

This parameter indicates whether or not a basic level of ESTAE support is desired for this program. If specified as YES a very simple ESTEA routine will be generated in the last CSECT LOCTR that will activate DSARETRY in DSADSECT. What that means is that if at the time the ESTAE routine is entered, if DSARETRY contains a non-zero value the routine will retry to that address. And if the value of DSARETRY is zero, percolation will occur.

PRINT=NOGEN Default: NOGEN

This parameter may be specified as GEN or NOGEN. Specifying PRINT=GEN will cause the macro to be expanded, where under normal circumstances it would not.

Parameters Related to the Assembly

DREG=YES Default: YES

This parameter is provided so that one may override the default which is to include IHADREG, IBM's macro that generates register equates.

BRC=YES Default: YES

This parameter is provided so that one may override the default behavior of the system which is to always include the copybook IHABRC, which changes the standard BDDD branch instruction to branch relative instructions.

OPSYN20= Default: YES

This parameter is provided so that one may override the default behavior of the system which is to include this copybook, a copybook I wrote found in my maclib, which will OPSYN out many of the BDDD RX and RS instructions and replace them with the BDLDH or 'Y' instruction set. For instance: L becomes LY, and ST becomes STY, and so forth. This is a key piece of supporting the 20-Bit base register as many IBM macros in maclib are not 20-Bit compliant.

KILLBDD= YES/NO – Kill BDDD Instructions

This parameter is there to provide an easy way to determine if a program is actually 20-bit compliant, or not. What it does is to add 4K of fluff to the program CSECT Storage, to the Constant Storage Area, to the Dynamic Storage Area (DSA), and to the Above-The-Bar Storage Area (ATB). Any BDDD instructions that were used in the program will therefore be outside the range of a single base and generate an assembly error, allowing the developer to assure that the code is 20-bit compliant.

Parameters Related to Tracing

TRACE=YES Default: YES
TRACE=NO

This parameter may be specified as NO if it is ever necessary to disable the traces for a given program.

BUFFDAT=NO, Default: NO
BUFFDAT=YES

This parameter may be specified as YES or NO to indicate if buffering of the trace output data is desirable for this TYPE=MAIN program. If requested data will be buffered in a round-robin table above the line until task termination, at which point it will be written to SYSOUT. If specified as NO then data will be written to SYSOUT as it is created.

BUFFMAX=1000 Default: 1000

This parameter may be used to specify the number of lines of trace output to buffer, when BUFFDAT=YES is specified.

TRCDD=TRCPRINT Default: TRCPRINT

This parameter can be used to specify the DDNAME used to activate the traces. If TRCPRINT is in the JCL then tracing will be in effect. If TRCPRINT is not in the JCL then no traces will be generated. This parameter was put in because I wanted to have the ability to use a stable version of the traces to test a working version of the traces. But this has not yet been fully tested.

TRCPGM=RDHTRACG Default: RDHTRACG

This parameter was put into the system in support of TRCDD and allows the trace program that will be called in a BASR version to be overridden. The production version is RDHTRACG which is the glue code to RDHTRACE for the IMP=BASR version of the traces. Again this functionality has not yet been fully tested.

PRINT=NOGEN Default: NOGEN

This parameter may be specified as GEN or NOGEN. Specifying PRINT=GEN will cause the macro to be expanded, where under normal circumstances it would not.

The following parameters are simply passed into the #TRACE macro that is generated within the PGMNTRY macro and are described more fully in the description for the #TRACE macro.

MSG=,	Message to Print
REGS=ALL,	Registers to Print
DATA=,	Data to Print

1.2 PGMEXIT Macro

The program exit macro does termination processing for the program. It frees storage, closes down the ESTAE(s) and sets the return and reason codes to be passed back to the calling program.

1.2.1 PGMEXIT Syntax

The syntax for the PGMEXIT macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
PGMEXIT	Macro name.
b	One or more blanks.

RC=	Return Code
RS=	Reason Code
MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.2.2 PGMEXIT Parameters

These parameters are described more fully below:

RC=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

RS=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

The following parameters are simply passed into the #TRACE macro that is generated within the PGMEXIT macro and are described more fully in the description for the #TRACE macro.

MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.3 SVCNTRY Macro

The SVCNTRY and SVCEXIT macros are not as robust as the PGMNTRY and PGMEXIT macros. It was written primarily for the purpose of handling the different entry and exit protocols that are inherent with SVC.

The SVCNTRY can share DSA with any TYPE=SUB PGMNTRY program that it calls, so the functionality of PGMNTRY can be utilized, by making the SVCNTRY program a stub that calls a PGMNTRY program to do the actual work.

Anyway, it's here if you need to write an SVC.

1.3.1 SVCNTRY Syntax

The syntax for the SVCNTRY macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
SVCNTRY	Macro name.
b	One or more blanks.

IMP=BASR	Implementation: BASR/SVC/TRAP2
SVCNO=	SVC Number for Protocol
SVTYP=4	SVC Type
TYPE=USER,	Type of Entry USER/HOOK
BASE=	Program Base Registers
CBAS=	Constant Base Registers
DBAS=	Additional Dynamic Storage Base Registers
AMODE=31	Default Amode 31/64
RMODE=ANY	Default Rmode
LEN=4	Length for DSA Getmain
PRINT=NOGEN	Print Mode GEN/NOGEN
<u>Tracing Options</u>	
TRACE=YES,	Allow Program Traces
BUFFDAT=NO	YES/NO to Buffer Trace Data
BUFFMAX=1000	Number of Lines to Buffer
MSG=	Message to Print
REGS=ALL	Registers to Print
DATA=	Data to Print

1.3.2 SVCNTRY Parameters

These parameters are described more fully below:

IMP=SVC Default: SVC
IMP=BASR
IMP=TRAP2

This parameter specifies the implementation that is to be used to invoke the traces. There are three different ways to implement the traces, and each has advantages and disadvantages, but they all end up in RDHTRACE, which is the trace point processor. The default is to use the SVC implementation, which is useful for Key(8) programs.

The BASR implementation was put in for me because it makes it easier to run my tests when I'm in there making changes to the traces. The disadvantage with BASR is that the calling program must be in Supervisor State.

The TRAP2 implementation is the most versatile in that it can be used in PC's, programs in cross-memory mode and within an SRB, though I don't have that support available as yet. But this will likely be the default in the future.

TYPE=USER Default: USER
TYPE=HOOK

This parameter refers to how the SVC will be installed, as a USER SVC or as a HOOK into an existing SVC. While this parameter exists on th macro, there is no support behind it at present for managing multiple hooks on an IBM SVC which is where I was going with this.

Unlike the PGMNTRY macro this parameter does not refer to whether this is a main program or a subprogram. All SVCs must be TYPE=MAIN, in that they have to get their own DSA. Other TYPE=SUBR programs may be called by the SVC and they will share the DSA

SVCNO= (No Default)

Specify the SVC number for a HOOK or a USER SVC and it will be placed in the protocol at the beginning of the program. At the moment nothing uses this information, but if support for hooking into IBM SVCs were to be added here this information would be important.

SVTYP= Default: 4

This information is placed into the protocol at the beginning of the program and may be specified as 3 or 4. It is used by the SVC install program to update the SVC table entry

BASE= (No Default)

This parameter is used to specify program base registers to be used to address program storage. A USING will also be generated that will provide addressability beginning at the label specified on the PGMNTRY macro. A maximum of four base registers are supported.

CBAS= (No Default)

This parameter is used to specify base registers for the constant area within the program for those programs that are written such that they do not require base registers for the source code. A USING will also be generated that will provide addressability beginning at the primary LOCTR for constants. A maximum of four base registers are supported.

DBAS= (No Default)

This parameter is used to specify additional base registers for the programs Dynamic Storage Area (DSA). A USING will also be generated that will provide addressability beginning at the primary LOCTR for constants. A maximum of four base registers are supported.

AMODE=31, Default: 31

This parameter is used to specify the addressing mode for the program and may be specified as either 31 or 64. I've made a conscious choice to not support amode 24-bit programs with this system.

RMODE=ANY Default: ANY

This parameter specifies the residency mode for the program. This can be 24, 31, or ANY.

LEN=4 Default: 4

This parameter is specified as a decimal number that represents the length of the DSA Upstack to OBTAIN. The default is 4K, and it should be specified as a decimal number in a multiple of 4K. LEN=4 means 4K.

PRINT=NOGEN Default: NOGEN

This parameter may be specified as GEN or NOGEN. Specifying PRINT=GEN will cause the macro to be expanded, where under normal circumstances it would not.

Parameters Related to Tracing

TRACE=YES Default: YES
TRACE=NO

This parameter may be specified as NO if it is ever necessary to disable the traces for a given program.

BUFFDAT=NO, Default: NO
BUFFDAT=YES

This parameter may be specified as YES or NO to indicate if buffering of the trace output data is desirable for this TYPE=MAIN program. If requested data will be buffered in a round-robin table above the line until task termination, at which point it will be written to SYSOUT. If specified as NO then data will be written to SYSOUT as it is created.

BUFFMAX=1000	Default: 1000

This parameter may be used to specify the number of lines of trace output to buffer, when BUFFDAT=YES is specified.

The following parameters are simply passed into the #TRACE macro that is generated within the PGMNTRY macro and are described more fully in the description for the #TRACE macro.

MSG=	Message to Print
REGS=ALL	Registers to Print
DATA=	Data to Print

1.4 SVCEXIT Macro

The SVC exit macro does termination processing for the program. It frees storage, closes down the ESTAE(s) and sets the return and reason codes to be passed back to the calling program.

1.4.1 SVCEXIT Syntax

The syntax for the SVCEXIT macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
SVCEXIT	Macro name.
b	One or more blanks.

RC=	Return Code
RS=	Reason Code
MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.4.2 SVCEXIT Parameters

These parameters are described more fully below:

RC=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

RS=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

R1=

This parameter is the value to be passed back to the calling program in R1. If not specified zero is returned.

The following parameters are simply passed into the #TRACE macro that is generated within the SVCEXIT macro and are described more fully in the description for the #TRACE macrto.

MSG= Message to Print to Traces

REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.5 SUBNTRY Macro

The SUBNTRY macro works in conjunction with the SUBEXIT macro to provide services for subroutines executing within the program. R14 is always saved be SUBNTRY and restored by SUBEXIT before returning to the caller. Additional registers may optionally be save with the SAV= parameter, and AMODE and ASC Environment may also be set by this macro.

1.5.2 SUBNTRY Syntax

The syntax for the SUBNTRY macro is as follows:

label	Label: Beginning in column 1. (Required)
b	One or more blanks.
SUBNTRY	Macro name.
b	One or more blanks.

SAVE=	Specify Registers to save.
AMODE=	Specify AMODE for this Subroutine (31/64)
ASCENV=	Specify ASC Environment (P/AR)
MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces
PRINT=NOGEN	Print Mode GEN/NOGEN

1.5.2 SUBNTRY Parameters

These parameters are described more fully below:

SAVE=

This parameter can be used to specify additional registers to be saved and restored by this subroutine. It breaks down to a STMY/LMY to do the save and restore. And if may specify numeric register values or register notation.

SAVE=(2,6) or SAVE=(R2,R6). Storage to save the registers is automatically carved out from the DSA LOCTR.

AMODE=

This parameter allows for changing the addressing mode for a given subroutine. It will issue the SAM31/SAM64 and also changes the SYSSTATE to indicate this. SUBEXIT will restore the previous state upon exit from the subroutine.

ASCENV=

This parameter allows for changing the ASC environment for this subroutine. It will issue the SAC 000/512 to change the state and also change the SYSTATE to indicate the change. SUBEXIT will restore the previous state upon exit from the subroutine.

The following parameters are simply passed into the #TRACE macro that is generated within the SUBNTRY macro and are described more fully in the description for the #TRACE macro.

MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.6 SUBEXIT Macro

The SUBEXIT macro performs the opposite function for the SUBNTRY macro. It will restore the original SYSSTATE if that was changed on SUBNTRY, and can set return code and reason codes if desired.

1.6.1 SUBNTRY Syntax

The syntax for the SUBEXIT macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
SUBEXIT	Macro name.
b	One or more blanks.

RC=	Return Code
RS=	Reason Code
R1=	Additional Parameter Register
MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.6.2 SUBEXIT Parameters

These parameters are described more fully below:

RC=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

RS=

This parameter is the return code to be passed back to the calling program. If not specified zero is returned.

R1=

This parameter is the value to be passed back to the calling program in R1. If not specified zero is returned.

The following parameters are simply passed into the #TRACE macro that is generated within the SVCEXIT macro and are described more fully in the description for the #TRACE macro.

MSG=	Message to Print to Traces
REGS=ALL	Registers to Print to Traces
DATA=	Any Desired Data to Print to Traces

1.7 #TRACE Macro

The #TRACE macro is the centerpiece of the whole system, the workhorse, if you will. There is one of these macros embedded in each of the other program management macros; PGMNTRY, PGMEXIT, SUBNTRY, and SUBEXIT. This is the macro that produces the trace point trap, whether it be a BASR, a TRAP2, or an SVC.

The macro generates an execute for one of the trap points in the DSATRAPS area of DSADSECT. It also generates a trap definition dsect area mapped by TRPDSECT in the constant area of the code. Upon execution of the trap RDHTRACE will get control to process the request and generate the trace output, either to a buffer or directly to SYSOUT.

Some of the parameters on the macro are active only for the PGMNTRY invocation and are intended to define the environment for the traces within this program. Others may be used on any trace point trap.

1.7.1 #TRACE Syntax

The syntax for the #TRACE macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#TRACE	Macro name.
b	One or more blanks.

TYPE=	TRAP/PGME/PGMX/SUBE/SUBX
NAME=	Force Name of Trace Point
REGS=	List of Registers to Trace
MSG=	Message to Print to Traces
DATA=	Any Desired Data to Print to Traces
SQUEEZE=YES	YES/NO Removes Multiple Spaces From Message
HOHEAD=NO	YES/NO Removes Heading for Trap Point in Traces
PRINT=NOGEN	GEN/NOGEN Suppresses macro Generation in Assembly

PGMNTRY Macros Only

DD=TRCPRINT	DD Name for SYSOUT
BUFFDAT=NO	YES/NO Write Trace Data to Round Robin Buffer
BUFFMAX=1000	Number of Lines in Buffer if BUFFDAT=YES

1.7.2 #TRACE Parameters

These parameters are described more fully below:

TYPE= Default: TRAP

This parameter defines the type of trace point trap to be generated. Valid values are TRAP, PGME, PGMX, SUBE, and SUBX. The type of trap will be indicated as a bit flag in TRPDSECT to inform RDHTRACE, and processing varies somewhat based on the type of trap.

NAME= (No Default)

This parameter allows the user to force Name of Trace Point. The name of the trace point can also be specified on the label of the macro. If neither is specified a name will be generated. A name is necessary to uniquely identify every trace point in a given program. In this way, any trace point trap can be identified with a fully qualified name in the form of lmod.csect.trapname.

REGS= Default: NONE

This parameter allows the user to specify a list of registers to be printed when the trace point is invoked. It is coded as a sublist, either the numeric value of the register of using the common equates. Examples: REGS=(R14,R15) or REGS=(2,3,4,5) . It may also specify ALL or NONE. Register when printed to the traces will be printed with the access register plus the 64-bit value in the GP registers.

MSG= (No Deafult)

The MSG= parameter is one of the two most valuable parameters on the #TRACE macro, and it gets pretty sophisticated. A MSG= parameter can be specified as a simple message enclosed in quotes such as:

```
MSG='We Got here 1.'
```

Or it can be specified as a sublist of text enclosed in single quotes and/or data names that may reside in any addressable data area, such as the constant area, DSA, ATB, or if you happen to be in AR Mode in an addressable area within a dataspace or address space. Some examples:

```
MSG=('ASCB Address: ',DSAASCBA,', ASXB Addr: ',DSAASXB)
MSG=('My child is  ',WORKH,' years old. Her name is ',WKCL9)
MSG=('The Hex value is: ',DSAXL8)
```

The macro generates an entry in the TRPDSECT that gets generate with an SL3 to identify the location of the data and some bit flags that identify the type and length of the data item in the sublist. This allows the code in RDHTRACE that process the trace point to know how to print the data. Data types of H, F, D, Z, or P will be taken as decimal data and the decimal value will be displayed. Character data will be included in the line as is, and any other data will be printed as a hexadecimal value when included on the line to be printed.

The ability to use type and length is quite valuable as it eliminated the need to do all that formatting stuff yourself. The one issue I have had here is that sometime the type used for data items in some csects were chosen without this facility in mind. So, when some codes an fullword dataarea in a dsect and uses that dataarea to store an address, the traces will see it's F-Type area and print it as a decimal value, when what's really wanted is a hex representation. The proper definition would be to use an A-Type declaration which would be printed properly.

However, you can't do that when the dsect is not your, such as an IBM data area. So, the way to fix that is to redefine the data area with an equate like this: STOKEN EQU ASCBSTKN,8,C'X'. Then you can use the equated value in the MSG= parameter and get the desired result.

DATA= (No Default)

This parameter is the coolest piece of this whole system, as it allows you to display virtually any data that you desire upon the execution of a trace point trap. DATA= is specified in a sublist as a series of one or more triplets with a title, an address, and a length. For example:

```
DATA=('My title',0(R5),122)  -or-
DATA=(('Area 51',A51,A51L),( ('Area 52',A52,A52L))
```

This will allow you display pretty much any storage area. It can be above-the-bar data if you happen to be in AMODE(64), it can be in a data space or another address space if you happen to be in ARMODE and have your access register loaded properly, and it can even display data in real storage address by specifying a real storage address.

Another capability is the support of a single level of indirection. Indirection is indicated by prefacing the address with one of the following indirection characters.
! – Indicates that the address is a pointer to a doubleword 64-bit value that contains the address of the data to be displayed.
? – Indicates that the address is a pointer to a fullword 31-bit value that contains the address of the data to be displayed.
% – Indicates that the address is a pointer to a fullword 24-bit value that contains the address of the data to be displayed in the low order three bytes.
@ - Indicate that the address is a real storage address and not a virtual address.

Another feature that I have found to be quite valuable is to specify the length in register notation allowing the length of the area to be displayed to be obtained from a register at runtime. For Example:

```
DATA=('ASXB',0(R3),(R4))
DATA=('ASCB Address',?ASCBADDR,256)
```

DATA=(Show me the DUCT in Real Storage',@0(R5),256)

SQUEEZE= Default: YES

This parameter allows the user to tell the system not to remove consecutive spaces from the output to be generated with the MSG= parameter. The default is to remove multiple spaces from the message. This can be useful if the desire is to produce some kind of columnar report to the trace dataset.

HOHEAD= Default: NO

This parameter allows the user to suppress the heading line that would normally be printed to the traces to identify the location of the trace point trap. This can be useful if the desire is to formatted print a report of some kind to the traces. The standard heading for the trace point trap look like this: T) Trap: TRP0036N At: RDHBST01.RDHBST01+025C. It identifies the load module, the csect and to the offset into the csect where the trace point was taken.

PRINT= Default: NOGEN

This parameter may be specified as GEN or NOGEN. Specifying PRINT=GEN will cause the macro to be expanded, where under normal circumstances it would not.

<u>PGMNTRY Macros Only</u>

DD= Default: TRCPRINT

This parameter, when specified on a PGMNTRY macro was intended to allow the user to indicate the DDNAME of the SYSOUT dataset to be used to trace this program. I'm not sure if it works or not. I can't remember if I tested it, and I haven't used it in recent times.

BUFFDAT= Default: NO

This parameter allows the user to indicate whether they want to have the traces written to an in-storage buffer and printed to SYSOUT at end-of-task, or if they want trace data written to SYSOUT directly. The default is to write data directly to SYSOUT, and I may change that in the future, because if makes more sense to buffer the data. At present the data is written to gotten storage above-the-line and that needs to be changed to an above-the-bar location at some future point. For now, it works.

BUFFMAX= Default: 1000

This parameter is specified as a numeric value to tell the system how many lines of trace data are to be buffered before the in-storage table begins to wrap. The trace output dataset is built of 128-byte records instead of the standards of 121 or 133 because it aligns in storage nicely.

1.8 CONBEG and CONEND Macros

These macros define the beginning and end of the constant area that under normal circumstances would be addressed via the CBAS= parameter of the PGMNTRY macro, when there are no program base registers defined. This is the situation where we are writing code without base registers and using IHABRC to redefine all the BDDD branch instructions to their Branch Relative cousins.

In this situation it is necessary to obtain addressability to the constant area only, and these macros provide the means of communicating that location via the LOCTRs that were defined in PGMNTRY.

It’s a very simple construct and the actual macros don’t do much. The CONBEG macro simply initiates the Constants LOCTR. And the CONEND issues a final LTORG to assure that literals are generated within the Constant Area.

There are no parameters on these macros and a label field is not supported.

The use of these macros is simulated below.

```
          CONBEG ,
*
** Define Program Constants Here.
*
HEXTABLE  DC     X”0123456789ABCDEF’
F0        DC     F’0’
F1        DC     F’1’
*
MSG1      DC     CL128’This is a test message.’
*
          CONEND ,
```

1.9 DSABEG and DSAEND Macros

These macros define the beginning and end of DSA (Dynamic Storage Area) that is addressed via R13, though if additional DSA base registers are needed, which they should not be in a 20-Bit displacement reality, additional base registers can be added via the DBAS= parameter of the PGMNTRY macro.

This storage is allocated below-the-line.

The DSA is key to this system. It contains variables to maintain the stacking DSA and the stacking ATB storage. The first 18 fullwords are a standard MVS savearea. The prefix to this area is defined in the DSADSECT macro.

One of the very useful features of this prefix is what I call propagated storage. This begins at label DSAPROPA. This area is propagated, which is to say copied, from the calling program to the called program. So, if a calling program places information into one of these fields, the PGMNTRY macro of the called program will copy that information into the DSADSECT of the called program where it will be available. This area is used primarily to support the traces, but there are several fields available to the developer of any system at label DSAPROPS. It provides a method of passing additional parameters from a TYPE=MAIN program to all of its subordinate TYPE=SUB programs.

This area also defines several general purpose work areas that I tend to use frequently. These are DSAWORKD, DSAWORKF, DSAWORKH and DSAWKCL9. These are intended to be volatile work areas and should not be used to store information for any lengthy period of time.

The PGMNTRY macro defaults to obtaining a 4K area of storage for DSA. However, for TYPE=MAIN programs I like to acquire a 32K area. TYPE=SUB programs will obtain an additional 4K area, if there is insufficient storage available from the calling program to satisfy its needs. And these areas will be chained forward and backwards. However, in designing your system, it's best to assure that the TYPE=MAIN program obtains the maximum storage up front, so that the TYPE=SUB programs can avoid the overhead of doing the GETMAINs and FREEMAINs, which is the main purpose of having a stacking DSA.

There are several LOCTRs setup in PGMGBINI and defined in a Globals Macro PGMGBLAS, that may be used by anyone writing a macro, to define storage within the DSA. This can be very useful when trying to avoid the need for MF=E and MF=L constructions. With these LOCTRs, such constructs are no longer needed. The macro itself can do it all.

The DSAEND macro is used at the end to define the end of the DSA area. It doesn't do much in it's own right, but there are a few functions that are integral to the system. It generates a label for the length of the DSA, sets an ending eyecatcher and generates all the register saveareas for any SUBNTRY macros in the program. It also generates a fullword called DSAOVCHK, which is a word that contains the address of that address, which is initialized in PGMNTRY. PGMEXIT checks that address to see if we have overlaid our DSA and if so forces an ABEND.

This construction makes it incredibly easy to write reentrant code, which is the primary reason for its existence.

Basically all you have to do is code the DSABEG macro to initiate DSA, code whatever variables are desired, then terminate the construct with the DSAEND macro.

You can look at some of the programs in the TST library to view how I have used DSA, but a simple version might look like this:

```
         DSABEG ,
*
** Define Program Dynamic Storage Area
*
DSADATAF DS    F                   Define Fullword Data Area
DSADATAD DS    D                   Define Doubleword Area
DSADCB   DS    9D                  Define Area for DCB
DSAPRINT DS    CL133               Define Print Line
*
         DSAEND ,
```

1.10 ATBBEG and ATBEND Macros

These macros define the beginning and end of ATB (Above-The Bar Storage Area) that is addressed via the register specified on the ABAS= parameter of the PGMNTRY macro.

This storage is allocated above-the-bar.

A TYPE=MAIN program that specifies an ABAS= parameter will obtain a single one megabyte page of above-the-bar storage to use for stacking ATB. Subsequent TYPE=SUB programs called by the main program will use residual storage within the page for whatever space is needed for that program. If it runs out of space, not likely, it will obtain another page and chain them forward and backward. The basic technology is identical to that which I use for DSA, except that this storage is above-the-bar, the 2-gigabyte bar.

The control items needed to manage ATB storage are maintained in DSADSECT, or the DSA prefix area and propagated in propagated storage from one program to the next called program. One of the cool benefits of this design is that we can have a main program that initiates ATB, that then calls a program that does not use ATB, that then calls a TYPE=SUB program that requires ATB and it will use the residual storage in that ATB from the original main program. Waste not, want not.

The macros that support this construction are ATBBEG, ATBEND and ATBDSECT. These macros don't do a lot, other than to establish the eyecatchers, overlay protection and sizing. Most of the data necessary to manage ATB is stored in the DSA prefix.

As with the DSA area, there are LOCTRs generated in PGMNTRY initialized in PGMGBINI and defined in PGMGBLAS. These LOCTRs are available for any macro to dynamically allocate any storage that they may require without having to resort to an MF=L construction. And that's way cool.

To utilize this area all you have to do is code the ATBBEG macro to initiate ATB, code whatever variables are desired, then terminate the construct with the ATBEND macro. You also have to code ABAS= on the PGMNTRY macro to establish addressability to the area.

You can look at some of the programs in the TST library to view how I have used ATB, but a simple version might look like this:

```
         ATBBEG ,
*
** Define Program Above-The-Bar Storage Area
*
ATBDATAF DS    F                  Define Fullword Data Area
ATBDATAD DS    D                  Define Doubleword Area
*
         ATBEND ,
```

Chapter 2 – Other Supporting Macros

Now that the basic system is complete, there are many different things that can be melded into the system to add value and make it a true "development platform." These supporting macros are however, ancillary to the base product. They add value and are helpful in their own right and in many cases stand on their own. And there are many more possibilities that I would like to add to this system over time. And the LOCTR structure in the basic system allow for some incredible possibilities for future development.

This section describes a few that are available now.

2.1 #CALL Macro

So, why in the hell do we need a #CALL macro? Because IBM's CALL macro is a piece of crap that does not support 64-bit programs, and I'm not going to wait for IBM to fix their code. I could grow old and die by the time they realize the error of their ways.

The reasons and why-for's for the need for this macro get real complicated and is based on the history of z/OS and how it came to be. So, for those of you who weren't around thirty years ago, let me start this explanation with a little history lesson.

You have to understand that IBM has an absolute commitment to this thing called "downward compatibility." What this means is that programs that were written for MVS/SP, MVS/XA and MVS/ESA must still be able to run on a z/OS system. And this requirement has hamstrung the developers of the hardware. And therein lays the problem. So, with that basic understanding let us move on.

z/OS began way early on, as this Operating System called MVT which stood for Mutli-tasking with a Variable number of Tasks. It was a 24-bit system without address spaces. Then came MVS/SP which added address spaces to the mix, but was still a 24-bit system. A 24-bit system is a system that was limited to 16-megabytes of storage. Back in those times, we used the BAL and BALR instructions to affect transfer between programs and subroutines. Nobody should be using these instructions today, they're antiquated. When these instructions were executed they stored bits 16-24 of the PSW in the first 8-bits of the return address.

This was a real problem when IBM created MVS/XA which provided the first rudimentary support for 31-bit addresses. The BAL/BALR could no longer be used, but because of "downward compatibility" could not be modified to support the new architecture. So, they added two new instructions BAS and BASR. These instructions did not muck with the high order byte of the return address.

At the same time and also to support downward compatibility, they needed a means to distinguish the difference between a 24-bit address and a 31-bit address and they use bit0 of all addresses to make this distinction. If bit0 is one, it's a 31-bit address, and if it's zero its 24-bit address and the machine will ignore the high order byte of the address. So, this is how they manifested "downward compatibility" in creating MVS/XA and later MVS/ESA which was a more robust version of 31-bit support. In ESA, they added several more instructions, but the ones relevant to this discussion are the BAKR and PR. This made use of the Linkage Stack for transferring control between programs. And this system here makes full use of that BAKR/PR sequence.

Okay, now we come to present day and the issues between z/OS, the Linkage Stack and "downward compatibility." When IBM created z/OS they needed a means of distinguishing the difference between 24/31-bit addresses and the 64-bit address. They had used the high order bit of a 32-bit address to distinguish between 24 and 31-bit addresses, which in a 64-bit reality is bit-33. What they did to distinguish a 64-bit address is to use bit-63, which if on, makes the address odd. And since all instructions in this machine are always a multiple of two, it was never used and so could then be used to indicate a 64-bit address.

It's not just the address, it's the AMODE (addressing mode) in the PSW that's at issue. If you're in 31-bit AMODE and you issue a BASR the return address in R14 will set bit-33 to indicate that upon return from a BAKR/PR sequence the calling program should get control back in AMODE(31). If you're a 24-bit caller bit-33 will not be set and the BAKR/PR sequence will return control in AMODE(24). And this technology has worked fine through XA and ESA.

So let me back up a little bit. The first time I wrote an AMODE(64) program that called another program with a BASR/BAKR/PR sequence that I had always used for an AMODE(31) program, I was shocked to receive and S0C4/11, a page-translation exception. I couldn't understand what happened; it all looked like good clean code. Then I examined the ABEND PSW and realized it was in AMODE(24), and I thought "How the hell did this happen?"

At first I thought that BAKR was broke, because when I looked at the linkage stack entry that was to be restored upon execution the PR, the PSW to be restored was a 24-bit PSW. Then I reviewed POPS, and saw that it was indeed working as designed. Then I blamed BASR, because BASR did not set the AMODE(64) bit, bit-63 in the return address and since bit-33 was also not set because I did the BASR in AMODE(64) I received control back in AMODE(24). And because the program was running above-the-line, the 24-bit address did not exist and a page-translation exception was the result.

When IBM went from 24-bit to 31-bit system, they replaced BAL/BALR with BAS/BASR to assure that the addressing mode bit would be set properly for return address. Now, you cannot change the way an existing instruction functions because of the requirement for "downward compatibility." But you can create a replacement for BAS/BASR that would properly set the AMODE(64) addressability bit in the return address, but IBM after eleven years of having z/OS have failed to produce that new instruction. So, we have to program around that limitation.

So, this is how the BASPM macro (Branch and Save and Preserve Mode) came into being. It's a simple macro that exists in the copybook #MAC64 and generates code that looks like this:

```
BSM   &B,0             Set AMode Bits in Target Addr
BASSM &A,&B            Set AMode Bits in Return Addr
```

This assures that the return and target addresses both have the AMODE(64) bit set in the addresses within these registers. And if this bit-63 is set in the return address, then BAKR will properly indicate that the program was in AMODE(64) and the PR instruction will return in the proper AMODE. I also created BAS31 and BAS64 macros that will set the proper AMODE for the target address if you know the AMODE of the called program.

