EY-2217E-SG-0001

Guide to VAX-11/780 System Troubleshooting

FOR INTERNAL USE ONLY

digital™

First Printing, November 1985

The material in this document is for informational purposes only and is subject to change without notice; it should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Some portions of this manual are copied from other manuals, microfiche, etc. The reason for this is to provide as much information as possible in a single, easily carried manual.

UNIX is a trademark of AT&T.

The following are trademarks of Digital Equipment Corporation:

digital	MASSBUS	RT
DEC	PDP	UNIBUS
DECmate	P/OS	VAX
DECUS	Professional	VAXstation
DECwriter	Rainbow	VMS
DIBOL	RSTS	VT
LSI	RSX	Work Processor

Contents

P R E F A C E

This Trouble-shooting Manual was written as an aid to D.E.C. Field Service Engineers for VAX-11/780 System problems.

This outline is not intended to tell you what module to replace, but instead, is meant to lead you in the right direction. It is assumed that you are familiar with at least the following:

1. VAX-11/780 Processor
    a. Understand CONSOL.SYS command language.
    b. Know Physical and Electrical Configurations.
    c. Know HEX.
    d. Can examine/deposit Memory,I/O Regs.,and ID Regs.
2. VMS Booting
    a. Know how to boot.
    b. Have a basic understanding how boot is done.
3. Basic use of VMS such as:
    a. Able to login.
    b. Able to run "SYE" or "SPEAR".
    c. Able to use an editor.
4. Know how to run all VAX-11/780 Diagnostics.

Often times reference is made to "DUMP., HANG., & SDA.COM" command files within this outline. These files are files that I have written to do specific functions. You can use the files I have written or you can create similiar files yourself. I have also written several VMS DCL command files that are meant to aid D.E.C. Field Service in doing certain time-consuming functions.

The "DUMP. and HANG." command files are CONSOL.SYS command files that should be generated by D.E.C. Field Service and placed on the "LOCAL CONSOLE Floppy". The purpose of these two command files are as follows:

DUMP.

Is a command file that dumps all the Hardware Register contents to the Console Terminal. This command file is executed as an indirect command file from CONSOL.SYS. The purpose of this command file is to provide D.E.C. Field Service with additional trouble-shooting information concerning crashes that bring the software down and control is passed back to the CONSOL.SYS program.

HANG.

Is a command file that dumps all the Hardware Register contents, a few PC's during single step mode (to determine Hung loop), and then

II.

initiates the "CRASH." Local Console Floppy command file so that a Software Dump will be taken. The purpose of this command file is to provide D.E.C. Field Service with additional trouble-shooting information concerning system software hangs.

SDA.COM

The "SDA.COM" file is a VMS DCL command file that creates an output file that contains basic information taken from a specified Software Dump file. This file should be used, by you, when you are gathering information about a Software Dump to take back to your Support Group.

SAVEDUMP.COM

It is very inportant for the Customer to save the Software Dump file, "SYS$SYSTEM:SYSDUMP.DMP", every time the system is rebooted due to an Operating System crash. The easiest way to assure that this happens on every crash is to put the appropriate commands, to do the save, in the "SYS$SYSROOT:[SYSMGR]SYSTARTUP.COM" command file. I have generated a command file that the Customer can execute from the SYSTARTUP command file that will save the SYSDUMP.DMP file in the area that the Customer specifies. This command file, SAVEDUMP.COM, will name the saved file with a name that specifies the date and time of the reboot, after the crash. By using SAVEDUMP.COM, it is much easier to match Software Dumps to the appropriate crash.

This GUIDE references two handbooks extensively. These handbooks should always accompany you when you are working on a VAX-11/780 System. These handbooks are:

VAX Maintenance Handbook, VAX Systems	#EK-VAXV1-HB-???
VAX Maintenance Handbook, VAX-11/780	#EK-VAXV2-HB-???

Any suggestions as to how to improve this manual will be appreciated.

Roy D. Fulton
D.E.C. Field Service

SECTION Description
****

SECTION I of this manual is the actual "VAX-11/780 Trouble-Shooting" Outline. This section should be used as a guideline as to how to attack VAX-11/780 System problems.

SECTION II of this manual contains information about the VMS Operating System, the Command files used to boot VMS, SYSGEN commands, Unibus autoconfiguration requirements, etc.

SECTION III of this manual contains information concerning special command files and special programs.

SECTION IV of this manual contains information on VAX Architecture that may be needed as a reference while trouble-shooting.

SECTION V of this manual contains information about the different buses used on the VAX-11/780 systems.

SECTION VI of this manual contains information about the UNIX Operating System errors.

SECTION VII of this manual contains miscellaneous tidbits of information.

SECTION VIII of this manual contains the defintions of the bits in the NEXUS registers. The definitions for the CPU's registers are contained in the VAX Maintenance Handbook for the VAX-11/780. This section was copied from various VAX-11/780 Nexus Hardware manuals and microfiche.

TROUBLE-SHOOTING APPROACH

Trouble-Shooting Approach
****

The following pages are an Outline as to some of the things that you should do for certain types of VAX-11/780 problems. It is assumed that you will use a sound trouble-shooting approach to fixing the problem. The following Trouble-Shooting Method is a proven approach that can be tailored to every situation.

The correct steps to take in trouble-shooting are:

1. RESEARCH and DEFINE the PROBLEM.

   The problem should be diagnosed to a certain type of problem that happens under certain types of conditions. This must be done so that you will be able to recognize the problem on the next failure even though it may not exhibit exactly the same symptoms on the next failure.

   The Definition of the Problem does not necessarily identify the failing unit or subsystem, but simply describes the problem symptoms.

   Do not proceed to the next step until this is accomplished.

   Enter all error symptoms in the Log book.

2. VENTURE a Testable EDUCATED GUESS as to the PROBLEM AREA.

   From the information examined in step #1, make an educated guess as to where within the VAX-11/780 SYSTEM the problem lies. In other words, what subsystem or unit do you believe the failure to be in based on past experience, training, and the failure data information examined.

   If you are not able to make an educated guess at this time, it may be necessary to either wait for another failure in order to obtain more information, and/or you may need to ask your sources for aid in diagnosis.

If unable to do this step, be sure that error information catching facilities are in place, wait for another failure, and then go back to step #1. The error catching facilities you may want to implement for the VAX-11/780 SYSTEM may be as follows:

a. Set the SYSGEN parameter "BUGREBOOT" to a "0", so that a Hardware Register Dump may be taken at failure time.

b. Set the SYSGEN parameter "DUMPBUG" to a "1", so that the Software Dump will be taken.

c. Set the SYSGEN parameter "BUGCHECKFATAL" to a "1", so that NON-FATAL Bugchecks will be treated as Fatal Bugchecks. You probably don't want to do this without also setting "BUGREBOOT" to a "0".

d. Education of customer as to how to dump the Hardware Registers.

e. Making sure that the Customer's SYSTARTUP.COM file saves the Software Dumps or at least make sure that the Customer has Software Dump Saving procedures in place.

f. An Error Log report should be taken, and available on Hardcopy, of the time prior to and at time of the failure(s).

Be sure you enter into the Log Book what your evaluation is and anything else you may have done.

3. SET-UP an TEST CASE in order to isolate the PROBLEM.

Using your knowledge of the system, past experience, and your Support resources (if you need them), make a decision as to what area of Hardware or Software should be replaced or swapped first. This replacement or swapping should be done in a educated manner.

DO NOT swap or replace parts within the believed problem area in a haphazard manner. You should be able to pick out the most suspectable area.

Once you have decided what parts to replace or swap, MARK in a CLEAR easily defined METHOD each part that will be used in the test case in such a way that you and your counterparts will readily be able to determine the following:

a. The ORIGINAL SOURCE of EACH UNIT involved in the test case swap or replacement.

b. The DATE and TIME of the SWAP or REPLACEMENT of EACH UNIT involved in the test case.

This may be accomplished by either tagging the appropriate units or by marking each unit with a different marking and then entering the appropriate information in the SYSTEM's LOG Book by referencing these markings. This is VERY IMPORTANT.

4. PREDICT the RESULTS of the TEST CASE.

Make an educated prediction as to what the results of the test case will be. In other words, if you swapped a couple of units within the SYSTEM or DEVICE, what type of failure do you suspect will happen if the expected failing unit does indeed fail again.

It is very important that this information be recorded in the SYSTEM's LOG Book, so that you and your counterparts will know what to suspect upon the next failure.

5. CONDUCT the TEST CASE.

Perform the appropriate changes in order to conduct the test case as planned. Be sure to log everything in the SYSTEM's LOG Book.

6. EVALUATE the DEFINITION of the PROBLEM upon the NEXT FAILURE.

Now that you have more information to work with, does this failure still fit under the first definition of the problem?

If it doesn't, then proceed according to the following:

a. Is there more than one problem? If there is more than one problem, each should be researched, defined, tested, etc., separately.

   Be sure to label all failure information so that you will know what information goes with what failure.

b. Has another problem been introduced as a result of units used in the test case? One of the units that you inserted into the SYSTEM may have gone bad. If so, then you will have to evaluate whether to insert another new unit and wait for another failure or should you just replace the original and conduct another test case.

If it does fit under the same definition, continue to step #7.

7. RESEARCH and REFINE the DEFINITION of the PROBLEM.

After each failure, it may be necessary to redefine the problem or to refine the definition of the problem due to the contents of the problem's dumps. Refine the problems definition at this point based on past experience, knowledge of the failing unit, input from your Support resources, and the added problem failure information taken at the last failure.

In other words, you may be able to give a better definition to the problem, at this point in time, that will make it easier to determine where the problem lies and easier to determine when it is fixed. Be sure to enter this information in the SYSTEM's LOG Book.

Be sure to enter into the Log Book exactly how each failure occurs and exhibits itself.

8. RETURN NON-FAILING UNITs to their ORIGINAL POSITIONs.

It is very important to return the non-failing units, moved as a result of the test case, to their original positions as soon as the test case is completed.

This should be recorded in the SYSTEM's LOG Book in order to prevent confusion in the future.

This step is probably the most often ignored step even though it is one of the most important steps.

9. REPLACE FAILING UNITs with SPAREs.

Replace the failing units with spares. Since step #8 was done, the new unit (a spare) should be going into the ORIGINAL position of the failing unit.

This information should be recorded in the SYSTEM LOG Book, and an entry should be made on the appropriate DEVICE's LOG Sheet.

10. REPEAT UNTIL the PROBLEM is SOLVED.

Repeat steps 2 thru 9 until the problem is solved.

When the Problem is declared solved should be a predetermined period of time after the last failure. This time must be mutually agreeable between D.E.C. Field Service and the Customer. The determination of how long the system must run, without the defined problem happening, should be primarily based upon two things. They are:

a. The T.B.F. (time-between-failures).

b. The minimum run time that the customer would feel comfortable with.

The elapsed time period before the problem is declared solved should be no less than twice the longest time between failures, and should be equal to or greater that the customer required time.

SYSTEM LOG BOOKS

Log Book maintenance is a very important part of SYSTEM trouble-shooting. The keeping of a Log book is not just the Site Representative's duty but is the duty of every person that goes onsite to fix any problem. Every time you are onsite, anywhere, you should not consider the call complete until the System's LOG book has been filled out giving a detailed description of everything that has expired during your visit.

When properly used, a System Log Book will:

1. Stop UNNEEDED CONFUSION about the status of the SYSTEM, what was replaced when, what is expected to happen, what to do next if a certain event happens, if a certain event doesn't reoccur, etc.

2. Provide SYSTEM History information.

3. Provide DEVICE History information.

4. Provide updated SYSTEM Configuration information.

5. Provide an Intermittent Problem Action Plan.

6. Provide SYSTEM and DEVICE PM Status.

7. Provide SYSTEM and DEVICE diagnostic Run sheets.

8. Provide a means of passing information from one D.E.C. Field Service Engineer to another.

9. Provide a means of obtaining SYSTEM uptime information.

10. Provide specific SITE Dependent information such as where the Diagnostics are kept, where the Prints are kept, what test to run, special security considerations, specific site dependent information, etc.

Every little tidbit of information about a problem should be entered into the Log Book. These tidbits may seem unimportant to you now, but may become valuable bits of information later on.

It is your duty to help maintain an accurate log book for each system that you work on, every time you are on site.

S O U R C E S of I N F O R M A T I O N
****

At times you may need some help in Problem Diagnosis, Repair, ECO information, Diagnostic information, and etc. A list of resources that you can use is listed below:

1. Your fellow workers in your Branch.
   This is an important resource. If you know someone in your group that probably knows the answer to your question(s), don't be afraid to ask for their help. Working together in this way also helps to build morale within a group.

2. Your Remote Support and local Support Groups.

3. The Remote Diagnosis Center, (RDC),in Colorado.

   RDC can :
   - Run Diagnostics.
   - Examine VMS Dumps.
   - Answer ECO problems.
   - Look up information in the Library.
   - Answer functional questions.
   - Run/Monitor extended testing.

   Be aware that the RDC now has a library of all kinds of information. When you call RDC, they will ask you your SYSTEM TYPE, at this point ask for the LIBRARY.

4. If you know ANYONE in D.E.C. that probably knows the answer to your question, feel free to call and ask. We, all D.E.C. employees, owe our jobs to our Customers. Therefore, there is no reason why anyone in D.E.C. should refuse to answer a question for you if they know the answer.

MAINTAINING CONTROL
****

This is probably the most misunderstood and abused concept of trouble-shooting basics, but it is of the utmost importance that the Field Service Engineer MAINTAINS CONTROL of the SITUATION at ALL times. This manual will not do you any good if you do not have the control needed to perform the steps specified. In order to fix Customer Problems quickly and efficiently, you must:

1. Be able to MAINTAIN CONTROL at all times.
2. Have good CUSTOMER RELATION SKILLS.
3. Have a sound TROUBLE-SHOOTING APPROACH.
4. Have the ABILITY TO BE SYMPATHATIC towards the CUSTOMER'S BUSINESS NEEDS.
5. Have KNOWLEDGE of the hardware and software.

What is meant by Maintaining Control as related to trouble-shooting?

Maintaining Control simply means that you, a D.E.C. Field Service Engineer that is attempting to fix a Customer's problem, must at all times approach the problem with you in command of the situation. This simply means that you make the decisions as to what to do, when to do it, and how to do it, while carefully considering the Customer's business needs (all your decisions must remain within the Problem Manager's guidelines). You must maintain control while also making sure that your decisions will impact the Customer's business as little as possible.

Loosing control, more often than not, causes longer overall downtime for the Customer and also causes you to start to loose confidence in your ability to do your job properly. Your job is to fix the Customer's problem(s) as soon as possible while affecting his/her business as little as possible. If you do not Maintain Control of the situation, you obviously cannot perform your job properly.

If at any time you feel that you are starting to loose control, immediately contact your management and get them involved. DO NOT allow yourself to loose complete control before contacting your management. Everyone needs help occassionally. Don't let your pride prevent you from doing what is best for the Customer and D.E.C..

The amount your decisions impact the Customer's business needs must always be considered carefully. Sometimes it makes more sense to take the System for an extended period of time, in order to reduce the total overall time spent repairing the Customer's problem.

Maintaining Control does not mean that you make your decisions without the Customer's input. One of the first steps you should always do when trouble-shooting a problem, is to gather as much information about the problem as possible. The first source of information about the problem comes from the Customer. The Customer may even have an idea about what is causing the problem. You should listen to everything that the Customer has to say. This does not mean that you base your trouble-shooting totally on the information received from the Customer. Research the problem thoroughly before jumping in and replacing things. You must also use input from the Customer when you are weighing what you want to do with how it will affect the Customer's business.

It is of the utmost importance not to abuse your control. Abuse of control can only result in a poor D.E.C./Customer relationship. You must maintain good D.E.C./Customer relations and supply the Customer with the best service possible within the guidelines of the Customer's Contract.

SUMMARY

Your goal is to fix the Customer's problem, in the shortest length of time, while maintaining complete control of the situation, and while constantly evaluating your decisions with respect to the Customer's needs, concerns, and Contract Coverage.

Keep in mind the following:

1. Maintain Control of the situation.

2. Always weigh your decisions with respect to the following:

    a. Customer's Business needs and concerns.
    b. D.E.C.'s Contract Obligations to the Customer.
    c. Is this the quickest approach to fixing the problem?

3. Maintain good D.E.C./Customer relations.

4. Always be CONSIDERATE, TRUTHFUL, and FAIR.

SECTION I

VAX-11/780 Trouble-Shooting Outline

VAX-11/780 Trouble-Shooting Basics

This outline is designed to aid you in isolating problems to either the Memory, the VAX-11/780 CPU, the VAX-11/780 Front-end Subsystem, a VAX-11/780 Nexus, a Peripheral Device or to Software. Peripheral Devices will only be covered in general while the VAX CPU, the MEMORY, and the NEXUSes will be covered more thoroughly.

1. An UNDEFINED PROBLEM exists. The problem is undefined until "YOU" make an educated guess as to where the problem probably lies.

2. GATHER all INFORMATION, from the Customer, that is available about the problem and its symptoms. At this point in time, you are not evaluating the problem but are merely gathering information that you can evaluate later. Keep an open mind, don't let any tidbit of information pass you by. Many problems could have been solved sooner if the Field Engineer had remembered seemingly insignificant tidbits of information that may appear unrelated to the problem. Be sure to record as much Symptom information as possible in the "LOG Book".

   a. How is the problem exhibited?

      1. This is found by talking to the Customer. Be sure to record this information in the "LOG Book".

      2. Is the problem intermittent or a solid problem?

      3. Can the problem be recreated at will?

      4. What is the MTBF (mean time between failures)?

      5. Does the problem seem to be related to only one or only a group of functions or programs?

      6. If it is a program that causes the problem, is this a customer program or a D.E.C. supported program?

      7. Is there anything common about the problem in either software or hardware?

      8. Did any System Environmental changes take place previous to or at time of the problem?

      9. Does the problem appear to be, or could the problem be, media related?

b. What is the customer's evaluation of the problem?

   1. Does the customer believe the problem to be in a certain area of the hardware or software? Be sure to record this information in the "LOG Book".

c. Is there a hard copy printout showing the problem symptom? If the Operating System crashed or hung, what you want is the Console Terminal output at the time of the failure.

   1. If a "Hard Copy Printout" is available, ask the customer for it.

d. Was a "Hardware Register Dump" taken at failure time?

   1. If the software crashed or hung, the customer should have taken a "Hardware Register Dump" (by using the "DUMP. or HANG." command file on the "LOCAL CONSOLE" floppy) immediately at time of failure. Ask for this dump.

e. Was a "Software Dump" taken and saved?

   1. If the software crashed a software dump should have occurred automatically as a result of the Operating System executing its crash routine (if SYSGEN parameter DUMPBUG=1). The dump should have been saved by the Customer when the system was rebooted. Ask the Customer for the "DDCU:[DIRECTORY]FILENAME.EXT" of the "SAVED SOFTWARE DUMP".

3. Get an ERROR LOG REPORT if possible at time of and prior to failure. If the problem is of the type that allows the Operating System to run, or is a problem that intermittently crashes the Operating System, attempt to get an "ERROR LOG" report by running either "SPEAR" or "SYE". Have the output go to a file, and then print that file. If you are running SPEAR, use the "SPEAR Analyze" function to analyze the errors prior to the crash. It may be necessary to do a full retrieve of all information. In some cases it may not be possible to do Error Log reporting at this time. In those cases, it is wise to run Error Log reporting as soon as the system is functional enough to do so. Operating System Error Logging is a great aid to trouble-shooting, use it whenever possible. The VAX Error logging utility is very good.

4. IDENTIFY the TYPE of PROBLEM if possible. Now that you have gathered as much information as you can about the problem, find a desk or table somewhere where you can sit down and attempt to "Isolate the Problem". The first step in problem isolation, is to determine the "Type of Problem".

VAX-11/780 Problems can be broken down into several basic types:

1. Operating System Crashes or Bugchecks.
   (This includes "Machine Checks" & "CPU DBLE-ERR HLT's")
2. Operating System Hangs.
3. Operating System Functional Problems.
4. Operating System Backup Problems.
5. Booting Problems.
6. Front-end Subsystem goes back to ODT, Hangs, Halts.
7. Unexplained Reboots or Power Restarts.
8. Problems on a Certain Device or Devices.
9. System or Peripheral Device won't Power-Up.
10. Something's Burning.
11. Problems Building the VMS Operating System.
12. Error Bits set in a NEXUS (DW780, RH780, etc.)
13. S.B.I. FAULTS

Determine which of the above types the problem fits under and then go to the outline for that problem type. Every problem should fit under one of these problem types.

VAX-11/780 System Trouble-Shooting tools:

Visual and Sensual Indications

a. LSI-11 Front panel indicators - DCON, RUN
b. VAX-11/780 Control panel indicators - ATTN, RUN, POWER
c. H7100 status indicators
   Power Normal, Regulator Failure, Overtemp,
   Overcurrent, & Power Inverter Failure
d. Smoke, fire, heat, burning smell

Registers

a. The CPU ID <00:3E> registers
   E/ID/L/N:3E 0
b. The Control/Status registers in each nexus
   E/L/P/N:x 200xx000 - for each nexus
c. The Control/Status registers in each UNIBUS device
   E/L/P/N:x 201xxxxx - for each UNIBUS device
d. The Control/Status registers in each MASSBUS device
   E/L/P/N:x 200xx400 - for each MASSBUS drive

Console Terminal Messages

a. LSI-11 ODT error messages
   xxxxxx <- PC at time of LSI macro program halt
   @ <- LSI ODT prompt
b. Error messages from CONSOL.SYS
   ? xxxxxxxxxxxxxxx
   >>>
c. Error messages from the OPERATING SYSTEM (VMS/UNIX)
   FATAL BUGCHECK <- From VMS. Is followed by
   a General register dump, Stack
   dump and a description of error.
d. Error messages from other programs

User Terminal Messages

a. Error messages from the OPERATING SYSTEM (VMS/UNIX)
b. Error messages from other programs

Error Log Information

a. "ERRLOG.SYS" for the VMS operating system
   in SYS$ERRORLOG:ERRLOG.SYS
b. "/messages" file for the UNIX operating system

System Manager/User Input

a. System Manager/User definition of the problem
b. System Manager/User feelings of problem area
c. Any customer known or initiated changes to system

System History

a. Past failure/repair history of the system
b. ECO status of the system
c. System configuration changes
d. Software changes
e. PM history
f. Enviromental Changes

VMS Operating System Crashes or Bugchecks

Due to the nature of these types of problems and the Software decisions that are made, the System may not be down when you arrive onsite. For this reason, these flows will not specify when to run diagnostics. If the Operating System is operative when you arrive onsite, it is probably best to gather the listed information about the crash and attempt to at least make a preliminary diagnosis before taking the system to run diagnostics. This preliminary diagnosis should point you to an area or subsystem on which to start running diagnostics.

In order to trouble-shoot these types of problems, it is necessary to gather as much information about the crash as possible. The amount of information that you will be able to gather will depend upon how the system is set up. Procedures to gather the following types of information should have been setup previous to the crash:

1. Doing a HARDWARE REGISTER DUMP.
2. SYSGEN parameter DUMPBUG set to a 1 so as to get a SOFTWARE dump.
3. Saving of SOFTWARE dumps on reboots.
4. Saving of Errlog (EVENT file) information.
5. Saving of the Console Terminal output (on hardcopy).
6. Accurate LOG BOOK information concerning all activity on the system.

"It is IMPOSSIBLE to gather TOO MUCH information about a problem."

In order to gather the Hardware Register Dumps, certain modifications to the LOCAL/REMOTE CONSOLE FLOPPY should be made. A "DUMP." and "HANG." command file should be created that is tailored to the associated system (see the procedures in Chapter 3 of this manual). It is possible to get all the Hardware Register information needed by using these two command files, but the customer must run the system with SYSGEN parameter BUGREBOOT = 0 and the AUTO-RESTART switch OFF. Then when the system crashes, you or the customer must take the dump by initiating either the "DUMP." or "HANG." CONSOL.SYS command file prior to rebooting of the system. This method is usually not desirable from the customer's standpoint of trying to have the system up as much as possible. A better method is to add the commands in the "DUMP." command file to the front of the "DEFBOO.CMD" & "RESTAR.CMD" command files on the LOCAL/REMOTE CONSOLE Floppy. This will allow the customer to run with AUTO-RESTART set ON and the SYSGEN parameter BUGREBOOT set to a 1, and the Hardware Register dump will automatically be taken prior to the system rebooting.

On these crashes, the Hardware Register Dump may not reflect the same ID register contents as were present at the actual time of the error due to VMS not halting immediately (several things are done prior to VMS halting, one of which is the writing of the Software Dump). The Hardware Register Dump may show some Device or Nexus errors though. It is always better to have extra information rather than not enough, so take the HARDWARE REGISTER DUMP whenever possible.

The ERRLOG.SYS file should be examined, whenever possible, to see if the Operating System was able to log any error information that may be causing the crashes. The VAX VMS "System Event File (ERRLOG.SYS)" is a very useful trouble-shooting tool that is very often overlooked. Many times the Fatal Bugchecks are caused by something that is logged in ERRLOG.SYS just prior to the crash.

There are many different types of "FATAL and/or NON-FATAL BUGCHECKS". BUGCHECKS can either be caused by Hardware errors or Software detected error conditions. The SOFTWARE detected errors "may not" be caused by any hardware failures.

Whether a BUGCHECK is declared "FATAL or NON-FATAL" depends upon what "MODE" the processors is in when the error occured. Basically, a "Non-Fatal Bugcheck" is a Bugcheck that has occured while the processor is in either the "USER" or "SUPERVISOR" mode. A "Fatal Bugcheck" is one that has occured while the processor is in either the "EXEC" or "KERNEL" mode. Chapter 2, of this manual, describes Fatal and Non-Fatal Bugcheck action.

The easiest of the Bugchecks to trouble-shoot is the MACHINE CHECKS. This is due to a specific logout procedure that the VAX-11/780 CPU microcode goes through to insure that you have the needed error information stored on the stack. Fatal Machine Check Bugchecks will print out the stack information on the console terminal. This is usually all the information that you need to trouble-shoot this type of bugcheck. Non-Fatal Machine Check Bugchecks will cause VMS to store the stack information in the system event file (ERRLOG.SYS). All of the other types of BUGCHECKS require a software knowledge to affectively trouble-shoot them since these errors usually are not the result of some hardware detected error but due to softwares detection of a problem. Therefore, you would probably have to know what the system software was attempting to do in order to effectively trouble-shoot them.

* FATAL BUGCHECK, VERSION-V3.1 MACHINECHK, Machine check while in kernel mode *

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

M A C H I N E C H E C K s

?? MACHINE CHECK EXCEPTION THROUGH VECTOR: 04(X)

VAX-11/780 MACHINE CHECK Error Logout

A Machine Check Exception indicates that the processor detected an INTERNAL ERROR in itself, an SBI TIMEOUT, or an SBI ERROR CONFIRMATION is received when the processor was attempted an SBI transfer. Software decides, on the basis of the logout information presented, whether to abort the current process or simply to continue.

The following steps show the basics of what happens when the VAX-11/780 CPU detects an error.

1. The VAX-11/780 CPU detects an error.

   The types of errors that can cause a Machine Check are:

   a. Control Store Parity Error
   b. Cache Parity Error
   c. Translation Buffer Parity Error
   d. Read Data Substitute Error
   e. SBI Read Timeout
   f. SBI Error Confirmation received
   g. VAX CPU Micro-code goes to unused Micro-code location

2. The VAX-11/780 Micro-code branches to a "Error Snapshot" micro-code routine that performs the saving of the "Machine Check Logout" information onto a specified Stack.

   MicroPC Error entry points for PCS (version 1.0) microcode

uPC	010F	-	Control Store Parity Error
uPC	0107	-	Translation Buffer Parity Error
uPC	0108	-	Cache Parity Error
uPC	010C	-	Read Data Substitute
uPC	010D	-	S.B.I. Read Timeout or Error Confirmation
uPC	0EE0	-	Microseqencer Error

3. The VAX-11/780 Micro-code saves certain CPU registers in T0 thru T9 (ID #30 thru 39) for temporary storage. These registers, along with the PSL, PC and a byte count, make up the "Machine Check Logout" information. The Registers that are saved are:

   a. The CPU Error Status Register (CES = ID #0C)
   b. The Trapped UPC Register (USTACK = ID #20)
   c. The VA/VIBA (from the VA/VAMX multiplexers)
   d. The D-Register (DQ = ID #08)
   e. The TB Error Register #0 (TB ERR #0 = ID #12)
   f. The TB Error Register #1 (TB ERR #1 = ID #13)
   g. The Timeout Address Register (TIME.ADR = ID #1A)
   h. The Cache Parity Error Register (PARITY = ID #1E)
   i. The SBI Error Register (SBI.ERR = ID #19)
   j. The D.SV Register (D.SV = ID #2E)

This data is first stored in T0 thru T9, in the following order, on execution of the micro-words at the specified PCS (version 1.0) micro-addresses:

uPC	Register Name	ID No.	Saved in
0EF1	CPU Error Status	0C	T1 - ID#31
0EF2	Trapped Micro-PC	20	T2 - ID#32
0EF4	VA/VIBA		T3 - ID#33
0EF5	D Register		T4 - ID#34
0EF8	TB Error Register 0	12	T5 - ID#35
0EFB	TB Error Register 1	13	T6 - ID#36
0EFC	Timeout Address	1A	T7 - ID#37
0F01	Cache Parity Register	1E	T8 - ID#38
0F03	S.B.I. Error Register	19	T9 - ID#39
0F06	Summary Parameter		T0 - ID#30

Then this MACHINE CHECK logout information is stored onto the stack, along with the PC and PSL.

This two step procedure is used in order to preserve 1st error information in the case of another error occuring while the micro-code is attempting to logout the 1st error's information onto the stack.

PCS (Version 1.0) micro-code entry points

uPC 0EE9 - "Error Snapshot micro-routine" that stores certain registers in the T0-T9 temporary registers.

uPC 0F10 - "Error Snapshot micro-routine" that stores the Temporary registers (T0:9) onto the stack along with the PC, PSL, and a Byte Count longword equal to a hex 00000028.

4. The VAX-11/780 Micro-code gets the SCBB data to be used to find the physical address of the "MACHINE CHECK VECTOR". The SCBB is a register (ID #3B) that contains the starting address of the System Control Block (SCB).

The System Control Block is the physical page in memory that contains the Exception and Interrupt Vectors.

5. Bits <1:0> of the data in "SCBB+4" are checked, by the Micro-code, to determine what Stack to use for the Machine Check Logout information storage, or to see if the CPU should halt.

   The Kernel Stack or the Interrupt Stack can be used as the storage place for the Machine Check Logout information.

6. The VAX-11/780 Micro-code saves the Machine Check Logout information onto the Stack specified by "SCBB+4" bits <1:0>, or will halt if these bits are both 1's (<1:0>=3). If the VAX CPU halts, control will pass back to the CONSOL.SYS program.

   The Machine Check Logout information consists of the Saved Register contents that were saved in Temporary Storage registers ID #30-39, the PC and PSL at the time of the error, and a byte count that specifies the total number of bytes dumped on the stack (which is "always" a hexadecimal 28 or decimal 40).

7. The VAX-11/780 Micro-code causes the Instruction_Set_Processor to jump to the routine whose Longword address is in "SCBB+4" if "SCBB+4 Bits <1:0>" is not equal to 3. If "SCBB+4 Bits <1:0>=3" then the VAX Instruction_Set_Processor is halted.

8. If "SCBB+4 Bits <1:0>" are not equal to 3, the VAX-11/780 Instruction_Set_Processor runs VAX Macro instructions to perform the specified trap routine.

   What is done in this routine depends totally on the running Macro-code program. When a Machine Check occurs while running the VMS Operating System, the Macro Routine that is run now will determine whether the Machine Check is FATAL or NON-FATAL and will act accordingly. When a Machine Check occurs while running the Diagnostic Supervisor, the Macro Routine that is run now will normally print out the STACK contents and then go back to the "DS>" prompt and await operator input. Other software may simply halt when the Machine Check occurs. Still other software may not properly set up a "SYSTEM CONTROL BLOCK" and all sorts of strange things may happen.

NOTE: If another Machine Check condition occurs while the microcode is performing the functions in steps 2 thru 6, the VAX microcode flags a "Double Error Halt" and turns control over to the LSI Subsystem's CONSOL.SYS program.

VMS Fatal Bugcheck printout example caused by a Machine Check

```
*** FATAL BUGCHECK, VERSION-V3.1 MACHINECHK, Machine check while in kernel mode

    CURRENT PROCESS = STARTUP
    REGISTER DUMP

        R0 = 00000206
        R1 = 00000028
        R2 = 0000792E
        R3 = 00000000
        R4 = 0000792E
        R5 = 0000021E
        R6 = 80027600
        R7 = 00002768
        R8 = 80137300
        R9 = 0000C768
        R10= 000015F8
        R11= 80137300
        AP = 7FFB4888
        FP = 7FFEADA8
        SP = 80154FC4
        PC = 800CF411
        PSL= 041F0008

  KERNEL/INTERRUPT STACK
        80154FCC        00000028
        80154FD0        00000000
        80154FD4        00010084
        80154FD8        00000224
        80154FDC        80027A18
        80154FE0        03FB5E6C
        80154FE4        00007E81
        80154FE8        00000040
        80154FEC        28005106
        80154FF0        00004000
        80154FF4        00009402
        80154FF8        00002C4D
        80154FFC        00DF0000

        HALT INST EXECUTED
        HALTED AT 800039ED

        (BOOTING)
        CPU HALTED
        INIT SEQ DONE
        HALT INST EXECUTED
        HALTED AT 200034F9

        G  0000000E  00000200
        LOAD DONE, 00004200 BYTES LOADED

  VAX/VMS Version V3.1 11-AUG-1982 16:21
```

Example of a Machine Check while running ESSAA

```
DS> RUN EVRAA

.. Program: VAX DISK AND TU58 RELIABILITY TESTS *EVRAA*, revision 12.0,
6 tests, at 07:19:26.94.

Testing: _DRB1

?? MACHINE CHECK EXCEPTION THROUGH VECTOR: 04(X)
CP READ TIMEOUT/SBI ERROR CONFIRMATION FAULT

MACHINE CHECK LOGOUT:
COUNT:                  00000028(X)
SUMMARY PARAMETER:      00000000(X)
CPU ERROR STATUS:       00010084(X)     ;NESTED ERROR, ALU C31

TRAPPED MICRO PC:       00000248(X)
VA/VIBA:                60014008(X)
D REGISTER:             0504A056(X)
TBER0:                  00007E00(X)     ;
                                        ;LAST REFERENCE = ADS,MCTE,MCT2
                                        MCT1,MCT0,IBWCHK
                                        ;FORCE TB PARITY ERROR=NO ERROR
                                        FORCED
TBER1:                  00000040(X)     ;LAST TB WRP
TIME.ADDR:              28005002(X)     ;MODE=KERNEL,PROT CHECK, SBI AD
                                        DR=20014008(X)
PARITY:                 00004000(X)     ;CP ERROR
SBI.ERR:                00009402(X)     ;RDS INT EN,CP TO, CP TO ST0,
                                        NOT BUSY
PC at error:            00051649(X)
PSL at error:           001F0000(X)     ;CUR=KERNEL,PRV=KERNEL,IPL=1F
User return PC:         0001A40D(X)
DS>
```

Breakdown of a VMS Machine Check printout example.

```
*** FATAL BUGCHECK, VERSION-V3.1 MACHINECHK, Machine check while in kernel mode
                                                          ^
    CURRENT PROCESS = STARTUP                             |
    REGISTER DUMP                                Specifies type of BUGCHECK.  This
                                                 breakdown defines the meaning of this
        R0 = 00000206  <-|                       printout when the BUGHECK type is a
        R1 = 00000028    |                       "Machine Check".
        R2 = 0000792E    |
        R3 = 00000000    |
        R4 = 0000792E    |
        R5 = 0000021E    |
        R6 = 80027600    |
        R7 = 00002768    |---  This register dump isn't of much use to us if
        R8 = 80137300    |     the BUGCHECK is a "Machine Check".  Is useful
        R9 = 0000C768    |     for other types of BUGCHECKs.
        R10= 000015F8    |
        R11= 80137300    |
        AP = 7FFB4888    |
        FP = 7FFEADA8    |
        SP = 80154FC4    |
        PC = 800CF411    |
        PSL= 041F0008  <-|

                                                 The following definitions apply only
    KERNEL/INTERRUPT STACK                       when BUGCHECK is a MACHINE CHECK.
        80154FCC        00000028  <----- Byte Count
        80154FD0        00000000  <----- Summary Parameter
        80154FD4        00010084  <----- CPU Error Status (ID#0C)
        80154FD8        00000224  <----- Trapped micro-PC (ID#20)
        80154FDC        80027A18  <----- VA/VIBA
        80154FE0        03FB5E6C  <----- D Register (ID#08)
        80154FE4        00007E81  <----- TB Err. Reg. #0 (ID#12)
        80154FE8        00000040  <----- TB Err. Reg. #1 (ID#13)
        80154FEC        28005106  <----- Timeout Address (ID#1A)
        80154FF0        00004000  <----- Cache Parity (ID#1E)
        80154FF4        00009402  <----- SBI Err. Reg. (ID#19)
        80154FF8        00002C4D  <----- PC (General Reg. #F)
        80154FFC        00DF0000  <----- PSL (ID#0F)

        HALT INST EXECUTED
        HALTED AT 800039ED

        (BOOTING)  <---------------  SYSGEN "BUGREBOOT = 1", so rebooting
        CPU HALTED                   via "DEFBOO.CMD" command file.
        INIT SEQ DONE
        HALT INST EXECUTED
        HALTED AT 200034F9  <------ ISP Rom Macro program finished.  "SP"
                                    now contains SA+200 of good 64K chunk.
        G  0000000E  00000200
        LOAD DONE, 00004200 BYTES LOADED  <----- VMB.EXE loaded into VAX mem.

    VAX/VMS Version V3.1 11-AUG-1982 16:21  <----- VMS is loaded and started.
```

Getting Started on MACHINE CHECKs

The contents of the "STACK" are used to trouble-shoot "Machine Checks". The Stack Contents are printed out on the "Console Terminal", by the VMS Operating System, immediately after a "FATAL" Machine Check. "Fatal Machine Checks" are basically those Machine Checks that happened when the VMS Operating System was in "Kernel" or "Exec" mode. Machine Checks that happened during "User" or "Supervisor" mode are normally considered "Non-Fatal" and will result in the Machine Check Logout information (the contents of the STACK and the General Registers) being saved in the System Event file ("SYS$ERRORLOG:ERRLOG.SYS") versus being printed out on the "Console Terminal".

If the Machine Check occurred while running a program that does not output the "Logout" information to the "Console Terminal" or "SYSTEM EVENT File", then you will have to dump the Stack yourself. To do this, first examine (using CONSOL.SYS) ID #12 and check to see if Bit 00 is set. Then use the appropriate command to examine the STACK.

```
ID #12 - Bit <00> = 0    ; Memory Management not enabled.
-------------------------------------------------------
        >>> E/L/H SP
        >>> E/L/H/P/N:30 @


ID #12 - Bit <00> = 1    ; Memory Management is Enabled.
-------------------------------------------------------
        >>> E/L/H SP
        >>> E/L/H/V/N:30 @
```

Now, use the following steps to determine what caused the "Machine Check":

1. Find the "LONGWORD" on the Stack that contains a "00000028".

   If you cannot find this longword, then the Stack doesn't contain the "Machine Check Logout" information. You will, therefore, have to make sure the appropriate error catching facilities are in place, and then must wait for the next "Machine Check" to occur.

2. The "LONGWORD" following the "00000028", is the "Summary Parameter" word. Using "byte 0" of this longword, check the SUMMARY PARAMETER DESCRIPTION, on the next page, to determine what type of error occurred.

3. Now that you know what type of error occurred, go to the appropriate section of this Trouble-Shooting Guide to find out how to use the Stack "Machine Check Logout" information to further isolate the problem.

Normally the Exception Vector bits <1:0> define the following:

0 - Service the EXCEPTION on the KERNEL STACK unless running on the INTERRUPT STACK.
1 - Service the EXCEPTION on the INTERRUPT STACK.
2 - Service the EXCEPTION in WCS, Pass Bits <15:02> to Micro PC.
3 - Halt

The Machine Check Exception Vector is found at "SCBB+4". The "System Control Block Base" is a Physical address and can be found by examining "ID Register #3B" or "Internal Register #11".

MACHINE CHECK LOGOUT Information Description:

	Description	Memory Loc.	ID Loc.	Notes
1.	Byte Count	(SP)	None	Must be a 28 (hex).
2.	Summary Parameter	(SP+4)	T0 (30)	See below.
3.	CPU Error Status	(SP+8)	T1 (31)	See ID #0C.
4.	Trapped UPC	(SP+12)	T2 (32)	See ID #20.
5.	VA/VIBA	(SP+16)	T3 (33)	Virtual address.
6.	D Register	(SP+20)	T4 (34)	See ID #08.
7.	TB ERROR 0	(SP+24)	T5 (35)	See ID #12.
8.	TB ERROR 1	(SP+28)	T6 (36)	See ID #13.
9.	Timeout Address	(SP+32)	T7 (37)	See ID #1A.
10.	Parity	(SP+36)	T8 (38)	See ID #1E.
11.	SBI Error	(SP+40)	T9 (39)	See ID #19.
12.	PC	(SP+44)	None	General Reg. #F.
13.	PSL	(SP+48)	None	See ID #0F.

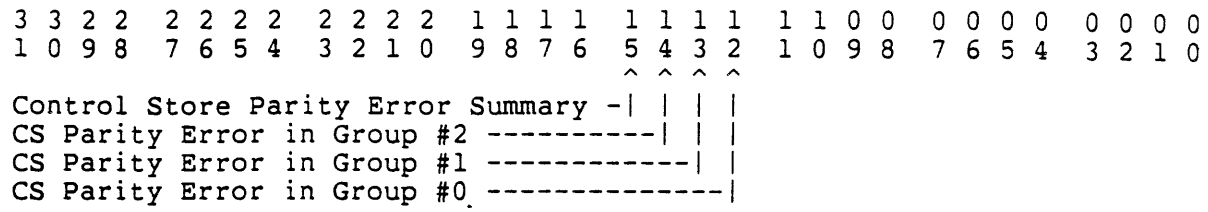
SUMMARY PARAMETER Description (use Byte #0, only!)

Page	Code	Description
1-55	00	CP Read Timeout or Error Confirmation Fault
1-38	02	CP Translation Buffer Parity Error Fault.
1-34	03	CP Cache Parity Error Fault.
1-92	05	CP Read Data Substitute Fault.
1-38	0A	IB Translation Buffer Parity Error Fault.
1-92	0C	IB Read Data Substitute Fault.
1-55	0D	IB Read Timeout or Error Confirmation Fault.
1-34	0F	IB Cache Parity Error Fault.
1-55	F0	CP Read Timeout or Error Confirmation Abort.
1-45	F1	Control Store Parity Error Abort.
1-38	F2	CP Translation Buffer Parity Error Abort.
1-34	F3	CP Cache Parity Error Abort.
1-92	F5	CP Read Data Substitute Abort.
1-94	F6	Microcode "not suppose to get here" Abort.

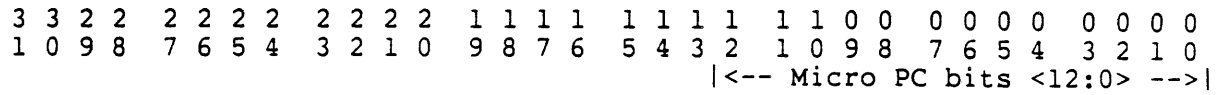
|
|-- Goto to this page to find out how to trouble-shoot associated error.

BIT BREAKDOWN OF STACK ENTRIES - *Showing error information*

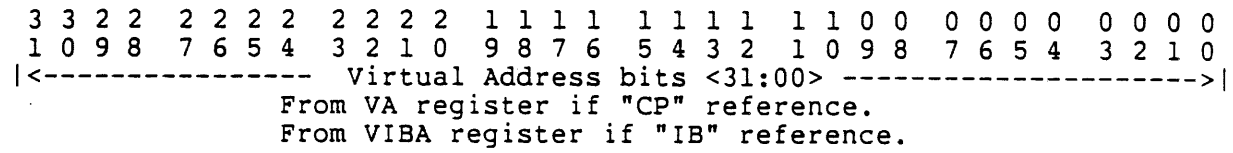
ID #0C - CES



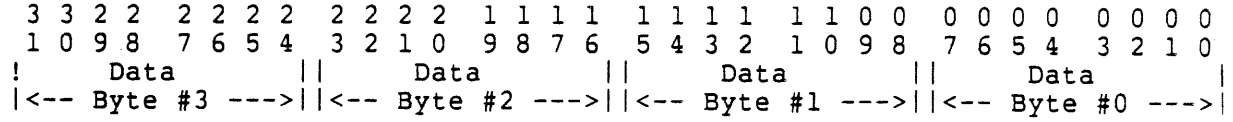
ID #20 - MICRO STACK



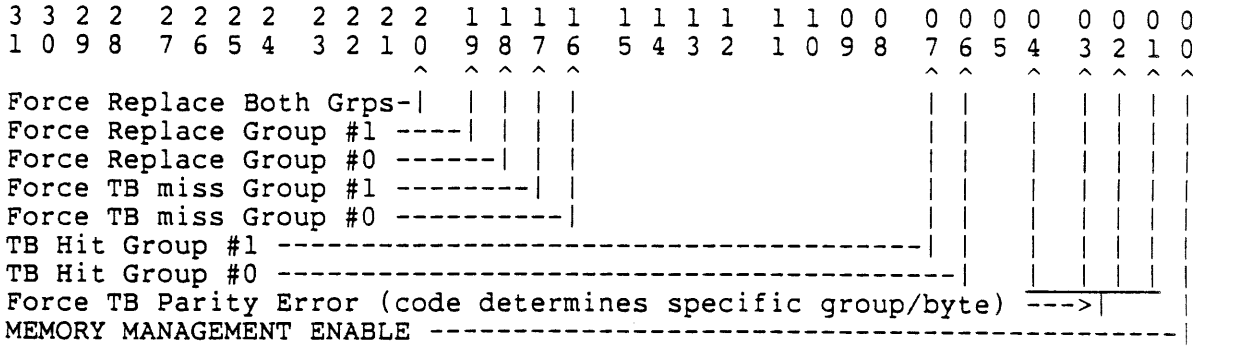
VA/VIBA - output of VAMX



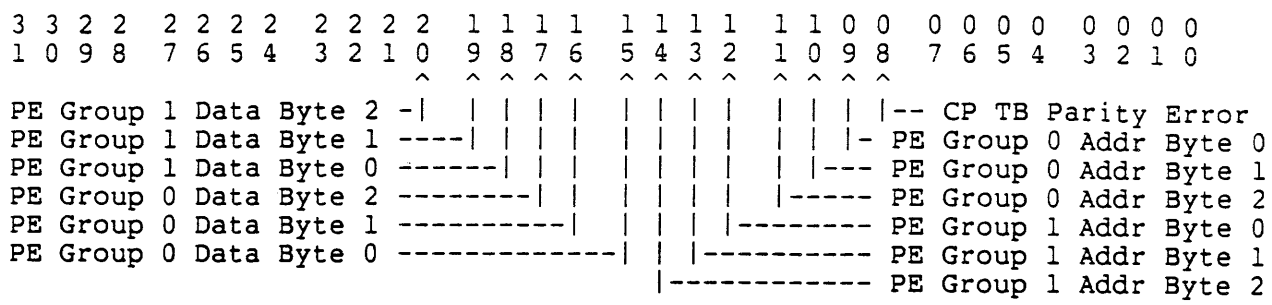
ID #08 - D Register



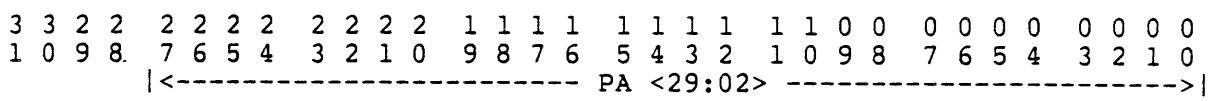
ID #12 - TBER0



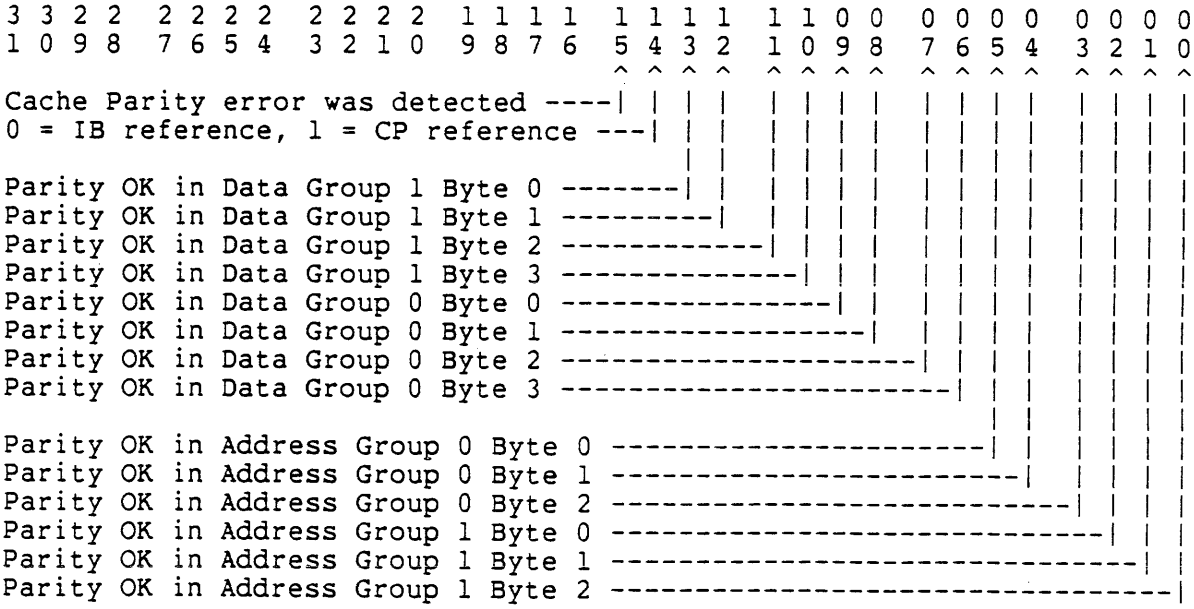
ID #13 - TBER1



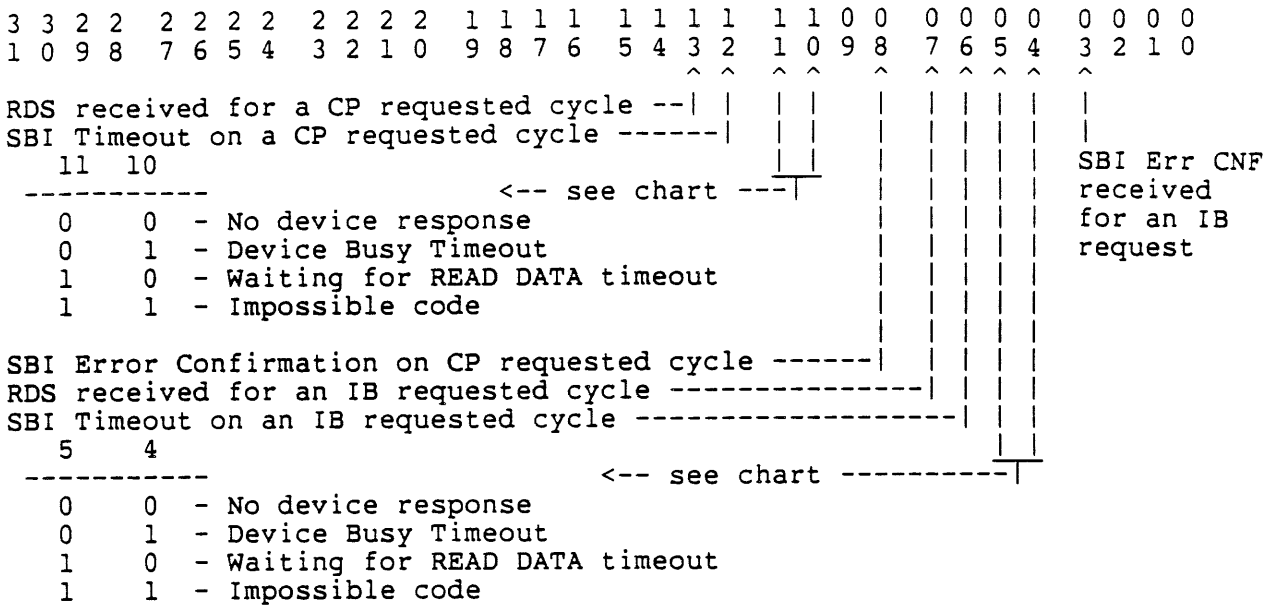
ID #1A - TIMEOUT ADDRESS



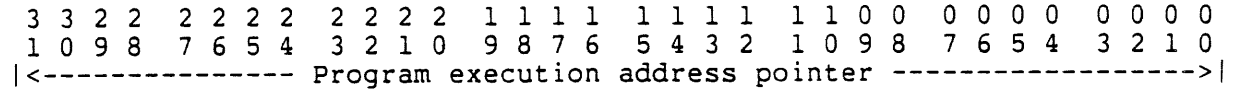
ID #1E - CACHE PARITY



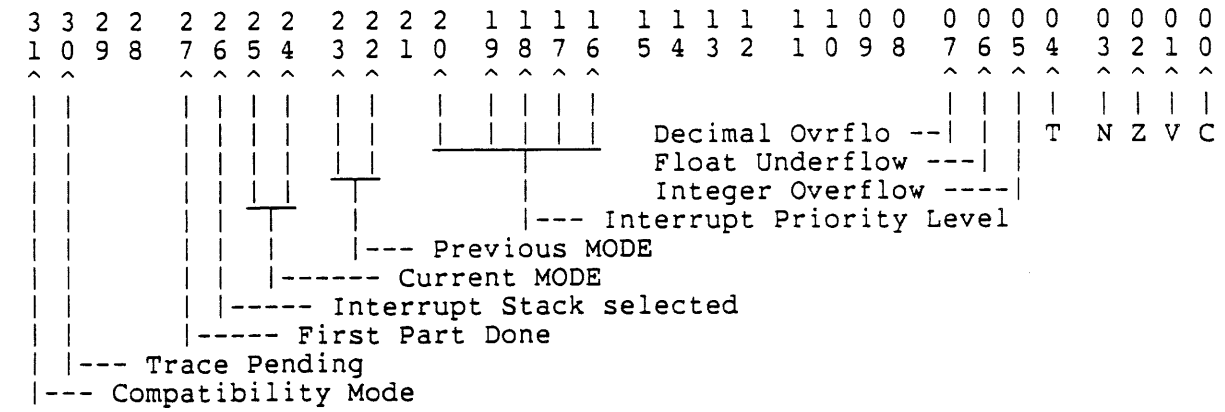
ID #19 - SBI.ERR

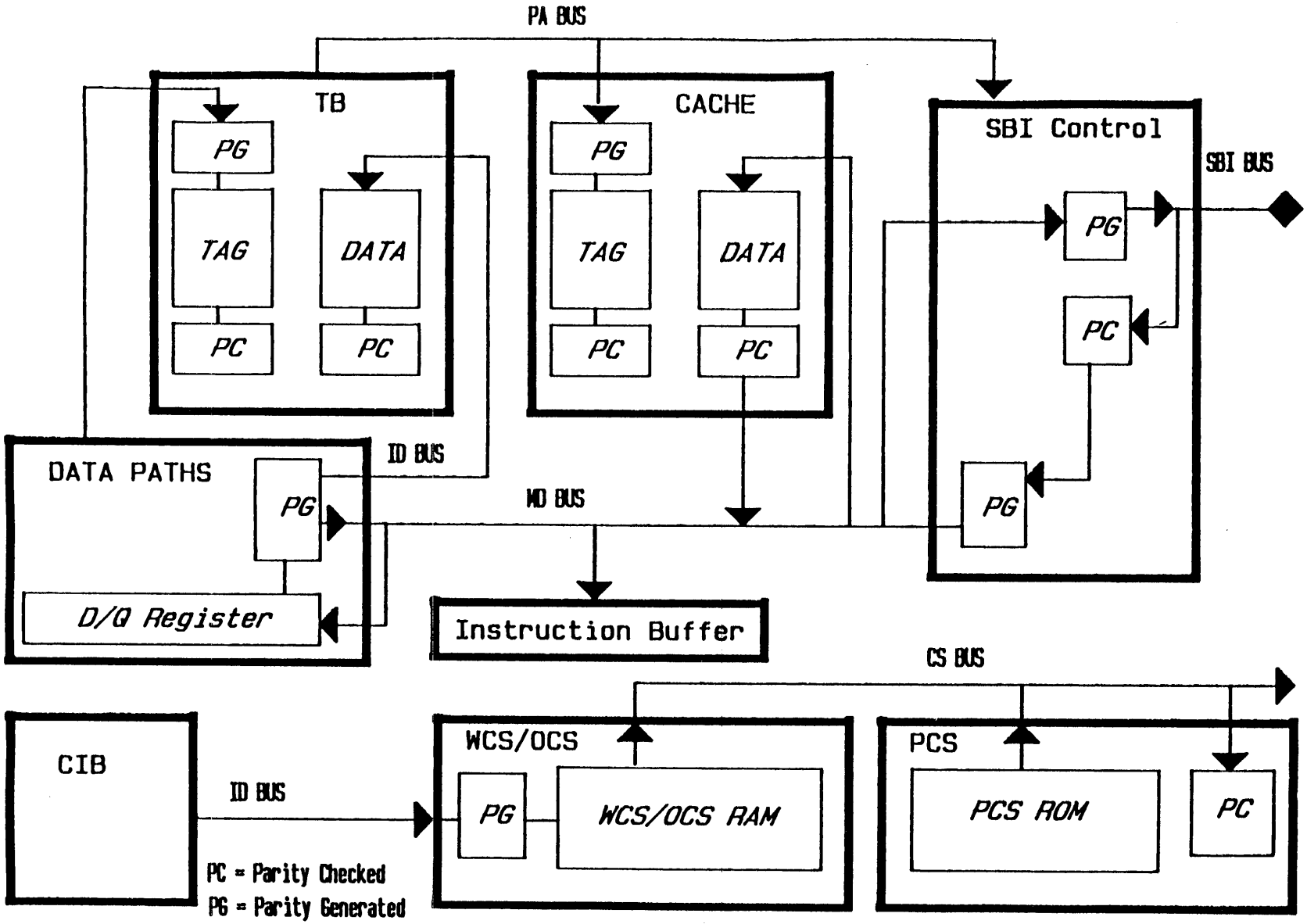


PC - General Register #F - *Program Counter*

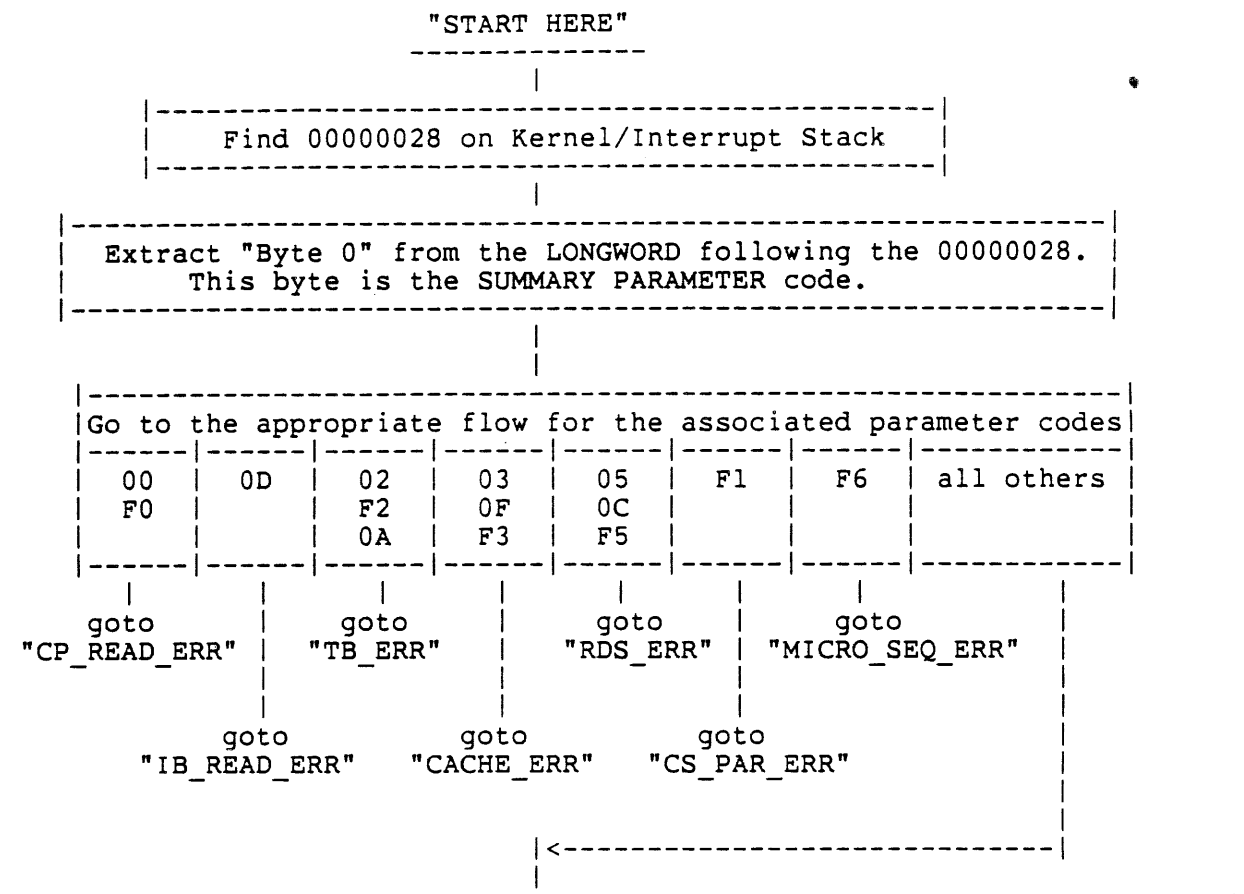


ID #0F - PSL - *Processor Status Longword*

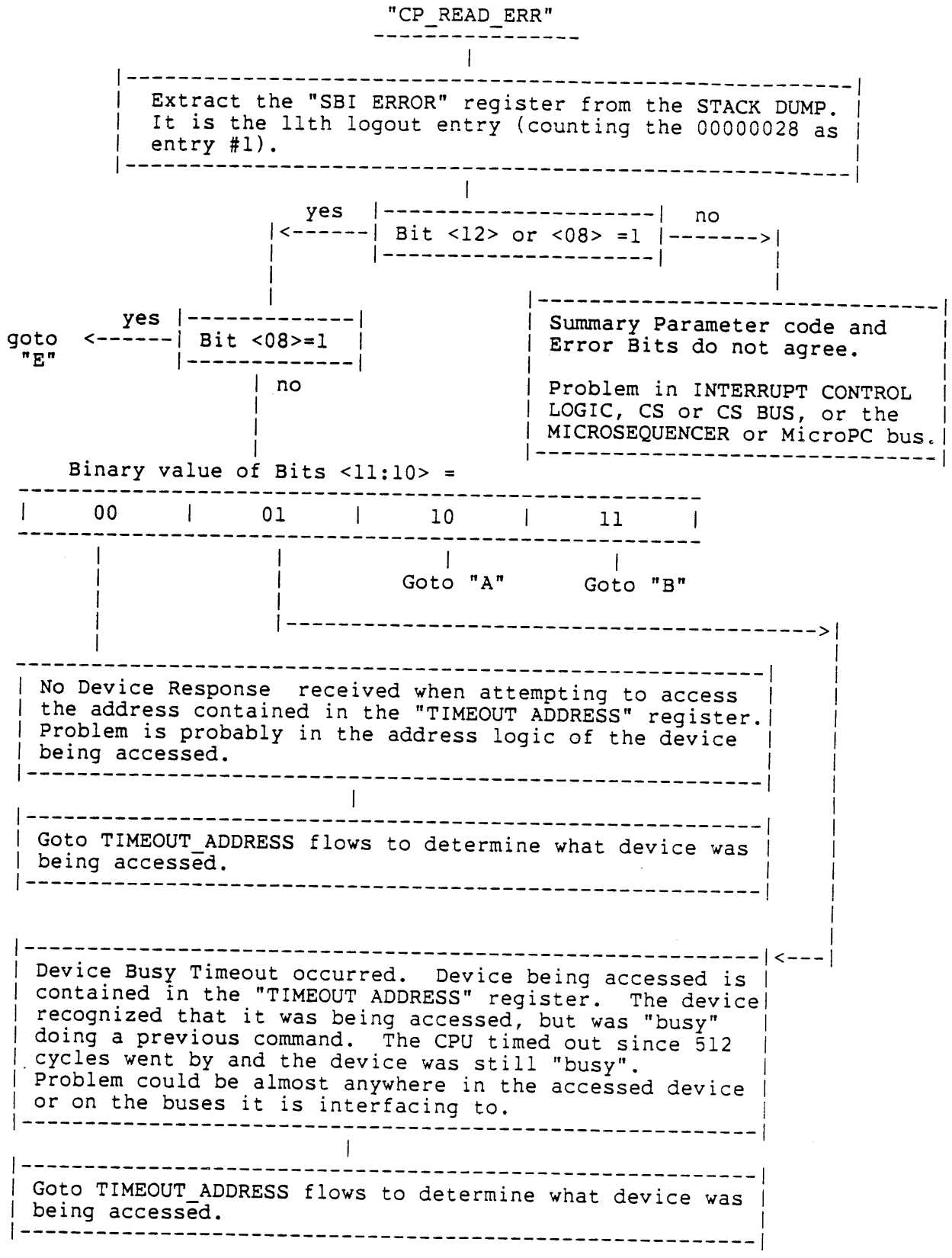


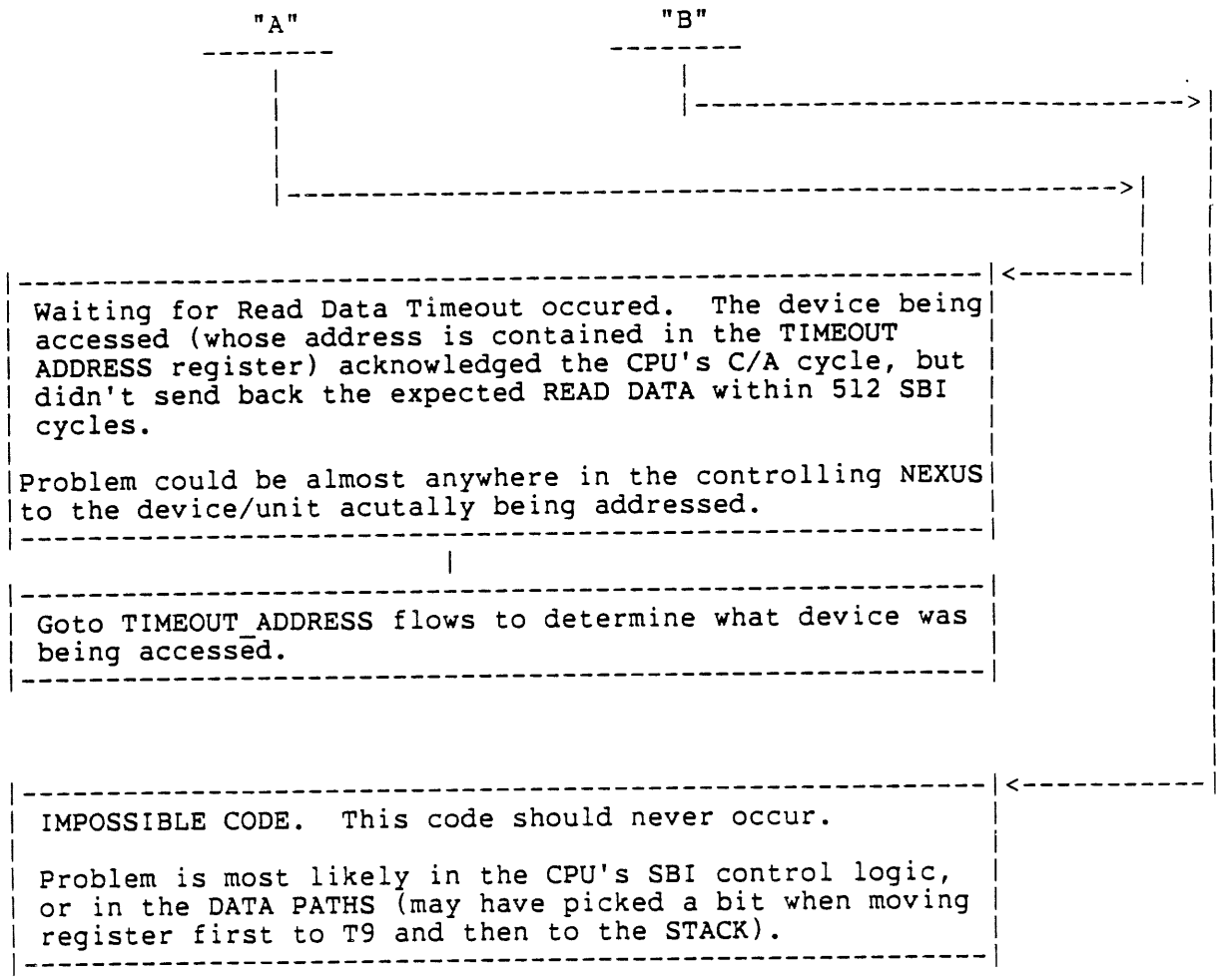

PA BUS
TB
PG
TAG
DATA
PC
PC
CACHE
PG
TAG
DATA
PC
PC
SBI Control
SBI BUS
PG
PC
PG
DATA PATHS
PG
ID BUS
MD BUS
D/Q Register
Instruction Buffer
CS BUS
CIB
ID BUS
WCS/OCS
PG
WCS/OCS RAM
PCS
PCS ROM
PC
PC = Parity Checked
PG = Parity Generated

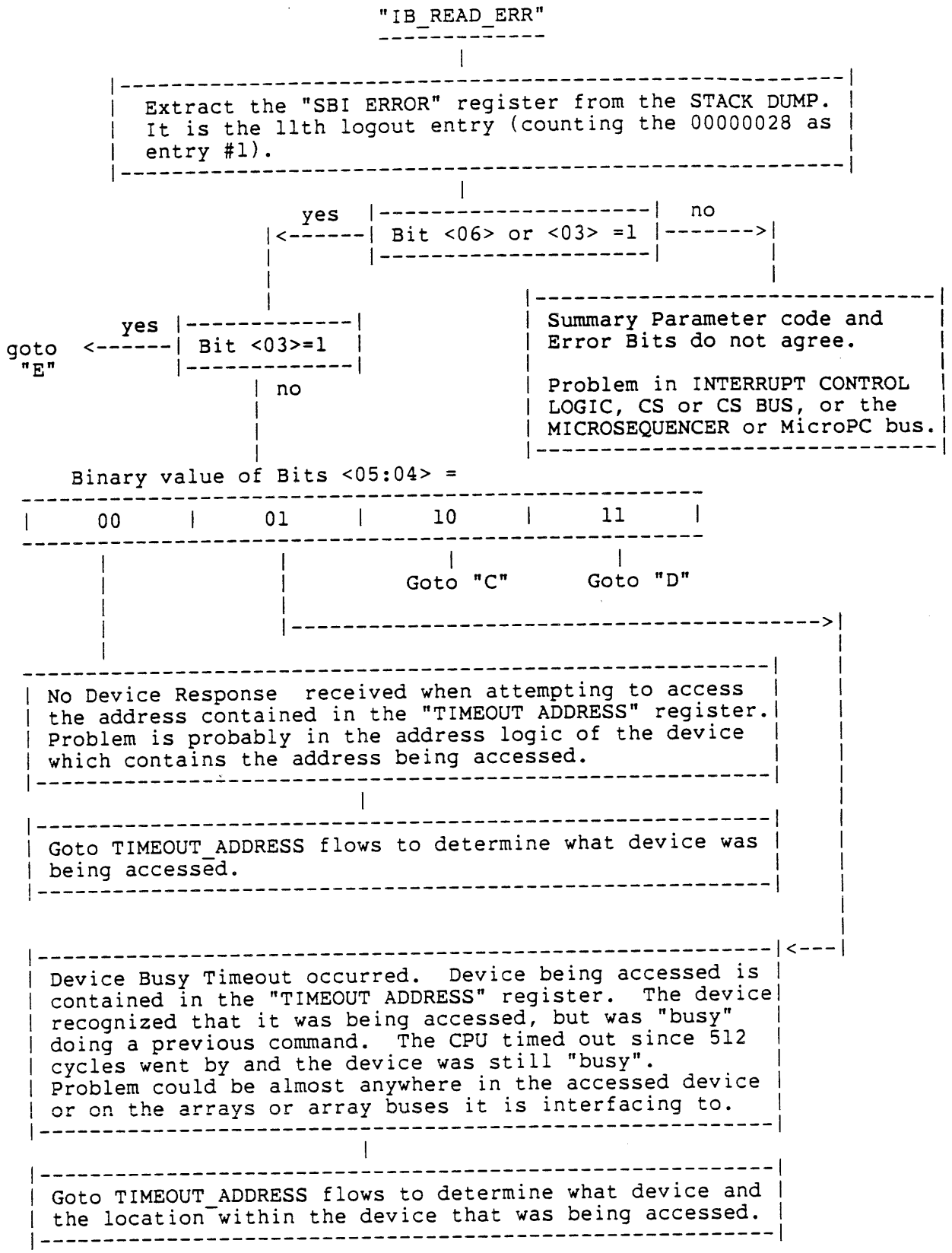
Machine Check Logout breakdown flowchart

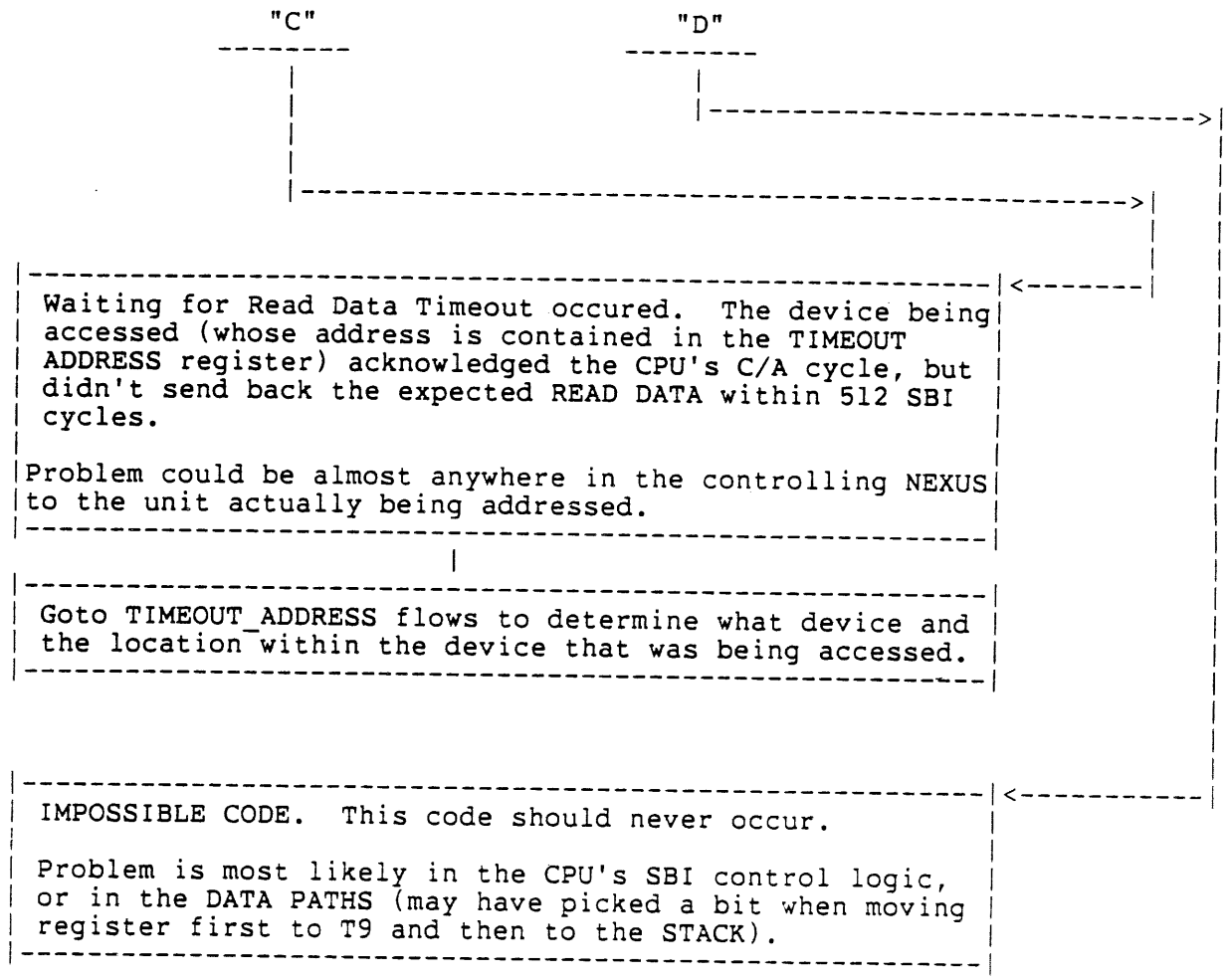


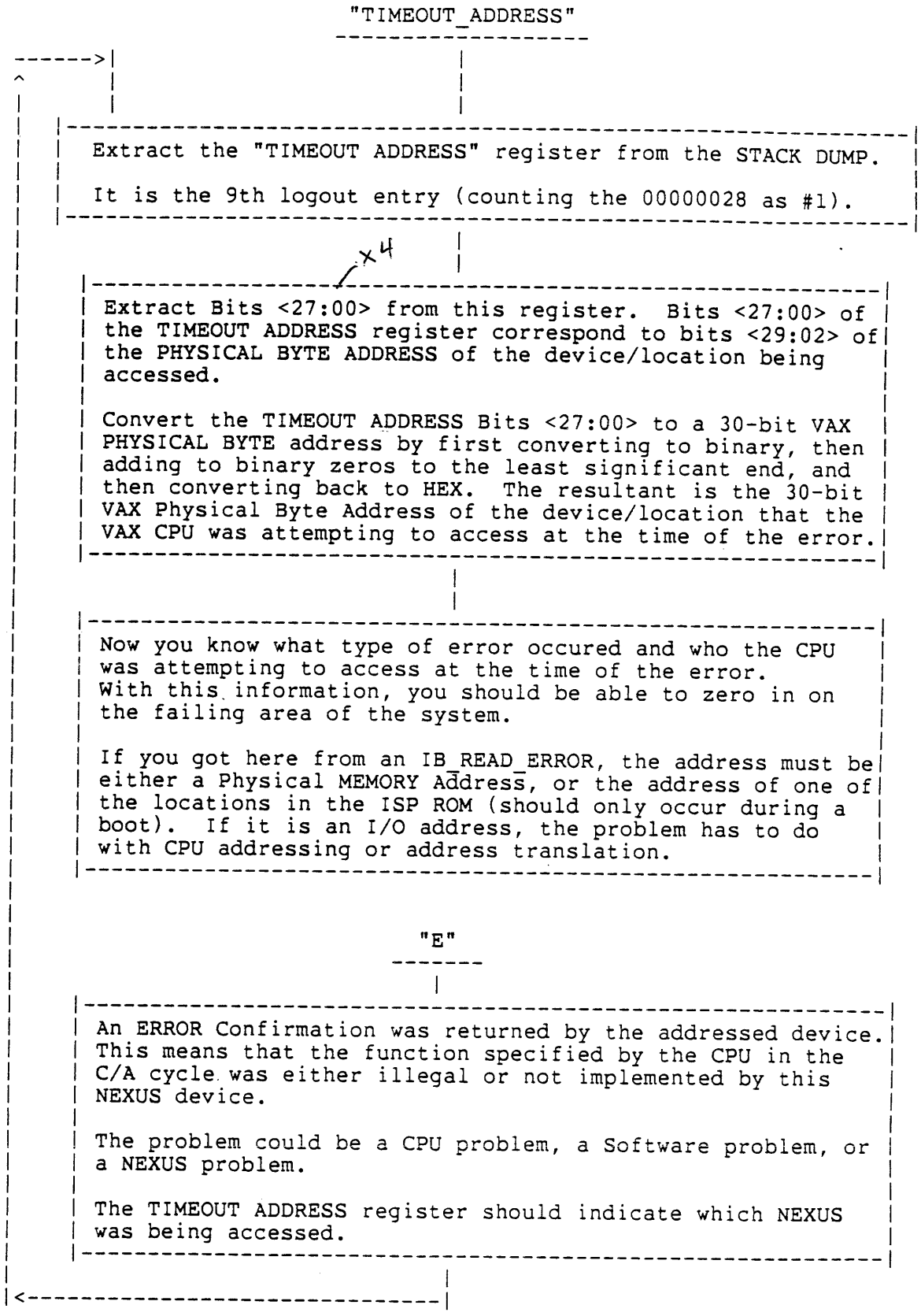
Error Type	Flowchart	Additional Info.
"CP_READ_ERR"	1.023	1.056
"IB_READ_ERR"	1.025	1.056
"TB_ERR"	1.028	1.038
"CACHE_ERR"	1.030	1.034
"RDS_ERR"	1.031	1.092
"CS_PAR_ERR"	1.032	1.046
"MICRO_SEQ_ERR"	1.033	1.094

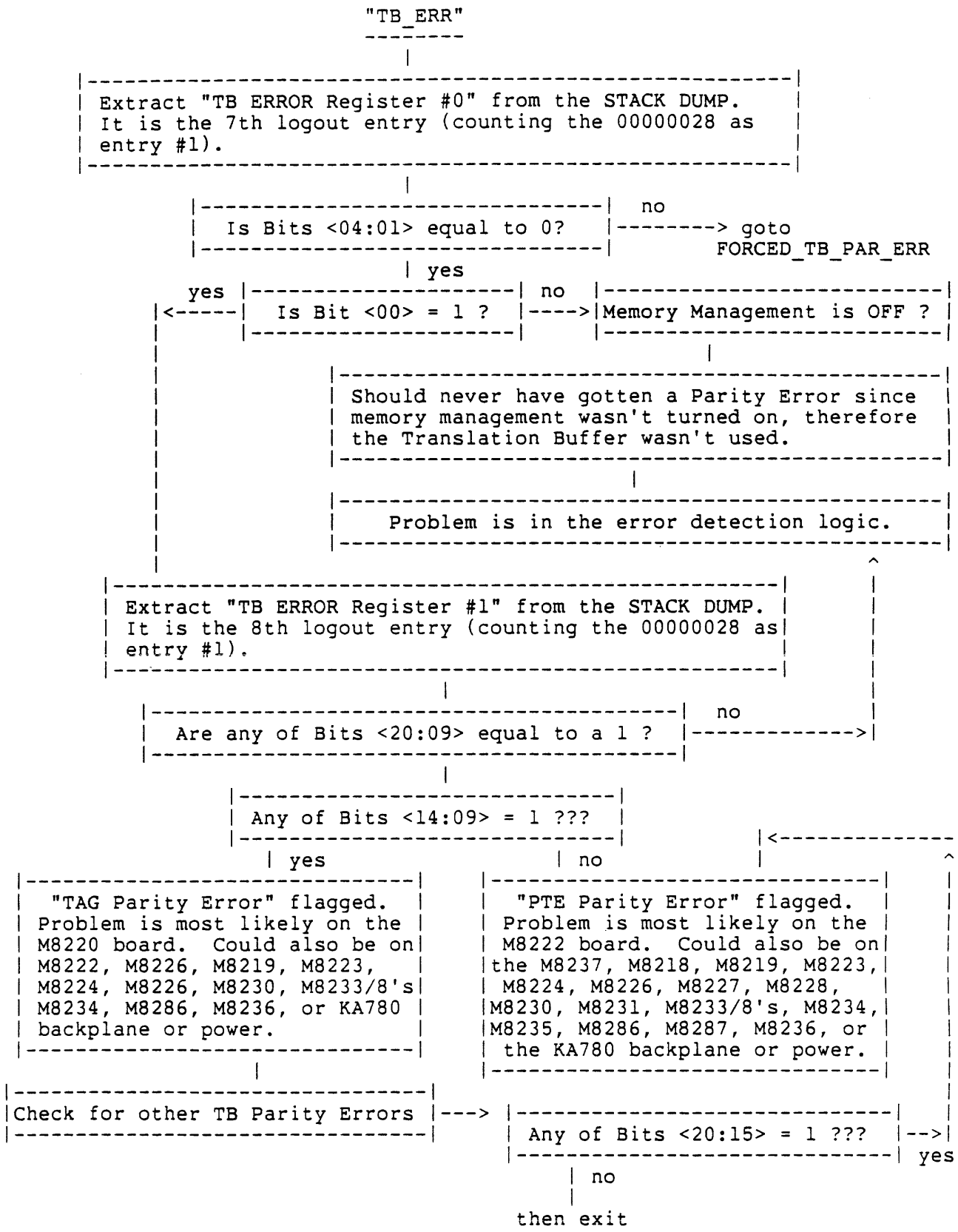

"CP_READ_ERR"
Extract the "SBI ERROR" register from the STACK DUMP. It is the 11th logout entry (counting the 00000028 as entry #1).
yes
Bit <12> or <08> =1
no
Summary Parameter code and Error Bits do not agree.
Problem in INTERRUPT CONTROL LOGIC, CS or CS BUS, or the MICROSEQUENCER or MicroPC bus.
yes
goto "E"
Bit <08>=1
no
Binary value of Bits <11:10> =
00
01
10
11
Goto "A"
Goto "B"
No Device Response received when attempting to access the address contained in the "TIMEOUT ADDRESS" register. Problem is probably in the address logic of the device being accessed.
Goto TIMEOUT_ADDRESS flows to determine what device was being accessed.
Device Busy Timeout occurred. Device being accessed is contained in the "TIMEOUT ADDRESS" register. The device recognized that it was being accessed, but was "busy" doing a previous command. The CPU timed out since 512 cycles went by and the device was still "busy". Problem could be almost anywhere in the accessed device or on the buses it is interfacing to.
Goto TIMEOUT_ADDRESS flows to determine what device was being accessed.

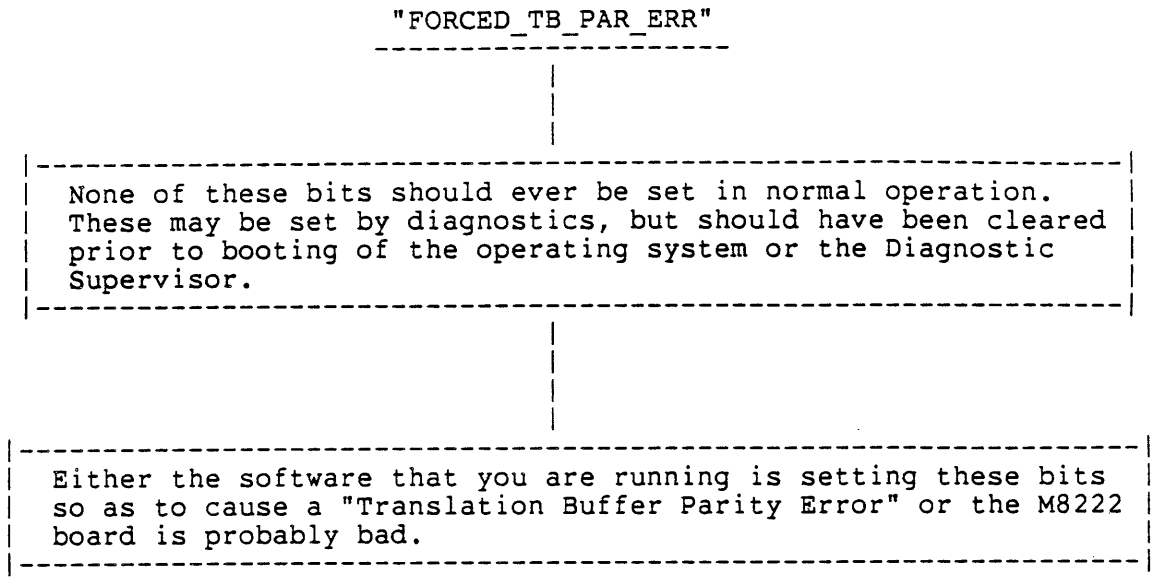

"A"
"B"
Waiting for Read Data Timeout occured. The device being accessed (whose address is contained in the TIMEOUT ADDRESS register) acknowledged the CPU's C/A cycle, but didn't send back the expected READ DATA within 512 SBI cycles.
Problem could be almost anywhere in the controlling NEXUS to the device/unit acutally being addressed.
Goto TIMEOUT_ADDRESS flows to determine what device was being accessed.
IMPOSSIBLE CODE. This code should never occur.
Problem is most likely in the CPU's SBI control logic, or in the DATA PATHS (may have picked a bit when moving register first to T9 and then to the STACK).

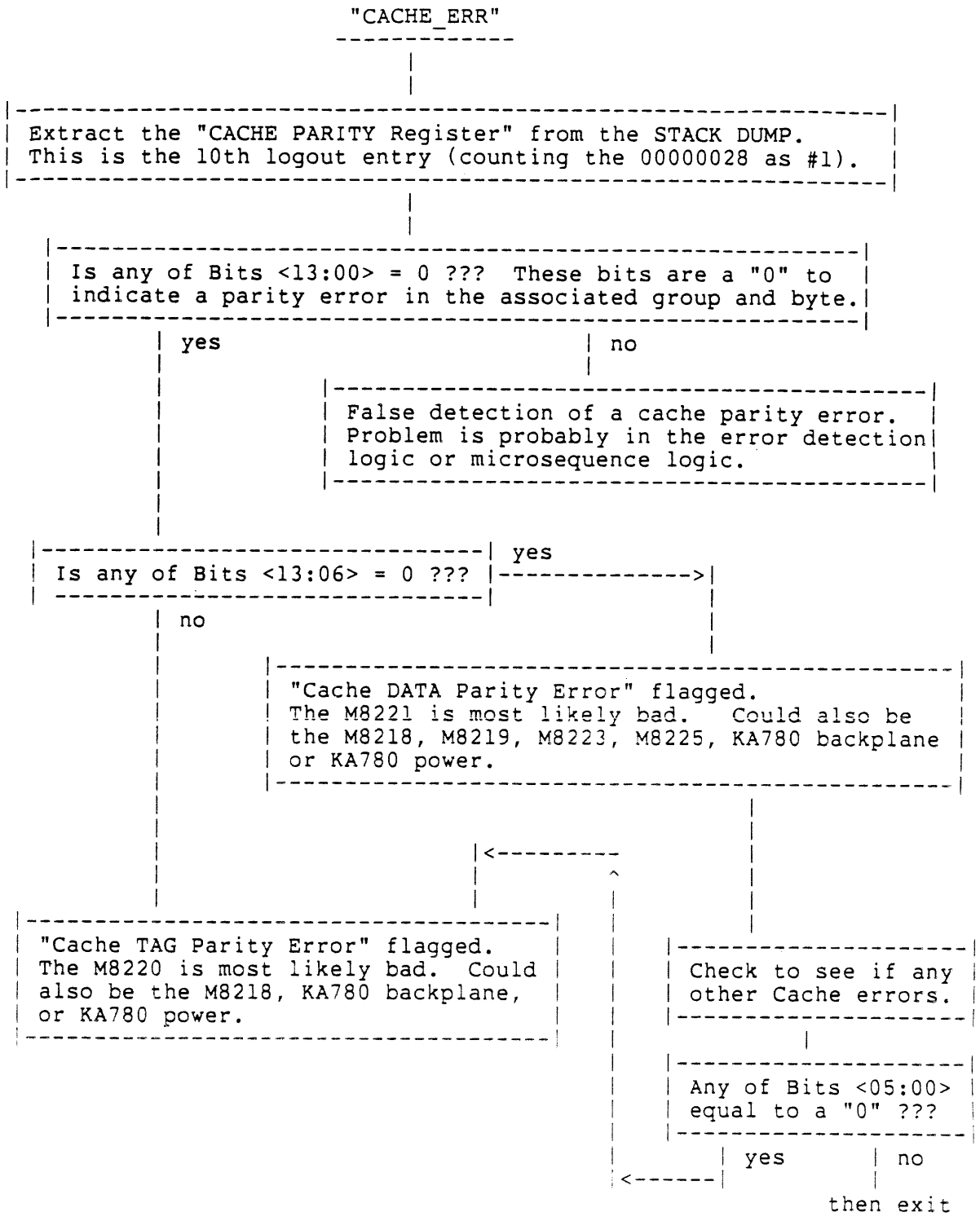

"IB_READ_ERR"
Extract the "SBI ERROR" register from the STACK DUMP. It is the 11th logout entry (counting the 00000028 as entry #1).
yes
Bit <06> or <03> =1
no
Summary Parameter code and Error Bits do not agree.
Problem in INTERRUPT CONTROL LOGIC, CS or CS BUS, or the MICROSEQUENCER or MicroPC bus.
yes
goto "E"
Bit <03>=1
no
Binary value of Bits <05:04> =
00
01
10
11
Goto "C"
Goto "D"
No Device Response received when attempting to access the address contained in the "TIMEOUT ADDRESS" register. Problem is probably in the address logic of the device which contains the address being accessed.
Goto TIMEOUT_ADDRESS flows to determine what device was being accessed.
Device Busy Timeout occurred. Device being accessed is contained in the "TIMEOUT ADDRESS" register. The device recognized that it was being accessed, but was "busy" doing a previous command. The CPU timed out since 512 cycles went by and the device was still "busy". Problem could be almost anywhere in the accessed device or on the arrays or array buses it is interfacing to.
Goto TIMEOUT_ADDRESS flows to determine what device and the location within the device that was being accessed.

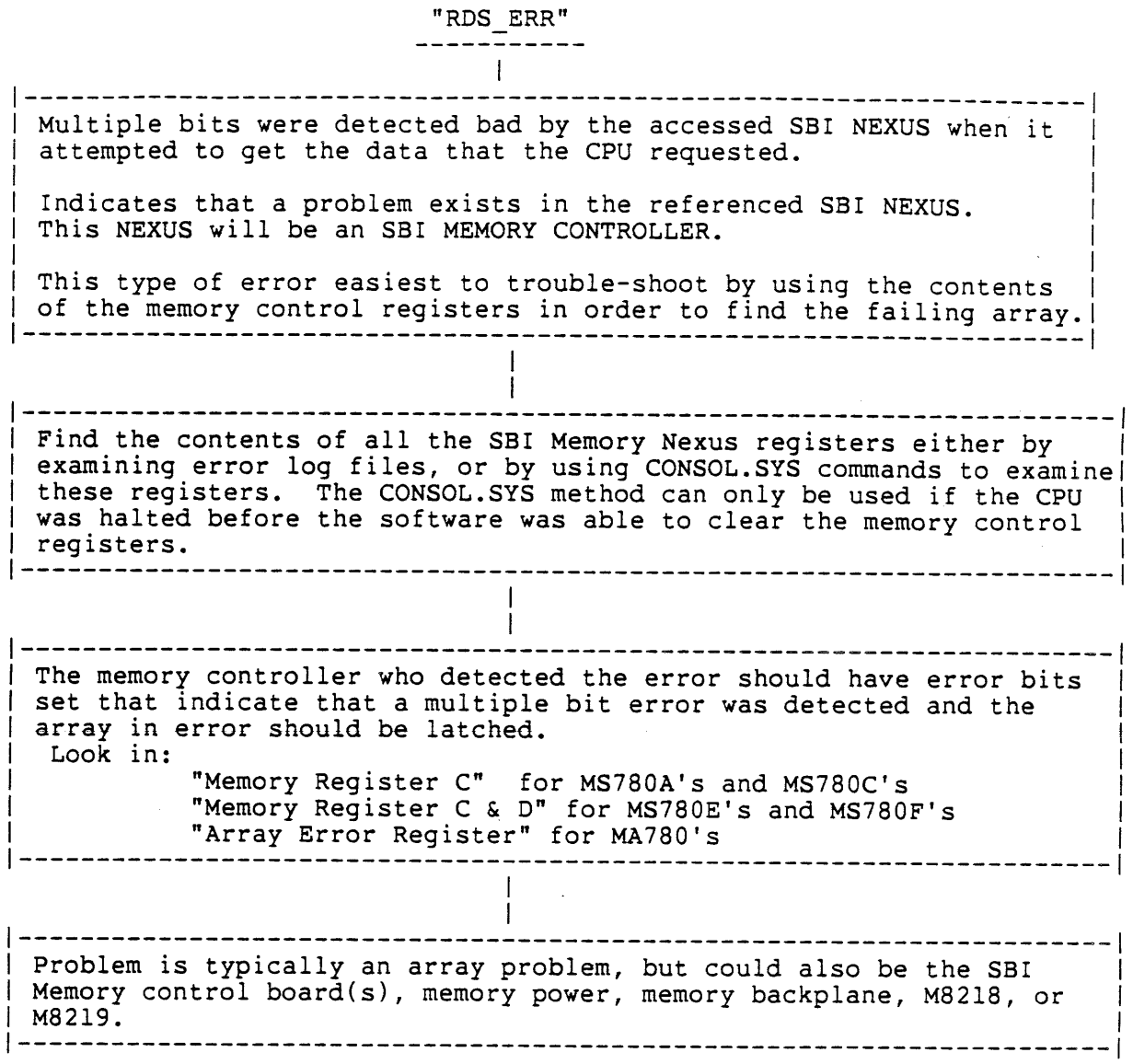

"C"
"D"
Waiting for Read Data Timeout occured. The device being accessed (whose address is contained in the TIMEOUT ADDRESS register) acknowledged the CPU's C/A cycle, but didn't send back the expected READ DATA within 512 SBI cycles.
Problem could be almost anywhere in the controlling NEXUS to the unit actually being addressed.
Goto TIMEOUT_ADDRESS flows to determine what device and the location within the device that was being accessed.
IMPOSSIBLE CODE. This code should never occur.
Problem is most likely in the CPU's SBI control logic, or in the DATA PATHS (may have picked a bit when moving register first to T9 and then to the STACK).

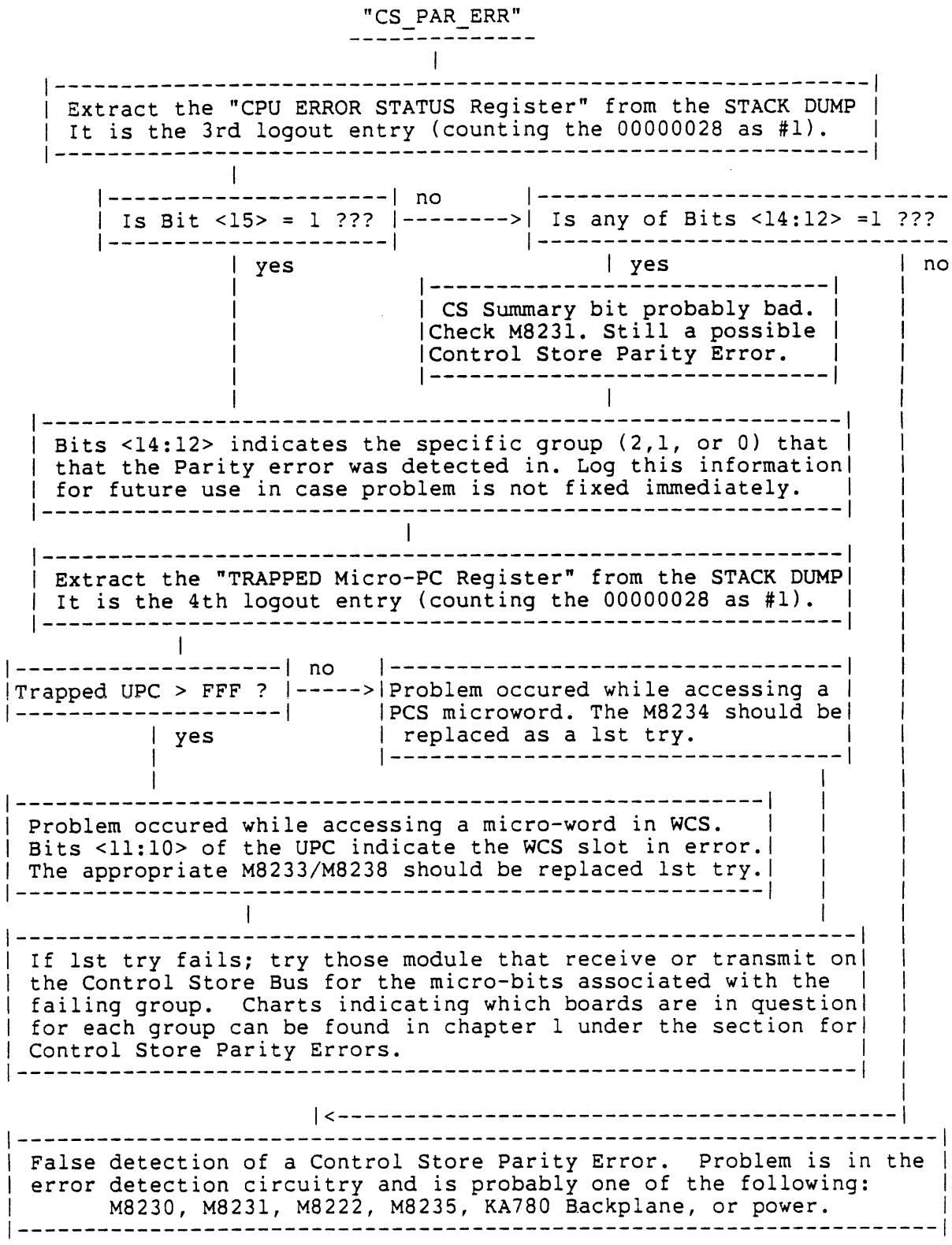

"TIMEOUT_ADDRESS"
Extract the "TIMEOUT ADDRESS" register from the STACK DUMP.
It is the 9th logout entry (counting the 00000028 as #1).
x4
Extract Bits <27:00> from this register. Bits <27:00> of the TIMEOUT ADDRESS register correspond to bits <29:02> of the PHYSICAL BYTE ADDRESS of the device/location being accessed.
Convert the TIMEOUT ADDRESS Bits <27:00> to a 30-bit VAX PHYSICAL BYTE address by first converting to binary, then adding to binary zeros to the least significant end, and then converting back to HEX. The resultant is the 30-bit VAX Physical Byte Address of the device/location that the VAX CPU was attempting to access at the time of the error.
Now you know what type of error occured and who the CPU was attempting to access at the time of the error. With this information, you should be able to zero in on the failing area of the system.
If you got here from an IB_READ_ERROR, the address must be either a Physical MEMORY Address, or the address of one of the locations in the ISP ROM (should only occur during a boot). If it is an I/O address, the problem has to do with CPU addressing or address translation.
"E"
An ERROR Confirmation was returned by the addressed device. This means that the function specified by the CPU in the C/A cycle was either illegal or not implemented by this NEXUS device.
The problem could be a CPU problem, a Software problem, or a NEXUS problem.
The TIMEOUT ADDRESS register should indicate which NEXUS was being accessed.

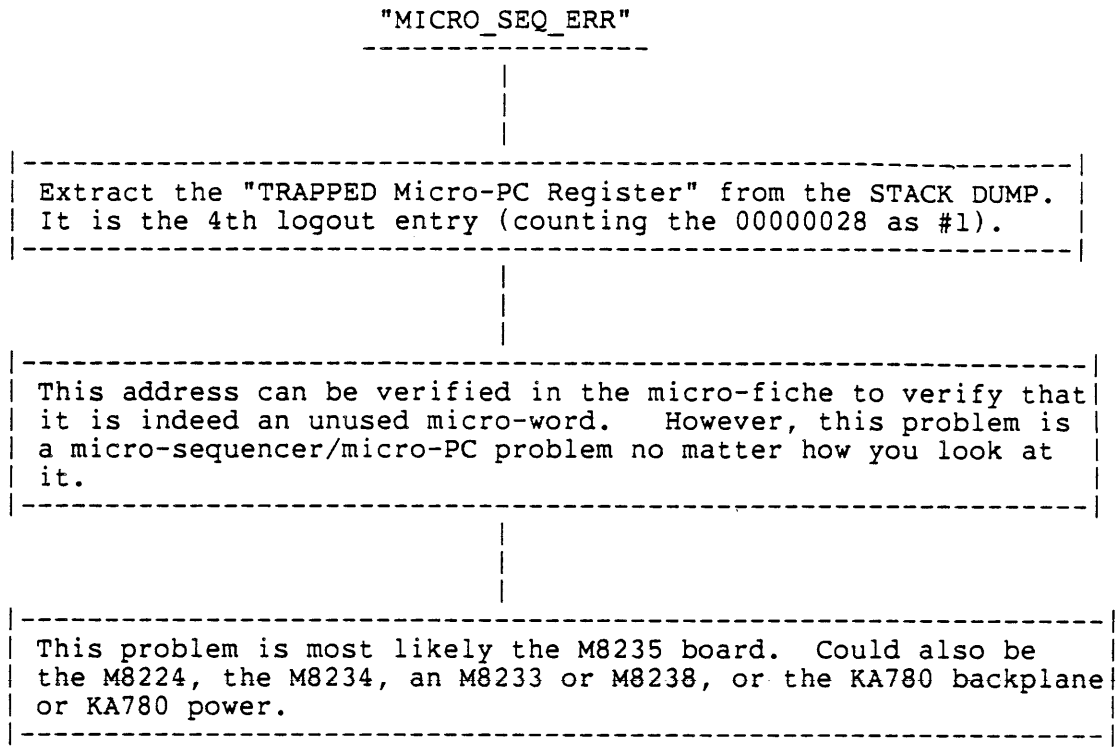

"TB_ERR"
Extract "TB ERROR Register #0" from the STACK DUMP. It is the 7th logout entry (counting the 00000028 as entry #1).
Is Bits <04:01> equal to 0?
no
goto FORCED_TB_PAR_ERR
yes
Is Bit <00> = 1 ?
yes
no
Memory Management is OFF ?
Should never have gotten a Parity Error since memory management wasn't turned on, therefore the Translation Buffer wasn't used.
Problem is in the error detection logic.
Extract "TB ERROR Register #1" from the STACK DUMP. It is the 8th logout entry (counting the 00000028 as entry #1).
Are any of Bits <20:09> equal to a 1 ?
no
Any of Bits <14:09> = 1 ???
yes
no
"TAG Parity Error" flagged. Problem is most likely on the M8220 board. Could also be on M8222, M8226, M8219, M8223, M8224, M8226, M8230, M8233/8's M8234, M8286, M8236, or KA780 backplane or power.
"PTE Parity Error" flagged. Problem is most likely on the M8222 board. Could also be on the M8237, M8218, M8219, M8223, M8224, M8226, M8227, M8228, M8230, M8231, M8233/8's, M8234, M8235, M8286, M8287, M8236, or the KA780 backplane or power.
Check for other TB Parity Errors
Any of Bits <20:15> = 1 ???
yes
no
then exit


"FORCED_TB_PAR_ERR"
None of these bits should ever be set in normal operation. These may be set by diagnostics, but should have been cleared prior to booting of the operating system or the Diagnostic Supervisor.
Either the software that you are running is setting these bits so as to cause a "Translation Buffer Parity Error" or the M8222 board is probably bad.


"CACHE_ERR"
Extract the "CACHE PARITY Register" from the STACK DUMP.
This is the 10th logout entry (counting the 00000028 as #1).
Is any of Bits <13:00> = 0 ??? These bits are a "0" to indicate a parity error in the associated group and byte.
yes
no
False detection of a cache parity error. Problem is probably in the error detection logic or microsequence logic.
Is any of Bits <13:06> = 0 ???
yes
no
"Cache DATA Parity Error" flagged. The M8221 is most likely bad. Could also be the M8218, M8219, M8223, M8225, KA780 backplane or KA780 power.
"Cache TAG Parity Error" flagged. The M8220 is most likely bad. Could also be the M8218, KA780 backplane, or KA780 power.
Check to see if any other Cache errors.
Any of Bits <05:00> equal to a "0" ???
yes
no
then exit


"RDS_ERR"
Multiple bits were detected bad by the accessed SBI NEXUS when it attempted to get the data that the CPU requested.
Indicates that a problem exists in the referenced SBI NEXUS. This NEXUS will be an SBI MEMORY CONTROLLER.
This type of error easiest to trouble-shoot by using the contents of the memory control registers in order to find the failing array.
Find the contents of all the SBI Memory Nexus registers either by examining error log files, or by using CONSOL.SYS commands to examine these registers. The CONSOL.SYS method can only be used if the CPU was halted before the software was able to clear the memory control registers.
The memory controller who detected the error should have error bits set that indicate that a multiple bit error was detected and the array in error should be latched.
Look in:
"Memory Register C" for MS780A's and MS780C's
"Memory Register C & D" for MS780E's and MS780F's
"Array Error Register" for MA780's
Problem is typically an array problem, but could also be the SBI Memory control board(s), memory power, memory backplane, M8218, or M8219.


"CS_PAR_ERR"
Extract the "CPU ERROR STATUS Register" from the STACK DUMP
It is the 3rd logout entry (counting the 00000028 as #1).
Is Bit <15> = 1 ???
no
yes
Is any of Bits <14:12> =1 ???
yes
no
CS Summary bit probably bad.
Check M8231. Still a possible
Control Store Parity Error.
Bits <14:12> indicates the specific group (2,1, or 0) that
that the Parity error was detected in. Log this information
for future use in case problem is not fixed immediately.
Extract the "TRAPPED Micro-PC Register" from the STACK DUMP
It is the 4th logout entry (counting the 00000028 as #1).
Trapped UPC > FFF ?
no
yes
Problem occured while accessing a
PCS microword. The M8234 should be
replaced as a 1st try.
Problem occured while accessing a micro-word in WCS.
Bits <11:10> of the UPC indicate the WCS slot in error.
The appropriate M8233/M8238 should be replaced 1st try.
If 1st try fails; try those module that receive or transmit on
the Control Store Bus for the micro-bits associated with the
failing group. Charts indicating which boards are in question
for each group can be found in chapter 1 under the section for
Control Store Parity Errors.
False detection of a Control Store Parity Error. Problem is in the
error detection circuitry and is probably one of the following:
M8230, M8231, M8222, M8235, KA780 Backplane, or power.


"MICRO_SEQ_ERR"
Extract the "TRAPPED Micro-PC Register" from the STACK DUMP. It is the 4th logout entry (counting the 00000028 as #1).
This address can be verified in the micro-fiche to verify that it is indeed an unused micro-word. However, this problem is a micro-sequencer/micro-PC problem no matter how you look at it.
This problem is most likely the M8235 board. Could also be the M8224, the M8234, an M8233 or M8238, or the KA780 backplane or KA780 power.

1.) ***** Cache Parity errors *****

The "Parity" register is normally all that is needed to trouble shoot this type of Machine Check.

Cache Parity is checked when the data is read from cache. The SBI control checks parity of SBI data as it is sent to the Cache from Memory. Cache Tag Parity is generated on the "CAM" board on a cache write and checked on the "CAM" board on a read from cache. Cache Data Parity is checked on the "CDM" board.

Bit #15 of ID Register #1E, equal to a 1, indicates that a Cache Parity error occurred.

If Bit #14 of ID #1E is set, the read reference was from the CP (micro-code). This bit is not really an error flag but simply states who made the reference that caused the Cache Parity Error. If Bit #15 is not set, this bit is of no real importance.

If Bit #14 of ID #1E is cleared, the read reference was from the IB (Instruction Buffer). Not an error flag bit.

Bits <13:06>, of ID #1E, define the Group and Byte of Bad DATA that the parity error was detected upon. Beware, the bad Group and Byte are indicated by a 0 in the appropriate bit location. These bits = 1 indicate parity was good.

Bits <05:00>, of ID #1E, define what Group and Byte has a bad address tag. Beware, the bad Group and Byte are indicated by a 0 in the appropriate bit location. These bits = 1 indicate parity was good.

ID Register #1E is stored in "(SP)+36" by the machine check logout. It is stored in ID Register #38 on a Double Error Halt's first error.

The Cache Data Matrix is on the "CDM" (M8221) board. The Cache Address Martix is on the "CAM" (M8220) board.

If ID Register #1E contains a parity error indication for the instruction buffer, the register is automatically cleared when the instruction buffer is flushed.

Problem areas if Cache "DATA Parity Error" :

1. Cache Data Matrix - M8221 - "CDM" - Slot 5

   Parity is checked as it is being read from the matrix. Parity that is written into the matrix comes directly from the MD bus (not checked or generated by the cache control logic on the way into the matrix).

2. SBI Control/Interface boards:

   a. If problem is in "BYTE #1 or #0"
      SBI Interface Low bits - M8218 - "SBL" - Slot 2

   b. If problem is in "BYTE #3 or #2"
      SBI Interface High bits - M8219 - "SBH" - Slot 3

   The SBI Control boards do not check the parity on the received MD Bus data but do generate parity for the data that is written from the SBI to the MD bus. The M8218 uses the output from the parity checkers to generate "SBLP G0 or G1 Par Err".

3. Instruction Buffer - M8223 - "IDP" - Slot 7

   Receives "Bus MD <31:00>" but NOT "Bus MD Byte <3:0> Par". Therefore, the Instruction Buffer does not check parity on the data used from the MD Bus.

4. Data Aligner - M8225 - "DBP" - Slot 9

   Transmits "Bus MD <31:00> + Bus MD Byte <3:0> Par". Receives "Bus MD <31:00>" but NOT "Bus MD Byte <3:0> Par", therefore, parity is not checked on "Bus MD <31:00> prior to use by the data path boards.

5. KA780 backplane

6. KA780 backplane power

Problem areas if Cache "TAG Parity Error" :

1. Cache Data Matrix - M8220 - "CAM" - Slot 4

   Parity is generated on the "Bus PA <29:12> bits" prior to being written into the Tag Matrix. Parity is checked as it is being read from the Tag Matrix, prior to use.

2. SBI Interface Low - M8218 - "SBl" - Slot 2

   This board uses the output from the parity checkers to create a "SBLP G0 Par Err" or "SBLP Gl Par Err" signal.

3. KA780 backplane

4. KA780 backplane power

Disabling CACHE *by KA780 backplane jumpers.*

If you cannot obtain the correct cache boards in order to fix a cache parity error problem, you may still be able to get the system up by installing the following jumpers:

D04P1 to a ground pin
D04P2 to a ground pin

This will cause a cache miss on all references. Therefore, the system will run much slower than normal.

This should only be done in case of an emergency. You must let the customer know that cache is disable, since it may cause problems due to program timing problems.

ID Register #1E

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Bits <31:16>
****

Not used, should be all zeros.

Bits <15:14>
****

15	14	
0	1	No Error
1	0	IB read reference caused the error
1	1	CP read reference caused the error

Bits <13:06>
****

Data Parity OK for specified Group and Byte if the associated bit is set.

13 - Parity OK for CDM Group 1 Byte 0
12 - Parity OK for CDM Group 1 Byte 1
11 - Parity OK for CDM Group 1 Byte 2
10 - Parity OK for CDM Group 1 Byte 3
09 - Parity OK for CDM Group 0 Byte 0
08 - Parity OK for CDM Group 0 Byte 1
07 - Parity OK for CDM Group 0 Byte 2
06 - Parity OK for CDM Group 0 Byte 3

Bits <05:00>
****

Address Parity OK for each Group and Byte if the associated bit is set.

05 - Parity OK for CAM Group 0 Byte 0
04 - Parity OK for CAM Group 0 Byte 1
03 - Parity OK for CAM Group 0 Byte 2
02 - Parity OK for CAM Group 1 Byte 0
01 - Parity OK for CAM Group 1 Byte 1
00 - Parity OK for CAM Group 1 Byte 2

2.) ***** **Translation Buffer Parity errors** *****

The TB ERR Registers, #0 and #1, are all that are needed to trouble-shoot Translation Buffer Parity error Machine Checks.

TB Data Parity is written as it was on the ID bus. TB Data Parity is checked on the "TBM" (M8222) board as it is read.

TB Tag Parity is generated on the "CAM" board and is checked on the "CAM" board on a read from the translation buffer. The TB Tag = "VAMX<30:15>" or "ID bus <31:26>".

The TB Data = "ID bus <20:00>".

Bits <20:09> of ID Register #13 define what the GROUP and whether it was a DATA Byte or ADDRESS Byte that caused the Translation Buffer Parity error.

The Translation Buffer ADDRESS matrix is on the "CAM" (M8220) board.

The Translation Buffer DATA matrix is on the "TBM" (M8222) board.

Problem areas if Translation Buffer "TAG Parity Error":

1. Cache Address/TB Address Matrix - M8220 - "CAM" - Slot 4

   Parity is generated on "VAMX bits <30:15>", and is written into TAG Matrix, on this board. Parity is checked, on this board, as the TAG is being read.

   The "Modify, Protect <3:0>, and Valid" bits are written from the ID bus. The associated parity bit for these bits also comes from the ID bus (it is NOT generated or checked on this board).

2. Translation Buffer Matrix - M8222 - "TBM" - Slot 6

   The output of the parity checkers goes to this board (used to set appropriate bits in "TB Register 0").

   The "VAMX bits" feed this module along with the "CAM" board (M8220).

3. Data Path bits <31:16> - M8226 - "DEP" - Slot 10

   The "VAMX bits" are created on this board and the "Bus ID bits <31:26>, that are used to write the "Modify, Protect <3:0>, and Valid" bits, are used on this board.

4. Any board that sends or receives the "Bus ID bits <31:26>". The following boards have receivers/drivers for these bits:

```
SBI Interface/Control High -  M8219    - "SBH" - Slot 3
Cache Address Matrix ------   M8220    - "CAM" - Slot 4
Instruction Data Path -----   M8223    - "IDP" - Slot 7
Instruction Decode --------   M8224    - "IRC" - Slot 8
Data Path bits <31:16> ----   M8226    - "DEP" - Slot 10
Condition Codes/Exceptions -  M8230    - "CEH" - Slot 14
Optional WCS --------------   M8233/8  - "OCS" - Slot 18
Writable Control Store ----   M8233/8  - "WCS" - Slot 20
Prom Control Store --------   M8234    - "PCS" - Slot 22
Fraction Multiplier High ---  M8286    - "FMH" - Slot 25
Console Interface Board  ---  M8236    - "CIB" - Slot 29
```

4. KA780 backplane

5. KA780 backplane power

Problem areas if Translation Buffer "DATA Parity Error":

1. Translation Buffer Matrix - M8222 - "TBM" - Slot 6

   Parity is not generated, on this board, for the data to be written into the DATA Matrix from the ID Bus. The parity bits that are written are the ID Bus Parity bits as received from the bus.

   Parity is checked as the data is read from the DATA matrix.

2. Any board on the ID bus that transmits or receives "Bus ID bits <20:00>". The following boards do this:

```
Terminator and Silo     - M8237   - "TRS" - Slot 1
SBI Interface Low Bits  - M8218   - "SBL" - Slot 2
SBI Interface High Bits - M8219   - "SBH" - Slot 3
Translation Buffer      - M8222   - "TBM" - Slot 6
Instruction Data Path   - M8223   - "IDP" - Slot 7
Instruction Decode      - M8224   - "IRC" - Slot 8
```

Data Path bits <31:16>	- M8226	- "DEP"	- Slot 10
Data Path bits <15:08>	- M8227	- "DDP"	- Slot 11
Data Path bits <07:00>	- M8228	- "DCP"	- Slot 12
Cond. Codes/Exceptions	- M8230	- "CEH"	- Slot 14
Interrupt Control	- M8231	- "ICL"	- Slot 15
Optional WCS	- M8233/8	- "OCS"	- Slot 18
Writable Control Store	- M8233/8	- "WCS"	- Slot 20
PROM Control Store	- M8234	- "PCS"	- Slot 22
Microsequence Control	- M8235	- "USC"	- Slot 23
Fraction Multiplier Hi	- M8286	- "FMH"	- Slot 25
Fraction Multiplier Low	- M8287	- "FML"	- Slot 26
Console Interface Board	- M8236	- "CIB"	- Slot 29

3. KA780 backplane

4. KA780 backplane power

ID #12 - Translation Buffer Register #0

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Bits <20:18> Force Replace
****

Directs TB writes to defined groups.

20 - Write Both
19 - Force Replace Group 1
18 - Force Replace Group 0

Bits <17:16> Force Miss
****

Force TB miss on the defined group.

17 - Group 1
16 - Group 0

Bits <15:08> Last Reference
****

Data on last non-nop memory reference.

15 ---- Status of micro-FS bit
14 ---- Status of micro-ADS bit
13:10 - Status of micro-MCT field
09 ---- 1 means IB WCHK existed on an IB reference
08 ---- 1 means reference delayed one cycle by IB auto-reload

Bits <07:06> TB Hit
****

Indicates which group was a TB hit.

07 - Group 1
06 - Group 0

Bits <04:01> Force TB Parity Error
****

Allows bad parity to be generated in the encoded Group and Byte.

Code of 0 - No errors
Code of 1 - No errors
Code of 2 - Group 0 Data Byte 0
Code of 3 - Group 0 Data Byte 1
Code of 4 - Group 0 Data Byte 2
Code of 5 - Group 1 Data Byte 0
Code of 6 - Group 1 Data Byte 1
Code of 7 - Group 1 Data Byte 2
Code of 8 - Group 0 Address Byte 0
Code of 9 - Group 0 Address Byte 1
Code of A - Group 0 Address Byte 2
Code of B - Group 1 Address Byte 0
Code of C - Group 1 Address Byte 1
Code of D - Group 1 Address Byte 2
Code of E - No errors
Code of F - No errors

Bit <00> MME
****

If a 1, enables Memory Management.

ID #13 - Translation Buffer Register #1

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Bits <20:09> TB Parity Error Status
****

Translation Buffer Error Status. When set indicates a parity error in the associated Group and Byte.

20 - Group 1 Data_Byte 2 Parity error

19 - Group 1 Data_Byte 1 Parity error
18 - Group 1 Data_Byte 0 Parity error
17 - Group 0 Data_Byte 2 Parity error
16 - Group 0 Data_Byte 1 Parity error

15 - Group 0 Data_Byte 0 Parity error
14 - Group 1 Address_Byte 2 Parity error
13 - Group 1 Address_Byte 1 Parity error
12 - Group 1 Address_Byte 0 Parity error

11 - Group 0 Address_Byte 2 Parity error
10 - Group 0 Address_Byte 1 Parity error
09 - Group 0 Address_Byte 0 Parity error

Bit <08> CP TB Parity Error
****

Indicates a TB micro-trap has been requested.

Bit <06> Last TB Write Pulse
****

Indicates which TB group was last written. Unpredictable if both were written into.

0 = Group 0
1 = Group 1

Bit <04> Bad IPA
****

Contents of IPA are not meaningful if this bit is set.

Bits <03:00> IPA information
****

Status of the last load from the IPA.

3 = 1 for TB miss on load.
2 = 1 for TB parity error.
1 = 1 for Protection violation or miss.
0 = 1 for automatic hardware initiated load.

3.) **** Control Store Parity errors (PCS, WCS, or OCS) ****

Two registers are used to trouble-shoot this type of Machine Check. They are as follows:

CPU Error Status (CES) - for Group the error occurred in.

Trapped UPC - for the micro-address of the error.

Bit <15> of the CES register must be a 1 if a Control Store parity error occurred. If CES Bit <15>=0, then the problem could be either the microcode board (M8234, M8238, or M8233) whose address appears in the "Trapped UPC", or one of the following boards:

M8231 - Contains the register ("CES") that holds the "CS Par Err Summary" bit and the "CS Par Err Group <2:0>" bits.

M8222 - Receives the "CS Parity Error" signals to use to stop TB operations.

M8230 - Creates the signals needed to trap the microcode for a CS Parity Error.

M8235 - Controls the micro-addressing.

Bit <12> of the "TRAPPED UPC" identifies where the Control Store Parity Error was generated from (WCS or PCS).

If Bit <12> is a 0, then the problem occurred as a result of a PCS (M8234) access.

If Bit <12> is a 1, then the problem occurred as a result of a WCS (M8233 or M8238) access. If an Optional WCS board is installed, further breakdown of the "TRAPPED UPC" address will reveal which WCS board is at fault.

The following statements will define the board at fault providing the lowest addressed WCS/OCS board is in slot 20 (addressing is controlled by VAX-11/780 backplane jumpers):

If there is an M8233 (1K board) in Slot 20 and:
Bit <12>=1, Bit <10>=0 then WCS in Slot 20 had the error.
Bit <12>=1, Bit <10>=1 then Optional WCS (slot 18) had the error.

If there is an M8238 (2K board) in Slot 20 and:
Bit <12>=1, Bit <11>=0 then WCS in Slot 20 had the error.
Bit <12>=1, Bit <11>=1 then Optional WCS (slot 18) had the error.

The "Parity Error" checking logic is located on the PCS (M8234) logic board. The 96 bit micro-word is broken into three 32 bit sections, each with an associated parity bit, and parity is checked on each section individually. Parity should be EVEN (an even number of ones in each 32 bit section counting the associated parity bit).

Use the bit configuration layout for ID register #20 for the "TRAPPED UPC" bit definitions.

The "TRAPPED UPC" is stored in "(SP)+12" on a MACHINE CHECK logout. The "TRAPPED UPC" remains in ID #32 for the first error of a DOUBLE ERROR HALT.

The "CPU Error Status Register" bits <14:12> define the failing 32 bit section of the 96 bit Micro-code word.

```
Bits <31:00> = Group 0  (CES Bit 12=1)  - Pin A22A1
Bits <63:32> = Group 1  (CES Bit 13=1)  - Pin A22S1
Bits <95:64> = Group 2  (CES Bit 14=1)  - Pin A22U2
```

If an OPTIONAL WCS board is installed in the System, Jumpers "W23 & W24" must be installed on the M8232, "CLK", board. If the jumpers are out, attempted access of the Optional WCS board will result in Control Store Parity Errors in all groups (the micro-word sent to "CS" bus is all ones). The same symptom will occur, when accessing any Micro-code board, if the clock lines are bad.

Be aware that the WCS data can only be written by the LSI Subsystem. WCS data cannot be read by the LSI Subsystem.

Problem areas:

PCS, WCS, or OPTIONAL WCS.
Incorrect setup of KA780 backplane jumpers for WCS and PCS.
No jumpers, for OPTIONAL WCS, on M8232.
Any Board on the "CS" bus.
M8232 Clock Board.
CPU Backplane.
CPU Power or Backplane Power Pin to Module connections.

ID #0C - CPU Error Status Register

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Bit <16> Nested Error
****

Used by Memory Management.

Bit <15> Control Store Parity Error Summary
****

Set if any of Bits <14:12> are set.

Bits <14:12> Control Store Parity Error bits
****

When set, indicates a parity error was detected in the associated group.

14 - Group 2 Parity Error
13 - Group 1 Parity Error
12 - Group 0 Parity Error

Bit <11> E ALU N
****

Bit <10> E ALU Z
****

Bit <09> ALU N
****

Bit <08> ALU Z
****

Bit <07> ALU C31
****

Bits <06:04> Arithmetic Trap Code
****

The octal code in these bits defines the type of arithmetic trap.

7 = Decimal divide by 0.
6 = Decimal overflow.
5 = Float underflow.
4 = Float divide by 0.
3 = Float overflow.
2 = Integer divide by 0.
1 = Integer overflow.
0 = No trap pending.

Bit <03> Performance Monitor Enable
****

Loaded or read by the microcode.

Bits <02:01> AST Level
****

Used to deliver AST SIR during RET.

ID #20 - Micro Stack Register

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Reading this register pops the top address from the micro stack.
Writing this register pushes an address onto the micro stack.

Bits <15:00> Control Store Address <15:00>
****

<15:00> = micro_Address <15:00>

Voltages to the Micro-code Boards

The Microcode boards use +5 volts and -5 volts. These voltages may be checked at the following places:

+5 volts should be on pins "A2" and "V1" of rows "A" thru "F" of each slot that contains a Microcode board.

-5 volts should be on pins "BL2" and "EK1" of each slot that contains a Microcode board.

Ground is on pins "C2" and "H1" of rows "A" thru "F" of each slot that contains a Microcode board.

M8235 LED description

The M8235, Micro Sequencer board, contains 14 LED's that reflect the following:

1.) "D1 - D13" = "Micro PC 00 - 12" respectively.
2.) "D14" = "STALL"

LED "D1" is the bottom most LED while LED "D14" is the uppermost.

CS Bus Groups and CS Bit Breakdown

Besides going to all the Microcode boards the "Bus CS" bits also go to the following boards:

Group 0	<12:00>	M8235 - USC - Slot 23
	<19:13>	M8227 - DDP - Slot 11
	<22:20>	M8230 - CEH - Slot 14
	<24:23>	M8227 - DDP - Slot 11
	<25>	M8228 - DCP - Slot 12
	<26>	M8230 - CEH - Slot 14
	<31>	M8230 - CEH - Slot 14
Group 1	<34:32>	M8228 - DCP - Slot 12
	<41:35>	M8229 - DAP - Slot 13
	<45:42>	M8231 - ICL - Slot 15
		M8222 - TBM - Slot 6
	<47:46>	M8222 - TBM - Slot 6
	<54:48>	M8229 - DAP - Slot 13
	<57:55>	M8229 - DAP - Slot 13
		M8289 - FCT - Slot 28
	<58>	M8231 - ICL - Slot 15
		M8228 - DCP - Slot 12
	<63>	M8231 - ICL - Slot 15
Group 2	<65:64>	M8235 - USC - Slot 23
	<69:66>	M8229 - DAP - Slot 13
	<71:70>	M8289 - FCT - Slot 28
	<74:72>	M8235 - USC - Slot 23
		M8231 - ICL - Slot 15
	<76:75>	M8235 - USC - Slot 23
	<77>	M8229 - DAP - Slot 13
	<79:78>	M8229 - DAP - Slot 13
		M8230 - CEH - Slot 14
	<87:80>	M8229 - DAP - Slot 13
	<91:88>	M8225 - DBP - Slot 9
	<95:92>	M8223 - IDP - Slot 7

Note: Remember that "Bus CS <95:00>" also go to slots 18,20 and 22.

Chart showing "Bus CS" bits to each Board by Board

M8222 - "TBM" - Slot 6

<45:42> - Group 1
<47:46> - Group 1

M8223 - "IDP" - Slot 7

<95:92> - Group 2

M8225 - "DBP" - Slot 9

<91:88> - Group 2

M8227 - "DDP" - Slot 11

<19:13> - Group 0
<24:23> - Group 0

M8228 - "DCP" - Slot 12

<25> - Group 0
<34:32> - Group 1
<58> - Group 1

M8229 - "DAP" - Slot 13

<41:35> - Group 1
<54:48> - Group 1
<57:55> - Group 1
<69:66> - Group 2
<77> - Group 2
<79:78> - Group 2
<87:80> - Group 2

M8230 - "CEH" - Slot 14

<22:20> - Group 0
<26> - Group 0
<31> - Group 0
<79:78> - Group 2

M8231 - "ICL" - Slot 15

<45:42> - Group 1
<58> - Group 1
<63> - Group 1
<74:72> - Group 2

M8235 - "USC" - Slot 23

<12:00> - Group 0
<65:64> - Group 2
<74:72> - Group 2
<76:75> - Group 2

M8289 - "FCT" - Slot 28

<57:55> - Group 1
<71:70> - Group 2

M8234 - "PCS" - Slot 22

<95:00> - Groups 0,1,2

M8233/M8238 - "OCS" - Slot 18

<95:00> - Groups 0,1,2

M8233/M8238 - "WCS" - Slot 20

<95:00> - Groups 0,1,2

Chart showing "Bus CS" Groups to each Board

Board	Group 0	Group 1	Group 2
M8222		X	
M8223			X
M8225			X
M8227	X		
M8228	X	X	
M8229		X	X
M8230	X		X
M8231		X	X
M8233	X	X	X
M8234	X	X	X
M8235	X		X
M8238	X	X	X
M8289		X	X

Using the Microcode Sync Point for scoping of the CS Bus

The VAX-11/780 CPU has a "Microcode Sync Point" that can be set up to provide a scope trigger whenever the Microcode reaches a specified address. To use this feature, proceed as follows:

1. Determine what Micro PC you want the Sync to trigger at.
2. Deposit, using the CONSOL.SYS program, the address into ID register #21.
3. Place your scopes SYNC on pin "A23V2" of the CPU backplane.
4. Start the failing Macro program.
5. You can now scope the "CS" bus to determine what bit(s) are bad.

The VAX-11/780 CPU also has logic that can stop the CPU when the microcode reaches a specified Micro PC. This feature may be used as follows:

1. Determine what Micro PC you want the CPU to halt at.
2. Deposit, using the CONSOL.SYS program, the desired address into ID register #21.
3. Set, using the CONSOL.SYS program, the Stop on Micro Match bit with the "SET SOMM" command.
4. Start the appropriate failing macro program. The VAX-11/780 CPU will halt when it reaches the Micro PC specified in ID register #21. You can then scope the logic in the static state.
5. Be sure to execute the "CLEAR SOMM" command before returning the system to the customer.

The Control Store Bit Backplane pin layout follows for the slots that contain WCS and PCS boards (18, 20, and 22).

Group 0 *(VAX CPU Slot 18,20, or 22.)*

Bus CS 00 - AB1	Bus CS 11 - AL1	Bus CS 22 - BM2
Bus CS 01 - AB2	Bus CS 12 - AL2	Bus CS 23 - AV2
Bus CS 02 - AC1	Bus CS 13 - AM2	Bus CS 24 - BD1
Bus CS 03 - AD1	Bus CS 14 - AR1	Bus CS 25 - BD2
Bus CS 04 - AD2	Bus CS 15 - AR2	Bus CS 26 - BN1
Bus CS 05 - AE1	Bus CS 16 - BA1	Bus CS 27 - BP1
Bus CS 06 - AE2	Bus CS 17 - BB1	Bus CS 28 - BP2
Bus CS 07 - AF1	Bus CS 18 - BB2	Bus CS 29 - BR1
Bus CS 08 - AJ2	Bus CS 19 - BC1	Bus CS 30 - BR2
Bus CS 09 - AK1	Bus CS 20 - BL1	Bus CS 31 - BS1
Bus CS 10 - AK2	Bus CS 21 - BM1	Bus CS UPAR 0 - FL2

Group 1 *(VAX CPU Slot 18,20, or 22.)*

Bus CS 32 - BE1	Bus CS 43 - CS1	Bus CS 54 - DU2
Bus CS 33 - BE2	Bus CS 44 - CS2	Bus CS 55 - DD2
Bus CS 34 - BF1	Bus CS 45 - CT2	Bus CS 56 - DF2
Bus CS 35 - BS2	Bus CS 46 - CU1	Bus CS 57 - DH2
Bus CS 36 - BT2	Bus CS 47 - CU2	Bus CS 58 - DJ1
Bus CS 37 - BU1	Bus CS 48 - CD2	Bus CS 59 - DJ2
Bus CS 38 - BU2	Bus CS 49 - CN1	Bus CS 60 - DM1
Bus CS 39 - BV2	Bus CS 50 - CP1	Bus CS 61 - DN1
Bus CS 40 - CC1	Bus CS 51 - DP2	Bus CS 62 - DP1
Bus CS 41 - CD1	Bus CS 52 - DT2	Bus CS 63 - DS2
Bus CS 42 - CP2	Bus CS 53 - DU1	Bus CS UPAR 1 - FM1

Group 2 *(VAX CPU Slot 18,20, or 22.)*

Bus CS 64 - EH2	Bus CS 75 - EN1	Bus CS 86 - FU2
Bus CS 65 - EJ1	Bus CS 76 - EP1	Bus CS 87 - FV2
Bus CS 66 - EP2	Bus CS 77 - EB1	Bus CS 88 - FP1
Bus CS 67 - ES2	Bus CS 78 - FA1	Bus CS 89 - FP2
Bus CS 68 - ET2	Bus CS 79 - FB1	Bus CS 90 - FR1
Bus CS 69 - EU1	Bus CS 80 - FC1	Bus CS 91 - FR2
Bus CS 70 - DC1	Bus CS 81 - FL1	Bus CS 92 - FJ1
Bus CS 71 - DD1	Bus CS 82 - FS1	Bus CS 93 - FJ2
Bus CS 72 - EJ2	Bus CS 83 - FS2	Bus CS 94 - FK1
Bus CS 73 - EK2	Bus CS 84 - FT2	Bus CS 95 - FK2
Bus CS 74 - EM1	Bus CS 85 - FU1	Bus CS UPAR 2 - FM2

4.) ***** **CPU READ Timeouts or Error Confirmation Aborts** *****
during Instruction Buffer ("IB") or Micro-code ("CP") accesses.