There is a parameter on the IBM CALL macro that allows you to specify the instruction that you wish to execute, which realistically could be BALR, BASR or BASSR. So, I tried specifying BASPM, but then I hit another problem. The CALL macro specifies the operand to the instruction, as a string '14,15'. And this specification does not give me two positional parameters, but only one, and since it doesn't have to be 14,15, the operands of BASPM can be any two registers, this solution would not work.

So, this is why I finally decided to write my own replacement for the CALL macro, and I called it #CALL, which uses BASPM internally. And since I have stacking DSA and the LOCTR structure, I get the extra benefit of being able to abolish the MF=L construction. And that's pretty cool.

Oh, and just one last thing? I don't support the VL bit, as it cannot be used in a 64-bit address, so I just dumped the concept. You need to be responsible for knowing how many parameters are being passed and received.

If you have a need for a variable number of parameters to be passed to a program, I would suggest that you use the first parameter, perhaps a halfword value, to indicate the number of parameters in the list, and use that method as the new standard.

2.1.1 #CALL Syntax

The syntax for the #CALL macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#CALL	Macro name.
b	One or more blanks.

Called Program	Program to Call (Positional)
(Sublist of Parms)	Positional List of Parms to Pass to Called Program
LAEY=LAE	Instruction to Use for loading up Parameter Addresses

2.1.2 #CALL Parameters

These parameters are described more fully below:

Called Program Name (Positional)

This positional parameter specifies the name of the program to call. It may be specified in register notation as in (15) or (R2) to indicate we are to obtain the address of the called program from that register, or it may be specified as the name of an object module which will generate a VCON inline which will be loaded into R15.

Sublist of Parameters (Positional)

This parameter specifies the parameters to be passed to the called program, as a sublist such as (DSADATA1,(2),ATBDATA2). Storage for the PLIST is generated automatically in DSA, so there is no need to provide an MF=L version of this macro. If we are in AMODE(31) the PLIST will generate 4-byte addresses. If we are in AMODE(64) the PLIST will be generated as 8-byte addresses. If we happen to be in ARMODE then the Access Register values for the corresponding General Purpose Registers will be stored immediately following the GPRs and in a corresponding order as 4-byte entries. The address PLIST will be passed in R1 as always. However, if we happen to be in ARMODE the address of the Access Register PLIST will be passed in R0.

LAEY=LAE

This parameter specifies the instruction to be used when loading up the addresses of the parameters specified in the parameter list. I would have like to have used LEAY for the compliance with 20-bit displacement architecture, as LAE if a 12-bit displacement version. Unfortunately, I have noticed that this is a relatively new instruction, and is not currently supported by all the boxes that are in the field. So, I had to cave on this one and default to LAE. Oops. At some point we will change this to default to LEAY. If your box supports LAEY you are encouraged to change the default in the macro.

2.2 #GPRM Macro

#GPRM - Load Parameter Addresses Into Registers

This macro was written to be used in conjunction with the #CALL macro, which places the input parameters into DSA and supports AMODE(31/64) and ARMODE(P/AR).

The macro is operand is specified as a sublist of quadlets in the following format:

```
        A B  C          D
GPRM  (1,R2,'Comment',Len),  -Or-

GPRM ((1,R2,'Comment',Len),
      (2,R3,'Comment',Len),
      (3,R4,'Comment',Len),
      (4,R5,'Comment',Len))
```

Where:
- A - Is the sequential number of the parameter specified, as a numeric digit from1 to N.
- B - Is the register to load the parameter address into.
- C - Is a comment that may be used to describe the data at the address the parameter points to.
- D - Is the length of the data at the address that the parameter points to.

Upon execution of the macro, the indicated registers will be loaded with the values from the parameter list pointed to by R1. If SYSSTATE is AMODE(31), the four byte values will be loaded. If SYSTATE indicates AMODE(64), the eight byte values will be loaded into the 64-bit registers specified.

If SYSTATE indicates ARMODE, then the access registers will also be loaded from the corresponding address pointed to by R0.

After the registers are loaded then a #TRACE DATA= macro is internally generated to display the input parameters to TRCPRINT when tracing is active. The C and D entries are used for this purpose.

Note:

R0 and R1 must contain the access and GP registers as they were on input to the program when this macro is executed. Also, this macro uses R14 and R15 as work registers.

2.2.1 #GPRM Syntax

The syntax for the #GPRM macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#GPRM	Macro name.
b	One or more blanks.

(A,B,C,D)	Sublist of Four Parameters (Positional)
TRACE=YES	Are Traces Desired? Default=YES

2.2.2 #GPRM Parameters

These parameters are described more fully below:

(A,B,C,D) -or-
((ABCD),(ABCD),…)

This parameter is specified as a sublist that may contain one or more quadlets. 'A' represents the sequential position of the parameter to load and is specified as a numeric value from 1 to N. 'B' is the register to receive the value. 'C' is a comment that is used to describe the data area that the parameter points to. 'D' is the length of that data area.

TRACE=YES

The 'C' and 'D' parameters are used in a generated #TRACE DATA= macro generated internally to print the parameter values to TRCPRINT in the event that traces are active. However sometimes this may not be desired. Therefore this parameter is provided to allow the user disable the traces for this invocation of the macro. To tell the macro not to generate the #TRACE macro, specify TRACE=NO. The default is YES.

2.3 GOSUB Macro

The purpose of this macro is to invoke the execution of a subroutine. Internally it generates a "JAS R14,SUBR" instruction. I didn't write this one because it was really necessary or provided any extra value. I wrote it for aesthetics.

Since I use all the ASMMSP macros in my coding, which includes the IF/ELSE/ENDIF, DO/ENDDO, and SWITCH/CASE constructs the assembler instruction JAS looked out of place and a little cryptic. So, I created this macro to make the code look prettier. It fits in better, and everyone knows what GOSUB means; GOTO and return from subroutine.

2.3.1 GOSUB Syntax

The syntax for the GOSUB macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
GOSUB	Macro name.
b	One or more blanks.

SUBR	Type of Request (Positional)
REG=R14	Return Register

2.3.2 GOSUB Parameters

These parameters are described more fully below:

SUBR

This positional parameter is used to specify the name of the subroutine to be executed.

REG=R14

This parameter specifies the return register on the JAS instruction generated. It defaults to R14, which is a requirement for SUBNTRY. However, while I never use anything but R14 as a return register, I have seen that others might. So, this is provided in the interest of remaining flexible.

2.4 #MSG Macro

This macro can be invoked to print messages to a SYSOUT dataset. The default DDNAME is MSGPRINT, however you can specify a different DDNAME if you wish, but only one SYSOUT dataset is allowed per task.

The #MSG, #WTO, and #WTOR macros all invoke the #MSGI macro internally to perform the bulk of the work, but I'm not going to that one document here, it's an internal macro. You can look at the macro if you like. These macros all call RDHMSGS to do the real processing at runtime.

These macros all provide the same functionality which is basically the same as the MSG= parameter on the #TRACE macro. They allow you to intersperse text data with data items where the macro can use type and length to define the data items for later printing by RDHMSGS at runtime. This eliminates the need for MF=L, and makes the coding of messages much easier.

These macros are also foundational to a concept I have that I want to expand upon in the future. You see, when you're writing software that is intended to be sold throughout the world, you need to be able to translate your messages easily from English to whatever language is you by the customer. So, by putting the messages in a clearly defined macro format the programs can be parsed to obtain a list of all such macros in the system. These can then be translated and placed into a PDS that can be loaded and indexed by message number. Then at runtime, the text in the message can be can be replaced by the desired language.

So, that's where I'm going with this eventually, and I have designed this system in such a way that these new features can be added easily in the future.

2.4.1 #MSG Syntax

The syntax for the #MSG macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#MSG	Macro name.
b	One or more blanks.

MSG	Message to Print (Positional)
MSGID=	Message ID in the form AAA9999x
DD=MSGPRINT	DDNAME to Use
SQUEEZE=YES	YES/NO to Squeeze Out Multiple Blanks

2.4.2 #MSG Parameters

These parameters are described more fully below:

MSG (Positional, No Default)

This parameter has all the same formatting capabilities as the MSG= parameter of the #TRACE macro. For a full explanation of the power of this macro please see the MSG= parameter of the #TRACE macro. It can be specified as a simple message enclosed in quotes such as:

'We Got here 1.'

Or it can be specified as a sublist of text enclosed in single quotes and/or data names that may reside in any addressable data area, such as the constant area, DSA, ATB, or if you happen to be in AR Mode in an addressable area within a dataspace or address space. Some examples:

```
('My child is  ',WORKH,' years old. Her name is ',WKCL9)
('The Hex value is: ',DSAXL8)
```

MSGID=

This optional parameter specifies a standard z/OS Message ID to identify the message to other products or components and should be unique within a given system. It can be useful for interfacing with automation products. The form is AAA9999X and if specified it is syntax checked for compliance.

Where:	AAA	- is a 3 character product identifier
	9999	- is a 4 digit message number
	X	- is an action code
		I - Informational
		A - Action Required
		E - Error Condition

DD=MSGPRINT

This parameter specifies the DDNAME of the SYSOUT dataset that this message will be written to. The LRECL of the message datasets is a little non-standard, as I set it to 128-bytes, the same as the TRCPRINT DD. I never liked 121 and 133, and 128 is a power of 2. So, I made a command decision here. Just so you understand, if you're expecting 133 you're not going to get it.

SQUEEZE=YES

This parameter is used to indicate whether or not you wish to have consecutive blank characters within the message removed from the message prior to printing. The default is YES, and that is proper if you're printing a standard message. However, if you wish to print a tabular report you might not want extra blanks removed. In that case you can specify SQUEEZE=NO.

2.5 #WTO Macro

This macro can be invoked to print messages to the Operator Console. It is essentially the same as the #MSG macro in all respects except for where the message shall be written and the few extra parameters that are specific to WTO messages.

The #MSG, #WTO, and #WTOR macros all invoke the #MSGI macro internally to perform the bulk of the work, but I'm not going to that one document here, it's an internal macro. You can look at the macro if you like. These macros all call RDHMSGS to do the real processing at runtime.

These macros all provide the same functionality which is basically the same as the MSG= parameter on the #TRACE macro. They allow you to intersperse text data with data items where the macro can use type and length to define the data items for later printing by RDHMSGS at runtime. This eliminates the need for MF=L, and makes the coding of messages much easier.

2.5.1 #WTO Syntax

The syntax for the #WTO macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#WTO	Macro name.
b	One or more blanks.

MSG	Message to Print (Positional)
MSGID=	Message ID in the form AAA9999x
SQUEEZE=YES	YES/NO to Squeeze Out Multiple Blanks
ROUTCDE=	WTO/WTOR Routing Codes
PLINE=	Area to Place Formatted Line

2.5.2 #WTO Parameters

These parameters are described more fully below:

MSG (Positional, No Default)

This parameter has all the same formatting capabilities as the MSG= parameter of the #TRACE macro. For a full explanation of the power of this macro please see the MSG= parameter of the #TRACE macro. It can be specified as a simple message enclosed in quotes such as:

'We Got here 1.'

Or it can be specified as a sublist of text enclosed in single quotes and/or data names that may reside in any addressable data area, such as the constant area, DSA, ATB, or if you happen to be in AR Mode in an addressable area within a dataspace or address space. Some examples:

```
('My child is  ',WORKH,' years old. Her name is ',WKCL9)
('The Hex value is: ',DSAXL8)
```

MSGID=

This optional parameter specifies a standard z/OS Message ID to identify the message to other products or components and should be unique within a given system. It can be useful for interfacing with automation products. The form is AAA9999X and if specified it is syntax checked for compliance.

```
Where:      AAA   - is a 3 character product identifier
            9999  - is a 4 digit message number
            X     - is an action code
                    I     - Informational
                    A     - Action Required
                    E     - Error Condition
```

SQUEEZE=YES

This parameter is used to indicate whether or not you wish to have consecutive blank characters within the message removed from the message prior to printing. The default is YES, and that is proper if you're printing a standard message. However, if you wish to print a tabular report you might not want extra blanks removed. In that case you can specify SQUEEZE=NO.

ROUTCDE=

This parameter is specified as a sublist enclosed in parentheses to indicate the WTO routing code desired. Routing code values are specified as numeric values between 1 and 16 separated by commas, as in ROUTCDE=(1,11).

PLINE=

This parameter is unique to the #WTO. It is used to indicate that we're not actually going to write the message to the console, but rather the macro will return the formatted message to the caller in a 128-byte buffer for the caller to do with as they wish. I added this functionality, when I put in the ABEND Diagnostics to RDHTRACE. This allowed RDHMSGS to do all the formatting in a tabular reality and I would then write the buffer to TRCPRINT. It saved me some work.

2.6 #WTOR Macro

2.6.1 #WTOR Syntax

The syntax for the #WTOR macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#WTOR	Macro name.
b	One or more blanks.

MSG	Message to Print (Positional)
MSGID=	Message ID in the form AAA9999x
SQUEEZE=YES	YES/NO to Squeeze Out Multiple Blanks
REPLY=	Reply Area Address
ECB=	Reply ECB Address

2.6.2 #WTOR Parameters

These parameters are described more fully below:

MSG (Positional, No Default)

This parameter has all the same formatting capabilities as the MSG= parameter of the #TRACE macro. For a full explanation of the power of this macro please see the MSG= parameter of the #TRACE macro. It can be specified as a simple message enclosed in quotes such as:

'We Got here 1.'

Or it can be specified as a sublist of text enclosed in single quotes and/or data names that may reside in any addressable data area, such as the constant area, DSA, ATB, or if you happen to be in AR Mode in an addressable area within a dataspace or address space. Some examples:

('My child is ',WORKH,' years old. Her name is ',WKCL9)
('The Hex value is: ',DSAXL8)

MSGID=

This optional parameter specifies a standard z/OS Message ID to identify the message to other products or components and should be unique within a given system. It can be useful for interfacing with automation products. The form is AAA9999X and if specified it is syntax checked for compliance.

Where:	AAA	- is a 3 character product identifier	
	9999	- is a 4 digit message number	
	X	- is an action code	
		I	- Informational
		A	- Action Required
		E	- Error Condition

SQUEEZE=YES

This parameter is used to indicate whether or not you wish to have consecutive blank characters within the message removed from the message prior to printing. The default is YES, and that is proper if you're printing a standard message. However, if you wish to print a tabular report you might not want extra blanks removed. In that case you can specify SQUEEZE=NO.

REPLY=

This parameter specifies an area of storage below-the-bar, where the reply to the message is to be placed when the Operator replies to the message.

ECB=

This parameter indicates the address of the ECB that will be posted when the message has been replied to. The caller can then issue a WAIT macro on this ECB, if desired.

2.7 #DD Macro

The #DD macro is intended to perform Dynamic Allocation with a single instruction. It’s a replication of the functionality that David Hempstead produced in the DDX macro for DMS for Sterling Software. Only it isn’t done.

The reason I’m documenting it herein is because if you look into the macro library, you’re going to see a lot of #DDxxx macros, and these are inner macros to #DD, and I feel it deserves and explanation. What I was going for was to produce all the functionality allowed for in the text units that could be input to dynamic allocation. So, I built out a framework that would allow for future growth. And this is why there are so many macros there.

However, in retrospect I have seen that I did a bad job of architecting this thing called #DD. I tried to get a little too sophisticated. In this macro, I generate the text units in the constant area then move them to the DSA prior to invoking SVC 99. This was the wrong approach, and it complicated the macro more than was necessary. My reasoning at the time was to limit the code needed at runtime, and perhaps saving a base register. Today, with a 20-bit displacement reality, I realize that I don’t need to do that. So, if I generate everything at runtime with literals instead of constants, I can simplify the macro significantly. So, at some point I’m going to have to rewrite this macro.

At this point in it’s evolution, the macro does not provide anywhere near the functionality that I envision for it. All you can do is to dynamically allocate a SYSOUT dataset. And there is only one place where it is currently used, in RDHTRACE when a subtask needs to produce trace output.

At some point I will enhance this to allow for dynamically allocation physical datasets, PS, DA, PDS and PDSE’s, and to support the allocation of new datasets. But for now, it is what it is. This is how it’s used in RDHTRACE:

```
IF (TM,TRCPFLAG,TRCPDDSO,O)   If We're Using SYSOUT
   IF (C,R1,NE,TCBJSTCB)         If We Are NOT Job Step TCB?
      MVC TRCDDNAM,=8C' '        ..Space Fill Receiving Field
      #DD ALLOC,                 ..Allocate a Sysout Dataset  *
      SYSOUT=A,                  ....Default Sysout Class     *
      FREE=CLOSE,                ....Deallocate at Close      *
      RTDDN=TRCDDNAM             ....Save DDNAME for Open
   ENDIF
ENDIF
```

2.7.1 #DD Syntax

The syntax for the #DD macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#DD	Macro name.
b	One or more blanks.

ALLOC	ALLOC, Type of Request (Positional)
SYSOUT=	SYSOUT Class for SYSOUT Dataset
FREE=CLOSE	Free Dataset Upon Close
RTDDN=	Area to Receive Dataset Name

2.7.2 #DD Parameters

These parameters are described more fully below:

ALLOC

This positional parameter indicates the type of request. ALLOC is all that is currently supported and it means we're going to dynamically allocate a dataset. And in this case a SYSOUT dataset.

SYSOUT=

This parameter specifies the SYSOUT class of the allocated SYSOUT dataset.

FREE=CLOSE

If specified the macro will generate a text unit that will call for freeing the dataset upon issuing a CLOSE macro.

RTDDN=

This parameter specifies the address whene the DDNAME of the allocated dataset is to be returned.

2.8 #SUICIDE Macro

Every now and then when writing code, you will find yourself in a situation which is unrecoverable, a “should not occur” situation. However, you still have to program for the possibility. This is because, “should not occur” situations, do sometimes occur. And this macro was written to deal with those possibilities.

It’s also intended to work in conjunction with #MSG, #WTO and #WTOR. And though that functionality is not presently here, I do intend to allow for issuing a #MSG or #WTO internally at a later date. If it’s in a macro, the source code can be parsed. This will allow us to grab up all the messages in a given system and verify that they are documented. And it also allows for translation into other languages. That’s a future, but the possibility is there.

At present it just takes whatever text message you specify, generates that as a DC and issues an EX 0,*, execute yourself, which is why I called it #SUICIDE. This will generate an S0C3 and the PSW will point to the text of the message, which will hopefully tell the diagnostician why we decided the kill the program at this point.

2.8.1 #SUICIDE Syntax

The syntax for the #SUICIDE macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#SUICIDE	Macro name.
b	One or more blanks.
MSG=	Message text to be generated.

2.8.2 #SUICIDE Parameters

These parameters are described more fully below:

MSG=

This parameter specifies the text of the message to be generated immediately after the EX 0,* instruction. This will be generated in the code as a DC instruction.

2.9 IFNTIOT Macro

This is a macro I wrote many years ago. And yet it is so incredibly valuable. I got tired of writing TIOT search routines to determine if a DDNAME was allocated. So, I wrote this macro to do it all for me. It works in conjunction with the IF/THEN/ELSE constructs that were originated by Concept-14 and then replicated by IBM in ASMMSP. This is how it's used in RHDTRACE:

```
         IFNTIOT DDNAME=TRCPRINT       If TRCPRINT is Present
            L  R9,TRGTRPAD             Get R1 at Trace Point
            L  R9,0(,R9)               Get TRPDSECT Address
*
            JAS R14,LTRC               Find TRCDSECT if its There
            IF (LTR,R8,R8,Z)              And Not Already Open
               JAS R14,OPEN               ...Open a SYSOUT Dataset
            ENDIF ,
            ST R9,TRCTRPAD             Save TRPDSECT Address
*
            JAS R14,TRACE              Process the TRACE Request
         ENDIF
```

2.9.1 SQUEEZE Syntax

The syntax for the IFNTIOT macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
IFNTIOT	Macro name.
b	One or more blanks.

NOT	Positional Parameter
DDNAME=	Name of DD to Search For
DDADDR=	Address of 8-Byte DDNAME

2.9.2 IFNTIOT Parameters

These parameters are described more fully below:

NOT

This positional parameter indicates that this is a negative condition. If this optional parameter is specified, instead of generating an IF (EQ) it will generate and IF (NE). So, it's essentially saying if the DDNAME is not in the TIOT do this stuff.

DDNAME=

This parameter specifies the name of the DD to search for as a self-defining term. If the DD is present the code will be executed, and if not it won't.

DDADDR=

This parameter is mutually exclusive with DDNAME= and specifies the address of an 8-byte data area that contains the DDNAME to search for.

2.10 MFREAL Macro

This macro came into existence when I was investigating the DUCT and the ASTE. You see these areas are in real storage, not virtual storage. The macro name stands for Move From Real and uses the LURA instruction to affect that result.

Understand that there is no serialization provided within the macro. If serialization is required, you must provide it yourself. Also, R0, R1, R14, and R14 are used as work registers by this macro. If you need to preserve these values, you must do that too.

2.10.1 MFREAL Syntax

The syntax for the MFREAL macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
MFREAL	Macro name.
b	One or more blanks.

TO	The location to move the data to (Positional)
FROM	The Location to move the data from (Positional
LEN=	The length of the data to move.

2.10.2 MFREAL Parameters

These parameters are described more fully below:

TO

This parameter specifies the virtual address where the data is to be moved to. It needs to be on a fullword boundary.

FROM

This parameter specifies the real address of the data to be moved. This also needs to be on a fullword boundary

LEN=

This parameter specifies the length of the area to be moved. It must be a multiple of 4-bytes, as it uses LURA nd ST to affect the result.

2.11 SQUEEZE Macro

The SQUEEZE macro is used in a few places. It's used in RDHTRACE to squeeze out multiple blanks from the MSG= parameter. It's also used in the #MSG, #WTO and #WTOR macros. The purpose is to remove all duplicate blanks in a message.

2.11.1 SQUEEZE Syntax

The syntax for the #SQUEEZE macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#SQUEEZE	Macro name.
b	One or more blanks.

BEG=	Beginning Address of Message Area
END=	Ending Address of Message Area Plus 1
MF=E/L	Executable or List Form
P=SQZ	Label Prefix

2.11.2 SQUEEZE Parameters

These parameters are described more fully below:

BEG=

This parameter specifies the address of the beginning of the message area to be squeezed.

END=

This parameter specifies the address of the end of the message to be squeezed plus one.

MF=E/L

I don't like this parameter. But the macro was written prior to my having defined the LOCTR structure. So, there it is just the way IBM would have written it.

P=SQZ

This parameter specifies the prefix to be used for labels in the generation. It needs to be usique if more than one SQUEEZE macro is used in a give assembly.

Chapter 3 – The #MAC64 Copybook & SSY Macros

#MAC64 is a copybook to provide support macros for 64-bit programs. The SSY macros are also included in this copybook which is generated via the PGMNTRY macro. There are several macros in this copybook that once copied in operate as inline macros for the program. These macros are available once the first PGMNTRY macro has been expanded within a given source module.

3.1 64-Bit Support Macros

The following macros are provided in support of 64-bit programs that call other programs, either 31-bit or 64-bit. There is a problem with BASR in that while it sets the AMODE bit bit-32 to one for a 31-bit return address, it does no set the AMODE bit, bit-63 to one for a 64-bit return address. This failure will cause a BAKR/PR sequence to return to the 64-bit program in 24-bit AMODE. So, these macros were developed to resolve this issue by assuring the AMODE bit, bit-63 is properly set when a 64-bit program issues a call to a program that returns via PR.

If you feel a need to use these macros, please look at their descripions in #MAC64 in the macro library.

BASPM	- Branch and Save and Preserve Amode
BAS31	- Branch and Save to a Program in that runs in AMODE(31)
BAS64	- Branch and Save to a Program in that runs in AMODE(64)

3.2 The 20-Bit SSY Macros

The following macros were developed to fill in a gap in the Zos instruction set for the SS instructions. Most of the older RX and RS instructions which use a BDDD base and displacement, have been replaced by the Y and G instructions which allow for a BDDDDD base and displacement. However, the SS instruction set, which is very commonly used have not, and cannot be replaced as that would require an 8-byte instruction and the hardware just can't support that.

So, in order to make full use of the 20-bit displacement base registers, allowing direct addressability of a megabyte of storage, it was necessary to provide a replacement for the SS instruction set via these macros.

These macros will generate a sequence of instructions similar to those shown below. They use R14 and R15 as work registers. If you need to preserve these work registers specify SAVE=YES on the MACRO.

```
LAY R14,&A
LAY R15,&B
MVC 0(L'&A,R14),0(R15)
```

I know this is a less than adequate implementation, as it doesn't support all the possible constructs for operands A & B that the assembler supports. To support all those constructs would require writing a parser in macro language and that would take a few weeks. I didn't have the time. But what we have works for now.

If you need to code something like MVCY 0(DSECTD1-DSECT,R9),0(R2) , you will need to code that construction yourself because the macros as written won't understand your desire. It won't get the length right.

The following macros are provided in this copybook.

CLCY	- Compare Logical Character W/20-Bit Displacements
MVCY	- Move Character
MVNY	- Move Numerics
MVZY	- Move Zones
MVOY	- Move With Offset
XCY	- Exclusive Or Character
NCY	- And Character
OCY	- Or Character
TRY	- Translate
TRTY	- Translate and Test
APY	- Add Packed
CPY	- Compare Packed
DPY	- Divide Packed
MPY	- Multiply Packed
SPY	- Subtract Packed
SRPY	- Shift and Round Packed
ZAPY	- Zero and Add Packed
EDY	- Edit
EDMKY	- Edit and Mark
PACKY	- Pack
UNPKY	- Unpack
MVCKY	- Move With Key
MVCPY	- Move to Primary
MVCSY	- Move to Secondary
MVCDKY	- Move With Destination Key
MVCSKY	- Move With Source Key
MVCOSY	- Move With Optional Specifications

Chapter 4 – ATB Getmain & Freemain

I had gotten to a point where I felt that this little system was ready to distribute. I had finished the SSY macros for 20-bit displacement support, I had the TRAP2, BASR and SVC implementations. It was all looking good.

Then I took a step back to gawk at my work. And it was then I realized something tragic. It wasn't good enough. At least it was not good enough for me to put out there as a sample of my work. I didn't feel that it was complete, as a 64/20 system. You see, I had the stacking DSA, the stacking ATB and that was all working fine and it looked good.

But the problem that I saw in reflection was that in order to be a fully functional 64-bit system, we needed to be able to issue GETMAIN and FREEMAIN for above-the bar storage in increments of less than one megabyte. And that capability is simply not there in z/OS. IBM hasn't written it.

If you're going to put your data above-the-bar, you need to be able GETMAIN 64-bytes, or 256-bytes, or even 4K of storage, and manage those GETMAINS and FREEMAIN properly, so as not to waste storage. And this capability does not exist in z/OS at the moment. And I don't know why, they've had ten years to do it.

So, I was a little despondent when I realized that this system would not be complete without that functionality. I was a little daunted the day I set out to write this code. But what was the alternative?

So, I pulled up a blank screen in the ISPF editor and started typing. What do I need, I asked? I need an anchor DSECT. Then I need a place to anchor it, so I noted that I would need an RME (Resource Manager Exit). And I continued to do the next thing, and then the next thing, and then the next thing. When I started I had no idea how long this would take. But I knew I could not distribute and feel good about it, without this piece.

I kept slogging along, as it began to take shape. In the end this program I wrote called RDHGATB turned out to be 1,700 lines of code. The programs to test the code were another 1,000 lines of code, for a total of 2,700 lines. The test programs were necessary to assure that every line of code had been tested thoroughly. You will find them in the TST library. The project was completed in five days or 40 hours, and fully debugged to GA level code.

The SVC used for this service is defaulted to 167, which is what I use. But there is a parameter in both of these macros (#GETATB & #FREATB), that will allow you to change the SVC number. This storage is obtained on 32-byte boundaries instead of the normal 8-byte boundary for below-the-bar storage. I did this on purpose, because when looking at a dump the storage will always be on an X'20' boundary, or beginning on the left side of the dump.

There is support for a BASR implementation for authorized programs which is essentially a branch entry interface, though it's not a true branch entry, as it requires statically linking the code into the program. The problem there is that I need to provide that kind of support for true branch entry in SVCNTRY and I haven't done that yet.

Storage is all task related. When the task goes away, all the storage acquired under that task also goes away. I do not support subpools, as I think it's a dumb idea, but I do support the acquisition of storage in any or all possible storage keys.

Conditional #GETATB is supported, though it's probably not necessary unless you get a runaway GETMAIN. So, you have the R, RU and RC forms of the macro.

4.1 The #GETATB Macro

The #GETATB macro allows the caller to obtain keyed storage from a pool of storage above-the-bar. Storage is task related and released by the system at task termination. We obtain one megabyte for storage management control blocks plus one megabyte of storage for each key of storage requested. This system can support 22,000 separate areas of allocated or free storage per task in the address space. If this is exceeded an ABEND will result.

The 64-bit address of the storage acquired will be returned in GR1 if the request was successful.

4.1.1 #GETATB Syntax

The syntax for the #GETATB macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#GETATB	Macro name.
b	One or more blanks.

R,RU,RC	Type of Request (Positional)
LV=	Length of Area Requested
KEY=	Key of Storage to Obtain (0-15)
SVC=167	SVC Number to Generate
BRANCH=NO	Wether Branch Entry is Requested

4.1.2 #GETATB Parameters

These parameters are described more fully below:

R, RU, RC

This parameter indicates the type of request. R and RU indicate unconditional requests and RC indicates a conditional request. For conditional requests, if the request could be satisfied GR1 will have the 64-bit address of the storage returned and GR15 will contain zeros. If the request could not be satisfied GR1 will contain zeros and GR15 will have a return code of 8. Unconditional requests that cannot be satisfied will generate an ABEND.

LV=

This parameter specifies the length of the area requested. It will be rounded up to a 32-byte boundary. It may be specified in register notation (i.e. (R1)), as a self defining term, or in the form (*,LABEL) to indicate value resides in a fullword or doubleword area of storage. In the latter case the assembler length value will determine the code generated.

KEY=

This parameter specifies the length of the area requested. It will be rounded up to a 32-byte boundary. It may be specified in register notation

SVC=167

This parameter indicates the SVC number to use for generating the #GETATB request. Currently IBM is not using SVC 167, however that may change in the future so this parameter is in the macro to quickly resolve any potential future conflict.

BRANCH=NO

This parameter assumes that an SVC request will be generated. However, there are times when an SVC cannot be used. So this provision allows for a type of branch entry to the support code that services #GETATB. It will generate a load of a VCON and a BASPM.