Two registers are needed to correctly trouble-shoot this type of Machine Check. They are as follows:

TIMEOUT Address	-	to determine what device or location was being referenced when error occurred.
SBI Error Reg.	-	to determine what type of error occurred.

The VAX-11/780 Processor initiates accesses to SBI NEXUS from two separate sources. The VAX microcode can initiate "read or write" accesses to any address and does so to obtain operands (these operands may be used to calculate source/destination addresses or may be the operand that will be operated on.). The Instruction Buffer is the other source from which the CPU can initiate an SBI data transfer. The IB, however, can only do read accesses and these accesses are used to fetch instruction stream data.

This type of error occurs whenever one of the following types of conditions has occurred:

1. An attempt was made to access a non-existent NEXUS address or a NEXUS did not respond when accessed. This will result in the VAX CPU receiving a "NO DEVICE RESPONSE" confirmation on the second cycle following the "Command Address" or "Write Data" cycle. A "NO DEVICE RESPONSE" confirmation is when the SBI CNF<1:0> lines are deasserted. The CPU will wait a cycle and then retry the cycle that got the NO DEVICE RESPONSE confirmation. If 512 cycles elapse, from the initial C/A cycle, before an ACKNOWLEDGE confirmation is received the CPU will timeout and a Machine Check Exception will occur.

2. The CPU detected a "Device Busy" response from a NEXUS and 512 SBI cycles later, still receives a "Device Busy" response on attempted accesses. Whenever the VAX CPU detects a "Device Busy" response, it will wait a cycle, and then will arbitrate for the bus in an attempt to retry the same transfer. This continues until the CPU receives an "Acknowledge" confirmation response or until 512 SBI cycles have occurred from the first transfer attempt (in this case, the CPU will timeout and a Machine Check exception will occur).

3. The CPU receives an "Error" confirmation to a "Command Address" transfer. This usually means that the CPU has requested a function that the NEXUS cannot perform.

The "TIMEOUT ADDRESS", ID Register #1A, latches the SBI PHYSICAL Longword Address whenever an SBI CP Timeout occurs. When using this ID register, be aware that ID #1A bits <27:00> are bits <29:02> of the PHYSICAL BYTE ADDRESS. The SBI address is a LONGWORD address. To convert to a BYTE Physical address, simply insert two BINARY ZERO's to the right of the least significant digit (first convert the LONGWORD S.B.I. address from HEX format to BINARY format, then add two zeros to the right, least significant digits, and then convert the result back to HEX format).

The "TIMEOUT ADDRESS", ID #1A, does not latch the physical SBI address for IB data timeouts, but ID #1A "may" still be valid.

The "TIMEOUT ADDRESS", ID Register #1A, is stored in "(SP)+32" on a MACHINE CHECK logout. It is stored in ID register #37 on the first error of a Double Error Halt.

Bits <27:00> of ID Register #1A, are Bits <29:02> of the SBI Physical Byte Address. The SBI uses Longword Addresses. Bits <31:30> contain the "MODE" of the reference and Bit 29 flags whether or not a hardware protection check was done.

Instruction Buffer accesses should always be "reads" in order to obtain instruction stream data. Therefore, they should always be to a memory location or ISP ROM location.

ID register #19 identifies the type of error that occurred. If Bit 12=1 a CP timeout occurred and Bits <11:10> identify the type of timeout. If Bit 06=1 an IB timeout occurred and Bits <5:4> identify the type of timeout that occurred. If Bit 08=1 the error was due to an Error Confirmation as a result of a CP reference. If Bit 03=1 the error was due to an Error Confirmation as a result of an IB reference.

ID register #19 also contains some other bits that may be of interest. Bit 13 flags the fact that a multiple bit error occurred in memory and that the data received by the VAX CPU is bad.
Bit 7 flags the fact that a multiple bit error occurred in memory while fetching data for the Instruction Buffer (IB).
Bit 2 flags a Multiple CP error, (another error occurred before the first error was cleared).

Problem areas:

CPU SBI interface.
CPU Memory/CACHE control.
Memory.
Any Nexus.
Power.
SBI cables.

ID #19 - SBI Error Register

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Bit <15> RDS Interrupt Enable

Enable interrupts for Read Data Substitute (Bad Data) errors.

Bit <14> CRD

Received corrected read data (CRD) from memory.

Bit <13> RDS

Received read data substitute (RDS) from memory.

Bits <12:10> CP Timeout Status

If 12 = 1, indicates a timeout occurred as a result of a CP requested cycle.

12	11	10	
1	0	0	"No Device Response" timeout.
1	0	1	"Device Busy" timeout.
1	1	0	"Waiting for Read Data" timeout.
1	1	1	Impossible code.

Bit <08> CP SBI Error Confirmation

Set when the CP requested cycle received an error confirmation to a command address transfer.

Bit <07> IB RDS

Read data substitute (RDS) data was received from memory on an IB data request.

Bits <06:04> IB Timeout Status
****

If bit <06>=1, a timeout occurred on an IB requested cycle.

06	05	04	
1	0	0	"No Device Response" timeout.
1	0	1	"Device Busy" timeout.
1	1	0	"Waiting for Read Data" timeout.
1	1	1	Impossible code.

Bit <03> IB SBI Error Confirmation
****

Set when an IB requested cycle receives an error confirmation.

Bit <02> Multiple CP Error
****

Set with pending CP timeout or CP SBI error confirmation not serviced.

Bit <01> SBI Not Busy
****

ID #1A - Timeout Address Register

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Latches the SBI PHYSICAL LONGWORD Address on an SBI Timeout. Will not latch for IB data timeouts.

Bits <31:30> Mode
****

31	30	
0	0	Kernel mode
0	1	Executive mode
1	0	Supervisor mode
1	1	User mode

Bit <29> Protection Check
****

Equal to a zero for references not subject to a hardware protection check.

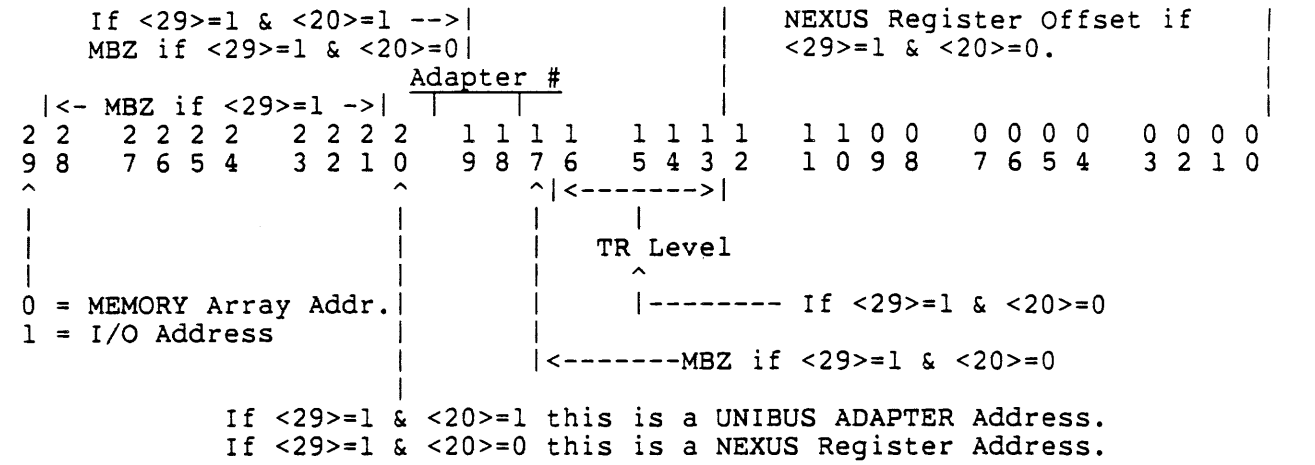
Bits <27:00> SBI Physical Longword Address
****

Contains the latched Physical Address bits <29:02> of the SBI Physical address.

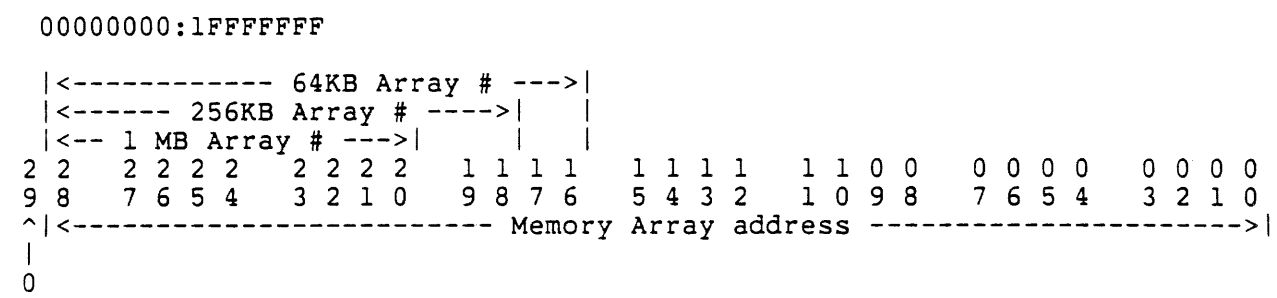
<27:00> = PA<29:02>

Note: A written step by step procedure on how to break down Physical VAX BYTE addresses follows these bit breakdown charts.

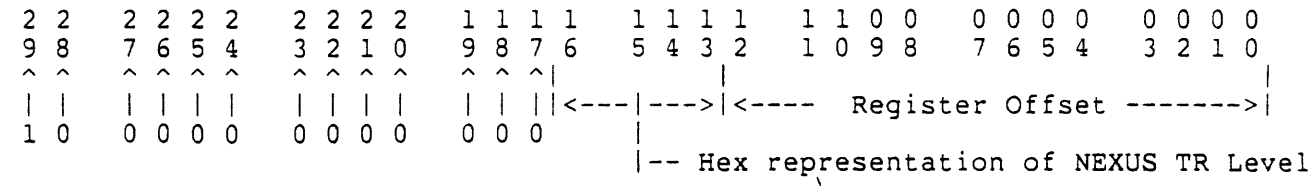
Breaking down PHYSICAL BYTE ADDRESSES



MEMORY ARRAY Physical Byte Addresses



NEXUS Physical Byte Addresses



UNIBUS Physical Byte Addresses

```
20100000:201FFFFF

2 2   2 2 2 2   2 2 2 2   1 1 1 1   1 1 1 1   1 1 0 0   0 0 0 0   0 0 0 0
9 8   7 6 5 4   3 2 1 0   9 8 7 6   5 4 3 2   1 0 9 8   7 6 5 4   3 2 1 0
^ ^   ^ ^ ^ ^   ^ ^ ^ ^   |   |<-- 18 Bit UNIBUS Address in HEX format -->|
| |   | | | |   | | | |   |<->|
1 0   0 0 0 0   0 0 0 1     |___ UNIBUS ADAPTER number
```

A maximum of 4 DW780 controllers, and therefore a maximum of 4 UNIBUSes, are supported on the VAX-11/780 system. Jumpers on the DW780 backplanes select the "UNIBUS Adapter #" for the associated UNIBUS.

RH780 External Register Physical Byte Addresses

```
200xx400:200xx7FC                          RH780          Drive number
xx = RH780 TR Level                      |<-TR # ->|       ____|____
                                         |         |      |         |
2 2   2 2 2 2   2 2 2 2   1 1 1 1   1 1 1 1   1 1 0 0   0 0 0 0   0 0 0 0
9 8   7 6 5 4   3 2 1 0   9 8 7 6   5 4 3 2   1 0 9 8   7 6 5 4   3 2 1 0
^ ^   ^ ^ ^ ^   ^ ^ ^ ^   ^ ^ ^           ^   ^ ^           |<---|--->|
| |   | | | |   | | | |   | | |           |   | |               |
1 0   0 0 0 0   0 0 0 0   0 0 0           0   0 1         Register number
```

Offsets (in bits <12:0>) from 400 thru 7FC are External Drive registers.

RH780 Internal Register Physical Byte Addresses

```
200xx000:200xx3FC                                          xx = RH780 TR Level
      and                              RH780
200xx800:200xxFC0                    |<-TR # ->|
                                     |         |
2 2   2 2 2 2   2 2 2 2   1 1 1 1   1 1 1 1   1 1 0 0   0 0 0 0   0 0 0 0
9 8   7 6 5 4   3 2 1 0   9 8 7 6   5 4 3 2   1 0 9 8   7 6 5 4   3 2 1 0
^ ^   ^ ^ ^ ^   ^ ^ ^ ^   ^ ^ ^           |<------------------------------>|
| |   | | | |   | | | |   | | |           | Register Byte offset.  Bits    |
1 0   0 0 0 0   0 0 0 0   0 0 0           |   <12:10> must not = 001.      |
```

Offsets of 000 thru 3FC are internal registers, except MAP registers.
Offsets of 800 thru FC0 are internal MAP registers.

Physical Byte Address breakdown procedure:

The 30 bit VAX Physical address spaces is broken into two equal parts. The upper 512 Megabytes of address space is called I/O space and is used to address NEXUS, UNIBUS, and MASSBUS registers. The lower 512 Megabytes is used to address Physical Memory Arrays. The address space is broken down as follows:

```
00000000 : 1FFFFFFF   =  Physical Memory array addresses
20000000 : 3FFFFFFF   =  Physical I/O register & I/O memory space.
```

The first step in breaking down a PHYSICAL BYTE ADDRESS is to determine if the address is an I/O or MEMORY ARRAY address. This is done by checking the state of PA bit <29>.

PA<29> = 1 indicates that the address is an I/O address.
PA<29> = 0 indicates that the address is a MEMORY ARRAY address.

PA bit <29> = 0 - Physical Memory Array address

The address is a Memory ARRAY address if PA<29>=0. The system configuration must be known in order to determine what array is being addressed. Further on in this section are several pages of charts and procedure that can be used to determine what addresses correspond to what physical array.

PA bit <29> = 1 - Physical I/O address

Use the following procedure to breakdown a Physical I/O address.

1. "PA Bit <29>" must be a 1 to indicate that the address is a VAX I/O address. If "PA Bit <29> = 0" the address is a Physical MEMORY address, and the remainder of this procedure cannot be used.

2. If PA Bit <29>=1 then PA Bits <28:21> must be zeros. If not, the address is illegal.

3. Check "PA Bit <20>" and proceed as indicated:

PA bit <20> = 1

If "PA Bit <20> = 1", the I/O address is in a "UNIBUS ADAPTER's" address region, and the following procedure can be used to find out what UNIBUS ADAPTER and UNIBUS DEVICE is being addressed:

a. "PA bits <19:18>" indicate the "UNIBUS ADAPTER number".

b. "PA bits <17:00>" contain the 18-bit UNIBUS DEVICE address.

NOTE: "UNIBUS ADAPTER" does not refer to which DW780 but indicates which "UNIBUS" is being referenced. In other words, the UNIBUS ADAPTER number is not an indication of how the "TR Level" jumpers are set for a DW780, but indicate how DW780 backplane jumpers "W1 & W2" are configured for the controlling DW780

PA bit <20> = 0

If "PA bit <20> = 0", the I/O address is a NEXUS Registers or MASSBUS Drive register address. In both cases, "PA<16:13>" will contain the "TR Level" of the NEXUS being addressed. The system configuration must be known in order to determine if the address is a NEXUS Register or MASSBUS Drive register address.

If PA Bit <29>=1 & PA Bit <20>=0, then PA Bit <17> and PA Bits <19:18> must be zero. If they do not, the address is illegal.

If "PA bits <16:13>" indicate an RH780 address, use the following procedure to determine if the address is for an RH780 NEXUS register or for an associated MASSBUS Drive register:

a. If "PA bits <12:10> do not = 1", then "PA<12:00>" contain the offset address for an RH780 (Internal) register.

b. If "PA bit <12:10> = 1", then "PA bits <9:7>" indicate the MASSBUS Drive addressed, and "PA bits <6:2>" indicate the register (EXTERNAL MASSBUS register) addressed.

I/O ADDRESS Ranges

NEXUS TR level	"LONGWORD" range	"BYTE" range
0 (see note)	8000000 - 80007FF	20000000 - 20001FFF
1	8000800 - 8000FFF	20002000 - 20003FFF
2	8001000 - 80017FF	20004000 - 20005FFF
3	8001800 - 8001FFF	20006000 - 20007FFF
4	8002000 - 80027FF	20008000 - 20009FFF
5	8002800 - 8002FFF	2000A000 - 2000BFFF
6	8003000 - 80037FF	2000C000 - 2000DFFF
7	8003800 - 8003FFF	2000E000 - 2000FFFF
8	8004000 - 80047FF	20010000 - 20011FFF
9	8004800 - 8004FFF	20012000 - 20013FFF
10	8005000 - 80057FF	20014000 - 20015FFF
11	8005800 - 8005FFF	20016000 - 20017FFF
12	8006000 - 80067FF	20018000 - 20019FFF
13	8006800 - 8006FFF	2001A000 - 2001BFFF
14	8007000 - 80077FF	2001C000 - 2001DFFF
15	8007800 - 8007FFF	2001E000 - 2001FFFF

Note: "TR#0 address space" is not assigned to any NEXUS. This is unused address space.

DW780 Adapter	NEXUS CODE	"Longword" Address ranges for UNIBUS devices	"BYTE" Address ranges for UNIBUS devices
0	28	8040000 - 804FFFF	20100000 - 2013FFFF
1	29	8050000 - 805FFFF	20140000 - 2017FFFF
2	2A	8060000 - 806FFFF	20180000 - 201BFFFF
3	2B	8070000 - 807FFFF	201C0000 - 201FFFFF

NOTE: Adapter numbers are assigned by DW780 backplane jumpers and do not have to correspond to any TR level scheme. The ADAPTER # simply indicates a particular UNIBUS.

Unused "LONGWORD" Address	Unused "BYTE" Address Ranges
8000000 - 80007FF	20000000 - 20001FFF
8008000 - 803FFFF	20020000 - 200FFFFF
8080000 - FFFFFFF	20200000 - 3FFFFFFF

Physical Memory Array Address range

"Longword" Address range	"Byte" Address range
0000000 - 7FFFFFF	00000000 - 1FFFFFFF

DW780 Register offsets

Reg. Name	Byte offset	Longword offset
CNFGR	000	000
UBACR	004	001
UBASR	008	002
DCR	00C	003
FMER	010	004
FUBAR	014	005
FMER	018	006
FUBAR	01C	007
BRSVR0	020	008
BRSVR1	024	009
BRSVR2	028	00A
BRSVR3	02C	00B
BRRVR4	030	00C
BRRVR5	034	00D
BRRVR6	038	00E
BRRVR7	03C	00F
DPR0	040	010
DPR1	044	011
DPR2	048	012
DPR3	04C	013
DPR4	050	014
DPR5	054	015
DPR6	058	016
DPR7	05C	017
DPR8	060	018
DPR9	064	019
DPR10	068	01A
DPR11	06C	01B
DPR12	070	01C
DPR13	074	01D
DPR14	078	01E
DPR15	07C	01F
resvd	080	020
.	.	.
.	.	.
resvd	7EC	1FF
MR0	800	200
MR1	804	201
MR3	808	202
MR4	80C	203
MR5	810	204
MR6	814	205
MR7	818	206
MR8	81C	207
MR9	820	208
MR10	824	209
MR11	828	20A
MR12	82C	20B

Reg. Name	Byte offset	Longword offset
MR13	830	20C
MR14	834	20D
MR15	838	20E
MR16	83C	20F
MR17	840	210
MR18	844	211
MR19	848	212
MR20	84C	213
MR22	850	214
MR23	854	215
MR24	858	216
MR25	85C	217
MR26	860	218
MR27	864	219
MR28	868	21A
MR29	86C	21B
MR30	870	21C
MR31	874	21D
MR32	878	21E
MR33	87C	21F
MR34	880	220
MR35	884	221
MR36	888	222
MR37	88C	223
.	.	.
.	.	.
.	.	.
.	.	.
MR480	F80	3E0
MR481	F84	3E1
MR482	F88	3E2
MR483	F8C	3E3
MR484	F90	3E4
MR485	F94	3E5
MR486	F98	3E6
MR487	F9C	3E7
MR488	FA0	3E8
MR489	FA4	3E9
MR490	FA8	3EA
MR491	FAC	3EB
MR492	FB0	3EC
MR493	FB4	3ED
MR494	FB8	3EE
MR495	FBC	3EF
resvd	FC0	3F0
.	.	.
.	.	.
resvd	FFC	3FF

RH780 Internal Register offsets

Reg. Name	Byte offset	Longword offset
CNFGR	000	000
MBACR	004	001
MBASR	008	002
MBAVAR	00C	003
MBABCR	010	004
MBADR	014	005
MBASMR	018	006
MBACAR	01C	007
resvd	.	.
(and MASSBUS Registers)		
resvd	.	.
MR0	800	200
MR1	804	201
MR1	808	202
MR2	80C	203
MR3	810	204
	.	.
	.	.
	.	.
MR37	88C	223
MR38	890	224
MR39	894	225
MR40	898	226
MR41	89C	227
MR42	8A0	228
MR43	8A4	229
MR44	8A8	22A
MR45	8AC	22B
MR46	8B0	22C
MR47	8B4	22D
MR48	8B8	22E
MR49	8BC	22F
MR50	8C0	230
MR51	8C4	231
MR52	8C8	232
MR53	8CC	233
MR54	8D0	234
MR55	8D4	235
MR56	8D8	236
MR57	8DC	237
MR58	8E0	238
MR59	8E4	239
MR60	8E8	23A
MR61	8EC	23B
MR62	8F0	23C
MR63	8F4	23D
MR64	8F8	23E

Reg. Name	Byte offset	Longword offset
MR65	8FC	23F
MR66	900	240
MR67	904	241
MR68	908	242
MR69	90C	243
MR70	910	244
MR71	914	245
MR72	918	246
MR73	91C	247
MR74	920	248
MR75	924	249
MR76	928	24A
MR77	92C	24B
MR78	930	24C
MR79	934	24D
MR80	938	24E
MR81	93C	24F
MR82	940	250
MR83	944	251
MR84	948	252
MR85	94C	253
	.	.
	.	.
	.	.
	.	.
MR467	F4C	3D3
MR468	F50	3D4
MR469	F54	3D5
MR470	F58	3D6
MR471	F5C	3D7
MR472	F60	3D8
MR473	F64	3D9
MR474	F68	3DA
MR475	F6C	3DB
MR476	F70	3DC
MR477	F74	3DD
MR478	F78	3DE
MR479	F7C	3DF
MR480	F80	3E0
	.	.
	.	.
	.	.
MR491	FB0	3EC
MR492	FB4	3ED
MR493	FB8	3EE
MR494	FBC	3EF
MR495	FC0	3F0

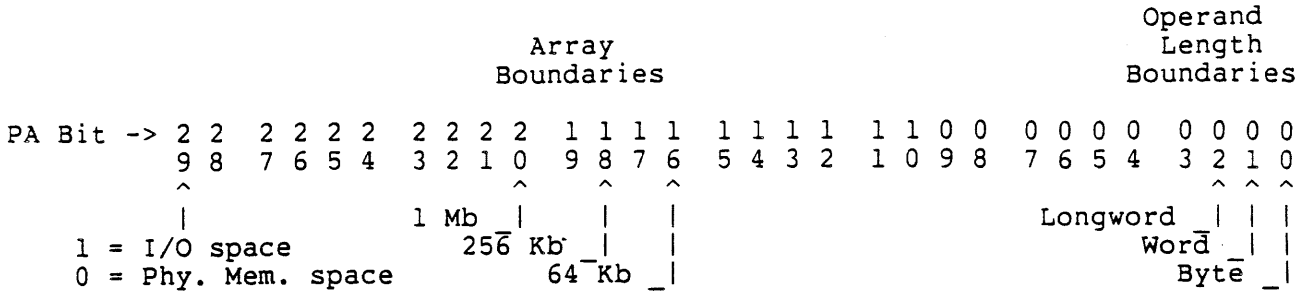
If PA<10>=1 then the register is a "MASSBUS" (external) register.
The "MASSBUS Register Offsets" are on the next page.

RH780 MASSBUS (EXTERNAL) Register offsets

Reg.	"Offsets for MASSBUS register of Drives 0 thru 7"							
No.	#0	#1	#2	#3	#4	#5	#6	#7
0	0	80	100	180	200	280	300	380
1	4	84	104	184	204	284	304	384
2	8	88	108	188	208	288	308	388
3	C	8C	100	18C	20C	28C	30C	38C
4	10	90	100	190	210	290	310	390
5	14	94	100	194	214	294	314	394
6	18	98	100	198	218	298	318	398
7	1C	9C	100	19C	21C	29C	31C	39C
8	20	A0	100	1A0	220	2A0	320	3A0
9	24	A4	100	1A4	224	2A4	324	3A4
A	28	A8	100	1A8	228	2A8	328	3A8
B	2C	AC	100	1AC	22C	2AC	32C	3AC
C	30	B0	100	1B0	230	2B0	330	3B0
D	34	B4	100	1B4	234	2B4	334	3B4
E	38	B8	100	1B8	238	2B8	338	3B8
F	3C	BC	100	1BC	23C	2BC	33C	3BC
10	40	C0	100	1C0	240	2C0	340	3C0
11	44	C4	100	1C4	244	2C4	344	3C4
12	48	C8	100	1C8	248	2C8	348	3C8
13	4C	CC	100	1CC	24C	2CC	34C	3CC
14	50	D0	100	1D0	250	2D0	350	3D0
15	54	D4	100	1D4	254	2D4	354	3D4
16	58	D8	100	1D8	258	2D8	358	3D8
17	5C	DC	100	1DC	25C	2DC	35C	3DC
18	60	E0	100	1E0	260	2E0	360	3E0
19	64	E4	100	1E4	264	2E4	364	3E4
1A	68	E8	100	1E8	268	2E8	368	3E8
1B	6C	EC	100	1EC	26C	2EC	36C	3EC
1C	70	F0	100	1F0	270	2F0	370	3F0
1D	74	F4	100	1F4	274	2F4	374	3F4
1E	78	F8	100	1F8	278	2F8	378	3F8
1F	7C	FC	100	1FC	27C	2FC	37C	3FC

Reg.#	RP0x	RM0x	TE16
0	CS1	RMCS1	CS1
1	DS	RMDS	DS
2	ER1	RMER1	ER
3	MR	RMMR1	MR
4	AS	RMAS	AS
5	DA	RMDA	FC
6	DT	RMDT	DT
7	LA	RMLA	CX
8	SN	RMSN	SN
9	OFF	RMOF	TC
A	DCA	RMOC	
B	CCA	RONR	
C	ER2	RMMR2	
D	ER3	RMER2	
E	ECCPOS	RMEC1	
F	ECCPAT	RMEC2	

Memory Array Address Bit breakdown



<28:16> = 64KB array number
<28:18> = 256KB array number
<28:20> = 1MegaByte array number

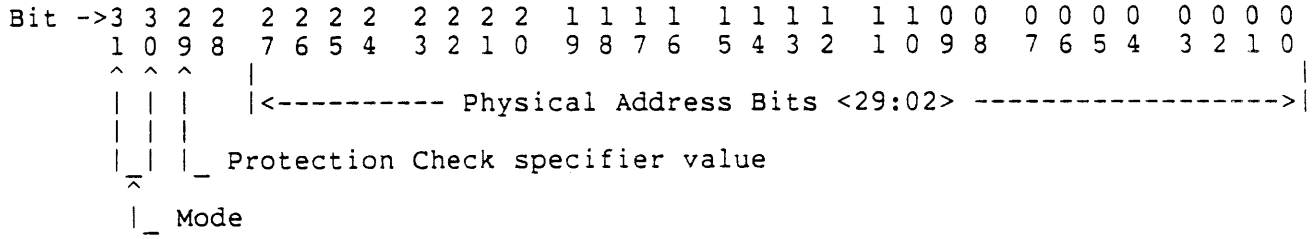
Physical Address bits 0 & 1 are not used on the SBI. All addresses to Nexus devices are Longword addresses, (only Physical Address bits <29:02> are used on the SBI).

The "SBI MASK" field is used to specify which byte(s) are to be referenced within a Longword.

There is a total of 1 GigaByte of Physical address space. This space is broken up into two equal sections.

512 Megabyte of Physical MEMORY Address Space,
512 Megabyte of Physical I/O Address Space

"Timeout Address" ID register bit breakdown (ID #1A)



Physical "BYTE" Address Space - *64KB boundries*

000.00 Megabyte	002.75 Megabyte	005.50 Megabyte
00000000 - 0000FFFF	002C0000 - 002CFFFF	00580000 - 0058FFFF
00010000 - 0001FFFF	002D0000 - 002DFFFF	00590000 - 0059FFFF
00020000 - 0002FFFF	002E0000 - 002EFFFF	005A0000 - 005AFFFF
00030000 - 0003FFFF	002F0000 - 002FFFFF	005B0000 - 005BFFFF
000.25 Megabyte	003.00 Megabyte	005.75 Megabyte
00040000 - 0004FFFF	00300000 - 0030FFFF	005C0000 - 005CFFFF
00050000 - 0005FFFF	00310000 - 0031FFFF	005D0000 - 005DFFFF
00060000 - 0006FFFF	00320000 - 0032FFFF	005E0000 - 005EFFFF
00070000 - 0007FFFF	00330000 - 0033FFFF	005F0000 - 005FFFFF
000.50 Megabyte	003.25 Megabyte	006.00 Megabyte
00080000 - 0008FFFF	00340000 - 0034FFFF	00600000 - 0060FFFF
00090000 - 0009FFFF	00350000 - 0035FFFF	00610000 - 0061FFFF
000A0000 - 000AFFFF	00360000 - 0036FFFF	00620000 - 0062FFFF
000B0000 - 000BFFFF	00370000 - 0037FFFF	00630000 - 0063FFFF
000.75 Megabyte	003.50 Megabyte	006.25 Megabyte
000C0000 - 000CFFFF	00380000 - 0038FFFF	00640000 - 0064FFFF
000D0000 - 000DFFFF	00390000 - 0039FFFF	00650000 - 0065FFFF
000E0000 - 000EFFFF	003A0000 - 003AFFFF	00660000 - 0066FFFF
000F0000 - 000FFFFF	003B0000 - 003BFFFF	00670000 - 0067FFFF
001.00 Megabyte	003.75 Megabyte	006.50 Megabyte
00100000 - 0010FFFF	003C0000 - 003CFFFF	00680000 - 0068FFFF
00110000 - 0011FFFF	003D0000 - 003DFFFF	00690000 - 0069FFFF
00120000 - 0012FFFF	003E0000 - 003EFFFF	006A0000 - 006AFFFF
00130000 - 0013FFFF	003F0000 - 003FFFFF	006B0000 - 006BFFFF
001.25 Megabyte	004.00 Megabyte	006.75 Megabyte
00140000 - 0014FFFF	00400000 - 0040FFFF	006C0000 - 006CFFFF
00150000 - 0015FFFF	00410000 - 0041FFFF	006D0000 - 006DFFFF
00160000 - 0016FFFF	00420000 - 0042FFFF	006E0000 - 006EFFFF
00170000 - 0017FFFF	00430000 - 0043FFFF	006F0000 - 006FFFFF
001.50 Megabyte	004.25 Megabyte	007.00 Megabyte
00180000 - 0018FFFF	00440000 - 0044FFFF	00700000 - 0070FFFF
00190000 - 0019FFFF	00450000 - 0045FFFF	00710000 - 0071FFFF
001A0000 - 001AFFFF	00460000 - 0046FFFF	00720000 - 0072FFFF
001B0000 - 001BFFFF	00470000 - 0047FFFF	00730000 - 0073FFFF
001.75 Megabyte	004.50 Megabyte	007.25 Megabyte
001C0000 - 001CFFFF	00480000 - 0048FFFF	00740000 - 0074FFFF
001D0000 - 001DFFFF	00490000 - 0049FFFF	00750000 - 0075FFFF
001E0000 - 001EFFFF	004A0000 - 004AFFFF	00760000 - 0076FFFF
001F0000 - 001FFFFF	004B0000 - 004BFFFF	00770000 - 0077FFFF
002.00 Megabyte	004.75 Megabyte	007.50 Megabyte
00200000 - 0020FFFF	004C0000 - 004CFFFF	00780000 - 0078FFFF
00210000 - 0021FFFF	004D0000 - 004DFFFF	00790000 - 0079FFFF
00220000 - 0022FFFF	004E0000 - 004EFFFF	007A0000 - 007AFFFF
00230000 - 0023FFFF	004F0000 - 004FFFFF	007B0000 - 007BFFFF
002.25 Megabyte	005.00 Megabyte	007.75 Megabyte
00240000 - 0024FFFF	00500000 - 0050FFFF	007C0000 - 007CFFFF
00250000 - 0025FFFF	00510000 - 0051FFFF	007D0000 - 007DFFFF
00260000 - 0026FFFF	00520000 - 0052FFFF	007E0000 - 007EFFFF
00270000 - 0027FFFF	00530000 - 0053FFFF	007F0000 - 007FFFFF
002.50 Megabyte	005.25 Megabyte	008.00 Megabyte
00280000 - 0028FFFF	00540000 - 0054FFFF	00800000 - 0080FFFF
00290000 - 0029FFFF	00550000 - 0055FFFF	00810000 - 0081FFFF
002A0000 - 002AFFFF	00560000 - 0056FFFF	00820000 - 0082FFFF
002B0000 - 002BFFFF	00570000 - 0057FFFF	00830000 - 0083FFFF

Physical "BYTE" Address Space - *64KB boundries*

008.25 Megabyte
00840000 - 0084FFFF
00850000 - 0085FFFF
00860000 - 0086FFFF
00870000 - 0087FFFF

008.50Megabyte
00880000 - 0088FFFF
00890000 - 0089FFFF
008A0000 - 008AFFFF
008B0000 - 008BFFFF

008.75 Megabyte
008C0000 - 008CFFFF
008D0000 - 008DFFFF
008E0000 - 008EFFFF
008F0000 - 008FFFFF

009.00 Megabyte
00900000 - 0090FFFF
00910000 - 0091FFFF
00920000 - 0092FFFF
00930000 - 0093FFFF

009.25 Megabyte
00940000 - 0094FFFF
00950000 - 0095FFFF
00960000 - 0096FFFF
00970000 - 0097FFFF

009.50 Megabyte
00980000 - 0098FFFF
00990000 - 0099FFFF
009A0000 - 009AFFFF
009B0000 - 009BFFFF

009.75 Megabyte
009C0000 - 009CFFFF
009D0000 - 009DFFFF
009E0000 - 009EFFFF
009F0000 - 009FFFFF

010.00 Megabyte
00A00000 - 00A0FFFF
00A10000 - 00A1FFFF
00A20000 - 00A2FFFF
00A30000 - 00A3FFFF

010.25 Megabyte
00A40000 - 00A4FFFF
00A50000 - 00A5FFFF
00A60000 - 00A6FFFF
00A70000 - 00A7FFFF

010.50 Megabyte
00A80000 - 00A8FFFF
00A90000 - 00A9FFFF
00AA0000 - 00AAFFFF
00AB0000 - 00ABFFFF

010.75 Megabyte
00AC0000 - 00ACFFFF
00AD0000 - 00ADFFFF
00AE0000 - 00AEFFFF
00AF0000 - 00AFFFFF

011.00 Megabyte
00B00000 - 00B0FFFF
00B10000 - 00B1FFFF
00B20000 - 00B2FFFF
00B30000 - 00B3FFFF

011.25 Megabyte
00B40000 - 00B4FFFF
00B50000 - 00B5FFFF
00B60000 - 00B6FFFF
00B70000 - 00B7FFFF

011.50 Megabyte
00B80000 - 00B8FFFF
00B90000 - 00B9FFFF
00BA0000 - 00BAFFFF
00BB0000 - 00BBFFFF

011.75 Megabyte
00BC0000 - 00BCFFFF
00BD0000 - 00BDFFFF
00BE0000 - 00BEFFFF
00BF0000 - 00BFFFFF

012.00 Megabyte
00C00000 - 00C0FFFF
00C10000 - 00C1FFFF
00C20000 - 00C2FFFF
00C30000 - 00C3FFFF

012.25 Megabyte
00C40000 - 00C4FFFF
00C50000 - 00C5FFFF
00C60000 - 00C6FFFF
00C70000 - 00C7FFFF

012.50 Megabyte
00C80000 - 00C8FFFF
00C90000 - 00C9FFFF
00CA0000 - 00CAFFFF
00CB0000 - 00CBFFFF

012.75 Megabyte
00CC0000 - 00CCFFFF
00CD0000 - 00CDFFFF
00CE0000 - 00CEFFFF
00CF0000 - 00CFFFFF

013.00 Megabyte
00D00000 - 00D0FFFF
00D10000 - 00D1FFFF
00D20000 - 00D2FFFF
00D30000 - 00D3FFFF

013.25 Megabyte
00D40000 - 00D4FFFF
00D50000 - 00D5FFFF
00D60000 - 00D6FFFF
00D70000 - 00D7FFFF

013.50 Megabyte
00D80000 - 00D8FFFF
00D90000 - 00D9FFFF
00DA0000 - 00DAFFFF
00DB0000 - 00DBFFFF

013.75 Megabyte
00DC0000 - 00DCFFFF
00DD0000 - 00DDFFFF
00DE0000 - 00DEFFFF
00DF0000 - 00DFFFFF

014.00 Megabyte
00E00000 - 00E0FFFF
00E10000 - 00E1FFFF
00E20000 - 00E2FFFF
00E30000 - 00E3FFFF

014.25 Megabyte
00E40000 - 00E4FFFF
00E50000 - 00E5FFFF
00E60000 - 00E6FFFF
00E70000 - 00E7FFFF

014.50 Megabyte
00E80000 - 00E8FFFF
00E90000 - 00E9FFFF
00EA0000 - 00EAFFFF
00EB0000 - 00EBFFFF

014.75 Megabyte
00EC0000 - 00ECFFFF
00ED0000 - 00EDFFFF
00EE0000 - 00EEFFFF
00EF0000 - 00EFFFFF

015.00 Megabyte
00F00000 - 00F0FFFF
00F10000 - 00F1FFFF
00F20000 - 00F2FFFF
00F30000 - 00F3FFFF

015.25 Megabyte
00F40000 - 00F4FFFF
00F50000 - 00F5FFFF
00F60000 - 00F6FFFF
00F70000 - 00F7FFFF

015.50 Megabyte
00F80000 - 00F8FFFF
00F90000 - 00F9FFFF
00FA0000 - 00FAFFFF
00FB0000 - 00FBFFFF

015.75 Megabyte
00FC0000 - 00FCFFFF
00FD0000 - 00FDFFFF
00FE0000 - 00FEFFFF
00FF0000 - 00FFFFFF

016.00 Megabyte
01000000 - 0100FFFF
01010000 - 0101FFFF
01020000 - 0102FFFF
01030000 - 0103FFFF

016.25 Megabyte
01040000 - 0104FFFF
01050000 - 0105FFFF
01060000 - 0106FFFF
01070000 - 0107FFFF

Physical "BYTE" Address Space - *256KB boundries*

016.0 Megabyte	026.0 Megabyte	036.0 Megabyte
01000000 - 0103FFFF	01A00000 - 01A3FFFF	02400000 - 0243FFFF
01040000 - 0107FFFF	01A40000 - 01A7FFFF	02440000 - 0247FFFF
01080000 - 010BFFFF	01A80000 - 01ABFFFF	02480000 - 024BFFFF
010C0000 - 010FFFFF	01AC0000 - 01AFFFFF	024C0000 - 024FFFFF
017.0 Megabyte	027.0 Megabyte	037.0 Megabyte
01100000 - 0113FFFF	01B00000 - 01B3FFFF	02500000 - 0253FFFF
01140000 - 0117FFFF	01B40000 - 01B7FFFF	02540000 - 0257FFFF
01180000 - 011BFFFF	01B80000 - 01BBFFFF	02580000 - 025BFFFF
011C0000 - 011FFFFF	01BC0000 - 01BFFFFF	025C0000 - 025FFFFF
018.0 Megabyte	028.0 Megabyte	038.0 Megabyte
01200000 - 0123FFFF	01C00000 - 01C3FFFF	02600000 - 0263FFFF
01240000 - 0127FFFF	01C40000 - 01C7FFFF	02640000 - 0267FFFF
01280000 - 012BFFFF	01C80000 - 01CBFFFF	02680000 - 026BFFFF
012C0000 - 012FFFFF	01CC0000 - 01CFFFFF	026C0000 - 026FFFFF
019.0 Megabyte	029.0 Megabyte	039.0 Megabyte
01300000 - 0133FFFF	01D00000 - 01D3FFFF	02700000 - 0273FFFF
01340000 - 0137FFFF	01D40000 - 01D7FFFF	02740000 - 0277FFFF
01380000 - 013BFFFF	01D80000 - 01DBFFFF	02780000 - 027BFFFF
013C0000 - 013FFFFF	01DC0000 - 01DFFFFF	027C0000 - 027F0000
020.0 Megabyte	030.0 Megabyte	040.0 Megatyte
01400000 - 0143FFFF	01E00000 - 01E3FFFF	02800000 - 0283FFFF
01440000 - 0147FFFF	01E40000 - 01E7FFFF	02840000 - 0287FFFF
01480000 - 014BFFFF	01E80000 - 01EBFFFF	02880000 - 028BFFFF
014C0000 - 014FFFFF	01EC0000 - 01EFFFFF	028C0000 - 028FFFFF
021.0 Megabyte	031.0 Megabyte	041.0 Megabyte
01500000 - 0153FFFF	01F00000 - 01F3FFFF	02900000 - 0293FFFF
01540000 - 0157FFFF	01F40000 - 01F3FFFF	02940000 - 0297FFFF
01580000 - 015BFFFF	01F80000 - 01FBFFFF	02980000 - 029BFFFF
015C0000 - 015FFFFF	01FC0000 - 01FFFFFF	029C0000 - 029FFFFF
022.0 Megabyte	032.0 Megabyte	042.0 Megabyte
01600000 - 0163FFFF	02000000 - 0203FFFF	02A00000 - 02A3FFFF
01640000 - 0167FFFF	02040000 - 0207FFFF	02A40000 - 02A7FFFF
01680000 - 016BFFFF	02080000 - 020BFFFF	02A80000 - 02ABFFFF
016C0000 - 016FFFFF	020C0000 - 020FFFFF	02AC0000 - 02AFFFFF
023.0 Megabyte	033.0 Megabyte	043.0 Megabyte
01700000 - 0173FFFF	02100000 - 0213FFFF	02B00000 - 02B3FFFF
01740000 - 0177FFFF	02140000 - 0217FFFF	02B40000 - 02B7FFFF
01780000 - 017BFFFF	02180000 - 021BFFFF	02B80000 - 02BBFFFF
017C0000 - 017FFFFF	021C0000 - 021FFFFF	02BC0000 - 02BFFFFF
024.0 Megabyte	034.0 Megabyte	044.0 Megabyte
01800000 - 0183FFFF	02200000 - 0223FFFF	02C00000 - 02C3FFFF
01840000 - 0187FFFF	02240000 - 0227FFFF	02C40000 - 02C7FFFF
01880000 - 018BFFFF	02280000 - 022BFFFF	02C80000 - 02CBFFFF
018C0000 - 018FFFFF	022C0000 - 022FFFFF	02CC0000 - 02CFFFFF
025.0 Megabyte	035.0 Megabyte	045.0 Megabyte
01900000 - 0193FFFF	02300000 - 0233FFFF	02D00000 - 02D3FFFF
01940000 - 0197FFFF	02340000 - 0237FFFF	02D40000 - 02D7FFFF
01980000 - 019BFFFF	02380000 - 023BFFFF	02D80000 - 02DBFFFF
019C0000 - 019FFFFF	023C0000 - 023FFFFF	02DC0000 - 02DFFFFF

Physical "BYTE" Address Space - *256KB boundries*

046.0 Megabyte
02E00000 - 02E3FFFF
02E40000 - 02E7FFFF
02E80000 - 02EBFFFF
02EC0000 - 02EFFFFF

047.0 Megabyte
02F00000 - 02F3FFFF
02F40000 - 02F7FFFF
02F80000 - 02FBFFFF
02FC0000 - 02FFFFFF

048.0 Megabyte
03000000 - 0303FFFF
03040000 - 0307FFFF
03080000 - 030BFFFF
030C0000 - 030FFFFF

049.0 Megabyte
03100000 - 0313FFFF
03140000 - 0317FFFF
03180000 - 031BFFFF
031C0000 - 031FFFFF

050.0 Megabyte
03200000 - 0323FFFF
03240000 - 0327FFFF
03280000 - 032BFFFF
032C0000 - 032FFFFF

051.0 Megabyte
03300000 - 0333FFFF
03340000 - 0337FFFF
03380000 - 033BFFFF
033C0000 - 033FFFFF

052.0 Megabyte
03400000 - 0343FFFF
03440000 - 0347FFFF
03480000 - 034BFFFF
034C0000 - 034FFFFF

053.0 Megabyte
03500000 - 0353FFFF
03540000 - 0357FFFF
03580000 - 035BFFFF
035C0000 - 035FFFFF

054.0 Megabyte
03600000 - 0363FFFF
03640000 - 0367FFFF
03680000 - 036BFFFF
036C0000 - 036FFFFF

055.0 Megabyte
03700000 - 0373FFFF
03740000 - 0377FFFF
03780000 - 037BFFFF
037C0000 - 037FFFFF

056.0 Megabyte
03800000 - 0383FFFF
03840000 - 0387FFFF
03880000 - 038BFFFF
038C0000 - 038FFFFF

057.0 Megabyte
03900000 - 0393FFFF
03940000 - 0397FFFF
03980000 - 039BFFFF
039C0000 - 039FFFFF

058.0 Megabyte
03A00000 - 03A3FFFF
03A40000 - 03A7FFFF
03A80000 - 03ABFFFF
03AC0000 - 03AFFFFF

059.0 Megabyte
03B00000 - 03B3FFFF
03B40000 - 03B7FFFF
03B80000 - 03BBFFFF
03BC0000 - 03BFFFFF

060.0 Megabyte
03C00000 - 03C3FFFF
03C40000 - 03C7FFFF
03C80000 - 03CBFFFF
03CC0000 - 03CFFFFF

061.0 Megabyte
03D00000 - 03D3FFFF
03D40000 - 03D7FFFF
03D80000 - 03DBFFFF
03DC0000 - 03DFFFFF

062.0 Megabyte
03E00000 - 03E3FFFF
03E40000 - 03E7FFFF
03E80000 - 03EBFFFF
03EC0000 - 03EFFFFF

063.0 Megabyte
03F00000 - 03F3FFFF
03F40000 - 03F7FFFF
03F80000 - 03FBFFFF
03FC0000 - 03FFFFFF

064.0 Megabyte
04000000 - 0403FFFF
04040000 - 0407FFFF
04080000 - 040BFFFF
040C0000 - 040FFFFF

Physical "BYTE" Address Space - *1MB boundries*

000 Megabyte	045 Megabyte	090 Megabyte
00000000 - 000FFFFF	02D00000 - 02DFFFFF	05A00000 - 05AFFFFF
00100000 - 001FFFFF	02E00000 - 02EFFFFF	05B00000 - 05BFFFFF
00200000 - 002FFFFF	02F00000 - 02FFFFFF	05C00000 - 05CFFFFF
00300000 - 003FFFFF	03000000 - 030FFFFF	05D00000 - 05DFFFFF
00400000 - 004FFFFF	03100000 - 031FFFFF	05E00000 - 05EFFFFF
005 Megabyte	050 Megabyte	095 Megabyte
00500000 - 005FFFFF	03200000 - 032FFFFF	05F00000 - 05FFFFFF
00600000 - 006FFFFF	03300000 - 033FFFFF	06000000 - 060FFFFF
00700000 - 007FFFFF	03400000 - 034FFFFF	06100000 - 061FFFFF
00800000 - 008FFFFF	03500000 - 035FFFFF	06200000 - 062FFFFF
00900000 - 009FFFFF	03600000 - 036FFFFF	06300000 - 063FFFFF
010 Megabyte	055 Megabyte	100 Megabyte
00A00000 - 00AFFFFF	03700000 - 037FFFFF	06400000 - 064FFFFF
00B00000 - 00BFFFFF	03800000 - 038FFFFF	06500000 - 065FFFFF
00C00000 - 00CFFFFF	03900000 - 039FFFFF	06600000 - 066FFFFF
00D00000 - 00DFFFFF	03A00000 - 03AFFFFF	06700000 - 067FFFFF
00E00000 - 00EFFFFF	03B00000 - 03BFFFFF	06800000 - 068FFFFF
015 Megabyte	060 Megabyte	105 Megabyte
00F00000 - 00FFFFFF	03C00000 - 03CFFFFF	06900000 - 069FFFFF
01000000 - 010FFFFF	03D00000 - 03DFFFFF	06A00000 - 06AFFFFF
01100000 - 011FFFFF	03E00000 - 03EFFFFF	06B00000 - 06BFFFFF
01200000 - 012FFFFF	03F00000 - 03FFFFFF	06C00000 - 06CFFFFF
01300000 - 013FFFFF	04000000 - 040FFFFF	06D00000 - 06DFFFFF
020 Megabyte	065 Megabyte	110 Megabyte
01400000 - 014FFFFF	04100000 - 041FFFFF	06E00000 - 06EFFFFF
01500000 - 015FFFFF	04200000 - 042FFFFF	06F00000 - 06FFFFFF
01600000 - 016FFFFF	04300000 - 043FFFFF	07000000 - 070FFFFF
01700000 - 017FFFFF	04400000 - 044FFFFF	07100000 - 071FFFFF
01800000 - 018FFFFF	04500000 - 045FFFFF	07200000 - 072FFFFF
025 Megabyte	070 Megabyte	115 Megabyte
01900000 - 019FFFFF	04600000 - 046FFFFF	07300000 - 073FFFFF
01A00000 - 01AFFFFF	04700000 - 047FFFFF	07400000 - 074FFFFF
01B00000 - 01BFFFFF	04800000 - 048FFFFF	07500000 - 075FFFFF
01C00000 - 01CFFFFF	04900000 - 049FFFFF	07600000 - 076FFFFF
01D00000 - 01DFFFFF	04A00000 - 04AFFFFF	07700000 - 077FFFFF
030 Megabyte	075 Megabyte	120 Megabyte
01E00000 - 01EFFFFF	04B00000 - 04BFFFFF	07800000 - 078FFFFF
01F00000 - 01FFFFFF	04C00000 - 04CFFFFF	07900000 - 079FFFFF
02000000 - 020FFFFF	04D00000 - 04DFFFFF	07A00000 - 07AFFFFF
02100000 - 021FFFFF	04E00000 - 04EFFFFF	07B00000 - 07BFFFFF
02200000 - 022FFFFF	04F00000 - 04FFFFFF	07C00000 - 07CFFFFF
035 Megabyte	080 Megabyte	125 Megabyte
02300000 - 023FFFFF	05000000 - 050FFFFF	07D00000 - 07DFFFFF
02400000 - 024FFFFF	05100000 - 051FFFFF	07E00000 - 07EFFFFF
02500000 - 025FFFFF	05200000 - 052FFFFF	07F00000 - 07FFFFFF
02600000 - 026FFFFF	05300000 - 053FFFFF	08000000 - 080FFFFF
02700000 - 027FFFFF	05400000 - 054FFFFF	08100000 - 081FFFFF
040 Megabyte	085 Megabyte	130 Megabyte
02800000 - 028FFFFF	05500000 - 055FFFFF	08200000 - 082FFFFF
02900000 - 029FFFFF	05600000 - 056FFFFF	08300000 - 083FFFFF
02A00000 - 02AFFFFF	05700000 - 057FFFFF	08400000 - 084FFFFF
02B00000 - 02BFFFFF	05800000 - 058FFFFF	08500000 - 085FFFFF
02C00000 - 02CFFFFF	05900000 - 059FFFFF	08600000 - 086FFFFF

Physical "BYTE" Address Space - *1MB boundries*

135 Megabyte	180 Megabyte	225 Megabyte
08700000 - 087FFFFF	0B400000 - 0B4FFFFF	0E100000 - 0E1FFFFF
08800000 - 088FFFFF	0B500000 - 0B5FFFFF	0E200000 - 0E2FFFFF
08900000 - 089FFFFF	0B600000 - 0B6FFFFF	0E300000 - 0E3FFFFF
08A00000 - 08AFFFFF	0B700000 - 0B7FFFFF	0E400000 - 0E4FFFFF
08B00000 - 08BFFFFF	0B800000 - 0B8FFFFF	0E500000 - 0E5FFFFF
140 Megabyte	185 Megabyte	230 Megabyte
08C00000 - 08CFFFFF	0B900000 - 0B9FFFFF	0E600000 - 0E6FFFFF
08D00000 - 08DFFFFF	0BA00000 - 0BAFFFFF	0E700000 - 0E7FFFFF
08E00000 - 08EFFFFF	0BB00000 - 0BBFFFFF	0E800000 - 0E8FFFFF
08F00000 - 08FFFFFF	0BC00000 - 0BCFFFFF	0E900000 - 0E9FFFFF
09000000 - 090FFFFF	0BD00000 - 0BDFFFFF	0EA00000 - 0EAFFFFF
145 Megabyte	190 Megabyte	235 Megabyte
09100000 - 091FFFFF	0BE00000 - 0BEFFFFF	0EB00000 - 0EBFFFFF
09200000 - 092FFFFF	0BF00000 - 0BFFFFFF	0EC00000 - 0ECFFFFF
09300000 - 093FFFFF	0C000000 - 0C0FFFFF	0ED00000 - 0EDFFFFF
09400000 - 094FFFFF	0C100000 - 0C1FFFFF	0EE00000 - 0EEFFFFF
09500000 - 095FFFFF	0C200000 - 0C2FFFFF	0EF00000 - 0EFFFFFF
150 Megabyte	195 Megabyte	240 Megabyte
09600000 - 096FFFFF	0C300000 - 0C3FFFFF	0F000000 - 0F0FFFFF
09700000 - 097FFFFF	0C400000 - 0C4FFFFF	0F100000 - 0F1FFFFF
09800000 - 098FFFFF	0C500000 - 0C5FFFFF	0F200000 - 0F2FFFFF
09900000 - 099FFFFF	0C600000 - 0C6FFFFF	0F300000 - 0F3FFFFF
09A00000 - 09AFFFFF	0C700000 - 0C7FFFFF	0F400000 - 0F4FFFFF
155 Megabyte	200 Megabyte	245 Megabyte
09B00000 - 09BFFFFF	0C800000 - 0C8FFFFF	0F500000 - 0F5FFFFF
09C00000 - 09CFFFFF	0C900000 - 0C9FFFFF	0F600000 - 0F6FFFFF
09D00000 - 09DFFFFF	0CA00000 - 0CAFFFFF	0F700000 - 0F7FFFFF
09E00000 - 09EFFFFF	0CB00000 - 0CBFFFFF	0F800000 - 0F8FFFFF
09F00000 - 09FFFFFF	0CC00000 - 0CCFFFFF	0F900000 - 0F9FFFFF
160 Megabyte	205 Megabyte	250 Megabyte
0A000000 - 0A0FFFFF	0CD00000 - 0CDFFFFF	0FA00000 - 0FAFFFFF
0A100000 - 0A1FFFFF	0CE00000 - 0CEFFFFF	0FB00000 - 0FBFFFFF
0A200000 - 0A2FFFFF	0CF00000 - 0CFFFFFF	0FC00000 - 0FCFFFFF
0A300000 - 0A3FFFFF	0D000000 - 0D0FFFFF	0FD00000 - 0FDFFFFF
0A400000 - 0A4FFFFF	0D100000 - 0D1FFFFF	0FE00000 - 0FEFFFFF
165 Megabyte	210 Megabyte	255 Megabyte
0A500000 - 0A5FFFFF	0D200000 - 0D2FFFFF	0FF00000 - 0FFFFFFF
0A600000 - 0A6FFFFF	0D300000 - 0D3FFFFF	10000000 - 100FFFFF
0A700000 - 0A7FFFFF	0D400000 - 0D4FFFFF	10100000 - 101FFFFF
0A800000 - 0A8FFFFF	0D500000 - 0D5FFFFF	10200000 - 102FFFFF
0A900000 - 0A9FFFFF	0D600000 - 0D6FFFFF	10300000 - 103FFFFF
170 Megabyte	215 Megabyte	260 Megabyte
0AA00000 - 0AAFFFFF	0D700000 - 0D7FFFFF	10400000 - 104FFFFF
0AB00000 - 0ABFFFFF	0D800000 - 0D8FFFFF	10500000 - 105FFFFF
0AC00000 - 0ACFFFFF	0D900000 - 0D9FFFFF	10600000 - 106FFFFF
0AD00000 - 0ADFFFFF	0DA00000 - 0DAFFFFF	10700000 - 107FFFFF
0AE00000 - 0AEFFFFF	0DB00000 - 0DBFFFFF	10800000 - 108FFFFF
175 Megabyte	220 Megabyte	265 Megabyte
0AF00000 - 0AFFFFFF	0DC00000 - 0DCFFFFF	10900000 - 109FFFFF
0B000000 - 0B0FFFFF	0DD00000 - 0DDFFFFF	10A00000 - 10AFFFFF
0B100000 - 0B1FFFFF	0DE00000 - 0DEFFFFF	10B00000 - 10BFFFFF
0B200000 - 0B2FFFFF	0DF00000 - 0DFFFFFF	10C00000 - 10CFFFFF
0B300000 - 0B3FFFFF	0E000000 - 0E0FFFFF	10D00000 - 10DFFFFF

Physical "BYTE" Address Space - *1MB boundries*

270 Megabyte	315 Megabyte	360 Megabyte
10E00000 - 10EFFFFF	13B00000 - 13BFFFFF	16800000 - 168FFFFF
10F00000 - 10FFFFFF	13C00000 - 13CFFFFF	16900000 - 169FFFFF
11000000 - 110FFFFF	13D00000 - 13DFFFFF	16A00000 - 16AFFFFF
11100000 - 111FFFFF	13E00000 - 13EFFFFF	16B00000 - 16BFFFFF
11200000 - 112FFFFF	13F00000 - 13FFFFFF	16C00000 - 16CFFFFF
275 Megabyte	320 Megabyte	365 Megabyte
11300000 - 113FFFFF	14000000 - 140FFFFF	16D00000 - 16DFFFFF
11400000 - 114FFFFF	14100000 - 141FFFFF	16E00000 - 16EFFFFF
11500000 - 115FFFFF	14200000 - 142FFFFF	16F00000 - 16FFFFFF
11600000 - 116FFFFF	14300000 - 143FFFFF	17000000 - 170FFFFF
11700000 - 117FFFFF	14400000 - 144FFFFF	17100000 - 171FFFFF
280 Megabyte	325 Megabyte	370 Megabyte
11800000 - 118FFFFF	14500000 - 145FFFFF	17200000 - 172FFFFF
11900000 - 119FFFFF	14600000 - 146FFFFF	17300000 - 173FFFFF
11A00000 - 11AFFFFF	14700000 - 147FFFFF	17400000 - 174FFFFF
11B00000 - 11BFFFFF	14800000 - 148FFFFF	17500000 - 175FFFFF
11C00000 - 11CFFFFF	14900000 - 149FFFFF	17600000 - 176FFFFF
285 Megabyte	330 Megabyte	375 Megabyte
11D00000 - 11DFFFFF	14A00000 - 14AFFFFF	17700000 - 177FFFFF
11E00000 - 11EFFFFF	14B00000 - 14BFFFFF	17800000 - 178FFFFF
11F00000 - 11FFFFFF	14C00000 - 14CFFFFF	17900000 - 179FFFFF
12000000 - 120FFFFF	14D00000 - 14DFFFFF	17A00000 - 17AFFFFF
12100000 - 121FFFFF	14E00000 - 14EFFFFF	17B00000 - 17BFFFFF
290 Megabyte	335 Megabyte	380 Megabyte
12200000 - 122FFFFF	14F00000 - 14FFFFFF	17C00000 - 17CFFFFF
12300000 - 123FFFFF	15000000 - 150FFFFF	17D00000 - 17DFFFFF
12400000 - 124FFFFF	15100000 - 151FFFFF	17E00000 - 17EFFFFF
12500000 - 125FFFFF	15200000 - 152FFFFF	17F00000 - 17FFFFFF
12600000 - 126FFFFF	15300000 - 153FFFFF	18000000 - 180FFFFF
295 Megabyte	340 Megabyte	385 Megabyte
12700000 - 127FFFFF	15400000 - 154FFFFF	18100000 - 181FFFFF
12800000 - 128FFFFF	15500000 - 155FFFFF	18200000 - 182FFFFF
12900000 - 129FFFFF	15600000 - 156FFFFF	18300000 - 183FFFFF
12A00000 - 12AFFFFF	15700000 - 157FFFFF	18400000 - 184FFFFF
12B00000 - 12BFFFFF	15800000 - 158FFFFF	18500000 - 185FFFFF
300 Megabyte	345 Megabyte	390 Megabyte
12C00000 - 12CFFFFF	15900000 - 159FFFFF	18600000 - 186FFFFF
12D00000 - 12DFFFFF	15A00000 - 15AFFFFF	18700000 - 187FFFFF
12E00000 - 12EFFFFF	15B00000 - 15BFFFFF	18800000 - 188FFFFF
12F00000 - 12FFFFFF	15C00000 - 15DFFFFF	18900000 - 189FFFFF
13000000 - 130FFFFF	15D00000 - 15DFFFFF	18A00000 - 18AFFFFF
305 Megabyte	350 Megabyte	395 Megabyte
13100000 - 131FFFFF	15E00000 - 15EFFFFF	18B00000 - 18BFFFFF
13200000 - 132FFFFF	15F00000 - 15FFFFFF	18C00000 - 18CFFFFF
13300000 - 133FFFFF	16000000 - 160FFFFF	18D00000 - 18DFFFFF
13400000 - 134FFFFF	16100000 - 161FFFFF	18E00000 - 18EFFFFF
13500000 - 135FFFFF	16200000 - 162FFFFF	18F00000 - 18FFFFFF
310 Megabyte	355 Megabyte	400 Megabyte
13600000 - 136FFFFF	16300000 - 163FFFFF	19000000 - 190FFFFF
13700000 - 137FFFFF	16400000 - 164FFFFF	19100000 - 191FFFFF
13800000 - 138FFFFF	16500000 - 165FFFFF	19200000 - 192FFFFF
13900000 - 139FFFFF	16600000 - 166FFFFF	19300000 - 193FFFFF
13A00000 - 13AFFFFF	16700000 - 167FFFFF	19400000 - 194FFFFF

Physical "BYTE" Address Space · *1MB boundries*

405 Megabyte
19500000 - 195FFFFF
19600000 - 196FFFFF
19700000 - 197FFFFF
19800000 - 198FFFFF
19900000 - 199FFFFF

410 Megabyte
19A00000 - 19AFFFFF
19B00000 - 19BFFFFF
19C00000 - 19CFFFFF
19D00000 - 19DFFFFF
19E00000 - 19EFFFFF

415 Megabyte
19F00000 - 19FFFFFF
1A000000 - 1A0FFFFF
1A100000 - 1A1FFFFF
1A200000 - 1A2FFFFF
1A300000 - 1A3FFFFF

420 Megabyte
1A400000 - 1A4FFFFF
1A500000 - 1A5FFFFF
1A600000 - 1A6FFFFF
1A700000 - 1A7FFFFF
1A800000 - 1A8FFFFF

425 Megabyte
1A900000 - 1A9FFFFF
1AA00000 - 1AAFFFFF
1AB00000 - 1ABFFFFF
1AC00000 - 1ACFFFFF
1AD00000 - 1ADFFFFF

430 Megabyte
1AE00000 - 1AEFFFFF
1AF00000 - 1AFFFFFF
1B000000 - 1B0FFFFF
1B100000 - 1B1FFFFF
1B200000 - 1B2FFFFF

435 Megabyte
1B300000 - 1B3FFFFF
1B400000 - 1B4FFFFF
1B500000 - 1B5FFFFF
1B600000 - 1B6FFFFF
1B700000 - 1B7FFFFF

440 Megabyte
1B800000 - 1B8FFFFF
1B900000 - 1B9FFFFF
1BA00000 - 1BAFFFFF
1BB00000 - 1BBFFFFF
1BC00000 - 1BCFFFFF

445 Megabyte
1BD00000 - 1BDFFFFF
1BE00000 - 1BEFFFFF
1BF00000 - 1BFFFFFF
1C000000 - 1C0FFFFF
1C100000 - 1C1FFFFF

450 Megabyte
1C200000 - 1C2FFFFF
1C300000 - 1C3FFFFF
1C400000 - 1C4FFFFF
1C500000 - 1C5FFFFF
1C600000 - 1C6FFFFF

455 Megabyte
1C700000 - 1C7FFFFF
1C800000 - 1C8FFFFF
1C900000 - 1C9FFFFF
1CA00000 - 1CAFFFFF
1CB00000 - 1CBFFFFF

460 Megabyte
1CC00000 - 1CCFFFFF
1CD00000 - 1CDFFFFF
1CE00000 - 1CEFFFFF
1CF00000 - 1CFFFFFF
1D000000 - 1D0FFFFF

465 Megabyte
1D100000 - 1D1FFFFF
1D200000 - 1D2FFFFF
1D300000 - 1D3FFFFF
1D400000 - 1D4FFFFF
1D500000 - 1D5FFFFF

470 Megabyte
1D600000 - 1D6FFFFF
1D700000 - 1D7FFFFF
1D800000 - 1D8FFFFF
1D900000 - 1D9FFFFF
1DA00000 - 1DAFFFFF

475 Megabyte
1DB00000 - 1DBFFFFF
1DC00000 - 1DCFFFFF
1DD00000 - 1DDFFFFF
1DE00000 - 1DEFFFFF
1DF00000 - 1DFFFFFF

480 Megabyte
1E000000 - 1E0FFFFF
1E100000 - 1E1FFFFF
1E200000 - 1E2FFFFF
1E300000 - 1E3FFFFF
1E400000 - 1E4FFFFF

485 Megabyte
1E500000 - 1E5FFFFF
1E600000 - 1E6FFFFF
1E700000 - 1E7FFFFF
1E800000 - 1E8FFFFF
1E900000 - 1E9FFFFF

490 Megabyte
1EA00000 - 1EAFFFFF
1EB00000 - 1EBFFFFF
1EC00000 - 1ECFFFFF
1ED00000 - 1EDFFFFF
1EE00000 - 1EEFFFFF

495 Megabyte
1EF00000 - 1EFFFFFF
1F000000 - 1F0FFFFF
1F100000 - 1F1FFFFF
1F200000 - 1F2FFFFF
1F300000 - 1F3FFFFF

500 Megabyte
1F400000 - 1F4FFFFF
1F500000 - 1F5FFFFF
1F600000 - 1F6FFFFF
1F700000 - 1F7FFFFF
1F800000 - 1F8FFFFF

505 Megabyte
1F900000 - 1F9FFFFF
1FA00000 - 1FAFFFFF
1FB00000 - 1FBFFFFF
1FC00000 - 1FCFFFFF
1FD00000 - 1FDFFFFF

510 Megabyte
1FE00000 - 1FEFFFFF
1FF00000 - 1FFFFFFF

Physical "Longword" Address Space *1 MB boundries*

000 Megabyte	044 Megabyte	088 Megabyte
0000000 - 003FFFF	0B00000 - 0B3FFFF	1600000 - 163FFFF
0040000 - 007FFFF	0B40000 - 0B7FFFF	1640000 - 167FFFF
0080000 - 00BFFFF	0B80000 - 0BBFFFF	1680000 - 16BFFFF
00C0000 - 00FFFFF	0BC0000 - 0BFFFFF	16C0000 - 16FFFFF
004 Megabyte	048 Megabyte	092 Megabyte
0100000 - 013FFFF	0C00000 - 0C3FFFF	1700000 - 173FFFF
0140000 - 017FFFF	0C40000 - 0C7FFFF	1740000 - 177FFFF
0180000 - 01BFFFF	0C80000 - 0CBFFFF	1780000 - 17BFFFF
01C0000 - 01FFFFF	0CC0000 - 0CFFFFF	17C0000 - 17FFFFF
008 Megabyte	052 Megabyte	096 Megabyte
0200000 - 023FFFF	0D00000 - 0D3FFFF	1800000 - 183FFFF
0240000 - 027FFFF	0D40000 - 0D7FFFF	1840000 - 187FFFF
0280000 - 02BFFFF	0D80000 - 0DBFFFF	1880000 - 18BFFFF
02C0000 - 02FFFFF	0DC0000 - 0DFFFFF	18C0000 - 18FFFFF
012 Megabyte	056 Megabyte	100 Megabyte
0300000 - 033FFFF	0E00000 - 0E3FFFF	1900000 - 193FFFF
0340000 - 037FFFF	0E40000 - 0E7FFFF	1940000 - 197FFFF
0380000 - 03BFFFF	0E80000 - 0EBFFFF	1980000 - 19BFFFF
03C0000 - 03FFFFF	0EC0000 - 0EFFFFF	19C0000 - 19FFFFF
016 Megabyte	060 Megabyte	104 Megabyte
0400000 - 043FFFF	0F00000 - 0F3FFFF	1A00000 - 1A3FFFF
0440000 - 047FFFF	0F40000 - 0F7FFFF	1A40000 - 1A7FFFF
0480000 - 04BFFFF	0F80000 - 0FBFFFF	1A80000 - 1ABFFFF
04C0000 - 04FFFFF	0FC0000 - 0FFFFFF	1AC0000 - 1AFFFFF
020 Megabyte	064 Megabyte	108 Megabyte
0500000 - 053FFFF	1000000 - 103FFFF	1B00000 - 1B3FFFF
0540000 - 057FFFF	1040000 - 107FFFF	1B40000 - 1B7FFFF
0580000 - 05BFFFF	1080000 - 10BFFFF	1B80000 - 1BBFFFF
05C0000 - 05FFFFF	10C0000 - 10FFFFF	1BC0000 - 1BFFFFF
024 Megabyte	068 Megabyte	112 Megabyte
0600000 - 063FFFF	1100000 - 113FFFF	1C00000 - 1C3FFFF
0640000 - 067FFFF	1140000 - 117FFFF	1C40000 - 1C7FFFF
0680000 - 06BFFFF	1180000 - 11BFFFF	1C80000 - 1CBFFFF
06C0000 - 06FFFFF	11C0000 - 11FFFFF	1CC0000 - 1CFFFFF
028 Megabyte	072 Megabyte	116 Megabyte
0700000 - 073FFFF	1200000 - 123FFFF	1D00000 - 1D3FFFF
0740000 - 077FFFF	1240000 - 127FFFF	1D40000 - 1D7FFFF
0780000 - 07BFFFF	1280000 - 12BFFFF	1D80000 - 1DBFFFF
07C0000 - 07FFFFF	12C0000 - 12FFFFF	1DC0000 - 1DFFFFF
032 Megabyte	076 Megabyte	120 Megabyte
0800000 - 083FFFF	1300000 - 133FFFF	1E00000 - 1E3FFFF
0840000 - 087FFFF	1340000 - 137FFFF	1E40000 - 1E7FFFF
0880000 - 08BFFFF	1380000 - 13BFFFF	1E80000 - 1EBFFFF
08C0000 - 08FFFFF	13C0000 - 13FFFFF	1EC0000 - 1EFFFFF
036 Megabyte	080 Megabyte	124 Megabyte
0900000 - 093FFFF	1400000 - 143FFFF	1F00000 - 1F3FFFF
0940000 - 097FFFF	1440000 - 147FFFF	1F40000 - 1F7FFFF
0980000 - 09BFFFF	1480000 - 14BFFFF	1F80000 - 1FBFFFF
09C0000 - 09FFFFF	14C0000 - 14FFFFF	1FC0000 - 1FFFFFF
040 Megabyte	084 Megabyte	
0A00000 - 0A3FFFF	1500000 - 153FFFF	*** BEWARE ***
0AC0000 - 0A7FFFF	1540000 - 157FFFF	These are LONGWORD address
0A80000 - 0ABFFFF	1580000 - 15BFFFF	ranges,(not byte ranges).
0AC0000 - 0AFFFFF	15C0000 - 15FFFFF	(PA<29:02>) / (ID#1A<27:00>)