4.2 The #FREATB Macro

The #FREATB macro is coded to free storage previously allocated by the #GETATB macro. Partial FREEMAINs of an area of gotten storage are not supported. The same amount of storage that was previously gotten must be specified on the #FREEATB macro, rounded to the next larger 32-byte boundary. Failure to do so will result in an ABEND.

4.2.1 #FREATB Syntax

The syntax for the #FREATB macro is as follows:

label	Label: Beginning in column 1.
b	One or more blanks.
#FREATB	Macro name.
b	One or more blanks.

R,RU	Type of Request (Positional)
LV=	Length of Area Requested
A=	Address of Area to be Freed
KEY=	Key of Storage to Obtain (0-15)
SVC=167	SVC Number to Generate
BRANCH=NO	Whether Branch Entry is Requested

4.2.2 #FREATB Parameters

These parameters are described more fully below:

R, RU

This parameter indicates the type of request. R and RU bith indicate unconditional requests, there is no significant difference between the two. It's just what you're used to coding.

LV=

This parameter specifies the length of the area to be freed. It will be rounded up to a 32-byte boundary and must be the same length as originally requested. It may be specified in register notation (i.e. (R1)), as a self defining term, or in the form (*,LABEL) to indicate value resides in a fullword or doubleword area of storage. In the latter case the assembler length value will determine the code generated.

A=

This parameter specifies the 64-bit address of the area to be freed. It may be specified in register notation (i.e. (R1)), as a self defining term, or in the form (*,LABEL) to indicate value resides in a doubleword area of storage.

KEY=

This parameter specified as a self-defining term is the storage key of the area to be freed and must be the same key as that originally requested.

SVC=167

This parameter indicates the SVC number to use for generating the #FREATB request. Currently IBM is not using SVC 167, however that may change in the future so this parameter is in the macro to quickly resolve any potential future conflict.

BRANCH=NO

This parameter assumes that an SVC request will be generated. However, there are times when an SVC cannot be used. So this provision allows for a type of branch entry to the support code that services #FREATB. It will generate a load of a VCON and a BASPM.

Chapter 5 – Desirable Future Enhancements

There is so much stuff that I would like to add to this Development Platform. But you have to understand that anything I could add would be ancillary, which is to say extra bells and whistles. The basic stuff is all done, that being the traces. That doesn't mean that they're not useful. Bells and whistles can bring a great augmentation to the basic system. And this is what I'm proposing for the future and what I'm hoping to provide as I move forward with this product.

Most of the enhancements that I have in mind will not be put into the GNU-GPL version of the product. If I have my way, these will only be made available in the Commercial Version. And yet, there are some that I might make available there. It depends. And of course, bug fixes will be ported down.

5.1 More Entry & Exit Macros

There are several more macros that I have intent to add to this system in the future, but have not yet written. These need to be there because they represent slightly different entry and exit protocols, and include ESTNTRY and ESTEXIT to support ESTAE recovery routines. These are needed because, while the current PGMNTRY and PGMEXIT will work for an ESTAE Exit Routine if the exit routine has an SDWA, that approach will break if there is no SDWA available. So, that's a problem, and I need a new protocol that will handle the situation where there is no SWDA.

Then we have PCNTRY and PCEXIT to support traces within a PC, be it stacking or standard PC. This can only be supported via the TRAP2, because you cannot execute an SVC under a PC. So, that's a really necessary enhancement. I can do it, it will just take a little time.

And then there's the SRBNTRY and SRBEXIT to support the generation of trace output from within an SRB, which can only be done with a TRAP2 implementation. This is a really cool possibility, because TRAP2 will work in an SRB and getting trace output within an SRB is a first. And I would love to be the first one who does that.

I'm sure you can see how the addition of these macros would make the system even more robust than it is today, especially the PC and SRB support. And given what I already have, it wouldn't be that much more work to produce these interfaces. It would be at most two weeks per interface, maybe less.

This will probably not go into the GNU-GPL version of the code. Most people don't write PC's or SRB's.

5.2 Rewriting ASMMSP

IBM did a bad job of writing ASMMSP when they tried to replicate the CONCEPT-14 macros. They're functional, but still a schlock job. I can do better. What I need to do is to add my traces to this functionality. I want to see IF/THEN/ELSE traces DO/ENDDO traces and SWITCH/CASE traces, or at least have the ability to show them on demand. So, this means rewriting ASMMSP or providing a replacement copybook. It's not that big of a job, perhaps two weeks, but it would be a great augmentation to the system as a whole and pretty easy to do.

This may or may not go into the GNU-GPL version of the code, though I'm kind of thinking it should.

5.3 Filtering Support

Filtering is key and must be provided in any Commercial Version of this platform. It's not really marketable without filtering. It's not a problem when you're writing a single program you get all the traces you need. But it does become a problem when you write a 50 program, or a 100 program, or a 1000 program system.

In diagnosing problems in a system you need to be able to limit the traces to the programs in error. You want to see PGMNTRY/PGMEXIT for the programs calling the program in error, SUBNTRY/SUBEXIT, and IF/THEN/ELSE traces for the program in error, just before the ABEND. And you need to be able to specify the programs you want to trace by name, and specifically what traces you want for any given program. And if we don't have that ability, you will be inundated with hundreds of thousands of lines of trace data that you don't need and that will get in the way of diagnosing the problem.

So, for a Commercial Version where you're writing a real system, filtering must be there.

It's not there yet, but it wouldn't be hard to do. What I need is a language, a Parser and a TRCSYSIN dataset. I need to have some way for the customer to easily indicate what they want to trace and ignore everything else. It's not a big project, maybe three weeks. But it might be four if I decide to make the parser a macro based system in its own right and add that to this development platform. And that makes more sense to me architecturally.

Filtering will not be added to the GNU-GPL version, it's Commercial Version only, but the Parser might be, if I choose to write it that way and I'm definitely leaning in that direction.

5.4 The @TRACE Macro

The @TRACE macro requires a little explanation. So, I'm going to tell a little story about how it came to be.

I was working for Serena Software and I had rewritten this program called CMNASIST. This was a program that would parse all the programs in a system looking for COPY statements and note them in a database. These were then used in CMNAUDIT to note what copybooks were invoked by what programs. Anyway, this program was single threaded and at CitiCorp the program would run for 21-hours, parsing some 200,000 programs. Oops. So, I had converted it to use multi-tasking with up to 99 subtasks, though the default was 16. This cut the time to run to about 3 hours, which was a good accomplishment but still a long time. Still it was 200,000 programs, so it was not bad.

Then came the problem. It was a SEV1 situation, CitiCorp had deleted their file and when they tried to run CMNASIST again it ABENDED half way through. Bummer, Dude. Then I got the call, because it was CMNASIST that failed and that program was my responsibility.

After the first day I realized that it wasn't my code that was in error. It was this other program that failed called CMNPARSE that was called by CMNASIST, which was another spaghetti-code program that was invoked by CMNASIST. The problem was that CMNPARSE did not use my PGMNTRY and PGMEXIT macros. So, I could not use the $TRACE macro as it is called for the version of this code that I wrote for Serena.

It wasn't really my problem, but I was the man on the hot seat and I needed to figure out what the issue was. So, I conceived of this thing I called the @TRACE macro. This was a version of the $TRACE macro that would work in foreign code. It had a larger footprint than the $TRACE, maybe 60-80 bytes rather than 24, but I wrote it and made it work and I did it in four hours. Necessity is the mother of invention, as they say. I needed this functionality, and I could copy most of it from the $TRACE.

The bug was in CMNPARSE and I didn't know anything about that program. And I needed help. So, that's how @TRACE came to be. It's a version of $TRACE that would work in any program, anywhere. To conclude the story, that problem went on for six days, and was a SEV1 CritSit (Critical Situation). ChangeMan was a mission critical product. And during that time 200 programmers could not do their job, because our product had failed. They could not put their new versions of many different systems into production because of this issue.

I was working non-stop 16/7. One problem was that I was in Maui and the customer was in New York, it was an eight hour time difference. The boy on the other end was great and very helpful. He was working all night long. But it took three hours for every test, which is why it took so long to resolve the issue. I would add trace points and ship the object module via e-mail. He would relink and run the test and ship me back the trace data. And in this way we got closer and closer to the problem.

About the time I finally narrowed it down and felt I had discovered the problem, I received a phone call from my Bosses Boss, Kevin Parker. Our CEO had received a call from the CEO of CitiCorp. Not Good. He then called Kevin, who then called me. Now folks, if you're a Developer on the Hot Seat, you never want to be called up by the VP of Development. But I was that day. Kevin wanted to know where I stood. He was not happy. Fortunately, because I had written the @TRACE on the fly while in the middle of a fire-fight, I was able to tell him that I think I had resolved the issue and this next test should prove it. Two hours later I received confirmation, and I called Kevin back to give him the good news. I skated out of that CritSit by the skin on my teeth, and it was the @TRACE that allowed me to resolve that situation.

The problem turned out to be a PL/I construct that CMNPARSE could not understand or comprehend that some programmer had coded in one program. I believe it was a syntax error. And when we identified the program and removed it from the library, all went well.

I hadn't known that the @TRACE was possible prior to that experience. I just needed it, and so I created it on the fly. But this is another enhancement that I would really like to see in any commercial version of this product. It has a heavy footprint, because I don't have the LOCTR structure, and it has the propensity to blow base registers. But it can be used in any assembler program and provides all the same functionality as the #TRACE macro. And this functionality is worth paying for.

5.5 Messaging Services

Messaging services is also really important to have in a commercial version of this product. This is really an extension of what I already have in #MSG, #WTO and #WTOR. I talked about it earlier in the User's Guide.

What we need here is to provide the ability to translate messages into languages other than English. That requires a few things. It requires a parsing program that will read in all programs in a system and build a file of all messages in the system. This can be useful to verify that all messages in the system are documented. But it also becomes the basis for translating messages into foreign languages.

This file once translated, can then be input to the system and used to replace the English message coded into the programs for any given customer. The RDHMSGS program would have to be modified to read in that file and replace the generated text with the file text. But it can match on message number, and that's why I wrote it that way, to provide a foundation for this future enhancement which I think is extremely valuable for multi-national development companies.

5.6 Parsing Macros

This is something I have to do anyway. I need this functionality in order to provide filtering support. I have to produce a language that will allow people to say what they do and do not wish to trace. That means I need a system that will provide a similar functionality to what macro language provides. I need verbs, I need positional parameters and I need keyword parameters.

I expect that the best way to provide that functionality is though a series of macros that then call a service routine that will perform the functions necessary.

If I do it this way, then the whole thing can be shipped as an augmentation to this system, and not just used by the system. Anybody could use it. And that's always a better design.

I don't see this as going into the GNU-GPL version, but it could certainly be shipped as a piece of the commercial version of the product. And it would make a great enhancement.

5.7 Sort & Search Macros

I love the idea of providing these sort and search macros. It would have to be a binary sort and a binary search, of course. I could never lower myself to write a bubble sort or search. But I think this would be a wonderful augmentation to this platform and it's not very hard to do.

I think they should handle eight different disjointed keys and it needs to be able to access data in data spaces and other address spaces. And they also need to be able to work with tables in above-the-bar storage.

It might take a week to write each of these macro systems. I'm not sure how to write it yet. With eight different disjointed keys it might be best to write it as a macro that then calls a service routine. We shall see.

5.8 BTREE Support

This is another war story. When I went to Serena they had a big problem with Audit. It could take as much as 21 hours to run Audit at CitiCorp with 200,000 programs , and Audit was my responsibility. At issue were the number of times that we had to search a database that had only two columns in it; programs and copybooks. Twenty-one hours was ridicules. So, I set out to resolve the issue.

I created this wonderful solution, a BTREE. Originally it placed the data, some 600,000 entries in above-the line storage, though I had intended to put this data in a dataspace with DIV backed by an LDS. I wrote the code in a week, it was about 5,000 lines of code, a beautiful BTREE solution. It was generic, it could handle eight different alternate indexes, well a primary index and seven alternates. It was way cool, a wonderful piece of work.

So, I told management that this was how I decided to resolve the issue. They then ran it by Marketing. And Marketing shot it down. They said no, we don't want it. The reason they didn't want it was that this solution would make obsolete the DB2 solution which was already for sale. The DB2 solution was a $20,000 option that provided $5,000 in maintenance revenue per customer per year that had DB2. And if I obsoleted that option, that maintenance revenue would go away and the company would loose money. Oops.

But I still had a problem. The customers that didn't have DB2 were still complaining about the performance of Audit. And that anger needed to be addressed. On the one hand, I was required to resolve the issue, but management was also saying don't fix it too well. So, the code went into the shredder.

I did resolve it eventually by adding a secondary index with a simple binary search, which was probably just as efficient as the DB2 option. It manifested a 50% performance improvement in the product, going from a bubble search to a binary search over 600,000 items. We just didn't tell the customer base.

But I really would have preferred the BTREE, though. It was a much better solution. Sometimes what's best for the product is not what's best for the company, and as a developer you need to understand that.

But this kind of functionality would be a wonderful augmentation to this platform. Of course, it would need to work with data above-the-bar and data in other data spaces or address spaces, which means ARMODE compliant, but that's no big deal.

It took me five days to write the original version, though it was not fully debugged. I think I could do it again in two weeks or less.

5.9 DIV Above-The-Bar

Oh, my GOD! This is something that should give any *real* Developer an erection the size of the Washington Monument, just the opportunity to do something like this. I conceived of it last year when I wrote an ASCRE, address space create.

The problem is that IBM's DIV (Data in Virtual) support is limited to a data space. And a data space is limited to two gigabytes. But DIV-ATB can be a thousand gigabytes. And it's not really a difficult task. You just need a servicing address space, an ASCRE. Then you start at the 2-gigibyte line and move up from there. It's important to have programs that can access ATB storage and run in ARMODE, but that's what I have here.

You can't back it by an LDS, because that's an OCO access method, but you can certainly back it by a DA dataset. All you need is full-cylinder I/O and everybody knows how to do that. Well, at least I do. It's no big deal. You have a servicing address space that wakes up on a STIMER, runs the ALEs in the ASTE searching out pages that have been updated and writes that data to disk. Then you have an RME that gets control at EOT and finalizes the updates. What could be simpler than that?

DIV-ATB is not something that most people would use. But it would be really cool, and I'm sure a few people would make great use of this functionality. And it would be a great augmentation to this development platform.

My best guess is that it should take about three months to produce.

5.10 Merlin

This requires a bit of story telling. MERLIN came into existence 25-years ago when I was working for Family Life, a wholly owned subsidiary of Merrill Lynch in Seattle. I was working on the Insurance System called Cypros. It was an all Assembler system.

I was working a lot in the CICS component of the system, and I found that I could use IMASPZAP to make simple fixes, like an LA that should have been a LH. In these cases I could do a simple refresh of the load module, and not have to take CICS up and down. This practice saved me five minutes here and five minutes there, and I was more productive as a result. I would simply zap in the fix, refresh and retest. Cool, it worked.

But one day I went to work in the early morning set off an IMASPZAP and received an S806. And I thought to myself, "What the Hell!" I went searching through LPALIB and LINKLIB and IMASPZAP was nowhere to be found. Someone had found out that I was using it and took it away.

That really pissed me off. I was livid. I sat there all day, sulking, angry, and resentful. Then I thought to myself, IMASPZAP is not an authorized program. If they were going to take it away, I can easily write my own. I don't need them, the Sysprogs in this shop. I'm better than they are.

So, that's what I did. This was a god damn Insurance Company for Christ's sake. The expectations were so low that a neophyte could easily meet management expectations. It was a place where I had to play down in order to not alienate my coworkers, but that's another story. So, I had plenty of time on my hands. I only had to work two maybe three days a week to consistently exceed management expectations.

Anyway, I was angry at the loss. So, I stayed late that night. And I sat down and began to write an online interactive version of IMASPZAP. It used TSO FULLSCREEN I/O, provided the ability to overtype the screen, hit the <ENTER> key and the update was immediate, for any member in any PDS load library. It was really cool. It took ten days and I still did my job on top of that personal project.

Then I wrote Core Zap and DASD Zap to add to this system. I couldn't do the full functionality at Family Life because I didn't have an Authorized load library. So, I polished it up when I went to Sterling. At IBM this last year I added support for the JES2 data spaces to the core zap functions, and MTTR addresses to the DASD Zap function.

So, that's what MERLIN is; Core Zap, PDS Zap and DASD Zap and it's a Security Administrators worst nightmare. With Core Zap I can see anything happening in any address space in the system. I can see all the data in any JES2 data space. And I can modify that data as pleases me. With DASD Zap I can see any block of data on any disk pack allocated to the system. But not only that, I can also modify any block of data on any DASD devise attached to the system by simply overtyping the screen and hitting <Enter>.

It's a very powerful system, but also very frightening to any security administrator, precisely because in its present form there is no tracking and no records kept of changes made or the people who made them. And this is why I'm not shipping it at the moment, even though it's done and complete. It's mine and I think I'll keep it close to my chest for the time being.

If I have time to put some tracking capabilities into it I may release it, as a part of this product. It needs a Log file, and one that can't be modified by DASD Zap. It is a very powerful tool and very useful in the right hands. But I need to provide accountability and tracking to satisfy the Security Administrator before I can release it. And I don't think I'll be shipping the source code on this one, OCO only.

5.11 AMODE(64,ANY) RN RU Support

The reason for doing this is that the current system is a bit of an embarrassment.

I'm representing this system as a 64/20 Development Platform, but there is only one program in the system, RDHGATB that is truly 64/ANY RN RU. And while that really doesn't matter with respect to the functionality provided, it would be nice if when somebody looks at the ISPF panel for the load library, it showed that all programs were link-edited as 64/ANY RN RU programs.

The current system works just fine linked as 31/ANY RN RU programs. But it's the image, you see. I could do this in a week. It just means making some minor changes and then retesting everything. And realistically, there is no real benefit to be gained here, no new functionality. It's just about how the system is perceived. But, I'll tell you this folks when you see a program displayed on that ISPF screen for the load library display as 64/ANY RN RU, that's really cool. And to have all my programs represented that way would be very impressive.

Internal Specifications

How it's all put together.

You'll need this information, Fellow Warrior,
As, I might not be there to support it for you,
You may just have to depend on your Self.

Introduction

This is a pretty cool system. It's only 17,000 lines of code plus 10,000 lines of test programs. It's not that much code, not really. But it is some very sophisticated code. Not very many people on this planet write code at this level. I'm not being arrogant, though I have the right to be. I'm just being realistic.

I don't think there are more that fifteen people in the whole world who have written a DUCT trap routine. The information is all there. I scrubbed it out of POPS, though it took four months. It's a defined interface, though not well defined. The information you need to put this together is in about four different places in POPS. And you have to be tenacious to pull it all together, so that you can actually write the code. It was definitely not easy.

The BASR implementation, that was easy. The SVC implementation was not difficult. But the TRAP2 implementation, well, that took some serious work. It wasn't hard to write after I understood how it was put together, but what was hard was sleuthing out how to do it in the first place, pulling it out of POPS, and understanding the DUCT, the Dispatchable Unit Control Table.

Anyway, it's done. And this work has laid the foundation for other things I can do in the future, the most important of which is Lazerine.

This document is here to tell you how this system is put together. It's a technical reference document and is intended to tell other developers how to maintain the code. This might be you. I don't know if I'm ever going to be able to produce a Commercial Version of this code. I need a support company to do that. And I don't know if there will be any company that will support me in producing that Commercial Version.

And if I can't, you may be on your own. You may have to support it yourself. So what I'm doing here is telling you how it's put together, all the why's and wherefore's so that if you happen to run into a bug, or want to make an enhancement, you will be empowered to do what you need to do.

If I am lucky, and I find a support company to sponsor the Commercial Version, well, this will tell other developers how to maintain the code.

Chapter 1 – The Three Implementations

I'm going to explain this from a top-down point of view, even though it was written from the bottom up. It makes more sense that way. Though, how it came to be was through an evolutionary process.

The first implementation was the BASR implementation, which is what I still primarily use for my own purposes in debugging and diagnosing problems within the system. And I'm sure there are still bugs in this system. There is no such thing as bug-free code. The BASR implementation is easier to work with for this purpose, because it's statically linked into whatever program you're running, while the SVC or DUCT implementations require the reinstallation of the SVC to make any changes. And while that's not a problem because I can dynamically reinstall the SVC, it does require an extra step.

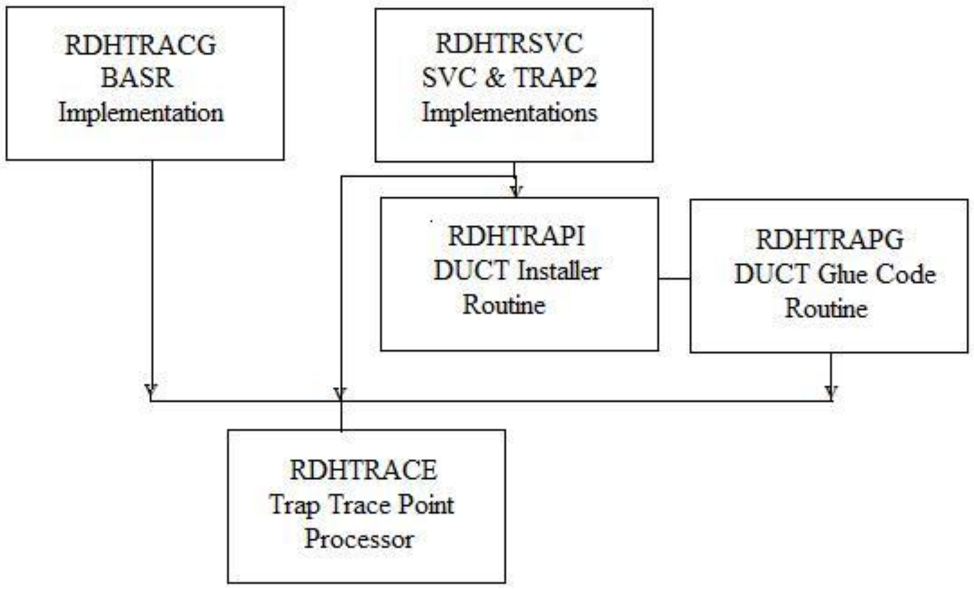
The problem with the BASR implementation is that it can only be used by an Authorized Program loaded from an Authorized Load Library. This is because RDHTRACE, which is the program that processes the trace point trap, needs to perform functions that can only be done by an Authorized Program.

So, that's why I produced the SVC version, which was the second implementation. This version can be used by any Key(8) Problem State program. This is because the code in the SVC is Authorized, even though the calling program is not. I use SVC 166 for this purpose, but you can use whatever SVC you like. See the installation instructions for more information on the SVC installation.

The TRAP2 implementation is the most powerful and the last to be devised. The reason that it's more powerful is that like the SVC implementation, it can be use by Problem State programs. It can also be used where SVCs cannot; in a SRB or a PC (at least I believe you can't use an SVC in a stacking PC). Though, the full implementation is not yet there because I have yet to write SRBNTRY/SRBEXIT and PCNTRY/PCEXIT. Oh well, I'll get there at some point. But the base code is there and available. The full functionality will come in time.

So, we have three functional implementations for this system. The TRAP2 implementation is tied to the SVC implementation. It uses the SVC issued within the PGMNTRY TYPE=MAIN macro to install the DUCT trap routine. This routine then gets control whenever a TRAP2 instruction X'01FF' is executed which will happen at every trace point subsequently executed. This is because we use an EXecute instruction to execute the SVC the TRAP2 or the BASR instruction. And these instructions are passed down through propagated DSA to all subordinate programs in the system being tested.

A picture is worth a thousand words, so they say. Below is a diagram of the flow of control that shows how these three interfaces to RDHTRACE have been architected. No matter what the implementation, they all end up at RDHTRACE where the trace point is processed. The actual interface programs are pretty small, a few hundred lines at most. The bulk of the work is done in RDHTRACE. Their function is to set up the DSECT that is passed into RDHTRACE, so that it has a standard interface no matter how it is entered. He doesn't need to know if it's BASR, SVC or TRAP2. He just does his job.



RDHTRACG is the glue code for the BASR interface program, a doorway to RDHTRACE. In the BASR implementation we load up a VCON and execute a BASR instruction. This forces RDHTRACE to be linked into the program. The downside of this implementation is that the test program must be authorized. The upside is that we don't have to reinstall the SVC. So, for me when I'm doing maintenance of this system, this is my preferred method.

RDHTRSVC does a couple of things. First it's the entry point that gets executed when someone executes the tracing SVC. It's usually going to be SVC 166, but that's just my recommendation. It could be any SVC. There is code in PGMNTRY that will search the SVCTABLE from SVC 255 to SVC 0 to locate the tracing SVC RDHTRSVC. The code knows the protocol and can discover the proper SVC to use.

The IMP=SVC glue code is different than the glue code for BASR and TRAP2 implementations, in that these latter implementations get control at the point of the trap and the first thing they do is to push an entry onto the linkage stack to save the current status so it can be reinstated after the trap. The SVC implementation does not do this, the status is stored in the SVRB and the XSB so in order to keep the environment the same from the point of view of RDHTRACE, who does not know how he was entered, we build a stack entry the way it would have looked just prior to the execution of the SVC. This way the code in RDHTRACE does not have to be aware of how it was entered. Upon return from RDHTRACE we pop the linkage stack entry and return to the caller via the standard SVC path, which is an SVC 3.

The second thing that RDHTRSVC does is to install the DUCT Trap Routine if IMP=TRAP2 was specified on the PGMNTRY TYPE=MAIN program. Another program RDHTRAPI is statically linked into the SVC and this program when called upon will install the DUCT trap routine for the given TCB. The DUCT Trap Routine itself is a second CSECT in RDHTRAPI called RDHTRAPG, the DUCT Glue Code Routine. RDGTRAPG will then get control whenever a TRAP2 is executed under that TCB, where it sets up the environment and calls RDHTRACE.

All of these programs create this data area called TRGDSECT, which contains the LSEDSECT, the Linkage Stack Entry, the Control Registers, the TCB and ASCB addresses. This is all the information that RDHTRACE needs to perform whatever needs to be done for that trace point.

Chapter 2 – RDHINSTL - The SVC Install Program

RDHINSTL is a program that will INSTALL, REMOVE or REPLACE an SVC dynamically, which is to say while the Operating System is up and running. It can be used during IPL to install any SVC in a production environment. But it's most useful when debugging an SVC that you have written. This is because you don't have to IPL to get the SVC into the system. You can just reinstall the new version and rerun the test. And this is very useful when you're writing a new SVC.

IBM does not use SVCs that much anymore. This is because an SVC cannot be used in an SRB, or a stacking PC. So, they have moved to the practice of using Stacking PCs instead of SVCs. So, there are plenty of SVCs available to do whatever you want. I'm using the 160 series for my purposes, but you can use whatever you want as long as it's available.

One of the things that this program does is to determine whether an SVC is available or not. It does this by getting the address of the entry points of all SVCs in the SVCTABLE. Then we get a count of how many times that address is pointed to by all SVCs in the SVCTABLE. You see, when an SVC is not in use, it points to an address that if executed it will generate an ABEND. So, all the available SVCs will point to that address. And when we determine that address, we can know that the SVC is not being used and we can use it. If you attempt to install an SVC on one that is not available, the program will refuse the request and issue a message.

The program is very simple. It uses the PARM= on the execute statement to determine the function to be performed. A sample of the proper JCL looks like this:

```
//*DUMP    EXEC PGM=RDHINSTL,PARM=DUMP
//*REMOVE  EXEC PGM=RDHINSTL,PARM='REMOVE,RDHTRSVC,166'
//*REPLACE EXEC PGM=RDHINSTL,PARM='REPLACE,RDHTRSVC,166'
//*
//INSERT   EXEC PGM=RDHINSTL,PARM='INSERT,RDHTRSVC,166'
//STEPLIB  DD  DISP=SHR,DSN=RDH.$TLS.LOD
//TRCPRINT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
```

The DUMP parameter indicates that you want to see the SVC table printed out. It will be written to SYSPRINT. This is useful in determining whether a given SVC is available or not. This same report is also printed out after a INSERT, REMOVE or REPLACE for verification purposes.

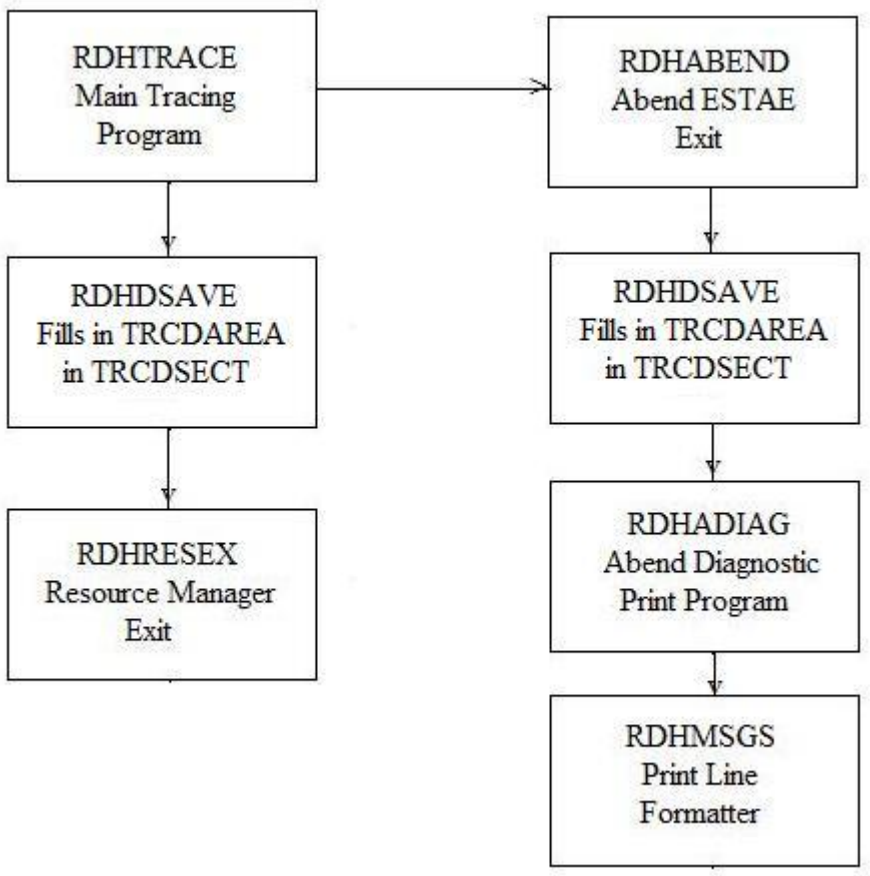
The INSERT, REMOVE and REPLACE requests have three parameters, the command, the load module name of the SVC and the number of the SVC to use, as you can see illustrated above. The SVC load module will be loaded into CSA for an INSERT operation and then hooked into the SVCTABLE. In a REMOVE operation the CSA storage will also be FREEMAINed. On the remove operation, we first remove the entry from the SVCTABLE, then wait 5 seconds before freeing the storage, just in case some task might be using the SVC.

So, you can use this installer to install your own SVCs should you wish to write one. The protocol is defined in SVCNTRY. If you use SVCNTRY and SVCEXIT to write your SVC you can also use RDHINSTL to install, remove or replace your user written SVC.

And incidentally, SVCNTRY and SVCEXIT also have all the same functionality as PGMNTRY and PGMEXIT with respect to the traces. And the trace data for the SVC will also be written to TRCPRINT if the DD was specified in the jobstream.

Chapter 3 – RDHTRACE Program Structure

RDHTRACE is a program with several CSECTs within the source module. The main program is, of course, RDHTRACE which services a trap trace point. But there are other CSECTs in that source module, including the Resource Manager Exit RDHRESEX, and RDHABEND and RDHADIAG which perform processing for ABEND Diagnostics as described in the Chart Below.



3.1 RDHTRACE – The Main Trap Trace Point Processor

The first thing we do in the beginning of this program is to determine if TRCPRINT is in the TIOT via an IFNTIOT macro, if it's not we simply exit and ignore the request. In this way the traces in the system are dormant and not executed. Though, this is realistically just a second line of defense. If TRCPRINT has not been allocated, PGMNTRY will not initiate traces, and instead of executing a BASR, SVC 166, or TRAP2, it will execute a NOP X'0700'. It's important that if tracing was not requested that the traces are truly dormant and do not effect the performance of the system.

So, if tracing has been requested as indicated by the existence of the TRCPRINT DD in the TIOT, then we go to this routine called LTRC. The purpose of this routine is to locate the

TRCDSECT area. This is a GETMAIN area below-the-line that was acquired during initialization processing and hung off an RME, which I will discuss in more detail in a moment.

If the TRCDSECT is found, we continue with processing the request. If it was not found, then it means that we haven't performed OPEN processing for this TCB, so we call this routine to perform OPEN processing for this task.

This routine called OPEN performs a lot of functions required to initiate traces for the task. First it obtains storage for the TRCDSECT below-the-line, which is the primary data area for managing the traces for a given task. Next, it issues a RESMGR macro to create a Resource Manager Exit that will get control at task termination. This Resource Manager Exit is called RDHRESEX. But another purpose for the exit is that we can identify an area to be passed into the exit upon execution. That area is TRCDSECT. And since I can run the RME chain from the TCB, I also have a means of locating TRCDSECT via that method on every call. And this is what LTRC does.

Then if we're not running under the Job Step TCB, we will issue a #DD macro to dynamically allocate a SYSOUT dataset that will then be used to write trace data for this task. And that dataset will be OPENed at that point.

If the PGMNTRY indicated that we should buffer the output, an above-the-line storage buffer will be acquired to store the trace data in a round-robin table. I know, this should be above-the-bar, but that's a future enhancement.

Next we write the initial heading lines.

And last we do something really cool, we set ourselves up to gain control at RDHABEND in the event of an ABEND. This is some slick code, that I think I'll discuss more fully when I talk about RDHABEND, later in this section.

So, that's initialization, or OPEN processing.

Okay after initialization, or during any trace point execution after that point this routine called TRACE gets executed to process the trace point. The first thing it does is to call RDHDSAVE. This program fills in the TRCDAREA that is contained within TRCDSECT which is the primary working storage for RDHTRACE.

Thereafter RDHTRACE calls a different subroutine depending upon the reason it had been called. There is routine PGME deals with Program Entry invocations, PGMX prints diagnostics at Program Exit, the SUBE subroutine processes a Subroutine Entry trace point, SUBX processes a Subroutine Exit, and the TRAP routine deals with generic #TRACE invocations.

These subroutines then use the information in the TRPDSECT, which is generated by #TRACE, the TRCDSECT and the TRCDAREA, the TRGDSECT and the knowledge of what kind of trace point we're processing to determine what kind of diagnostics need to be written to TRCPRINT for this trace point.

I'm not going to discuss each of these routines, you can look at the code and the documentation in the program is sufficient to inform you of what's going on therein.

3.1.1 The Problem with the SL3

When I was putting in support for the 20-bit displacement properties of this system, I ran into a problem with IBM's High Level Assembler. It doesn't support an SL3 Constant. If you code it you will get an assembly error.

I was kind of shocked when I discovered this. Obviously, nobody at IBM has tried to write code that was 20-bit displacement compliant, or the people responsible for the Assembler would have fixed it. Oh, well, I had a problem and it needed to be resolved.

You see the data items that #TRACE generates for either the DATA= parameter or the MSG= parameter are generated into the TRPDSECT area. But the addresses of these data areas are presented as SCON values. In this way we can determine the real address of the data at runtime, which may be in DSA, the constant area, or ATB, or any other piece of storage including dataspaces or other address spaces if the program is running in ARMODE.

And to fully support 20-bit displacements in this system, I needed to present this SCON as a 20-bit offset to any given base, and that requires an SL3. But it doesn't exist and the Assembler does not support that construct. So, I had to get creative, because there is no way I'm going to wait for IBM to fix their bug.

Now, prior to adding the 20-bit support, I had used a standard SCON, an SL2, and the TRPDSECT area for any given trace point was byte-aligned. I was trying to save space, every byte I could because a 4K base register does not give you a lot of room. And at a 24-byte basic footprint for each trace point, you can blow a 4K base register pretty easily. However, it's not such a problem with a one megabyte base register.

So, the way I solved this problem was to use a SLAG instruction within #TRACE to generate an SL3 in DH+DL format. But by using an instruction for generating the SCON I had to give up byte-alignment for the TRPDSECT. This is because all instructions are double-byte aligned. And so, I had to go to double-byte alignment for each data item address generated within the TRPDSECT, which includes the MSG= and DATA= parameter values.

In doing it this way, I lost a few bytes of filler, but gained the one-megabyte base register.

3.2 RDHDSAVE – Build TRCDAREA in TRCDSECT Data Area

This is an important little subprogram that fills in the TRCDAREA that is contained within TRCDSECT, which is the primary working storage for RDHTRACE and is pointed to by the RME.

This area is important because once filled in it contains all the information in the Save Area Trace. This can come from a standard MVS Save Area or from the Linkage Stack Entry in the event that we're dealing with an F1SA. It gets all kinds of information about each entry in the chain, including the Load Module Name, the CSECT name, the offset into the CSECT where the call was made and other related information that can later be used to print calling and called module information to be written to TRCPRINT.

RDHDSAVE is called by both RDHTRACE at any given trace point, and also by RDHABEND in preparation for printing ABEND diagnostics.

3.3 RDHRESEX – The Resource Manager Exit

The RDHRESEX Resource Manager Exit is initiated on the first execution of a #TRACE macro for any given TCB in the address space. A Resource Manager Exit gets control at task termination, whether it is a normal or abnormal termination.

There is not too much that this routine does, it's a pretty small program. For one, it deletes the Resource Manager Exit for this task, which is simply cleanup.

Then if buffering was requested on the PGMNTRY TYPE=MAIN program for the task, it will write the data in those buffers to TRCPRINT. This is important to support the buffering of trace data, which limits the amount of trace data written to SYSOUT. Buffering of trace data will also be crucially important to provide traces for SRBs and PCs, which I will provide at some point in the future.

After that it simply closes the SYSOUT dataset that contains the trace data, and frees the storage for TRCDSECT.

That's it, it can't be more than a hundred lines of code.

But one of the major reasons for having a Resource Manager Exit has nothing to do with performing cleanup at task termination. It has to do with providing an Anchor for TRCDSECT hung off the RME chain which can be run from the TCB.

3.4 RDHABEND – ABEND ESTAE Exit Routine

RDHABEND is the ESTAE exit routine that gets executed in the event of an unexpected ABEND for a PGMNTRY TYPE=MAIN program. In itself, it's a pretty simple program. It sets up the environment by calling RDHDSAVE and then calls RDHADIAG which is the program that actually prints the ABEND diagnostics.

However, the way it gets invoked is strange and very unusual. And this needs a little explanation.

You see I ran into a problem when I was adding ABEND diagnostics to this system. My original design was to simply issue an ESTAEX in OPEN processing for RDHTRACE which is executed upon the first #TRACE macro for any given task. What I found was a shock. I apparently did not fully understand how RTM (Recovery Termination Manager), actually worked. And I discovered an anomaly that was totally unexpected.

What I found was that an SCB (built by ESTAEX) is Linkage Stack related and will not survive over a BAKR/PR sequence. The first time I ran the code my recovery routine was not executed, and I could not understand why. I don't know how IBM does this, but if you issue a BAKR, then issue an ESTAEX and then issue a PR to return to the calling program, the SCB will disappear and will not be executed by RTM in the event of an ABEND. It must be some kind of microcode in the PR instruction, which seems really strange to me.

I've seen some of other people's code that used the BAKR extensively to execute subroutines within a program. But the way RTM works, as shown by my testing, you cannot BAKR to a routine that issues and ESTAEX and expect it to be in affect when you return via PR.

The problem I had was that the first thing I do in RDHTRACE is a BAKR, and I PR back to the calling program. As such, my original design was never going to work, given this revelation. So, I had to come up with another means of producing the desired result.

What I did was something strange. I issue the ESTAEX inside the PGMNTRY TYPE=MAIN program, that does nothing. The ESTAEX is separate from the ESTAEX issued if ESTAE=YES was specified on the PGMNTRY macro. And it simply points to a BR R14 instruction in the Constant Area which if executed without traces in effect, will simply percolate. But the SCB is created and if traces are in effect RDHTRACE will use that SCB to affect ABEND diagnostics.

In the OPEN routine of RDHTRACE, we are in Supervisor State and Key(0). What we do in that routine is to modify the SCB. We change the address of the routine to be executed from the BR14 to the address of RDHABEND. But in addition to that we modify the SCB to indicate that the ESTAEX routine RDHABEND should be entered in Supervisor State and Key(0).

So, that was the solution to the eccentricities of RTM. It's some slick code. I don't usually like to get slick, but it was necessary to resolve this particular problem, and it deserves an explanation.

3.5 RDHADIAG – Program to Print ABEND Diagnostics

This program prints every conceivable kind of diagnostic information that you could possibly want in the event of an ABEND, but not just the current ABEND. It runs the chain of RTM2WA's pointed to by TCBRTWA and prints diagnostics for each ABEND that has occurred in the chain.

This can happen if for instance, when you get an ABEND in a main program that executes an ESTAE routine that then also causes an ABEND. In that event you would have two RTM2WA's in the chain. And this sub-program will print diagnostics for both ABEND's separately one after the other. And this can really help in diagnosing errors in error recovery which are some of the most difficult errors to diagnose.

So, we print out all kinds of stuff to TRCPRINT in this event. We print out all the basic stuff, the PSW at entry to ABEND, the General Purpose 64-bit registers, Access Registers, and the Control Registers.

Then comes the cool stuff, we print out the entire Save Area Chain but not like you would see it in an MVS dump. If it was an MVS save area you get the 18 fullwords, but if it was an F1SA you would get the full linkage stack entry instead. But in addition to that, if the program was a PGMNTRY program, you also get a listing of the full DSA data area followed by the full ATB data area if it was present for that program.

So, when using this system and you receive an ABEND you should have all the information needed to diagnose the problem. And as stated before, this code can be run at the customer's site, and the doc shipped back via e-mail, making supporting the customer much more effective and less time consuming, which I assert will result in a significant cost savings when it comes to maintaining any product written with this system.

I invite you to look at the code. It's a subprogram in RDHTRACE. Additionally, there are some programs in the /TST library that are specifically designed to test out these ABEND diagnostics. So, go ahead and run them, and you'll see what I mean. I'm sure you'll be impressed.

3.6 RDHMSGS – Provides Formatting for ABEND Diagnostics

This program came about a long time ago when I got real tired of having to do MF=E/L for WTO. I wanted to create a solution that would give me the same functionality that I already had for the MSG= parameter of the #TRACE macro. So, one day I just got fed up and sat down and wrote this code. Actually, I didn't have to write much, I just stole the code I had already written from RDHTRACE and created a new program I called RDHMSGS.

This new program is really a service routine that is called by the #MSG macro, the #WTO macro and the #WTOR macro. I never bothered to write #TPUT, I didn't need it.

A little later on in the process while I was writing ABEND Diagnostics for RDHTRACE, I noticed that the RDHMSGS service routine would give me all the functionality that I needed. Please understand, I had to print maybe a hundred different types of lines for the ABEND Diagnostics, and I didn't want to have to code it all by hand.

So, I had these macros that already did all that formatting for me and I decided to use that structure instead of rewriting all that code. That was when I added the PLINE= parameter to the #WTO macro. This then provided the functionality that said we're not going to actually issue the WTO, we're just going to format the data and return the formatted line in a 128-byte data area passed as the value on PLINE.

In this way I was able to code many #WTO PLINE= macros within RDHADIAG instead of coding the actual instructions to do the same thing. I never like writing the same code twice, and this is why I make copious use of macros. I think this methodology saved me at least a thousands lines of code, maybe more.

3.7 RDHGASVC – SVC for #GETATB and #FREATB

This is a really small program. It doesn't do much. All it really does is to issue the SVCNTRY and SVCEXIT macros to handle the SVC protocol. Thereafter, it calls RDHGATB which does all the processing for the #GETATB and #FREATB requests. The reason it exists is that RDHGATB needs to be an authorized program in order to perform its function. And the SVC entry allows Problem State Key(8) program to utilize the service.

3.8 RDHGATB – #GETATB and #FREATP Processor

This is the processor that provides the service to allow for GETMAIN and FREEMAIN of above-the-bar or ATB storage in increments of less than one megabyte. This was a key piece of providing a true 64/20 development platform.

You don't always need a megabyte. Sometimes you just need 4K or 64-bytes of storage when you're writing an application. And to obtain one megabyte when what you need is a much smaller area is wasteful. So, that's why I wrote this code, to make more efficient use of ATB storage. This program creates a very effective means of managing ATB storage.

In this code, I keep track of both allocated storage and free areas in a series of DSECTs defined in an appendix to this document. Storage is task related, which means it goes away when the task terminates. But, you can obtain and release storage in any of the sixteen possible keys. Though, I don't support subpools, because I think it's a stupid idea. Keyed storage is important, but subpools are not.

This program was not very difficult to write, though I didn't think so when I set out to put fingers to keyboard. At first I thought of this project as a daunting task. It turned out not to be. It was only a five day project to write and fully debug to a GA level. It's 1,700 lines of code plus 1,000 lines of test programs.

I guess the hardest part of this was doing the FREEMAIN. So, if you're going to muck around with this code please be careful in that area. It can get pretty complicated. This is because the FREEMAIN section of the program must consolidate adjacent free areas of storage, whether they be prior free areas or after the newly freed area of storage, and sometimes both.

This is one cool piece of code and I invite you to look at the source. I'm proud of my work. It does what I need. And it's the only true 64-ANY program in this system. Super way, cool.

Maybe one day, IBM will step up to the plate and provide true support for 64-bit storage allocation and deallocation, but I'm not willing to hold my breath and you shouldn't be either. Until that day many years from now, probably long after I'm dead, you can use this User SVC to get you what you need.

Appendix A: – RDHTRACE Supporting Data Areas

These are the primary data areas that exist to support this system. You can find the real macros in the macro library. The version presented here is for documentation purposes only.

A.1 TRGDSECT – Trace Glue Code DSECT

This is the DSECT passed into RDHTRACE from all three of the various implementations, RDHTRACG, RDHTRSVC and RDHTRAPG. It contains all the information about what was going on at the moment the trace point trap was taken. This includes the Linkage Stack Entry, the Control Registers and a few other data items, like the TCB and ASCB addresses and the address of the TRPDSECT for the Trace Point.

```
***********************************************************************
*                                                                     *
** TRGDSECT - #TRACE Glue Code Data Area                              *
*                                                                     *
*   This DSECT area defines the machine status at the time of the     *
* execution of the #TRACE macro.  This includes the PSW, GP Registers *
* Access Registers, and Control Register in effect and is the primary *
* input to the tracing program RDHTRACE which produces the diagnostic *
* output.                                                             *
*                                                                     *
*                                                                     *
* Date     Developer Cng# Comments                                    *
* ======== ========= ==== =========================================== *
* 00/00/00 R.Harper  M000 Macro Developed                             *
*                                                                     *
***********************************************************************
*
TRGDSECT DSECT ,
TRGEYECT DS    CL8'TRGDSECT'         EyeCatcher
*
TRGFLAG1 DS    B                     Flag Byte #1
TRG1SRBM EQU   X'80'                 ..Running in SRB Mode
TRG1SVCI EQU   X'40'                 ..IMP=SVC
TRG1TRPI EQU   X'20'                 ..IMP=TRAP2
TRG1BSRI EQU   X'10'                 ..IMP=BASR
TRG1ARMD EQU   X'08'                 ..ARMODE PSW
TRG1AM64 EQU   X'04'                 ..AMODE=64
TRG1AM31 EQU   X'02'                 ..AMODE=31
TRG1AM24 EQU   X'01'                 ..AMODE=24
         DS    BL3                   (Reserved)
*
TRGASCB  DS    A                     Current ASCB Address
TRGTCB   DS    A                     Current TCB  Address
*
TRGTRPAD DS    A                     Addr of TRPDSECT This Trace Point
TRGPLIST DS    2F                    PLIST Area
TRGHLNTH EQU   *-TRGDSECT            Header Length
```

```
*
** Define Space for Copy of Linkage Stack Entry
*
TRGLSEN  DS    XL(LSENLNTH)          Space for Linkage Stack Entry
*
** Define Space for Control Registers
*
TRGCNTRL DS    0D                    Control Registers in Effect
TRGCR0   DS    D                     (Same)
TRGCR1   DS    D                     (Same) - PASN ASCE
TRGCR2   DS    D                     (Same) - DUCT Origin
TRGCR3   DS    D                     (Same)
TRGCR4   DS    D                     (Same)
TRGCR5   DS    D                     (Same)
TRGCR6   DS    D                     (Same)
TRGCR7   DS    D                     (Same) - SASN ASCE
TRGCR8   DS    D                     (Same)
TRGCR9   DS    D                     (Same)
TRGCR10  DS    D                     (Same)
TRGCR11  DS    D                     (Same)
TRGCR12  DS    D                     (Same)
TRGCR13  DS    D                     (Same) - HASN ASCE
TRGCR14  DS    D                     (Same)
TRGCR15  DS    D                     (Same) - Linkage Stack Pointer
*
         DS    0D                    Doubleword Aligned
TRGLNGTH EQU   *-TRGDSECT            Length of User Area
         MEND  ,
```

A.2 LSEDSECT – Linkage Stack Entry

The LSEDSECT is my DSECT that represents the format of the standard MVS Linkage Stack Entry generated via BAKR or PC. A copy of the Linkage Stack Entry is placed in the TRGDSECT above that is then passed into RDHTRACE to process a trace point.

```
.*********************************************************************
.*                                                                  *
.** LSEDSECT - Linckage tack Entry DSECT(s)                         *
.*                                                                  *
.*    This dsect is used to define the linkage stack entries.       *
.*                                                                  *
.*                                                                  *
.** Change History                                                  *
.*                                                                  *
.* Date     Developer Cng# Comments                                 *
.* ======== ========= ==== ========================================== *
.* 10/12/08 R.Harper  M000 Macro Developed                          *
.*                                                                  *
.*********************************************************************
.*
LSED     DSECT ,                        Linkage Stack Entry Descripter  LSEDTYPE DS
B                        Type of Stack Entry             LSEDTUSB EQU   X'80'
..Unstack Supression Bit        LSEDTHDR EQU   B'0001001'               ..Header Entry
LSEDTTRL EQU   B'0001010'               ..Trailer Entry                 LSEDTBKR EQU
B'0001100'               ..BAKR Entry                    LSEDTPCE EQU   B'0001101'
..PC Entry
```

```
LSEDSI   DS    X                        Section Identification
LSEDRFS  DS    H                        Remaining Free Space
LSEDNES  DS    H                        Next Entry Size
LSEDRES  DS    H                        (Reserved)                      LSEDLNTH EQU
*-LSED                   Length of Descripter
*
LSHD     DSECT ,                        Linkage Stack Header Entry
         DS    3F                       (Reeserved)                     LSHDPREV DS
A                        Previous Page                 LSHDLNTH EQU    *-LSHD
Length of Header Entry
*
LSEN     DSECT ,                        PC/BAKR Entryes                 LSENGRGS DS
0XL128                   GP Registers in Stack 64-bit
LSENGR0  DS    D                        ..General Register R0
LSENGR1  DS    D                        ..General Register R1
LSENGR2  DS    D                        ..General Register R2
LSENGR3  DS    D                        ..General Register R3
LSENGR4  DS    D                        ..General Register R4
LSENGR5  DS    D                        ..General Register R5
LSENGR6  DS    D                        ..General Register R6
LSENGR7  DS    D                        ..General Register R7
LSENGR8  DS    D                        ..General Register R8
LSENGR9  DS    D                        ..General Register R9           LSENGR10 DS
D                        ..General Register R10          LSENGR11 DS    D
..General Register R11          LSENGR12 DS    D                        ..General
Register R12          LSENGR13 DS    D                        ..General Register R13
LSENGR14 DS    D                        ..General Register R14          LSENGR15 DS
D                        ..General Register R15
*                                                                       LSENSTAT DS
0D                       Other Status Information
LSENPKM  DS    Y                        ..PSW KEY MASK Keys in CR3      LSENSASN DS
Y                        ..Secondary ASN in CR3
LSENEAX  DS    Y                        ..Extended Authorization Ndx    LSENPASN DS
Y                        ..Primary ASN ub CR4          LSENPSW1 DS    D
..Extended PSW Bits 0-63        LSENBRAD DS    0D                       ..Branch
Address if 64-bit      LSENSPID DS    F                        ....Called Space ID 64-
bit PC   LSENPCNB DS    F                        ....PC Number idf PC Entry PC
*                                       ....Branch Address if BAKR      LSEN31BT EQU
X'80'                    ....31-Bit Mode Indicator
LSENMODA DS    0D                       ..Modifiable Area               LSENMOD1 DS
A                        ..Modifiable Area - Word1       LSENMOD2 DS    A
..Modifiable Area - Word2
         DS    D                        ..(Reserved)                    LSENPSW2 DS
D                        ..Extended PSW Bits 64-127      LSENSAST DS    F
..Secondary ASTEIN              LSENPAST DS    F                        ..Primary
ASTEIN
         DS    5D                       ..(Reserved)
*                                                                       LSENARGS DS
0XL64                    AR Registers in Stack 32-bit
LSENAR0  DS    F                        ..Access Register R0
LSENAR1  DS    F                        ..Access Register R1
LSENAR2  DS    F                        ..Access Register R2
LSENAR3  DS    F                        ..Access Register R3
LSENAR4  DS    F                        ..Access Register R4
LSENAR5  DS    F                        ..Access Register R5
LSENAR6  DS    F                        ..Access Register R6
LSENAR7  DS    F                        ..Access Register R7
LSENAR8  DS    F                        ..Access Register R8
LSENAR9  DS    F                        ..Access Register R9            LSENAR10 DS
F                        ..Access Register R10          LSENAR11 DS    F
..Access Register R11          LSENAR12 DS    F                        ..Access
Register R12          LSENAR13 DS    F                        ..Access Register R13
```

```
LSENAR14 DS    F                        ..Access Register R14           LSENAR15 DS
F                    ..Access Register R15
*                                                                       LSENLNTH EQU
*-LSEN               Length of Entry
.*
.MEND    ANOP
         MEND  ,
```

A.3 TRPDSECT – Trace Point Trap Data Area

This DSECT defines the data area generated by the #TRACE macro. It's pretty much variable in nature depending upon what was specified on the macro. The data area, the data area title or the message area may or may not exist depending on what was specified on the #TRACE macro.

```
TRPDSECT DSECT ,
TRPNAME  DS    CL8                    Name of This Trap
TRPFLAG1 DS    B                      Flag Byte
TRPFTRAP EQU   X'80'                  ..Trace Point Trap
TRPFPGME EQU   X'40'                  ..Program Entry Trap
TRPFPGMX EQU   X'20'                  ..Program Exit  Trap
TRPFSUBE EQU   X'10'                  ..Subroutine Entry Trap
TRPFSUBX EQU   X'08'                  ..Subroutine Exit  Trap
TRPFLAG2 DS    B                      Flag Byte
TRPFSQEZ EQU   X'80'                  ..SQUEEZE=YES Specified
TRPFNOHD EQU   X'40'                  ..NOHEAD=YES Specified
TRPFREGS EQU   X'08'                  ..REGS= Was Specified
TRPFMSGS EQU   X'04'                  ..MSG=  Was Specified
TRPFDATA EQU   X'02'                  ..DATA= Was Specified
TRPLNGTH DS    HL2                    Length of This Trap
TRPTRPAD DS    AL4                    Addr of TRPDSECT+00 Open Segment
TRPOFFST DS    XL2                    Offset Into CSECT in Hex
TRPPREGS DS    BL2                    Registers to Display 0-15
TRPHLNTH EQU   *-TRPDSECT             Length of Header Area
*
TRPOPENA DS    0X                 -*- Define Open Segment - 1st Segment
TRPOFLG1 DS    B                      Flag Byte - Same as TRPFLAG1
TRPOBUFF EQU   X'04'                  ..Buffering is in Effect
TRPOFLG2 DS    B                      Protocol Type
TRPOPGME EQU   X'80'                  ..PGME
TRPOSVCE EQU   X'40'                  ..SVCE
TRPOESTE EQU   X'20'                  ..ESTAE
TRPOSRBE EQU   X'10'                  ..SRBE
TRPOPCE  EQU   X'08'                  ..PCE
TRPOBMAX DS    HL2                    Max Lines to Buffer
TRPDDNAM DS    CL8                    DDNAME Used
TRPCSECT DS    CL8                    CSECT Name of Program
TRPOLNTH EQU   *-TRPOPENA             Length of Open Area
*
         ORG   TRPOPENA               Overlay The Open Stuff
TRPMSG   DS    0H                 -*- Message Area Variable Segment
TRPMLNTH DS    YL2                    Length of MSG= Segment
TRPMDATA EQU   *                      Beginning of Message Data
*
TRPEVAR  DSECT ,                      Embedded Variable Description
TRPEVID  DS    X                      Donates Beginning of Variable
TRPEVVAR EQU   X'FE'                  ..Variable indicater
TRPEVFIL EQU   X'FD'                  ..Filler Byte - For Alignment
TRPETYP  DS    C                      Type of Constant to Insert
TRPETYPC EQU   C'C'                   ..C - Character Data
```

```
TRPETYPZ EQU   C'Z'                   ..Z - Zoned Data
TRPETYPF EQU   C'F'                   ..F - Fullword Data
TRPETYPH EQU   C'H'                   ..H - Halfword Data
TRPETYPP EQU   C'P'                   ..P - Packed Data
TRPETYPX EQU   C'X'                   ..X - Hexadecmil Data
TRPESCON DS    SL2                    SCON Address of Data to Insert
TRPESCDH DS    BL1                    DH Part of SCON
TRPEDLEN DS    YL1                    Length of Variable
TRPESLEN EQU   *-TRPEVAR              Length of a Variable Area (6)
*
TRPDDATA DSECT ,                  -*- Data Area Description
TRPDLNTH DS    H                      LENGTH of DATA= Area
TRPDCNT  DS    AL2                    Count of DATA= Entries (1,2,3)
*
TRPDSTRT DS    0X                     Start of One Segment
TRPDHDR  DS    0X                     Start of Area Heading
TRPDFLAG DS    B                      Flag Byte
TRPDFREG EQU   X'80'                  ..Length is in Register
TRPDFRDR EQU   X'70'                  ..Redirection in Effect
TRPDF64R EQU   X'40'                  ..Addr is !64-Bit Redirect
TRPDF31R EQU   X'20'                  ..Addr is ?31-Bit Redirect
TRPDF24R EQU   X'10'                  ..Addr is %24-Bit Redirect
TRPDFRAD EQU   X'08'                  ..Addr is is a Real Address
TRPDTLEN DS    AL1                    Length of Title
TRPDADRL DS    SL2                    BDDD Address of Area
TRPDADRH DS    BL1                    DH   Address of Area
TRPDATLN DS    FL2                    Length of Area to Print
TRPDTITL DS    0X                     Title For Area
TRPDSLEN EQU   *-TRPDSTRT             Length of DATA= Segment
         MEND ,
```