Physical "Longword" Address Space *1 MB boundries*

128 Megabyte
2000000 - 203FFFF
2040000 - 207FFFF
2080000 - 20BFFFF
20C0000 - 20FFFFF

132 Megabyte
2100000 - 213FFFF
2140000 - 217FFFF
2180000 - 21BFFFF
21C0000 - 21FFFFF

136 Megabyte
2200000 - 223FFFF
2240000 - 227FFFF
2280000 - 22BFFFF
22C0000 - 22FFFFF

140 Megabyte
2300000 - 233FFFF
2340000 - 237FFFF
2380000 - 23BFFFF
23C0000 - 23FFFFF

144 Megabyte
2400000 - 243FFFF
2440000 - 247FFFF
2480000 - 24BFFFF
24C0000 - 24FFFFF

148 Megabyte
2500000 - 253FFFF
2540000 - 257FFFF
2580000 - 25BFFFF
25C0000 - 25FFFFF

152 Megabyte
2600000 - 263FFFF
2640000 - 267FFFF
2680000 - 26BFFFF
26C0000 - 26FFFFF

156 Megabyte
2700000 - 273FFFF
2740000 - 277FFFF
2780000 - 27BFFFF
27C0000 - 27FFFFF

160 Megabyte
2800000 - 283FFFF
2840000 - 287FFFF
2880000 - 28BFFFF
28C0000 - 28FFFFF

164 Megabyte
2900000 - 293FFFF
2940000 - 297FFFF
2980000 - 29BFFFF
29C0000 - 29FFFFF

168 Megabyte
2A00000 - 2A3FFFF
2AC0000 - 2A7FFFF
2A80000 - 2ABFFFF
2AC0000 - 2AFFFFF

172 Megabyte
2B00000 - 2B3FFFF
2B40000 - 2B7FFFF
2B80000 - 2BBFFFF
2BC0000 - 2BFFFFF

176 Megabyte
2C00000 - 2C3FFFF
2C40000 - 2C7FFFF
2C80000 - 2CBFFFF
2CC0000 - 2CFFFFF

180 Megabyte
2D00000 - 2D3FFFF
2D40000 - 2D7FFFF
2D80000 - 2DBFFFF
2DC0000 - 2DFFFFF

184 Megabyte
2E00000 - 2E3FFFF
2E40000 - 2E7FFFF
2E80000 - 2EBFFFF
2EC0000 - 2EFFFFF

188 Megabyte
2F00000 - 2F3FFFF
2F40000 - 2F7FFFF
2F80000 - 2FBFFFF
2FC0000 - 2FFFFFF

192 Megabyte
3000000 - 303FFFF
3040000 - 307FFFF
3080000 - 30BFFFF
30C0000 - 30FFFFF

196 Megabyte
3100000 - 313FFFF
3140000 - 317FFFF
3180000 - 31BFFFF
31C0000 - 31FFFFF

200 Megabyte
3200000 - 323FFFF
3240000 - 327FFFF
3280000 - 32BFFFF
32C0000 - 32FFFFF

204 Megabyte
3300000 - 333FFFF
3340000 - 337FFFF
3380000 - 33BFFFF
33C0000 - 33FFFFF

208 Megabyte
3400000 - 343FFFF
3440000 - 347FFFF
3480000 - 34BFFFF
34C0000 - 34FFFFF

212 Megabyte
3500000 - 353FFFF
3540000 - 357FFFF
3580000 - 35BFFFF
35C0000 - 35FFFFF

216 Megabyte
3600000 - 363FFFF
3640000 - 367FFFF
3680000 - 36BFFFF
36C0000 - 36FFFFF

220 Megabyte
3700000 - 373FFFF
3740000 - 377FFFF
3780000 - 37BFFFF
37C0000 - 37FFFFF

224 Megabyte
3800000 - 383FFFF
3840000 - 387FFFF
3880000 - 38BFFFF
38C0000 - 38FFFFF

228 Megabyte
3900000 - 393FFFF
3940000 - 397FFFF
3980000 - 39BFFFF
39C0000 - 39FFFFF

232 Megabyte
3A00000 - 3A3FFFF
3A40000 - 3A7FFFF
3A80000 - 3ABFFFF
3AC0000 - 3AFFFFF

236 Megabyte
3B00000 - 3B3FFFF
3B40000 - 3B7FFFF
3B80000 - 3BBFFFF
3BC0000 - 3BFFFFF

240 Megabyte
3C00000 - 3C3FFFF
3C40000 - 3C7FFFF
3C80000 - 3CBFFFF
3CC0000 - 3CFFFFF

244 Megabyte
3D00000 - 3D3FFFF
3D40000 - 3D7FFFF
3D80000 - 3DBFFFF
3DC0000 - 3DFFFFF

248 Megabyte
3E00000 - 3E3FFFF
3E40000 - 3E7FFFF
3E80000 - 3EBFFFF
3EC0000 - 3EFFFFF

252 Megabyte
3F00000 - 3F3FFFF
3F40000 - 3F7FFFF
3F80000 - 3FBFFFF
3FC0000 - 3FFFFFF

*** BEWARE ***
These are LONGWORD address ranges,(not byte ranges).
(PA<29:02>) / (ID#1A<27:00>)

Physical "Longword" Address Space *1 MB boundries*

256 Megabyte
4000000 - 403FFFF
4040000 - 407FFFF
4080000 - 40BFFFF
40C0000 - 40FFFFF

260 Megabyte
4100000 - 413FFFF
4140000 - 417FFFF
4180000 - 41BFFFF
41C0000 - 41FFFFF

264 Megabyte
4200000 - 423FFFF
4240000 - 427FFFF
4280000 - 42BFFFF
42C0000 - 42FFFFF

268 Megabyte
4300000 - 433FFFF
4340000 - 437FFFF
4380000 - 43BFFFF
43C0000 - 43FFFFF

272 Megabyte
4400000 - 443FFFF
4440000 - 447FFFF
4480000 - 44BFFFF
44C0000 - 44FFFFF

276 Megabyte
4500000 - 453FFFF
4540000 - 457FFFF
4580000 - 45BFFFF
45C0000 - 45FFFFF

280 Megabyte
4600000 - 463FFFF
4640000 - 467FFFF
4680000 - 46BFFFF
46C0000 - 46FFFFF

284 Megabyte
4700000 - 473FFFF
4740000 - 477FFFF
4780000 - 47BFFFF
47C0000 - 47FFFFF

288 Megabyte
4800000 - 483FFFF
4840000 - 487FFFF
4880000 - 48BFFFF
48C0000 - 48FFFFF

292 Megabyte
4900000 - 493FFFF
4940000 - 497FFFF
4980000 - 49BFFFF
49C0000 - 49FFFFF

296 Megabyte
4A00000 - 4A3FFFF
4AC0000 - 4A7FFFF
4A80000 - 4ABFFFF
4AC0000 - 4AFFFFF

300 Megabyte
4B00000 - 4B3FFFF
4B40000 - 4B7FFFF
4B80000 - 4BBFFFF
4BC0000 - 4BFFFFF

304 Megabyte
4C00000 - 4C3FFFF
4C40000 - 4C7FFFF
4C80000 - 4CBFFFF
4CC0000 - 4CFFFFF

308 Megabyte
4D00000 - 4D3FFFF
4D40000 - 4D7FFFF
4D80000 - 4DBFFFF
4DC0000 - 4DFFFFF

312 Megabyte
4E00000 - 4E3FFFF
4E40000 - 4E7FFFF
4E80000 - 4EBFFFF
4EC0000 - 4EFFFFF

316 Megabyte
4F00000 - 4F3FFFF
4F40000 - 4F7FFFF
4F80000 - 4FBFFFF
4FC0000 - 4FFFFFF

320 Megabyte
5000000 - 53FFFFF
5040000 - 57FFFFF
5080000 - 5BFFFFF
50C0000 - 5FFFFFF

324 Megabyte
5100000 - 513FFFF
5140000 - 517FFFF
5180000 - 51BFFFF
51C0000 - 51FFFFF

328 Megabyte
5200000 - 523FFFF
5240000 - 527FFFF
5280000 - 52BFFFF
52C0000 - 52FFFFF

332 Megabyte
5300000 - 533FFFF
5340000 - 537FFFF
5380000 - 53BFFFF
53C0000 - 53FFFFF

336 Megabyte
5400000 - 543FFFF
5440000 - 547FFFF
5480000 - 54BFFFF
54C0000 - 54FFFFF

340 Megabyte
5500000 - 553FFFF
5540000 - 557FFFF
5580000 - 55BFFFF
55C0000 - 55FFFFF

344 Megabyte
5600000 - 563FFFF
5640000 - 567FFFF
5680000 - 56BFFFF
56C0000 - 56FFFFF

348 Megabyte
5700000 - 573FFFF
5740000 - 57FFFFF
5780000 - 57BFFFF
57C0000 - 57FFFFF

352 Megabyte
5800000 - 583FFFF
5840000 - 587FFFF
5880000 - 58BFFFF
58C0000 - 58FFFFF

356 Megabyte
5900000 - 593FFFF
5940000 - 597FFFF
5980000 - 59BFFFF
59C0000 - 59FFFFF

360 Megabyte
5A00000 - 5A3FFFF
5A40000 - 5A7FFFF
5A80000 - 5ABFFFF
5AC0000 - 5AFFFFF

364 Megabyte
5B00000 - 5B3FFFF
5B40000 - 5B7FFFF
5B80000 - 5BBFFFF
5BC0000 - 5BFFFFF

368 Megabyte
5C00000 - 5C3FFFF
5C40000 - 5C7FFFF
5C80000 - 5CBFFFF
5CC0000 - 5CFFFFF

372 Megabyte
5D00000 - 5D3FFFF
5D40000 - 5D7FFFF
5D80000 - 5DBFFFF
5DC0000 - 5DFFFFF

376 Megabyte
5E00000 - 5E3FFFF
5E40000 - 5E7FFFF
5E80000 - 5EBFFFF
5EC0000 - 5EFFFFF

380 Megabyte
5F00000 - 5F3FFFF
5F40000 - 5F7FFFF
5F80000 - 5FBFFFF
5FC0000 - 5FFFFFF

*** BEWARE ***
These are LONGWORD address ranges,(not byte ranges).
(PA<29:02>) / (ID#1A<27:00>)

Physical "Longword" Address Space *1 MB boundries*

384 Megabyte
6000000 - 603FFFF
6040000 - 607FFFF
6080000 - 60BFFFF
60C0000 - 60FFFFF

388 Megabyte
6100000 - 613FFFF
6140000 - 617FFFF
6180000 - 61BFFFF
61C0000 - 61FFFFF

392 Megabyte
6200000 - 623FFFF
6240000 - 627FFFF
6280000 - 62BFFFF
62C0000 - 62FFFFF

396 Megabyte
6300000 - 633FFFF
6340000 - 637FFFF
6380000 - 63BFFFF
63C0000 - 63FFFFF

400 Megabyte
6400000 - 643FFFF
6440000 - 647FFFF
6480000 - 64BFFFF
64C0000 - 64FFFFF

404 Megabyte
6500000 - 653FFFF
6540000 - 657FFFF
6580000 - 65BFFFF
65C0000 - 65FFFFF

408 Megabyte
6600000 - 663FFFF
6640000 - 667FFFF
6680000 - 66BFFFF
66C0000 - 66FFFFF

412 Megabyte
6700000 - 673FFFF
6740000 - 677FFFF
6780000 - 67BFFFF
67C0000 - 67FFFFF

416 Megabyte
6800000 - 683FFFF
6840000 - 687FFFF
6880000 - 68BFFFF
68C0000 - 68FFFFF

420 Megabyte
6900000 - 693FFFF
6940000 - 697FFFF
6980000 - 69BFFFF
69C0000 - 69FFFFF

424 Megabyte
6A00000 - 6A3FFFF
6AC0000 - 6A7FFFF
6A80000 - 6ABFFFF
6AC0000 - 6AFFFFF

428 Megabyte
6B00000 - 6B3FFFF
6B40000 - 6B7FFFF
6B80000 - 6BBFFFF
6BC0000 - 6BFFFFF

432 Megabyte
6C00000 - 6C3FFFF
6C40000 - 6C7FFFF
6C80000 - 6CBFFFF
6CC0000 - 6CFFFFF

436 Megabyte
6D00000 - 6D3FFFF
6D40000 - 6D7FFFF
6D80000 - 6DBFFFF
6DC0000 - 6DFFFFF

440 Megabyte
6E00000 - 6E3FFFF
6E40000 - 6E7FFFF
6E80000 - 6EBFFFF
6EC0000 - 6EFFFFF

444 Megabyte
6F00000 - 6F3FFFF
6F40000 - 6F7FFFF
6F80000 - 6FBFFFF
6FC0000 - 6FFFFFF

448 Megabyte
7000000 - 703FFFF
7040000 - 707FFFF
7080000 - 70BFFFF
70C0000 - 70FFFFF

452 Megabyte
7100000 - 713FFFF
7140000 - 717FFFF
7180000 - 71BFFFF
71C0000 - 71FFFFF

456 Megabyte
7200000 - 723FFFF
7240000 - 727FFFF
7280000 - 72BFFFF
72C0000 - 72FFFFF

460 Megabyte
7300000 - 733FFFF
7340000 - 737FFFF
7380000 - 73BFFFF
73C0000 - 73FFFFF

464 Megabyte
7400000 - 743FFFF
7440000 - 747FFFF
7480000 - 74BFFFF
74C0000 - 74FFFFF

468 Megabyte
7500000 - 753FFFF
7540000 - 757FFFF
7580000 - 75BFFFF
75C0000 - 75FFFFF

472 Megabyte
7600000 - 763FFFF
7640000 - 767FFFF
7680000 - 76BFFFF
76C0000 - 76FFFFF

476 Megabyte
7700000 - 773FFFF
7740000 - 777FFFF
7780000 - 77BFFFF
77C0000 - 77FFFFF

480 Megabyt
7800000 - 783FFFF
7840000 - 787FFFF
7880000 - 78BFFFF
78C0000 - 78FFFFF

484 Megabyte
7900000 - 793FFFF
7940000 - 797FFFF
7980000 - 79BFFFF
79C0000 - 79FFFFF

488 Megabyte
7A00000 - 7A3FFFF
7A40000 - 7A7FFFF
7A80000 - 7ABFFFF
7AC0000 - 7AFFFFF

492 Megabyte
7B00000 - 7B3FFFF
7B40000 - 7B7FFFF
7B80000 - 7BBFFFF
7BC0000 - 7BFFFFF

496 Megabyte
7C00000 - 7C3FFFF
7C40000 - 7C7FFFF
7C80000 - 7CBFFFF
7CC0000 - 7CFFFFF

500 Megabyte
7D00000 - 7D3FFFF
7D40000 - 7D7FFFF
7D80000 - 7DBFFFF
7DC0000 - 7DFFFFF

504 Megabyte
7E00000 - 7E3FFFF
7E40000 - 7E7FFFF
7E80000 - 7EBFFFF
7EC0000 - 7EFFFFF

508 Megabyte
7F00000 - 7F3FFFF
7F40000 - 7F7FFFF
7F80000 - 7FBFFFF
7FC0000 - 7FFFFFF

*** BEWARE ***
These are LONGWORD address ranges,(not byte ranges).
(PA<29:02>) / (ID#1A<27:00>)

The following charts show the TIMEOUT ADDRESS (ID #1A) range for the VAX NEXUS devices. The address ranges actually show the Longword Address ranges for each device. The address shown in "ID #1A" bits <27:00> are equal to the Physical address bits PA<29:02>, which is actually a Longword address. The charts showing memory array address ranges assume that the memories are not interleaved.

MS780C "0-4 Megabyte"/MS780A "0-1 Megabyte" ("LONGWORD" ranges)

Array	Slot	MS780C M8210 Arrays	MS780A M8211 Arrays
0	17	0000000 - 000FFFF	0000000 - 0003FFF
1	16	0010000 - 001FFFF	0004000 - 0007FFF
2	15	0020000 - 002FFFF	0008000 - 000BFFF
3	14	0030000 - 003FFFF	000C000 - 000FFFF
4	13	0040000 - 004FFFF	0010000 - 0013FFF
5	12	0050000 - 005FFFF	0014000 - 0017FFF
6	11	0060000 - 006FFFF	0018000 - 001BFFF
7	10	0070000 - 007FFFF	001C000 - 001FFFF
8	9	0080000 - 008FFFF	0020000 - 0023FFF
9	8	0090000 - 009FFFF	0024000 - 0027FFF
A	7	00A0000 - 00AFFFF	0028000 - 002BFFF
B	6	00B0000 - 00BFFFF	002C000 - 002FFFF
C	5	00C0000 - 00CFFFF	0030000 - 0033FFF
D	4	00D0000 - 00DFFFF	0034000 - 0037FFF
E	3	00E0000 - 00EFFFF	0038000 - 003BFFF
F	2	00F0000 - 00FFFFF	003C000 - 003FFFF

MS780C "4-8 Megabyte"/MS780A "1-2 Megabyte" ("LONGWORD" ranges)

Array	Slot	MS780C M8210 Arrays	MS780A M8211 Arrays
0	17	0100000 - 010FFFF	0040000 - 0043FFF
1	16	0110000 - 011FFFF	0044000 - 0047FFF
2	15	0120000 - 012FFFF	0048000 - 004BFFF
3	14	0130000 - 013FFFF	004C000 - 004FFFF
4	13	0140000 - 014FFFF	0050000 - 0053FFF
5	12	0150000 - 015FFFF	0054000 - 0057FFF
6	11	0160000 - 016FFFF	0058000 - 005BFFF
7	10	0170000 - 017FFFF	005C000 - 005FFFF
8	9	0180000 - 018FFFF	0060000 - 0063FFF
9	8	0190000 - 019FFFF	0064000 - 0067FFF
A	7	01A0000 - 01AFFFF	0068000 - 006BFFF
B	6	01B0000 - 01BFFFF	006C000 - 006FFFF
C	5	01C0000 - 01CFFFF	0070000 - 0073FFF
D	4	01D0000 - 01DFFFF	0074000 - 0077FFF
E	3	01E0000 - 01EFFFF	0078000 - 007BFFF
F	2	01F0000 - 01FFFFF	007C000 - 007FFFF

MA780A Array "Longword" Address Ranges

Array	Slot	MA780A M8210 Arrays 0 to 2 Megabyte	MA780A M8210 Arrays 2 to 4 Megabyte
0	8	0000000 - 000FFFF	0080000 - 008FFFF
1	7	0010000 - 001FFFF	0090000 - 009FFFF
2	6	0020000 - 002FFFF	00A0000 - 00AFFFF
3	5	0030000 - 003FFFF	00B0000 - 00BFFFF
4	4	0040000 - 004FFFF	00C0000 - 00CFFFF
5	3	0050000 - 005FFFF	00D0000 - 00DFFFF
6	2	0060000 - 006FFFF	00E0000 - 00EFFFF
7	1	0070000 - 007FFFF	00F0000 - 00FFFFF

Array	Slot	MA780A M8210 Arrays 4 to 6 Megabyte	MA780A M8210 Arrays 6 to 8 Megabyte
0	8	0100000 - 010FFFF	0180000 - 018FFFF
1	7	0110000 - 011FFFF	0190000 - 019FFFF
2	6	0120000 - 012FFFF	01A0000 - 01AFFFF
3	5	0130000 - 013FFFF	01B0000 - 01BFFFF
4	4	0140000 - 014FFFF	01C0000 - 01CFFFF
5	3	0150000 - 015FFFF	01D0000 - 01DFFFF
6	2	0160000 - 016FFFF	01E0000 - 01EFFFF
7	1	0170000 - 017FFFF	01F0000 - 01FFFFF

MS780-E Array "Longword" Address Ranges

Internally Interleaved

A total of 16 Megabytes per MS780-E are possible when internally interleaving. In order to internal interleave the MS780-E, the following configuration guidelines must be followed:

- Memory Controllers (M8375's) must be installed in slots 10 & 12.
- Each controller must have the same amount of memory arrays, of the same type and capacity, in their respective array slots.
- There cannot be any gaps between the arrays on each controller. In other words, the Lower Controller expands from slot 9 towards slot 2 while the Upper Controller expands from slot 13 towards slot 20.

When the MS780-E is configured with two controllers, the memory is internally interleaved as follows:

- The "Lower Controller's Arrays" contain the "EVEN Physical QUADWORD" addresses, (Physical Address bit 3=0).
- The "Upper Controller's Arrays" contain the "ODD Physical QUADWORD" addresses, (Physical Address bit 3=1).

Array	Slot	MS780-E M8373 Arrays 0 to 16 Megabyte		MS780-E M8373 Arrays 16 to 32 Megabyte
L0	9	0000000 - 007FFFF	(PA3=0)	0400000 - 047FFFF
U0	13		(PA3=1)	
L1	8	0080000 - 00FFFFF	(PA3=0)	0480000 - 04FFFFF
U1	14		(PA3=1)	
L2	7	0100000 - 017FFFF	(PA3=0)	0500000 - 057FFFF
U2	15		(PA3=1)	
L3	6	0180000 - 01FFFFF	(PA3=0)	0580000 - 05FFFFF
U3	16		(PA3=1)	
L4	5	0200000 - 027FFFF	(PA3=0)	0600000 - 067FFFF
U4	17		(PA3=1)	
L5	4	0280000 - 02FFFFF	(PA3=0)	0680000 - 06FFFFF
U5	18		(PA3=1)	
L6	3	0300000 - 037FFFF	(PA3=0)	0700000 - 077FFFF
U6	19		(PA3=1)	
L7	2	0380000 - 03FFFFF	(PA3=0)	0780000 - 07FFFFF
U7	20		(PA3=1)	

L = Lower Controller's Array
U = Upper Controller's Array
PA3 = Physical Address bit 3

MS780-E Array "Longword" Address Ranges (cont'd)

No Internal Interleaving

The MS780-E may be operated in NO_INTERNAL_INTERLEAVING mode but the total amount of memory per MS780-E is reduced to a total of 8 Megabytes. This mode of operation is accomplished whenever the following configuration guidelines are followed:

- There is only one Memory Controller (M8375) installed in the MS780-E backplane. This controller can be installed in either slot 10 or 12 (slot 10 is preferred).
- If the memory controller is installed in slot 10, the memory array modules must be installed in slots 9 through 2 only. No memory arrays are to be installed in slots 13 through 20.
- If the memory controller is installed in slot 12, the memory array modules must be installed in slots 13 through 20 only. No memory arrays are to be installed in slots 9 through 2.
- The memory arrays must be installed with no gaps between arrays and no gap between the memory controller and the first array.

Array	Slot	MS780-E M8373 Arrays 0 to 8 Megabyte	MS780-E M8373 Arrays 8 to 16 Megabyte
0	9 or 13	0000000 - 003FFFF	0200000 - 023FFFF
1	8 or 14	0040000 - 007FFFF	0240000 - 027FFFF
2	7 or 15	0080000 - 00BFFFF	0280000 - 02BFFFF
3	6 or 16	00C0000 - 00FFFFF	02C0000 - 02FFFFF
4	5 or 17	0100000 - 013FFFF	0300000 - 033FFFF
5	4 or 18	0140000 - 017FFFF	0340000 - 037FFFF
6	3 or 19	0180000 - 01BFFFF	0380000 - 03BFFFF
7	2 or 20	01C0000 - 01FFFFF	03C0000 - 03FFFFF

In the above chart, the slot of the array depends upon which memory controller is installed. Slots 2 through 9 are used if the Memory Controller is in slot 10, and slots 13 through 20 are used if the Memory Controller is in slot 12.

MS780-E Array "Longword" Address Ranges (cont'd)

Externally Interleaved

A total of 16 Megabytes are possible when externally interleaving two MS780-E controllers. In order to externally interleave 2 MS780-E memory backplanes, the following configuration guidelines must be followed:

- Both memory backplanes must be configured to operate in the non_internal_interleaved mode.
- Both memory subsystems must have the same amount of memory arrays, of the same type and capacity, and in corresponding slot locations.
- Both memory subsystems must have the same assigned starting address.
- The memory subsystems must have adjacent "TR Levels" assigned to them.
- "Bit <0>" of both memory subsystems' "CNFG A register" must be set prior to memory usage.

When two MS780-E's are configured for EXTERNAL interleaving, the following rules are used to determine what address are located in what memory.

- The Memory Subsystem assigned the "Lower TR Level" contains the "EVEN Quadword" addresses.
- The Memory Subsystem assigned the "Higher TR Level" contains the "ODD Quadword" addresses.

Array	Slot	MS780-E M8373 Arrays 0 to 16 Megabyte	
L0	9/13	0000000 - 007FFFF	(PA3=0)
U0	9/13		(PA3=1)
L1	8/14	0080000 - 00FFFFF	(PA3=0)
U1	8/14		(PA3=1)
L2	7/15	0100000 - 017FFFF	(PA3=0)
U2	7/15		(PA3=1)
L3	6/16	0180000 - 01FFFFF	(PA3=0)
U3	6/16		(PA3=1)
L4	5/17	0200000 - 027FFFF	(PA3=0)
U4	5/17		(PA3=1)
L5	4/18	0280000 - 02FFFFF	(PA3=0)
U5	4/18		(PA3=1)
L6	3/19	0300000 - 037FFFF	(PA3=0)
U6	3/19		(PA3=1)
L7	2/20	0380000 - 03FFFFF	(PA3=0)
U7	2/20		(PA3=1)

L = Memory Subsystem with the Lower assighed "TR Level".
H = Memory Subsystem with the Higher assigned "TR Level".
PA3 = Physical Address bit 3

Converting a "UNIBUS Byte *(Octal Format)* Address" to a "VAX *(Hex Format)* Longword" address

1. Take the UNIBUS address and drop off the 2 least significant "binary" bits (Unibus address bits <1:0> are not used).

```
 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
|     use these binary bits for conversion        | not |
                                                    used
```

2. Change the Unibus address bits <17:02> to a Hexadecimal number by breaking the "binary" representation of these address bits into 4-bit sections. Do not use address bits <1:0>.

```
| 17 16 15 14 | 13 12 11 10 | 09 08 07 06 | 05 04 03 02 |
```

3. Using the hexadecimal number converted in steps 1 and 2, add it to one of the following DW780 Adapter Base addresses (which one you use depends on which DW780 the UNIBUS device resides).

DW780 Adapter	DW780 Unibus Space Longword Base Address
0	8040000
1	8050000
2	8060000
3	8070000

4. The resulting hexadecimal number (should be 7 hexadecimal digits in length) is the SBI Longword address that is used to access the UNIBUS address just converted for that particular Adapter's UNIBUS. This is the address that will be stored in the "TIMEOUT ADDRESS" (ID #1A) register on a CP Timeout. If you want to find out what the Physical Byte address is, simply convert the Longword address by adding two binary zeros as the least significant bits and then reconvert back to hexadecimal.

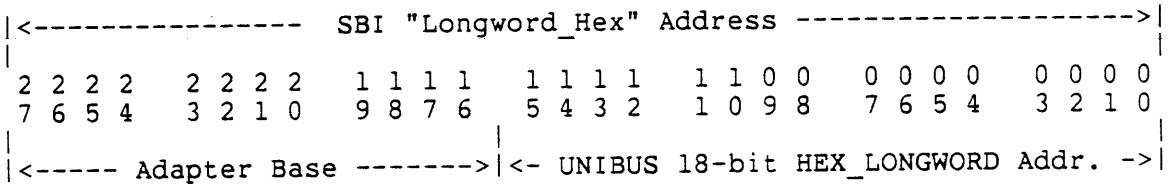
Converting a "VAX LONGWORD *(Hex Format)*" address to a "UNIBUS_BYTE *(Octal Format)*" address

1. First of all, you must make sure that you have an SBI address that is assigned to a DW780 Adapters' Unibus space. Check to see that the address falls in one of the following ranges:

   ```
   Adapter #0 SBI UNIBUS_Address_Space  -  8040000 thru 804FFFF
   Adapter #1 SBI UNIBUS_Address_Space  -  8050000 thru 805FFFF
   Adapter #2 SBI UNIBUS_Address_Space  -  8060000 thru 806FFFF
   Adapter #3 SBI UNIBUS_Address_Space  -  8070000 thru 807FFFF
   ```

2. If the address to be converted falls in one of these ranges, then you do have a VAX Physical Longword Unibus address.

   Drop off the 3 most significant digits (804, 805, 806, or 807), and use the remaining four digits to find the equivalent UNIBUS 18-bit octal address.



3. Change these four HEX digits (the digits labeled "UNIBUS 18-bit HEX_LONGWORD Address" in the diagram above) to their BINARY representation. You should now have 16 binary digits written down.

4. Add two binary zeros to the least significant end (far right end) of this BINARY number. This will change the "UNIBUS LONGWORD address" to a "UNIBUS BYTE address". You should now have 18 binary bits written down with the last two digits on the right being zeros.

5. Convert the result back to octal by breaking up into three digit sections. You should end up with six octal digits. This is the "UNIBUS BYTE address" in octal representation.

Converting "VAX PHYSICAL BYTE *(Hex Format)*" address to "UNIBUS *(Octal Format)*" address

1. First of all, you must make sure that you have a Physical Byte address that is assigned to one of the DW780 Adapters' UNIBUS space. Check to see that the address falls in one of the following ranges:

```
Adapter #0 UNIBUS_Byte_Address space   -   20100000 thru 2013FFFF
Adapter #1 UNIBUS_Byte_Address space   -   20140000 thru 2017FFFF
Adapter #2 UNIBUS_Byte_Address space   -   20180000 thru 201BFFFF
Adapter #3 UNIBUS_Byte_Address space   -   201C0000 thru 201FFFFF
```

2. If the address to be converted falls in one of these ranges, then you do have a VAX (hex) Physical Byte UNIBUS address.

   Extract the 18 least significant digits from this address. These 18 bits represent the HEX representation of the UNIBUS address.

```
|<----------------  VAX Physical Byte UNIBUS Address  ---------------->|
|                                                                       |
 3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
 1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
|                           |                                           |
|<----- Adapter Base ------>|<-- UNIBUS 18-Bit HEX_BYTE Address --->|
```

3. Change these 5 HEX digits (the digits labeled "UNIBUT 18-Bit HEX_BYTE Address" in the diagram above) to their BINARY representation.

```
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

4. Now, break this BINARY representation up into 3 bit sections so as to convert to OCTAL representation.

```
1 1 1 *  1 1 1 *  1 1 0 *  0 0 0 *  0 0 0 *  0 0 0
7 6 5 *  4 3 2 *  1 0 9 *  8 7 6 *  5 4 3 *  2 1 0
```

5. Read this broken up Binary representation in OCTAL. The result is the UNIBUS BYTE address in OCTAL representation.

Converting a "UNIBUS *Octal_BYTE*" Address to a "VAX *Hex_PHYSICAL_BYTE*" Address

1. Take the OCTAL_UNIBUS_BYTE_Address and change it to its BINARY representation.

```
1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

2. Now change this BINARY representation, of the UNIBUS_BYTE_Address, to its HEX representation by breaking up the Binary representation into 4 digit sections (you must start with the least significant digit <00> and work towards the most significant digit <17>).

```
1 1  *  1 1 1 1  *  1 1 0 0  *  0 0 0 0  *  0 0 0 0
7 6  *  5 4 3 2  *  1 0 9 8  *  7 6 5 4  *  3 2 1 0
```

3. Add the appropriate DW780 UNIBUS Adapter_Base_Address onto the resultant HEX number converted in the preceeding step. The following chart shows the Adapter_Base_Addresses for the 4 possible DW780 adapters.

```
Adapter #0  =  20100000          Adapter #1  =  20140000
Adapter #2  =  20180000          Adapter #3  =  201C0000
```

4. The resultant HEX number is the HEX representation of the VAX PHYSICAL BYTE UNIBUS Address.

```
 201x0000  <--- UNIBUS_Space_Base of desired DW780 Adapter
+   yyyyy  <--- Hex representation of UNIBUS address
 --------
 201zzzzz  <--- HEX representation of the UNIBUS_Address
                converted to a VAX_PHYSICAL_BYTE_UNIBUS_Address.
```

UNIBUS device Address		Equivalent DW780 Adapter "Longword" address #0	#1	#2	#3
760010	--	804F802	805F802	806F802	807F802
760020	--	804F804	805F804	806F804	807F804
760030	--	804F806	805F806	806F806	807F806
760040	--	804F808	805F808	806F808	807F808
760050	--	804F80A	805F80A	806F80A	807F80A
760060	--	804F80C	805F80C	806F80C	807F80C
760070	--	804F80E	805F80E	806F80E	807F80E
760100	--	804F810	805F810	806F810	807F810
760110	--	804F812	805F812	806F812	807F812
760120	--	804F814	805F814	806F814	807F814
760130	--	804F816	805F816	806F816	807F816
760140	--	804F818	805F818	806F818	807F818
760150	--	804F81A	805F81A	806F81A	807F81A
760160	--	804F81C	805F81C	806F81C	807F81C
760170	--	804F81E	805F81E	806F81E	807F81E
760200	--	804F820	805F820	806F820	807F820
760210	--	804F822	805F822	806F822	807F822
760220	--	804F824	805F824	806F824	807F824
760230	--	804F826	805F826	806F826	807F826
760240	--	804F828	805F828	806F828	807F828
760250	--	804F82A	805F82A	806F82A	807F82A
760260	--	804F82C	805F82C	806F82C	807F82C
760270	--	804F82E	805F82E	806F82E	807F82E
760300	--	804F830	805F830	806F830	807F830
760310	--	804F832	805F832	806F832	807F832
760320	--	804F834	805F834	806F834	807F834
760330	--	804F836	805F836	806F836	807F836
760340	--	804F838	805F838	806F838	807F838
760350	--	804F83A	805F83A	806F83A	807F83A
760360	--	804F83C	805F83C	806F83C	807F83C
760370	--	804F83E	805F83E	806F83E	807F83E
760400	--	804F840	805F840	806F840	807F840
760410	--	804F842	805F842	806F842	807F842
760420	--	804F844	805F844	806F844	807F844
760430	--	804F846	805F846	806F846	807F846
760440	--	804F848	805F848	806F848	807F848
760450	--	804F84A	805F84A	806F84A	807F84A
764004	--	804FA01	805FA01	806FA01	807FA01
764014	--	804FA03	805FA03	806FA03	807FA03
764024	--	804FA05	805FA05	806FA05	807FA05
770460	--	804FC4C	805FC4C	806FC4C	807FC4C
772410	--	804FD42	805FD42	806FD42	807FD42
774400	--	804FE40	805FE40	806FE40	807FE40
777160	--	804FF9C	805FF9C	806FF9C	807FF9C
777440	--	804FFC8	805FFC8	806FFC8	807FFC8
777514	--	804FFD3	805FFD3	806FFD3	807FFD3

UNIBUS device Address		Equivalent DW780 Adapter "BYTE" address #0	#1	#2	#3
760010	--	2013E008	2017E008	201BE008	201FE008
760020	--	2013E010	2017E010	201BE010	201FE010
760030	--	2013E018	2017E018	201BE018	201FE018
760040	--	2013E020	2017E020	201BE020	201FE020
760050	--	2013E028	2017E028	201BE028	201FE028
760060	--	2013E030	2017E030	201BE030	201FE030
760070	--	2013E038	2017E038	201BE038	201FE038
760100	--	2013E040	2017E040	201BE040	201FE040
760110	--	2013E048	2017E048	201BE048	201FE048
760120	--	2013E050	2017E050	201BE050	201FE050
760130	--	2013E058	2017E058	201BE058	201FE058
760140	--	2013E060	2017E060	201BE060	201FE060
760150	--	2013E068	2017E068	201BE068	201FE068
760160	--	2013E070	2017E070	201BE070	201FE070
760170	--	2013E078	2017E078	201BE078	201FE078
760200	--	2013E080	2017E080	201BE080	201FE080
760210	--	2013E088	2017E088	201BE088	201FE088
760220	--	2013E090	2017E090	201BE090	201FE090
760230	--	2013E098	2017E098	201BE098	201FE098
760240	--	2013E0A0	2017E0A0	201BE0A0	201FE0A0
760250	--	2013E0A8	2017E0A8	201BE0A8	201FE0A8
760260	--	2013E0B0	2017E0B0	201BE0B0	201FE0B0
760270	--	2013E0B8	2017E0B8	201BE0B8	201FE0B8
760300	--	2013E0C0	2017E0C0	201BE0C0	201FE0C0
760310	--	2013E0C8	2017E0C8	201BE0C8	201FE0C8
760320	--	2013E0D0	2017E0D0	201BE0D0	201FE0D0
760330	--	2013E0D8	2017E0D8	201BE0D8	201FE0D8
760340	--	2013E0E0	2017E0E0	201BE0E0	201FE0E0
760350	--	2013E0E8	2017E0E8	201BE0E8	201FE0E8
760360	--	2013E0F0	2017E0F0	201BE0F0	201FE0F0
760370	--	2013E0F8	2017E0F8	201BE0F8	201FE0F8
760400	--	2013E100	2017E100	201BE100	201FE100
760410	--	2013E108	2017E108	201BE108	201FE108
760420	--	2013E110	2017E110	201BE110	201FE110
760430	--	2013E118	2017E118	201BE118	201FE118
760440	--	2013E120	2017E120	201BE120	201FE120
760450	--	2013E128	2017E128	201BE128	201FE128
764004	--	2013E804	2017E804	201BE804	201FE804
764014	--	2013E80C	2017E80C	201BE80C	201FE80C
764024	--	2013E814	2017E814	201BE814	201FE814
770460	--	2013F130	2017F130	201BF130	201FF130
772410	--	2013F508	2017F508	201BF508	201FF508
774400	--	2013F900	2017F900	201BF900	201FF900
777160	--	2013FE70	2017FE70	201BFE70	201FFE70
777440	--	2013FF20	2017FF20	201BFF20	201FFF20
777514	--	2013FF4C	2017FF4C	201BFF4C	201FFF4C

5.) ***** Read Data Substitute *(RDS)* Faults and Aborts *****

The Machine Check Logout information is not very good for this type of Machine Check. The associated Memory's status registers are more helpful for this type of problem.

MS780A and MS780C memories have error correction logic that can supposedly correct one bad bit per 72 bit array word. For read accesses that result in one bad bit being detected, the memories' error correction logic will correct the bad bit and will flag the data that is returned as "Corrected Read Data". If there are a multiple even number of bad bits detected during the 72 bit array read access, the data returned will be flagged as "Read Data Substitute". This indicates that the data has not been corrected and the quadword returned "may" be bad (the bits bad may have been in the ECC code so, therefore, the actual data returned may be good). BEWARE that MS780A & C memories cannot correctly detect and signal a multiple odd number of bad bits read from the 72 bit array. If this condition happens, the memory will send back the data and report it as "Corrected Read Data". It is, therefore, a good idea to swap out any arrays that are giving single bit errors for those types of problems that are intermittent and cannot seem to be fixed by other means.

This type of Machine Check means that a Double Bit Error has been detected, by memory, when the CPU was accessing a memory location.

Bit #13 of ID Register #19 should be set for this type of error to have occured.

"(SP)+16" contains a Virtual Address within the quadword location at fault. If the system has not been rebooted or disturbed, you may be able to use this "Virtual Address" in the following console command to find the Physical BYTE Address causing the error.

```
>>> E/L/V xxxxxxxx     ; where xxxxxxxx = contents of "(SP)+16".
```

The CONSOL.SYS program should respond with the following type of output:

```
P yyyyyyyy zzzzzzzz
```

Where "yyyyyyyy" is the Physical Address at fault. If you get a "Mic-err", the necessary PTE to make the Virtual to Physica translation isn't available from memory or the TB.

If this command was successful, you can use this Physical Address to determine what array is at fault. This address is a "Physical BYTE Address".

If you were not able to find the failing array by the procedure above, your only other choice is to use the "SYSTEM EVENT File" to see if any memory errors have been recorded.

Remember that the first array is array #0 not array #1.

MS780A & MS780C memories:

If bit <28> = 1, in "Memory Register C", then Bits <27:24> should reflect the array that had the error.

```
Memory Register "C"
-------------------
Bit <28>        =  Error Log Request
Bits <27:24>    =  Array Select
```

MS780E memories:

If bit <28> = 1, in "Memory Register C" or "Memory Register D", then bit <27> will indicate the controller and bits <26:24> will indicate the array within that controller that had the error.

```
Memory Registers "C & D"  ("C" - Lower Controller)
------------------------  ("D" - Upper Controller)

Bit <28>        -  Error Log Request
Bit <27>        -  Controller Select
Bits <26:24>    -  Array Select
Bits <23:22>    -  Array Bank Select
Bits <21:11>    -  RAM page address for 256K RAMs
Bits <19:11>    -  RAM page address for 64K RAMs
Bit <10>        -  Multiple bit error
Bit <09>        -  Single bit error detected and corrected
```

MA780 memories:

If bit <28> = 1, in the "Array Error Register", then bits <27:24>, of the same register, will indicate the Array in error.

```
Array Error Register
--------------------
Bit <28>        -  Error Log Request
Bit <08>        -  1 = Upper Word, 0 = Lower Word
Bits <22:09>    -  Chip address presented to the memory chip
Bit <23>        -  1 = Upper Bank, 0 = Lower Bank
Bits <27:24>    -  Array card with the error
```

Problem areas:

A Memory Array or the MEMORY Control.
Memory or CPU Backplane.
Memory or CPU Power.
SBI/CPU interface.
SBI cables.

6.) ***** VAX Micro-Code NOT SUPPOSE TO GET HERE *****

The "Trapped UPC" is about the only data saved, in the Machine Check Logout information, that may help you trouble-shoot this type of problem.

This type of Machine Check exception occurs whenever the microcode finds itself accessing a microcode location that it should never make it to. The unused microwords contain jumps that will direct the Micro-PC to the micro_routine that flags this error.

The Micro-stack register, ID #20, should contain the Control Store Address that the microcode wasn't suppose to get to. The microcode stores this register in ID #32 for a Double Error Halt and on the stack at "(SP)+12" on a Machine Check exception. Verify that the address is unused via the micro-fiche MICROCODE listing.

Problem areas:

Micro-code address logic (M8235).

Any board on the "micro_PC" bus.
- PCS (M8234).
- WCS (M8233 or M8238 in slot 20).
- OPTIONAL WCS (M8233 or M8238 in slot 18).
- IR Decode Logic.

WCSxxx.PAT on the LOCAL CONSOLE Floppy (or whatever floppy the WCS was loaded from).

WCS load path
(Floppy -> LSI -> CIB -> ID Bus -> WCS).

Clock Board (M8232).

CPU Power.

ID #20 - Micro Stack Register

```
3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
1 0 9 8  7 6 5 4  3 2 1 0  9 8 7 6  5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0
```

Reading this register pops the top address from the micro stack.
Writing this register pushes an address onto the micro stack.

Bits <15:00> Control Store Address <15:00>
****

<15:00> = micro_Address <15:00>

Micro PC Wirelist and Slot chart

Signal	Pin	Slot 18 Opt. WCS	Slot 20 WCS	Slot 22 PCS	Slot 23 M8235	Slot 08 M8224
Bus uPC 00	EA1	X	X	X	X	X
Bus uPC 01	ER2	X	X	X	X	X
Bus uPC 02	ES1	X	X	X	X	X
Bus uPC 03	EU2	X	X	X	X	X
Bus uPC 04	EV2	X	X	X	X	X
Bus uPC 05	FB2	X	X	X	X	X
Bus uPC 06	FD1	X	X	X	X	X
Bus uPC 07	FD2	X	X	X	X	X
Bus uPC 08	FE1	X	X	X	X	no
Bus uPC 09	FE2	X	X	X	X	no
Bus uPC 10	FF1	X	X	X	X	no
Bus uPC 11	FF2	X	X	X	X	no
Bus uPC 12	FH2	X	X	X	X	no

"WCS" and Opt. WCS" boards are either M8238's (2K) or M8233's (1K).

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

C P U D O U B L E E R R O R H A L T S

```
$

?CPU DBLE-ERR HLT
HALTED AT 8007E2A8

>>>
```

If the Problem is a "CPU Double Error Halt", "?CPU DBLE-ERR HLT" was printed on the console terminal, diagnosis is similiar to the "Machine Check" trouble-shooting. The difference is in where the information is stored by the VAX-11/780 CPU microcode. A "Double Error Halt" is simply a trap upon a trap.

On a "Double Error Halt" the logout information is stored in 2 places. The information for the first trap, "Machine Check", is stored in ID Bus registers 30 thru 39, and the information for the second trap, "Machine Check", is stored in the associated ERROR/STATUS Registers and the Memory NEXUS registers.

It is, therefore, very important to take a dump of all the Processor ID Bus Registers and all the Memory NEXUS registers at the time of the crash, before any other commands are given. This can be done with the following CONSOL.SYS commands:

```
>>> E/L/H/ID/N:17 0
>>> R E/L/H/ID 18

;allow CONSOL.SYS to do at least 15 examines, then type "^C".

        ^C
>>> E/L/H/ID/N:25 19
>>> E/L/H/P/N:x 200yy000          ; x  = depends on memory type
                                  ; yy = depends on Mem TR level

; Repeat the last command, changing "x" and "yy" as needed
; in order to gather all the Memory NEXUS registers.

>>> E/L/H/P 200yy000              ; yy = depends on TR level

; Repeat the last command, changing "yy" as needed, in order
; to obtain the contents of all NEXUS Configuration Registers.
```

An easier method to dump the needed information would be to use a "DUMP." CONSOL.SYS command file, built as outlined in Chapter 3 of this manual.

Unlike "MACHINE CHECKS", "Double Error Halts" bring the VAX completely down to a HALT. Control passes back to the VAX-11/780 CONSOL.SYS program. Therefore, the System Event file, ERRLOG.SYS, will not contain information at the time of the crash. ERRLOG.SYS may, however, contain some pertinent information about something that happened just prior to the crash, (such as a Double Bit Error in Memory). If you have not isolated the problem by examining the Hardware Registers, it may be worth your time to try to bring the VMS Operating System back up and examine the Error log file.

The information for the first error of a DOUBLE ERROR HALT will be found in the Temporary Registers, ID Registers #30 thru #39 (see note). The information for the second error of a DOUBLE ERROR HALT will be found in the associated error/status registers. Therefore, it is very important to examine all the Hardware Registers in order to trouble-shoot Double Error Halts. The Hardware Register Dump must be

taken immediately before anything else is done, in order to assure that the Register Contents are valid for the time of the error.

Note: ***** If the second error was a "Control Store Parity Error" or a "Micro-sequencing Error", the information in "T0-T9" MAY NOT BE VALID for the first error. The safest thing to due is to check ID #0C, see if bit <15>=1, and IF IT IS DO NOT USE "T0-T9" (which are ID #30-39). If the second error was a "Micro-sequencing Error", there will not be any other error bits set. In either case, if the SECOND error is found to be either a "Control Store Parity Error" or "Micro-sequencing Error", the information in T0 thru T9 may not be valid.

Use the MACHINE CHECK outline to trouble-shoot the first error. The only difference is that the LOGOUT information is found in "ID 30" thru "ID 39" instead of on the stack. These ID registers must be dumped by you, or the customer, prior to anything else being done.

Examine the rest of the ID registers and all the Memory NEXUS hardware registers in order to determine what the second error was. It is best to determine what the second error was prior to checking the first error since the information for the first error may not be valid, due to the second error occuring before the "T0-T9" logout was completed. i.e. "Control Store Parity Error" or "Micro-Sequencing Error".

If Bits <19>, <17>, & <16> of ID #1B are equal to a 1, then an S.B.I. FAULT has occured. Use ID #1B, the S.B.I. Silo dump, and the Configuration registers in the NEXUS devices, to determine the cause of the FAULT.

DOUBLE ERROR HALT Information

Description	Register	1st Error ID Location	2nd Error ID Location
Summary Parameter	T0	30	none
CPU Error Status	T1	31	0C
Trapped UPC	T2	32	20
VA/VIBA	T3	33	none
D Register	T4	34	08
TB Error 0	T5	35	12
TB Error 1	T6	36	13
Timeout Address	T7	37	1A
Parity	T8	38	1E
SBI Error	T9	39	19
Fault Status	none	none	1B

If the second error is caused by a RDS error, then the associated memory registers will reflect the array in error.

BEWARE: The 1st error information MAY NOT BE VALID if ID ≠0C Bit<15> = 1, or if a micro-sequencer problem has occured.

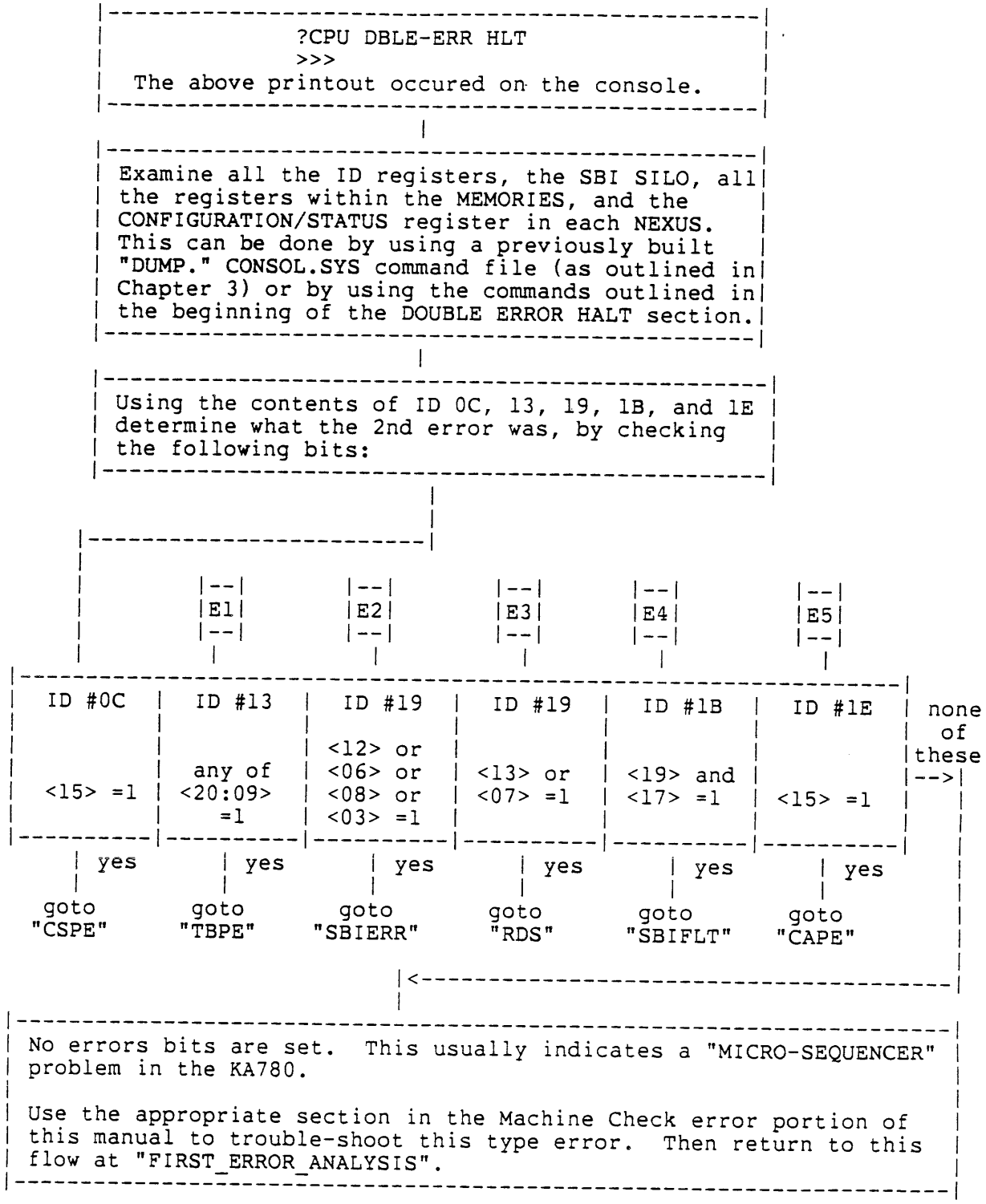
Due to the possibility of the processor detecting a non-existent error condition, it is a good idea to constantly make certain validity checks of the error information that you have gathered.

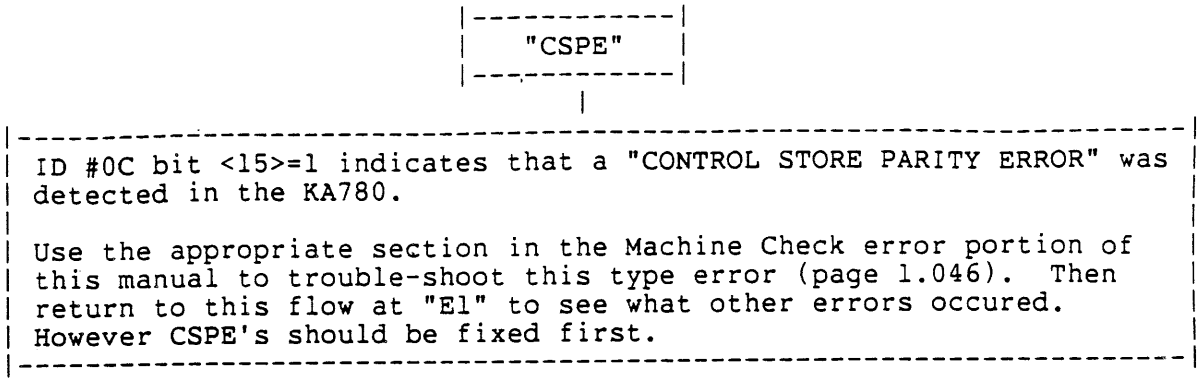
In the case of "Double Error Halts", some of the error information may not be correctly stored away due to a second error occuring while the first error is being stored. Therefore, you should always make a validity check of the information stored in ID #30:39 (the 1st error). It is a good idea to always make validity checks on the errors.

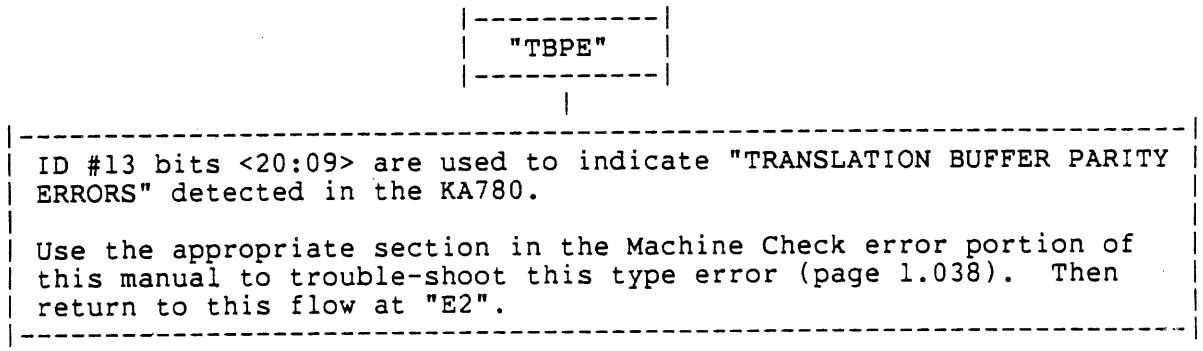
CPU Detected Error VALIDITY CHECKS:

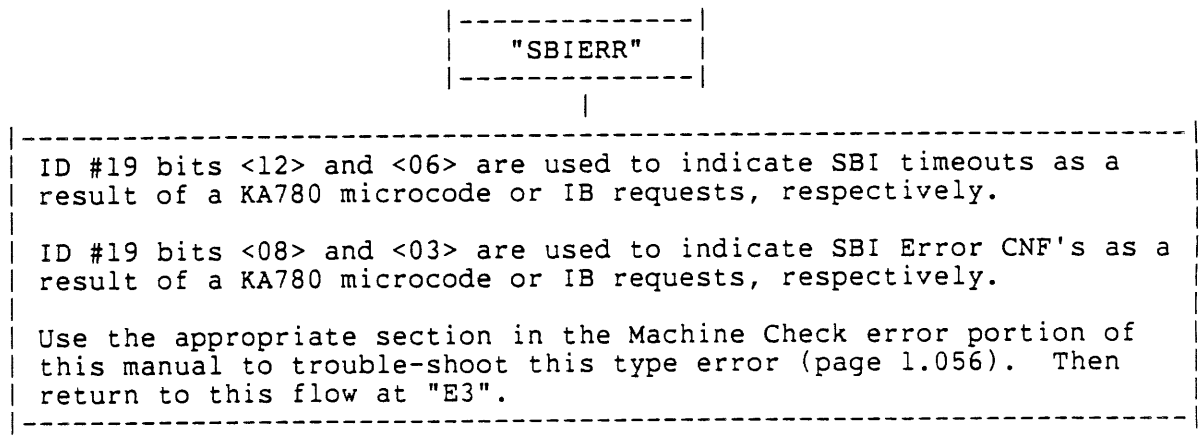
Control Store Parity Error --	"CPU Error Status Register", ID #0C, must have Bit <15>=1.
CP/IB Read Timeouts --------	"SBI Error Register", ID #19, must have either Bit <12>=1 or Bit <06>=1.
CP/IB Error Confirmations ---	"SBI Error Register", ID #19, must have either Bit <08>=1 or Bit <03>=1
CP/IB RDS Faults ----------	"SBI Error Register", ID #19, must have Bit<13>=1 or Bit <07>=1.
TB Parity Errors ----------	"Translation Buffer Register #1", ID #13, must have at least one of Bits <20:09>=1.
Cache Parity Errors ---------	"Cache Parity Register", ID #1E, must have Bit <15>=1.
S.B.I. Fault ----------	"SBI Fault Status Register", ID #1B, must have Bit <19>=1, and Bit <17>=1. Also, at least one of the NEXUS should have at least one of Bits <31:27> set(=1) in their associated Configuration Status Registers. The VAX-11/780 is also a NEXUS and its equivalent register is ID #1B.

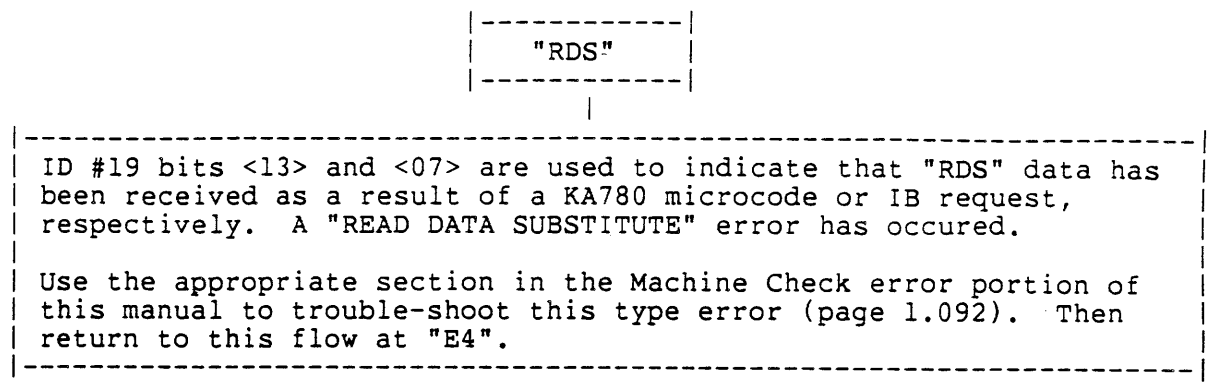
CPU Double Error Halt Flowchart

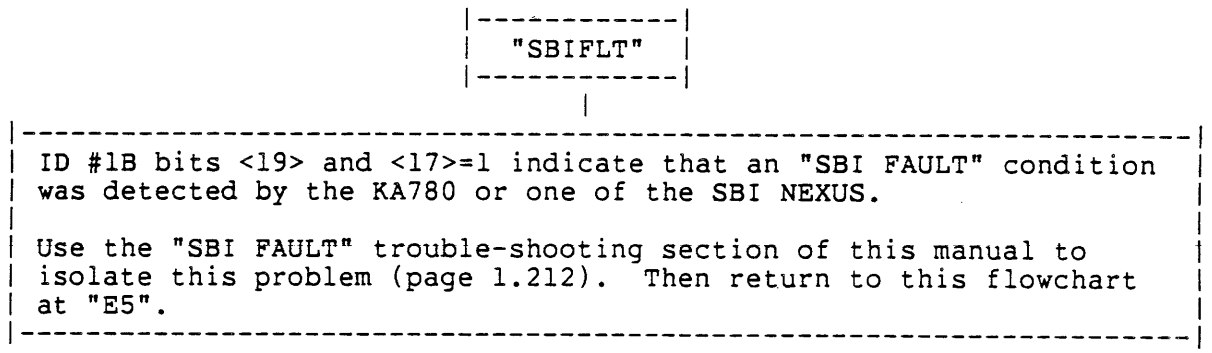


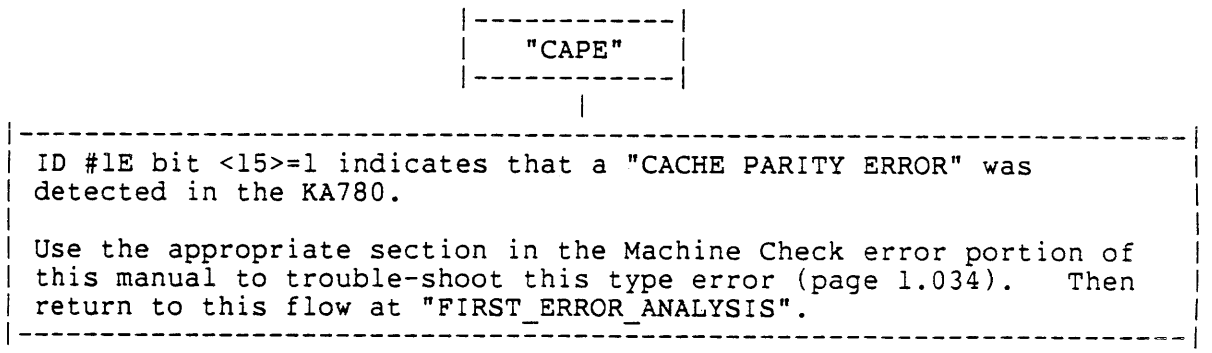

"CSPE"
ID #0C bit <15>=1 indicates that a "CONTROL STORE PARITY ERROR" was detected in the KA780.
Use the appropriate section in the Machine Check error portion of this manual to trouble-shoot this type error (page 1.046). Then return to this flow at "E1" to see what other errors occured. However CSPE's should be fixed first.

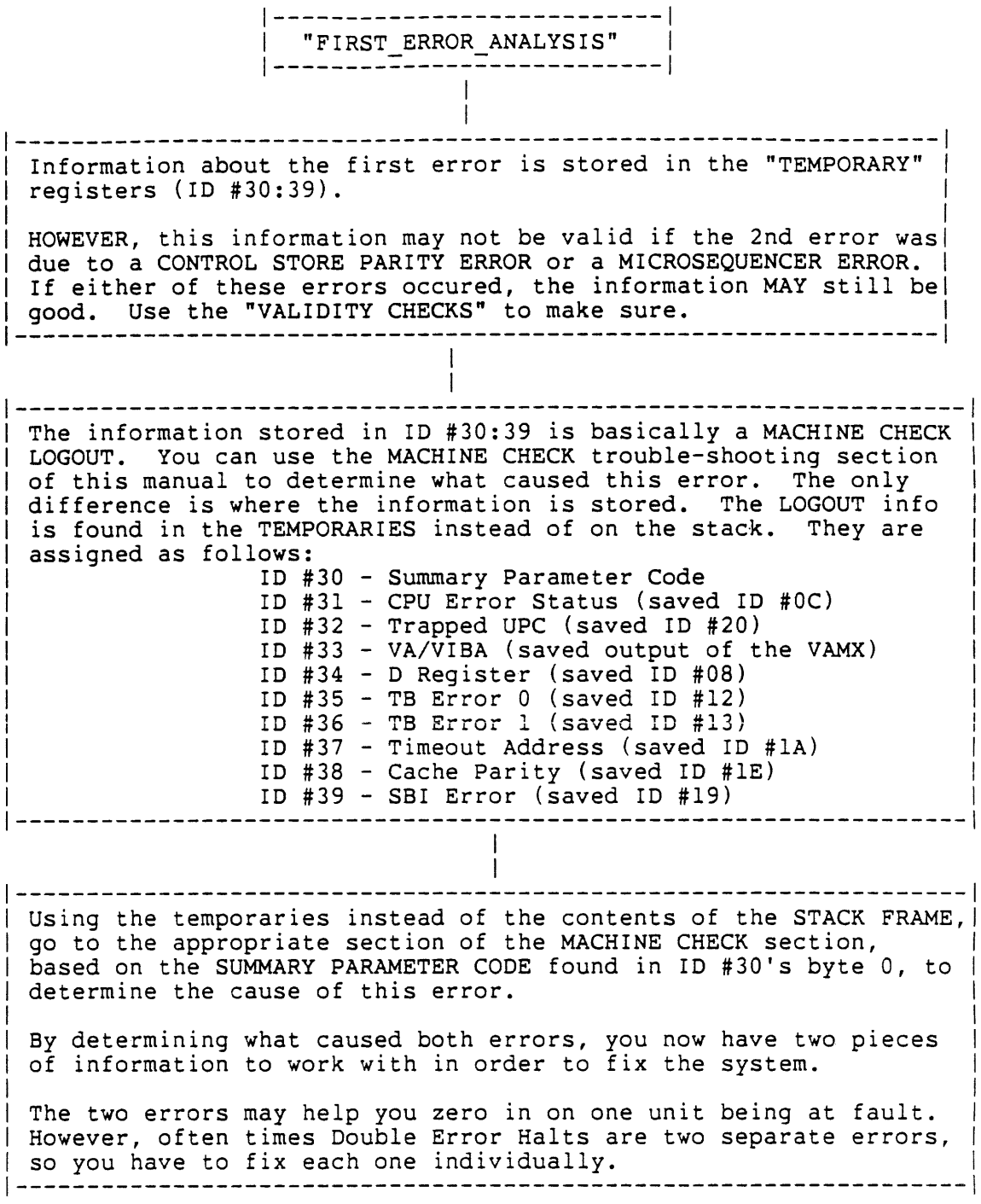

"TBPE"
ID #13 bits <20:09> are used to indicate "TRANSLATION BUFFER PARITY ERRORS" detected in the KA780.
Use the appropriate section in the Machine Check error portion of this manual to trouble-shoot this type error (page 1.038). Then return to this flow at "E2".