A.4 TRCDSECT – RDHTRACE Primary Internal Data Area

This is the primary data area used by RDHTRACE to process a trace point for a given TCB. Its address is anchored off an RME associated for the task.

```
         MACRO
&LBL     TRCDSECT ,
***********************************************************************
*                                                                     *
** TRCDSECT - Primary Data Repository for the RDHTRACE System         *
*                                                                     *
*    This data area contains all the data necessary to provide        *
*  support for the Label level Tracing Facility.  It is task          *
*  related, which is to say one of these data areas exist for         *
*  each MVS Task being traced within the address space.               *
*    It is Anchored off of RMEPARAM, the parameter passed into        *
*  RESMGR.  The RME data areas are a forward/backward chained         *
*  set of control blocks pointed to by STCBRMEF.                      *
*    This control block is created the first time a #TRACE Macro      *
*  is executed under any subtask within the address space, and        *
*  freed by the resource manager exit RDHRESEX at task termination.   *
*  RDHRESEX is located within RDHTRACE.                               *
*                                                                     *
* Date     Developer Cng# Comments                                    *
* ======== ========= ==== =========================================== *
* 00/00/00 R.Harper  M000 Macro Developed                             *
*                                                                     *
```

```
**********************************************************************
*
TRCDSECT DSECT ,                       Define TRC Persistant Data Area
TRCEYECT DS    CL8                     Eye Catcher C'TRCDSECT'
TRCCDATE DS    CL8                     Date Initiated
TRCCTIME DS    CL8                     Time Initiated
TRCTCBAC DS    CL8                     TCB Address in Character
TRCRC    DS    F                       Return Code
TRCRS    DS    F                       Reason Code
*
TRCPFLAG DS    B                       Processing Flag
TRCPDDSO EQU   X'80'                   ..TRCPRINT is SYSOUT
TRCPDDPS EQU   X'40'                   ..TRCPRINT is PS
TRCPDDPO EQU   X'20'                   ..TRCPRINT is PO
*
TRCBFLAG DS    B                       Buffering Flags
TRCBBFON EQU   X'80'                   ..Buffering is Indicated
TRCBBATL EQU   X'40'                   ..Buffer is Above-The-Line
TRCBBATB EQU   X'20'                   ..Buffer is Above-The-Bar
*
         DS    BL7                     (Reserved) - For More Flags
*
TRCTCBAD DS    A                       TCB Address That Own This Area
TRCPARMA DS    A                       R1 Value On Entry
TRCTRPAD DS    A                       Address of Current TRPDSECT
TRCENDAT DS    A                       Address of End of DATA
TRCDADDR DS    A                       Address of TRCDAREA Data Area
*
TRCBUFAD DS    A                       Address of Print Buffer
TRCBUFLN DS    F                       Length of Print Buffer
TRCBUFND DS    A                       Address of End of Buffer
TRCBUFCR DS    A                       Current Position in Buffer
*
TRCDDNAM DS    CL8                     DDNAME for This TCB
TRCDSNAM DS    CL44                    TRCPRINT DSNAME
*
TRCHEADT DS    CL8                     Type of Trap Heading
TRCNAME  DS    CL8                     Name For Trap Heading
*
         DS    0D
TRCPNTRY DS    XL(TRCDLNTH)            Previous Save Area Entry
         DS    0D
TRCCNTRY DS    XL(TRCDLNTH)            Current  Save Area Entry
*
TRCREG#  DS    PL2                     Current Register Number
TRCPTKEY DS    XL1                     Current Key
TRCWKCL9 DS    CL9                     General Work Area
TRCWKC16 DS    XL16                    (Same)
TRCWORKC DS    XL20                    (Same)
TRCWORKX DS    XL32                    (Same)
*
TRCSAVE4 DS    4A                      Register Save Area
TRCWORKD DS    D                       General Purpose Workarea
TRCWORKF DS    F                       (Same)
TRCLMOD  DS    CL8                     Load Module Name
TRCLMODC DS    CL8                     Load Module For Calling Procram
*
TRCPLINE DS    CL128                   Print Line
TRCMSGWK DS    CL255                   MSG= Work Area
*
** Estae Recovery Variables for TRCESTAE - RDHTRACE Recovery Program
*
TRCRETRY DS    F                       Address of Recovery RTN or Zero
```

```
TRCEFAIL DS    F                        Address of Fail Recovery Rtn
TRCEACNT DS    H                        Count of Abends
TRCEACMP DS    XL3                      Abend Completion Code
TRCEPSW  DS    2F                       PSW at Time of Error
TRCEREGS DS    16F                      Registers at Time of Abend 0-15
*
** Abend Recovery Variables for TRCABEND - Print Abend Diagnostics
*
TRCABSAV DS    9D                       Register Savearea
TRCPLIST DS    4F                       PLIST for CALL Macro
*
***********************************************************************
*
** Indention Levels
*
*    Each time a subroutine is called we increase the level of
*  indention by three bytes on the output to increase readabillity.
*  We allow a maximum of ten levels of indention within a program.
*  After that we stop indenting, as we've used up thirty bytes.
*   The low order byte of the TRCINDNT entry is used to maintain
*  the current number of bytes to indent.  The high order byte is
*  used to maintain the number of times we have entered a new
*  subroutine after we have overflowed.  In this way we can always
*  return to the proper indention level for the calling program upon
*  return from a called program.
*    This whole mechanism allows for 265 levels of nested subroutines
*  within a program and one hundred levels of nested programs per
*  task being traced. That should be enough.
*    Each time we enter a new program, we start again left justified
*  on the page and move to a new counter in TRCINDNT by incrementing
*  TRCINCUR.  When we exit a program we zero out that entry in
*  TRCINDNT and decrement TRCINCUR which should ultimately return to
*  zero upon return to MVS.
*
TRCINLVL DS    H                        Indention Level for This Call
*
TRCINCUR DS    H                        Index to Current Program in Tbl
TRCINDNT DS    100H                     Indention Levels for 100 Pgms
*
***********************************************************************
*
***********************************************************************
* Define Space for MF=L Constructs                                    *
***********************************************************************
*
*
         DS   0D
TRCDCB   DS   XL(MDLDCBL)               Reserve Space for DCB
TRCDCBID DS   CL4                       Trace DCB Ident C'TRAC'     @RDH
*
TRCRJFCL DS   F                         Reserve Space for RDJFCB MF=L
TRCEXLST DS   F                         Space for DCB EXLST
TRCJFCB  DS   0D,CL176                  Reserve Space for JFCB
*
         DS   0D
TRCOPL   DS   XL(MDLOPLL)               Reserve Space for OPEN Parm
         DS   0D
TRCCLL   DS   XL(MDLCLLL)               Reserve Space for CLOSE Parm
*
         DS   0D
TRCRESM  DS   XL(MDLRESML)              Reserve Space for RESMGR
TRCRESMP DS   D                         Parm Passed to Resource Manager
TRCRESMT DS   F                         Token From RESMGR
```

```
*
TRCESTAL ESTAEX ,CT,                    Exit Routine Address, Not Here  X
               XCTL=NO,                 XCTL Cancels ESTAE Exit         X
               PURGE=NONE,              Do Not Purge I/O on Abend       X
               ASYNCH=YES,              Allow ASYNC to Continue         X
               TERM=NO,                 Don't Enter if Task Cancelled   X
               MF=L                     This is List Form
*
         SQUEEZE MF=L                   Get Data Areas for SQUEEZE
TRCTMFL  TIME LINKAGE=SYSTEM,MF=L
*
         DS    0D
TRCLNGTH EQU   *-TRCDSECT               Length of Data Area
*
***********************************************************************
* TRCDAREA - Savearea Info DSECT - Pointed to by TRCDADDR             *
***********************************************************************
*
*   This area is filled in by TRCDSAVE on each call to RDHTRACE and
* any call to TRCADIAG during an abend.  It contains all the data
* related to programs in the savearea chain or linkage stack entries.
*
TRCDAREA DSECT ,
TRCDEYE  DS    CL8                      Eye Catcher
TRCDLEN  DS    A                        Length of Area for Freemain
TRCDCNT  DS    F                        Count of Enties in Table
TRCPLTH  EQU   *-TRCDAREA               Length of Prefix Area
*
TRCDNTRY DS    0D                       Initial CSECT Entry
TRCDLMOD DS    CL8                      LMOD Name
TRCDEPAD DS    A                        EP Address
         DS    A
*
TRCDCSCT DS    CL8                      CSECT Name
TRCDCSEP DS    A                        CSECT EP Address - R15 on Entry
TRCDCOFS DS    A                        CSECT Offset From LMOD
*
TRCDRTLM DS    CL8                      Return LMOD  Name
TRCDRTCS DS    CL8                      Return CSECT Name
TRCDRTAD DS    A                        Return Address
TRCDROFS DS    A                        Offset Into CSECT
*
TRCDLSEN DS    A                        Linkage Stack Entry Address
TRCDR13  DS    A                        R13 Value
*
TRCDPGID DS    CL32                     Program ID Data
TRCDLNTH EQU   *-TRCDNTRY               Length of Each Entry
*
TRCDSIZE EQU   12*1024                  Size for Getmain - 12K
TRCDMAXC EQU   TRCDSIZE/TRCDLNTH        Max Entries in Table
         MEND  ,
```

Appendix B – RDHGATB DSECT Data Areas

Below is the actual code from the program. These DSECT areas are what we use to manage above-the-bar or ATB storage for the #GETATB and #FREATB macros. Now, this code and these DSECTs may change as enhancements are made over time. So, for a current version, know that the DSECTs are in the program RDHGATB.

Unlike IBM who only keeps track of free storage, I keep track of both allocated and free storage using these control blocks. And I keep track of the address of the issuer of the GETMAIN, so that we can more easily diagnose problems with runaway GETMAINs.

```
*
*----------------------------------------------------------------------*
* Define Internal RDHGATB Data Areas                                   *
*----------------------------------------------------------------------*
*
         PRINT GEN
         RMEDSECT ,                    Define RMEDSECT Area
         LSEDSECT ,                    Define Linkage Stack Area
*
** Virtual Above-The-Bar Storage Management DSECT Areas
*
*
*  VABDSECT - Virtual ATB Primary Anchor, Exists Below the Line and
*             is pointed to by an RME
*  VAADSECT - Virtual ATB Primary Anchor Page, exists above the bar
*             and is pointed to by VABANCOR.
*  VAKDSECT - Virtual ATB Anchor for a Given Key, space is defined
*             within VAA at VAASTKEY, 16 copies are defined, one
*             for each possible key.
*
*  VASDSECT - Storage Entry, defines characteristics of allocated
*             or free storage pages.
*
*---------------------------------------------------------------------*
* VABDSECT - Define below the line Anchor Block pointed to by RME
*---------------------------------------------------------------------*
*
VABDSECT DSECT ,
VABEYECT DS    CL8                     EyeCatcher 'VABDSECT'
VABTCBAD DS    A                       TCB Address
VABLENTH DS    F                       Length of This Area
*
         DS    0D
VABRESMP DS    D                       Parm Passed to Resource Manager
VABRESMT DS    F                       Token From RESMGR Delete
VABRESM  DS    XL(MDLRESML)            Reserve Space for RESMGR
*
VABPAGCT DS    D                       Count of ATB Pages (1)
VABANCOR DS    AD                      1st ATB Page on Chain
VABANTOK DS    AD                      User Token for IARV64
*
         DS    0D                      End on Doubleword
VABLNGTH EQU   *-VABDSECT              Length of Area
*
*---------------------------------------------------------------------*
* VAADSECT - Primary Above the Bar Anchor Page
*---------------------------------------------------------------------*
*
```

```
VAADSECT DSECT ,                  -*- ATB Task Anchor Dsect
VAAEYECT DS    CL8                    EyeCatcher 'VAADSECT'
VAATCBAD DS    A                      TCB Address
VAAPAGCT DS    F                      Count of Pages in This Extent
VAANEXTA DS    AD                     Next VAADSECT Extent
*
VAASTABL DS    AD                     Beg of Storage Entry Table
VAASTCHN DS    AD                     Chain of Free Table Entries
         DS   3D                      (Reserved)
*
         DS    0D                     On Doubleword Boundry
VAASTKEY DS    16XL(VAKLNGTH)         One Entry Per Possible Key
*
         DS    0D
VAALNGTH EQU   *-VAADSECT             Length of Area
VAAMAXNT EQU   (1024*1024-VAALNGTH)/VASLNGTH   Max Entries Per Page
*
*--------------------------------------------------------------------*
* VAKDSECT - Define Pointers to Alloc/Free storage by Key
*--------------------------------------------------------------------*
*
*   There are 16 VAKDSECT entries defined one for each possible key.
* these areas are defined within VAADSECT at label VAASTKEY.
*
*   This table contains three chains of of VASDSECT entries that
* define areas of allocated or free storage within ATB pages we
* have acquired on behalf of the caller.
*
* VAKSTPAG - Will point at a VAS that have the address of a page
*            of ATB Storage we have acquired to fill a request. We
*            monitor this so we can free the storage at task term.
*
* VAKSTFRE - Points to a chain of VAS entries that represent areas
*            of storage within an ATB page that is free and available
*            for allocation.
*
* VAKSTALO - Points to a chain of VAS Entries that represent areas
*            of storage within an ATB page that is currently allocated.
*
*
VAKDSECT DSECT ,                  -*- ATB Per Key Storage Entry
VAKEYECT DS    CL3                    VAK Eye Catcher
VAKSTKEY DS    X                      Storage Key For This Entry
         DS    F
VAKSTPAG DS    AD                     Chain of Gotten Page   Entries
VAKSTFRE DS    AD                     Chain of Free  Storage Entries
VAKSTALO DS    AD                     Chain of Alloc Storage Entries
*
         DS    0D                     End on Doubleword
VAKLNGTH EQU   *-VAKDSECT             Length of Area - 32 -Bytes
*
*--------------------------------------------------------------------*
* VASDSECT - Define a Storage Entry for a Page, Allocated or Free Area
*--------------------------------------------------------------------*
*
VASDSECT DSECT ,                  -*- Storage Entry - Aloc/Free
VASEYECT DS    CL3                    VAS Eye Catcher
VASSTKEY DS    X                      Storage Key For This Entry
VAS1FLAG DS    B                      Flag Byte
VAS1NUSE EQU   X'80'                  This Storage Entry is In Use
VAS1SPAG EQU   X'40'                  Entry for Gotten Pages Chain
VAS1SFRE EQU   X'20'                  Entry for Free Space Chain
VAS1SALO EQU   X'10'                  Entry for Aloc Space Chain
```

```
         DS    BL3
VASCHAIN DS    AD                      Pointer to Next Entry Aloc/Free
VASSTPTR DS    AD                      Pointer to Storage Area
VASSTSIZ DS    A                       Size of Storage Area
VASPSWAD DS    A                       Return PSW From #GETATB
         DS    0D                      End on Doubleword
VASLNGTH EQU   *-VASDSECT              Length of Area - 32-Bytes
```

Appendix C – Sample Trace Output

I'm presenting the trace output for three of the test programs that I have in the /TST directory, you can run these programs too if you wish. The intention is that when somebody first looks at this Doc, that they can get a good feel for what this Development Platform can do. As I have said it's a pretty robust system.

You might want to look at the description of the #TRACE macro, so that you can understand how the MSG= and DATA= parameters work. But then really, there are so many examples of that in these programs that it might not really be necessary. If you just match up the #TRACE macros in the programs with the data produced in the trace output, you will understand what I have here.

The system produces trace points, and trace output in many different forms. You get a trace point for PGMNTRY and PGMEXIT, for SUBNTRY and SUBEXIT, and for all the #TRACE macros in the program.

These traces are activated only if the TRCPRINT DD is in the JCL, otherwise they are dormant, so as not to effect performance. If TRCPRINT is not in the JCL, each trace point will EXecute a NOPR, instead of the BASR, SVC, or TRAP2, and as such will be ineffective, produce no output and not effect performance.

So, please take a look at these programs, and see how I use this system. It's really cool, I assure you, and expect that you will be impressed. But you will have to decide for yourself. It's all there, an open book.

C.1 Test for AM64BS01- 64-Bit And ABEND Diagnostics

The first test is for RDH64AB1. This is a program that was designed to test out the propagation of ATB (Above-The-Bar) storage when a 64-bit PGMNTRY TYPE=MAIN program calls a 31-bit PGMNTRY TYPE=SUB program, that then calls a 64-bit PGMNTRY TYPE=SUB program. What I wanted to assure with this test was that the stacking ATB from the main program would be used by the second 64-bit TYPE=SUB, even if we had a 31-bit program in the middle. So, it's a pretty cool test that demonstrates the 64-bit support built into this system.

The second function of this test was to test out the ABEND diagnostics that will be generated when a 64-bit program ABENDs. It was important to assure that all the ATB storage, as well as the DSA storage, was properly printed for all the programs in the chain in the event of an ABEND. So, that's what this test was intended to show.

Code for AM64BS01

```
***********************************************************************
*                                                                     *
** RDH64AB1 - 64-Bit Adend Diagnostics Test Program                   *
*                                                                     *
*             *** This Program Intended to Abend ***                  *
*                                                                     *
*    This is a test program to test abend diagnostics for programs    *
*  running in AMODE(64).  In these cases we not only want to see the  *
*  standard diagnostics that we see with 31-bit programs which        *
*  includes DSA for all programs in the chain, we also want to see    *
*  the ATB storage for all programs in the chain.                     *
*    This program starts with an AMODE(64) TYPE=MAIN program that     *
*  then calls an AMODE(31) TYPE=SUB program sharing DSA, that then    *
*  calls an AMODE(64) TYPE=SUB program sharing DSA and sharing ATB    *
*  with the original TYPE=MAIN program.                               *
*    At that point we abend the program with a S0C3 and view the      *
*  abend diagnostics in TRCPRINT.                                     *
*                                                                     *
*                                                                     *
***********************************************************************
*
*---------------------------------------------------------------------*
* Define MVS System Data Areas                                        *
*---------------------------------------------------------------------*
*
         PRINT OFF
         IHAPSA  ,                     PSA  Deect
         CVT   DSECT=YES               CVT  Dsect
         IHASCVT ,                     SCVT Dsect
*
         IHAASCB ,                     ASCB Dsect
         IHAASXB ,                     ASXB Dsect
         IHAASSB ,                     ASSB Dsect
*
         IKJTCB  ,                     TCB  Dsect
         IHASTCB ,                     STCB Dsect
         IHARB   ,                     PRB  Dsect
         IHASDWA ,
         PRINT ON,NOGEN
*
***********************************************************************
* RDH64AB1 - Program Main Entry Point                                 *
***********************************************************************
*
         USING PSA,R0                  Establish Addressability
*
RDH64AB1 PGMNTRY TYPE=MAIN,            Establish Main Entry Point      *
               BASE=(R12,R11),         (Same)                          *
               ABAS=R10,               (Same)                          *
               AMODE=64,               (Same)                          *
               RMODE=ANY,              (Same)                          *
               IMP=BASR,PRINT=GEN
*
** Program MainLine
*
         GOSUB INIT                    Do Init Processing
         GOSUB MAIN                    Do Main Processing
         GOSUB TERM                    Do Term Processing
*
** Return to Caller
```

```
*
EXIT     DS    0H
         PGMEXIT RC=0                  Return to Caller
*
*=====================================================================*
** MAIN - Main Processing Routine                                     *
*=====================================================================*
*
MAIN     SUBNTRY ,
         CALL  RDH64SB1,               Call TYPE=SUB Program           *
               (CONPARM1,              Passing Parameters              *
               CONPARM2,               (Same)                          *
               DSAPARM1),              (Same)                          *
               LINKINST=BASSM,         Needed for 64-Bit Called Pgm    *
               MF=(E,DSMPLIST)
         ORG *-2
         BASPM R14,R15
*
*
MAIN999  DS    0H
         SUBEXIT ,
*
*=====================================================================*
** INIT - Do Initialization Processing                                *
*=====================================================================*
*
INIT     SUBNTRY ,
         MVC    ATBP1,=CL64'This is Data in ATBP1'
         MVC    ATBP2,=CL64'This is More Data in ATBP2'
*
         #TRACE DATA=('DSA Storage',0(R13),DSAPFXLN)
         #TRACE DATA=('ATB Storage',0(R10),256)
*
*
INIT999  DS    0H
         SUBEXIT ,
*
*=====================================================================*
** TERM - Do Termination Processing                                   *
*=====================================================================*
*
TERM     SUBNTRY ,
*
*
TERM999  DS    0H
         SUBEXIT ,
*
***********************************************************************
* Define Literials and Constants                                      *
***********************************************************************
*
         CONBEG ,
*
** Equates
*
CONPARM1 DC    CL20'This is a Parameter'
CONPARM2 DC    AD(TERM)
*
HEXFF    EQU   X'FF'                   Common Equates
HEX00    EQU   X'00'                   (Same)
*
** Constants
*
```

```
*
         CONEND ,
*
***********************************************************************
* Define Dynamic Storage Area                                         *
***********************************************************************
*
         DSABEG ,                      Define DSA Prefix
*
DSMPLIST DS      4AD
DSAPARM1 DS      D
*
         DSAEND ,                      Define DSA Suffex
*
         ATBBEG ,                      Define ATB Prefix
ATBP1    DS    CL64
ATBP2    DS    CL64
         ATBEND ,                      Define ATB Suffix
         DROP   ,                      Drop Everything
*
***********************************************************************
*                                                                     *
** RDH64SB1 - Sub-Program 1                                           *
*                                                                     *
***********************************************************************
*
         USING PSA,R0                  Establish Addressability
*
RDH64SB1 PGMNTRY TYPE=SUB,             Establish Main Entry Point      *
               BASE=(R12,R11),         (Same)                          *
               AMODE=31,IMP=BASR,      (Same)                          *
               RMODE=ANY,PRINT=GEN     (Same)
*
         LG    R10,DSAATBAD            Get ATB Storage Address
*
         #TRACE DATA=('Parameters Input',0(R1),32)
         #TRACE MSG='Propogating ATB Storage Over an AMODE(31) Pgm.',  *
               REGS=(R10)
*
         #TRACE DATA=('DSA Storage',0(R13),DSAPFXLN)
         SAM64
         #TRACE DATA=('ATB Storage',0(R10),256)
         SAM31
*
** Program MainLine
*
         GOSUB SUB1                    Do Main Processing
*
** Return to Caller
*
         PGMEXIT RC=0                  Return to Caller
*
*=====================================================================*
** SUB1 - Main Processing Routine                                     *
*=====================================================================*
*
SUB1     SUBNTRY ,
         CALL  RDH64SB2,               Call TYPE=SUB Program           *
               ((R10),                 Passing Parameters              *
               MYDATA),                (Same)                          *
               LINKINST=BASSM,         Needed for 64-Bit Called Pgm    *
               MF=(E,DS1PLIST)
         ORG *-2
         BASPM R14,R15
*
*
SUB1999  DS    0H
         SUBEXIT ,
*
***********************************************************************
* Define Literials and Constants                                      *
***********************************************************************
*
         CONBEG ,                        Define Constants Prefix
MYDATA   DC    CL8'My Data'
HEXTAB   DC    C'0123456789ABCDEF'       Hex Conversion Table
         CONEND ,                        Define Constants Suffix
*
         DSABEG ,                        Define DSA Prefix
DS1PLIST DS    4AD
         DSAEND ,                        Define DSA Suffix
*
***********************************************************************
*                                                                     *
** RDH64SB2 - Sub-Program 2                                           *
*                                                                     *
***********************************************************************
*
         USING PSA,R0                    Establish Addressability
*
RDH64SB2 PGMNTRY TYPE=SUB,               Establish Main Entry Point      *
               BASE=(R12,R11),           (Same)                          *
               ABAS=R10,                 (Same)                          *
               AMODE=64,IMP=BASR,        (Same)                          *
               RMODE=ANY,PRINT=GEN       (Same)
*
         MVC   ATBS1,=CL64'This is Data in ATBP1'
         MVC   ATBS2,=CL64'This is More Data in ATBP2'
*
         #TRACE DATA=('DSA Storage',0(R13),DSAPFXLN)
         #TRACE DATA=('ATB Storage',0(R10),256)
*
** Program MainLine
*
         GOSUB SUB2                      Do Main Processing
*
** Return to Caller
*
         PGMEXIT RC=0                    Return to Caller
*
*=====================================================================*
** SUB1 - Main Processing Routine                                     *
*=====================================================================*
*
SUB2     SUBNTRY ,
         #SUICIDE  MSG='Mercy Killing'
*
*
SUB2999  DS    0H
         SUBEXIT ,
*
***********************************************************************
* Define Literials and Constants                                      *
***********************************************************************
*
         CONBEG ,                        Define Constants Prefix
         CONEND ,                      Define Constants Suffix
*
         DSABEG ,                      Define DSA Prefix
         DSAEND ,                      Define DSA Suffex
*
         ATBBEG ,                      Define ATB Prefix
ATBS1    DS    CL64
ATBS2    DS    CL64
         ATBEND ,                      Define ATB Suffix
         END   ,
```

Trace Output Generated for AM64BS01

```
03/02/10        TRCPRINT - Label Level Tracing - TCB: 007E6968        15:32:16


 P) PGME: RDH64AB1 At: RDH64AB1.RDH64AB1+0000, Fr: Zos
          State(Problem), Key(8), Mode(Primary), Amode(64)
    R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
    R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
    R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
    R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
    R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
    R10 - 00000000/00000000_00000000   R11 - 00000000/00000000_007E6E88
    R12 - 00000000/00000000_832B1CEA   R13 - 00000000/00000000_00006F60
    R14 - 00000000/00000000_00FDC811   R15 - 00000000/00000000_19100E40

    S) SUBE: INIT At: RDH64AB1.RDH64AB1+04C2, Fr: RDH64AB1.RDH64AB1+0382
       T) Trap: TRP0107N At: RDH64AB1.RDH64AB1+04EA

          64-Bit Address    Ofst  DSA Storage
          ===============================================================================
          00000000_00007000 0000  C4E2C100 C6F1E2C1 00000000 00000000  *DSA F1SA        *
          00000000_00007010 0010  00000000 00000000 00000000 00000000  *                *
          00000000_00007020 0020  00000000 00000000 00000000 00000000  *                *
          00000000_00007030 0030  00000000 00000000 00000000 00000000  *                *
          00000000_00007040 0040  00000000 00000000 D9C4C8F6 F4C1C2F1  *        RDH64AB1*
          00000000_00007050 0050  00000000 00000000 E0000000 00000E30  *                *
          00000000_00007060 0060  00001000 000001D0 00010000 40400000  *                *
          00000000_00007070 0070  00000000 00000001 00000000 00000000  *                *
          00000000_00007080 0080  00000000 00000000 000000FE 00007000  *                *
          00000000_00007090 0090  00000008 00000000 00000000 00000000  *                *
          00000000_000070A0 00A0  00000000 00000000 00000000 00000000  *                *
          00000000_000070B0 00B0  00000000 00000000 00000000 00000000  *                *
          00000000_000070C0 00C0  00000008 00000000 00000000 00000001  *                *
          00000000_000070D0 00D0  00000000 00000100 00000000 000FFF00  *                *
          00000000_000070E0 00E0  000000FE 00007000 00000000 00000000  *                *
          00000000_000070F0 00F0  00000000 00000000 00000000 00000000  *                *
          00000000_00007100 0100  00000000 00000000 00000000 191011C2  *               B*
          00000000_00007110 0110  00000000 19100E40 00000000 FD000008  *                *
          00000000_00007120 0120  00000000 00006FF8 00000000 00000000  *       8        *
          00000000_00007130 0130  00000000 00000000 00000000 00000000  *                *
          00000000_00007140 0140  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
          00000000_00007150 0150  00000000 00000000 00000000 00000000  *                *
          00000000_00007160 0160  00000000 00000000 00000000 00000000  *                *
          00000000_00007170 0170  00000000 00000000 00000000 00000000  *                *

       T) Trap: TRP0109N At: RDH64AB1.RDH64AB1+050C

          64-Bit Address    Ofst  ATB Storage
          ===============================================================================
          00000008_00000000 0000  C1E3C2C4 E2C5C3E3 D9C4C8F6 F4C1C2F1  *ATBDSECTRDH64AB1*
          00000008_00000010 0010  00000000 00000100 00000000 00000000  *                *
          00000008_00000020 0020  00000000 00000000 00000000 00000000  *                *
          00000008_00000030 0030  00000000 00000000 00000000 00000000  *                *
          00000008_00000040 0040  C481A381 40899540 C1E3C240 E2A39699  *Data in ATB Stor*
          00000008_00000050 0050  81878540 C1E3C2D7 F14B4040 40404040  *age ATBP1       *
          00000008_00000060 0060  40404040 40404040 40404040 40404040  *                *
          00000008_00000070 0070  40404040 40404040 40404040 40404040  *                *
          00000008_00000080 0080  C481A381 40899540 C1E3C240 E2A39699  *Data in ATB Stor*
          00000008_00000090 0090  81878540 C1E3C2D7 F24B4040 40404040  *age ATBP2       *
          00000008_000000A0 00A0  40404040 40404040 40404040 40404040  *                *
          00000008_000000B0 00B0  40404040 40404040 40404040 40404040  *                *
          00000008_000000C0 00C0  00000000 00000000 00000000 00000000  *                *
          00000008_000000D0 00D0  00000000 00000000 00000000 00000000  *                *
          00000008_000000E0 00E0  00000000 00000000 00000000 00000000  *                *
          00000008_000000F0 00F0  00000008 000000F0 00000000 C1E3C2C5  *       0    ATBE*

    S) SUBX: INIT At: RDH64AB1.RDH64AB1+052E, To: RDH64AB1.RDH64AB1+0382
```

```
S) SUBE: MAIN At: RDH64AB1.RDH64AB1+0454, Fr: RDH64AB1.RDH64AB1+0386

P) PGME: RDH64SB1 At: RDH64AB1.RDH64SB1+0000, Fr: RDH64AB1.RDH64AB1+0498
         State(Supervisor), Key(0), Mode(Primary), Amode(64)
   R00 - 00000000/00000000_000071B8   R01 - 00000000/00000000_00007198
   R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
   R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
   R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
   R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
   R10 - 00000000/00000008_00000000   R11 - 00000000/00000000_19101E40
   R12 - 00000000/00000000_19100E40   R13 - 00000000/00000000_00007000
   R14 - 00000000/00000000_191012D9   R15 - 00000000/00000000_19101661

   T) Trap: TRP0146N At: RDH64AB1.RDH64SB1+025C

      64-Bit Address    Ofst  DSA Storage
      ===============================================================================
      00000000_000071D0 0000  C4E2C100 C6F1E2C1 00000000 00000000  *DSA F1SA        *
      00000000_000071E0 0010  00000000 00000000 00000000 00000000  *                *
      00000000_000071F0 0020  00000000 00000000 00000000 00000000  *                *
      00000000_00007200 0030  00000000 00000000 00000000 00000000  *                *
      00000000_00007210 0040  00000000 00000000 D9C4C8F6 F4E2C2F1  *        RDH64SB1*
      00000000_00007220 0050  00000000 00000000 00000000 00000C70  *                *
      00000000_00007230 0060  00000000 000001C0 00000000 00000000  *                *
      00000000_00007240 0070  00000000 00000000 00000000 00000000  *                *
      00000000_00007250 0080  00000000 00000000 00000000 00000000  *                *
      00000000_00007260 0090  00000000 00000000 00000000 00000000  *                *
      00000000_00007270 00A0  00000000 00000000 00000000 00000000  *                *
      00000000_00007280 00B0  00000000 00000000 00000000 00000000  *                *
      00000000_00007290 00C0  00000008 00000000 00000000 00000000  *                *
      00000000_000072A0 00D0  00000000 00000100 00000000 000FFF00  *                *
      00000000_000072B0 00E0  00000000 00000000 00000000 00000000  *                *
      00000000_000072C0 00F0  00000000 00000000 00000000 00000000  *                *
      00000000_000072D0 0100  00000000 00000000 00000000 191012D9  *               R*
      00000000_000072E0 0110  00000000 19101661 00000000 000071B8  *       /        *
      00000000_000072F0 0120  00000000 00007198 00000000 00000000  *       q        *
      00000000_00007300 0130  00000000 00000000 00000000 00000000  *                *
      00000000_00007310 0140  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
      00000000_00007320 0150  00000000 00000000 00000000 00000000  *                *
      00000000_00007330 0160  00000000 00000000 00000000 00000000  *                *
      00000000_00007340 0170  00000000 00000000 00000000 00000000  *                *

   S) SUBE: SUB1 At: RDH64AB1.RDH64SB1+02EE, Fr: RDH64AB1.UNKNOWN+18DC

   P) PGME: RDH64SB2 At: RDH64AB1.RDH64SB2+0000, Fr: RDH64AB1.        +0000
            State(Supervisor), Key(0), Mode(Primary), Amode(31)
      R00 - 00000000/00000000_000071B8   R01 - 00000000/00000000_00007358
      R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
      R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
      R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
      R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
      R10 - 00000000/00000008_00000000   R11 - 00000000/00000000_19102660
      R12 - 00000000/00000000_19101660   R13 - 00000000/00000000_000071D0
      R14 - 00000000/00000000_99101988   R15 - 00000000/00000000_19101B08

      T) Trap: TRP0219N At: RDH64AB1.RDH64SB2+0394

         64-Bit Address    Ofst  DSA Storage
         ===============================================================================
         00000000_00007390 0000  C4E2C100 C6F1E2C1 00000000 00000000  *DSA F1SA        *
         00000000_000073A0 0010  00000000 00000000 00000000 00000000  *                *
         00000000_000073B0 0020  00000000 00000000 00000000 00000000  *                *
         00000000_000073C0 0030  00000000 00000000 00000000 00000000  *                *
         00000000_000073D0 0040  00000000 00000000 D9C4C8F6 F4E2C2F2  *        RDH64SB2*
         00000000_000073E0 0050  00000000 00000000 20000000 00000AD0  *                *
         00000000_000073F0 0060  00000000 000001A0 00000000 00000000  *                *
         00000000_00007400 0070  00000000 00000000 00000000 00000000  *                *
         00000000_00007410 0080  00000000 00000000 00000000 00000000  *                *
         00000000_00007420 0090  00000000 00000000 00000000 00000000  *                *
         00000000_00007430 00A0  00000000 00000000 00000000 00000000  *                *
         00000000_00007440 00B0  00000000 00000000 00000000 00000000  *                *
         00000000_00007450 00C0  00000008 00000100 00000000 00000000  *                *
         00000000_00007460 00D0  00000000 00000100 00000000 000FFE00  *                *
         00000000_00007470 00E0  000000FE 00007390 00000000 00000000  *                *
         00000000_00007480 00F0  00000000 00000000 00000000 00000000  *                *
         00000000_00007490 0100  00000000 00000000 00000000 99101988  *             r  h*
         00000000_000074A0 0110  00000000 19101B08 00000000 000071B8  *                *
         00000000_000074B0 0120  00000000 00007358 00000000 00000000  *                *
         00000000_000074C0 0130  00000000 00000000 00000000 00000000  *                *
         00000000_000074D0 0140  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
         00000000_000074E0 0150  00000000 00000000 00000000 00000000  *                *
         00000000_000074F0 0160  00000000 00000000 00000000 00000000  *                *
         00000000_00007500 0170  00000000 00000000 00000000 00000000  *                *

      S) SUBE: SUB2 At: RDH64AB1.RDH64SB2+0468, Fr: RDH64AB1.RDH64SB2+03B4
         T) Trap: TRP0236N At: RDH64AB1.RDH64SB2+0490

            64-Bit Address    Ofst  DSA Storage
            =============================================================================
            00000000_00007390 0000  C4E2C100 C6F1E2C1 00000000 00000000  *DSA F1SA        *
            00000000_000073A0 0010  00000000 00000000 00000000 00000000  *                *
            00000000_000073B0 0020  00000000 00000000 00000000 00000000  *                *
            00000000_000073C0 0030  00000000 00000000 00000000 00000000  *                *
            00000000_000073D0 0040  00000000 00000000 D9C4C8F6 F4E2C2F2  *        RDH64SB2*
            00000000_000073E0 0050  00000000 00000000 20000000 00000AD0  *                *
            00000000_000073F0 0060  00000000 000001A0 00000000 00000000  *                *
            00000000_00007400 0070  00000000 00000000 00000000 00000000  *                *
            00000000_00007410 0080  00000000 00000000 00000000 00000000  *                *
            00000000_00007420 0090  00000000 00000000 00000000 00000000  *                *
            00000000_00007430 00A0  00000000 00000000 00000000 00000000  *                *
            00000000_00007440 00B0  00000000 00000000 00000000 00000000  *                *
            00000000_00007450 00C0  00000008 00000100 00000000 00000000  *                *
            00000000_00007460 00D0  00000000 00000100 00000000 000FFE00  *                *
            00000000_00007470 00E0  000000FE 00007390 00000000 00000000  *                *
            00000000_00007480 00F0  00000000 00000000 00000000 00000000  *                *
            00000000_00007490 0100  00000000 00000000 00000000 19101EBC  *                *
            00000000_000074A0 0110  00000000 19101B08 00000000 000071B8  *                *
            00000000_000074B0 0120  00000000 00007358 00000000 00000000  *                *
            00000000_000074C0 0130  00000000 00000000 00000000 00000000  *                *
            00000000_000074D0 0140  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
            00000000_000074E0 0150  00000000 00000000 00000000 00000000  *                *
            00000000_000074F0 0160  00000000 00000000 00000000 00000000  *                *
            00000000_00007500 0170  00000000 00000000 00000000 00000000  *                *

         T) Trap: TRP0238N At: RDH64AB1.RDH64SB2+04B2

            64-Bit Address    Ofst  ATB Storage
            =============================================================================
            00000008_00000100 0000  C1E3C2C4 E2C5C3E3 D9C4C8F6 F4E2C2F2  *ATBDSECTRDH64SB2*
            00000008_00000110 0010  00000000 00000100 00000000 00000000  *                *
            00000008_00000120 0020  00000000 00000000 00000000 00000000  *                *
            00000008_00000130 0030  00000000 00000000 00000000 00000000  *                *
            00000008_00000140 0040  C481A381 40899540 C1E3C240 E2A39699  *Data in ATB Stor*
            00000008_00000150 0050  81878540 C1E3C2E2 F14B4040 40404040  *age ATBS1       *
            00000008_00000160 0060  40404040 40404040 40404040 40404040  *                *
            00000008_00000170 0070  40404040 40404040 40404040 40404040  *                *
            00000008_00000180 0080  C481A381 40899540 C1E3C240 E2A39699  *Data in ATB Stor*
            00000008_00000190 0090  81878540 C1E3C2E2 F24B4040 40404040  *age ATBS2       *
            00000008_000001A0 00A0  40404040 40404040 40404040 40404040  *                *
            00000008_000001B0 00B0  40404040 40404040 40404040 40404040  *                *
            00000008_000001C0 00C0  00000000 00000000 00000000 00000000  *                *
            00000008_000001D0 00D0  00000000 00000000 00000000 00000000  *                *
            00000008_000001E0 00E0  00000000 00000000 00000000 00000000  *                *
            00000008_000001F0 00F0  00000008 000001F0 00000000 C1E3C2C5  *       0    ATBE*


*****************************************************************************
*                           -> Program Abended <-                           *
*****************************************************************************
*
```

```
** Diagnostics for RTM2WA at: 7F591E28
*

  ABEND CODE: S0C3    REASON CODE:  00000003   ILC: 4
         PSW: 07042001 80000000   00000000 19101FDA
  DATA @ PSW:          19101FD4 - 00044400 C4CEE3E0 D1800004 EBE1D108

  State(Supervisor), Key(0), Mode(Primary), Amode(64)

  Active Load Module: RDH64AB1, EP: 19100E40, Offset: 00119A
       Current Csect: RDH64SB2, EP: 19101B08, Offset: 0004D2

    AR/GR Registers at Entry to Abend
    ---------------------------------
    0: 00000000/00000000_000071B8    1: 00000000/00000000_00007358
    2: 00000000/00000000_00000040    3: 00000000/00000000_007D99D4
    4: 00000000/00000000_007D99B0    5: 00000000/00000000_007E6E88
    6: 00000000/00000000_007C8FE0    7: 00000000/00000000_FD000000
    8: 00000000/00000000_007FCE28    9: 00000000/00000000_007E6B30
    A: 00000000/00000008_00000100    B: 00000000/00000000_19102B08
    C: 00000000/00000000_19101B08    D: 00000000/00000000_00007390
    E: 00000000/00000000_19101EBC    F: 00000000/00000000_19101B08

    Control Registers at Entry to Abend
    -----------------------------------
    CR0-3:   DF08EE60  3DB94007    249DCAC0  00C00031
    CR4-7:   00000031  3DE79C40    FE000000  FE000000
    CR8-11:  00000000  00000000    00000000  00000000
    CR12-15: 35B3E1F7  3DB94007    DF8BDECF  7F569388


    Save Area Chain in Reverse Order
    --------------------------------

      LS BAKR Entry At: 7F569268, For: RDH64SB2
      -----------------------------------------
      PSW:  07041000 80000000   00000000 19101988
      MSTK: 07041000 80000000   F1SA: 00007390
      PASN: 0031   SASN: 0031   PKM:  00C0 EAX: 0000

         0: 00000000/00000000_000071B8    1: 00000000/00000000_00007358
         2: 00000000/00000000_00000040    3: 00000000/00000000_007D99D4
         4: 00000000/00000000_007D99B0    5: 00000000/00000000_007E6E88
         6: 00000000/00000000_007C8FE0    7: 00000000/00000000_FD000000
         8: 00000000/00000000_007FCE28    9: 00000000/00000000_007E6B30
         A: 00000000/00000008_00000000    B: 00000000/00000000_19102660
         C: 00000000/00000000_19101660    D: 00000000/00000000_000071D0
         E: 00000000/00000000_99101988    F: 00000000/00000000_19101B08

         Dymamic Storage At: 00007390, For: RDH64SB2
         -----------------------------------------
         DSA Prefix
         00007390  C4E2C100 C6F1E2C1 00000000 00000000  *DSA F1SA        *
         000073A0  00000000 00000000 00000000 00000000  *                *
         000073B0  00000000 00000000 00000000 00000000  *                *
         000073C0  00000000 00000000 00000000 00000000  *                *
         000073D0  00000000 00000000 D9C4C8F6 F4E2C2F2  *        RDH64SB2*
         000073E0  00000000 00000000 20000000 00000AD0  *                *
         000073F0  00000000 000001A0 00000000 00000000  *                *
         00007400  00000000 00000000 00000000 00000000  *                *
         00007410  00000000 00000000 00000000 00000000  *                *
         00007420  00000000 00000000 00000000 00000000  *                *
         00007430  00000000 00000000 00000000 00000000  *                *
         00007440  00000000 00000000 00000000 00000000  *                *
         00007450  00000008 00000100 00000000 00000000  *                *
         00007460  00000000 00000100 00000000 000FFE00  *                *
         00007470  000000FE 00007390 00000000 00000000  *                *
         00007480  00000000 00000000 00000000 00000000  *                *
         00007490  00000000 00000000 00000000 19101EBC  *                *
         000074A0  00000000 19101B08 00000000 000071B8  *                *
         000074B0  00000000 00007358 00000000 00000000  *                *
```

```
   000074C0  00000000 00000000 00000000 00000000  *                *
   000074D0  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
   000074E0  00000000 00000000 00000000 00000000  *                *
   000074F0  00000000 00000000 00000000 00000000  *                *
   00007500  00000000 00000000 00000000 00000000  *                *

   Program DSA Storage
   00007510  00000000 19101EBC 00000000 00000000  *                *

   DSA Suffix
   00007520  00000000 00000000 00007528 C4E2C1C5  *            DSAE*


   Above-The-Bar Storage At: 00000008_00000100, For: RDH64SB2
   -----------------------------------------------------------
   ATB Prefix
   00000008_00000100  C1E3C2C4 E2C5C3E3 D9C4C8F6 F4E2C2F2  *ATBDSECTRDH64SB2*
   00000008_00000110  00000000 00000100 00000000 00000000  *                *
   00000008_00000120  00000000 00000000 00000000 00000000  *                *
   00000008_00000130  00000000 00000000 00000000 00000000  *                *

   Program ATB Storage
   00000008_00000140  C481A381 40899540 C1E3C240 E2A39699  *Data im ATB Stor*
   00000008_00000150  81878540 C1E3C2E2 F14B4040 40404040  *age ATBS1       *
   00000008_00000160  40404040 40404040 40404040 40404040  *                *
   00000008_00000170  40404040 40404040 40404040 40404040  *                *
   00000008_00000180  C481A381 40899540 C1E3C240 E2A39699  *Data im ATB Stor*
   00000008_00000190  81878540 C1E3C2E2 F24B4040 40404040  *age ATBS2       *
   00000008_000001A0  40404040 40404040 40404040 40404040  *                *
   00000008_000001B0  40404040 40404040 40404040 40404040  *                *
   00000008_000001C0  00000000 00000000 00000000 00000000  *                *
   00000008_000001D0  00000000 00000000 00000000 00000000  *                *
   00000008_000001E0  00000000 00000000 00000000 00000000  *                *

   ATB Suffix
   00000008_000001F0  00000008 000001F0 00000000 C1E3C2C5  *       0    ATBE*


LS BAKR Entry At: 7F569140, For: UNKNOWN
-----------------------------------------
PSW:  07041001 80000000   00000000 191012D8
MSTK: 07041001 80000000   F1SA: 000071D0
PASN: 0031   SASN: 0031   PKM:  00C0 EAX: 0000

   0: 00000000/00000000_000071B8    1: 00000000/00000000_00007198
   2: 00000000/00000000_00000040    3: 00000000/00000000_007D99D4
   4: 00000000/00000000_007D99B0    5: 00000000/00000000_007E6E88
   6: 00000000/00000000_007C8FE0    7: 00000000/00000000_FD000000
   8: 00000000/00000000_007FCE28    9: 00000000/00000000_007E6B30
   A: 00000000/00000008_00000000    B: 00000000/00000000_19101E40
   C: 00000000/00000000_19100E40    D: 00000000/00000000_00007000
   E: 00000000/00000000_191012D9    F: 00000000/00000000_19101661

   Dymamic Storage At: 000071D0, For: UNKNOWN
   ------------------------------------------
   DSA Prefix
   000071D0  C4E2C100 C6F1E2C1 00007390 00000000  *DSA F1SA        *
   000071E0  00000000 00000000 00000000 00000000  *                *
   000071F0  00000000 00000000 00000000 00000000  *                *
   00007200  00000000 00000000 00000000 00000000  *                *
   00007210  00000000 00000000 D9C4C8F6 F4E2C2F1  *        RDH64SB1*
   00007220  00000000 00000000 00000000 00000C70  *                *
   00007230  00000000 000001C0 00000000 00000000  *                *
   00007240  00000000 00000000 00000000 00000000  *                *
   00007250  00000000 00000000 00000000 00000000  *                *
   00007260  00000000 00000000 00000000 00000000  *                *
   00007270  00000000 00000000 00000000 00000000  *                *
   00007280  00000000 00000000 00000000 00000000  *                *
   00007290  00000008 00000000 00000000 00000000  *                *
   000072A0  00000000 00000100 00000000 000FFF00  *                *
   000072B0  00000000 00000000 00000000 00000000  *                *
   000072C0  00000000 00000000 00000000 00000000  *                *
   000072D0  00000000 00000000 00000000 991018DC  *            r   *
   000072E0  00000000 19101661 00000000 000071B8  *                *
   000072F0  00000000 00007198 00000000 00000000  *       q        *
   00007300  00000000 00000000 00000000 00000000  *                *
   00007310  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
   00007320  00000000 00000000 00000000 00000000  *                *
   00007330  00000000 00000000 00000000 00000000  *                *
   00007340  00000000 00000000 00000000 00000000  *                *

   Program DSA Storage
   00007350  00000000 991018DC 00000000 191019AC  *    r           *
   00007360  00000000 00000000 00000000 00000000  *                *
   00007370  00000000 00000000 00000000 00000000  *                *

   DSA Suffix
   00007380  00000000 00000000 00007388 C4E2C1C5  *            hDSAE*


LS BAKR Entry At: 7F569018, For: RDH64AB1
-----------------------------------------
PSW:  07850001 80000000   00000000 00FDC810
MSTK: 07850001 80000000   F1SA: 00007000
PASN: 0031   SASN: 0031   PKM:  00C0 EAX: 0000

   0: 00000000/00000000_FD000008    1: 00000000/00000000_00006FF8
   2: 00000000/00000000_00000040    3: 00000000/00000000_007D99D4
   4: 00000000/00000000_007D99B0    5: 00000000/00000000_007E6E88
   6: 00000000/00000000_007C8FE0    7: 00000000/00000000_FD000000
   8: 00000000/00000000_007FCE28    9: 00000000/00000000_007E6B30
   A: 00000000/00000000_00000000    B: 00000000/00000000_007E6E88
   C: 00000000/00000000_832B1CEA    D: 00000000/00000000_00006F60
   E: 00000000/00000000_00FDC811    F: 00000000/00000000_19100E40

   Dymamic Storage At: 00007000, For: RDH64AB1
   ------------------------------------------
   DSA Prefix
   00007000  C4E2C100 C6F1E2C1 000071D0 00000000  *DSA F1SA        *
   00007010  00000000 00000000 00000000 00000000  *                *
   00007020  00000000 00000000 00000000 00000000  *                *
   00007030  00000000 00000000 00000000 00000000  *                *
   00007040  00000000 00000000 D9C4C8F6 F4C1C2F1  *        RDH64AB1*
   00007050  00000000 00000000 E0000000 00000E30  *                *
   00007060  00001000 000001D0 00010000 40400000  *                *
   00007070  00000000 00000001 00000000 00000000  *                *
   00007080  00000000 00000000 000000FE 00007000  *                *
   00007090  00000008 00000000 00000000 00000000  *                *
   000070A0  00000000 00000000 00000000 00000000  *                *
   000070B0  00000000 00000000 00000000 00000000  *                *
   000070C0  00000008 00000000 00000000 00000001  *                *
   000070D0  00000000 00000100 00000000 000FFF00  *                *
   000070E0  000000FE 00007000 00000000 00000000  *                *
   000070F0  00000000 00000000 00000000 00000000  *                *
   00007100  00000000 00000000 00000000 191011C6  *               F*
   00007110  00000000 19100E40 00000000 FD000008  *                *
   00007120  00000000 00006FF8 00000000 00000000  *       8        *
   00007130  00000000 00000000 00000000 00000000  *                *
   00007140  99102300 0CEF0CEF 0CEF0CEF 00000000  *r               *
   00007150  00000000 00000000 00000000 00000000  *                *
   00007160  00000000 00000000 00000000 00000000  *                *
   00007170  00000000 00000000 00000000 00000000  *                *

   Program DSA Storage
   00007180  00000000 191011C6 00000000 191011C2  *       F       B*
   00007190  00000000 00000000 00000000 191013D2  *               K*
   000071A0  00000000 191013E8 00000000 000071B8  *       Y        *
   000071B0  00000000 00000000 00000000 00000000  *                *

   DSA Suffix
   000071C0  00000000 00000000 000071C8 C4E2C1C5  *            HDSAE*
```

```
      Above-The-Bar Storage At: 00000008_00000000, For: RDH64AB1
      ----------------------------------------------------------
      ATB Prefix
      00000008_00000000  C1E3C2C4 E2C5C3E3 D9C4C8F6 F4C1C2F1  *ATBDSECTRDH64AB1*
      00000008_00000010  00000000 00000100 00000000 00000000  *                *
      00000008_00000020  00000000 00000000 00000000 00000000  *                *
      00000008_00000030  00000000 00000000 00000000 00000000  *                *

      Program ATB Storage
      00000008_00000040  C481A381 40899540 C1E3C240 E2A39699  *Data im ATB Stor*
      00000008_00000050  81878540 C1E3C2D7 F14B4040 40404040  *age ATBP1       *
      00000008_00000060  40404040 40404040 40404040 40404040  *                *
      00000008_00000070  40404040 40404040 40404040 40404040  *                *
      00000008_00000080  C481A381 40899540 C1E3C240 E2A39699  *Data im ATB Stor*
      00000008_00000090  81878540 C1E3C2D7 F24B4040 40404040  *age ATBP2       *
      00000008_000000A0  40404040 40404040 40404040 40404040  *                *
      00000008_000000B0  40404040 40404040 40404040 40404040  *                *
      00000008_000000C0  00000000 00000000 00000000 00000000  *                *
      00000008_000000D0  00000000 00000000 00000000 00000000  *                *
      00000008_000000E0  00000000 00000000 00000000 00000000  *                *

      ATB Suffix
      00000008_000000F0  00000008 000000F0 00000000 C1E3C2C5  *       0    ATBE*


   Zos Standard Savearea At: 00006F60, For: Zos
   ------------------------------------------------
   00006F60  00000000 00000000 00007000 00000000  *                *
   00006F70  00000000 00000000 00000000 00000000  *                *
   00006F80  00000000 00000000 00000000 00000000  *                *
   00006F90  00000000 00000000 00000000 00000000  *                *
   00006FA0  00000000 00000000                    *                *
```

C.2 Test for ARMDBS01- ARMODE Programs

This is a test for programs running in ARMODE to assure that the code in RDHTRACE will properly print data from another address space or dataspace when the #TRACE macro specifies an address where the access register for the base of that data contains an ALET.

In this test we're simply using dataspaces, though it will certainly work for other address spaces to. I create three dataspaces within the PASN-AL and two for the DU-AL. Then I use #TRACE macros to print out the PASN-AL and DU-AL before and after dataspace creation, in addition to printing the data from those dataspaces.

Another thing this program tests is all paths for all the various data types that are supported by this system when accessing data in another address space or dataspace. These include character, packed, zoned, halfword, fullword and hexadecimal types of data.

One of the things I learned in playing with the ASTE, the DUCT, the PASN-AL and the DU-AL, was how to create an ALET from the ALE in these ASTE tables. I no longer need the ALESREV or the DSPSERVE macros. I can do it myself, to access any data in any address space or any dataspace in the system.

If you look at the ASTE for the Before entry in the trace output you might notice that there are some X'37' or 55 ALEs in that table. Each of these represents a dataspace or an address space for which an ALET can be created. I know because I have a program that will display data in every one of these 55 spaces. That's how I learned to create my own ALETs. Most of these will be anchored off the JESAUX or Master Scheduler's address space. Most of these seem to be used internally by IBM, for things like RACF and WTO messages.

This program was written prior to when I added support for '@' indirection to the #TRACE macro, which will allow you to specify a real storage address for the data to be displayed. Back then I was using the MFREAL macro to move real storage data to a virtual address prior to executing #TRACE. I don't need to do that now, but there is no sense in rewriting a program that works just fine. If it ain't broke, don't fix it, eh?

Another short point, is that it was also written prior to when I developed #WTOR. So, I actually use the MVS version of WTOR, though it's deactivated in this version. I wanted to be able to stop the code, so that I could use MERLIN or take a dump and use IPCS to poke around. That's not needed at this point, but the code is still there, just commented out.

Anyway, this is still a very interesting program. I hope you enjoy the view.