"SBIERR"
ID #19 bits <12> and <06> are used to indicate SBI timeouts as a result of a KA780 microcode or IB requests, respectively.
ID #19 bits <08> and <03> are used to indicate SBI Error CNF's as a result of a KA780 microcode or IB requests, respectively.
Use the appropriate section in the Machine Check error portion of this manual to trouble-shoot this type error (page 1.056). Then return to this flow at "E3".


"RDS"
ID #19 bits <13> and <07> are used to indicate that "RDS" data has been received as a result of a KA780 microcode or IB request, respectively. A "READ DATA SUBSTITUTE" error has occured.
Use the appropriate section in the Machine Check error portion of this manual to trouble-shoot this type error (page 1.092). Then return to this flow at "E4".

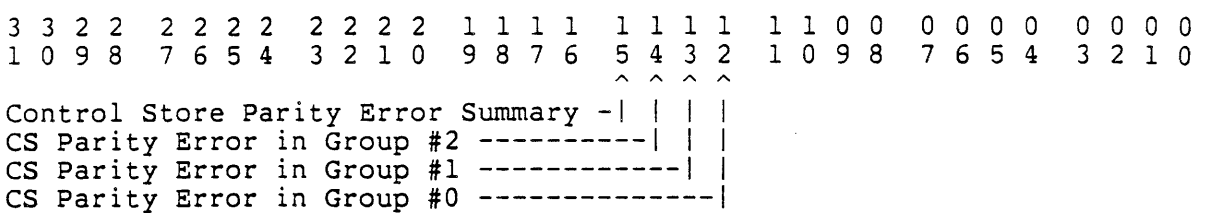

"SBIFLT"
ID #1B bits <19> and <17>=1 indicate that an "SBI FAULT" condition was detected by the KA780 or one of the SBI NEXUS.
Use the "SBI FAULT" trouble-shooting section of this manual to isolate this problem (page 1.212). Then return to this flowchart at "E5".


"CAPE"
ID #1E bit <15>=1 indicates that a "CACHE PARITY ERROR" was detected in the KA780.
Use the appropriate section in the Machine Check error portion of this manual to trouble-shoot this type error (page 1.034). Then return to this flow at "FIRST_ERROR_ANALYSIS".

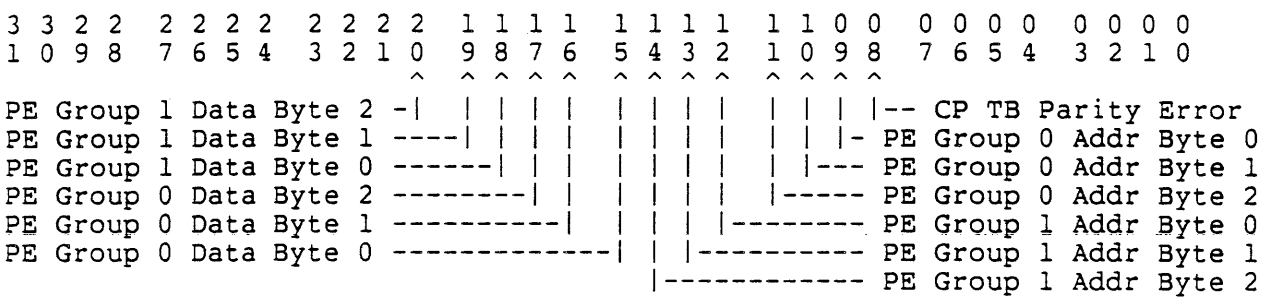

"FIRST_ERROR_ANALYSIS"
Information about the first error is stored in the "TEMPORARY" registers (ID #30:39).
HOWEVER, this information may not be valid if the 2nd error was due to a CONTROL STORE PARITY ERROR or a MICROSEQUENCER ERROR. If either of these errors occured, the information MAY still be good. Use the "VALIDITY CHECKS" to make sure.
The information stored in ID #30:39 is basically a MACHINE CHECK LOGOUT. You can use the MACHINE CHECK trouble-shooting section of this manual to determine what caused this error. The only difference is where the information is stored. The LOGOUT info is found in the TEMPORARIES instead of on the stack. They are assigned as follows:
ID #30 - Summary Parameter Code
ID #31 - CPU Error Status (saved ID #0C)
ID #32 - Trapped UPC (saved ID #20)
ID #33 - VA/VIBA (saved output of the VAMX)
ID #34 - D Register (saved ID #08)
ID #35 - TB Error 0 (saved ID #12)
ID #36 - TB Error 1 (saved ID #13)
ID #37 - Timeout Address (saved ID #1A)
ID #38 - Cache Parity (saved ID #1E)
ID #39 - SBI Error (saved ID #19)
Using the temporaries instead of the contents of the STACK FRAME, go to the appropriate section of the MACHINE CHECK section, based on the SUMMARY PARAMETER CODE found in ID #30's byte 0, to determine the cause of this error.
By determining what caused both errors, you now have two pieces of information to work with in order to fix the system.
The two errors may help you zero in on one unit being at fault. However, often times Double Error Halts are two separate errors, so you have to fix each one individually.

VAX-11/780 "ID" Register ERROR information

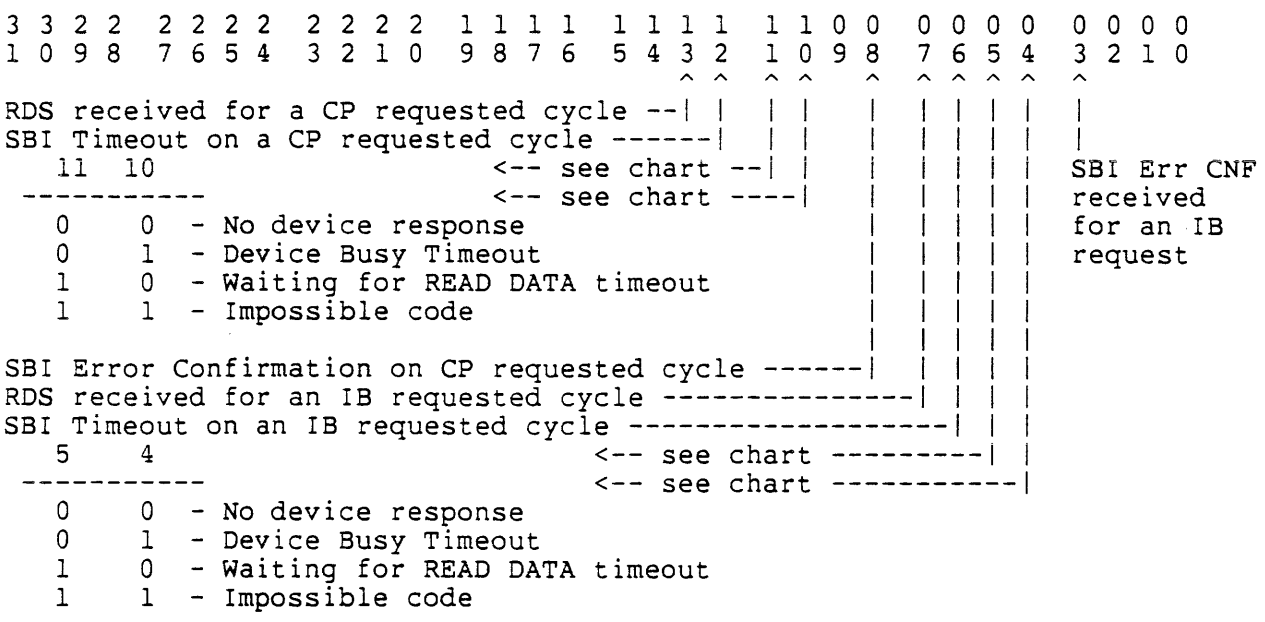
ID #0C - *CES*



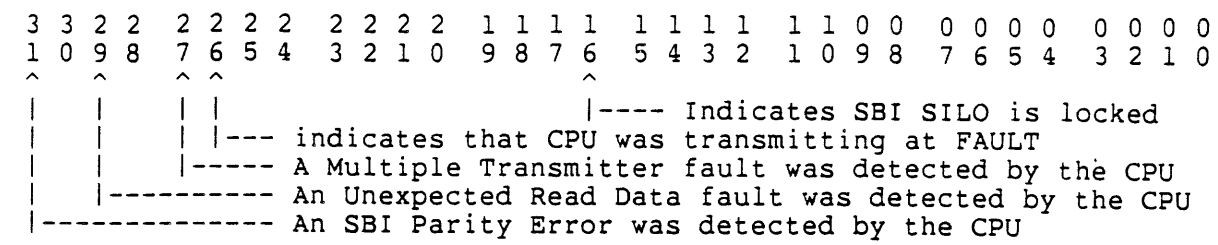
ID #13 - *TBER1*



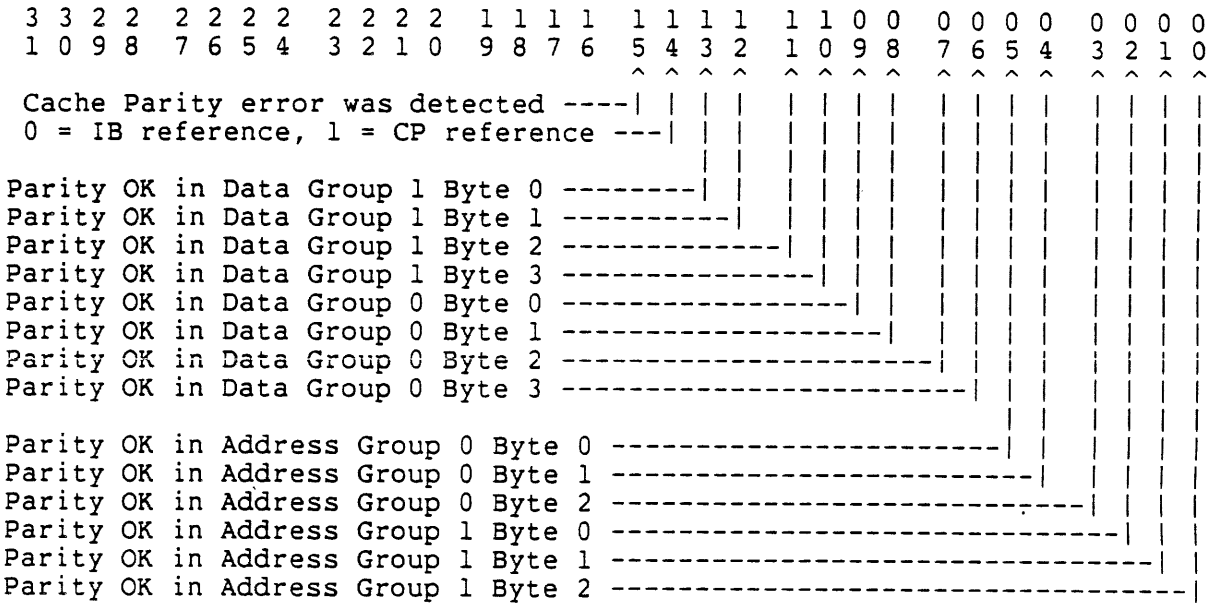
ID #19 - *SBI.ERR*



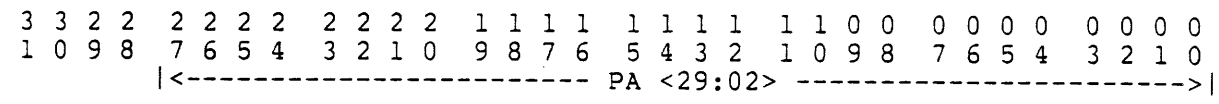
ID #1B - *FAULT*



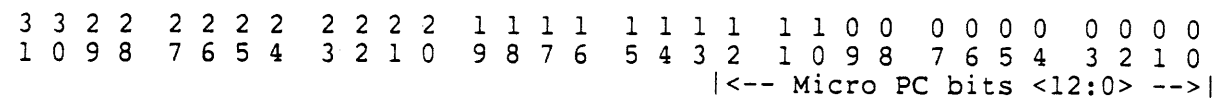
ID #1E - *CACHE PARITY*



ID #1A - *TIMEOUT ADDRESS*



ID #20 - *MICRO STACK*



Example of a Double Error Halt and a hardware dump

```
      ?CPU DBLE-ERR HLT
      HALTED AT 8007E2A8
>>>E/L/H/ID/N:3F 0
                                (1st Error)
      ID 00000000  8F590908   CP RDS Fault -> ID 00000030  00000005
      ID 00000001  5CFAB525   Summary Code    ID 00000031  00000002
      ID 00000002  00000000                   ID 00000032  00001116
      ID 00000003  013A0260                   ID 00000033  00000040
      ID 00000004  00000040                   ID 00000034  7FFD7130
      ID 00000005  00000000                   ID 00000035  00007C81
      ID 00000006  00000040                   ID 00000036  00000000
      ID 00000007  00000000                   ID 00000037  00001FA3
      ID 00000008  00000000                   ID 00000038  00004000
      ID 00000009  00000000 CP RDS bit set -> ID 00000039  0000A002
      ID 0000000A  000080C1                   ID 0000003A  00021074
      ID 0000000B  FFFFEA96                   ID 0000003B  0007CE00
      ID 0000000C  00000184                   ID 0000003C  00000000
      ID 0000000D  03630054                   ID 0000003D  003FFDE9
      ID 0000000E  001A0000                   ID 0000003E  00000800
         041F0000                             ID 0000003F  00080000
      ID 00000010  00007C41          >>> E/L/P/N:2 20002000
      ID 00000011  00000000                P  20002000  00002E10
      ID 00000012  00007C41                P  20002004  F0001400
      ID 00000013  00000000                P  20002008  3B080200
      ID 00000014  00000000                              ^
      ID 00000015  00000000           >>>                |
      ID 00000016  00000000                        Array #11.
      ID 00000017  00000000                        had an error
      ID 00000018  00000000
      ID 00000019  0000A082 <-- CP RDS bit set (2nd Error)
      ID 0000001A  0001F8AA
      ID 0000001B  02040000
      ID 0000001C  00000000
      ID 0000001D  0021C000
      ID 0000001E  00004000
      ID 0000001F  FFFF0000
      ID 00000020  00000FCF
      ID 00000021  00000000
      ID 00000022  00000000
      ID 00000023  00000030
      ID 00000024  800D7000
      ID 00000025  7F0DF000
      ID 00000026  0007E000
      ID 00000027  33333333
      ID 00000028  7FFEAD1C
      ID 00000029  7FFEBD04
      ID 0000002A  7FFED4FC
      ID 0000002B  0000C6C0
      ID 0000002C  800C3C00
      ID 0000002D  00000B80
      ID 0000002E  00000000
      ID 0000002F  00000B80
```

1st and 2nd error indicate that "Read Data Substitute" data was received, by the CPU, on a CP request. The registers within the memory at TR#1 indicate that Array #11. was the array at fault.

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

INTERRUPT STACK NOT VALID Halts

```
$

?INT-STK INVLD
HALTED AT 8007E2A8

>>>
```

"INTERRUPT STACK NOT VALID halts" are exceptions that indicate that the interrupt stack was not valid or that a memory error occurred while the processor was pushing information onto the stack during the initiation of an exception or interrupt. In other words, an "Interrupt Stack Not Valid" means that, while pushing information onto the STACK a reference was made to a Virtual Address not currently mapped to Physical Memory or that a Fatal Error occurred while referencing the STACK. No further interrupt requests are acknowledged on this processor.

This problem is detected by VAX CPU microcode, which tells the "LSI Front-end Subsystem", which will halt the VAX CPU and print out the "?INT-STK INV" message. For this reason, the VAX software will not have been able to take a Software Dump as the system crashes. In order to get a Software dump, the "AUTO RESTART" switch must be "on", or, the "RESTAR.CMD" indirect command file can be used to restart the operating system. The RESTAR.CMD file will attempt to reboot the system but will fail, therefore allowing a software dump to be taken. In either case, the "RESTAR.CMD" command file should have been previously modified to cause a dump (such as done by the "DUMP." indirect command file) to be taken prior to rebooting.

This type of problem can be caused by any number of things but listed below are the most common reasons:

a. Memory Errors

    1. Double Bit Errors (Uncorrectable errors)
    2. Hardware problems causing NXM (Non-existent Memory) errors.
    3. SBI interface in VAX-11/780 CPU or Memory has problems.

b. Some device interrupting excessively.

c. Memory Management or System Disk problems.

d. The contents of "SP" and "Internal Register #4" should be equal. If they are not, the problem is probably the M8229 or M8225.

The following information should be used to trouble-shoot the "Interrupt Stack Not Valid" problem:

a. A "Hardware Register Dump" should be used to see if any hardware errors have occured.

b. The contents of the Stack should be dumped for further examination. This should be available from the "Hardware Register Dump", if it is set up as the "DUMP." example in this manual. Look for repeated Machine Check Logouts or Exceptions on the Stack. This could indicate the hidden reason for the "INT STK INV".

c. If the VMS Operation System is operative, an Error Log report should be taken at least of the time immediately prior to and at the time of the failure. Does it show any errors logged?

d. A Software Dump should have been taken. This dump can be analyzed by either RDC, Remote Support, D.E.C. Software Support, your District Support Group, or yourself. It may be necessary to examine several dumps in order to see what is commonly happening or what device is commonly being accessed.

This type of problem can be caused by Non-D.E.C. Supported Device drivers. Find out from the Customer if any new drivers have been installed recently or if some new foreign equipment has recently been added to the system. If one or more of these have recently been added to the system, see if the problem occurs when these devices are no longer used.

If the Problem is of intermittent nature, it is may be faster to trouble-shoot this type of problem from a software approach. The Software Dumps may not point a finger directly at a unit or subsystem, but will at least let you know what was happening at the time of the crash. This information along with any Hardware register dumps that are available, may point you to a unit or subsystem.

The following should be done in order to enable gathering of the information that is needed to trouble-shoot intermittent type INTERRUPT STACK not VALID's:

1. Make sure that the SYSGEN Parameter "DUMPBUG" is set (=1). This will cause a Software Dump to be taken, (on the way back up for "?INT-STK INV's").

2. Have the Customer take a Hardware Register Dump when the System Crashes. After the Hardware Dump is taken, the Customer can then reboot the Operating System. This step will be automatically taken care of if the "RESTAR.CMD" file has been modified to include the "DUMP." commands.

3. Have the Customer save the Software Dump when the System is rebooted after a crash. This can either be saved on MAGTAPE or the "SYS$SYSTEM:SYSDUMP.DMP" can be "Copied" to another file.

4. An ERROR LOG report should be taken for the time just prior to and at the time of the crash.

5. Have the Customer save the Hardware Dump Output, the Software Dump, the Console Terminal Output at the time of the crash, and the ERROR LOG report for you to examine.

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

KERNEL STACK NOT VALID Aborts

"KERNEL STACK NOT VALID ABORTs" are exceptions that indicate that the Kernel Stack was not valid while the processor was pushing information onto the stack during the initiation of an exception or interrupt. Usually this is a indication of stack overflow or another executive software error. The attempted exception is transformed into an abort that uses the interrupt stack. No information other than the PSL and PC is pushed onto the Interrupt Stack. Software may abort the process without aborting the Operating System; however, because of the lost information, the process cannot be continued. If the Kernel Stack is not valid during the execution of an instruction, the processor initiates a normal Memory Management fault, and if the exception vector <1:0> for Kernel Stack not Valid is 0 or 3, the behavior of the processor is undefined. If the problem is of an intermittent nature, certain things should be done in order to enable gathering of the information needed to aid problem diagnosis. If the problem is solid, you should be able to gather most of the following information without the aid of the customer. In either case, the following steps should be taken:

1. Make sure that the SYSGEN Parameter "BUGREBOOT" is cleared (=0). This will cause the Operating System to halt after a FATAL Bugcheck.

2. Make sure that the SYSGEN Parameter "DUMPBUG" is set (=1). This will cause a Software Dump to be taken as the Operating System is coming down.

3. Have the Customer take a Hardware Register Dump when the System Crashes. After the Hardware Dump is taken, the Customer can then reboot the Operating System.

4. Have the Customer save the Software Dump when the System is rebooted after a crash. This can either be saved on MAGTAPE or "SYS$SYSTEM:SYSDUMP.DMP" can be "Copied" to another file.

5. An ERROR LOG report should be taken for the time just prior to and at the time of the crash.

6. Have the Customer save the Hardware Dump Output, the Software Dump, the Console Terminal Output at the time of the crash, and the ERROR LOG report for you to examine.

NOTE: Steps #1 and #3 may be eliminated if the "DEFBOO.CMD" command file has been modified so that it will dump all the Hardware Registers. The "REMOTE LOCAL CONSOLE" floppy should be used in case you should deceide to use the "Remote Diagnostic Center" as a tool for problem diagnosis. The "DEFBOO.CMD" and "RESTAR.CMD" command files should be modified, on this floppy, so as to take a hardware register dump upon reboot. The "DUMP." and "HANG." files should also be installed on this floppy.

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

OTHER TYPES of CRASHES

There are many other types of Crashes that can occur. The basic trouble-shooting method for them should be as follows:

a. Obtain an Error log report of the Systems Events that happened prior to and at the time of the crash. This report will many times show the error.

b. Examine the Console Terminal printout at the time of the crash.

c. Examine the Hardware Register Dump, if taken. If it wasn't taken, try to recreate the problem and make sure the Hardware Register Dump is taken.

d. If a Software Dump was taken, examine it to determine what was happening at the time of the failure. If you do not know how to analyze a Software Dump, get either D.E.C. Software or Remote Support to analyze it for you.

e. After making a preliminary analysis of the problem from the above information, run all diagnostics on the device, and associated controllers, that you feel may be at fault. If none of these fail, run diagnostics on everything that may be remotely related to the problem. It doesn't hurt to spend the time to run all diagnostics for that particular system configuration.

f. Using both a SCOPE and a DVM, check the VOLTAGEs and the POWER MONITORING signals (ACLO & DCLO, etc.) for both the correct level and for the amount of NOISE riding on the voltage levels. Correct any that are out of specification.

   Power and Power Monitoring Signal problems cause many strange problems that can lead you around in circles for quite awhile. Never overlook these. Always check them no matter what type of problem you have.

g. Margining, heating, cooling, and vibrating may be used to recreate and isolate some problems.

h. Many times problems are intermittent and diagnosis is not possible on the first crash. If this is the case, try to obtain as many of the following things as are possible and take them to your District Support Group so that they may aid you in diagnosing the problem:

1. Console Terminal output just prior to and at the time of the crash.

2. Hardware Register Dump printout.

3. Error Log report. If you are using "SYE", get a "STANDARD" printout. If you are using "SPEAR", get the "FULL" "RETREIVE" printout and also the "ANALYZE" output.

4. Get an "SDA" output from the examination of the Software Dump that contains at least the following:

    a. SHOW CRASH
    b. SHOW PROCESS/ALL
    c. SHOW STACK/ALL
    d. SHOW DEVICE
    e. SHOW PFN_DATA/ALL
    f. SHOW SUMMARY
    g. EXAMINE/P0

5. Get a copy of the Software Dump on Magtape at 1600 B.P.I. if possible.

6. If a copying machine is available, copy your LOG Book entry that tells the problem symptoms that you gathered.

i. If the problem is not a solid problem and if the SYSGEN parameters are set up so that the Operating System reboots, ask the customer to change them so that the system does not reboot automatically and educate the customer on the procedure for taking a hardware registe dump.

The "REMOTE LOCAL CONSOLE" floppy should be used in case you should deceide to use the "Remote Diagnostic Center" as a tool for problem diagnosis. The "DEFBOO.CMD" and "RESTAR.CMD" command files should be modified, on this floppy, to take a hardware register dump upon reboot. The "DUMP." and "HANG." command files should also be installed on this floppy.

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

VMS OPERATING SYSTEM Hangs

Hangs are perhaps the hardest problems to diagnose. A HANG can occur due to:

1. Software is stuck in a loop waiting for a certain event event, or interrupt, to happen.

2. Hardware has failed in such a way that an asynchrounous event didn't occur so that normal execution can proceed.

Diagnostic Hangs can also be trouble-shot in much the same way as for an Operating System Hangs, except that Software Dumps cannot be taken when running diagnostics in stand-alone mode.

Proceed to trouble-shoot a Hang as follows:

1. If you are trouble-shooting an Operating System Hang, determine if the whole system is hung. Record your findings in the Log.

   a. Does the Console Terminal respond?

   b. Does any other terminal respond?

   c. Are any of the Peripherals doing anything?

      1. Are the disks seeking occassionally?
      2. Are the tapes moving?
      3. Is the printer printing?
      4. Etc.

2. Take a Hardware Register Dump by typing a "^P" on the Console terminal and then examining the registers by using the "HANG." command file. Console Terminal commands to do this would be:

```
          ^P
>>> @HANG
```

   If typing a "^P" does not put you back into "CONSOL" mode, check the following:

   a. Is the KEY Switch on the VAX-11/780 Front panel in one of the "DISABLE" positions? If so, turn the KEY to "LOCAL". and retry the "^P".

   b. Check the "DC ON" LED and the "RUN" LED on the Console Subsystem LSI-11 Front Panel. If they are not lit, your problem is in the Console Subsystem or is a Power/ACLO/DCLO problem. If both LED's are lit, proceed to next check.

c. If neither of the above are the problem, then the problem is either the Console Terminal or the LSI-11/DLV11 subsystem. Be sure that the console terminal is not in "LOCAL" or out of paper.

If you are unable to get a response when typing "^P", trouble-shoot this problem.

3. After the Hardware Register Dump has been taken, the "HANG." command file will single step the VAX several times (so that it may be determined where the software is hung), and then it will crash the software as done by the "CRASH." command file. This will insure that a Software Dump is taken.

   If the SYSGEN parameter "BUGREBOOT" is set (=1), the system will reboot automatically. You will then have to bring the system back down in order to run diagnostics.

   The single stepping portion of the "HANG." output may indicate a reason for the hang. Check for one of the following:

   a. A PC = 80002EB0 (Version 3.x of VMS) indicates that the DW780 is getting a UNIBUS vector of "000000".

   b. A PC = 80007B06 (a NULL job address in Version 3.x of VMS) indicates that a Software resource is exhausted.

   c. A PC = 80016400 (Version 3.x of VMS), with an IPL of 14-17 or 8-B, usually means the software is executing a driver.

   d. An IPL of "1F" indicates a SYSTEM DISK ERROR or MEMORY Power problem.

   e. A PC without bit 31 or 30 set usually indicates a process that is compute bound and running at a high priority.

   f. A Loop that goes through about 10 addresses close together, then jumps to a new range of address for about 10 instructions, then back to the first range. This condition usually indicates a terminal, DZ11, DMF32, etc., type of problem.

4. Using a DVM, check all System Voltage levels and the levels of all ACLO and DCLO signals. Are any out of spec.?

5. Using a Scope, check all voltages, ACLO signals, and DCLO signals for excessive noise. Be sure to use a good Ground on your scope lead.

6. Run at least the following diagnostics:

   a. VAX-11/780 micro diagnostics (#1, #2, and #3 if applicable)
   b. EVKAA (if the "DS>" prompt appears when this program is started, deposit zero into physical location "FE00" and restart)
   c. EVKAB,C,D,E
   d. ESCAA
   e. ESCBA
   f. Disk, Tape, and Unibus peripheral Reliability dignostics.

   If you get a diagnosic failure, trouble-shoot that problem.

   After the diagnostics all run O.K., continue on to the next step. It is important not to assume the HANG problem to be fixed at this time. You may have fixed another problem other than the one you were initially trouble-shooting.

7. If you are trouble-shooting an Operating System Hang, attempt to reboot the system at this point and, if you are successful, take an Error Log report of the time prior to and at the time of the Hang.

8. Attempt to diagnose the problem. Use whatever D.E.C. resources you need to analyze the information that you have gathered. If you have found a problem before you got to this step, you may have fixed the Hang problem. Do not assume this yet. Keep all the information that you have gathered so far with the SYSTEM LOG Book, just in case the HANG problem reoccurs.

   The Software Dump can be analyzed by RDC, Remote Support, Software Support, or District/Regional Support.

   On the Hardware Dump examination, look for such things as:

   1. Attentions on Massbus Devices.
   2. Adapter Power "UP" or "DOWN" status.
   3. Interrupt enables having been cleared, which may indicate power glitches or problems with the power monitoring logic.
   4. Who was interrupting at the time the Hardware Register Dump was taken?
   5. Any error bits set?

If the problem is not a solid problem and if the SYSGEN parameters are set up so that the Operating System does reboot, ask the customer to change them so that the system does not reboot automatically and educate the customer on the procedure for taking a hardware register dump.

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

OPERATING SYSTEM Functional Problems

"Operating System Functional problems" are those problems that do not crash the Operating System but either do not complete properly or do the wrong thing even though they appear to be working properly.

In order to diagnose the problem, you will first need to know a few things about the problem. Such as:

1. Is the problem a result of running D.E.C. supported software or Customer software? This will indicate to you how far you need to persue the problem. We are not responsible for fixing Customer Software or even defining where in the Customer's software the problem lies. We, D.E.C., are only responsible in verifying that the D.E.C. hardware and D.E.C. Supported Software are not at fault.

2. Can the problem be recreated at will. It will probably be necessary to recreate the problem in order to trouble-shoot it.

3. If the problem cannot be recreated at will, what is the Time Between Failures.

4. You will need to know at what time the last failure occured. If the customer doesn't know, then the problem will have to be recreated so that you will know at what time to look for errors in the error log file.

These problems may or may not be caused by Hardware. In order to determine if the problem is Hardware related, check the following:

1. Take an Error log report that covers the time immediately prior to, at time of, and immediately after the "Failing Function" was attempted. Does the report show any errors or strange events?

2. If the "Failing Function" uses a particular device, run the appropriate diagnostics on that device.

3. If the "Failing Function" uses a particular device, check the Voltages and AC/DCLO signals (if appropriate) on that device.

4. Check with Remote/District/Regional Support to see if this is a known or common problem. There might be a Hardware or Software fix for this problem. D.E.C. RDC is also a good place to check to see if the problem is similiar to any known or common problems.

If the problem is not found to be a hardware problem, (after doing the above checks), it may be necessary to get the help of D.E.C. Software in order to find out how to diagnose the problem.

Note: If all else fails to fix your problem, the VAX-11/780 Data Paths may be at fault.

The VAX-11/780 Data Paths, as with most processors, do not check parity within themselves as the data moves around within the data path elements. This is not done since "parity checkers" are extremely slow when compared to the speed needed within the data paths.

Parity is checked on the "MD Bus" data coming into the Data Paths by the Cache logic. The Data Paths generates parity for the data that it is sending out of the "D Register". Data going into and out of the other Data Path buses, the ID Bus and the VA Bus, does not have parity checking or generation done by the Data Paths.

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

OPERATING SYSTEM BACKUP or REBUILD Problems

Backup or System Rebuild problems are often considered to be a seperate type of problem. There are really only a few areas that may be at fault. Check the following:

a. Check to see if the problem can fit into one of the other "Types of Problems" listed at the beginning of this Trouble-shooting Outline. If it does fit under another type, use that types' outline to trouble-shoot the problem.

   An example would be, while attempting to do a stand-alone backup or restore, the system crashed with "?INT-STK INVLD". If this was the case, you should go to the "Interrupt Stack Not VAlid" flow which is under the "Operating System Crashes or Bugchecks" section.

b. For "Stand-Alone" Backup or Restore, the following devices are used :

   1. LSI-11 Subsystem
   2. VAX-11/780 CPU and MEMORY
   3. Disk Drive that contains media being used.
   4. Magtape that contains media being written to or read from.
   5. Associated SBI Nexus for Disk Drive and Tape Drive.
   6. SBI Terminator

   These devices should be checked for correct operation by testing with the appropriate diagnostics.

   Don't forget to check the voltages and Power Monitoring signals (AC/DCLO) for these devices.

   Other System units may affect the operation of these devices even though they are not being used. It may become necessary to remove them, temporarily, from the System in order to verify that they are not at fault.

c. The MEDIA may be at fault. Try other media on both the disk and magtape if at all possible to verify that the media is not at fault.

If the VMS Operating System was running immediately prior to the attempted BACKUP or REBUILD, it would be a good idea to verify that it still runs O.K.. This step will tell you that most of the hardware is in good shape.

If a VMS/DIAGNOSTIC Field Service Pack is available attempt to back it up, as a test to help isolate whether there is a media or hardware problem. A Restore could also be done, to a SCRATCH pack, with the tape just generated in order to get a better idea of how much hardware is in reasonably good working order.

If BACKUP or REBUILD is being done in stand-alone mode, it is somewhat harder to trouble-shoot since you have lost two valuable sources of information. There isn't any "ERROR LOG" facility under stand-alone operation and there aren't any facilities that will provide Software Dumps. Therefore, the only sources of information available to you are the Console Terminal output and any Hardware register dumps that may have been taken.

If a NON-D.E.C. Disk or Tape drive is being used for the BACKUP or REBUILD operation, it may be the source of the problem. We, D.E.C., do not support our Device Drivers being used on foreign equipment.

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

BOOTING Problems

Power-up booting outline

I. On a system POWER-UP, "CONSOL.SYS" is booted as follows:

A. BPOK & BDCOK, on LSI subsystem H780 Power Supply, goes high.

B. KD11-F jumpers, W5 & W6, specify what to do on power up.
   1. Trap to 24
   2. Halt - go into ODT and print "@" prompt
   3. Jump to 173000

   The VAX-11/780's KD11-F is setup to "jump to 173000".

C. Execute PDP-11 MACROCODE routine that starts at Q-Bus address 173000.

```
173000 / 000137
173002 / 140200
```

D. The CIB ROM starts at 140000, is 4Kwords long, and goes to 157776. This ROM's main purpose is to load in the bootblock off the RXV11 media, and transfer control to it. The entry point at 140200 causes the following to be done:

   1. Run some Q-Bus Memory tests.
   2. Assign Q-Bus Terminal addresses.
   3. Run LSI CPU tests.
   4. Read Boot block from RXV11.

   The remaining steps happen if the LOCAL or REMOTE CONSOLE FLOPPY is in the RXV11 disk drive:
   5. Read Directory on RXV11 media.
   6. Load CONSOL.SYS into Q-Bus memory.
   7. Start CONSOL.SYS program.

E. Upon initialization, CONSOL.SYS does the following:

   1. Does a "SHOW" command.
   2. Inits VAX CPU.
   3. Load WCS microcode.
   4. Does a "SHOW VERSIONS" command.

F. CONSOL.SYS checks "AUTO RESTART SWITCH" to see what to do next.

   1. if "AUTO-RESTART SWITCH" = "off"

      a. use "DEFBOO.CMD", on RX01 floppy, to boot the system. This command file contains CONSOL.SYS commands that will perform the following functions:

         1.) setup VAX R<0:5> to indicate following:
            a.) Boot NEXUS
            b.) Primary Bootstrap
            c.) Operator intervention (stop in SYSBOOT)
            d.) Media device type where Secondary bootstrap is stored.
            e.) Boot device unit number

2.) start the VAX macrocode program that is resident in the ISP ROM. This programs main job is to find a good 64KB of VAX memory where the primary VAX bootstrap can be loaded. The ISP ROM program will exit, upon successful completion, with the starting address of the good 64KB chunk of memory +200 in the STACK POINTER (R14).

3.) load VMB.EXE (primary bootstrap), from RX01 floppy, into VAX memory starting at the address specified in the SP.

4.) start VMB.EXE (a VAX macro-code program).

b. VMB.EXE loads secondary bootstrap, per flags that are setup in VAX R<0:5>, which loads program;

1.) [SYSMAINT]DIAGBOOT.EXE if R5<4>=1
   a.) loads [SYSMAINT]ESSAA.EXE

2.) [SYSEXE]SYSBOOT.EXE if R5<4>=0
   a.) loads [SYSEXE]SYS.EXE

2. if "AUTO-RESTART SWITCH" = "on"

a. use "RESTAR.CMD", on RX01 floppy, to reboot the system.

1.) setup VAX R<0:5> to indicate what mapping registers to use.

2.) start ISP ROM at WARM RESTART location "20003004".

3.) WARM RESTART code attempts to find RPB (restart parameter block).

4.) if RPB found, restart power interrupted routine via contents of the RPB.

b. If unable to reboot via Warm restart, VAX ISP ROM program (VAX Macrocode) sends code to CONSOL.SYS indicating a WARM RESTART FAILURE.

1.) CONSOL.SYS then attempts a reboot by using the DEFBOO.CMD file.

Booting Problems occur in many different types of ways but the method of trouble-shooting is fairly simple.

Proceed as follows:

1. Determine WHERE in the VMS Boot outline that the System is experiencing problems. How far the Boot Procedure got will tell you how much hardware you have to diagnose.

   If it is failing before the VMB.EXE program is started, the following hardware may be at fault:

   a. Any part of the LSI-11 Subsystem.
   b. The LOCAL CONSOLE Floppy.
   c. VAX Memory.
   d. VAX-11/780 CPU.
   e. Power Supplies and Power Monitoring circuits.

   From this point on, any hardware on the System could cause failures. However, the most likely problem areas will be listed here. Just beware that any hardware could be at fault from this point on.

   If it is failing after the VMB.EXE program is started, but before the VMS identification message is typed on the Console Terminal, the following hardware is most likely to be at fault:

   a. VAX-11/780 CPU.
   b. VAX Memory.
   c. VAX Power Supplies and Power Monitoring circuits.
   d. System Disk and SBI controller.

   If it is failing after the VMS identification message, then the most likely hardware to be at fault is:

   a. VAX-11/780 CPU.
   b. VAX Memory.
   c. Power Supplies and Power Monitoring circuits.
   d. System Disk and SBI controller.
   e. DW780 Unibus Devices

2. Check to see if the problem can fit into one of the other "Types of Problems" listed at the beginning of this Trouble-shooting Outline. If it does fit under another type, use that types' outline to trouble-shoot the problem.

   An example would be, while attempting to boot the Operating System, it crashes with "?INT-STK INVLD". If this was the case, you should go to the "Interrupt Stack Not VAlid" flow which is under the "Operating System Crashes or Bugchecks" section.

3. Hardware Register Dumps can be taken to see if there are any hardware errors set at failure time.

4. The problem could also be a software problem. Try another SYSTEM Pack if available. Here is where a Field Service VMS/DIAGNOSTIC Pack would be very useful.

Overview of LSI-11 Subsystem Bootstrapping

1. With the power-on sequence, the Console ROM bootstrap program is started (this requires the operator action of applying power). The Console ROM is located on the CIB (M8236) board and is initiated by the LSI CPU executing macro instructions starting at ROM location 173000. The LSI CPU board contains jumpers that enable it to jump to 173000 on power up.

2. A series of LSI-11 tests are executed by the CIB ROM macro instructions. These are PDP-11 macro instructions that are executed by the LSI-11 processor.

3. The Console program, CONSOL.SYS, is then loaded from the Floppy disk drive (the LOCAL CONSOLE or REMOTE CONSOLE floppy must be installed in the Floppy Disk Drive) into LSI-11 memory. This is accomplished by execution of macro instrucions in the CIB ROM.

4. The Console program, CONSOL.SYS, is then started. The initiation of the CONSOL.SYS program prints the same information that you would get with a Console "SHOW" command followed by a line indicating that an INIT VAX-11/780 CPU has finished, and that is followed by a statement specifing where the VAX CPU is halted The following is an example of the type of printout that should occur on the LSI-11 Console Terminal:

```
CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000
INIT SEQ DONE
HALTED AT 00000000
```

5. The Console program, CONSOL.SYS, then loads the WCSxxx.PAT file from the Console Floppy into the WCS portion of the VAX-11/780 CPU, (xxx = current version of WCS code on Floppy). The following is an example of the type of printout that should now be printed on the LSI Console Terminal:

```
(RELOADING WCS)
LOAD DONE, 0800 MICROWORDS LOADED
VER: PCS=01 WCS=0E-10 FPLA=0E CON=V07-00-L
```

6. If the AUTO RESTART switch is ON, the CPU bootstrap is now initiated.

7. If the AUTO RESTART switch is OFF, the console is held in the Console I/O mode of operation awaiting operator input. The LSI Console Terminal will print the CONSOL.SYS prompt and remain in input mode. Prompt is as follows:

```
>>>
```

Overview of VAX CPU bootstrapping

1. With the power-on sequence, the VAX CPU goes to the initialization routines of the VAX CPU microcode.

2. The CPU then waits for the start of a console boot sequence. The console boot can be initiated by one of the following ways:

    a. Console BOOT command entered to the CONSOL.SYS program by the operator. The CONSOL.SYS program executes the appropriate command file from the CONSOLE Floppy.

    b. VAX BOOT switch is pressed by the operator. The CONSOL.SYS program executes the DEFBOO.CMD command file from the CONSOLE Floppy.

    c. An Auto-restart sequence is initiated, by the AUTO RESTART switch being ON, and a Warm Restart is attempted. The CONSOL.SYS program executes the RESTAR.CMD command file from the CONSOLE Floppy.

        If a warm restart fails,go to step 3.

        If a warm restart succeeds, go to step 5.

3. When any one of the preceding conditions occur, the console (CONSOL.SYS) loads a bootstrap into the VAX CPU's memory from the CONSOLE FLOPPY. The bootstrap is VMB.EXE.

4. The Console program, CONSOL.SYS, starts the VAX CPU in the VMB.EXE (that was just loaded). VMB.EXE loads and starts the secondary bootstrap (SYSBOOT.EXE or DIAGBOOT.EXE).

5. The Console Program, CONSOL.SYS, enters its PROGRAM I/O mode of operation.

6. Any output to the Console Terminal now comes from the running VAX-11/780 macro program via the CONSOL.SYS program. The CONSOL.SYS program passes data to the terminal from the VAX CPU.

7. Any input is passed from the Console terminal to the running VAX macro program via the CONSOL.SYS program. EXCEPT, if a "CTRL""P" (^P) is typed on the Console terminal, the CONSOL.SYS program will then go back to "CONSOLE I/O" mode and you will then be talking to CONSOL.SYS directly again.

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

FRONT-END SUBSYSTEM Problems

The Front-end Subsystem (LSI-11 and Associated Peripherals) can have many types of problems, also. The subsystem is a very simple and easy to fix system. There are a few things that should be kept in mind while trouble-shooting subsystem problems.

1. Be sure to check that the jumpers of the modules that you are placing into the system matches those on the module that you have taken out.

2. Be sure to mark all original modules so that you will not get then mixed up later on.

3. Remember that the CIB (M8236) module is part of the CONSOLE SUBSYSTEM.

4. Remember that AC/DCLO signals from the VAX-11/780 are turned into FAIL/DEAD on the Q-Bus.

5. Don't forget about checking Voltages and Power Monitoring Signals.

6. Have you run all the LSI-11 Subsystem Diagnostics?

LSI Subsystem TRAPS

Whenever the LSI processor hardware detects errors, it will execute a trap sequence. This trap sequence does the following steps:

1. Pushes the "PSW" onto the STACK.

2. Pushes the "PC", at the time of the error, onto the STACK.

3. Places the contents of the "TRAP_VECTOR" into the "PC".

4. Places the contents of the "TRAP_VECTOR+2" into the "PSW".

5. Resumes executing macro instructions from the "NEW" PC.

Trap Vector Assignments

000000 *Reserved. (an Error Trap)*
Also indicates a Trap within a Trap.
Got here due to an error occuring while servicing another error, or by some instruction modifying the PC to 000000.

If the TRAP-CATCHER is installed, the LSI will halt with the PC pointing to 000004 if this error occurs.

000004 *CPU Errors. (an Error Trap)*
Non-existent Memory Errors
Sack Timeouts
Odd Addressing Errors

If the TRAP-CATCHER is installed, the LSI will halt with the PC pointing to 000010 if this error occurs.

000010 *Illegal and Reserved Instruction. (an Error Trap)*
An attempt was made to execute an illegal or reserved instruction opcode.

If the TRAP-CATCHER is installed, the LSI will halt with the PC pointing to 000014 if this error occurs.

000014 *BPT (Breakpoint Trap) executed.*
Got here due to the BPT instruction executed.

000020 *IOT (Input/Output Trap) executed.*
Got here due to the IOT instruction executed.

000024 *Power-Fail detected. (an Error Trap)*
Got here due to detection of a Power Failure.

If the TRAP-CATCHER is installed, the LSI will halt with the PC pointing to 000030 if this error occurs.

000030 *EMT (Emulator Trap) executed.*
Got here due to the EMT instruction being executed.

000034 *TRAP instruction executed.*
Got here due to the TRAP instruction being executed.

Traps are usually easy to trouble-shoot as long as the proper information is gathered at the time of the failure. A software dump of certain locations is very helpful in isolating the source of the error in all of Error traps listed above except for a Power Fail trap. In order to gather this software information, you must first install an LSI/PDP-11 TRAP CATCHER in memory and then wait for the next error to occur.

LSI/PDP-11 TRAP CATCHER

The later versions of CONSOL.SYS have software routines for the different LSI traps that can occur, i.e. "Trap-4". These routines, unfortunately, do not dump any of the information that you need to trouble-shoot them. In order to get a Software Dump of these traps, you must deposit a TRAP CATCHER into LSI memory prior to getting the error. To do this, use LSI ODT commands to deposit the TRAP CATCHER.

```
$ ^P                    <--- Type "CTRL/P" to VMS prompt.
>>>                     <--- Place LSI "HALT/ENABLE" switch to "HALT".
yyyyyy                  <--- LSI PC at time halted. Remember for later.
@0/ xxxxxx 2<line feed>
000002/ xxxxxx 0<line feed>
000004/ xxxxxx 6<line feed>
000006/ xxxxxx 0<line feed>
000010/ xxxxxx 12<line feed>
000012/ xxxxxx 0<return> <-- Place "HALT/ENABLE" to "ENABLE"
@yyyyyyP                <--- restarts CONSOL.SYS where left off.
>>> SET TERMINAL PROGRAM<return><return>
$                       <--- Now back to VMS.  Wait for error.
```

POWER FAIL Traps

The trap is caused by one of the following:

1. A true drop in power below the specifications of the power supplies that have their Power Monitoring signals connected to the LSI's "BPOK" and "BDCOK" circuits.

2. A false detection of a Drop in power by one of the "BPOK" and/or "BDCOK" circuits, or interconnected Power monitoring signals.

3. Noisy Power supply and/or Power Monitoring signals.
   a. AC/DCLO on H7420 type supplies should be at least a +3.5 volt level to insure proper noise immunity.
   b. H7100 AC/DCLO signals should be at least a -9.5 volt level to insure proper noise immunity.

4. LSI Subsystem failure causing the LSI Processor to enter the trap vector.

Possible Problem areas are:

1. The H780 LSI Power Supply.
2. The H780 LSI Power Supply Power Monitoring Circuits.
3. The VAX-11/780 CPU/Nexus H7100 Power Supplies.
4. The VAX-11/780 CPU/Nexus H7100 Power Monitoring Circuits.

The Power Monitoring signals in the following supplies are or-ed together and then feed the LSI CPU's Power Fail Circuits:

1. H780 LSI Power Supply
2. VAX-11/780 H7100 Power Supply #1, #2, and #3

Gathering LSI Software DUMP (should be halted in a TRAP CATCHER)

If the LSI-11 is trapping, the following ODT commands can be used to gather information to determine what instruction or address is failing, (assuming that you have installed the TRAP CATCHER and the LSI halts). Type those things within double quotation marks. Things within a single quotation mark signifies what keyboard key to type.

Take the following dump first thing after LSI-11 goes to ODT mode (@).

Upper Case characters must be used when talking to ODT.

```
1.  Type "M"                        ; Get LSI Maintenance Register.
2.  Type 'RETURN'
3.  Type "RS/"                      ; Get the Processor Status Word.
4.  Type 'RETURN'
5.  Type "R0/"                      ; Get Contents of R0.
6.  Type 'LINEFEED' 6 times         ; Get Contents of R1 thru R6.
7.  Type "@"                        ; Get Failure PC off Stack.
8.  Type "@"                        ; Get contents of Failure location.
9.  Type "^" 15 times               ; Get Instruction Stream.
10. Type 'RETURN'
11. Type "R6/"                      ; Prepare to get information in
12. Type "@"                        ; case the mode used in the failing
13. Type "@"                        ; instruction was either PC mode 6
14. Type "^"                        ; or PC mode 7 addressing.
15. Type " "                        ; Get PC mode 6 or 7 information.
16. Type "@"                        ; Get PC mode 7 operand.
17. Type 'RETURN'
```

A dump of the LSI is now complete. Proceed to next step if you want to RESTART the LSI subsystem or REBOOT the Operating System.

18. If you want to attempt to reboot the Operating System do one of the following:

    a. If you want to do a complete Operating System reboot, type the following if at the ODT prompt (@):

       "173000G"

    b. If you only want to reboot the LSI subsystem without rebooting the Operating System, type the following:

       To "@" (ODT) prompt type - 141330P
       To ">>>" (CONSOL.SYS) prompt type - SET TERMINAL PROGRAM

The Dump just taken can be analyzed in order to determine what address or instruction caused the trap.

ANALYZING LSI Software Dumps taken after Halting in a Trap Catcher.

LSI-11 Software Dumps for crashes that have been halted in a Trap Catcher are fairly easy to analyze if you have at least a general understanding of the PDP-11 Instruction set, the PDP-11 Addressing modes, and how a PDP-11 trap occurs. The following steps assume that you have at least this level of knowledge.

1. The LSI CPU will do the following steps whenever it detects a TRAP condition:

   a. Pushes the PSW onto the stack. The stack is AUTO-DECREMENTED prior to pushing this data onto it.

   b. Pushes the PC onto the stack. Again, the stack is AUTO-DECREMENTED prior to pushing this data onto it.

   c. A new PC is fetched from the TRAP VECTOR location in physical memory. The actual location will be one of the following:

      LSI Memory Location 000000 if DOUBLE BUS ERROR Trap.
      LSI Memory Location 000004 if BUS ERROR Trap.
      LSI Memory Location 000010 if ILLEGAL/RESERVED INSTRUCTION Trap.
      LSI Memory Location 000024 if POWER FAIL/RECOVER Trap.

   d. A new PSW is fetched from the TRAP VECTOR+2 location in physical memory. The actual location will be one of the following:

      LSI Memory Location 000002 if DOUBLE BUS ERROR Trap.
      LSI Memory Location 000006 if BUS ERROR Trap.
      LSI Memory Location 000012 if ILLEGAL/RESERVED INSTRUCTION Trap.
      LSI Memory Location 000026 if POWER FAIL/RECOVER Trap.

   e. The LSI will then continue MACRO-CODE execution starting at the new PC. If a TRAP-CATCHER has been deposited, like the one specified in this section, the LSI CPU will execute a HALT (code = 000000) instruction.

2. At this point in time, the LSI DUMP procedure should be executed. This will gather the needed information to allow you to analyze what was happening, or who was being accessed, at the time of the TRAP. In most cases, this analysis will point directly to the failing unit.

3. To analyze the dump, proceed as follows:

   a. Find out the contents of LSI "General Register #6" (R6, %6, or SP). This data is the address of the current bottom of the STACK. The STACK builds from high address towards lower address, therefore the contents of R6 will be pointing to the last entry pushed onto the STACK. This entry will be the saved PC of where the LSI instruction set processor was running at the time of the trap.

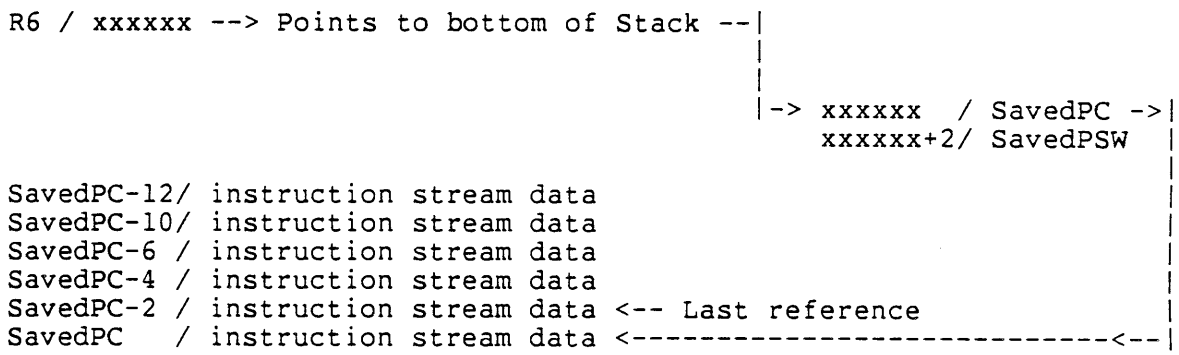
   b. Using the "contents of R6" as an "address", examine this memory location. The data from this last examine is the PC at the time of the TRAP.

c. Subtract 2 from this PC to find the address of the last memory reference prior to the TRAP, or at the time of the TRAP. The "PC-2" contains INSTRUCTION, or OPERAND-SPECIFIER-DATA, that was being used at the time of the Trap.

d. Now is when you need the general knowledge about the PDP-11 instruction set, how it works, and how the addressing modes work. With this knowledge you should be able to look back through the location prior to the "PC-2" location and determine what was happening prior to the trap. You must use your knowledge of the PDP-11 instruction set to find out where the instructions actually start.

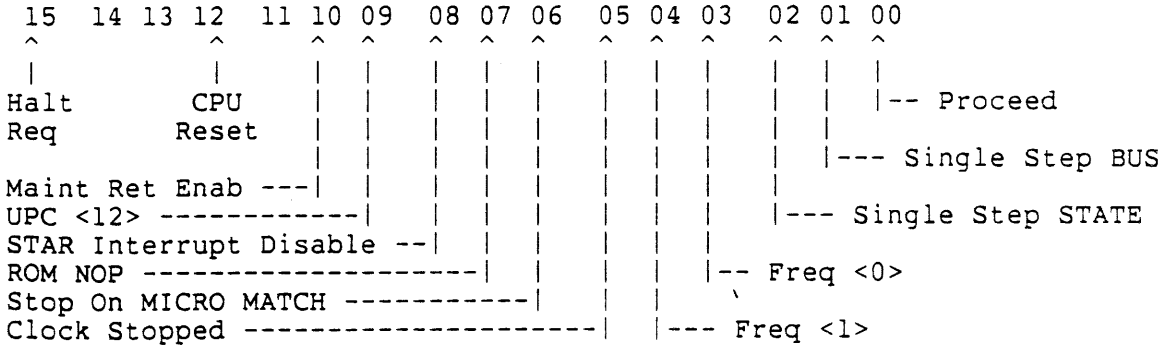
4. If you cannot make sense of the dump, either Remote Support or your local Support groups should be able to analyze the dump.

LSI-Traps Software Dump Analysis Flow

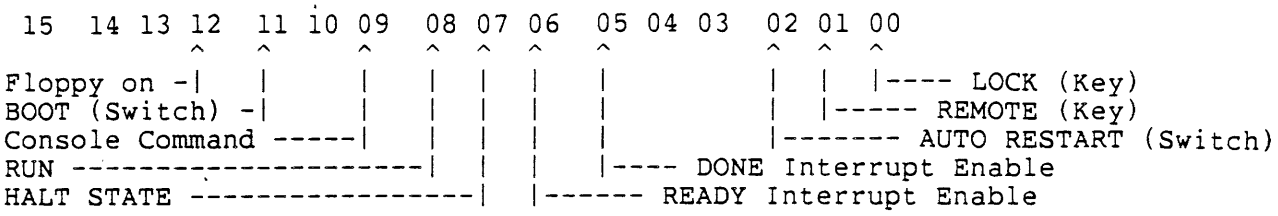


Device	Q-Bus Address	Register Name	Vector
RXV11	177170	RXCS	264
	177172	RXDB	
DLV11	177560	RCSR	60 - Reciever
	177562	RBUF	
	177564	XCSR	64 - Transmitter
	177566	XBUF	
DLV11-E	175610	RCSR	310 - Receiver
	175612	RBUF	
	175614	XCSR	314 - Transmitter
	175616	XBUF	
CIB	173000	ROM 0	300 - RX Done
	173002	ROM 1	304 - TX Ready
	173004	spare	
	173006	ID Data LO	
	173010	ID Data HI	
	173012	spare	
	173014	RX DONE	
	173016	TX READY	
	173020	TO ID Lo	
	173022	TO ID Hi	
	173024	FM ID Lo	
	173026	FM ID Hi	
	173030	ID C/S	
	173032	MCR	
	173034	MCS	
	173036	V-BUS	
	Note:	The above addresses are dependent on the CIB W1 jumper being INSTALLED. If W1 is OUT the addresses would be 1630xx instead.	
	140000 to 157777	CIB Bootstrap ROM	

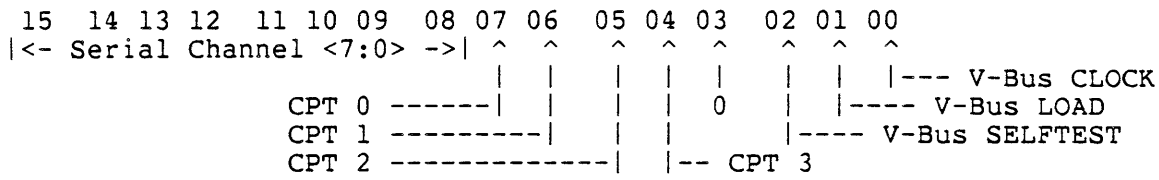
MCR - *Q-Bus address = 173032*



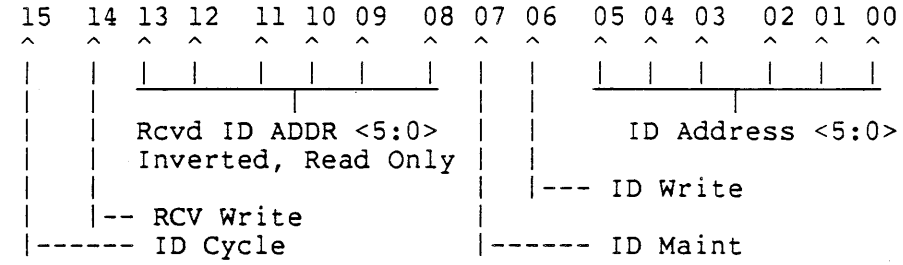
MCS - *Q-Bus address 173034*



V-BUS *Q-Bus address 173036*



ID C/S *Q-Bus address 173030*



RX DONE - *Q-Bus address 173014*
Contains one bit. Bit <07> = RX DONE

TX READY - *Q-Bus address 173016*
Contains one bit. Bit <07> = TX READY

ID Data Lo, TO ID Lo, FM ID Lo all contain the low order bits, bits <15:00> of a 32 bit data word.

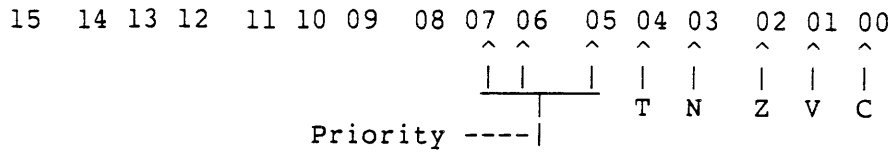
ID Data Hi, TO ID Hi, FM ID Hi all contain the high order bits, bits <31:16> of a 32 bit data word.

PDP-11 Instruction Set

```
000000  HALT
000001  WAIT
000002  RTI
000003  BPT
000004  IOT
000005  RESET
000006  RTT
000007
  to    reserved
000077
0001DD  JMP
00020R  RTS
000210
  to    reserved
000227
000240  NOP
000241  CLC
000242  CLV
000244  CLZ
000250  CLN
000257  CLNZVC
000260  NOP
000261  SEC
000262  SEV
000264  SEZ
000270  SEN
000277  SECVZN
0004xXX BR
0010xXX BNE
0014xXX BEQ
0020xXX BGE
0024xXX BLT
0030xXX BGT
0034xXX BLE
004RDD  JSR
0050DD  CLR
0051DD  COM
0052DD  INC
0053DD  DEC
0054DD  NEG
0055DD  ADC
0056DD  SBC
0057DD  TST
0060DD  ROR
0061DD  ROL
0062DD  ASR
0063DD  ASL
0064NN  MARK
0065SS  MFPI
0066DD  MTPI
0067DD  SXT
007000
  to    reserved
007777
01SSDD  MOV
02SSDD  CMP
03SSDD  BIT
04SSDD  BIC
05SSDD  BIS
06SSDD  ADD
070RSS  MUL
071RSS  DIV
072RSS  ASH
073RSS  ASHC
074RDD  XOR
07500R  FADD
07501R  FSUB
07502R  FMUL
07503R  FDIV
075040
  to    reserved
076777
077RNN  SOB
1000xXX BPL
1004xXX BMI
1010xXX BHI
1014xXX BLOS
1020xXX BVC
1024xXX BVS
1030xXX BCC,BHIS
1034xXX BCS,BLO
104000
  to    EMT
104377
104400
  to    TRAP
104777
1050DD  CLRB
1051DD  COMB
1052DD  INCB
1053DD  DEGB
1054DD  NEGB
1055DD  ADCB
1056DD  SBCB
1057DD  TSTB
1060DD  RORB
1061DD  ROLB
106400
  to    reserved
106477
1065SS  MFPD
1066DD  MTPD
106700
  to    reserved
107777
11SSDD  MOVB
12SSDD  CMPB
13SSDD  BITB
14SSDD  BICB
15SSDD  BISB
16SSDD  SUB
170000
  to    Floating
177777  Point inst.
```

xXX = 8-bit offset that when sign extended and added to the PC results in the new PC.

PDP-11 Processor Status Word



R0-R6 mode addressing

Mode	Name	Symbolic		Description
0	register	R	-------	(R) is operand
1	register deferred	(R)	-------	(R) is address of operand
2	auto-increment	(R)+	-------	(R) is address of operand, a 1 or 2 is added to (R) after use.
3	auto-increment deferred	@(R)+	-------	(R) is address of the address of operand. A 1 or 2 is added to (R) after use.
4	auto-decrement	-(R)	-------	(R) is decremented by 1 or 2 and the resulting (R) is the address of the operand.
5	auto-decrement deferred	@-(R)	-------	(R) is decremented by 1 or 2 and the resulting (R) is the address of the address of the operand.
6	index	x(R)	-------	(R) is added to "x" and the result is the address of the operand.
7	index deferred	@x(R)	-------	(R) is added to "x" and the result is the address of the address of the operand.

PC Mode addressing

Mode	Name	Symbolic		Description
2	immediate	#n	-------	operand, "n", follows the instruction or source operand.
3	absolute	@#A	-------	address of the operand, "A", follows the instruction or source operand.
6	relative	A	-------	Instruction Address + 4 + X is the address of the operand. "A" is the address of the operand.
7	relative deferred	@A	-------	Instr. address + 4 + X is the address of the address of the operand. The contents of "A" is the address of the operand.

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

UNEXPLAINED REBOOTS & POWER RESTARTS

Symptoms of spurious Reboots and Power Restarts

This type of problem can be identified by finding the following type of output on the console terminal.

1. The system is running along printing out normal operating system type information.

2. Then, without any prior error printouts, a message appears that is like or resembles (depending on the actual version of the Console Floppy) the following:

```
$      <-- This line may contain any type of VMS output

CPU HALTED,SOMM CLEAR,STEP=NONE,CLOCK=NORM
RAD=HEX,ADD=PHYS,DAT=LONG,FILL=00,REL=00000000
INIT SEQ DONE
HALTED AT 00000000

(RELOADING WCS)
LOAD DONE, 0800 MICROWORDS LOADED
VER: PCS=01 WCS=0E-10 FPLA=0E CON=V07-00-L
(AUTO-RESTART)       <-- From here on depends on the position of
CPU HALTED                   the "Auto-Restart Switch".
INIT SEQ DONE

$
```

3. The Operating System may or may not reboot depending on the position of the "Auto-Restart Switch" on the VAX control panel.

This type of problem is usually power related.

Check the following things:

1. Are the VAX-11/780 CPU, MEMORY, and SBI NEXUS power supply voltages O.K.? Check them with a Scope (for Noise) and with a DVM (for correct level).

2. Are the LSI-11 Subsystem Voltages O.K.? Check the H780 power supply with a Scope and a DVM for correct levels and the absence of noise.

3. Check all Power Monitoring signals (AC/DCLO) on both the VAX supplies and the LSI-11 supply for both the correct level and absence of noise. Use a Scope and a DVM. The actual H7100 AC/DCLO signals that can cause this problem are H7100 Supplies #1,#2,#3, and #4.

4. Verify that the 869 Power Controller is not dropping power to the system.

5. Check the input AC power to the 869 Power Controller. Is it low? It may be necessary install a DRANETZ to monitor input power to the system.

The above mentioned Supplies and Power Controller may need to be replaced one at a time in order to isolate the problem. Always put back the original whenever it is determined that it was not at fault.

There are four modules that may be causing the Power Restarts. If everything else checks O.K., try replacing them. They are:

1. M8232 - Clock board. Monitors "Supplies #1,#2,#3, & #4" and generates its own ACLO/DCLO signals.

2. M8236 - CIB board. Sends the VAX system ACLO/DCLO signal onto the Q-Bus BPOK/BDCOK lines so that the LSI knows that the VAX detected a power problem.

3. M8224 - IRC board. Monitors the T.O.D. Battery DCLO signal and passes it on to the Clock board.

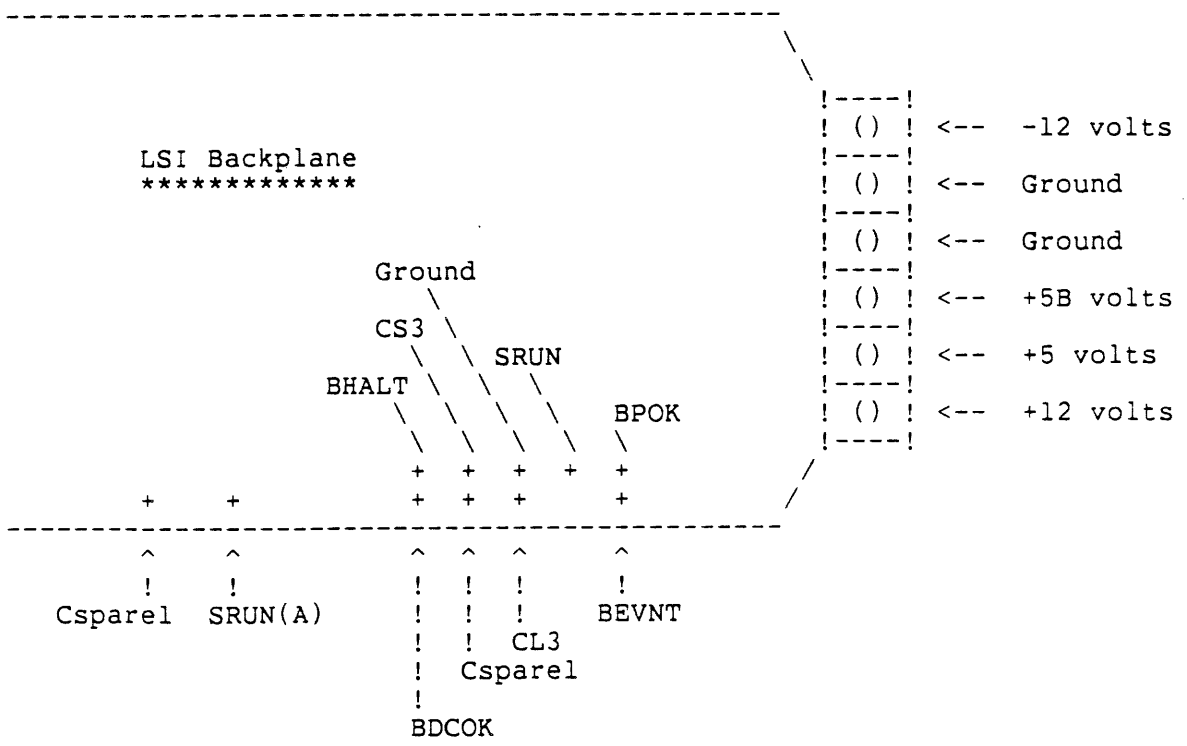
4. KD11-F LSI CPU Board. This board receives the power fail indication and causes the reboot.