Code for ARMDBS01

```
***********************************************************************
*                                                                     *
** RDHARB01 - Test Program for for ARMODE Programs with IMP=BASR      *
*                                                                     *
*    This program will test the situation where the data we wish      *
*  to display is in a different address space or dataspace.  This     *
*  The program will be in ARMODE when executing the #TRACE and        *
*  RDHTRACE will trigger off that bit in the PSW to determine if      *
*  it needs to go into ARMODE in order to access the data to be       *
*  written to the traces.                                             *
*                                                                     *
***********************************************************************
*
*---------------------------------------------------------------------*
* Define MVS System Data Areas                                        *
*---------------------------------------------------------------------*
*
         PRINT OFF
         IHAPSA  ,                    PSA  Deect
         CVT   DSECT=YES              CVT  Dsect
         IHASCVT ,                    SCVT Dsect
*
         IHAASVT ,                    Define ASVT DSECT                 00080600
         IHAASCB ,                    Define ASCB DSECT                 00080700
         IHAASXB ,                    ASXB Dsect
         IHAASSB ,                    Define ASSB DSECT                 00080800
         IHAASTE ,                    Define ASTE DSECT                 00080900
*
         IKJTCB  ,                    TCB  Dsect
         IHASTCB ,                    STCB Dsect
         IHARB   ,                    PRB  Dsect
         PRINT ON,NOGEN
*                                                                       00081202
         DUCDSECT                     Define DUCT DSECTs                00081302
         PRINT ON,NOGEN                                                 00081400
*                                                                       00081202
***********************************************************************
* RDHARB01 - Program Main Entry Point                                 *
***********************************************************************
*
         USING PSA,R0               Establish Addressability
         USING DSPDSECT,R9          Establish Addressability
*
RDHARB01 PGMNTRY TYPE=MAIN,         Establish Main Entry Point         *
               BASE=(R12,R11),      (Same)                             *
               IMP=BASR
*
** Program MainLine
*
         GOSUB INIT                   Do Init Processing                00130000
         GOSUB MAIN                   Do Main Processing                00131000
         GOSUB WTOR                   Issue WTOR & Wait for Response    00131104
         GOSUB TERM                   Do Term Processing                00132000
*
** Return to Caller
*
EXIT     DS    0H
         PGMEXIT RC=0               Return to Caller
*                                                                       00301000
```

```
*=====================================================================* 00302000
** INIT - Do Initialization Processing                                * 00303000
*=====================================================================* 00304000
*                                                                       00305000
*   We have to first obtain a Play Page to move data from real          00305102
* storage locations where this data is stored because the #TRACE        00305202
* macro has yet to be enhanced to display real storage locations        00305302
* or to access data in AR mode.                                         00305402
*                                                                       00305502
INIT     SUBNTRY ,                                                      00306000
         GETMAIN RU,LV=PLAYPGLN,       Obtain 4K for a Play Page       *00306400
               BNDRY=PAGE              (Same)                           00306500
         ST    R1,DSAPLAYP             Save Play Page Address           00306600
*                                                                       00308301
** Show ASTE from CR5 and DUCT from CR2                                 00308402
*                                                                       00308502
         STCTG CR0,CR15,DSACNTRL       Store Control Registers          00308602
         #TRACE DATA=('Control Registers On Entry',DSACNTRL,16*8)       00308702
*                                                                       00308802
         LG    R3,DSACNTRL+5*8         Get Address of Primary ASCE      00308902
         L     R2,DSAPLAYP             Set To Address                   00309002
         LH    R4,=Y(64)               Set Length                       00309102
         MFREAL 0(R2),0(R3),           Show me the Segment Table       *00309202
               LEN=(R4)                (Same)                           00309302
         #TRACE DATA=('ASTE From CR5',0(R2),64),REGS=(R3)               00309402
*                                                                       00310202
         LG    R3,DSACNTRL+2*8         Get Address of DUCT              00309602
         L     R2,DSAPLAYP             Set To Address                   00309702
         LH    R4,=Y(DUCTLNTH)         Set Length                       00309802
         MFREAL 0(R2),0(R3),           Show me the Segment Table       *00309902
               LEN=(R4)                (Same)                           00310002
         L     R3,DUCTDUAL-DUCTO(,R2)  Get Real Address of DUAL         00310302
         ST    R3,DSADUALD             Address of DUAL From Duct
*                                                                       00310802
** Make Us non-Swapable - Required for Common Dataspaces                00310904
*                                                                       00311002
         XR    R1,R1                   Swapping Out This ASID           00311104
         SYSEVENT TRANSWAP             Make Us Noon-Swapable            00311304
*                                                                       00312304
** Return to Caller                                                     00319502
*                                                                       00319502
INIT999  DS    0H                                                       00319602
         SUBEXIT ,                                                      00319702
*                                                                       00319802
K        EQU   1024                                                     00319902
PLAYPGLN EQU   4*K                                                      00320002
*                                                                       00320102
*=====================================================================* 00320202
** TERM - Do Termination Processing                                   * 00320302
*=====================================================================* 00320402
*                                                                       00320502
TERM     SUBNTRY ,                                                      00320602
         SYSEVENT OKSWAP               Remove Non-Swappability          00320704
*                                                                       00320804
         L     R2,DSAPLAYP             Get Play Page Address            00320902
         FREEMAIN RU,LV=PLAYPGLN,      Free Storage for Segment Table  *00321002
               A=(R2)                  (Same)                           00321102
*                                                                       00321202
         GOSUB DDSP                    Delete Test Dataspaces           00321302
*                                                                       00321402
** Return to Caller                                                     00321402
*                                                                       00321402
TERM999  DS    0H                                                       00321502
```

```
         SUBEXIT ,                                                      00321602
*                                                                       00427804
*=====================================================================* 00427904
** WTOR - issue WTOR & Wait for Response                              * 00428004
*=====================================================================* 00428104
*                                                                       00428204
*   This routine alllows me to stop after initializing the environment  00428304
* providing the opportunity to dump the address space and view things   00428404
* under IPCS.                                                           00428504
*                                                                       00428604
WTOR     SUBNTRY ,                                                      00428704
         MVC   DSAWTOR,RWTOR                                            00428804
*temp    WTOR  'RDH0001: RDHTPALV waiting to terminate.',              *00428904
               DSAWORKD,8,DSAECBWT,MF=(E,DSAWTOR)                       00429004
*temp    WAIT ECB=DSAECBWT           Wait for operator Response         00429104
*                                                                       00429204
** Return to Caller                                                     00429204
*                                                                       00429204
WTOR999  DS    0H                                                       00429304
         SUBEXIT ,                                                      00429404
*                                                                       00429504
RWTOR    WTOR  'RDH0001: RDHTPALV waiting to terminate.',              *00429604
               *-*,8,*-*,MF=L                                           00429704
RWTORL   EQU   *-RWTOR                                                  00429804
*
*=====================================================================*
** MAIN - Main Processing Routine                                     *
*=====================================================================*
*
MAIN     SUBNTRY ,
         GOSUB DASTB                   Display Tables Before            00312504
         GOSUB CDSP                    Create Test Dataspaces           00312604
         GOSUB DASTA                   Display Tables After             00313002
*
         #TRACE MSG=' ',NOHEAD=YES
         #TRACE DATA=('Single STKN/ORGN/ALET 1',DSA1STKN,32),NOHEAD=YES 00409602
         #TRACE DATA=('Common STKN/ORGN/ALET 2',DSA2STKN,32),NOHEAD=YES 00413404
         #TRACE DATA=('All    STKN/ORGN/ALET 3',DSA3STKN,32),NOHEAD=YES 00416304
         #TRACE MSG=' ',NOHEAD=YES
*
** Test Printing Data From Dataspace With #TRACE
*
         GOSUB TEST                    #TRACE in ARMODE Testing Here
*
** Return to Caller
*
MAIN999  DS    0H
         SUBEXIT ,
*
*=====================================================================*
** TEST - Test #TRACE Macro in an ARMODE Program                      *
*=====================================================================*
*
TEST     SUBNTRY ,
         XGR   R2,R2                   Dataspace Starts at Loc Zero
         SAC   512                     Enter ARMADE
*
** Test DATA= Getting Data From Dataspace
*
         #TRACE MSG='Testing DATA= Parameter With Data in Dataspace.'
         LAM   AR2,AR2,DSA1ALET        Load ALET for Dataspace 1
         #TRACE DATA=('Data From Dataspace 1',0(R2),128)
*
```

```
         LAM   AR2,AR2,DSA2ALET        Load ALET for Dataspace 2
         #TRACE DATA=('Data From Dataspace 2',0(R2),128)
*
         LAM   AR2,AR2,DSA3ALET        Load ALET for Dataspace 3
         #TRACE DATA=('Data From Dataspace 3',0(R2),128)
*
** Test MSG= Getting Variable Data From Dataspace
*
         #TRACE MSG=('Showing Dataspace CL8 Data: ',DSPWKCL8)
         #TRACE MSG=('Showing Dataspace ZL8 Data: ',DSPWKZL8)
         #TRACE MSG=('Showing Dataspace XL8 Data: ',DSPWKXL8)
         #TRACE MSG=('Showing Dataspace WKF Data: ',DSPWORKF)
         #TRACE MSG=('Showing Dataspace WKH Data: ',DSPWORKH)
*
         #TRACE MSG=('Showing Dataspace PL8 Data: ',DSPWKPL8) Bug
*
         SAC   000                     Exit  ARMADE
*
** Return to Caller
*
TEST999  DS    0H
         SUBEXIT ,
*                                                                 00401000
*===============================================================* 00402000
** CDSP - Create a Few of Data Spaces                           * 00403004
*===============================================================* 00404000
*                                                                 00405000
CDSP     SUBNTRY MSG='Creating Dataspaces'                        00406002
*                                                                 00406104
** Build RDHDSP01 - SCOPE=SINGLE                                  00406204
*                                                                 00406304
         DSPSERV CREATE,           -*- Create a Data Space       *00406404
               NAME==C'RDHDSP01',      ..Name                    *00406502
               STOKEN=DSA1STKN,        ..Stoken Returned         *00406602
               BLOCKS==A(10),          ..Get 40K 10-4K Blocks    *00406702
               ORIGIN=DSA1ORGN,        ..Starting at Zero        *00406802
               SCOPE=SINGLE,           ..Local Dataspace         *00406904
               MF=(E,DSADSPL)                                     00407004
*                                                                 00407102
         LA    R3,DSA1STKN             Point at Dataspace STOKEN  00407202
         XC    DSAALE,DSAALE           Zero Parameter Area        00407302
         ALESERV ADD,              -*- Get ALET for Dataspace - PASN *00407404
               STOKEN=0(R3),           STOKEN from MVS DSPSERV   *00407502
               ALET=(R4),              Output ALET address       *00407602
               CHKEAX=NO,              Don't check EAX table     *00407702
               CHKPT=IGNORE,           Don't inhibit CHKPTs      *00407802
               AL=PASN,                Create ALET on PASN list  *00407902
               MF=(E,DSAALE)           Execute form               00408002
         ST    R4,DSA1ALET             Save ALET Value            00408102
*                                                                 00407102
         LA    R3,DSA1STKN             Point at Dataspace STOKEN  00407202
         XC    DSAALE,DSAALE           Zero Parameter Area        00407302
         ALESERV ADD,              -*- Get ALET for Dataspace - DUAL *00407404
               STOKEN=0(R3),           STOKEN from MVS DSPSERV   *00407502
               ALET=(R4),              Output ALET address       *00407602
               CHKEAX=NO,              Don't check EAX table     *00407702
               CHKPT=IGNORE,           Don't inhibit CHKPTs      *00407802
               AL=WORKUNIT,            Create ALET on PASN list  *00407902
               MF=(E,DSAALE)           Execute form               00408002
         ST    R4,DSA1ALED             Save ALET Value            00408102
*                                                                 00408204
         LAM   AR0,AR15,=16A(0)    -*- Initialize The Dataspace   00408304
         SAC   512                     Shift Into AR Mode         00408404
         LAM   AR2,AR2,DSA1ALET        Get First ALET                  00408604
         XR    R2,R2                   Start at Location Zero          00408704
         MVC   0(8,R2),=C'RDHDSP01'    Put Eyecatcher in Dataspace     00408804
         LAM   AR2,AR2,DSA1ALED        Get DUAL  ALET                  00408604
         MVC   8(8,R2),=C'DUALALET'    Put Eyecatcher in Dataspace     00408804
         LAE   R9,0(,R2)
         GOSUB PDSP                    Populate With test Data
         SAC   000                     Back Into Primary Mode          00409404
*                                                                      00409702
** BUILD RDHDSP02 - SCOPE=COMMON                                       00409804
*                                                                      00409904
         DSPSERV CREATE,           -*- Create a Data Space            *00410004
               NAME==C'RDHDSP02',      ..Name                         *00410104
               STOKEN=DSA2STKN,        ..Stoken Returned              *00410204
               BLOCKS==A(10),          ..Get 40K 10-4K Blocks         *00410304
               ORIGIN=DSA2ORGN,        ..Starting at Zero             *00410404
               SCOPE=COMMON,           ..Common Database              *00410504
               MF=(E,DSADSPL)                                          00410604
*                                                                      00410704
         LA    R3,DSA2STKN             Point at Dataspace STOKEN       00410804
         XC    DSAALE,DSAALE           Zero Parameter Area             00410904
         ALESERV ADD,              -*- Get ALET for Dataspace - PASN  *00411004
               STOKEN=0(R3),           STOKEN from MVS DSPSERV        *00411104
               ALET=(R4),              Output ALET address            *00411204
               CHKEAX=NO,              Don't check EAX table          *00411304
               CHKPT=IGNORE,           Don't inhibit CHKPTs           *00411404
               AL=PASN,                Create ALET on PASN list       *00411504
               MF=(E,DSAALE)           Execute form                    00411604
         ST    R4,DSA2ALET             Save ALET Value                 00411704
*                                                                      00412004
         SAC   512                 -*- Initialize the Dataspace        00412204
         LAM   AR2,AR2,DSA2ALET        Get First ALET                  00412804
         XR    R2,R2                   Start at Location Zero          00412904
         MVC   0(8,R2),=C'RDHDSP02'    Put Eyecatcher in Dataspace     00413004
         MVC   8(8,R2),=C'PASNALET'    Put Eyecatcher in Dataspace     00408804
         LAE   R9,0(,R2)
         GOSUB PDSP                    Populate With test Data
         SAC   000                     Back Into Primary Mode          00413204
*                                                                      00413304
** BUILD RDHDSP03 - SCOPE=ALL                                          00413604
*                                                                      00413704
         DSPSERV CREATE,           -*- Create a Data Space            *00413804
               NAME==C'RDHDSP03',      ..Name                         *00413904
               STOKEN=DSA3STKN,        ..Stoken Returned              *00414004
               BLOCKS==A(10),          ..Get 40K 10-4K Blocks         *00414104
               ORIGIN=DSA3ORGN,        ..Starting at Zero             *00414204
               SCOPE=ALL,              ..An All Dataspace             *00414304
               MF=(E,DSADSPL)                                          00414404
*                                                                      00414504
         LA    R3,DSA3STKN             Point at Dataspace STOKEN       00414604
         XC    DSAALE,DSAALE           Zero Parameter Area             00414704
         ALESERV ADD,              -*- Get ALET for Dataspace - PASN  *00414804
               STOKEN=0(R3),           STOKEN from MVS DSPSERV        *00414904
               ALET=(R4),              Output ALET address            *00415004
               CHKEAX=NO,              Don't check EAX table          *00415104
               CHKPT=IGNORE,           Don't inhibit CHKPTs           *00415204
               AL=PASN,                Create ALET on PASN list       *00415304
               MF=(E,DSAALE)           Execute form                    00415404
         ST    R4,DSA3ALET             Save ALET Value                 00415504
*                                                                      00414504
         LA    R3,DSA3STKN             Point at Dataspace STOKEN       00414604
         XC    DSAALE,DSAALE           Zero Parameter Area             00414704
         ALESERV ADD,              -*- Get ALET for Dataspace - DUAL  *00414804
```

```
               STOKEN=0(R3),          STOKEN from MVS DSPSERV        *00414904
               ALET=(R4),             Output ALET address            *00415004
               CHKEAX=NO,             Don't check EAX table          *00415104
               CHKPT=IGNORE,          Don't inhibit CHKPTs           *00415204
               AL=WORKUNIT,           Create ALET on PASN list       *00415304
               MF=(E,DSAALE)          Execute form                    00415404
         ST    R4,DSA3ALED            Save ALET Value                 00415504
*                                                                     00415604
         SAC   512                -*- Initialize the Dataspace        00415704
         LAM   AR2,AR2,DSA3ALET       Get First ALET                  00415804
         XR    R2,R2                  Start at Location Zero          00415904
         MVC   0(8,R2),=C'RDHDSP03'   Put Eyecatcher in Dataspace     00416004
         LAM   AR2,AR2,DSA3ALED       Get DUAL  ALET                  00408604
         MVC   8(8,R2),=C'DUALALET'   Put Eyecatcher in Dataspace     00408804
         LAE   R9,0(,R2)
         GOSUB PDSP                   Populate With test Data
         SAC   000                    Back Into Primary Mode          00416104
*                                                                     00417702
** Return To Caller                                                   00417702
*                                                                     00417702
CDSP999  DS    0H                                                     00417802
         SUBEXIT ,                                                    00417902
*                                                                     00423404
*=====================================================================* 00423504
** PDSP - Populate Dataspace                                          * 00423604
*=====================================================================* 00423704
*                                                                     00423804
PDSP     SUBNTRY ,                                                    00423904
         MVC   DSPWKCL8,=CL8'ABCDEFGH'
         MVC   DSPWKZL8,=CL8'12345678'
         MVC   DSPWKXL8,=XL8'0102030405060708'
         ZAP   DSPWKPL8,=PL8'1235'
         MVC   DSPWORKF,=F'1222'
         MVC   DSPWORKH,=H'166'
*                                                                     00417702
** Return To Caller                                                   00417702
*                                                                     00417702
PDSP999  DS    0H                                                     00417802
         SUBEXIT ,                                                    00417902
*                                                                     00423404
*=====================================================================* 00423504
** DDSP - Delete Our Test Data Spaces                                 * 00423604
*=====================================================================* 00423704
*                                                                     00423804
DDSP     SUBNTRY MSG='Deleting Data Spaces'                           00423904
         DSPSERV DELETE,              Delete a Data Space            *00424004
               STOKEN=DSA1STKN,       ..Stoken for Delete Request    *00424104
               MF=(E,DSADSPL)                                         00424204
*                                                                     00424304
         DSPSERV DELETE,              Delete a Data Space            *00424404
               STOKEN=DSA2STKN,       ..Stoken for Delete Request    *00424504
               MF=(E,DSADSPL)                                         00424604
*                                                                     00424304
         DSPSERV DELETE,              Delete a Data Space            *00424404
               STOKEN=DSA3STKN,       ..Stoken for Delete Request    *00424504
               MF=(E,DSADSPL)                                         00424604
*                                                                     00424704
** Return to Caller                                                   00424704
*                                                                     00424704
DDSP999  DS    0H                                                     00424804
         SUBEXIT ,                                                    00424904
*                                                                     00425004
*=====================================================================* 00425104
```

```
** DASTB - Display Tables Before Dataspace Create                    * 00425204
*====================================================================* 00425304
*                                                                      00425404
DASTB    SUBNTRY ,                                                     00425504
         #TRACE MSG=' ',NOHEAD=YES
         #TRACE MSG='===================================',NOHEAD=YES    00312404
         #TRACE MSG='***> Before Dataspace Creation <***',NOHEAD=YES    00312404
         #TRACE MSG='===================================',NOHEAD=YES    00312404
         #TRACE MSG=' ',NOHEAD=YES
*
         L     R10,PSAAOLD             Get Current ASCB Address         00311804
         USING ASCB,R10                & Addressability                 00311904
         L     R10,ASCBASTE            Get Base ASTE Address            00312004
         USING ASTE,R10                & Addressability                 00312104
         #TRACE DATA=('Base ASTE From ASCBASTE',0(R10),64),REGS=(R10)   00425804
*                                                                      00425904
         L     R3,ASTEATO              Auth Table Origin                00426004
         N     R3,=A(X'7FFFFFFC')      (Same)                           00426104
         L     R2,DSAPLAYP             Set To Address                   00426204
         LH    R4,=Y(768)              Set Length                       00426304
         MFREAL 0(R2),0(R3),           Show me the Authorization Table *00426404
               LEN=(R4)                (Same)                           00426504
*temp    #TRACE DATA=('ASTE ATO',0(R2),768),REGS=(R2,R3,R4)             00426604
*                                                                      00426704
         L     R3,ASTEPALD             PASN Access list                 00426804
         N     R3,=A(X'7FFFFF80')      (Same)                           00426904
         L     R2,DSAPLAYP             Set To Address                   00427004
         LH    R4,=Y(1024)             Set Length                       00427104
         MFREAL 0(R2),0(R3),           Show me the Segment Table       *00427204
               LEN=(R4)                (Same)                           00427304
         #TRACE DATA=('ASTEPALD - PASN-AL Before',0(R2),1024),         *00427404
               REGS=(R2,R3,R4)                                          00427404
*                                                                      00310202
         LG    R3,DSACNTRL+2*8         Get Address of DUCT              00309602
         L     R2,DSAPLAYP             Set To Address                   00309702
         LH    R4,=Y(DUCTLNTH)         Set Length                       00309802
         MFREAL 0(R2),0(R3),           Show me the Segment Table       *00309902
               LEN=(R4)                (Same)                           00310002
         #TRACE DATA=('DUCT From CR2',0(R2),DUCTLNTH),REGS=(R3)         00310102
*                                                                      00427504
         L     R2,DSAPLAYP             Set To Address                   00309002
         L     R3,DUCTDUAL-DUCTO(,R2)  Get Real Address of DUAL         00310302
         LH    R4,=Y(512)              Set Length                       00310402
         MFREAL 0(R2),0(R3),           Show me the Segment Table       *00310502
               LEN=(R4)                (Same)                           00310602
         #TRACE DATA=('DU-AL From DUCT Before',0(R2),256),REGS=(R3)     00310702
*                                                                      00427504
** Return to Caller                                                    00427504
*                                                                      00427504
DASTB99  DS    0H                                                      00427604
         SUBEXIT ,                                                     00427704
*                                                                      00425004
*====================================================================* 00425104
** DASTA - Display Table After Dataspace Create                      * 00425204
*====================================================================* 00425304
*                                                                      00425404
DASTA    SUBNTRY ,                                                     00425504
         #TRACE MSG=' ',NOHEAD=YES
         #TRACE MSG='==================================',NOHEAD=YES     00312404
         #TRACE MSG='***> After Dataspace Creation <***',NOHEAD=YES     00312404
         #TRACE MSG='==================================',NOHEAD=YES     00312404
         #TRACE MSG=' ',NOHEAD=YES
*
```

```
         L     R10,PSAAOLD            Get Current ASCB Address         00311804
         USING ASCB,R10               & Addressability                 00311904
         L     R10,ASCBASTE           Get Base ASTE Address            00312004
         USING ASTE,R10               & Addressability                 00312104
         #TRACE DATA=('Base ASTE From ASCBASTE',0(R10),64),REGS=(R10)   00425804
*                                                                      00425904
         L     R3,ASTEATO             Auth Table Origin                00426004
         N     R3,=A(X'7FFFFFFC')     (Same)                           00426104
         L     R2,DSAPLAYP            Set To Address                   00426204
         LH    R4,=Y(768)             Set Length                       00426304
         MFREAL 0(R2),0(R3),          Show me the Authorization Table *00426404
               LEN=(R4)               (Same)                           00426504
*temp    #TRACE DATA=('ASTE ATO',0(R2),768),REGS=(R2,R3,R4)            00426604
*                                                                      00426704
         L     R3,ASTEPALD            PASN Access list                 00426804
         N     R3,=A(X'7FFFFF80')     (Same)                           00426904
         L     R2,DSAPLAYP            Set To Address                   00427004
         LH    R4,=Y(1024)            Set Length                       00427104
         MFREAL 0(R2),0(R3),          Show me the Segment Table       *00427204
               LEN=(R4)               (Same)                           00427304
         #TRACE DATA=('ASTE PSALD After',0(R2),1024),REGS=(R2,R3,R4)   00427404
*                                                                      00310202
         LG    R3,DSACNTRL+2*8        Get Address of DUCT              00309602
         L     R2,DSAPLAYP            Set To Address                   00309702
         LH    R4,=Y(DUCTLNTH)        Set Length                       00309802
         MFREAL 0(R2),0(R3),          Show me the Segment Table       *00309902
               LEN=(R4)               (Same)                           00310002
         #TRACE DATA=('DUCT From CR2',0(R2),DUCTLNTH),REGS=(R3)        00310102
         L     R3,DUCTDUAL-DUCTO(,R2)  Get Real Address of DUAL        00310302
         ST    R3,DSADUALD            Address of DUAL From Duct
*                                                                      00427504
         L     R2,DSAPLAYP            Set To Address                   00309002
         L     R3,DSADUALD            Address of DUAL From Duct
         LH    R4,=Y(512)             Set Length                       00310402
         MFREAL 0(R2),0(R3),          Show me the Segment Table       *00310502
               LEN=(R4)               (Same)                           00310602
         #TRACE DATA=('DU-AL From DUCT After',0(R2),256),REGS=(R3)     00310702
*                                                                      00416204
** Show ALET's Returned                                                00416204
*                                                                      00416204
         #TRACE MSG=' ',NOHEAD=YES
         #TRACE DATA=('Single STKN/ORGN/ALET 1',DSA1STKN,24),NOHEAD=YES 00409602
         #TRACE DATA=('Common STKN/ORGN/ALET 2',DSA2STKN,24),NOHEAD=YES 00413404
         #TRACE DATA=('All    STKN/ORGN/ALET 3',DSA3STKN,24),NOHEAD=YES 00416304
         #TRACE MSG=' ',NOHEAD=YES
*                                                                      00427504
** Return to Caller                                                    00427504
*                                                                      00427504
DASTA99  DS    0H                                                      00427604
         SUBEXIT ,                                                     00427704
*
*=====================================================================*
** SUBR - Model Subroutine for Replication                            *
*=====================================================================*
*
SUBR     SUBNTRY ,
*
*
SUBR999  DS    0H
         SUBEXIT ,
*
***********************************************************************
* Define Literials and Constants                                      *
```

```
***********************************************************************
*
         CONBEG ,
*
** Equates
*
HEXFF    EQU   X'FF'                   Common Equates
HEX00    EQU   X'00'                   (Same)
*
** Constants
*                                                                       00510100
ALEMODL  DS    0D                                                       00511000
         ALESERV MF=L                Access a data space                00512000
ALEMODLL EQU   *-ALEMODL                                                00513000
*
HEXTAB   DC    C'0123456789ABCDEF'   Hex Conversion Table
*
         CONEND ,
*
***********************************************************************
* Define Dynamic Storage Area                                         *
***********************************************************************
*
         DSABEG ,                      Define DSA Prefix
*
DSACNTRL DS    16D                       Control Registers              00640101
DSAPLAYP DS    A                         Play Page Address              00641000
DSABASTE DS    A                         Base ASTE Address              00642000
DSADUALD DS    A                         DUAL Address From DUCT         00642000
*                                                                       00650000
DSA1STKN DS    D                         Dataspace 1 STOKEN             00650100
DSA1ORGN DS    A                         Dataspace 1 Origin             00650200
DSA1ALET DS    A                         Dataspace 1 ALET PASN          00650300
DSA1ALED DS    A                         Dataspace 1 ALET DUAL          00650300
DSA1ASN2 DS    A                         Dataspace 1 ASN PASN           00650402
DSA1ASD2 DS    A                         Dataspace 2 ASN DU             00650902
         DS    A
*                                                                       00650502
DSA2STKN DS    D                         Dataspace 2 STOKEN             00650602
DSA2ORGN DS    A                         Dataspace 2 Origin             00650702
DSA2ALET DS    A                         Dataspace 2 ALET PASN          00650802
DSA2ALED DS    A                         Dataspace 2 ALET DUAL          00650802
DSA2ASN2 DS    A                         Dataspace 2 ASN PASN           00650902
DSA2ASD2 DS    A                         Dataspace 2 ASN DU             00650902
         DS    A
*                                                                       00650502
DSA3STKN DS    D                         Dataspace 3 STOKEN             00650602
DSA3ORGN DS    A                         Dataspace 3 Origin             00650702
DSA3ALET DS    A                         Dataspace 3 ALET PASN          00650802
DSA3ALED DS    A                         Dataspace 3 ALET DUAL          00650802
DSA3ASN2 DS    A                         Dataspace 3 ASN PASN           00650902
DSA3ASD2 DS    A                         Dataspace 3 ASN DU             00650902
         DS    A
*                                                                       00651002
DSAECBWT DS    A                         ECB for WTOR                   00651104
*                                                                       00651204
         DS    0D
DSAWK256 DS    CL256                     GP Workarea
DSAASN2E DS    CL128                     ASN 2ND Level Entry
*                                                                       00651204
** Define list Forms of IBM Macros - Uck! A Necessary Evil.             00651304
*                                                                       00651404
         DS    0D                                                       00651504
```

```
DSAWTOR  DS    XL(RWTORL)'0'          MF=L for WTOR                 00651604
         DS    0D                                                   00651704
*                                                                   00651804
DSAALE   DS    XL(ALEMODLL)           Space for ALESERV List Form   00651904
         DSPSERV MF=(L,DSADSPL)                                     00652004
*
         DSAEND ,                     Define DSA Suffex
*
** Parameters in Dataspace
*
DSPDSECT DSECT ,
DSPNAME  DS    CL8                    Dataspace Name
DSPALTYP DS    CL8                    ALET TYPL DUAL/PASN
*
DSPWKCL8 DS    CL8                    Diaplay Char     Data
DSPWKZL8 DS    ZL8                    Display Zoned    Data
DSPWKXL8 DS    XL8                    Display Hex      Data
DSPWKPL8 DS    PL8                    Display Packed   Data
DSPWORKF DS    F                      Display Fullword Data
DSPWORKH DS    H                      Display halfword Data
         END   ,
```