Isolating the problem via disconnecting AC/DCLO signals

Sometimes it is necessary to be able to eliminate some of the hardware by disconnecting sources of the Power Monitoring signals. Here is how this can be done:

1. The H780 Power Supply can be isolated by disconnecting the BPOK and BDCOK signals that it generates so that they never reach the LSI CPU. In order to do this, you can bend the two pins that receive BDCOK and BPOK so that they are not connected when you reinstall the H780 to LSI Backplane Gray ribbon cable. Use the following diagram to locate the BPOK and BDCOK signal pins on the LSI backplane.

   BE VERY CAREFUL not to bend these two backplane pins any more than is absolutely necessary. These pins are easily broken if you bend them too far. "ONLY" bend them far enough to allow the cable to be put back on "ONLY" far enough to allow the other signal pins to be connected.



2. The H7100 Power Supplies can be eliminated by diconnecting the "BPOK & BDCOK" signals that are generated on the "CIB" board and are transimitted on the Q-Bus lines. To do this, you can bend "Pins K & M" on "P7" sightly so that you can reinsert the "J7" connector with the two pins disconnected. This will prevent any spurious ACLO/DCLO signals, from the H7100A Power Supplies (Supplies #1,#2,#3, or #4), from the CIB (M8236) board, or from the CLOCK (M8232) board, causing the system to be rebooted or restarted. The actual H7100's that are connected to the "Supply #1 thru #4" connectors are Power Supplies #1,#2, and #3. All the other H7100 Power Supplies feed logic in there associated NEXUS' that uses the SBI FAIL/DEAD lines to signal the VAX CPU of power problems.

KA780 Backplane AC/DCLO Supply connector assignments:

Supply #1 = J17	Supply #2 = J16
Supply #3 = J15	Supply #4 = J14

T.O.D. Clock's Battery DCLO:

J20 - A08F1

No FP780 installed:

J14 - J16 - SFT - PS #2
J15 - PS #3
J16 - (J14 - DW780 #1 - J17) - SFT - PS #2
J17 - DW780 #1 - J14 - J16 - SFT - PS #2
J14 and J16 are connected via the KA780 backplane.

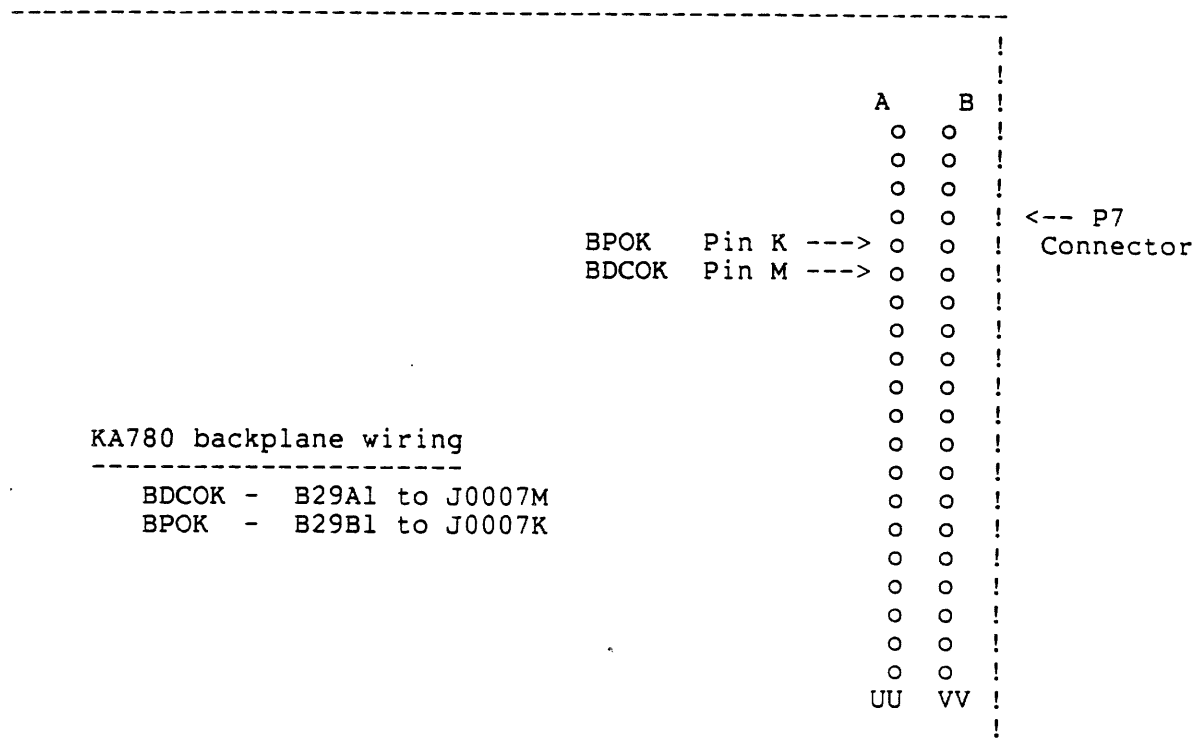
With FP780 installed:

J14 - J16 - SFT - PS #2
J15 - PS #3
J16 - (J14 - DW780 #1) - SFT - PS #2
J17 - PS #1
J14 and J16 are connected via the KA780 backplane.

BE VERY CAREFUL not to bend the two pins any more than absolutely necessary or they may break when you attempt to restraighten them after the problem has been isolated. It is better if you don't actually bend the pins at all but simply hold them out of the way while reinstalling the "P7" cable no more that a third of the way onto the other "J7" pins.

"Q-Bus BPOK/BDCOK" are generated on the CIB board as a result of any H7100 Supply generating an ACLO/DCLO signal or by the T.O.D. clock Battery backup generating a "BAT DCLO".

Top right of KA780 Backplane (view from pin-side)



KA780 backplane wiring

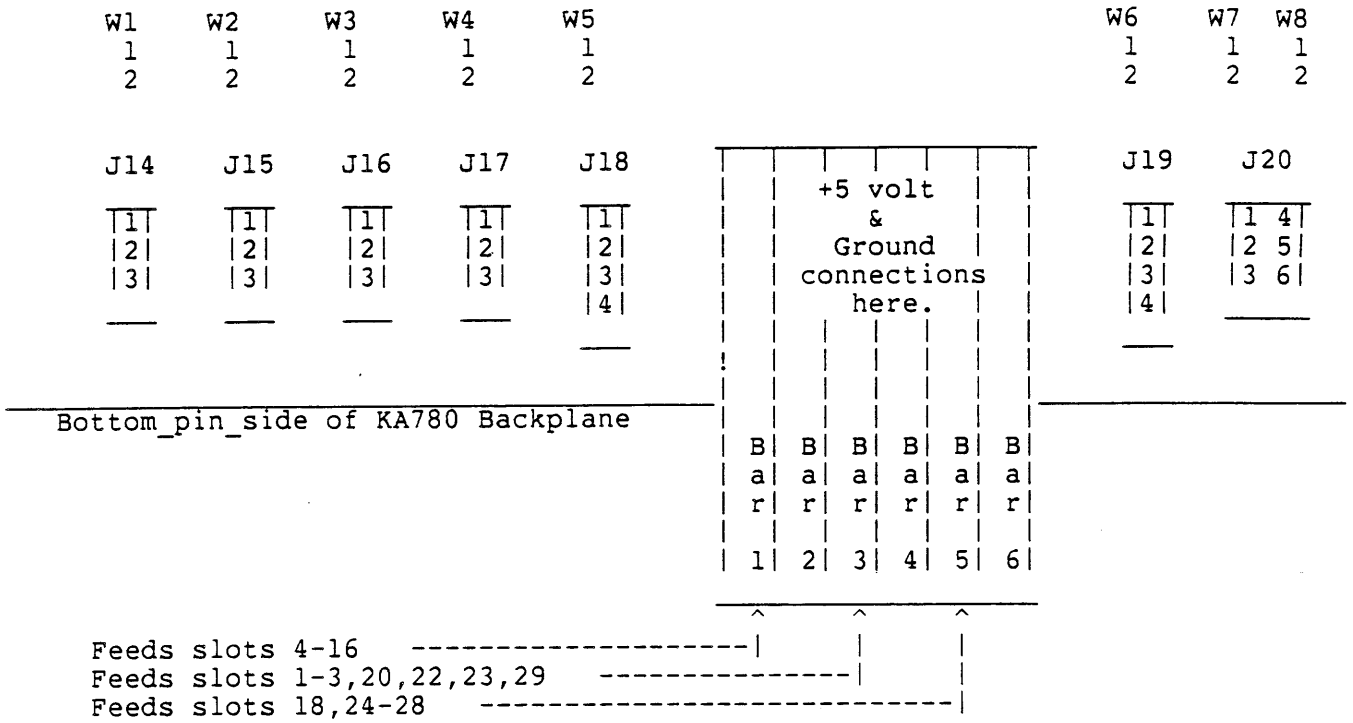
BDCOK	-	B29A1 to J0007M
BPOK	-	B29B1 to J0007K

After disconnecting "BPOK" and "BDCOK" from "J0007", as described above, the type of failure symptom that occurs next will indicate in which half the problem lies.

1. If the LSI still reloads CONSOL.SYS and the VAX WCS, then the problem is in the LSI Subsystem.

2. If an error occurs that indicates that the VAX is hung or halted, then the problem is in the VAX CPU or VAX Power Supplies.

The backplane connectors, J14 thru J20, are located on the pin side of the VAX-11/780 CPU backplane, at the bottom, and are as follows:



J14 connections
****

W1-1 1 - Supply 4 ACLO
W1-2 2 - Supply 4 DCLO
3 - Ground

J15 connections
****

W2-1 1 - Supply 3 ACLO
W2-2 2 - Supply 3 DCLO
3 - Ground

J16 connections
****

W3-1 1 - Supply 2 ACLO
W3-2 2 - Supply 2 DCLO
3 - Ground

J17 connections
****

W4-1 1 - Supply 1 ACLO
W4-2 2 - Supply 1 DCLO
3 - Ground

J18 connections
****

W5-1 1 - -5v (to BL2 pins)
W5-2 2 - -5v (to EK1 pins)
3 - Ground
4 - Ground

J19 connections
****

W6-1 1 - +5v to Front Panel
W6-2 2 - CIBP FLOPPY ON H
3 - Ground
4 - Ground

J20 connections
****

W7-1 1 - Time of Day Clock's "+5v"
2 - unused
3 - Ground
W8-1 4 - Time of Day Clock's "Battery DCLO"
5 - unused
6 - Ground

Bar 1 - +5v from Power Supply #3
Bar 2 - Return from Power Supply #3

Bar 3 - +5v from Power Supply #2 (also supplies the first DW780)
Bar 4 - Return from Power Supply #2

If FP780 is not installed:

Bar 5 - +5v from Power Supply #2
Bar 6 - Return from Power Supply #2

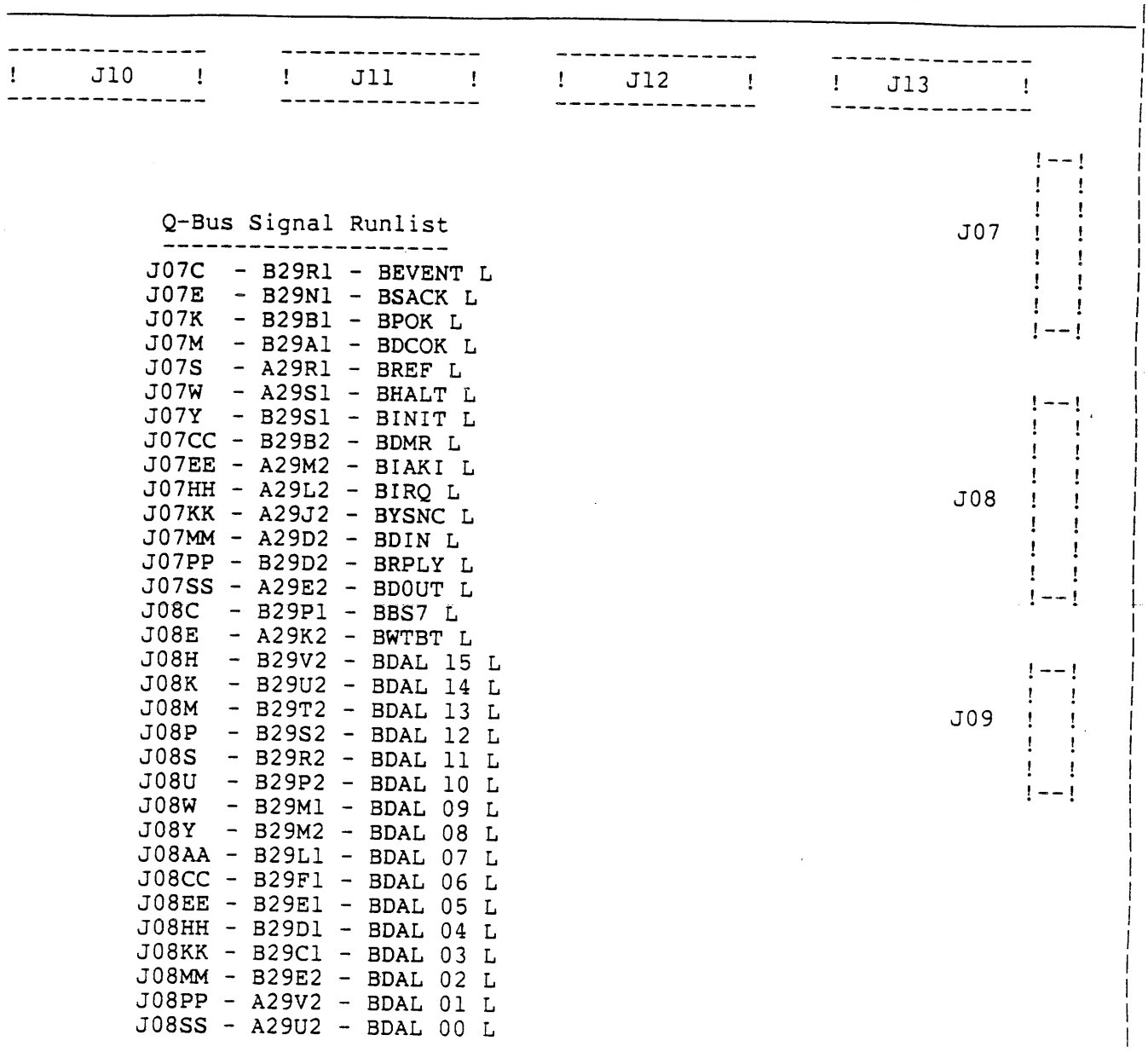
If FP780 is installed:

Bar 5 - +5v from Power Supply #1
Bar 6 - Return from Power Supply #1

Voltage Pins on the KA780 Backplane

+5 volts is present on all slots and rows at pins A2 and V1.
-5 volts is present on all slots at pins BL2 and EK1.

Top_right_hand portion of KA780 backplane (Pin_side view)



Q-Bus Signal Runlist

J07C - B29R1 - BEVENT L
J07E - B29N1 - BSACK L
J07K - B29B1 - BPOK L
J07M - B29A1 - BDCOK L
J07S - A29R1 - BREF L
J07W - A29S1 - BHALT L
J07Y - B29S1 - BINIT L
J07CC - B29B2 - BDMR L
J07EE - A29M2 - BIAKI L
J07HH - A29L2 - BIRQ L
J07KK - A29J2 - BYSNC L
J07MM - A29D2 - BDIN L
J07PP - B29D2 - BRPLY L
J07SS - A29E2 - BDOUT L
J08C - B29P1 - BBS7 L
J08E - A29K2 - BWTBT L
J08H - B29V2 - BDAL 15 L
J08K - B29U2 - BDAL 14 L
J08M - B29T2 - BDAL 13 L
J08P - B29S2 - BDAL 12 L
J08S - B29R2 - BDAL 11 L
J08U - B29P2 - BDAL 10 L
J08W - B29M1 - BDAL 09 L
J08Y - B29M2 - BDAL 08 L
J08AA - B29L1 - BDAL 07 L
J08CC - B29F1 - BDAL 06 L
J08EE - B29E1 - BDAL 05 L
J08HH - B29D1 - BDAL 04 L
J08KK - B29C1 - BDAL 03 L
J08MM - B29E2 - BDAL 02 L
J08PP - A29V2 - BDAL 01 L
J08SS - A29U2 - BDAL 00 L

VAX Control Panel Runlist

J09B - A29A1 - SCPA BOOT SW H
J09D - A29B2 - SCPA AUTO RESTART H
J09F - A29F1 - SCPA LOCK H
J09K - A29D1 - CIBN RUN H
J09L - A29E1 - CIBN ATTN H
J09J - A29K1 - SCPA REMOTE H

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

Problems on CERTAIN DEVICE(s)

This area is extremely variable in the types of problem symptoms that can occur. Therefore, this discussion will only point out a few of the questions that you should ask yourself while trouble-shooting peripheral device problems.

On such peripheral devices as Magtapes and Disks try to eliminate the media as being a possible problem as soon as possible. Media problems are most often seen on Magtapes versus Disks. It is best to use D.E.C. Certified Magtapes to isolate Magtape Problems. Not only is the media suspect on Read/Write problems but also should be suspected on AUTO-LOAD problems.

The Error Log (System Event File) report is a valuable source of information for problem diagnosis.

If the problem is of the type that points to one particular device, the following questions should be answered if appropriate:

1. For a failing device that is on a common controller bus with other devices, answer the following questions for yourself.

   a. Do all the other devices run O.K.?

      If they don't, then goto the next step that covers multiple failing devices on the same common controller interface bus.

      If this is the only device that fails on its associated interface bus, then the problem could either be in that device or could still be a controller or bus problem. In order to make sure it is not a controller or bus problem, check to see if there is anything about the failing device, as related to how it interfaces to the controller, that is different than the running devices.

   b. If all other devices on the controller run O.K., and you have exhausted all other ideas, the problem still could be with one of the other devices on the same bus somehow interferring with the failing device. Make sure this is not the case by removing all other devices from the bus.

2. For failing devices that are on a common controller bus with other devices, answer the following question for yourself.

   a. Does any other Device on the same common controller bus run O.K.?

      If there isn't, then the problem may be one of the following:

      1. A Bus problem.
      2. A Controller problem.
      3. Another Device may be causing failures on other devices on the bus.
      4. A Bus Loading problem.
      5. A Bus Termination Problem.

      If there is, then you know that the Bus is in "Fairly" good shape. Do not, however, totally eliminate the bus as being a source of the problem at this time.

      If there is, then you also know that the Controller is in "Fairly" good shape also. Again, do not totally eliminate the Controller as being a source of the problem yet.

   b. If any other Device on the same common controller bus runs O.K., then answer this question. Are all the failing devices of the same "DEVICE TYPE" ?

      If they aren't, what are the differences as related to how the Controller treats them? For example, are the failing devices Interrupt Driven and the non-failing devices of Direct Memory Access type, or vice-versa? Look for differences that may give you a clue as to what may be causing the problem.

3. For a Failing Device or Devices that are not on a common bus but are on a Controller that is of the multi-port varity, then answer the following question.

    a. Do all the devices fail on the controller?

        If all do then the problem is probably in the controller or a Software problem.

        If some run O.K., then try a running device on the port that fails. If that port runs O.K. now, do not immediately blame the other device, but first evaluate the answer to the following question:

            Are both Devices of the same "DEVICE TYPE" and if they aren't, are there any differences in the way that the controller uses the failing and the running devices?

Never totally eliminate the possibility that the problem may be caused by Software. However before you start looking into software problems, it is most often best to eliminate the "hardware" possibilities first.

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

Won't POWER-UP

This type of problem does not warrant much of a discussion, but a few things to look for will be mentioned.

Check the following:

1. Is the Input Power O.K.?
    a. All voltages present?
    b. All voltages within specification?

2. Are any Circuit Breakers tripped?
    a. If the problem is in the LSI, don't forget to check the position of the LSI Power Supply's ON/OFF switch. This switch is in the back panel of the H780 Power Supply.
    b. Are all the H7100 Breakers set?
    c. Are all the 869 Power Controller breakers set?
    d. Are the breakers for all BAll Boxes set?
    e. Are all breakers set in any expander cabinets.

3. Are any Fuses blown?
    a. Check with a meter and tap to make sure that the connection is solid within the fuse.

4. Are all Interlocks O.K.?
    a. These can be checked by scoping the circuit or by simply defeating the interlocks.

5. Are the AC Power Controllers' outputs O.K.?
    a. All voltages present?
    b. All voltages within specification?
    c. Does it work O.K. in the "LOCAL" position? (D.E.C. Remote Power Bus may be at fault)

6. Are there any THERMAL Switches that may be tripped?
    a. These can be checked by scoping the circuit or by simply defeating the switches.

7. Are there any AIR FLOW Sense switches that may be failing?
    a. These can be checked by scoping the circuit or by simply defeating the switches.

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

SOMETHING'S BURNING

This one is completely up to you. There isn't much that can be said about how to isolate these problems except to use your body's senses and possibly diagnostics.

Here are a couple of things to keep in mind:

1. The part that burnt may have been caused to burn due to failure of another part or possibly due to a short. Look for shorts between:

    a. a VOLTAGE and GROUND.
    b. two or more VOLTAGE runs.
    c. a SIGNAL and GROUND.
    d. a SIGNAL and a VOLTAGE.
    e. two or more SIGNAL runs.

2. Heat and excessive currents weaken components. Their failure may not occur immediately but may show up several weeks or months later. Be sure to mention the fact that this particular part burnt in the System log book. If later on this device starts failing intermittently, you may want to change some parts which were in the same circuit as the burnt part.

3. If a Burning Smell occurs and disappears with you unable to locate the problem, run diagnostics on the device to see if you can locate the problem that way.

When looking for shorts, remember that signal runs should never be at ground (0.00 volts). There is a voltage drop, no matter how minute, across all solid state junctions.

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

Problems BUILDING VMS

If the problem cannot be classified under any of the other problem types, then there isn't very much that can be at fault. The problem should be one of the following types.

a. If the problem is in the booting of the VMS Backup utility, do the following:

1. Check out the hardware via diagnostics.
2. Try another set of floppies for Stand-alone Backup.
3. Check Power.

b. If the problem is one such that the VMS pack cannot be built from the Distrubition tape, then the problem is probably one of the following:

1. The Distrubition Tape
2. The Magtape Drive
3. The Disk Media
4. The Disk Drive
5. Memory
6. Processor
7. Foreign Disk drives or Magtape drives may be at fault.

The hardware can be checked out by diagnostics.

c. If the problem exhibits itself in the booting of VMS from a newly built Disk Pack, check the following:

1. Are the Startup files correct?
2. Are the Unibus devices configured correctly?
3. Foreign devices will probably have to be Connected via the SYSGEN utility.
4. Foreign equipment will probably need special Device drivers installed.

! ! H E L P ! !

! ! H E L P ! !

! ! H E L P ! !

! ! H E L P ! !

N O N - D U P L I C A T A B L E ,

I N T E R M I T T E N T

&

W H A T t o d o N O W

P r o b l e m s

! ! H E L P ! !

! ! H E L P ! !

! ! H E L P ! !

! ! H E L P ! !

Ocassionally the Customer will come up with a problem that we cannot duplicate, is of a very intermittent nature, and you have simply just done everything you could think of to try. Here is a list of things that can be done to verify the functionality of the D.E.C. equipment for non-duplicatable problems and that may pull out the cause of intermittent problems or at least show up a means of more rapidly duplicating the problem. This list can also be used as a What's Left checklist.

1. Check voltages on the system (Complete system) with a DVM. Make appropriate adjustments and/or supply replacement.

   a. Verify that the Power connections are good. Tap wires to verify good contact at connectors.
   b. Vibrate Power supply if possible.

2. Check AC/DCLO's on the system (complete system) with a DVM. Repair any that are out of spec..

   a. Verify that the AC/DCLO connections are good. Tap wires to verify good contact at connectors.
   b. Vibrate Power monitoring section of Supply if possible.
   c. H7420 type AC/DCLO signals should be at least at a +3.5 volt level.
   d. H7100 type AC/DCLO signals should be at least at a -9.5 volt level.

3. Check voltages on the system (Complete system) with a scope for excessive noise. Locate the source of the noise, (Power Supply, Wiring, etc.) and repair.

4. Check AC/DCLO's on the system (complete system) with a scope. Repair any that have excessive noise on them.

5. Run all diagnostics.

   a. Run LSI Front-end Subsystem diagnostics.
   b. Run VAX-11/780 Micro-diagnostics #1 and #2.
   c. Run VAX-11/780 Functional diagnostics. (Don't forget to run EVKAA)
   d. Run appropriate VAX-11/780 Nexus diagnostics.
   e. Run appropriate Unibus Peripheral diagnosics.
   f. Run appropriate Massbus Peripheral diagnostics.
   g. Run all other appropriate peripheral diagnostics.
   i. Run UETP if VMS Operating System Pack is available. Don't use the Customer's only Pack.

6. While running the appropriate diagnostics, vibrate each device's modules, backplane, and Power Supplies.

7. Margin all devices that have any of the following types of margining facilities:

    a. Voltage margins. This is possible with any device that has an adjustable power supply.

    b. Clock margins. For example, the VAX-11/780 CPU can run at a SLOW and FAST system clock rate. The CONSOL.SYS program has a command that specifies the desired VAX-11/780 CPU clock rate.

    These margins should be performed while running the appropriate device diagnostics.

8. Heat testing can be done, by blocking or disconnecting the appropriate fans, if the problem is suspected to be heat related. This should not be overdone since heat will damage components. This damage may not be seen immediately but may show up as an intermittent problem later on. I would suggest using Heat testing ONLY as a last resort.

9. You can COOL specific components/boards with Freon if you suspect the problem to be of this nature. Again, do not overdo.

10. Beware that any Foreign Equipment, that may be on the system, could be at fault. If all of the above checks O.K., request that the Customer remove the Foreign Equipment in order to eliminate it as a possible cause of failure.

11. Be sure to show the Customer any diagnostic failures that are due to the Foreign Equipment. If the problem is found to be the Foreign Equipment, the Customer may be billed "Per Call" rates (a Local Management decision).

12. If the system has MS/MA780 memory on it, replace or remove all arrays that are getting Single Bit Errors. The reason for this is due to the possibility of the following occuring:

    The ECC logic of the MS/MA780 memory controllers cannot correctly report the conditions in which an array has a MULTIPLE ODD NUMBER of BAD BITS (ex. 3,5,7, or etc. bad bits per 72 bit array word). The memory controller will correct a bit (not necessarily one of the bad bits) and will send out the data as "Corrected Read Data".

13. If the system has an FP780 Floating Point Accelerator on it, remove it and see if the problem still occurs. The VAX-11/780 microcode will execute all of the Floating Point instructions if the FP780 isn't present.

14. Parity is checked as it is received by the VAX-11/780 CPU Data Paths and is generated for the data going out of the Data Paths as just prior to it being transmitted.

    Therefore, data manipulations and transfers within the Data Paths don't have any type of parity checking done on them as they move between Data Path Registers, ALU's, SHIFTERs, etc..

    If all else fails to fix your problem, the VAX-11/780 Data Paths may be at fault.

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

VIBRATION Testing

Vibration testing is a valid way of verifying connections and internal component damage if done properly. It is not a valid test if you vibrate so hard that you either bend or damage the components under test. In fact, vibration testing that is done to hard may cause more problems.

Here are a few common sense rules that you should keep in mind when trouble-shooting via vibration testing.

1. Always run an appropriate diagnostic that will test the device that you are vibrating. You must know how to determine quickly that a failure has occured so that you will be able to determine what you vibrated at the time of the failure.

2. Always vibrate in sections. Do not make a big sweep of all the modules/components/backplane pins and expect to know what components caused the failure when vibrated.

3. Vibrate with enough force to jar the components under test, but do not use so much force that you damage components or backplanes.

4. Do not OVERDO vibration testing. To much vibrating will eventually loosen components, loosen connections, or fracture etches and wires. Vibrate enough to verify, to yourself, that the device is not vibrational and then don't vibrate any more.

5. Be very careful when vibration testing cables. Such cables as the SBI cables are easily damaged.

6. When vibrating backplane pins, use a non-conducting flat piece of material and drag it along the pins. Use common sense in applying pressure. Do not vibrate with so much pressure that you bend the pins together or with so much pressure that you cut the insulation on the wires that are against the pins. The object of backplane vibration testing is to determine if the following exists:

    a. Broken insulation on wires surrounding the pins that intermittently short against the pin.
    b. Poor pin to etch run connections.
    c. Poor pin to module connections.
    d. Free floating pieces of conducting material may be lodged within the backplane causing intermittent shorts.

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

OPERATING TEMPERATURE CHANGE Testing

Some problems occur more rapidly or only when the circuit components are warmed up or only when they are cool. In order to increase the failure rate or aid in isolating problems, the circuits can either be either heated above normal operating temperature or cooled below normal operating temperature. This can be accomplished in several ways.

It is important to realize the this type of testing may show up additional problems other than the one you are trouble-shooting.

Use this type of testing on only one device at a time.

Heat Testing

Heat is an enemy to electronic and most mechanical components. Therefore, moderation is the key to successful, non-damaging heat testing. Whenever you heat test a device or component, be sure that you don't overheat. It is best to only heat up a circuit a few degrees warmer than it is normally operating at. This can usually be accomplished by simply disconnecting or blocking fans temporarily. Constantly monitor the rise in temperature and reconnect or unblock the fans when the temperature rises extremely.

Make sure that you run the appropriate diagnostic that will exercise the device/component, under test, while heat testing.

If you are able to isolate a heat problem down to a few components, a Heat Gun, or a Hair Dryer, may make it easier for you to isolate the bad component.

Testing by Cooling

Extreme cold can also be an enemy to Electronic components and most Mechanical components. Moderation in cooling is also the key to successful testing by cooling. Excessive cooling can cause component damage, fractured etches and wires, etc. Cooling a circuit is not as easy to do as heating a circuit. A couple of ways that cooling can be accomplished are as follows:

1. Sometimes, simply opening a cabinet or removing of the device's skins can cool the system enough to cause failures.

2. The skins can be removed and an additional, free-standing, fan can be directed into the circuit. Beware, sometimes this actually increases the device's operating temperature since air flow is blocked or funneled in such a way that the proper air flow is not obtained.

3. If you are able to isolate the problem down to several boards/components/etc. you can then use canned Freon in order to isolate the problem further.

4. A "Carbon Dioxide Fire Extinguisher" can also be used if the unit under test is large.

Make sure that you run the appropriate diagnostic that will exercise the device/component, under test, while cooling.

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

MARGIN Testing

Margin testing is another means of sometimes increasing failure rate or isolating the problem area. There are two basic types of margins you can use on the VAX-11/780 system and its' devices. These are, Clock Margins and Voltage Margins. When you are doing any type of margining, be sure to run diagnostics that will exercise the device that you are margining.

It is important to realize that whenever any type of margining is done, you may find other problems other than the one you initially started trouble-shooting.

CLOCK Margins

On the VAX-11/780 system, the System Clock rate can be varied above and below the normal clock rate. This is set by a command to the CONSOL.SYS program. Simply set the desired Margin Clock Rate and then run functional diagnostics on the VAX-11/780 CPU, Memory, and SBI Nexus Controllers.

Other devices may contain clock margining facilities also.

Clock Margin only one device at a time.

VOLTAGE Margins

Any device that has Power Supplies that can be adjusted is capable of being Voltage Margined. When voltage margining, do not take the voltage above or below the specified component operating levels that that voltage supplies. Again, moderation is the key. Excessive voltage margining effects the life of electronic components. The reason for this is that a change in voltage causes a change in circuit current, which causes a change in heat dissapated by the circuit.

Be sure to run the appropriate diagnostics that will exercise the device being margined.

When voltage margining, be sure to adjust the voltages with a DVM and then return the voltage to the appropriate level upon completion of testing.

Voltage Margin only one voltage at a time.

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

D W 7 8 0 E R R O R S

Parity Fault - *UBA_CONFIG<31>* *"PAR FLT"*

If "Parity Fault - CONFIG<31>" only in this DW780.

M8270
Flakey Power for any NEXUS

If "Parity Fault - CONFIG<31>" in multiple NEXUS.

M8270, SBI Cables, Other NEXUS SBI Interface
SBI Terminator, Flakey Power for any NEXUS.

Write Sequence Fault - *UBA_CONFIG<30>* *"WSQ FLT"*

M8270, Other NEXUS
SBI Cables
Flakey Power for this NEXUS

Unexpected Read Data Fault - *UBA_CONFIG<29>* *"URD FLT"*

Other NEXUS, M8270
SBI Cables
Flakey Power for this NEXUS

Interlock Sequence Fault - *UBA_CONFIG<28>* *"ISQ FLT"*

Software
M8270, Other NEXUS
SBI Cables
Flakey Power for this NEXUS or CPU

Multiple Transmitter Fault - *UBA_CONFIG<27>* *"MXT FLT"*

If "Transmitter During Fault - UBA_CONFIG<26>" = 0

Another NEXUS
M8270, M8271, SBI cables
Flakey Power for any NEXUS

If "Transmitter During Fault - UBA_CONFIG<26>" = 1

M8270, M8271, Another NEXUS
SBI Cables
Flakey Power for any NEXUS

Adapter Power Down - *UBA_CONFIG<23>* *"AD PDN"*

H7100 Power Supply for this NEXUS
M8273
Input AC power.

Adapter Power Up - *UBA_CONFIG<22>* *"AD PUP"*

Normally asserted.

UNIBUS Power Down - *UBA_CONFIG<17>* *"UB PDN"*

Could be a legal entry if the UNIBUS box was powered off.
UNIBUS Power supply ACLO logic.
Any UNIBUS device that can assert ACLO.
Input AC power.
M9044

UNIBUS Power Up - *UBA_CONFIG<16>* *"UBIC"*

Normally asserted.

Read Data Timeout - *UBA_STATUS<10>* *"RDTO"*

SBI Memory, M8270, M8272, M8273
M8271, M9044, UNIBUS device that is requesting SBI Memory data.
Flakey Power for this NEXUS or for the UNIBUS device.

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Read Data Substitute - *UBA_STATUS<9>* *"RDS"*

SBI Memory Array.
SBI Memory NEXUS control.
M8270, SBI Cables, Flakey SBI Memory NEXUS power.

The UNIBUS device that was requesting the SBI Memory data will not receive the requested data, therefore, its' non-existent memory bit should be set.

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Corrected Read Data - *UBA STATUS<8>* *"CRD"*

SBI Memory Array.
SBI Memory NEXUS control.
M8270, SBI Cables, Flakey SBI Memory NEXUS power.

Command Transmit Error - *UBA_STATUS<7>* *"CXTER"*

M8270
M8271, M8272, M8273, M9044
NEXUS to which this UBA initiates a data transfer.
SBI Cables, Flakey Power for NEXUS or assoc. UNIBUS.
UNIBUS device issuing data transfer command.

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Command Transmit Timeout - *UBA_STATUS<6>* *"CXTMO"*

NEXUS to which this UBA initiates a data transfer.
M8270
M8271, M8272, M8273, M9044
UNIBUS device issuing data transfer command.
SBI Cables, Flakey Power

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Data Path Parity Error - *UBA_STATUS<5>* *"DPPE"*

M8272
M8270, M8271, M8273
Flakey power for this NEXUS

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Invalid Map Register - *UBA_STATUS<4>* *"IVMR"*

M8272
M8273, M9044, M8270, UNIBUS device requesting data transfer.
Software
Flakey power for this NEXUS.

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Map Register Parity Fail - *UBA_STATUS<3>* *"MRPF"*

M8272
M8270
Flakey power for this NEXUS.

The FMER register is locked on the occurrence of this error. The FMER contains the Map Register number associated with the failure. The FMER contents determine the number of the data path that failed.

Lost Error Bit - *UBA_STATUS<2>* *"LEB"*

This bit indicates that another error has occurred after the locking field has already been locked. The RDTO, RDS, CXTER, CXTMO, DPPE, IVMR, and MRPF bits form the locking field that locks the FMER.

UNIBUS Select Timeout - *UBA_STATUS<1>* *"UBSTO"*

Some UNIBUS device.
M9044, M8273, M8271
Flakey power for this NEXUS or the associated UNIBUS devices.
M8272

The FUBAR register is latched when this error occurs. It contains the upper 16 bits of the UNIBUS address translated from an SBI address.

UNIBUS SSYN Timeout - *UBA_STATUS<0>* *"UBSSYNTO"*

The UNIBUS device to which data transfer is taking place.
M8273, M9044, M8271
M8272
Flakey power for this NEXUS or the associated UNIBUS devices.

The FUBAR register is latched when this error occurs. It contains the upper 16 bits of the UNIBUS address translated from an SBI address.

Buffer Transfer Error - *UBA_DPR 0-15 Bit <30>* *"BTE"*

M8272, M8271
M8273
Flakey power for this NEXUS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

S . B . I . FAULTS

An S.B.I. FAULT condition can be caused by any one of the following conditions having been detected on the S.B.I. Bus:

Parity Fault -

An S.B.I. parity error can be detected on ANY cycle by ANY NEXUS.

The S.B.I. P<1:0> lines provide even parity for their associated groups of S.B.I. lines. "S.B.I. P<0>" provides EVEN Parity for the group of S.B.I. lines consisting of the TAG<2:0>, ID<4:0>, and the M<3:0> lines. "S.B.I. P<1>" provides EVEN Parity for the group of S.B.I. lines consisting of the B<31:00> lines.

Whenever a NEXUS detects a parity error, on any given S.B.I. cycle, ALL OTHER NEXUS should also detect the same parity error.

Write Sequence Fault -

Is the result when a NEXUS which has received a COMMAND/ADDRESS Cycle specifying any type of WRITE COMMAND, does not receive the anticipated WRITE DATA in the next sequential S.B.I. cycle(s).

This type of FAULT is only detected by the NEXUS to which the write command was sent.

Unexpected Read Data Fault -

Is the result when a NEXUS whose is not waiting for READ DATA receives READ DATA.

The destination of the READ DATA is specified by the S.B.I. ID<4:0> lines. Each device checks the ID field on all Read Data cycles to see if the B<31:00> lines contain data that is being sent to them.

Only the NEXUS receiving the Unexpected Read Data detects the FAULT.

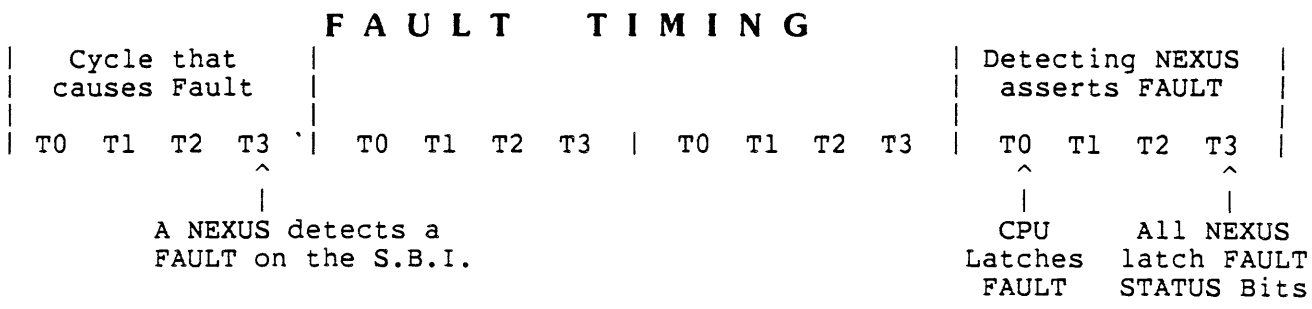
Interlock Sequence Fault -

Is the result when a NEXUS receives an INTERLOCK WRITE COMMAND and the INTERLOCK has not been set by an INTERLOCK READ Command.

Only the NEXUS recieving the INTERLOCK WRITE Command will detect this fault.

Multiple Transmitter Fault -
Is the result when a TRANSMITTING NEXUS detects multiple transmitters in the same cycle that it is transmitting. This is detected by comparing the ID<4:0> field sent, at S.B.I. T0 time, with the ID<4:0> field recieved, at S.B.I. T3 time. If they do not match, this fault is detected.

Detected by only the NEXUS that are transmitting in the faulting cycle.



Trouble-Shooting S.B.I. FAULTS:
Whenever an S.B.I. FAULT occurs, each NEXUS will latch its FAULT STATUS bits which are contained in its CONFIGURATION/STATUS REGISTER (the 1st I/O register for each NEXUS) Bits <31:26>. These bits indicate the type of FAULT that the respective NEXUS has detected. It is very important to gather the contents of these registers immediately after the FAULT occurs. If the FAULT occured while running VMS, the needed register information may be saved in ERRLOG.SYS. It is still a good idea to modify the DEFBOO.CMD and the RESTAR.CMD command files, located on the LOCAL/REMOTE CONSOLE Floppy, so that a complete register dump will be done prior to rebooting. If the FAULT occured while running diagnostics, you must do the register dump yourself. The register dump can be taken by using the following commands that are appropriate for the system.

```
>>> E/L/H/P 20002000          ! Examine TR#1  Configuration Register
>>> E/L/H/P 20004000          ! Examine TR#2  Configuration Register
>>> E/L/H/P 20006000          ! Examine TR#3  Configuration Register
>>> E/L/H/P 20008000          ! Examine TR#4  Configuration Register
>>> E/L/H/P 2000A000          ! Examine TR#5  Configuration Register
>>> E/L/H/P 2000C000          ! Examine TR#6  Configuration Register
>>> E/L/H/P 2000E000          ! Examine TR#7  Configuration Register
>>> E/L/H/P 20010000          ! Examine TR#8  Configuration Register
>>> E/L/H/P 20012000          ! Examine TR#9  Configuration Register
>>> E/L/H/P 20014000          ! Examine TR#10 Configuration Register
>>> E/L/H/P 20018000          ! Examine TR#12 Configuration Register
>>> E/L/H/P 2001A000          ! Examine TR#13 Configuration Register
>>> E/L/H/P 2001C000          ! Examine TR#14 Configuration Register
>>> E/L/H/P 2001E000          ! Examine TR#15 Configuration Register
>>> E/L/H/ID 1B               ! Examine CPU's Configuration Register
```

Of course you do not need to attempt to examine the registers of the TR's in which there are no devices for your specific system configuration.

In addition each NEXUS latching up its FAULT STATUS bits in its CONFIGURATION/STATUS register, the KA780 CPU will latch the "S.B.I. SILO". The "S.B.I. SILO" is a 16 location RAM that contains the states of certain S.B.I. signals for the last 16 S.B.I. cycles. The "SILO" is located in the KA780's S.B.I. control logic. This SILO can be examined by reading "ID register #18" 16 times. The first word stored in the SILO will be the first word read out, the second word stored will be the second word read out, etc.. You can use the following CONSOL.SYS commands to gather the contents of the "S.B.I. SILO":

Method #1

```
>>> R E/L/ID 18          ! Repeat examine ID#18.
>>> ^C                   ! Type "CTRL C" after 16 examines.
```

Method #2

```
>>> E/L/ID 18            ! Repeat this command 16 times.
```

Now that you have all the needed information, all that remains is to breakdown the registers to determine the type of S.B.I. FAULT. Start off by breaking down the CONFIGURATION/STATUS Registers, (only need bits <31:26>), to determine the type of FAULT and to find out which NEXUS(es) detected it. Once you have found the type of FAULT and who all detected it, you should be able to determine which one or two NEXUS are most likely at fault. Now you can break down the SILO information to see if it will help determine who is the most likely device at fault.

When looking at the S.B.I. SILO dump, break down each entry completely, and then check to see what went wrong. You must know how the S.B.I. works in order to do this.

Beware: Any NEXUS can intermittently polute the S.B.I. and can cause the problem to appear to be someone else's fault.

SILO Interpretation - ID #18

```
    3 3 2 2  2 2 2 2  2 2 2 2  1 1 1 1  1 1 1 1  1 1 0 0  0 0 0 0  0 0 0 0
    1|0|9 8  7 6 5|4  3 2|1 0  9 8|7 6| 5 4 3 2  1 0 9 8  7 6 5 4  3 2 1 0|
     |I|          |       |          |   |                                  |
After|T|    ID    |  TAG  |   MASK   |CNF|            TR                    |
Fault|L|          |       |    or    |   |                                  |
     |K|          |       |FUNCTION  |   |                                  |
```

****

Bit <31> is the "After Fault" bit which will be set only on the first SILO entry after the S.B.I. FAULT condition has been cleared.

****

Bit <30> is the "S.B.I. Interlock" bit. The S.B.I. Interlock line is asserted by the commanding nexus when issuing an interlock read and then by the recieving nexus upon assertion of the ACK confirmation.

****

Bits <29:25> reflect the "S.B.I. ID<4:0> lines" which indicate the logical source or intended destination of the information on the B<31:0> lines. These lines reflect the hex representation of the TR level.

TAG not = 0 then, ID<4:0> reflect the source.
TAG = 0 then, ID<4:0> reflect the destination.

****

Bits <24:22> reflect the state of the "S.B.I. TAG<2:0>" lines. The TAG lines indicate the type of cycle being transmitted on the S.B.I. bus...

TAG = 0 Read Data Cycle
TAG = 3 Command/Address Cycle
TAG = 5 Write Data Cycle
TAG = 6 Interrupt Summary Read Cycle

****

Bits <21:18> reflect the state of the MASK <3:0> lines if the TAG isn't equal to 3 (indicating a Command/Address Cycle), in which case it will reflect the state of the B<31:28> lines (which indicate the FUNCTION).

TAG = 3 then, Bits <21:18> reflect the type of FUNCTION.

FUNCTION = 1 - Read Masked Function
FUNCTION = 2 - Write Masked Function
FUNCTION = 4 - Interlock Read Masked Function
FUNCTION = 7 - Interlock Write Masked Function
FUNCTION = 8 - Extended Read Function
FUNCTION = B - Extended Write Masked Function

TAG = 0,6 then, Bits <21:18> reflect the M<3:0> lines.

MASK = 0 - Read Data in B<31:00> is good data.

MASK = 1 - A single bit error was detected and corrected by the transmitting NEXUS, therefore, the data in B<31:00> is now good.

MASK = 2 - Multiple bits were detected as being bad and the transmitting NEXUS could not repair them. Therefore the data in B<31:00> is BAD.

TAG = 5 then, Bits <21:18> reflect the M<3:0> lines.

The MASK <3:0> lines reflect the respective BYTE(s) that are being written.

****

Bits <17:16> reflect the state of the "S.B.I. CNF<1:0> lines". These lines are the confirmation lines. Each and every transmitted cycle must have a corresponding confirmation code returned two cycles later.

CNF = 0 - NO RESPONSE from destination NEXUS

CNF = 1 - Indicates an ACKNOWLEDGE from the destination NEXUS. If this confirmation is received two cycles after a C/A cycle, it indicates that the device knows the command is to him and also indicates that he can perform the specified command encoded in the B<31:28> lines.

CNF = 2 - Indicates that the destination NEXUS knows you want him to do something, but he is currently BUSY. The transmitting NEXUS should try the command again later.

CNF = 3 - Indicates that the destination NEXUS recognizes that the transmitting NEXUS is talking to him but the command encoded in the B<31:00> lines specifies a function that the destination NEXUS cannot perform.

****

Bits <15:00> reflect the state of the "S.B.I. TR<15:00> lines".

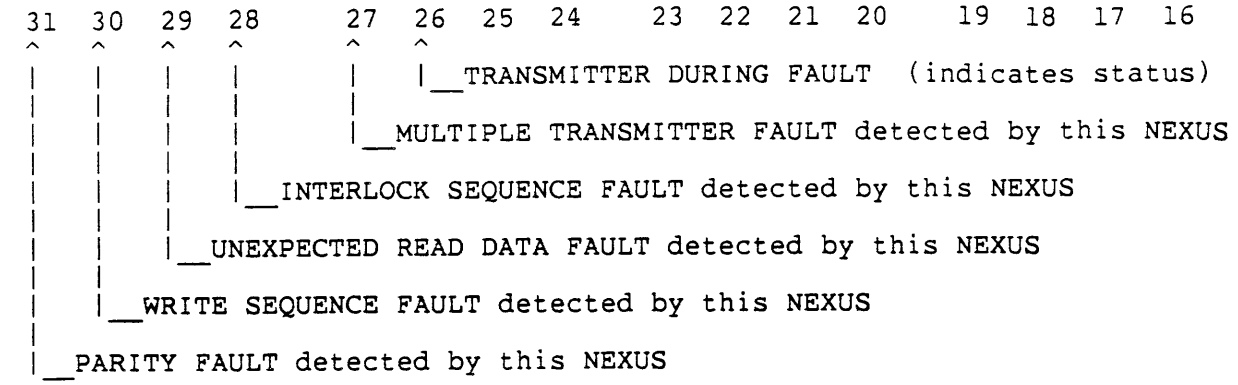
If a bit is set, it indicates that the respective TR arbitration line is asserted on the S.B.I., for that cycle.

****

BEWARE:

In some cases, the S.B.I. SILO may not contain valid information because of improper setup prior to the FAULT occuring. In order to verify that the SILO contains valid information, check ID #1B and make sure that Bit 16=1. If it isn't, don't bother checking the SILO information since it will not indicate what happened just prior to the fault.

CONFIGURATION/STATUS Register Interpretation



The CONFIGURATION/STATUS Register is the first I/O Address assigned to each NEXUS. The CPU's equivalent register is ID #1B. All NEXUS, and the CPU, have the above assignments for these bits within their CONFIGURATION/STATUS registers. If a device cannot detect any particular type of FAULT, it will not have anything assigned to that particular bit and that bit will always be read as ZERO.

Device at	-	Configuration/Status Register
TR #1	-	20002000
TR #2	-	20004000
TR #3	-	20006000
TR #4	-	20008000
TR #5	-	2000A000
TR #6	-	2000C000
TR #7	-	2000E000
TR #8	-	20010000
TR #9	-	20012000
TR #10	-	20014000
TR #11	-	20016000
TR #12	-	20018000
TR #13	-	2001A000
TR #14	-	2001C000
TR #15	-	2001E000
CPU	-	ID #1B

```
EVKAA - V5.1 done
EVKAA - V5.1 done

HALTED AT 1805

>>>
```

Trouble-shooting

using the

SYSTEM CONTROL BLOCK

```
$


?ILL I/E VEC
HALTED AT 100


>>>
```

Unexpected EXCEPTIONs or INTERRUPTs can result in a variety of symptoms. Typically, programmers will setup a trap-catcher, in the SYSTEM CONTROL BLOCK, in order to prevent these unexpected events from destroying their program or data. The trap-catcher will either flag the error and continue on, or will cause the CPU to halt. Depending upon how the software is setup the symptom may be a "HALTED AT xxxxxxxx" message printed on the console terminal, a "FATAL BUGCHECK" or "NON-FATAL" bugcheck message on the console or users terminal, an "?ILL I/E VEC" message is printed on the console terminal followed by the CONSOL.SYS prompt, or the system software may do unexpected things due to re-vectoring.

Even though the symptoms are many, determing if you have a problem in this area is easily done by examination of the SYSTEM CONTROL BLOCK, examination of a listing of the failing software, using the data typed out with the "?ILL I/E VEC" messager, or by installing a trap-catcher and then reproducing the problem.

HALTED AT xxxxxxxx

If the symptom results in a "HALTED AT xxxxxxxx" being printed on the console terminal, followed by the CONSOL.SYS prompt (>>>), either a listing of the software or examination of the SYSTEM CONTROL BLOCK is necessary. A listing of the software being run is probably the easiest method of determing what went wrong. Simply look up the "xxxxxxxx" PC that was printed out in the "HALTED AT xxxxxxxx" message.

If you don't have a listing of the failing software, examination of the SCB is necessary. To do this, you must first examine the SCCB register in order to find the start of the SCB. The SCCB (ID #3B) contains a longword aligned PHYSICAL ADDRESS that points to the beginning of the SYSTEM CONTROL BLOCK (SCB). In order to dump the SCB, use the following CONSOL.SYS commands:

```
>>> E/L/ID 3B
>>> E/L/P/N:7F @
```

Now check to see if the "xxxxxxxx" PC, in the "HALTED AT xxxxxxxx" message, is somewhere in the address range of the SCB. If it is, the offset from the SCBB can be used to determine what type of EXCEPTION or INTERRUPT caused the problem by using the SCB vector assignment chart on the following pages.

If the "xxxxxxxx" PC isn't in the address range of the SCB table, check to see if any of the longword vectors can re-vector the software to a location near this address. If there is, examine memory starting at the vector address up to the "xxxxxxxx" address. Does the macrocode indicate that re-vectoring could cause a halt at the specified PC? If so, use the SCB vector assignment chart to determine what type of EXCEPTION or INTERRUPT caused the problem.

Building and using a VAX Trap Catcher

In some cases, it may be necessary to modify the SCB so as to form a TRAP CATCHER. The idea is to deposit vectors into the SCB that will cause a halt whenever an EXCEPTION or ITERRUPT occurs. Furthermore, this must result in causing the system to halt at an address unique to EXCEPTION or INTERRUPT. The easiest way to do this is create a SCB with each vector pointing to its respective SCB location, with both bits <1> and <0> set to a 1 (if bits <1:0>=3, the VAX-11/780 will halt with a PC one byte past the specified vector address).

After such a Trap Catcher is installed, the symptom must be reproduced. Any EXCEPTIONs or INTERRUPTs that occur should then cause the CPU to halt followed by a "HALTED AT xxxxxxxx" message printed on the console terminal. The "xxxxxxxx" PC will indicate what type of EXCEPTION or INTERRUPT occured by determining what vector was used. This can be determined by subtracting the address of the vector used from the contents of the SCBB, and then using the SCB vector assignment charts to determine the type of EXCEPTION or INTERRUPT.

Note about installing Trap Catchers

The only problem with this method of trouble-shooting is that care must be taken not to modify the vectors for EXCEPTIONs and INTERRUPTs that the software program uses. Only those that software doesn't expect to happen should be modified.

VMB V4.02 Trap Catcher generation

Unexpected EXCEPTIONs or INTERRUPTs can cause system software booting failures. If the booting failures are in VMB, a unique way of modifing the SCB is required due to VMB's use of the SCB.

In order to use this procedure, you will have to manually perform the CONSOL.SYS commands of the command file used to boot the system, with the following modifications:

Use the same commands as the appropriate boot file used to boot the system, but don't use the "WAIT DONE" command, and DON'T START VMB YET.

After VMB has been loaded, patch VMB by first finding the contents of the SP (this was examined just prior to loading of VMB if you followed the normal boot process).

```
>>> SET RELOCATION:@
>>> E/L/P 2400           ! Should contain an xxxxCF9E.
>>> D/L/P 2400 2038F3C   ! Replace with a MOVZWL #203,R6.
>>> E/W/P 240A           ! Should contain a 56D0.
>>> D/W/P 240A 76DE      ! Replace with a MOVAL -(R6),-(R7).
>>> SET RELOCATION:0
>>> E/L SP
>>> START @
```

If an unexpected EXCEPTION or INTERRRUPT occurs while VMB is running, with the above patch, a "HALTED AT xxxxxxxx" message will be typed on the console terminal. Now save the "xxxxxxxx" in this message and do the following procedure:

Examine the SCBB with the following command:

>>> E/L/ID 3B

Subtract the saved "xxxxxxxx" from the contents of ID #3B.

The result is the offset into the SCB for the EXCEPTION or INTERRUPT that occured. Use the SCB vector assignment chart to determine the type of unexpected EXCEPTION or INTERRUPT.

Note about VMS

None of these procedures should need to be used with VMS. VMS will report unexpected EXCEPTIONs and INTERRUPTs by way of its BUGCHECK routine.

?ILL I/E VEC

This is the easiest to trouble-shoot. The data printed out with this error message is the offset into the SCB. Use the SCB vector assignment chart to determine what type of EXCEPTION or INTERRUPT caused the failure.

Now what?

If one of these procedures have isolated the problem to a certain type, then go to the section of this chapter that deals with that type of problem.

i.e. If the vector was at SCBB+04, the failure was due to a MACHINE CHECK, therefore use the MACHINE CHECK portion of this guide to find out how to trouble-shoot this problem.

If the vector was at SCBB+5C, the failure was due to an SBI FAULT, therefore use the SBI FAULT portion of this guide to find out how to trouble-shoot this problem.

If the vector was at one of the SBI REQUEST level locations, the device at fault can be isolated by determining what REQUEST LEVEL, and what TR Level the interrupt occured on. Knowledge of the system configuration, plus these two levels, should narrow down the suspects.

VAX-11/780 System Control Block vector assignments

000	Unused, reserved
004	Machine Check
008	Kernel Stack Not Valid
00C	Power Fail
010	Reserved/Privileged Instr.
014	Customer Reserved Instr.
018	Reserved Operand
01C	Reserved Addressing Mode
020	Access Control Violation
024	Translation Not Valid
028	Trace
02C	Breakpoint
030	Compatibility
034	Arithmetic
038	Unused, reserved
03C	Unused, reserved
040	CHMK
044	CHMB
048	CHMS
04C	CHMU
050	SBI Silo Compare
054	CRD/RDS
058	SBI ALERT
05C	SBI FAULT
060	Asynchronous Write
064	Unused, reserved
068	Unused, reserved
06C	Unused, reserved
070	Unused, reserved
074	Unused, reserved
078	Unused, reserved
07C	Unused, reserved
080	Unused, reserved
084	Software Level 1
088	Software Level 2
08C	Software Level 3
090	Software Level 4
094	Software Level 5
098	Software Level 6
09C	Software Level 7
0A0	Software Level 8
0A4	Software Level 9
0A8	Software Level A
0AC	Software Level B
0B0	Software Level C
0B4	Software Level D
0B8	Software Level E
0BC	Software Level F
0C0	Unused, reserved
0C4	Unused, reserved
0C8	Unused, reserved
0CC	Unused, reserved
0D0	Unused, reserved
0D4	Unused, reserved
0D8	Unused, reserved
0DC	Unused, reserved
0E0	Unused, reserved
0E4	Unused, reserved
0E8	Unused, reserved
0EC	Unused, reserved
0F0	Unused, reserved
0F4	Unused, reserved
0F8	CNSL Receive Interrupt
0FC	CNSL Transmit Interrupt
100	SBI REQ 4 - TR #0
104	SBI REQ 4 - TR #1
108	SBI REQ 4 - TR #2
10C	SBI REQ 4 - TR #3
110	SBI REQ 4 - TR #4
114	SBI REQ 4 - TR #5
118	SBI REQ 4 - TR #6
11C	SBI REQ 4 - TR #7
120	SBI REQ 4 - TR #8
124	SBI REQ 4 - TR #9
128	SBI REQ 4 - TR #10
12C	SBI REQ 4 - TR #11
130	SBI REQ 4 - TR #12
134	SBI REQ 4 - TR #13
138	SBI REQ 4 - TR #14
13C	SBI REQ 4 - TR #15
140	SBI REQ 5 - TR #0
144	SBI REQ 5 - TR #1
148	SBI REQ 5 - TR #2
14C	SBI REQ 5 - TR #3
150	SBI REQ 5 - TR #4
154	SBI REQ 5 - TR #5
158	SBI REQ 5 - TR #6
15C	SBI REQ 5 - TR #7
160	SBI REQ 5 - TR #8
164	SBI REQ 5 - TR #9
168	SBI REQ 5 - TR #10
16C	SBI REQ 5 - TR #11
170	SBI REQ 5 - TR #12
174	SBI REQ 5 - TR #13
148	SBI REQ 5 - TR #14
17C	SBI REQ 5 - TR #15
180	SBI REQ 6 - TR #0
184	SBI REQ 6 - TR #1
188	SBI REQ 6 - TR #2
18C	SBI REQ 6 - TR #3
190	SBI REQ 6 - TR #4
194	SBI REQ 6 - TR #5
198	SBI REQ 6 - TR #6
19C	SBI REQ 6 - TR #7

```
1A0     SBI REQ 6 - TR #8
1A4     SBI REQ 6 - TR #9
1A8     SBI REQ 6 - TR #10
1AC     SBI REQ 6 - TR #11
1B0     SBI REQ 6 - TR #12
1B4     SBI REQ 6 - TR #13
1B8     SBI REQ 6 - TR #14
1BC     SBI REQ 6 - TR #15
1C0     SBI REQ 7 - TR #0
1C4     SBI REQ 7 - TR #1
1C8     SBI REQ 7 - TR #2
1CC     SBI REQ 7 - TR #3
1D0     SBI REQ 7 - TR #4
1D4     SBI REQ 7 - TR #5
1D8     SBI REQ 7 - TR #6
1DC     SBI REQ 7 - TR #7
1E0     SBI REQ 7 - TR #8
1E4     SBI REQ 7 - TR #9
1E8     SBI REQ 7 - TR #10
1EC     SBI REQ 7 - TR #11
1F0     SBI REQ 7 - TR #12
1F4     SBI REQ 7 - TR #13
1F8     SBI REQ 7 - TR #14
1FC     SBI REQ 7 - TR #15
```

The SCBB register contains a longword aligned address that points to the first vector in the System Control Block.

If bits <1:0> are both set to a one in the vector, and the CPU traps to that vector, the VAX-11/780 will HALT. This is an easy way of developing a VAX-11/780 Trap Catcher.

SECTION II

VMS Information

VMS SYSGEN Error Control Parameters

On certain types of errors, the VMS Operating System will attempt to reboot itself. There are a couple of "SYSGEN" parameters that can disable this function. It is very important to have these parameters set correctly when trouble-shooting a problem so that the appropriate Hardware and Software information can be gathered.

BUGCHECKFATAL is a parameter that enables, when set to a 1, the conversion of NONFATAL Bugchecks to FATAL Bugchecks. This causes the Operating system to crash and reboot. Setting this parameter to a 1 is useful whenever ERRLOG.SYS does not give you enough information in order to diagnose the problem. When set to a 1, and BUGREBOOT is set to a 0, the Operating System will not reboot. Therefore, you will then be able to take a Hardware Register Dump or do some scoping. It has no effect on bugchecks from USER or SUPERVISOR mode.

BUGREBOOT is a parameter that enables, when set to a 1, the automatic rebooting of the Operating System if a FATAL BUGCHECK occurs. It may become necessary to clear, set to a 0, this parameter after the first FATAL Bugcheck occurs so that Hardware Register Dumps can be taken, and so that scoping may be done whenever the FATAL Bugcheck occurs. BUGREBOOT=1 causes the LSI to reboot the VAX via DEFBOO.CMD or RESTAR.CMD.

DUMPBUG is a parameter that enables, when set to a 1, the writing of error log buffers and memory contents to SYS$SYSTEM:SYSDUMP.DMP when a FATAL Bugcheck occurs. The Software Dump is written to the "SYS$SYSTEM:SYSDUMP.DMP" file immediately after the Operating System detects the Error and before the "BUGREBOOT" bit is checked to see if a reboot is to be attempted. In other words, the Software Dump is taken on the way down.

The state of these "SYSGEN" parameters should be checked if you are not getting all the dump information that you require. The customer should have the system set up so that the Software Dump, a function of the "DUMPBUG" parameter, is always taken. The State of the other two parameters depends on the Customer's operating enviroment. However, once a problem occurs, it will probably be necessary to change the setting of these parameters until the problem is fixed.

No matter how the SYSGEN parameters are set up, certain types of failures will crash the system in such a way that the Software DUMP will not be taken.

V M S C R A S H H A N D L I N G

The following is an overview of what happens when VMS detects a NON-FATAL BUGCHECK.

1. A message describing the error is passed to the Error Logger.
2. System operation continues.

The following is an overview of what happens when VMS detects a FATAL BUGCHECK from USER or SUPERVISOR .

1. A message describing the error is passed to the Error Logger.
2. Execution continues as follows:
    a. If the process is executing a single image, the process is deleted.
    b. If the process is executing in interactive or batch mode, the current image exits and control is passed to the CLI to receive the next command..

The following is an overview of what happens when VMS detects a FATAL BUGCHECK from KERNEL or EXECUTIVE modes.

---

1. A small amount of information describing the Bugcheck is typed on the Console Terminal. This information may include the following:

   a. The contents of the VAX General Registers.

   b. The Kernel and Executive stack contents.

   c. The contents of certain VAX Internal registers.

   d. A summary of the reason for the Bugcheck.

2. If the "DUMPBUG" SYSGEN parameter is set to a 1, a software dump file will be written to SYS$SYSTEM:SYSDUMP.DMP. This file contains the following information:

   a. The Dump Header. This header contains such information as the contents of certain VAX registers, the Time of the crash,bugcheck crash code, and other information.

   b. The contents of the two errlog buffers.

   c. The contents of physical memory.

3. If the "BUGREBOOT" SYSGEN parameter is set to a 1, a VMS system reboot will be attempted as follows:

   a. VMS sends a special code to CONSOL.SYS (^XF02) via the console transmit buffer register (TXDB).

   b. VMS executes a HALT instruction so as to transfer control back to the CONSOL.SYS program.

   c. CONSOL.SYS attempts to reboot the VAX via the "DEFBOO.CMD" command file if "AUTO RESTART switch = 0" or via the "RESTAR.CMD" command file if "AUTO RESTART switch = 1". If the appropriate command file is not found, CONSOL.SYS prints its' prompt (>>>) and awaits operator input.

If the "BUGREBOOT" SYSGEN parameter is set to a 0, the VMS operating system will simply loop (with IPL=31) and await input on the console terminal. The "CONSOL.SYS" program prints its' prompt (>>>) on the console terminal and awaits operator input.

Assigning Addresses and Vectors to Unibus Devices

In order to make the proper assignments to those Unibus devices that have Floating Addresses and/or Floating Vectors, you must first understand the SYSGEN rules for configuration. SYSGEN uses the following rules for configuration:

- Devices with fixed CSR addresses and fixed vector addresses must be attached according to the SYSGEN device table settings.
- Devices with floating CSR or vector addresses must be attached in the order in which they are listed in the SYSGEN device table.
- An 8-byte gap must be reserved between each different type of device that is located in floating CSR address space.
- An 8-byte gap must be reserved in floating CSR address space for each device type that has no controller in its configuration.
- An extra 8-byte gap must be reserved between the KW11C and the RX11 in floating CSR address space.

The SYSGEN Device table is found in the "VAX/VMS Guide to Writing a Device Driver" manual (Order no. AA-H499C-TE), in the chapter about "Loading a Device Driver".

Unless the Unibus Devices, that have floating address and/or vectors, are properly assigned, the SYSGEN "AUTOCONFIGURE ALL" command will not be able to properly assign the devices. In these cases, the devices must be connected individually.

By using the above rules and the tables on the following page, you should be able to properly configure Floating Unibus Devices on the VAX-11/780 system.

The tables, on the following page,list the devices in the order that SYSGEN looks for them if they have floating address and/or vector assignments.

NOTE:
An alternate method of determining where to configure UNIBUS devices is to use the "CONFIG" command under "SYSGEN", (providing that you have a system, running VMS, that you can do this on). See the "SYSGEN" help file, in this chapter, for help on using this command.

Unibus Device Floating Address Table	Unibus Device Floating Vector Table
760010	300
DJ11	DC11
DH11	TU58
DQ11	DN11
DU11	DM11B
DUP11	DR11C
LK11	PR611
DMC11/DMR11 (DMCs before DMRs)	PP611
DZ11/DZ32 (DZ11s before DZ32s)	DT11
KMC11	DX11
LPP11	DL11C
VMV21	DJ11
VMV31	DH11
DWR70	GT40
RL11	LPS11
LPA11 (2nd)	DQ11
KW11C	KW11W
RSV	DU11
RX211	DUP11
DR11W	DV11
DR11B (3rd)	LK11
DMP11	DMC11/DMR11
DPV11	DZ11
ISB11	DZ32
DMV11	KMC11
UNA	LPP11
UDA	VMV21
DMF32 (see TECH. Manual)	VMV31
KMS11	DWR70
	RL11
	TS11
	LPA11
	KW11C
	RSV
	RX211
	DR11W
	DR11B (2nd & 3rd)
	DMP11
	DPV11
	ISB11
	DMV11
	UNA
	UDA
	DMF32
	KMS11
	PLC11

SYSGEN Commands

The following commands are those SYSGEN commands that manipulate drivers.

- LOAD (requires CMKRNL privilege)
- CONNECT (requires CMKRNL privilege)
- RELOAD (requires CMKRNL privilege)
- SHOW/ADAPTER (requires CMEXEC privilege)
- SHOW/CONFIGURATION (requires CMEXEC privilege)
- SHOW/DEVICE (requires CMEXEC privilege)
- AUTOCONFIGURE ALL (requires CMKRNL privilege)

LOAD command

This command is used to load a DEVICE DRIVER. If the CONTROLLER has only a single unit attached to it, issue the CONNECT command.

Format:

```
LOAD driver_file_spec
```

CONNECT Command

This command creates I/O base control blocks for devices. It can also load the driver if it has not been previously loaded into System memory.

Format:

```
CONNECT device_name required_quals optional_quals
```

Required Qualifiers:

```
/[NO]ADAPTER=nexus
/CSR=csr_address
/VECTOR=vector_address
```

Optional Qualifiers:

```
/NUMVEC=number_interrupt_vectors
/DRIVERNAME=driver_name
/ADPUNIT=unit_number
/MAXUNITS=maximum_number_of_units
```

RELOAD Command

This command loads a driver and removes a previously loaded version of that driver. Performs the same function as the LOAD command except it will load the driver regardless of whether it is already loaded.

Format:

```
RELOAD driver_file_spec
```

SHOW/ADAPTER Command

This command displays nexus numbers and generic names of the Unibus and Massbus adapters, memory controllers, and device interconnects such as the DR32.

Format:

```
SHOW/ADAPTER
```

SHOW/CONFIGURATION

This command displays information about the system configuration.

Format:

```
SHOW/CONFIGURATION      [/ADAPTER=nexus]
                        [/COMMAND_FILE]
                        [/OUTPUT=file_spec]
```

SHOW/DEVICE

This command displays the location of a driver and the I/O data base describing its devices in system virtual memory.

Format:

```
SHOW/DEVICE  [=driver_name]
```

AUTOCONFIGURE ALL

Configures D.E.C. supported devices to the system automatically.

Format:

```
AUTOCONFIGURE ALL
```

CONFIGURE [/INPUT=file-spec] [/OUTPUT=file-spec][/(NO)RESET]

This command request the UNIBUS device names and then outputs the set of CSR and VECTOR addresses that are required for AUTOCONFIGURE to use.

When executing this command, SYSGEN prompts you with "DEVICE>". Enter the device names in the following format:

```
device,n,p      ; where "device" = device's name
                ; and "n" = how many of this device, and
                ; "p" = the optional number of devices on all
                ; previous UNIBUSes in a multiple UNIBUS
                ; system.
```

This command can be used as follows to determine where UNIBUS devices should be addressed:

```
SYSGEN> CONFIGURE <return>
DEVICE> <device,n,p> <return>
DEVICE> <device,n,p> <return>

continue for all Unibus devices, and then:

DEVICE> ^Z

SYSGEN will then print the desired configuration.
```

CONNECT CONSOLE

Connects the Console Floppy Drive and loads its driver.

CREATE file-spec /SIZE=block-count [/(NO)CONTIGUOUS]

Creates or extends a paging, swapping, or dump file.

DISABLE CHECKS

Disables range checks.

ENABLE CHECKS

Enables range checks.

EXIT

Terminates SYSGEN. Go back to the VMS DCL prompt.

INSTALL file-spec ./PAGEFILE /SWAPFILE

Activates a secondary paging or swapping file.

SET /OUTPUT [=] file-spec

Defines an output file for the SYSGEN session.

SET /STARTUP file-spec

Names the current site-independent startup command procedure.

SHARE MPMn mpm-name

This command connects multiport memory units and initializes them to the Operating System.

/INITIALIZE	/GLBSECTIONS=glb
/MAILBOXES=mail	/MAXGLBSECTIONS=max-glb
/MAXMAILBOXES=max-mail	/POOLBCOUNT=block-cnt
/POOLBSIZE=block-size	/PRQCOUNT=prq-cnt
/CEFCLUSTERS=cef	/MAXCEFCLUSTERS=max-cef

SHOW parameter /xxx

Where xxx can be any of the following:

/ACP	/ALL	/DYNAMIC
/GEN	/JOB	/MAJOR
/NAMES	/PQL	/RMS
/SCS	/SPECIAL	/SYS
/TTY	[/HEX]	

Displays the values of the system parameters in the SYSGEN work area, plus the default, minimum, and maximum values of the parameter and their units of measure.

SET parameter-name value

Modifies the value of a system generation parameter in the SYSGEN work area.

SHOW /UNIBUS

Displays the addresses in UNIBUS I/O space that can be addressed.

USE file-spec CURRENT ACTIVE DEFAULT

Initializes the SYSGEN work area with system parameter values from a parameter file, the current system image, the active system, or the default list.

WRITE file-spec CURRENT ACTIVE

Writes the system parameter values from the SYSGEN work area to a parameter file, the current system image, or the active system.

Using SYSGEN to determine UNIBUS device Address/Vector Assignments

The SYSGEN "CONFIG" command can be used to determine the proper address and vector assignments for the VAX UNIBUS devices. The following steps can be used to do this.

1. Log into the VMS operating system.

2. Execute the following commands:

```
$ MCR SYSGEN
SYSGEN> CONFIG
DEV> device_name,number_of_devices        <-- Enter all device names,
DEV> next device_name,number_of_devices       one device type per
DEV> next device_name,number_of_devices       "DEV>" prompt.
DEV> ^Z                                   <-- "^Z" to end input mode.
```

3. When you type the "^Z", SYSGEN will determine the correct addresses and vectors for the devices and will print them out on your terminal. The devices must be assigned these addresses in order for the SYSGEN utility to be able to auto-configure them.

LOCAL CONSOLE Boot Command Files

The Local Console Floppy contains the command files necessary to boot either VMS or the Diagnostic Supervisor. These command files do four important things:

1. Initialize the VAX-11/780 CPU and the S.B.I. Nexus.

2. Setup the VAX-11/780 CPU's General Registers in such a way as to tell VMB.EXE who to boot from, what to boot, and how to start what was booted.

3. Initiate the ISP rom program to find a good 64K chunk of Memory.

4. Load and Start the VMB.EXE program.

Following is an example of one of the Local Console Boot command files. This command file boots VMS from Massbus drive #0 on RH780 #0 (TR=8).

```
!
!       DB0 Boot command file - DB0BOO.CMD
!
HALT                        ! Halt the Processor.
UNJAM                       ! Unjam the SBI.
INIT                        ! Init the Processor.
DEPOSIT/I 11 20003800       ! Set-up the SCBB.
DEPOSIT R0 0                ! Disk Pack Device Type.
DEPOSIT R1 8                ! MBA TR=8.
DEPOSIT R2 0                ! Adapter Unit = 0.
DEPOSIT R3 0                ! Controller Unit = 0.
DEPOSIT R4 0                ! Boot Block LBN (unused)
DEPOSIT R5 0                ! Software Boot Flags
DEPOSIT FP 0                ! Set no Machine Check expected.
START 20003000              ! Start ROM Program.
WAIT DONE                   ! Wait for Completion.
EXAMINE SP                  ! Show address of working Memory+^X200.
LOAD VMB.EXE/START:@        ! Load the Primary Bootstrap
START @                     ! and start it.
```

The parameters for VMB.EXE are described on pages 67 thru 70 of the VAX Systems Maintenance Handbook (EK-VAXV1-HB-001).

RESTAR.CMD

This command file is invoked in the event of Power Recovery and other console detected restart conditions if the "AUTO RESTART" switch is set. It can also be invoked manually with the following command to CONSOL.SYS:

```
>>> @RESTAR.CMD
```

The following RESTAR.CMD command file is an example of the type of restart file used for systems without interleaved memory:

```
HALT                                ! HALT the Processor.
UNJAM                               ! UNJAM the SBI.
INIT                                ! INITialize the Processor.
DEPOSIT/I 11 20003800               ! Set address of SCBB.
DEPOSIT R0 0                        ! Clear unused Register.
DEPOSIT R1 xxx                      ! xxx=TR of Boot Disk NEXUS.
DEPOSIT R2 0                        ! Clear unused Register.
DEPOSIT R3 0                        ! Clear unused Register.
DEPOSIT R4 0                        ! Clear unused Register.
DEPOSIT R5 0                        ! Clear unused Register.
DEPOSIT FP 0                        ! No Machine Check expected.
START 20003004                      ! Start RESTART REFEREE.
```

DSC or BACKUP Boot Command File

This command file boots either STAND-ALONE DSC or STAND-ALONE BACKUP from Floppies.

```
HALT                                ! HALT the Processor.
UNJAM                               ! UNJAM the SBI.
INIT                                ! INITialize the Processor.
DEPOSIT/I 11 20003800               ! Set address of SCBB.
DEPOSIT R0 40                       ! Console Floppy Device.
DEPOSIT R1 0
DEPOSIT R2 0
DEPOSIT R3 1                        ! Unit Number.
DEPOSIT R4 0                        ! Boot Block LBN (unused).
DEPOSIT R5 0                        ! Software Boot Flags.
DEPOSIT FP 0                        ! No Machine Check expected.
DEPOSIT SP 200                      ! Addr. of working Mem +^X200.
LOAD VMB.EXE/START:200              ! Load Primary Bootstrap
START 200                           ! and Start it.
```

RESTAR.ILV

This command file should replace the RESTAR.CMD command file for those systems that have two interleaved memory controllers. This command file assumes that the memory controllers are at TR levels 1 and 2.

This command file is invoked in the event of Power Recovery and other Console detected restart conditions if the Auto Restart switch is set. It can also be invoked with the following command entered to the CONSOL.SYS pompt (">>>"):

@RESTAR.CMD

The RESTAR.ILV command file consists of the following CONSOL.SYS commands:

```
HALT                        ! Halt VAX Processor.
INIT                        ! Initialize the VAX CPU.
DEPOSIT/I 11 20003800       ! Set Address of SCBB in ISP rom.
DEPOSIT R0 0                ! Clear unused Register.
DEPOSIT R1 xxx              ! xxx = TR of Boot Disk NEXUS.
DEPOSIT R2 0                ! Clear unused Register.
DEPOSIT R3 0                ! Clear unused Register.
DEPOSIT R4 0                ! Clear unused Register.
DEPOSIT R5 0                ! Clear unused Register.
DEPOSIT FP 0                ! No MACHINE CHECK expected.
DEPOSIT 20002000 101        ! Enable TR#1 Memory's interleaving.
DEPOSIT 20002004 4000       ! Force starting address to 00000000.
DEPOSIT 20004000 101        ! Enable TR#2 Memory's interleaving.
DEPOSIT 20004004 4000       ! Force starting address to 00000000.
START 20003004              ! Start Restart Referee in ISP rom.
```

RMEM.

This command file is used to reset the starting addresses of the memories, in a MS780 and MA780 system configuration, so that the MS780 will again be low memory. This is necessary in order to run diagnostics on a VAX-11/782 system.

This command file is located on the LOCAL CONSOLE FLOPPY and is executed by typing "@RMEM" to the CONSOL.SYS prompt (">>>").

```
!
! Command file to RESET Memory Controller Restart Addresses.
!
DEPOSIT 20002004 00004000       ! SET TR=1 MEMORY TO START AT 0.0MB.
DEPOSIT 2000400C 00200001       ! SET TR=2 MEMORY OUT OF THE WAY.
DEPOSIT 2000600C 00A00001       ! SET TR=3 MEMORY OUT OF THE WAY.
DEPOSIT 2000800C 01200001       ! SET TR=4 MEMORY OUT OF THE WAY.
DEPOSIT 2000A00C 01A00001       ! SET TR=5 MEMORY OUT OF THE WAY.
```

SECTION III

Special COMMAND files / Programs

Hardware Dump File Maintenance/Generation

Two files should be added and two files modified on the LOCAL CONSOLE floppy in order to take Hardware Register Dumps. "DUMP." and "HANG." should be generated and installed on the Local Console Floppy by you. You should also modify the existing DEFBOO.CMD & RESTAR.CMD command files so that a Hardware Dump is taken before the system is rebooted (whenever BUGREBOOT=1). DEFBOO.CMD is used when BUGREBOOT=1 and AUTO RESTART switch is "OFF". RESTAR.CMD is used when BUGREBOOT=1 and AUTO RESTART switch is "ON".

The "REMOTE LOCAL CONSOLE" floppy should also be modified to contain the files mentioned above.

DUMP.

This command file dumps all the Hardware Registers via the CONSOL.SYS program commands. It should be tailored to the system on which it will be used so that all the Hardware Registers will be dumped.

This command file is generated and then placed on the Local Console Floppy. The customer should be educated, by you, as to WHEN and HOW this command file should be used.

HANG.

This command file should do three things. They are as follows:

1. Dumps all Hardware Registers as defined by DUMP.
2. Single Steps the VAX CPU so as to find out where the software is hung.
3. Initiates an @CRASH. so as to obtain a Software Dump.

This command file is to be used for dumping system software hangs. It is your responsibility to educate the customer as to HOW and WHEN to use this command file.

DEFBOO.CMD and RESTAR.CMD

These command files are supplied on the distributed Local Console Floppy. Their purpose is to attempt a restart of the Operating System on certain failure conditions. UNFORTUNATELY, the DEFBOO.CMD command file is also used whenever a "B" or "BOOT" is entered to CONSOL.SYS in order to reboot the system. The information gathered by appending a set of hardware register dump commands to the beginning of these files is very useful when trouble shooting system crashes on a system that is set up to reboot automatically. It is best to get an O.K. from the customer before modifying these command files. If need be, a seperate LOCAL CONSOLE floppy (preferably a "REMOTE LOCAL CONSOLE" floppy) can be made to use when the system has problems.

Version 3.x VMS dump file generation

The following commands can be used to create the HANG., DUMP., and modify the DEFBOO.CMD & RESTAR.CMD command files on the LOCAL CONSOLE floppy of a VERSION 3.x VMS Operating System.

First of all, you must log into an account that has the "SETPRV" privilege or into an account that has enough privileges to access "CSA1:". Do the following steps on "VERSION 3.x VMS" systems:

```
$ SET PROCESS/PRIV=ALL
$ MCR SYSGEN
CONNECT CONSOLE
EXIT
$ MOUNT/FOR CSA1:
$ MCR FLX /RS=CS1:DEFBOO.CMD/RT/FA
$ MCR FLX /RS=CS1:RESTAR.CMD/RT/FA
$ RENAME RESTAR.CMD RESTAR.OLD
$ RENAME DEFBOO.CMD DEFBOO.OLD
$ EDIT DUMP.

        ; Create the DUMP. command file that you wish
        ; to place on the LOCAL CONSOLE floppy.  Be sure
        ; to have examines for all registers on the system.
        ; An example of a DUMP. command file is given in this
        ; manual.

$ COPY DUMP. HANG.
$ EDIT HANG.
        ; Now create the HANG. command file by modifying the
        ; just made DUMP. command file to include the commands
        ; shown below.  These commands should be appended to
        ; the end of the file.

        !
        ! Set single step mode and gather some PC's
        ! in order to determine where software is hung.
        !
   SET DEFAULT HEX,LONG,PHYSICAL
   D/ID 0A 00008080              ! Turn off/Clear Interval Timer.
   SET STEP INSTRUCTION          ! Set single step mode.
   NEXT 30                       ! Find program loop.
   CLEAR STEP                    ! Disable single step mode.
   NEXT 1                        ! Continue clock.
   D/ID 0A 00000040              ! Re-enable Interval Timer.
        !
        ! Now execute the equivalent of @CRASH in order
        ! to cause a Software Dump to be taken.
        !
   HALT                          ! Halt the system.
   E PC                          ! Get current PC.
   E PSL                         ! Get contents of PSL.
   E/I/N:4 0                     ! Get Stack pointers.
   D PC -1                       ! Invalidate PC.
   D PSL 1F0000                  ! Set Kernel mode, IPL 31.
   CONTINUE                      ! Continue macro program.
```

```
$ COPY DUMP. DEFBOO.CMD
$ COPY DUMP. RESTAR.CMD
$ APPEND DEFBOO.OLD DEFBOO.CMD
$ APPEND RESTAR.OLD RESTAR.CMD
$ COPY/CONTIG RESTAR.CMD RESTAR.CMD
$ COPY/CONTIG DEFBOO.CMD DEFBOO.CMD
$ COPY/CONTIG DUMP. DUMP.
$ COPY/CONTIG HANG. HANG.
$ PURGE DEFBOO.CMD
$ PURGE RESTAR.CMD
$ PURGE DUMP.
$ PURGE HANG.
$ MCR FLX CS1:/RT=DEFBOO.CMD/RS/FA
$ MCR FLX CS1:/RT=RESTAR.CMD/RS/FA
$ MCR FLX CS1:/RT=DUMP./RS/FA
$ MCR FLX CS1:/RT=HANG./RS/FA
$ MCR FLX CS1:/RT/LI
$ DISMOUNT CSA1:
```

Now you should bring down the system and test the files you have just created. To test the command files, proceed as follows from the CONSOL.SYS prompt:

```
>>>@DUMP

        ; All registers should be examined.  Several errors may
        ; occur on the Memory Stack examines. Ignore them.

>>>@HANG

        ; All registers should be examined.  Several errors may
        ; occur on the Memory Stack examines. Ignore them.

>>>B

        ; A hardware register dump should occur as with the DUMP.
        ; command file and then the system should reboot.
```

Now place the AUTO-RESTART switch to "ON" and turn the power "OFF" and back "ON". The Operating system should reboot via the modified RESTAR.CMD command file.

Version 4.x VMS dump file generation

The following commands can be used to create the HANG., DUMP., and modify the DEFBOO.CMD & RESTAR.CMD command files on the LOCAL CONSOLE floppy of a VERSION 4.x VMS Operating System.

First of all, you must log into an account that has the "SETPRV" privilege or into an account that has enough privileges to access "CSA1:". Do the following steps on "VERSION 4.x VMS" systems:

```
$ SET PROCESS/PRIV=ALL
$ MCR SYSGEN
CONNECT CONSOLE
EXIT
$ EXCHANGE
COPY CSA1:RESTAR.CMD RESTAR.OLD
COPY CSA1:DEFBOO.CMD DEFBOO.OLD
EXIT
$ EDIT DUMP.

        ; Create the DUMP. command file that you wish
        ; to place on the LOCAL CONSOLE floppy.  Be sure
        ; to have examines for all registers on the system.
        ; An example of a DUMP. command file is given in this
        ; manual.

$ COPY DUMP. HANG.
$ EDIT HANG.
        ; Now create the HANG. command file by modifying the
        ; just made DUMP. command file to include the commands
        ; shown below.  These commands should be appended to
        ; the end of the file.

        !
        ! Set single step mode and gather some PC's
        ! in order to determine where software is hung.
        !
  SET DEFAULT HEX,LONG,PHYSICAL
  D/ID 0A 00008080              ! Turn off/Clear Interval Timer.
  SET STEP INSTRUCTION          ! Set single step mode.
  NEXT 30                       ! Find program loop.
  CLEAR STEP                    ! Disable single step mode.
  NEXT 1                        ! Continue clock.
  D/ID 0A 00000040              ! Re-enable Interval Timer.
       !
       ! Now execute the equivalent of @CRASH in order
       ! to cause a Software Dump to be taken.
       !
  HALT                          ! Halt the system.
  E PC                          ! Get current PC.
  E PSL                         ! Get contents of PSL.
  E/I/N:4 0                     ! Get Stack pointers.
  D PC -1                       ! Invalidate PC.
  D PSL 1F0000                  ! Set Kernel mode, IPL 31.
  CONTINUE                      ! Continue macro program.
```

```
$ COPY DUMP. DEFBOO.CMD
$ COPY DUMP. RESTAR.CMD
$ APPEND DEFBOO.OLD DEFBOO.CMD
$ APPEND RESTAR.OLD RESTAR.CMD
$ COPY/CONTIG RESTAR.CMD RESTAR.CMD
$ COPY/CONTIG DEFBOO.CMD DEFBOO.CMD
$ COPY/CONTIG DUMP. DUMP.
$ COPY/CONTIG HANG. HANG.
$ PURGE DEFBOO.CMD
$ PURGE RESTAR.CMD
$ PURGE DUMP.
$ PURGE HANG.
$ EXCHANGE
COPY DEFBOO.CMD CSA1:DEFBOO.CMD
COPY RESTAR.CMD CSA1:RESTAR.CMD
COPY DUMP. CSA1:DUMP.
COPY HANG. CSA1:HANG.
EXIT
$ DISMOUNT CSA1:
```

Now you should bring down the system and test the files you have just created. To test the command files, proceed as follows from the CONSOL.SYS prompt:

```
>>>@DUMP

        ; All registers should be examined.  Several errors may
        ; occur on the Memory Stack examines. Ignore them.

>>>@HANG

        ; All registers should be examined.  Several errors may
        ; occur on the Memory Stack examines. Ignore them.

>>>B

        ; A hardware register dump should occur as with the DUMP.
        ; command file and then the system should reboot.
```

Now place the AUTO-RESTART switch to "ON" and turn the power "OFF" and back "ON". The Operating system should reboot via the modified RESTAR.CMD command file.

DUMP. *Command File*

This is an example of how a Hardware Register Dump command file should look.
This command file should be tailored for the system it is to be used on.
----> Lines marked with an "*" are system configuration dependent. <----

```
---------------------------------------------------------------------------
        !     H A R D W A R E    R E G I S T E R    D U M P
        ! Date & Time :
        ! Customer Name:
        ! VAX Serial No.:
        ! Don't "^C", even if there is "?MIC" and/or "?MEM-MAN" errors.
        SHOW                    ! Check to see if VAX is running.
        HALT                    ! Make sure that the VAX is halted.
        SET RELOCATION:0
        SET DEFAULT HEX,LONG,PHYSICAL
        E/ID/N:17 0             ! VAX CPU ID Registers.
        E/ID 18                 ! "15" cycles prior to  "SBI FAULT".
        E/ID *                  ! "14" cycles prior to  "SBI FAULT".
        E/ID *                  ! "13" cycles prior to  "SBI FAULT".
        E/ID *                  ! "12" cycles prior to  "SBI FAULT".
        E/ID *                  ! "11" cycles prior to  "SBI FAULT".
        E/ID *                  ! "10" cycles prior to  "SBI FAULT".
        E/ID *                  ! "09" cycles prior to  "SBI FAULT".
        E/ID *                  ! "08" cycles prior to  "SBI FAULT".
        E/ID *                  ! "07" cycles prior to  "SBI FAULT".
        E/ID *                  ! "06" cycles prior to  "SBI FAULT".
        E/ID *                  ! "05" cycles prior to  "SBI FAULT".
        E/ID *                  ! "04" cycles prior to  "SBI FAULT".
        E/ID *                  ! "03" cycles prior to  "SBI FAULT".
        E/ID *                  ! "02" cycles prior to  "SBI FAULT".
        E/ID *                  ! "01" cycle prior to  "SBI FAULT".
        E/ID *                  ! Last cycle stored prior to latching SILO.
        E/ID/N:25 19            ! Remaining CPU ID Registers.
        E IR                    ! Examine the contents of the IR.
        E PC                    ! Get current PC.
        E/L/V -                 ! Get some instruction stream data.
        E/L/V -
        E/L/V -
        E/I 0/N:4               ! Examine STACK "Internal Regs."
*       E/N:2 20002000          ! MEMORY Registers (TR=1).
*       E/N:7 20006000          ! DW780 Registers (TR=3).
*       E/N:6 20010000          ! RH780 (TR=8) Registers.
*       E/N:6 20012000          ! RH780 (TR=9) Registers.
*       E/N:F 20010400          ! DBA0: - RP06 #0 on RH780 at TR#8.
*       E/N:F 20010480          ! DRA1: - RM05 #1 on RH780 at TR#8.
*       E/N:9 20012400          ! MTA0: - TE16 #0 on RH780 at TR#9.
*       E/W/N:3 2013E038        ! XMA - (760070) - on Adapter #0.
*       E/W/N:3 2013E040        ! XMB - (760100) - on Adapter #0.
*       E/W/N:3 2013E050        ! TTA - (760120) - on Adapter #0.
*       E/W/N:3 2013E058        ! TTB - (760130) - on Adapter #0.
        E/ID 3B                 ! Get System Control Block Base address.
        E/L/P/N:7F @            ! Dump System Control Block contents.
        E/G/N:F 0               ! General Registers.
        E SP                    ! Get Stack Pointer.
        E/V/N:60 @              ! Contents of STACK.   IGNORE examine errors.
        SHOW VERSION
```

HANG. *Command File*

This command file should be tailored to the System it will be used on. The first part of this command file should almost identical to the DUMP. file. Some commands are added to the end of the DUMP. file so that some information can be gathered concerning the hang.
----> Lines marked with an "*" are system configuration dependent. <----

```
---------------------------------------------------------------------------
        !      SYSTEM HANG - Hardware & Software Dump file
        ! Date & Time :
        ! Customer Name:
        ! VAX Serial No.:
        ! Don't "^C", even if there is "?MIC" and/or "?MEM-MAN" errors.
        SHOW                    ! Check to see if VAX is running.
        HALT                    ! Make sure that the VAX is halted.
        SET RELOCATION:0
        SET DEFAULT HEX,LONG,PHYSICAL
        E/ID 0/N:3E             ! VAX CPU ID Registers.
        E IR                    ! Get the contents of the IR.
        E/I 0/N:4               ! Stacks via Internal Registers.
*       E/N:2 20002000          ! MEMORY Registers (TR=1)
*       E/N:7 20006000          ! DW780 Registers (TR=3)
*       E/N:6 20010000          ! RH780 (TR=8) Registers
*       E/N:6 20012000          ! RH780 (TR=9) Registers
*       E/N:F 20010400          ! DBA0: - RP06 #0 on RH780 at TR#8.
*       E/N:F 20010480          ! DRA1: - RM05 #1 on RH780 at TR#8.
*       E/N:9 20012400          ! MTA0: - TE16 #0 on RH780 at TR#9.
*       E/W/N:3 2013E038        ! XMA - (760070) - on Adapter #0.
*       E/W/N:3 2013E040        ! XMB - (760100) - on Adapter #0.
*       E/W/N:3 2013E050        ! TTA - (760120) - on Adapter #0.
*       E/W/N:3 2013E058        ! TTB - (760130) - on Adapter #0.
        E/G/N:F 0               ! General Registers
        E SP                    ! Get Stack Pointer.
        E/V/N:60 @              ! Contents of STACK.  IGNORE examine errors.
        E PC                    ! Get some instruction stream data in case
        E/L/V -                 ! hung in a very tight, one or two instruction
        E/L/V -                 ! loop.
        E/L/V -
        SHOW VERSION
        ! .... Single Step the system to gather some program loop PC's ....
        SET DEFAULT HEX,LONG,PHYSICAL
        D/ID 0A 00008080        ! Turn off/Clear Interval Timer.
        SET STEP INSTRUCTION    ! Set single step mode.
        NEXT 60                 ! Show program loop.
        CLEAR STEP              ! Disable single step mode.
        NEXT 1                  ! Continue clock.
        D/ID 0A 00000041        ! Re-enable Interval Timer.
        !  ....  Simulate an "@CRASH" so as to get Software Dump ....
        HALT
        E PC
        E PSL
        D PC -1                 ! Invalidate PC
        D PSL 1F0000            ! KERNEL mode / IPL=31
        CONTINUE
```

S A V E D U M P . C O M

A command file that saves SYS$SYSTEM:SYSDUMP.DMP in a specified area. This command file should be executed from the SYS$SYSROOT:[SYSMGR]SYSTARTUP.COM command file so that the Software Dump is saved.

```
$       SAVE_VERIFY = 'F$VERIFY("NO")'
$ !
$ !     Written by Roy D. Fulton,  D.E.C. Field Service
$ !
$ ! This command file copies the Software System Dump File,
$ ! SYS$SYSTEM:SYSDUMP.DMP, to a file located in 'AREA_NAME'
$ ! with a name that reflects the reboot DAY, MONTH, HOUR and MINUTE.
$ !
$ ! This command file should be entered by specifing the "AREA_NAME"
$ ! as parameter "P1", as follows:
$ !
$ !     @yyySAVEDUMP xxx         ;where xxx = DDCU:DIRECTORY of area
$ !                               in which to save the dump file and,
$ !
$ !                              ;where yyy = DDCU:DIRECTORY of where
$ !                               the "SAVEDUMP.COM" file is located.
$ !
$ !
$ ! If the "AREA_NAME" is not specified as parameter "P1", the dump
$ ! file will be copied to the area "SYS$LOGIN" (the default directory
$ ! of the area that you logged into).
$ !
$ ! This command file should be executed as part of the VMS System
$ ! "SYS$SYSROOT:[SYSMGR]SYSTARTUP.COM" command file by placing the
$ ! above command in that command file.
$ !
$       IF P1 .NES. "" THEN AREA_NAME := 'P1'
$       IF P1 .EQS. "" THEN AREA_NAME := 'F$LOGICAL("SYS$LOGIN")'
$       DTIM := 'F$TIME()'
$       LT = 'F$LOCATE("-",DTIM)'
$       DAY := 'F$EXTRACT(0,LT,DTIM)'
$       IF LT .EQ. 1 THEN DAY := 0'DAY'
$       LT = 'LT'+1
$       MONTH := 'F$EXTRACT(LT,3,DTIM)'
$       LT = 'F$LOCATE(":",DTIM)'+1
$       MIN := 'F$EXTRACT(LT,2,DTIM)'
$       LT = 'LT'-3
$       HOUR := 'F$EXTRACT(LT,2,DTIM)'
$       NEW_NAME := 'DAY''MONTH''HOUR''MIN'.DMP
$       NEW_LIST := 'DAY''MONTH''HOUR''MIN'.LIS
$       SEXE := 'F$LOGICAL("SYS$SYSTEM")'
$       WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT ".... Copying"
$ WRITE SYS$OUTPUT "             from :     ''SEXE'SYSDUMP.DMP"
$ WRITE SYS$OUTPUT "               to :     ''AREA_NAME''NEW_NAME'     "
$       ANALYZE/CRASH_DUMP SYS$SYSTEM:
COPY 'AREA_NAME''NEW_NAME'
SET OUTPUT 'AREA_NAME''NEW_LIST'
SHOW CRASH
SHOW STACK
SHOW SUMMARY
SHOW PROCESS/PCB/PHD/REG
SHOW SYMBOL/ALL
EXIT
$       IF SAVE_VERIFY THEN SET VERIFY
$       EXIT
```

SPEAR Batch Command File

S P E A R B A T C H C O N T R O L

This is an example of a VMS Indirect Command file that can be used to run SPEAR as a BATCH job. When this command file is executed, a BATCH job will be submitted that will run the "ANALYZE" portion of SPEAR and will queue the output to the printer. The command file is executed by typing "@DDCU:SPEAR" at the VMS prompt, where "DDCU" is equal to the directory designation of where the SPEAR.COM file resides. The actual command file would appear as follows for VMS:

```
$ ! VMS COMMAND FILE TO RUN SPEAR
$ !
$ ! FIRST RE-QUEUE THE JOB TO RUN TOMORROW
$ SUBMIT SPEAR.COM/AFTER:TOMORROW/WSDEFAULT=400/WSQUOTA=0
$ !
$ RUN SYS$SYSTEM:SPEAR
SUMMARIZE
 !FILE NAME
 !FROM
 !TO
SUMMAR.RPT
 !GO
ANALYZE
 !FILE NAME
 !FROM
 !TO
SPEAR.RPT
NL:
 !GO
EXIT
$ !
$ ! PRINT THE RESULT
$ PRINT SUMMAR.RPT
$ PURGE SUMMAR.RPT/KEEP=1
$ PRINT SPEAR.RPT
$ PURGE SPEAR.RPT/KEEP=1
```

S D A . C O M

This command file is used to make a printable file containing some information from a specified system software dump file.

```
$       SAVE_VERIFY = 'F$VERIFY("NO")'
$ !
$ !     Written by Roy D. Fulton,  D.E.C. Field Service
$ !
$       VERSION := 2.0
$       SET NOON
$       ON ERROR THEN CONTINUE
$       ON CONTROL_Y THEN EXIT
$       TYPE SYS$INPUT
-------------------------------------------------------------------------

This command file examines the desired SOFTWARE Dump file, and
creates a file of basic information (about the crash) that can
be used for most crash analysis.

Answer all questions with a "Y" for YES  or an "N" for NO, unless
otherwise stated.  A "<return>" is equal to a YES.

If you require help, simply type a "?" or "H".

-------------------------------------------------------------------------


$ START:
$ INQUIRE DMP "What System Dump  ( DDCU:[DIRECTORY]FILENAME.EXT )?     "
$       IF DMP .NES. "/H" .AND. DMP .NES. "H" .AND. DMP .NES. "/HELP" -
        .AND. DMP .NES. "HELP" .AND. DMP .NES. "?" THEN GOTO CONTIN0
$       TYPE SYS$INPUT


Enter the name of the file that you wish to examine and also
state the device and directory of where the file is located, in
the following format:


DDCU:[DIRECTORY]FILENAME.EXT

        where,
                DDCU            = device on which file is stored.
                DIRECTORY       = directory in which dump is located.
                FILENAME        = filename of the dump file.
                EXT             = three digit extension of dump file.

-------------------------------------------------------------------------


$ !
$       GOTO START
```

```
$ CONTIN0:
$       IF DMP .EQS. "" THEN DMP :== SYS$SYSTEM:SYSDUMP.DMP
$       INQUIRE T0 "... You wish to examine ''DMP' ? "
$       IF T0 .NES. "Y" .AND. T0 .NES. "" .AND. T0 .NES. "YES" THEN -
        GOTO START
$ PRTOUT:
$       INQUIRE PRT "... Output to be printed on the  SYS$PRINT  device?"
$       IF PRT .EQS. "Y" .OR. PRT .EQS. "" .OR. PRT .EQS. "YES" THEN PRT := Y
$       IF PRT .EQS. "N" .OR. PRT .EQS. "NO"  THEN PRT := N
$       IF PRT .EQS. "N" .OR. PRT .EQS. "Y" THEN GOTO CONTIN1
$       TYPE SYS$INPUT
```

The output will go to a file in your disk area that is equal to the Dump's filename with an extension equal to "LST". This file can be queued to the SYS$PRINT queue, which is usually a Line Printer queue, if you so desire. Answer this question with one of the following responses:

Y	=	Queue listing to SYS$PRINT device. File is deleted after printing.
<return>	=	same as "Y".
N	=	Do not print the file. File remains in disk area.
?	=	Displays this help file.
H	=	same as "?"

```
$ WRITE SYS$OUTPUT "    The SYS$PRINT device is  ''F$LOGICAL("SYS$PRINT")'  ."
$       WRITE SYS$OUTPUT " "
$       GOTO PRTOUT
$ CONTIN1:
$       TMP := 'DMP'
$       LT = 'F$LENGTH(DMP)'
$       AA = 'F$LOCATE(":",DMP)'
$       BB = 'LT'-'AA'
$       IF AA .NE. LT THEN TMP := 'F$EXTRACT(AA+1,BB-1,DMP)'
$       LT = 'F$LENGTH(TMP)'
$       AA = 'F$LOCATE("]",TMP)'
$       BB = 'LT'-'AA'
$       IF AA .NE. LT THEN TMP := 'F$EXTRACT(AA+1,BB-1,TMP)'
$       LT = 'F$LENGTH(TMP)'
```

```
$       AA = 'F$LOCATE(".",TMP)'
$       IF AA .NE. LT THEN TMP := 'F$EXTRACT(0,AA,TMP)'
$       TMP := 'F$LOGICAL("SYS$LOGIN")''TMP'.LST
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT ".... Creating     ''TMP'    ... Please wait ...  "
$ ANALYZ:
$       ON ERROR THEN EXIT
$       RUN SYS$SYSTEM:SDA
'DMP'
SET OUTPUT 'TMP'
SHOW CRASH
SHOW PROCESS/ALL
SHOW STACK/ALL
SHOW DEVICE
SHOW SUMMARY
SHOW PFN_DATA/ALL
EXAMINE/P0
EXIT
$ !
$       ON ERROR THEN CONTINUE
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       IF PRT .EQS. "Y" THEN WRITE -
SYS$OUTPUT " .....''TMP'  is being queued to ''F$LOGICAL("SYS$PRINT")'"
$       IF PRT .EQS. "Y" THEN PRINT/DELETE 'TMP'
$       IF PRT .EQS. "Y" THEN GOTO EXITH
$       WRITE SYS$OUTPUT "..... Information about  -  ''DMP'"
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT "..... is stored in       -  ''TMP'"
$ EXITH:
$       WRITE SYS$OUTPUT " "
$       WRITE SYS$OUTPUT " "
$       IF SAVE_VERIFY THEN SET VERIFY
```

FP780 Control Programs

These programs are used to turn the Floating Point Accelerator "ON" or "OFF". These two programs may be created with an editor and then assembled and linked.

FPAOFF.MAR

```
.TITLE FPAOFF.MAR

        ;Written by Roy D. Fulton, D.E.C. Field Service
        ;
        ;This routine "Turns OFF" the Floating Point Accelerator
        ;
        ;This routine needs the "CMKRNL" privilege.
        ;

START:  .WORD
        $CMKRNL_S FPAOFF
        RET

FPAOFF: .WORD

        MFPR #^X3E,SID          ;get SID Register contents

        CMPZV #^D24,#^D4,SID,#^X1       ;is System a VAX-11/780
        BEQL VAX780                     ;Branch if yes
        CMPZV #^D24,#^D4,SID,#^X2       ;is System a VAX-11/750
        BEQL VAX780                     ;Branch if yes
        CMPZV #^D24,#^D4,SID,#^X3       ;is System a VAX-11/730
        BEQL VAX780                     ;Branch if yes

        MOVL #^X901,R0          ;Unsupported Processor Type
        RET                     ;exit routine due to no such processor

SID:    .WORD                   ;holds contents of SID register
        .WORD


VAX780:
        MTPR #0,#^X28           ;Turn off FPA enable on 11/780
        MOVL #^X1,R0            ;11/780 successful completion
        RET                     ;exit routine

        .END START
```

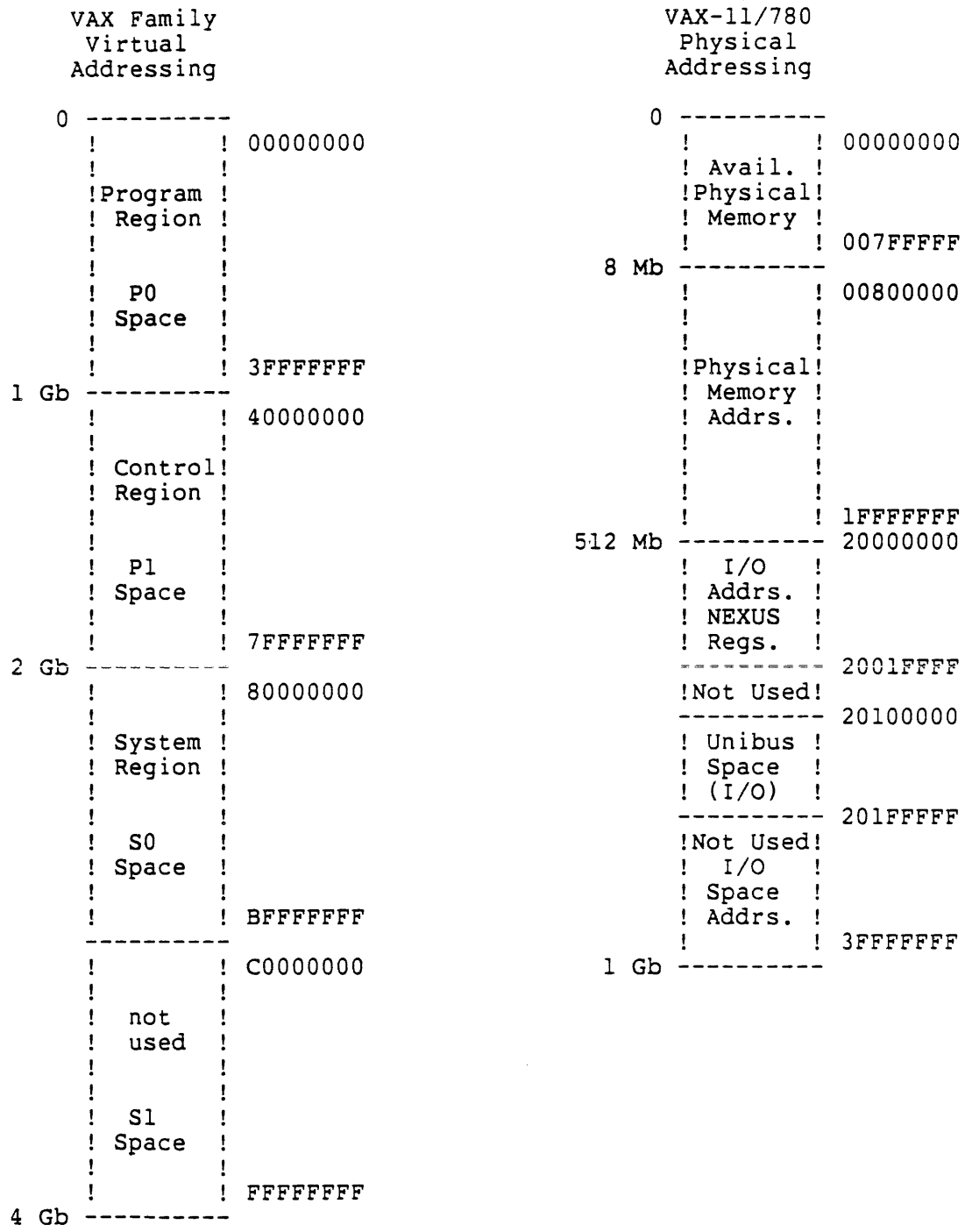
FPAON.MAR

```
.TITLE FPAON.MAR

        ;Written by Roy D. Fulton, D.E.C. Field Service
        ;
        ;This routine "Turns ON" the Floating Point Accelerator
        ;
        ;This routine needs the "CMKRNL" privilege.
        ;

START:  .WORD
        $CMKRNL_S FPAOFF
        RET

FPAOFF: .WORD

        MFPR #^X3E,SID          ;get System Identification Register contents

        CMPZV #^D24,#^D4,SID,#^X1       ;is System a VAX-11/780
        BEQL VAX780                     ;Branch if yes
        CMPZV #^D24,#^D4,SID,#^X2       ;is System a VAX-11/750
        BEQL VAX780                     ;Branch if yes
        CMPZV #^D24,#^D4,SID,#^X3       ;is System a VAX-11/730
        BEQL VAX780                     ;Branch if yes

        MOVL #^X901,R0          ;Unsupported Processor Type
        RET                     ;exit routine due to no such processor

SID:    .WORD                   ;holds contents of SID register
        .WORD

VAX780:
        MTPR #8000,#^D40        ;Turn ON FPA enable on 11/780
        MOVL #^X1,R0            ;setup R0 for successful completion flag
        RET                     ;exit routine with FPA off

        .END START
```

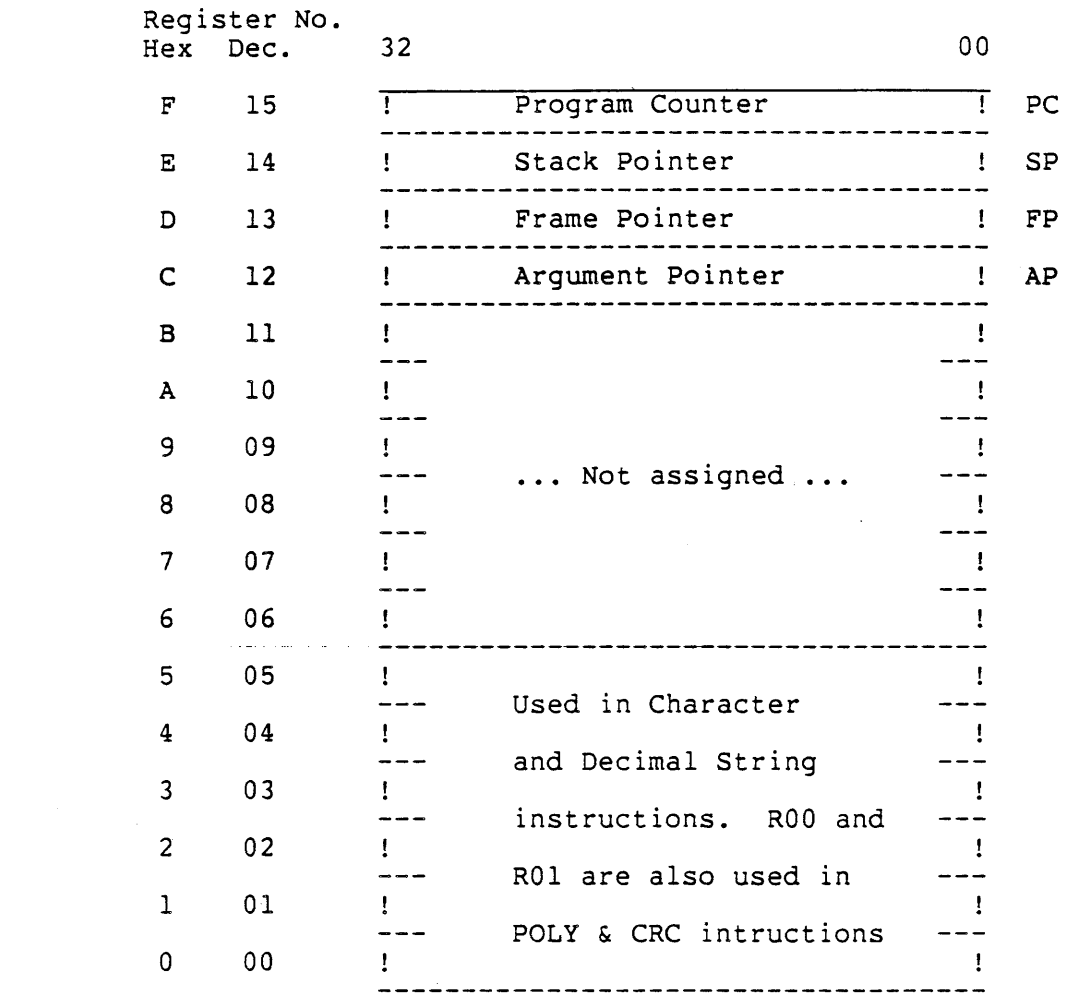
SECTION IV

VAX-11/780 BASICS

VAX Virtual & Physical Address Space



VAX-11/780 General Registers Assignments



R3,R5 - Address Counter in Character and Decimal instructions.
R2,R4 - Length Counter in Character and Decimal instructions.
R1 - Result of POLYD. Address Counter in Character and Decimal instructions.
R0 - Result of POLY,CRC. Length counter in Character and Decimal instructions.

SUBROUTINE Usage & Operation

Subroutines are portions of code that may be used many times within a program at different times. In order to save memory space, this common code can be written as a subroutine and can be called and exited with the instructions listed below. The processor saves the current PC on the STACK whenever a subroutine is called so that the return instruction will know where to return to. The Processor Status Longword is not saved on the STACK when calling a subroutine, nor is it modified by the Subroutine call and return instructions.

Three instructions are used for calling a subroutine. They are as follows:

BSBB --- Branch to subroutine with Byte Displacement.
Displaces PC a maximum of +127 or -128 bytes.

BSBW --- Branch to Subroutine with Word Displacement.
Displaces PC a maximum of +32767 or -32768 bytes.

The PC is pushed onto the STACK as a longword. The sign-extended (to 32 bits) branch displacement is added to the PC and the PC is replaced with the result.

```
-(SP) <--- PC
PC <--- PC + SEXT Displacement
```

JSB --- Jump to subroutine.

The destination address is calculated from the Operand Specifier byte. The PC is then pushed onto the STACK. Finally, the PC is replaced by the calculated destination address.

```
-(SP) <--- PC
PC <--- Destination
```

One instruction is used to return from a subroutine. It is:

RSB --- Return from subroutine

Is used to return from subroutines called by the BSBB, BSBW and JSB instructions. The PC is replaced by a longword popped from the STACK.

```
PC <--- (SP)+
```

Note: RSB is equivalent to JMP @(SP)+, but is one byte shorter.

PROCEDURE Usage & Operation

PROCEDUREs are general purpose routines that use argument lists passed automatically by the processor and use only local variables for data storage. A PROCEDURE CALL INSTRUCTION provides several services to the programmer that occur automatically by the processor.

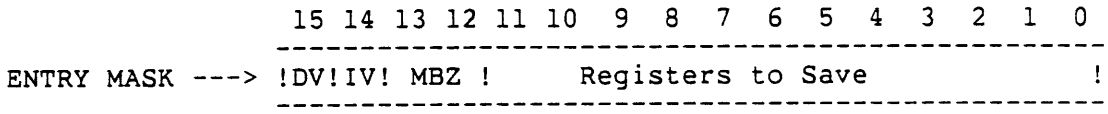
A Procedure Call Instruction:

- Saves all the registers (R00 - R11), that the procedure uses, before entering the called procedure. This is accomplished by the programmer specifying which registers are to be saved in an ENTRY MASK when the Procedure is written. The ENTRY MASK is the first WORD of a Procedure.
- Passes an argument list to a Procedure. This is done in two ways. The argument list can be stored anywhere in memory, in which case the CALLG instruction is used, or the list can be stored on the STACK, in which case the CALLS instruction is used.
- Maintains the STACK, FRAME and ARGUMENT Pointer registers.
- Initializes the Arithmetic Trap ENABLES to a given state. This is accomplished by the ENTRY MASK.

When a PROCEDURE completes execution, it issues the RET (Return from Procedure) instruction. RET uses the Frame Pointer register to find the registers that were saved by the Procedure Call Instruction. It restores the original contents to these registers, cleans up data left on the Stack (including nested routine linkages), and can return values using the argument list or other registers.

ENTRY MASK

- Is one Word in length
- Bits 2 thru 11 select Registers to be Saved upon Procedure Call. A one in the respective bit position SAVES that register before Procedure is executed.
- Bits 0 & 1 are not normally used by software to save Registers 0 & 1, respectively, due to Procedure Calling standard. They will be saved if you set the respective bit.
- Bit 15 is used to enable/disable Decimal Overflow (DV).
- Bit 14 is used to enable/disable Integer Overflow (IV).
- Bits 12 & 13 must be zero.
- Is located in First WORD of Procedure.



ARGUMENT LIST

- An Argument List is simply Data that is needed for the Procedure to use. This list of data may be something like a group of numbers that must be added together by the Procedure.
- The Argument List may be stored anywhere in memory or it may be stored on the Stack. If the Argument List is stored on the Stack, the "CALLS" Procedure call instruction is used to enter the Procedure. If the Argument List is stored somewhere other than on the Stack, the "CALLG" Procedure call instruction is used to enter the Procedure.

"CALLG" Procedure Call Operation

Format:

opcode arglist.ab, dst.ab

opcode	=	"FA"
arglist.ab	=	Specifies starting address of Argument List in memory.
dst.ab	=	Specifies starting address of the Procedure to be entered.

Description:

1. SP is saved in a temporary register and then bits 1:0 are replaced by 0 so that the stack is longword aligned.

2. The PROCEDURE ENTRY MASK is scanned from Bit 11 to 00 and the contents of those registers whose number corresponds to the set bits in the ENTRY MASK are pushed on the Stack as LONGWORDs.

3. The "PC","FP", and "AP" are then pushed on the Stack, also as LONGWORDs.

4. The CONDITION CODES are cleared in the Processor Status Longword (PSL).

5. A LONGWORD is pushed on the Stack containing:
   - the two low bits of the saved SP in Bits 31:30
   - a 0 in Bits 29 & 28
   - the low 12 bits of the ENTRY MASK in Bits 27:16
   - the low word of the PSL in Bits 15:00 with the "T" bit cleared

6. A LONGWORD = 000000 is pushed on the Stack.

7. The "FP" is replaced by the "SP".

8. The "AP" is replaced by the "arglist operand".

9. The Trap enables are set to a known state in the PSL.
   - IV and DV are setup according to bits 14 & 15 of the ENTRY MASK, respectively
   - Floating underflow bit is cleared
   - T-bit is unaffected

10. The "PC" is replaced by the sum of the destination operand plus 2, which transfers control to the called procedure at the byte beyond the ENTRY MASK.

"CALLS" Procedure Call Operation

Format:

opcode numarg.rl, dst.ab

opcode	=	"FB"
numarg.rl	=	number of arguments on stack
dst.ab	=	specifies starting address of the procedure in memory

Description:

1. The "numarg" operand is pushed on the Stack as a Longword.
   - Byte 0 contains the number of arguments
   - The High order 24 bits are used by DEC software

2. The "SP" is saved in a temporary register and bits <1:0> of the "SP" are replaced by 0 so that the stack is Longword aligned.

3. The Procedure ENTRY MASK is scanned from bit 11 to bit 00 and the contents of the registers whose number corresponds to the set bits of the Entry Mask are pushed on the Stack.

4. The "PC","FP", and "AP" are pushed on the Stack as Longwords.

5. The Condition Codes are cleared in the Processor Status Longword (PSL).

6. A LONGWORD is pushed on the Stack containing:
   - the two low bits of the saved SP in Bits 31:30
   - a 1 in Bit 29
   - a 0 in Bit 28
   - the low 12 bits of the ENTRY MASK in Bits 27:16
   - the low word of the PSL in Bits 15:00 with the "T" bit cleared

7. A LONGWORD = 000000 is pushed on the Stack.

8. The "FP" is replaced by the "SP".

9. The "AP" is set to the value of the Stack Pointer after the "numarg operand" was pushed on the Stack.

10. The Trap enables are set to a known state in the PSL.
    - IV and DV are setup according to bits 14 & 15 of the ENTRY MASK, respectively
    - Floating underflow bit is cleared
    - T-bit is unaffected

11. The "PC" is replaced by the sum of the destination operand plus 2, which transfers control to the called procedure at the byte beyond the ENTRY MASK.

"RET" Procedure Return Operation

Format:

opcode opcode = "04"

Description:

1. The "SP" is replaced by the "FP" plus 4.

2. A Longword is popped from the Stack, and stored in a temporary register, containing:
   - Stack Alignment bits in bits 31:30
   - CALLS/CALLG flag in bit 29 (1 = CALLS, 0 = CALLG)
   - Low 12 bits of the Procedure ENTRY MASK in bits 27:16
   - A saved PSW (low order word of PSL) in bits 15:00

3. The "PC", "FP", and "AP" are replaced by Longwords popped from the Stack.

4. A register restore mask is formed from bits 27:16 of the temporary register.

5. Scanning from bit 00 to bit 11 of the restore mask, the contents of the registers whose number is indicated by set bits in the restore mask, are replaced by Longwords popped from the Stack.

6. The "SP" is incremented by 31:30 of the temporary register.

7. The PSW (low order word of PSL) is replaced by bits 15:00 of the temporary register.

8. If bit 29 in the temporary register is a 1 (indicating that the procedure was called by a CALLS instruction), a Longword containing the number of arguments is popped from the Stack. Four times the unsigned value of the low byte of this Longword is added to the "SP" and the "SP" is replaced by the result.

9. At this point, the Return has been executed. Program control continues with at the current PC.

Procedure Call (CALLS/CALLG) notes:
---

1. If bits 13:12 of the ENTRY MASK are not 0, a reserved operand fault occurs.

2. On a reserved operand fault, Condition Codes are UNPREDICTABLE...

3. The procedure calling standard and the condition handling facility require the following register saving conventions:

    - R0 & R1 are always available for function return values and are therefore never saved in the Entry Mask.
    - All registers, R2 thru R11, which are modified in the called Procedure, must be preserved by setting the respective bits in the Entry Mask.

4. When using "CALLS" Procedure Call, normal use is to push the arglist onto the stack in reverse order prior to the CALLS instruction. On RETurn, the arglist is removed from the Stack automatically by the processor.

Return from Procedure(RET) notes:
---

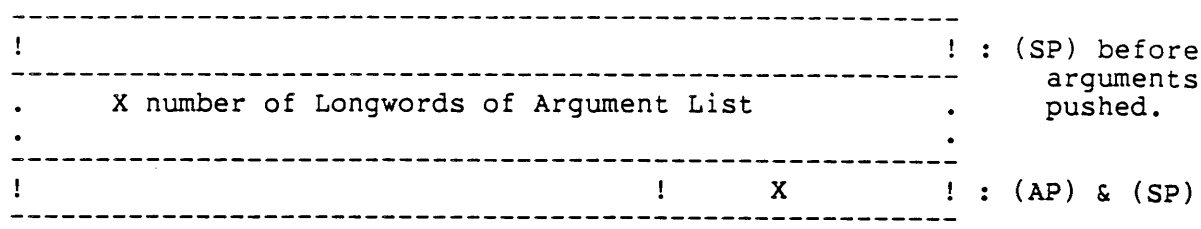
1. A reserved operand fault occurs if the Temporary Register bits 15:08 is not equal to 0.

2. On a reserved operand fault, the condition codes are UNPREDICTABLE. The value of the Temporary Register bit 28 is ignored.

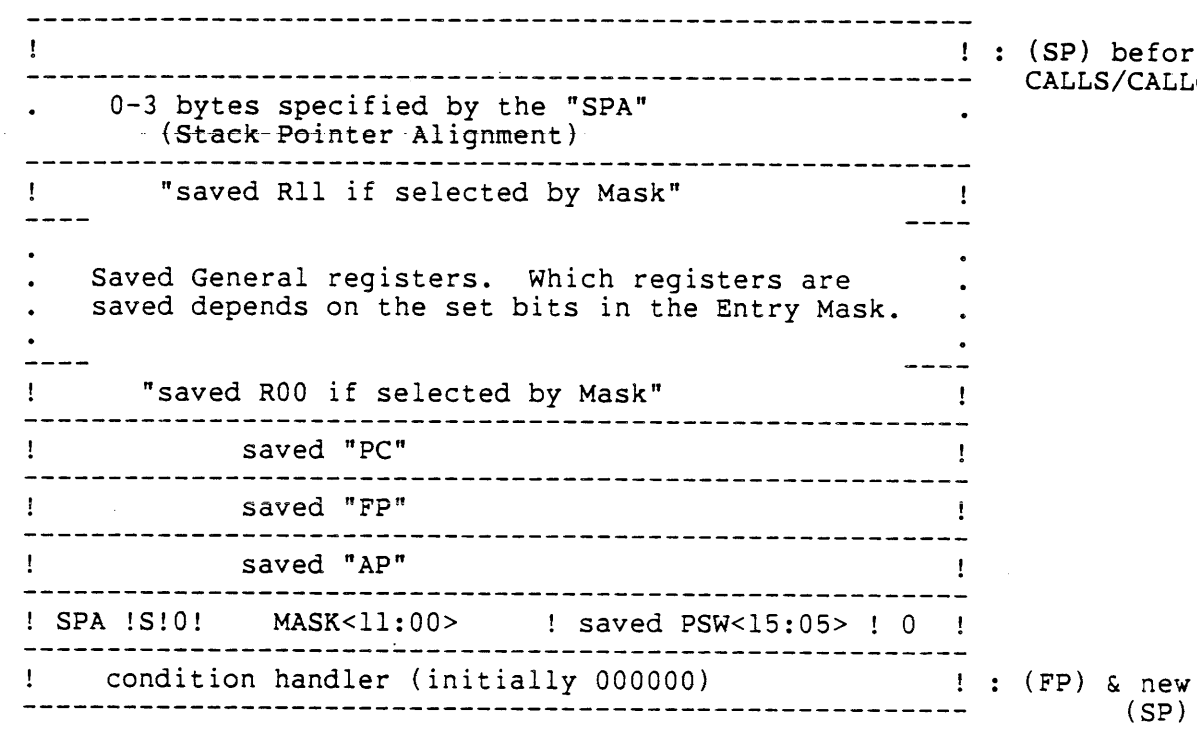
3. The procedure calling standard and the condition handling facility assume that procedures which return a function value or a status code, do so in R0 or R0 & R1.

PROCEDURE CALL Stack Layout

Before execution of a "CALLS", the Procedure Arguments are stored on the Stack as follows.

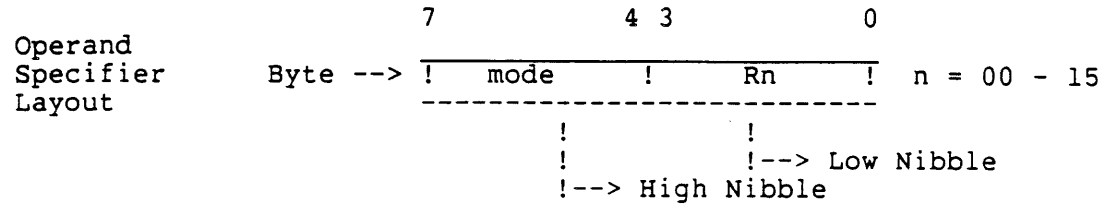


Typical stack layout as a result of a Call to a Procedure.



VAX-11/780 NATIVE ADDRESSING MODES

- Many of the modes are very similiar to PDP-11 addressing modes
- Indexing can be combined with many of the addressing modes
- Operand Specifier consists of 1 Byte that contains the MODE and the General Register to be used
- The VAX addressing modes are as follows:

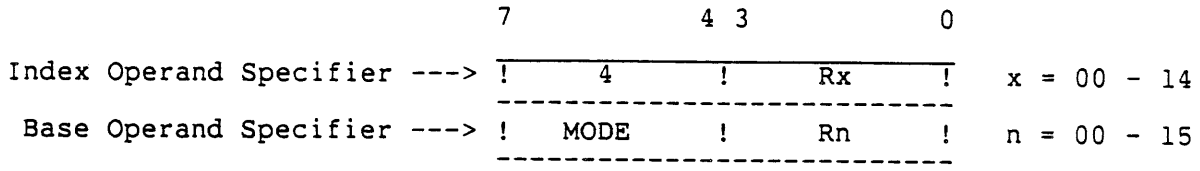


Mode	Notation	Mode Name	Description
0-3	S^#num	Literal	- Mode and Operand are contained in the same BYTE. Operand is contained in low 6 bits of addressing mode.
5	Rn	Register	- Rn contains operand.
6	(Rn)	Register Deferred	- Rn contains Address of the operand.
7	-(Rn)	Autodecrement	- Rn is first decremented. The resulting Rn contains Address of the operand.
8	(Rn)+	Autoincrement	- Rn contains the Address of the operand. Rn is incremented after use.
8	#num	Immediate (PC)	- same as autoincrement mode with R15 (PC) used as the general register.
9	@(Rn)+	Autoincrement Deferred	- Rn contains an Address that contains the Address of the operand. Rn is incremented after use.
9	@#ADDR	Absolute (PC)	- same as autoincrement deferred mode with R15 (PC) used as the General Reg.

Mode	Notation	Mode Name	Description
A - Byte C - Word E - Longword	B^d(Rn) W^d(Rn) L^d(Rn)	Displacement	- Displacement (R15 contains address of Displacement) is first sign extended if Byte or Word displacement is used. Then the displacement value is added to Rn. The resulting value is the Address of the operand.
A - Byte C - Word E - Longword	B^ADDR W^ADDR L^ADDR	Relative (PC)	- same as displacement mode with R15 (PC) used as the general register.
B - Byte D - Word F - Longword	@B^(Rn) @W^(Rn) @L^(Rn)	Displacement Deferred	- Displacement (R15 contains address of displacement) is first sign extended if Byte or Word displacement is used. Then the displacement value is added to Rn. The resulting value is the Address of the Address of the operand.
B - Byte D - Word F - Longword	@B^ADDR @W^ADDR @L^ADDR	Relative Deferred (PC)	- same as Displacement Deferred mode with R15 (PC) used as the general register.

. Indexing (Mode = 4) can be used with the following modes as long R15 is not used as the index register:

1. Register Deferred
2. Autodecrement
3. Autoincrement
4. Immediate
5. Autoincrement Deferred
6. Absolute
7. Displacement
8. Relative
9. Displacement Deferred
10. Relative Deferred



Mode	Notation
Register Deferred Indexed	(Rn)[Rx]
Autodecrement Indexed	-(Rn)[Rx]
Autoincrement Indexed	(Rn)+[Rx]
Immediate Indexed	#num[Rx]
Autoincrement Deferred Indexed	@(Rn)+[Rx]
Absolute Indexed	@#ADDR[Rx]
Displacement Indexed	disp(Rn)[Rx]
Relative Indexed	ADDR[Rx]
Displacement Deferred Indexed	@disp(Rn)[Rx]
Relative Deferred Indexed	@ADDR[RX]

. Index Specifiers are physically positioned, in memory, before the Operand Specifier (that is being indexed).

. Indexing is accomplished as follows:

1. The contents of the index register is modified by multiplying the contents of the index register by the value that reflects the context of the data type specified.

Multiply by	1 for byte
	2 for word
	4 for longword & F_floating
	8 for quadword, D & G_floating
	16 for octaword & H_floating

2. Calculate the Address of the Operand specified by the "Base" Operand Specifier.

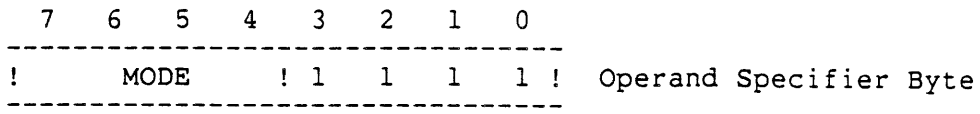
3. Add the results of steps 1 & 2 together in order to obtain the "address of the Desired Operand"...

. The following restrictions are placed on the Index register Rx:

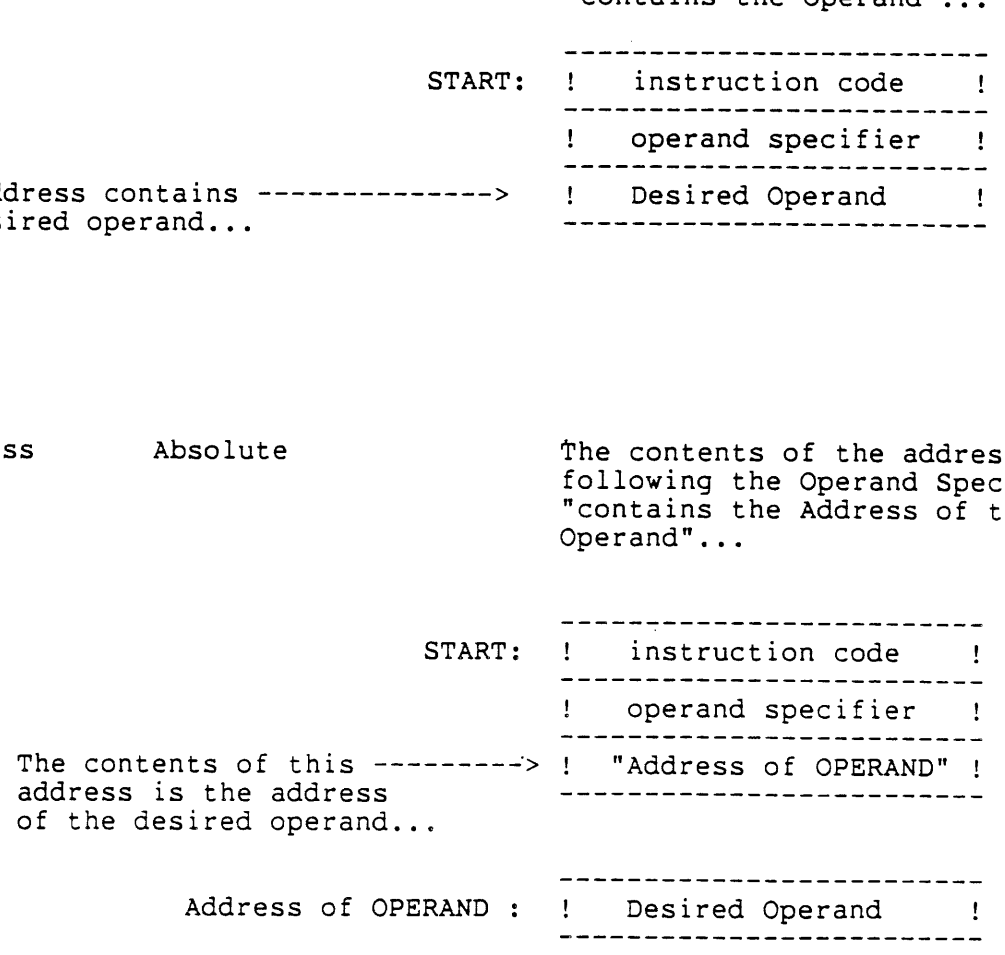
1. The PC (R15) cannot be used as an index register. If it is, a reserved addressing mode fault occurs.