Trace Output Generated for ARMDBS01

03/02/10 TRCPRINT - Label Level Tracing - TCB: 007E6968 16:32:39

```
 P) PGME: RDHARB01 At: RDHARB01.RDHARB01+0000, Fr: Zos
          State(Problem), Key(8), Mode(Primary), Amode(31)
    R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
    R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
    R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
    R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
    R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
    R10 - 00000000/00000000_00000000   R11 - 00000000/00000000_007E6E88
    R12 - 00000000/00000000_832B1CEA   R13 - 00000000/00000000_00006F60
    R14 - 00000000/00000000_80FDC810   R15 - 00000000/00000000_99100838

    S) SUBE: INIT At: RDHARB01.RDHARB01+02D0, Fr: RDHARB01.RDHARB01+023E
       T) Trap: TRP0070N At: RDHARB01.RDHARB01+0302

          64-Bit Address    Ofst  Control Registers On Entry
          ===============================================================================
          00000000_000071D8 0000  00000000 DF08EE60 00000000 3DB94007  *       -        *
          00000000_000071E8 0010  00000000 249DCAC0 00000001 00C00031  *                *
          00000000_000071F8 0020  00000001 00000031 00000000 3DE79C40  *             X  *
          00000000_00007208 0030  00000000 FE000000 00000000 3DB94007  *                *
          00000000_00007218 0040  00000000 00000000 00000000 00000000  *                *
          00000000_00007228 0050  00000000 00000000 00000000 00000000  *                *
          00000000_00007238 0060  00000000 35CD2A57 00000000 3DB94007  *                *
          00000000_00007248 0070  00000000 DF8BDECF 00000000 7F569138  *              j *

       T) Trap: TRP0088N At: RDHARB01.RDHARB01+0350
          R03 - 00000000/00000000_3DE79C40


          64-Bit Address    Ofst  ASTE From CR5
          ===============================================================================
          00000000_1910C000 0000  35DC8D71 00000030 00000000 3DB94007  *                *
          00000000_1910C010 0010  3FFBA01F 00000001 B5DC900F 00FCA580  *              v *
          00000000_1910C020 0020  00000000 00000000 00000000 00000001  *                *
          00000000_1910C030 0030  00000000 00000000 00000000 00000000  *                *

    S) SUBX: INIT At: RDHARB01.RDHARB01+03B4, To: RDHARB01.RDHARB01+023E
    S) SUBE: MAIN At: RDHARB01.RDHARB01+04BA, Fr: RDHARB01.RDHARB01+0242
       S) SUBE: DASTB At: RDHARB01.RDHARB01+0AB8, Fr: RDHARB01.RDHARB01+04DA

             ===================================
             ***> Before Dataspace Creation <***
             ===================================

          T) Trap: TRP0222N At: RDHARB01.RDHARB01+0B68
             R10 - 00000000/00000000_0234EC40


             64-Bit Address    Ofst  Base ASTE From ASCBASTE
             ===============================================================================
             00000000_0234EC40 0000  35DC8D71 00000030 00000000 3DB94007  *                *
             00000000_0234EC50 0010  3FFBA01F 00000001 B5DC900F 00FCA580  *              v *
             00000000_0234EC60 0020  00000000 00000000 00000000 00000001  *                *
             00000000_0234EC70 0030  00000000 00000000 00000000 00000000  *                *

          T) Trap: TRP0256N At: RDHARB01.RDHARB01+0BEC
             R02 - 00000000/00000000_1910C000   R03 - 00000000/00000000_3FFBA000
             R04 - 00000000/00000000_00000400


             64-Bit Address    Ofst  ASTEPALD - PASN-AL Before
             ===============================================================================
             00000000_1910C000 0000  80000000 00000000 00000000 00000000  *                *
             00000000_1910C010 0010  80000000 00000000 00000000 0037801F  *                *
             00000000_1910C020 0020  00000000 00000000 3F1B0800 00000000  *                *
             00000000_1910C030 0030  01000002 00000000 3F1B0880 00000000  *                *
             00000000_1910C040 0040  00FF0001 00000000 3DE2A000 00000000  *         S      *
             00000000_1910C050 0050  00FF0001 00000000 3DE2A080 00000000  *         S      *
             00000000_1910C060 0060  00FF0001 00000000 3DE2A100 00000000  *         S      *
             00000000_1910C070 0070  00FF0001 00000000 3DE2A180 00000000  *         S      *
             00000000_1910C080 0080  00FF0001 00000000 3DE2A200 00000000  *         Ss     *
             00000000_1910C090 0090  00FF0001 00000000 3DE2A280 00000000  *         Ss     *
             00000000_1910C0A0 00A0  00FF0001 00000000 3DE2A300 00000000  *         St     *
             00000000_1910C0B0 00B0  00FF0001 00000000 3DE2A380 00000000  *         St     *
             00000000_1910C0C0 00C0  00FF0001 00000000 3DE2A400 00000000  *         Su     *
             00000000_1910C0D0 00D0  00FF0001 00000000 3DE2A480 00000000  *         Su     *
             00000000_1910C0E0 00E0  00FF0001 00000000 3DE2A500 00000000  *         Sv     *
             00000000_1910C0F0 00F0  00FF0001 00000000 3DE2A580 00000000  *         Sv     *
             00000000_1910C100 0100  00FF0001 00000000 3DE2A600 00000000  *         Sw     *
             00000000_1910C110 0110  00FF0001 00000000 3DE2A680 00000000  *         Sw     *
             00000000_1910C120 0120  00FF0001 00000000 3DE2A700 00000000  *         Sx     *
             00000000_1910C130 0130  00FF0001 00000000 3DE2A780 00000000  *         Sx     *
             00000000_1910C140 0140  00FF0001 00000000 3DE2A800 00000000  *         Sy     *
             00000000_1910C150 0150  00FF0001 00000000 3DE2A880 00000000  *         Sy     *
             00000000_1910C160 0160  00FF0001 00000000 3DE2A900 00000000  *         Sz     *
             00000000_1910C170 0170  00FF0001 00000000 3DE2A980 00000000  *         Sz     *
             00000000_1910C180 0180  00FF0001 00000000 3DE2AA00 00000000  *         S      *
             00000000_1910C190 0190  00FF0001 00000000 3DE2AA80 00000000  *         S      *
             00000000_1910C1A0 01A0  00FF0001 00000000 3DE2AB00 00000000  *         S      *
             00000000_1910C1B0 01B0  00FF0001 00000000 3DE2AB80 00000000  *         S      *
             00000000_1910C1C0 01C0  00FF0001 00000000 3DE2AC00 00000000  *         S      *
             00000000_1910C1D0 01D0  00FF0001 00000000 3DE2AC80 00000000  *         S      *
             00000000_1910C1E0 01E0  00FF0001 00000000 3DE2AD00 00000000  *         S      *
             00000000_1910C1F0 01F0  00FF0001 00000000 3DE2AD80 00000000  *         S      *
             00000000_1910C200 0200  00FF0001 00000000 3DE2AE00 00000000  *         S      *
             00000000_1910C210 0210  00FF0001 00000000 3DE2AE80 00000000  *         S      *
             00000000_1910C220 0220  00FF0001 00000000 3DE2AF00 00000000  *         S      *
             00000000_1910C230 0230  00FF0001 00000000 3DE2AF80 00000000  *         S      *
             00000000_1910C240 0240  00FF0001 00000000 3DE29000 00000000  *         S      *
             00000000_1910C250 0250  00FF0001 00000000 3DE29080 00000000  *         S      *
             00000000_1910C260 0260  00FF0001 00000000 3DE29100 00000000  *         Sj     *
             00000000_1910C270 0270  00FF0001 00000000 3DE29180 00000000  *         Sj     *
             00000000_1910C280 0280  00FF0001 00000000 3DE29200 00000000  *         Sk     *
             00000000_1910C290 0290  00FF0001 00000000 3DE29280 00000000  *         Sk     *
             00000000_1910C2A0 02A0  00FF0001 00000000 3DE29300 00000000  *         Sl     *
             00000000_1910C2B0 02B0  00FF0001 00000000 3DE29380 00000000  *         Sl     *
             00000000_1910C2C0 02C0  00FF0001 00000000 3DE29400 00000000  *         Sm     *
             00000000_1910C2D0 02D0  00FF0001 00000000 3DE29480 00000000  *         Sm     *
             00000000_1910C2E0 02E0  00FF0001 00000000 3DE29500 00000000  *         Sn     *
             00000000_1910C2F0 02F0  00FF0001 00000000 3DE29580 00000000  *         Sn     *
             00000000_1910C300 0300  00FF0001 00000000 3DE29600 00000000  *         So     *
             00000000_1910C310 0310  00FF0001 00000000 3DE29680 00000000  *         So     *
             00000000_1910C320 0320  00FF0001 00000000 3DE29700 00000000  *         Sp     *
             00000000_1910C330 0330  00FF0001 00000000 3DE29780 00000000  *         Sp     *
             00000000_1910C340 0340  00FF0001 00000000 3DE29800 00000000  *         Sq     *
             00000000_1910C350 0350  00FF0001 00000000 3DE29880 00000000  *         Sq     *
             00000000_1910C360 0360  01810003 00000000 3DE791C0 00000001  * a       Xj     *
             00000000_1910C370 0370  80000000 00000000 00000000 00380000  *                *
             00000000_1910C380 0380  80000000 00000000 00000000 00390000  *                *
             00000000_1910C390 0390  80000000 00000000 00000000 003A0000  *                *
             00000000_1910C3A0 03A0  80000000 00000000 00000000 003B0000  *                *
             00000000_1910C3B0 03B0  80000000 00000000 00000000 003C0000  *                *
             00000000_1910C3C0 03C0  80000000 00000000 00000000 003D0000  *                *
             00000000_1910C3D0 03D0  80000000 00000000 00000000 003E0000  *                *
             00000000_1910C3E0 03E0  80000000 00000000 00000000 003F0000  *                *
             00000000_1910C3F0 03F0  80000000 00000000 00000000 00400000  *                *

          T) Trap: TRP0274N At: RDHARB01.RDHARB01+0C3A
             R03 - 00000000/00000000_249DCAC0


             64-Bit Address    Ofst  DUCT From CR2
             =====================================================================
             00000000_1910C000 0000  3DE79C40 00000000 00000000 00000000  * X              *
             00000000_1910C010 0010  249DCF00 00000000 00000000 00000000  *                *
             00000000_1910C020 0020  00000000 00000000 00000000 00000000  *                *
             00000000_1910C030 0030  00000000 00000000 00000000           *                *

          T) Trap: TRP0292N At: RDHARB01.RDHARB01+0C86
             R03 - 00000000/00000000_249DCF00


             64-Bit Address    Ofst  DU-AL From DUCT Before
             ===============================================================================
             00000000_1910C000 0000  80000000 00000000 00000000 00000000  *                *
             00000000_1910C010 0010  80000000 00000000 00000000 00001000  *                *
             00000000_1910C020 0020  0000FFFF 00000000 3DE79C40 00000001  *        X       *
             00000000_1910C030 0030  80000000 00000000 00000000 00000000  *                *
             00000000_1910C040 0040  80000000 00000000 00000000 00000000  *                *
             00000000_1910C050 0050  80000000 00000000 00000000 00000000  *                *
             00000000_1910C060 0060  80000000 00000000 00000000 00000000  *                *
             00000000_1910C070 0070  80000000 00000000 00000000 00000000  *                *
             00000000_1910C080 0080  3DE79C40 00000000 00000000 00000000  * X              *
             00000000_1910C090 0090  249DCF00 00000000 00000000 00000000  *                *
             00000000_1910C0A0 00A0  00000000 00000000 00000000 00000000  *                *
             00000000_1910C0B0 00B0  00000000 00000000 00000000 00000000  *                *
             00000000_1910C0C0 00C0  00000000 00000000 00000000 00000000  *                *
             00000000_1910C0D0 00D0  00000000 00000000 C7C1E3C5 01000000  *       GATE     *
             00000000_1910C0E0 00E0  E1000028 D9C4C840 40404040 80000000  *    RDH         *
             00000000_1910C0F0 00F0  00000000 C5987250 7F5B0000 7FFF8B20  *    Eq   $      *

       S) SUBX: DASTB At: RDHARB01.RDHARB01+0CA8, To: RDHARB01.RDHARB01+04DA
       S) SUBE: CDSP At: RDHARB01.RDHARB01+0712, Fr: RDHARB01.RDHARB01+04DE
          Creating Dataspaces
          S) SUBE: PDSP At: RDHARB01.RDHARB01+099C, Fr: RDHARB01.RDHARB01+0802
          S) SUBX: PDSP At: RDHARB01.RDHARB01+09E2, To: RDHARB01.RDHARB01+0802
          S) SUBE: PDSP At: RDHARB01.RDHARB01+099C, Fr: RDHARB01.RDHARB01+089A
          S) SUBX: PDSP At: RDHARB01.RDHARB01+09E2, To: RDHARB01.RDHARB01+089A
          S) SUBE: PDSP At: RDHARB01.RDHARB01+099C, Fr: RDHARB01.RDHARB01+096E
          S) SUBX: PDSP At: RDHARB01.RDHARB01+09E2, To: RDHARB01.RDHARB01+096E
       S) SUBX: CDSP At: RDHARB01.RDHARB01+0978, To: RDHARB01.RDHARB01+04DE
       S) SUBE: DASTA At: RDHARB01.RDHARB01+0CCC, Fr: RDHARB01.RDHARB01+04E2

             ==================================
             ***> After Dataspace Creation <***
             ==================================

          T) Trap: TRP0310N At: RDHARB01.RDHARB01+0D7C
             R10 - 00000000/00000000_0234EC40


             64-Bit Address    Ofst  Base ASTE From ASCBASTE
             ===============================================================================
             00000000_0234EC40 0000  35DC8D71 00000030 00000000 3DB94007  *                *
             00000000_0234EC50 0010  1D52601F 00000001 B5DC900F 00FCA580  *  -          v *
             00000000_0234EC60 0020  00000000 00000000 00000000 00000001  *                *
             00000000_0234EC70 0030  00000000 00000000 00000000 00000000  *                *

          T) Trap: TRP0344N At: RDHARB01.RDHARB01+0E00
             R02 - 00810004/00000000_1910C000   R03 - 00000000/00000000_1D526000
             R04 - 00000000/00000000_00000400


             64-Bit Address    Ofst  ASTEPALD - PASN-AL After
             ===============================================================================
             00000000_1910C000 0000  80000000 00000000 00000000 00000000  *                *
             00000000_1910C010 0010  80000000 00000000 00000000 0039201F  *                *
             00000000_1910C020 0020  00000000 00000000 3F1B0800 00000000  *                *
             00000000_1910C030 0030  01000002 00000000 3F1B0880 00000000  *                *
             00000000_1910C040 0040  00FF0001 00000000 3DE2A000 00000000  *         S      *
             00000000_1910C050 0050  00FF0001 00000000 3DE2A080 00000000  *         S      *
             00000000_1910C060 0060  00FF0001 00000000 3DE2A100 00000000  *         S      *
             00000000_1910C070 0070  00FF0001 00000000 3DE2A180 00000000  *         S      *
             00000000_1910C080 0080  00FF0001 00000000 3DE2A200 00000000  *         Ss     *
```

```
    00000000_1910C090 0090  00FF0001 00000000 3DE2A280 00000000  *          Ss      *
    00000000_1910C0A0 00A0  00FF0001 00000000 3DE2A300 00000000  *          St      *
    00000000_1910C0B0 00B0  00FF0001 00000000 3DE2A380 00000000  *          St      *
    00000000_1910C0C0 00C0  00FF0001 00000000 3DE2A400 00000000  *          Su      *
    00000000_1910C0D0 00D0  00FF0001 00000000 3DE2A480 00000000  *          Su      *
    00000000_1910C0E0 00E0  00FF0001 00000000 3DE2A500 00000000  *          Sv      *
    00000000_1910C0F0 00F0  00FF0001 00000000 3DE2A580 00000000  *          Sv      *
    00000000_1910C100 0100  00FF0001 00000000 3DE2A600 00000000  *          Sw      *
    00000000_1910C110 0110  00FF0001 00000000 3DE2A680 00000000  *          Sw      *
    00000000_1910C120 0120  00FF0001 00000000 3DE2A700 00000000  *          Sx      *
    00000000_1910C130 0130  00FF0001 00000000 3DE2A780 00000000  *          Sx      *
    00000000_1910C140 0140  00FF0001 00000000 3DE2A800 00000000  *          Sy      *
    00000000_1910C150 0150  00FF0001 00000000 3DE2A880 00000000  *          Sy      *
    00000000_1910C160 0160  00FF0001 00000000 3DE2A900 00000000  *          Sz      *
    00000000_1910C170 0170  00FF0001 00000000 3DE2A980 00000000  *          Sz      *
    00000000_1910C180 0180  00FF0001 00000000 3DE2AA00 00000000  *          S       *
    00000000_1910C190 0190  00FF0001 00000000 3DE2AA80 00000000  *          S       *
    00000000_1910C1A0 01A0  00FF0001 00000000 3DE2AB00 00000000  *          S       *
    00000000_1910C1B0 01B0  00FF0001 00000000 3DE2AB80 00000000  *          S       *
    00000000_1910C1C0 01C0  00FF0001 00000000 3DE2AC00 00000000  *          S       *
    00000000_1910C1D0 01D0  00FF0001 00000000 3DE2AC80 00000000  *          S       *
    00000000_1910C1E0 01E0  00FF0001 00000000 3DE2AD00 00000000  *          S       *
    00000000_1910C1F0 01F0  00FF0001 00000000 3DE2AD80 00000000  *          S       *
    00000000_1910C200 0200  00FF0001 00000000 3DE2AE00 00000000  *          S       *
    00000000_1910C210 0210  00FF0001 00000000 3DE2AE80 00000000  *          S       *
    00000000_1910C220 0220  00FF0001 00000000 3DE2AF00 00000000  *          S       *
    00000000_1910C230 0230  00FF0001 00000000 3DE2AF80 00000000  *          S       *
    00000000_1910C240 0240  00FF0001 00000000 3DE29000 00000000  *          S       *
    00000000_1910C250 0250  00FF0001 00000000 3DE29080 00000000  *          S       *
    00000000_1910C260 0260  00FF0001 00000000 3DE29100 00000000  *          Sj      *
    00000000_1910C270 0270  00FF0001 00000000 3DE29180 00000000  *          Sj      *
    00000000_1910C280 0280  00FF0001 00000000 3DE29200 00000000  *          Sk      *
    00000000_1910C290 0290  00FF0001 00000000 3DE29280 00000000  *          Sk      *
    00000000_1910C2A0 02A0  00FF0001 00000000 3DE29300 00000000  *          Sl      *
    00000000_1910C2B0 02B0  00FF0001 00000000 3DE29380 00000000  *          Sl      *
    00000000_1910C2C0 02C0  00FF0001 00000000 3DE29400 00000000  *          Sm      *
    00000000_1910C2D0 02D0  00FF0001 00000000 3DE29480 00000000  *          Sm      *
    00000000_1910C2E0 02E0  00FF0001 00000000 3DE29500 00000000  *          Sn      *
    00000000_1910C2F0 02F0  00FF0001 00000000 3DE29580 00000000  *          Sn      *
    00000000_1910C300 0300  00FF0001 00000000 3DE29600 00000000  *          So      *
    00000000_1910C310 0310  00FF0001 00000000 3DE29680 00000000  *          So      *
    00000000_1910C320 0320  00FF0001 00000000 3DE29700 00000000  *          Sp      *
    00000000_1910C330 0330  00FF0001 00000000 3DE29780 00000000  *          Sp      *
    00000000_1910C340 0340  00FF0001 00000000 3DE29800 00000000  *          Sq      *
    00000000_1910C350 0350  00FF0001 00000000 3DE29880 00000000  *          Sq      *
    00000000_1910C360 0360  01810003 00000000 3DE791C0 00000001  * a        Xj      *
    00000000_1910C370 0370  00810001 00000000 35BC1000 0000000B  * a                *
    00000000_1910C380 0380  00810001 00000000 35440780 00000004  * a                *
    00000000_1910C390 0390  80000000 00000000 00000000 003A0000  *                  *
    00000000_1910C3A0 03A0  80000000 00000000 00000000 003B0000  *                  *
    00000000_1910C3B0 03B0  80000000 00000000 00000000 003C0000  *                  *
    00000000_1910C3C0 03C0  80000000 00000000 00000000 003D0000  *                  *
    00000000_1910C3D0 03D0  80000000 00000000 00000000 003E0000  *                  *
    00000000_1910C3E0 03E0  80000000 00000000 00000000 003F0000  *                  *
    00000000_1910C3F0 03F0  80000000 00000000 00000000 00400000  *                  *

T)  Trap: TRP0362N At: RDHARB01.RDHARB01+0E4E
    R03 - 00000000/00000000_249DCAC0


    64-Bit Address    Ofst  DUCT From CR2
    ===============================================================================
    00000000_1910C000 0000  3DE79C40 00000000 00000000 00000000  * X                *
    00000000_1910C010 0010  249DC400 00000000 00000000 00000000  *  D               *
    00000000_1910C020 0020  00000000 00000000 00000000 00000000  *                  *
    00000000_1910C030 0030  00000000 00000000 00000000           *                  *

T)  Trap: TRP0380N At: RDHARB01.RDHARB01+0EA2
    R03 - 00000000/00000000_249DC400
```

```
        64-Bit Address    Ofst  DU-AL From DUCT After
        ===============================================================================
        00000000_1910C000 0000  80000000 00000000 00000000 00000000  *                *
        00000000_1910C010 0010  80000000 00000000 00000000 00050000  *                *
        00000000_1910C020 0020  0000FFFF 00000000 3DE79C40 00000001  *         X      *
        00000000_1910C030 0030  00810001 00000000 35BC1000 0000000B  * a              *
        00000000_1910C040 0040  00810001 00000000 35440780 00000004  * a              *
        00000000_1910C050 0050  80000000 00000000 00000000 00060000  *                *
        00000000_1910C060 0060  80000000 00000000 00000000 00070000  *                *
        00000000_1910C070 0070  80000000 00000000 00000000 00000000  *                *
        00000000_1910C080 0080  00000000 00000000 00000000 00000000  *                *
        00000000_1910C090 0090  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0A0 00A0  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0B0 00B0  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0C0 00C0  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0D0 00D0  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0E0 00E0  00000000 00000000 00000000 00000000  *                *
        00000000_1910C0F0 00F0  E2C4E501 00000000 00000000 00000000  *SDV             *


        64-Bit Address    Ofst  Single STKN/ORGN/ALET 1
        ===============================================================================
        00000000_00007268 0000  80001301 00000026 00000000 01810037  *             a  *
        00000000_00007278 0010  00810003 00000000                    * a              *


        64-Bit Address    Ofst  Common STKN/ORGN/ALET 2
        ===============================================================================
        00000000_00007288 0000  80001401 00000027 00000000 01FF001E  *                *
        00000000_00007298 0010  00000000 00000000                    *                *


        64-Bit Address    Ofst  All    STKN/ORGN/ALET 3
        ===============================================================================
        00000000_000072A8 0000  80001501 00000028 00000000 01810038  *             a  *
        00000000_000072B8 0010  00810004 00000000                    * a              *


S) SUBX: DASTA At: RDHARB01.RDHARB01+0F50, To: RDHARB01.RDHARB01+04E2


    64-Bit Address    Ofst  Single STKN/ORGN/ALET 1
    ===============================================================================
    00000000_00007268 0000  80001301 00000026 00000000 01810037  *             a  *
    00000000_00007278 0010  00810003 00000000 00000000 00000000  * a              *


    64-Bit Address    Ofst  Common STKN/ORGN/ALET 2
    ===============================================================================
    00000000_00007288 0000  80001401 00000027 00000000 01FF001E  *                *
    00000000_00007298 0010  00000000 00000000 00000000 00000000  *                *


    64-Bit Address    Ofst  All    STKN/ORGN/ALET 3
    ===============================================================================
    00000000_000072A8 0000  80001501 00000028 00000000 01810038  *             a  *
    00000000_000072B8 0010  00810004 00000000 00000000 00000000  * a              *


S) SUBE: TEST At: RDHARB01.RDHARB01+059C, Fr: RDHARB01.RDHARB01+0572
   T) Trap: TRP0154N At: RDHARB01.RDHARB01+05C0
      Testing DATA= Parameter With Data in Dataspace.
   T) Trap: TRP0156N At: RDHARB01.RDHARB01+05E0

        64-Bit Address    Ofst  Data From Dataspace 1
        ===============================================================================
        00000000_00000000 0000  D9C4C8C4 E2D7F0F1 C4E4C1D3 C1D3C5E3  *RDHDSP01DUALALET*
        00000000_00000010 0010  C1C2C3C4 C5C6C7C8 F1F2F3F4 F5F6F7F8  *ABCDEFGH12345678*
        00000000_00000020 0020  01020304 05060708 00000000 0001235C  *                *
        00000000_00000030 0030  000004C6 00A60000 00000000 00000000  *    F w         *
```

```
            00000000_00000040 0040  00000000 00000000 00000000 00000000  *                *
            00000000_00000050 0050  00000000 00000000 00000000 00000000  *                *
            00000000_00000060 0060  00000000 00000000 00000000 00000000  *                *
            00000000_00000070 0070  00000000 00000000 00000000 00000000  *                *

         T) Trap: TRP0158N At: RDHARB01.RDHARB01+0600

            64-Bit Address    Ofst  Data From Dataspace 2
            ===============================================================================
            00000000_00000000 0000  D9C4C8C4 E2D7F0F2 D7C1E2D5 C1D3C5E3  *RDHDSP02PASNALET*
            00000000_00000010 0010  C1C2C3C4 C5C6C7C8 F1F2F3F4 F5F6F7F8  *ABCDEFGH12345678*
            00000000_00000020 0020  01020304 05060708 00000000 0001235C  *                *
            00000000_00000030 0030  000004C6 00A60000 00000000 00000000  *   F w          *
            00000000_00000040 0040  00000000 00000000 00000000 00000000  *                *
            00000000_00000050 0050  00000000 00000000 00000000 00000000  *                *
            00000000_00000060 0060  00000000 00000000 00000000 00000000  *                *
            00000000_00000070 0070  00000000 00000000 00000000 00000000  *                *

         T) Trap: TRP0160N At: RDHARB01.RDHARB01+0620

            64-Bit Address    Ofst  Data From Dataspace 3
            ===============================================================================
            00000000_00000000 0000  D9C4C8C4 E2D7F0F3 C4E4C1D3 C1D3C5E3  *RDHDSP03DUALALET*
            00000000_00000010 0010  C1C2C3C4 C5C6C7C8 F1F2F3F4 F5F6F7F8  *ABCDEFGH12345678*
            00000000_00000020 0020  01020304 05060708 00000000 0001235C  *                *
            00000000_00000030 0030  000004C6 00A60000 00000000 00000000  *   F w          *
            00000000_00000040 0040  00000000 00000000 00000000 00000000  *                *
            00000000_00000050 0050  00000000 00000000 00000000 00000000  *                *
            00000000_00000060 0060  00000000 00000000 00000000 00000000  *                *
            00000000_00000070 0070  00000000 00000000 00000000 00000000  *                *

         T) Trap: TRP0162N At: RDHARB01.RDHARB01+063C
            Showing Dataspace CL8 Data: ABCDEFGH
         T) Trap: TRP0164N At: RDHARB01.RDHARB01+0658
            Showing Dataspace ZL8 Data: 12345678
         T) Trap: TRP0166N At: RDHARB01.RDHARB01+0674
            Showing Dataspace XL8 Data: 0102030405060708
         T) Trap: TRP0168N At: RDHARB01.RDHARB01+0690
            Showing Dataspace WKF Data: 1,222
         T) Trap: TRP0170N At: RDHARB01.RDHARB01+06AC
            Showing Dataspace WKH Data: 166
         T) Trap: TRP0172N At: RDHARB01.RDHARB01+06C8
            Showing Dataspace PL8 Data: 1,235
      S) SUBX: TEST At: RDHARB01.RDHARB01+06EE, To: RDHARB01.RDHARB01+0572
   S) SUBX: MAIN At: RDHARB01.RDHARB01+0578, To: RDHARB01.RDHARB01+0242
   S) SUBE: WTOR At: RDHARB01.RDHARB01+0438, Fr: RDHARB01.RDHARB01+0246
   S) SUBX: WTOR At: RDHARB01.RDHARB01+0460, To: RDHARB01.RDHARB01+0246
   S) SUBE: TERM At: RDHARB01.RDHARB01+03D8, Fr: RDHARB01.RDHARB01+024A
      S) SUBE: DDSP At: RDHARB01.RDHARB01+0A06, Fr: RDHARB01.RDHARB01+040E
         Deleting Data Spaces
      S) SUBX: DDSP At: RDHARB01.RDHARB01+0A94, To: RDHARB01.RDHARB01+040E
   S) SUBX: TERM At: RDHARB01.RDHARB01+0414, To: RDHARB01.RDHARB01+024A

P) PGMX: RDHARB01 At: RDHARB01.RDHARB01+0254, To: Zos
         State(Supervisor), Key(0), Mode(Primary), Amode(31)
   R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
   R02 - 01810038/00000000_1910C000   R03 - 00000000/00000000_249DC400
   R04 - 00000000/00000000_00000200   R05 - 00000000/00000000_007E6E88
   R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
   R08 - 00000000/00000000_007FCE28   R09 - 00810004/00000000_00000000
   R10 - 00000000/00000000_0234EC40   R11 - 00000000/00000000_19101838
   R12 - 00000000/00000000_19100838   R13 - 00000000/00000000_00007000
   R14 - 00000000/00000000_99100A82   R15 - 00000000/00000000_00000000
```

C.3 Test for TSTSVC1P Calling TSTSVC1 – Traces in a User SVC

This is a test where we create an SVC which I then install as SVC 230. This SVC uses the SVCNTRY and SVCEXIT macros which are compatible with the PGMNTRY and PGMEXIT macros. What I was doing in this test was assuring that I could get traces out from within a User SVC.

It's a simple test and doesn't really do anything. All I have is a program that issues a test SVC. But the cool thing is that in the trace output, we see the traces for the program, and then we see the traces from the SVC, followed by the final traces for the program. So, this can be very helpful for anyone writing a user SVC.

This ability to provide traces within an SVC was particularly helpful when I wrote the SVC to support RDHGATB, which is GETMAIN and FREEMAIN support for above-the-bar storage in increments of less than a megabyte.

This technology also establishes a model that I can later use when I develop PCNTRY, PCEXIT and SRBNTRY, SRBEXIT. If I can do it with an SVC, I can do it with a PC or an SRB also. Fate accompli, when I get the time.

Code for TSTSVC1P

```
***********************************************************************
*                                                                     *
** RDHTTSVC - Test Label Level Tracing Within an SVC                  *
*                                                                     *
*    This test program is intended to test the traces within an SVC   *
*  environment.  This program issues an SVC 230, which should be      *
*  RDHTSSVC, which is a test SVC usng a BASR implementation of the    *
*  traces.  Its a very simple program, but an important test.         *
*    This program also has tracing active, so both programs will      *
*  write trace output to the TRCPRINT DD in the JCL.                  *
*                                                                     *
*                                                                     *
***********************************************************************
*
*---------------------------------------------------------------------*
* Define MVS System Data Areas                                        *
*---------------------------------------------------------------------*
*
         PRINT OFF
         IHAPSA  ,
         PRINT ON,NOGEN
*
***********************************************************************
* RDHTTSVC - Program Main Entry Point                                 *
***********************************************************************
*
         USING PSA,R0                Establish Addressability
*
RDHTTSVC PGMNTRY TYPE=MAIN,          Establish Main Entry Point        *
               BASE=(R12,R11),PRINT=GEN,IMP=BASR
*
** Program MainLine
*
*        GOSUB INIT                  Do Init Processing
         GOSUB MAIN                  Do Main Processing
*        GOSUB TERM                  Do Term Processing
*
** Return to Caller
*
EXIT     DS    0H
         PGMEXIT RC=0                Return to Caller
*
*=====================================================================*
** MAIN - Main Processing Routine                                     *
*=====================================================================*
*
MAIN     SUBNTRY ,
         LM    R2,R5,=A(1,2,3,4)
         LA    R1,*+6
         SVC   230
*
MAIN999  DS    0H
         SUBEXIT ,
*
*=====================================================================*
** INIT - Do Initialization Processing                                *
*=====================================================================*
*
INIT     SUBNTRY ,
*
```

```
*                                                                       00591005
INIT999  DS    0H                                                       00600000
         SUBEXIT ,                                                      00610000
*                                                                       00620000
*=====================================================================* 00630000
** TERM - Do Termination Processing                                   * 00640000
*=====================================================================* 00650000
*                                                                       00660000
TERM     SUBNTRY ,                                                      00670000
*                                                                       00690000
*                                                                       00710000
TERM999  DS    0H                                                       00720000
         SUBEXIT ,                                                      00730000
*                                                                       00740000
*=====================================================================* 00750000
** SUBR - Model Subroutine for Replication                            * 00760000
*=====================================================================* 00770000
*                                                                       00780000
SUBR     SUBNTRY ,                                                      00790000
*                                                                       00800000
*                                                                       00810000
SUBR999  DS    0H                                                       00820000
         SUBEXIT ,                                                      00830000
*                                                                       00840000
*********************************************************************** 00850000
* Define Literials and Constants                                      * 00860000
*********************************************************************** 00870000
*                                                                       00880000
         CONBEG ,                                                       00890000
         LTORG ,                     Define Literial Pool               00900000
*                                                                       00910000
** Equates                                                              00920000
*                                                                       00930000
HEXFF    EQU   X'FF'                 Common Equates                     00940000
HEX00    EQU   X'00'                 (Same)                             00950000
*                                                                       00960000
** Constants                                                            00970000
*                                                                       00980000
HEXTAB   DC    C'0123456789ABCDEF'   Hex Conversion Table               00981000
*                                                                       01010000
         CONEND ,                                                       01020000
*                                                                       01030000
*********************************************************************** 01040000
* Define Dynamic Storage Area                                         * 01050000
*********************************************************************** 01060000
*                                                                       01070000
         DSABEG ,                    Define DSA Prefix                  01080000
*                                                                       01090000
** MF=L Work Areas                                                      01100000
*                                                                       01110000
         DS    0D                    Define Area for ESTAE              01120000
DSAESTA0 DS    XL(ESTAE0L)           (Same)                             01130000
DSARETRY DS    A                     Retry Address or Zero for Perc     01140000
*                                                                       01150000
DSAPLIST DS    4F                    Parameter List Work area           01160000
*                                                                       01170000
         DSAEND ,                    Define DSA Suffex                  01180000
         END   ,                                                        01319300
```

Code for TSTSVC1

```
**********************************************************************
*                                                                    *
** RDHTSSVC - Test SVC to Test Label Level Tracing in an SVC         *
*                                                                    *
*    This program uses IMP=BASR implementation of the traces to test *
*  the SVCNTRY/SVCEXIT code and the execution of the traces within   *
*  an SVC.                                                           *
*                                                                    *
**********************************************************************
*
*--------------------------------------------------------------------*
* Define MVS System Data Areas                                       *
*--------------------------------------------------------------------*
*
          PRINT OFF
          IHAPSA ,                     Define PSA DSECT Area
          PRINT ON
*
**********************************************************************
* RDHTSSVC - Program Main Entry Point                                *
**********************************************************************
*
         USING PSA,R0                  Define PSA Addressability
*
RDHTSSVC SVCNTRY SVCNO=230,            Program Entry Code             *
               BASE=(R12,R11),         Use Standard Bases             *
               PRINT=GEN,              (Same)                         *
               IMP=BASR                (Same)
*
         GOSUB INIT                    Perform Initialization
         GOSUB MAIN                    Do Main Processing
         GOSUB TERM                    Do Termination Processing
*
** Return to Caller
*
MAINEXIT DS    0H
         SVCEXIT RC=0                  Define Program Exit Code
*
**********************************************************************
* MAIN- Program Main Processing                                      *
**********************************************************************
*
MAIN     SUBNTRY ,
*
** Return to Caller
*
MAINEND  DS    0H
         SUBEXIT ,

*
**********************************************************************
* INIT - Perform Program Initialization                              *
**********************************************************************
*
INIT     SUBNTRY ,                 -*- Open SYSPRINT File
*
** Return to Caller
*
INITEND  DS    0H
         SUBEXIT ,
*
***********************************************************************
* TERM - Perform Program Termination                                  *
***********************************************************************
*
TERM     SUBNTRY ,
*
** Return to Caller
*
TERMEND  DS    0H
         SUBEXIT ,

***********************************************************************
*  Literials and Constants                                            *
***********************************************************************
*
         CONBEG ,
*
HEXFF    EQU   X'FF'
*
         CONEND ,
*
***********************************************************************
* Define Dynamic Storage Area - DSA                                   *
***********************************************************************
*
         DSABEG ,                <*** Begin Dynamic Storage Area
*
DSARC    DS    F                      Return Code
DSAPRWD1 DS    F                      SVRB Address
*
DSAFLAG  DS    B                      Flag Byte
*
         DSAEND ,                <*** End Dynamic Storage Area
         END ,
```

Trace Output for TSTSVC1P Calling TSTSVC1

```
03/02/10        TRCPRINT - Label Level Tracing - TCB: 007E6968        16:25:50


 P) PGME: RDHTTSVC At: RDHTTSVC.RDHTTSVC+0000, Fr: Zos
          State(Problem), Key(8), Mode(Primary), Amode(31)
    R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
    R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
    R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
    R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
    R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
    R10 - 00000000/00000000_00000000   R11 - 00000000/00000000_007E6E88
    R12 - 00000000/00000000_832B1CEA   R13 - 00000000/00000000_00006F60
    R14 - 00000000/00000000_80FDC810   R15 - 00000000/00000000_99100D10

    S) SUBE: MAIN At: RDHTTSVC.RDHTTSVC+02C4, Fr: RDHTTSVC.RDHTTSVC+023E

    P) PGME: RDHTSSVC At: RDHTTSVC.RDHTSSVC+0000, Fr: Zos
             State(Problem), Key(8), Mode(Primary), Amode(31)
       R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
       R02 - 00000000/00000000_00000040   R03 - 00000000/00000000_007D99D4
       R04 - 00000000/00000000_007D99B0   R05 - 00000000/00000000_007E6E88
       R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
       R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
       R10 - 00000000/00000000_00000000   R11 - 00000000/00000000_007E6E88
       R12 - 00000000/00000000_832B1CEA   R13 - 00000000/00000000_00006F60
       R14 - 00000000/00000000_80FDC810   R15 - 00000000/00000000_99100D10

       S) SUBE: INIT At: RDHTTSVC.RDHTSSVC+0274, Fr: RDHTTSVC.RDHTTSVC+4484
       S) SUBX: INIT At: RDHTTSVC.RDHTSSVC+0296, To: RDHTTSVC.RDHTTSVC+4484
       S) SUBE: MAIN At: RDHTTSVC.RDHTSSVC+022E, Fr: RDHTTSVC.RDHTTSVC+4488
       S) SUBX: MAIN At: RDHTTSVC.RDHTSSVC+0250, To: RDHTTSVC.RDHTTSVC+4488
       S) SUBE: TERM At: RDHTTSVC.RDHTSSVC+02BA, Fr: RDHTTSVC.RDHTTSVC+448C
       S) SUBX: TERM At: RDHTTSVC.RDHTSSVC+02DC, To: RDHTTSVC.RDHTTSVC+448C

    P) PGMX: RDHTSSVC At: RDHTTSVC.RDHTSSVC+01A6, To: Zos
             State(Supervisor), Key(0), Mode(Primary), Amode(31)
       R00 - 00000000/00000000_00000000   R01 - 00000000/00000000_00000000
       R02 - 00000000/00000000_7FFF9788   R03 - 00000000/00000000_00FDC7C0
       R04 - 00000000/00000000_007E6968   R05 - 00000000/00000000_007FD650
       R06 - 00000000/00000000_94FC5000   R07 - 00000000/00000000_00FC5400
       R08 - 00000000/00000000_00000000   R09 - 00000000/00000000_01968020
       R10 - 00000000/00000000_007FD710   R11 - 00000000/00000000_14FC6000
       R12 - 00000000/00000000_94FC5000   R13 - 00000000/00000000_007C1000
       R14 - 00000000/00000000_94FC519C   R15 - 00000000/00000000_00000000

    S) SUBX: MAIN At: RDHTTSVC.RDHTTSVC+02F0, To: RDHTTSVC.RDHTTSVC+023E

 P) PGMX: RDHTTSVC At: RDHTTSVC.RDHTTSVC+0248, To: Zos
          State(Supervisor), Key(0), Mode(Primary), Amode(31)
    R00 - 00000000/00000000_FD000008   R01 - 00000000/00000000_00006FF8
    R02 - 00000000/00000000_00000001   R03 - 00000000/00000000_00000002
    R04 - 00000000/00000000_00000003   R05 - 00000000/00000000_00000004
    R06 - 00000000/00000000_007C8FE0   R07 - 00000000/00000000_FD000000
    R08 - 00000000/00000000_007FCE28   R09 - 00000000/00000000_007E6B30
    R10 - 00000000/00000000_00000000   R11 - 00000000/00000000_19101D10
    R12 - 00000000/00000000_19100D10   R13 - 00000000/00000000_00007000
    R14 - 00000000/00000000_99100F4E   R15 - 00000000/00000000_00000000
```