2. If the Base Operand Specifier is for an addressing mode which results in register modification (auto-increment,autoincrement deferred, or autodecrement) the same register cannot be the index register. If it is, the primary operand address is "UNPREDICTABLE"...

PC (R15) Mode Operand Specifiers

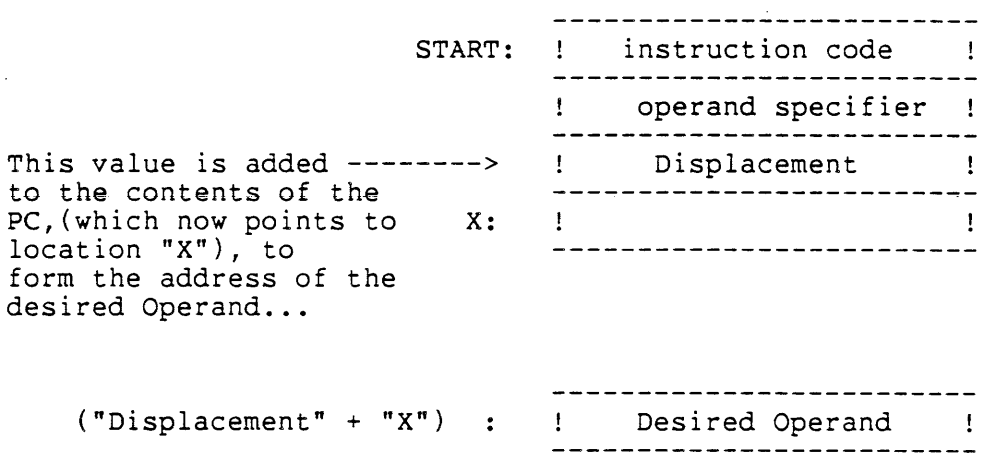


Mode	Notation	Mode Name	Description
8	#num	Immediate	The contents of the address following the Operand Specifier "contains the Operand"...
9	@#address	Absolute	The contents of the address following the Operand Specifier "contains the Address of the Operand"...



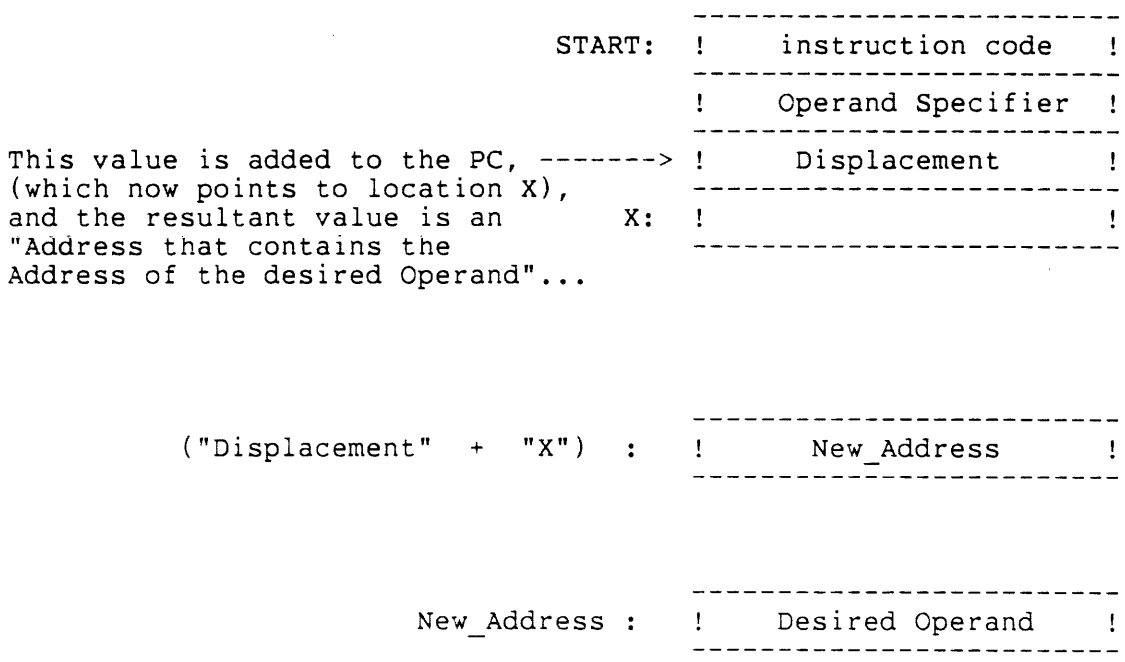
PC (R15) Mode Operand Specifiers (continued)

Mode	Notation	Mode Name	Description
A	B^address	Byte displacement	The contents of the address following the operand specifier "contains a displacement that, is first sign extended to 32 bits and is then added to the contents of R15 to form the address of the Operand"...
C	W^address	Word displacement	
E	L^address	Longword displacement	



PC (R15) Mode Operand Specifiers (continued)
****

Mode	Notation	Mode Name	Description
B	@B^address	Byte Relative Deferred	The contents of the location following the Operand Specifier when sign extended to 32 bits, is added to the PC to form the "Address of the Address of the Operand"...
D	@W^address	Word Relative Deferred	
F	@L^address	Longword Relative Deferred	



SECTION V

... BUSes used on VAX-11/780 Systems ...

SYNCHRONOUS BACKPLANE INTERCONNECT

... The VAX-11/780 Bus ...

S.B.I. Signal Pin Layout

Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
AB1	B00	CV2	FAIL AC LO (401)	EP1	UNJAM
AC1	B01	DA1	DEAD DC LO (516)	EP2	ALERT
AD1	B02			EU1	FAULT
AE1	B03	DB1	M0		
AM1	B04	DC1	M1	ES2	CNF0
AN1	B05	DD1	M2	ET2	CNF1
AP1	B06	DD2	M3		
AP2	B07				
AS2	B08	DM1	TAG0	FB1	TR00
AT2	B09	DN1	TAG1	FC1	TR01
AU1	B10	DP1	TAG2	FD1	TR02
AU2	B11			FE1	TR03
BF2	B12	DP2	ID0	FF2	TR04
BH2	B13	DS2	ID1	FH2	TR05
BJ1	B14	DT2	ID2	FJ1	TR06
BJ2	B15	DU1	ID3	FJ2	TR07
BS2	B16	DU2	ID4	FM1	TR08
BT2	B17			FN1	TR09
BU1	B18	EB1	REQ4	FP1	TR10
BU2	B19	EC1	REQ5	FP2	TR11
CB1	B20	ED1	REQ6	FS2	TR12
CC1	B21	ED2	REQ7	FT2	TR13
CD1	B22			FU1	TR14
CD2	B23	EF1	TP H	FU2	TR15
CM1	B24	EF2	TP L		
CN1	B25				
CP1	B26	EH2	PCLK H		
CP2	B27	EJ1	PCLK L		
CS2	B28	EJ2	PDCLK H	*A2	+5V
CT2	B29	EK2	PDCLK L	*V1	+5V
CU1	B30				
CU2	B31	EM1	MP1		
		EN1	MP2	*C2	GND
DF2	P0			*H1	GND
DH2	P1	FA1	INTLK	*N2	GND
				*T1	GND

... All Signals are Low when True unless otherwise specified ...

... Signals are found on slot #1 of any NEXUS ...

SBI/CPU Time State Equivalents

The VAX-11/780 CPU and SBI time states have different names.

M8232 LED	CPU Time State	SBI Time State
D1 (top)	CPT0	SBI T1
D2	CPT1	SBI T2
D3	CPT2	SBI T3
D4 (bottom)	CPT3	SBI T0
	CPTP	SBI TP
	CPPCLK	SBI PCLK
	CPPDCLK	SBI PDCLK

SBI T0 clock time

1. Nexus that has control of the SBI enables the SBI Drivers at this time.
2. TR Lines are asserted by the NEXUS wishing use of the SBI Bus.

SBI T1 clock time

1. NEXUS dependent.

SBI T2 clock time

1. "ALL" NEXUS open their Receiver Latches at this time.

SBI T3 clock time

1. "ALL" NEXUS clock their Receiver Latches at this time.
2. All NEXUS who have their TR Line asserted, determine if they are next in line to get control of the SBI Bus at the next "SBI T0" time.

S.B.I. WRITE Transfer example to show timing

	1st Cycle	2nd Cycle	3rd Cycle	4th Cycle	5th Cycle	6th
	T0 T1 T2 T3	T0 T1 T2 T3	T0 T1 T2 T3	T0 T1 T2 T3	T0 T1 T2 T3	T0 T1
ARBITRATION	Requester asserts assigned TR line	If Highest Priority, assigned TR is dropped, and TR#0 is asserted to hold next cycle for the WRITE DATA.	TR#0 is deasserted, UNLESS func. was an EXTENDED WRITE - in which case TR#0 will remain asserted	If function was an EXT. WRITE, TR#0 would be deasserted here.		
INFORMATION	TAG<2:0>=? ID<4:0> =? B<31:00>=? M<3:0> = ?	TAG = C/A ID = source B =Func/Addr M =Bytes to be written	TAG = Write Data ID = code of commander B = Data M = Bytes to be written	If function was an EXT. WRITE, the 2nd longword would be sent here. Format would be same as the previous Write Data cycle.		
CONFIRMATION	CNF<1:0> = ?	CNF<1:0> = ?	CNF<1:0> = ?	CNF=response from the destination to verify acceptance of C/A information	CNF=response from the destination to verify acceptance of the Write Data	CNF for second Write Data longwd if func was an EXT. Write

VAX-11/780 INTERNAL DATA BUS

ID BUS

ID Bus chart showing what Modules are fed by what bits

Name	MODULE Number	Slot	ID Bus Bits 0-7	8-15	16-20	21-25	26-31
TRS	M8237	1	X	X			
SBL	M8218	2	X	X			
SBH	M8219	3	X		X	X	X
CAM	M8220	4					X
TBM	M8222	6	X	X	X		
IDP	M8223	7	X	X	X	X	X
IRC	M8224	8	X	X	X	X	X
DEP	M8226	10			X	X	X
DDP	M8227	11		X			
DCP	M8228	12	X				
CEH	M8230	14	X		X	X	X
ICL	M8231	15	X	X			
OCS	M8233/8	18	X	X	X	X	X
WCS	M8233/8	20	X	X	X	X	X
PCS	M8234	22	X	X	X	X	X
USC	M8235	23	X	X			
FMH	M8286	25			X	X	X
FML	M8287	26	X	X			
CIB	M8236	29	X	X	X	X	X

ID Bus Parity bits chart showing who uses these bits

The ID Bus has 4 parity bits ("Bus ID PTY <3:0>"), however, only one board generates parity and only two others use these bits to check the parity. The following chart show who these modules are that generate (G) and use (U) these parity bits.

Name	Module Number	Slot	"Bus ID PTY" bits <0>	<1>	<2>	<3>
TBM	M8222	6	U	U	U	
CAM	M8220	4				U
DBP	M8225	9	G	G	G	G
Signal Pin and Row number			CB1	CM1	AS1	AU2

ID Bus bits <31:00> Backplane Pin list

The ID Bus bits go to the same pin row and pin number on each of the modules that they feed. The following chart shows the row and pin number of each ID bus bit:

Bus ID bit <00> - CA1	Bus ID Bit <16> - AF2
Bus ID bit <01> - CB2	Bus ID Bit <17> - AH2
Bus ID bit <02> - CE1	Bus ID Bit <18> - AJ1
Bus ID bit <03> - CE2	Bus ID Bit <19> - AM1
Bus ID bit <04> - CF1	Bus ID Bit <20> - AN1
Bus ID bit <05> - CF2	Bus ID Bit <21> - AP1
Bus ID bit <06> - CH2	Bus ID Bit <22> - AP2
Bus ID bit <07> - CJ1	Bus ID Bit <23> - AS2
Bus ID bit <08> - CJ2	Bus ID Bit <24> - AT2
Bus ID bit <09> - CK1	Bus ID Bit <25> - AU1
Bus ID bit <10> - CK2	Bus ID Bit <26> - BF2
Bus ID bit <11> - CM2	Bus ID Bit <27> - BH2
Bus ID bit <12> - DR1	Bus ID Bit <28> - BJ1
Bus ID bit <13> - DR2	Bus ID Bit <29> - BJ2
Bus ID bit <14> - DS1	Bus ID Bit <30> - BK1
Bus ID bit <15> - DV2	Bus ID Bit <31> - BK2

SECTION VI

UNIX ERROR REPORTING

To be added in a later release.

SECTION VII

Miscellaneous Information

Using EVSBA.EXE , the Diagnostic Autosizer.

EVSBA is an autosizing program that runs under the Diagnostic Supervisor (ESSAA.EXE) in stand-alone mode. This program will determine the current configuration of the VAX System. It sizes the VAX system and builds a data base of Diagnostic Supervisor ATTACH commands based on the hardware it found during the sizing process. This ATTACH information can then be passed to the Diagnostic Supervisor. The operator can cause the system to be sized completely automatically or can perform the sizing operation in MANUAL or SELFTEST mode. In MANUAL or SELFTEST mode, the operator has the capability to change device names or other device parameters before the information is passed to the Diagnostic Supervisor. If the QUICK flag is set in the Diagnostic Supervisor, no check is made for terminals on the DZ11's, so the program runs very quickly.

The Autosizer program will probe the system buses to determine what devices are connected to the system. Each bus requires a different technique to determine which devices are present. On the SBI each adapter has a type code in the configuration register which clearly identifies the adapter type. Similiarly, each MASSBUS device contains a register which uniquely identifies the device type. The UNIBUS is the most complicated because of floating CSR and VECTOR assignments in addition to fixed CSR's and VECTOR's. Each device optionally requires extra information in order for a diagnostic to verify its operation. The Autosizer will attempt, on a device by device basis, to glean the required information from the device itself. This, of course, assumes that the hardware involved is operating properly. The information gathered by the sizing process can be edited by the operator to fix any errors in sizing. It can then be fed to the Diagnostic Supervisor. The information generated by the Autosizer can be written to an ASCII script file on the Console Floppy.

The Autosizer will size the system bus first to determine what adapters are present. Next, each adapter is considered. Every device on the adapter is probed and the information saved as to characteristics and addresses. If a device has units connected to it, each unit is sized and appropriate information is saved. If fields are required for the ATTACH command and the information cannot be determined from the device the Autosizer will use a predefined value for the field. When this occurs, the operator will be notified that the field may be incorrect for that device.

EVSBA Autosizer Default Mode Operation:

```
DS> SET FLAG QUICK
DS> RUN EVSBA
```

This will cause the Autosizer to size the hardware, pass the information to the Diagnostic Supervisor, and exit back to the Diagnostic Supervisor. No modification of the ATTACH commands as generated by the Autosizer can be done.

EVSBA Autosizer MANUAL or SELFTEST Mode Operation:

In MANUAL or SELFTEST mode, the operator is immediately prompted with "COMMAND?". The operator is given the option of reading the current configuration file from the console floppy or of automatically sizing the system. In either case, the operator is then given an opportunity to change or list any device or parameter. When sizing in the selftest mode, each line is printed as it is produced. Once satisfied with the configuration, the operator may have it passed directly to the Diagnostic Supervisor. The file may also be written to the Console Floppy.

An example of using the Autosizer whenever you are making configuration changes, while isolating a problem, could be as follows:

```
>>> LOAD ESSAA.EXE/ST:FE00          ;Load the Diagnostic Supervisor.
>>> START 10000                     ;Start the Diagnostic Supervisor.
DS> SET QUICK                       ;Elimate "Terminal" autosizing.
DS> RUN EVSBA/SEC:SELFTEST          ;Run Autosizer in "SELFTEST" mode.
COMMAND>SIZE                        ;Autosize the system and list.
COMMAND>ATTACH                      ;Transfer configuration to Diag.Super..
COMMAND>EXIT                        ;Return to Diag. Super..
DS>SELECT ALL                       ;Select attached devices.
```

EVSBA Autosizer Commands for MANUAL or SELFTEST Mode:

READ filespec - This command reads the specified file from the load device and stores the information. If the filename is not specified, "CONFIG.COM" is used. Any information previously known to the Autosizer is lost.

SIZE - Performs the process that sizes the buses and records the configuration information. Any information previously known to the Autosizer is lost.

LIST device - Type out all information about the devices, based on generic names.

HELP - Type out the help test.

WRITE filesp - Write the current file in memory to the Console floppy. If no filespec is given, "CONFIG.COM" will be used.

CHANGE device-field-value
- The specified field(s) for the specified device are given the values specified.

EXIT - Control is returned to the Diagnostic Super..

ATTACH - For each device in the device database, pass the information to the Diagnostic Supervisor.

S tandard P erformance E rror A nalysis R eporting
****

SPEAR is a library of five on-line Field Service maintenance functions. Four of the functions (Analyze, Summarize, Retrieve, and Compute) are designed to help you evaluate system performance and analyze the content of system event files. The fifth function, Instruct, is designed to help you learn to use the the Spear Library to calculate system availability and isolate intermittent system failures.

RETRIEVE - extracts and translates (or saves) system event file entries.

SUMMARIZE - summarizes the contents of system event files.

ANALYZE - attempts to localize the cause of intermittent system failures.

COMPUTE - calculates system availability and crash and uptime statistics.

INSTRUCT - explains how to use the extended Spear Library functions.

SPEAR was designed with ease of usage in mind. This is accomplished by making help information available to each question.

At the SPEAR prompt, you can type:

1. "?" to list the supported Spear Library functions.
2. the "name" of the Spear Library function that you want to execute.
3. "/HELP" for an explanation of the universal Spear Library switches.
4. "@HELP" for information about response streams and indirect files.
5. "EXIT" to exit Spear and return to the operating system.

S P E A R
****

At any function prompt you can type:

/BREAK to return to the Spear prompt.

/REVERSE (or press the BACKSPACE key) to repeat the last prompt.

/SHOW to display the current prompt/responses list.

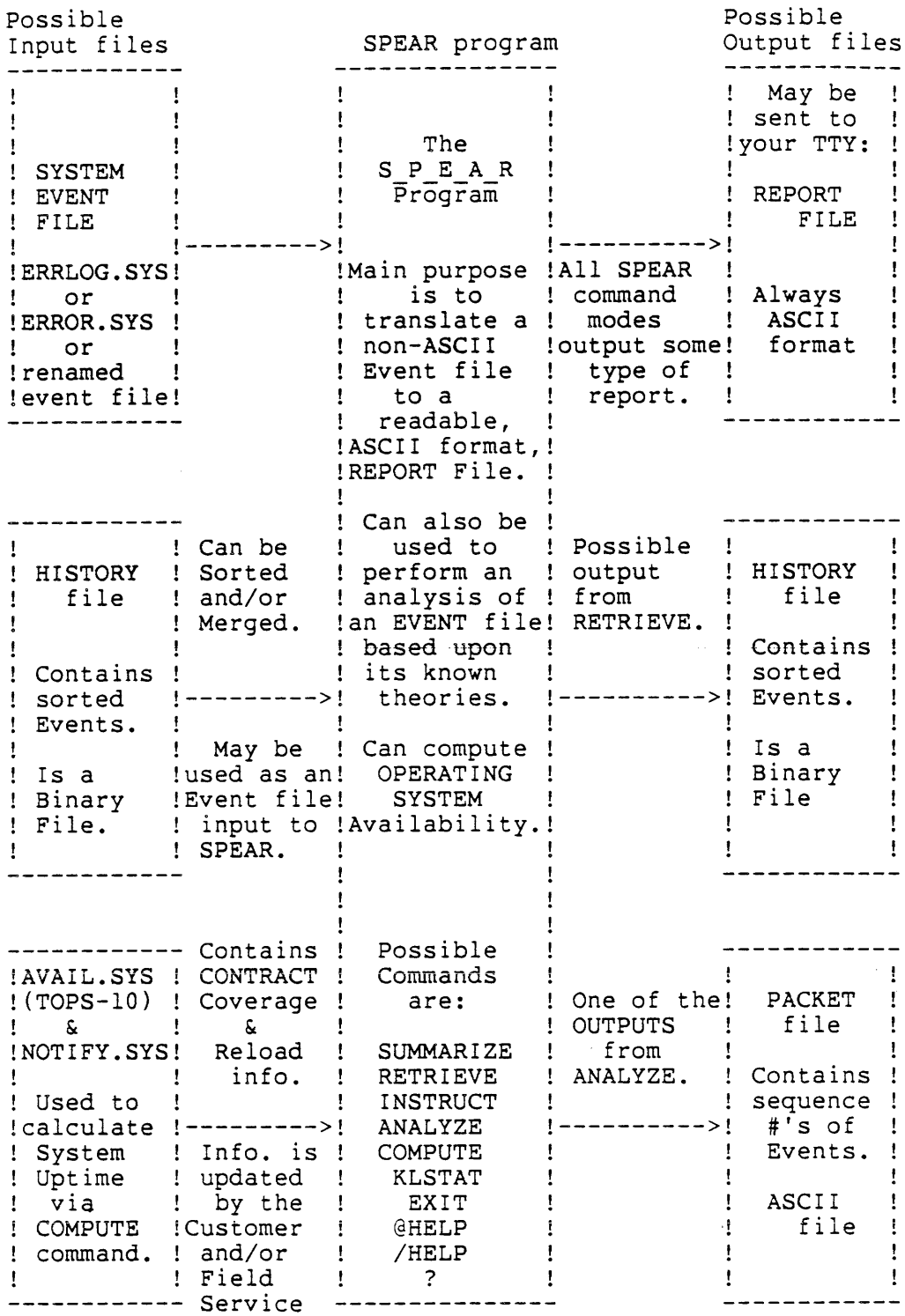
/GO to execute the current prompt/response list.

/CLEAR to clear listed items, at or subordinate to, the current prompt
Listed items are: sequence numbers, entry codes, device types, etc.

/? to display this list without the explanations.

Type @HELP for information about Response Streams and Indirect Files.
Press the RETURN key to specify the default or terminate a response.
Press the ESCAPE (or Altmode) key to: display the default, or complete a partially typed response. There is no default at the SPEAR> prompt.

You can enter a response stream at the main Spear prompt. A response stream is a single line of consecutive responses, separated by spaces, and terminated with a carriage return (use the Escape or Altmode key to insert defaults). Note: The response stream capability is included only as a convenience for those Spear users who do not wish to be prompted.


Possible Input files
SPEAR program
Possible Output files
SYSTEM EVENT FILE
ERRLOG.SYS or ERROR.SYS or renamed event file
The S_P_E_A_R Program
Main purpose is to translate a non-ASCII Event file to a readable, ASCII format, REPORT File.
Can also be used to perform an analysis of an EVENT file based upon its known theories.
Can compute OPERATING SYSTEM Availability.
All SPEAR command modes output some type of report.
May be sent to your TTY:
REPORT FILE
Always ASCII format
HISTORY file
Contains sorted Events.
Is a Binary File.
Can be Sorted and/or Merged.
May be used as an Event file input to SPEAR.
Possible output from RETRIEVE.
HISTORY file
Contains sorted Events.
Is a Binary File
AVAIL.SYS (TOPS-10) & NOTIFY.SYS
Used to calculate System Uptime via COMPUTE command.
Contains CONTRACT Coverage & Reload info.
Info. is updated by the Customer and/or Field Service
Possible Commands are:
SUMMARIZE
RETRIEVE
INSTRUCT
ANALYZE
COMPUTE
KLSTAT
EXIT
@HELP
/HELP
?
One of the OUTPUTS from ANALYZE.
PACKET file
Contains sequence #'s of Events.
ASCII file

```
      ************************************************
      !  Example of How to initiate the SPEAR program  !
      !           on a VAX/VMS System.                 !
      ************************************************


Username:  FIELD
Password:
         Welcome to VAX/VMS version V3.0 on node NEDVAX

$ RUN SYS$SYSTEM:SPEAR
         or
$ MCR SPEAR

 Welcome to SPEAR for VMS, Version 1(35)
 Type "?" for help.


SPEAR>
```

At this Point you may enter the appropiate SPEAR command. If you don't know the commands, simply type "?" and a brief help file will be typed out showing the available commands to this prompt.

```
SPEAR>  ?

      Enter one of the following modes

 Instruct    --   in the usage of SPEAR
 Retrieve    --   individual event file entries
 Analyze     --   a system event file
 Compute     --   system availability ststistics
 Summarize   --   various event counts
 EXIT        --   to exit from SPEAR (if at SPEAR> prompt)
 For more info type /HELP

 For further information type: HELP

SPEAR>
```

The commands to the SPEAR prompt can be abbreviated to one character as follows:

Analyze	=	A
Retrieve	=	R
Compute	=	C
Summarize	=	S
Instruct	=	I
Exit	=	E

```
                Summary of QUESTIONs asked by SPEAR
                ***********************************


SUMMARIZE question.
-------------------

    Event File (SYS$SYSDISK:[SYSERR]ERRLOG.SYS)
    Time from (EARLIEST):
    Time to (LATEST):
    Report to (SUMMAR.RPT)
    Type [cr] to confirm (/GO):


RETRIEVE questions.
-------------------

    Event File (SYS$SYSDISK:[SYSERR]ERRLOG.SYS)
    Selection to be (INCLUDED):
    Selection type (ALL):
            Error class (ALL):
                    Sequence numbers:
                    Event codes:
                    Next error class (FINISHED):
    Time from (EARLIEST):
    Time to (LATEST):
    Output mode (ASCII):
    Report format (SHORT):
    Output to (RETRIE.RPT):
    Type [cr] to confirm (/GO):


ANALYZE questions.
------------------

    Event File (SYS$SYSDISK:[SYSERR]ERRLOG.SYS)
    Time from (-1):
    Time to (LATEST):
    Report to (Ammdd.RPT):
    Packets to (Ammdd.PAK):
    Type [cr] to confirm (/GO):


COMPUTE questions.
------------------

    Event File (SYS$SYSDISK:[SYSERR]ERRLOG.SYS)
    Report period (LAST-WEEK):
    Availability report to (COMPUT.RPT):
    Reload report to (RELOAD.RPT):
    Type [cr] to confirm (/GO):
```

Examining Unibus Registers

It is sometimes a pain in the neck to recalculate the SBI physical address of Unibus Devices while trouble-shooting. Here is a method of examining/depositing Unibus Device Registers that eliminates the need to calculate the SBI physical address. Set up CONSOL.SYS as follows:

```
>>> SET DEFAULT HEX                        ! Just in case not already set.
>>> SET REL:20100000                       ! Set offset for UBA at TR #3.
>>> SET DEFAULT OCTAL,WORD,PHYSICAL        ! Change defaults for exam/dep
```

Now you can examine/deposit Unibus Device registers by specifing the Unibus Device Address. For example, to examine the LP11 status reg., simply type the following to the CONSOL.SYS prompt:

```
>>> E 777514            (LP11)
```

LP11 Diagnostic check under VMS

The Line Printer diagnostic must be run under VMS. In order to do this, the Line Printer Queue must first be stopped,(if there is a Line Printer queue on your particular system). The following commands are used to stop the queue for LPA0:. If you are testing another printer, use the appropriate designation.

```
$ STOP/QUEUE/NEXT LPA0:
$ DELETE/QUEUE LPA0:
$ RUN ESSAA
DS> @CONFIG
DS> RUN EVAAA
```

To Restore LP11 queue

After the Line Printer diagnostic is run, you must now restart the Line Printer Queue. Use the following commands:

```
$ INIT/QUEUE/FLAG LPA0:
$ START/QUEUE LPA0:
```

Defining and Starting Print queues (LP11)

The following commands can be used to initialize Print queues. If you want to initialize a queue for a device other than "LPA0:", simply replace "LPA0:" with the appropriate designation.

```
$ SET PRINTER/PAGE=64/LP11 LPA0:
$ SET DEVICE/SPOOL LPA0:
$ INIT/QUEUE/FLAG/GENERIC SYS$PRINT
$ INIT/QUEUE/FLAG LPA0:
$ START/QUEUE SYS$PRINT
$ START/QUEUE LPA0:
```

Defining and Starting Terminal queues (LA36)

The following commands can be used to initialize queues for terminals:

```
$ INIT/QUEUE/TERM TTXY:                    (XY = Terminal name)
$ SET TERM/PERM/LA36/PAGE=66/NOBROADCAST TTXY:
$ SET DEVICE/SPOOLED=TTXY: TTXY:
$ START/QUEUE TTXY:
```

Bugcheck or Crash Restart with message: "UNEXPECTED UNIBUS ADAPT. INTERRUPT"

R0 thru R5 contain the following information:

R0 = UBA CONFIGURATION REGISTER
R1 = UBA CONTROL REGISTER
R2 = UBA STATUS REGISTER
R3 = UBA DIAGNOSTIC CONTROL REGISTER
R4 = UBA FAILED MAP REGISTER
R5 = UBA FAILED UNIBUS ADDRESS REGISTER

Interleaving Memories

The following commands can be entered to CONSOL.SYS in order to interleave two MS780 memories (memories must be at TR#1 and TR#2).

```
>>> D 20002000 101
>>> D 20002004 4000
>>> D 20004000 101
>>> D 20004004 4000
```

Booting with CACHE Disabled

The following commands can be entered to CONSOL.SYS in order to boot the SYSTEM with CACHE Disabled:

```
>>> D/ID 1D 18000
>>> D R0 0/N:5
>>> D R1 8
>>> D FP 0
>>> D/I 11 20003800
>>> D SP 200
>>> L VMB.EXE/ST:200
>>> D PC 200
>>> CONT
```

H7100 Power Regulator LED's

The following chart shows what the LED's on the front of the H7100 Power Regulators mean:

LED indicator	Description
POWER NORMAL	Power is O.K..
PLUG IN REGULATOR FAILURE	Problem with one or more of the following regulators: +5 +5B or -5B +12
OVER CURRENT	+5v at 120 amps or more. (120% over).
OVER VOLTAGE	+5v is +6.2v or greater.
POWER INVERTER FAILURE	Main +5v failure.
OVER TEMP	Internal Temperature at 150 degrees F or more.

M8232, Clock Board, Jumpers
****

W1 thru W14	Installed when FP780 is installed.
W23, W24	Installed when Optional WCS is installed. Optional WCS is in slot 18.
W15, W16	Installed to ENABLE FAIL/DEAD onto SBI if there is a Power Failure.
W17 thru W22	Installed to ENABLE SBI Clock signals onto SBI Bus.
W15 thru W22	REMOVED only when the associated CPU receives its clock signals from another device on the SBI, such as a second CPU.

LSI-11 Controls and Indicators
****

DC ON	Illuminates when the DC ON/OFF toggle switch is set to ON and proper DC output voltages are being produced by the LSI power supply. If either the +5v or +12v outputs from the LSI are faulty, the DC ON indicator does not go on.
RUN	Illuminates when the LSI-11 processor is in the run state (refer to ENABLE/HALT)
DC ON/OFF	When set to ON, enables the DC outputs. The DC ON indicator illuminates if the DC output voltages are of proper values. When set to OFF, the DC outputs are disabled and the DC ON indicator is extinguished.
ENABLE/HALT	When set to ENABLE, the B HALT L line is not asserted and the processor is in the run mode (RUN indicator illuminated). When set to HALT, the B HALT L line is asserted allowing the processor to execute the console ODT microcode (RUN indicator is extinguished).
LTC ON/OFF	When set to ON, enables the generation of the Line Time Clock "BEVNT L" signal.

VAX-11/780 Controls and Indicators
****

AUTO RESTART	When in the ON (down) position, the VAX-11/780 CPU is restarted automatically following a Power Recovery or Error Halt.
BOOT	When pressed, the operation system is boot-strapped. When the bootstrap operation is completed, the console is set to the "Program I/O" mode of operation.
ATTN	When lit, indicates that the VAX-11/780 CPU is halted.
RUN	When lit, indicates that the VAX-11/780 CPU is strobing interrupts (microcode running properly).
POWER	When lit, indicates that the +5v power supply is on.
REMOTE	When lit, indicates that remote access is enabled.

Key Switch

OFF	In this position, the power is turned OFF.
LOCAL DISABLE	In this position, Remote access is disabled and Console I/O mode is inhibited.
LOCAL	In this position, Remote access is disabled, but Console I/O mode is not inhibited.
REMOTE DISABLE	In this position, Remote access is enabled and Console I/O mode of operation is inhibited.
REMOTE	In this position, Remote access is enabled and the Console I/O mode is not inhibited.

MS780/MA780 Error Correction Logic

The ECC (Error Correction Logic) within the MS780 and the MA780 can give you false indications in a couple of special cases.

1. If the MOS Array outputs an "ODD MULTIPLE number of BAD Bits", to the MOS DATA bus on a memory read, the Memory's Error Correction logic will send the DATA to the SBI as "Corrected Read Data" (after making an attempt to correct a bit, which may be a bit that wasn't even bad in the first place).

   For example: If the MOS Array outputs a quadword with 3, 5, 7, 9, or etc. bad bits, the Error Correction Logic will think that a Single Bit Error has occured, will correct a bit (possibly not even one that was really bad), and will then send the data to the SBI as "Corrected Read Data".

2. If a "Single Bit Error" has occured and has not been serviced before a "Double Bit Error" occurs in the same memory controller's arrays, the registers will contain information about the "Single Bit Error" and the information about the "Double Bit Error" will be lost.

EVKAA.EXE

This diagnostic is a valuable diagnostic that should be run after the running of the micro-diagnostics and before attempting to run the "Diagnostic Supervisor". This diagnostic is a VAX macro functional diagnostic and does not use the "Diagnostic Supervisor". Run this program as follows:

```
>>> LOAD EVKAA.EXE/ST:0
>>> START 200
```

Sometimes, when restarting EVKAA, the "DS>" prompt will appear on the Console Terminal. This is caused by the APT control flag, bit 31 of physical location FE00, being set. Clear this control flag and restart EVKAA as follows:

```
DS> ^P
>>> HALT
>>> D/L/P/H FE00 0
>>> START 200
```

S E C T I O N VIII

NEXUS Register Bit Definitions

This chapter contains the definitions of the NEXUS registers as defined by the individual NEXUS manuals. The purpose of this chapter is to provide all the bit definitions in one place since the VAX Maintenance guides do not include the definitions of the NEXUS register bits. This information was copied from the various VAX-11/780 NEXUS Hardware manuals and microfiche.

DW780 Configuration Register

CNFGR *Offset = 000(16)*

Bit 31, Parity Fault (PAR FLT)

Is set shen the UBA detects an SBI parity error.

Bit 30, Write Sequence Fault (WSQ FLT)

Is set when the UBA receives a write masked or interlock write masked command and does not receive the expected write data in the following cycle.

Bit 29, Unexpected Read Data Fault (URD FLT)

Is set when the UBA receives data for which a read masked, extended read, or interlock read masked command has not been issued.

Bit 28, Interlock Sequence Fault (ISQ FLT)

Is set when an interlock write masked command to UNIBUS address space is received by the UBA without a previous interlock read masked command.

Bit 27, Multiple Transmitter Fault (MXT FLT)

Is set when the UBA is transmitting on the SBI and the ID bits transmitted by the UBA do not match those latched from the SBI. The lack of correspondence indicates a multiple transmitter condition.

Bit 26, Transmit Fault (XMT FLT)

Is set if the UBA was the transmitter during a detected fault condition. When the software subsequently reads the configuration and status registers of each of the nexus on the SBI in order to identify the source of the fault, the UBA will be identified as that source if bit 26 is set.

Bit 25,24 Reserved

Should be cleared (zero).

Bit 23 Adaptor Power Down (AD PDN)

Is set when the UBA power supply asserts ACLO. It is cleared by writing a one to the bit location or when the Adaptor Power Up bit is set.

Bit 22 Adaptor Power Up (AD PUP)

Is set by the negation of power supply ACLO. It is cleared by writing a one to the bit location or by the setting of the Adaptor Power Down bit.

Bits <21:19> Reserved

Should be cleared (zero).

Bit 18 Unibus Init Asserted (UB INIT)

The assertion of UNIBUS INIT will set this bit. It is cleared by the setting of the Unibus Initialization Complete bit or by the writing of a one to this bit location.

Bit 17 Unibus Power Down (UB PDN)

Is set when UNIBUS ACLO is asserted. It indicates that the Unibus has initiated a power down sequence. The setting of the UBIC bit or writing a one to this location will clear UB PDN.

Bit 16 Unibus Initialization Complete (UBIC)

Is set by a successful completion of a power up sequence on the UNIBUS. It is the last of the status bits to be set during a UBA initialization sequence, and it can be interpreted to mean that the UBA and the UNIBUS are ready. The assertion of Unibus INIT, or the writing of a one to this bit location will clear UBIC.

Bits <7:0> Adaptor Code

Bits 5 & 3 = 1
Bits 7,6,4 & 2 = 0
Bits <1:0> are determined by backplane jumpers and reflect the UBA number. The adaptor codes indicate the starting address of the Unibus address space associated with the UBA.

DW780 Control Register

UACR *Offset = 004(16)*

Bit 31 Reserved

Should be cleared (zero).

Bits <30:26> Map Register Disable (4:0) (MDR)

This field of five read/write bits disables map registers in groups of sixteen, according to the binary value contained in the field. The MRD bits prevent the UBA from responding to a UNIBUS address that points to a disabled map register. The software will load this field with a binary value equal to the number of 4k word units of memory attached to the UNIBUS. DMA transfers to addresses pointing to disabled map registers are not recognized by the UBA. No error bits are set and no SBI transfers are initiated. However, SBI access to disabled map registers is permitted. The MRD field is initialized as zero, with all map registers enabled. Note, however, that in the initialized state the map registers are all invalid. False UNIBUS transfers are prevented in this way.

Bits <25:07> Reserved

Should be cleared (zero).

Bit 5 Bus Request Interrupt Enable (BRIE)

When this bit is set it allows the UBA to pass interrupts from the UNIBUS to the VAX CPU, providing that the IFS is set. The power up state of the BRIE bit is 0. The bit is also cleared by the Adaptor INIT, SBI UNJAM, and SBI DEAD signals.

Bit 4 UNIBUS to SBI Error Field Interrupt Enable (USEFIE)

This bit enables an interrupt request to the VAX CPU whenever any of the following status register bits are set on a DMA transfer:

- RDTO - Read Data Timeout
- RDS - Read Data Substitute
- CXTER - Command Transmit Error
- CXTMO - Command Transmit Timeout
- DPPE - Data Path Parity Error
- IVMR - Invalid Map Register
- MRPF - Map Register Parity Failure

The power up state of this bit (USEFIE) is 0. SBI UNJAM and Adaptor INIT will clear USEFIE.

Bit 3 SBI to UNIBUS Error Field Interrupt Enable (SUEFIE)

If this bit is set, the UBA will generate interrupt requests to the VAX CPU when one of the two bits in the SBI to UNIBUS data transfer error field of the status register is set:

UBSTO - Unibus Select Timeout
UBSSYNTO - Unibus Slave Sync Timeout

The power up state of the SUEFIE bit is 0. SBI UNJAM, SBI DEAD, and Adaptor INIT will also clear this bit.

Bit 2 Configuration Interrupt Enable (CNFIE)

If this bit is set, the UBA will initiate an interrupt request to the VAX CPU whenever any of the environmental status bits of the Config. register is set. These bits are:

AD PDN - Adaptor Power Down
AD PUP - Adaptor Power Up
UB INIT - Unibus Init Asserted
UB PDN - Unibus Power Down
UBIC - Unibus Initialization Complete

The power up state of the CNFIE bit is a 1. CNFIE is cleared by Adaptor INIT, SBI UNJAM, and SBI DEAD.

Bit 1 UNIBUS Power Fail (UPF)

When set, it initiates a power fail sequence on the UNIBUS, asserting ACLO, DCLO, and INIT in their proper sequence. The software uses this bit to initialize the UNIBUS. The UNIBUS will remain powered down as long as UPF is set. The clearing of the UPF bit will initiate a UNIBUS power up sequence if or when the UNIBUS power down sequence has finished and UNIBUS power is OK. Thus, the software can initialize the UNIBUS by setting and the clearing the UPF bit.

Bit 0 Adaptor INIT (ADINIT)

When this bit is set it will completely initialize the UBA and the UNIBUS. The map registers, the data path registers, the status reg., and the control register will be cleared. The UBA will start the initialization routine in the microsequencer, and it will generate a power fail sequence on the UNIBUS. The UBA initialization sequence takes only 500 microseconds to complete, while the UNIBUS power fail sequence requires approximately 25 milleseconds.

Only the configuration register and the diagnostic control register can be read during the adaptor initialization sequence. Only the configuration register, the diagnostic control register, the control register, and the status register can be written during the adapter initialization sequence.

Once the sequence has been completed, all UBA registers can be accessed. However, the UNIBUS cannot be accessed until the UNIBUS initialization has been completed as well. The software can test for this condition by reading the UBIC bit of the configuration register, or by setting the CNFIE bit of the control register and looking for the interrupt generated by the setting of the UBIC bit. Note, however, that the assertion of either UNIBUS INIT or UNIBUS power down will also initiate an interrupt (UBINIT). The Adaptor INIT bit can be set by writing a one to the bit location; it is self clearing.

DW780 STATUS Register

USAR *Offset = 008(16)*

Bits <31:28>

Reserved and zero.

Bits <27:24> BR Receve Vector Register Full

Bit 27 = BRRVR 7 Full
Bit 26 = BRRVR 6 Full
Bit 25 = BRRVR 5 Full
Bit 24 = BRRVR 4 Full

These bits indicate the state of the SBI addressable BRRVR's. Each bit is loaded into the corresponding BRRVR during a UNIBUS interrupt transaction, providing that the SBI processor is fielding UNIBUS device interrupts.

Each bit is cleared by the successful completion of a read data transmission following a read BRRVR command. The software will see these bits set only after a read data failure has occurred during the execution of a read BRRVR command and the UNIBUS interrupt vector has been saved by the UBA. These bits are cleared only by a subsequent read to the corresponding BRRVR or by an adaptor initialization sequence.

Bits <23:11>

Reserved and zero.

Bit 10 Read Data Time Out (RDTO)

The UBA sets the RDTO bit when the following contitions are true:

- A UNIBUS device has initiated a DMA transfer.
- The UBA has successfully transmitted a read command on the SBI.
- The SBI memory has not returned the requested data within 100 microseconds, and the UNIBUS device has not timed out.

Note that the normal UNIBUS timeout is 10 microseconds and after the 10 microseconds, the UNIBUS device will set its non-existent memory bit.

Thus, the RDTO bit will be set in the UBA status register only if the UNIBUS device timeout function is inoperative, or takes more than 100 microseconds to timeout. This bit is not set for a BDP to SBI prefetch.

Bit 9 Read Data Substitute (RDS)

This bit is set if a read data substitute is received in response to a UNIBUS to SBI read command (DMA read transfer). No data will be sent to the UNIBUS device, and when the device timeout occurs it will set the non-existent memory bit in the device's register.

Bit 8 Corrected Read Data (CRD)

The UBA sets this bit when it receives corrected read data in response to an SBI read command during a DMA read transfer.

Bit 7 Command Transmit Error (CXTER)

The UBA sets this bit when it receives an error confirmation in response to an SBI command transmission during a UNIBUS to SBI access, a BDP to SBI read, a BDP to SBI write, or a PURGE operation. This bit is not set for a BDP to SBI prefetch.

Bit 6 Command Transmit Timeout (CXTMO)

The UBA sets this bit when it fails to complete an SBI command transfer within 100 microseconds for any of the following operation:

- a BDP to SBI write
- a BDP purge operation
- a BDP to SBI read operation for which the UNIBUS device has not timed out

This bit is not set for a timeout for a BDP to SBI prefetch.

Bit 5 Data Path Parity Error (DPPE)

This bit is set when a parity error in a buffered data path occurs during either a UNIBUS to BDP read, BDP to SBI write, or a BDP purge operation.

Bit 4 Invalid Map Register (IVMR)

The UBA sets this bit during a UNIBUS DMA transfer or purge operation when the UNIBUS address points to a map register that has not been validated by the software and has not been disabled by the MRD bits.

Bit 3 Map Register Parity Failure (MRPF)

This bit is set with the occurrence of a map register parity error during one of the following operations:

- A UNIBUS access in which the UNIBUS address points to a map register that has a parity error in the upper 16 bits, providing that the map register has not been disabled by the MRD bits.
- Mapping a UNIBUS address to an SBI address during a direct data path to SBI operation or a BDP to SBI read operation (but not during a prefetch).
- Mapping an address from a buffered data path to an SBI address during a purge operation or a BDP to SBI write.

Seven of the previously defined bits (RDTO, RDS, CXTER, CXTMO, DPPE, IVMR, and MRPF) form an error locking field. If any of these bits is set, the field is locked, thereby preventing the setting of other bits within this field, until the bit indicating the error is cleared. The failed map entry register (FMER) is also locked and unlocked with this field. The setting of any of these bits will cause the UBA to initiate an interrupt request if the interrupt enable bit for the UNIBUS to SBI data transfer error field (USEFIE) in the control register is set.

Bit 2 Lost Error Bit (LEB)

The UBA sets this bit if the locking error field is locked and another error within the field occurs. The lost error bit does not initiate an interrupt request.

Bit 1 Unibus Select Time Out (UBSTO)

The UBA sets this bit if it cannot gain access to the UNIBUS within 50 microseconds in the execution of a software initiated transfer (SBI to UNIBUS transfer). When UBSTO is set it indicates that the UBA has issued NPR on the UNIBUS but has not become bus master. This condition indicates the presence of a hardware problem on the UNIBUS. The UNIBUS may be inoperative, or one device may be holding it for extended periods of time. Note that if the UNIBUS does become inoperative, it may be possible to clear the problem with the assertion of UNJAM on the SBI, the setting and clearing of the UNIBUS POWER FAIL bit (Control register bit 1) or the setting of ADAPTOR INIT (Control register bit 0).

Bit 0 UNIBUS Slave Sync Time Out (UBSSYNTO)

This bit is set when an SBI to UNIBUS transfer (software initiated transfer) times out during the data transfer cycle on the UNIBUS. The timeout occurs after 12.8 microseconds. "UBSSYNTO" indicates a transfer failure resulting when a non-existent memory or device on the UNIBUS is addressed.

NOTE:
"UBSTO" and "UBSSYNTO" form an SBI to UNIBUS transfer error locking field. They are set by the occurrence of the conditions mentioned and cleared by writing a one to the bit location. The setting of either bit will cause the UBA to make an interrupt request on the SBI if the SBI to UNIBUS error interrupt enable bit (SUEFIE) is set. The setting of either UBSTO or UBSSYNTO will lock the failed UNIBUS address register (FUBAR), thus storing the high 16 bits of the UNIBUS address identified with the failure. The FUBAR will remain locked until the UBSTO and UBSSYNTO bits are cleared.

DW780 Diagnostic Control Register

DCR *Offset = 00C(16)*

Bit 31 SPARE

This read/write bit has no effect on any UBA operation. It can be set by writing a zero to the bit location. SBI DEAD, Adaptor INIT, and a power up sequence on the UBA will clear this bit.

Bit 30 Disable Interrupt (DINTR)

When it is set, this bit will prevent the UBA from recognizing interrupts on the UNIBUS. It is useful in testing the response of the UBA to the passive release condition during a UNIBUS interrupt transaction. This bit is set by writing a one and cleared by writing a zero to the bit location. SBI DEAD, Adaptor INIT, and the power up sequence on the UBA will also clear DINTR.

Bit 29 Defeat Map Parity (DMP)

When it is set, this read/write bit will inhibit the parity bits of the map registers from entering the map register parity checkers. The map register parity generator/checkers generate and check parity on eight bit quantities. Each parity field (eight data bits and one parity bit) is implemented so that the total number of ones in the field is odd.

For example, if bits <7:0> of the map register equals zero or contain an even number of ones then the parity bit equals one. However, if the DMP bit is set, then the parity bit is disabled and the parity checkers will see all zeros. This results in a map register parity failure. Then if the DMP bit is cleared, the parity checkers will see correct parity. Note, however, that if bits <7:0> of the map register contains an odd number of ones, the generated parity bit will be zero. The state of the DMP bit, therefore, will have no effect on the parity result in this case.

When the integrity of the parity generator/checkers is to be tested, the map register must contain data such that at least one of the bytes contains and even number of ones. The DMP bit, when set, will disable the parity bit, and the map register parity failure can be detected during a DMA transfer. SBI DEAD, Adaptor INIT, and the power up sequence on the adaptor will clear the DMP bit.

Bit 28 Defeat Data Path Parity (DDPP)

The DDPP bit functions in the same manner as the DMP bit. When it is set, the DDPP bit will inhibit the parity bits of the data path RAM from entering the parity checkers. The data path parity generator/ checkers generate and check parity on eight bit data units. Each parity field (8 data bits and 1 parity bit) is implemented so that the total number of ones in the field is odd. When the integrity of the parity generator/checkers is to be tested through use of the DDPP bit, a data path parity failure will result during a UNIBUS to BDP read, a BDP to SBI write, or a purge operation. SBI DEAD, Adaptor INIT, and the power up sequence on the UBA will clear the DDPP bit.

Bit 27 Microsequencer OK (MIC OK)

The MIC OK bit is a read only bit which indicates that the UBA micro-sequencer is in the idle state. The microsequencer will enter the idle state after it has completed the initialization sequence or once it has completed a UBA function.

The MIC OK bit can be used by the diagnostic to determine whether or not the microsequencer has completed a successful power up sequence and whether or not it is caught up in any loops. Note that SBI DEAD, UBA power supply DCLO, and Adaptor INIT force the microsequencer into the intitialization routine. Once the routine has been completed and the microsequencer has entered the idle state, MIC OK will be true (1).

Bits <26:24>

Reserved and zero.

Bits <23:00>

These bits are the same as bits <23:00> of the Configuration Register.

DW780 Failed Map Entry Register

FMER *Offset = 010.018(16)*

The FMER contains the map register number used for either DMA transfer or a purge operation that has resulted in the setting of one of the following error bits of the status register: IVMR, MRPF, DPPE, CXTMO, CXTER, RDS, RDTO. This register is locked and unlocked with the UNIBUS to SBI data transfer error field of the status register. The FMER is a read only register. Attempts to write to the FMER will result in an SBI error confirmation. When the FMER is not locked, its contents are invalid. The software can read the FMER to obtain the map register number associated with the failure. It can then read the contents of the failing map register to determine the number of the data path that failed.

Bits <31:09>

Reserved and zero.

Bits <08:00> Map Register Number (MRN)

These bits contain the binary value of the number of the map register that was in use at the time of a failure. Bits <08:00> correspond to bits <17:09> of the UNIBUS address.

DW780 Failed Unibus Address Register

FUBAR *Offset = 014.01C (16)*

The FUBAR contains the upper 16 bits of the UNIBUS address translated from an SBI address during a previous software intiated data transfer. The occurrence of either "Unibus Select Time Out (UBSTO)" or "UNIBUS Slave Sync Time Out (UBSSYNTO)" will lock the FUBAR. When the error bit is cleared, the register will be unlocked.

The FUBAR is a read only register. Attempting to write to the register will result in an error confirmation. No signals or conditions will clear the register.

Bits <31:16>

Reserved and zero.

Bits <15:00> Failed Unibus Address Bits <17:00>

Bits <15:00> are the UNIBUS address bits <17:00>, respectively, of the of the failing UNIBUS memory or device address.

DW780 Buffer Selection Verification Registers 0-3

BRVSR 0-3 *Offsets 020-02C(16)*

These four read/write do-nothing registers are provided to give the diagnostic software a means of accessing and testing the integrity of the data path RAM. Four locations in the data path RAM have been assigned to these registers. Writing and reading the BRSVR's has no effect on the behavior of the UBA. The BRVSR bit configuration is as follows:

Bits <31:16>	Bits <15:00>
Not used.	Test Data bits.

DW780 BR Receive Vector Registers 4-7

BRRVR4-7 *Offsets = 030-03C(16)*

The UBA contains four BRRVRs: BRRVR7, BRRVR6, BRRVR5, and BRRVR4. Each BRRVR corresponds to a UNIBUS interrupt bus request level: 7, 6, 5,& 4. Each BRRVR is a read only register and will contain the interrupt vector of the UNIBUS device interrupting at the corresponding BR level. Each BRRVR is read by the software as part of the UBA interrupt service routine. Note that the UBA interrupt service routine is the routine to which the VAX CPU will transfer control once it has determined that the UBA or the UNIBUS has issued an interrupt request to the SBI.

If the IFS and BRIE bits on the control register are set so that UNIBUS interrupt requests are passed to the SBI, then the CPU responds with an interrupt summary read command. The UBA sends its request sublevel as an interrupt summary response. The software then invokes the UBA interrupt service routine, initiating a read transfer to the appropriate BRRVR. The UBA will assert the contents of the BRRVR on the SBI as read data if the corresponding BRRVR full bit in the status register is set. If the BRRVR full bit is not set, the read BRRVR command causes the UBA to fetch the interrupt vector from the interrupting UNIBUS device. The interrupt vector is loaded into the BRRVR only at the successful completion of the UNIBUS transaction. The UBA will then send the contents of the BRRVR to the SBI as read data. Following this exchange, the UBA interrupt service routine will use the contents of the BRRVR to branch to the appropriate UNIBUS device service routine.

There are 5 types of adnormal completion conditions that may occur during a UNIBUS to SBI interrupt sequence.

1. If the software attempts to read a BRRVR for which a BR interrupt line is not asserted, and the BRRVR is not full, the zero vector (all zeros data) will be sent as read data.

2. If the BR line causes an interrupt sequence to begin on the SBI but is released before the interrupt summary read transfer (passive release), then the interrupt summary response from the UBA will be zero.

3. If the BR line asserted by the interrupting UNIBUS device is released after the interrupt summary read transfer but before the read BRRVR (passive release), then zero will be sent as read data for the read BRRVR command.

4. If the vector has been received from the interrupting device, but an ACT confirmation is not received following the interrupt summary response (read data transmission), then the BRRVR will not be cleared, and the BRRVR full bit will remain set. Subsequent read commands to the full BRRVR will cause the UBA to send the stored vector, but the BRRVR will remain full until the UBA receives an ACK confirmation for the read data. Note that the BRRVR full bits always reflect the state of the BRRVRs.

5. If the IFS bit in the control register is cleared and the software reads a BRRVR, then the zero vector will be sent as read data to the SBI.

The contents of the BRRVRs are also used by the software to determine whether or not the UBA itself has an interrupt pending. Bit 31 of the BRRVR is the adaptor interrupt request indicator. Although the bit is present in all four BRRVRs, it will be active only in the BRRVR corresponding to the interrupt request level that has been assigned to the UBA. If bit 31 is set when the software reads the BRRVR, then an adaptor interrupt request is pending.

Bit 31 Adaptor Interrupt Request Indicator

0 = No UBA interrupt pending.
1 = UBA interrupt pending.

Bits <30:16>

Reserved and zero.

Bits <15:00> Device Interrupt Vector Field

These bits contain the device interrupt vector loaded by the UBA from the UNIBUS during the UNIBUS interrupt transaction.

DW780 Data Path Register 0-15

DPR0-15 *Offsets = 40-7C(16)*

The UBA contains 16 data path registers (DPR0 thru DPR15), each of which corresponds to one of the 16 data paths. The DPRs contain status info relative to the buffered data paths and provide the means for purging and initializing the BDPs at the completion of a UNIBUS block transfer for DP1:DP15. DPR0 corresponds to the DDP and is, therefore, always zero.

Bit 31 Buffer Not Empty (BNE)

Each DRP contains a data path status bit called Buffer Not Empty.

1 = Buffer Not Empty
0 = Buffer Empty

The BNE bit reflects the state of the associated BDP. If this bit is set (1), the BDP contains valid data. If clear (0), then the BDP does not contain valid data. The UBA uses the bit to determine the proper action for DMA transfers via the BDP. If bit 31 is set as a DATI transfer begins, the data in the BDP will be asserted to the UNIBUS. If bit 31=0 on a DATI, the UBA will initiate a read transfer to the SBI memory, gate the addressed data to the UNIBUS, and then load the read data into the BDP, thereby setting bit 31.

For DMA write transfer via the associated BDP, the BNE bit is set each time UNIBUS data is loaded into the BDP. The bit is then cleared when the contents of the BDP are transferred to SBI memory.

The software will write a one to the BNE bit to initiate a purge operation at the completion of a DMA transfer using the corresponding buffered data path (BDP). The UBA executes purge operations as follows:

1. Write Transfers To Memory - If any bytes of data remain in the corresponding BDP (BNE is set), the UBA will transfer this data to the SBI location addressed. The UBA will then initialize the BDP and clear the BNE bit. If no data remains to be transferred (BNE=0), the purge operation will be treated as a no-op.

2. Read Transfers To Memory - If any bytes of data remain in the BDP, the UBA will initialize the BDP by clearing the BNE bit.

In addition, the following considerations apply to the purge operation:

- For purge operations in which data is transferred to memory, the SBI transfer takes about 2 microseconds. The UBA will not respond to data path register read transfers during this period (a BUSY confirmation is returned on attempted accesses) thereby preventing a race condition when testing for the BNE bit.

- A purge operation to data path register 0 (Direct Data Path) is treated by the UBA as a no-op.

Bit 30 Buffer Transfer Error (BTE)

This is a read-write-one-to-clear bit. The UBA sets the BTE bit if a failure occurs during a BDP to SBI write or purge, or for a buffer parity failure during a UNIBUS to BDP read access. If bit 30=1, any additional DMA transfers via the BDP will be aborted until the bit is cleared by the software. Note that if a parity error on the UNIBUS occurs during a DMA read, the UNIBUS PB signal will be asserted, giving the UNIBUS device the opportunity to abort its own DMA transfer. If the device does not abort its own transfer, the UBA will abort the transfer on the next access. The purge operation does not clear the BTE bit. The software clears this bit by writing a one to the bit.

Bit 29 Data Path Function (DPF)

The DPF is a read only bit. This bit indicates the function of the DMA transfer using this data path.

0 = DMA Read
1 = DMA Write

Bits <28:24>

Not used.

Bits <23:16> Buffer State (BS)

These eight read only bits indicate the state of each of the eight byte buffers of the associated BDP during a DMA write transfer. They are included in the data path register for diagnostic purposes only. The UBA generates the SBI mask bits from the BS bits during a BDP to SBI write transfer or purge operation. The bits are set as each byte is written from the UNIBUS. The bits are cleared during the SBI write operation.

0 = Empty
1 = Full

Bits <15:00> Buffered Unibus Address (BUBA)

This portion of each DPR contains the upper 16 bits of the UNIBUS address, UA<17:02>, asserted during a UNIBUS to BDP write transfer using the associated BDP. If the transfer through the associated BDPs is in the byte offset mode, and the last UNIBUS transfer has spilled over into the next quadword, then these bits contain UA<17:02>.

BUBA<15:00> = Upper 16 bits of Unibus Address<17:00> + Byte Offset

This is the UNIBUS address from which the SBI address will be mapped during the purge operation.

DW780 Map Registers 0-495

MR0-495(10) *Offsets = 800-FBC(16)*

The UBA contains 496(10) map registers, one for each UNIBUS memory page address (a page of UNIBUS addresses = 512(10) bytes).

When a DMA transfer begins, the upper nine address bits asserted by the UNIBUS device selects a MAP register. The UBA tests whether the MAP reg. has been validated by the software, steers the transfer throught one of the 16 data paths, determines whether or not the transfer will take place in byte offset mode if a BDP has been selected, and maps the UNIBUS page address to an SBI page address.

The map registers are numbered sequentially from 0 thru 495(10). There is a 1-1 correspondence between each map register and the UNIBUS memory page address. Each map register contains the information required to effect the data transfer of the UNIBUS device addressing that page:

1. The fact that the software has loaded or not loaded the MAP register (MAP registe Valid).

2. The number of the data path to be used by the transfer and, if a BDP is used, whether it is in byte offset mode or not.

3. The SBI page to which the transfer will be mapped.

NOTE: For the rest of this description, "this UNIBUS page" refers to "the UNIBUS memory page corresponding to this MAP register".

Bit 31 Map Register Valid (MRV)
---

0 = Not Valid - initialized state
1 = Valid

The MRV is set by the software to indicate that the contents of this map register are valid. The MRV is tested each time that "this UNIBUS page" is accessed. If the bit = 1, the transfer continues. If the bit = 0, the UNIBUS transfer is aborted (non-existent memory error in the UNIBUS device) and the invalid map register bit is set in the UBA status register, providing that the map register has not been disabled by the MRD bits of the control register.

The MRV bit can be set and cleared by software.

Bits <30:26> Unused

Reserved read/write bits.

Bit 25 Byte Offset Bit (BO)

This is a read/write bit. If set, and "this UNIBUS page" is using one of the BDPs, and the transfer is to an SBI memory address, then the UBA will perform a byte offset operation on the current UNIBUS data transfer. The software can interpret this operation as increasing the physical SBI memory address, mapped from the UNIBUS address, by 1 byte. This allows word-aligned UNIBUS devices to transfer to odd byte memory addresses.

UNIBUS transfers via the DDP or to SBI I/O addresses will ignore the byte offset bit.

This bit is cleared on initialization.

Bits <24:21> Data Path Designator Bits (DPDB)
---

0 = Direct Data Path (DDP)
1-F = Buffered Data Path 1 thru F respectively.

The DPDBs are read/write bits that are set and cleared by the software to designate the data path that "this UNIBUS page" will be using.

The software can assign more than one UNIBUS transfer to the DDP. The software must ensure that no more than one active UNIBUS transfer is assigned to any BDP.

The DPDBs are cleared on initialization.

Bits <20:00> SBI Page Address [SPA<27:07>] (Page Frame Number)

The SPA bits contain the SBI page address to which "this UNIBUS page" will be mapped. These bits perform the UNIBUS to SBI page address translation. When an SBI transfer is initiated, the contents of SPA<27:07> are concatenated with UNIBUS address bits UA<08:02> to form the 28 bit SBI address.

RH780 Configuration/Status Register

CSR *Offset = 000(16)*

The configuration/status register is a read/write MBA register that contains fault status, interrupt status, adapter dependent status, and adaptor code bits.

Bit 31 SBI Parity Error (PE)

Set when an SBI parity error is detected. Cleared by power up or by the deassertion of the SBI FAULT signal. The setting of this bit will cause SBI FAULT to be asserted for one cycle.

Bit 30 Write Data Sequence (WS)

Set when no write data is received (TAG lines not equal to "Write Data" and ID lines do not contain ID of device that transmitted the command) following a write command. Cleared by power up or the deassertion of the SBI FAULT signal. The setting of this bit will cause SBI FAULT to be asserted for one cycle.

Bit 29 Unexpected Read Data (URD)

Set when read data is received and was not expected (no read command was transmitted by the MBA). Cleared by power up sequence or the deassertion of SBI FAULT. The setting of this bit will cause SBI FAULT to be asserted for one cycle.

Bit 28 Unused

Reserved for future use.

Bit 27 Multiple Transmitter Error (MT)

Set when the ID on the SBI does not agree with the ID transmitted by the MBA while the MBA is transmitting data on the SBI. Cleared by power up sequence or by the deassertion of SBI FAULT. The setting of this bit will cause the SBI FAULT signal to be asserted for one cycle starting at the normal confirmation time.

Bit 26 Transmit Fault (XMTFLT)

Set when the SBI FAULT is detected at the 2nd cycle after the MBA transmits information onto the SBI. Cleared by the power up sequence or by the deassertion of the SBI FAULT signal.

Bits <25:24> UNUSED

Read as all zeros. Reserved for future use.

Bit 23 Adapter Power Down (PD)

Set when the MBA power goes down. Cleared when power comes back up. The setting of this bit will cause an interrupt to the VAX CPU if the IE bit is set.

Bit 22 Adapter Power Up (PU)

Set when the MBA power comes up. Is cleared when the power goes down, assertion of INIT, SBI UNJAM, DCLO or by writing a one to this bit. The setting of this bit will cause an interrupt if the IE bit is set.

Bit 21 Over Temperature (OT)

Always zero.

Bits <20:08> Unused

Read as all zeros. Reserved for future use.

Bits <7:0> Adapter Code

Equal a hex 20 to signify an RH780 adapter.

RH780 Control Register

CR *Offset = 004(16)*

The MBA Control register is a read/write register that contains the control bits: Interrupt Enable, Abort, and Initialize. This register is used to put the RH780 into Maintenance Mode.

Bits <31:04> Unused

Read as all zeros. Reserved for future use.

Bit 3 Maintenance Mode (MM)

The setting of this bit will put the MBA in the maintenance mode, which will allow the diagnostic software to exercise and examine the MASSBUS operations without a MASSBUS device. When this bit is set, the MBA will block MASSBUS RUN, MASSBUS DEMAND, and assert FAIL on the MASSBUS so that all the devices on the MASSBUS will be logically detached. This bit can only be set if a data transfer is not in progress.

Bit 2 Interrupt Enable (IE)

Allows the MBA to interrupt the VAX CPU when certain conditions occur. Set by writing a one to the bit and by the power up sequence. Cleared by writing a zero to the bit or by INIT set to a one.

Bit 1 Abort Data Transfer (ABORT)

The setting of this bit will initiate the data transfer abort sequences that will stop sending of commands and addresses, and stop the byte counter. It will also negate MASSBUS RUN, assert MASSBUS EXC, wait for MASSBUS EBL, and set ABORT to a 1 at the trailing edge of MASSBUS EBL.

Set by writing a one. Cleared by writing a zero, INIT set to one, or by assertion of SBI UNJAM.

Bit 0 Initialize (INIT)

This bit is self-clearing. Always reads as zero. The setting of this bit will:

1. Clear status bits in the MBA Configuration/Status register.
2. Clear ABORT and IE in the MBA Control register.
3. Clear the MBA Status register.
4. Clear the MBA Byte Count register.
5. Clear control and status bits of the diagnostic registers.
6. Cancel all pending commands except read data pending abort data transfers.
7. Asserts MASSBUS INIT.

RH780 Status Register

SR *Offset = 008(16)*

The MBA Status register is a read/write register that contains MBA status information such as error indications, timeouts, and busy indicators. All interrupts will occur immediatedly if there isn't a data transfer in progress. If a data transfer is in progress, the interrupt will be postponed until the data transfer has terminated.

Bit 31 Data Transfer Busy (DTBUSY)

Read only. Set when a data transfer command is received. Cleared when a data transfer is aborted.

Bit 30 No Response Confirmation (NRCONF)

Set when the MBA receives a no-response confirmation for the read read command, or no-response confirmation for the write command and the write data sent to the SBI. The setting of this bit will cause retry of the command.

Cleared by writing a one to this bit or by INIT.

Bit 29 Corrected Read Data (CRD)

Set when corrected read data is received from memory. Cleared by writing a one to this bit or INIT.

Bits <28:20> Unused

Read as all zeros. Reserved for future use.

Bit 19 Programming Error (PGE)

The setting of this bit will cause an interrupt to the VAX CPU if the IE bit in the control register is set. Cleared by writing a one to this bit. Set when one or more of the following conditions exists:

1. The program tries to initiate a data transfer when the MBA is currently performing one.
2. The program tries to load MAP, VAR, or the BYTE COUNTer while the MBA is performing a data transfer operation.
3. The program tries to set MBA maintenance mode during a data transfer operation.
4. The program tries to initiate a nonacceptable data transfer command.

Bit 18 Non-existent Drive (NED)

Set when a drive fails to assert MASSBUS TRA within 1.5 microseconds after the MBA asserts MASSBUS DEM. The setting of this bit will send zero read data back to the SBI, and interrupt the VAX CPU if the IE bit is set in the MBA Control register. Cleared by writing a one to this bit location.

Bit 17 Massbus Control Parity Error (MCPE)

Set when a MASSBUS Control Bus Parity error occurs. The setting of this bit will cause an interrupt to the VAX CPU is the IE bit, in the Control Register, is set. This bit is cleared by writing a 1 to it.

Bit 16 Attention from the Massbus (ATTN)

Set when the ATTeNtion line on the MASSBUS is asserted. The setting of this bit will cause an interrupt to the VAX CPU if the IE bit, in the Control Register, is set.

The ATTN line can be asserted due to any of the following conditions:

1. An error occurs while no data transfer is taking place (asserted immediately).
2. Upon completion of a data transfer command if an error occured during the data transfer (asserted at the end of the data transfer).
3. Upon completion of a mechanical motion command (seek, recalibrate, etc.) or a search command.
4. As a result of the Medium_On_Line (MOL) bit changing states (except in the unload operation). In the dual MBA configuration, a change in state of MOL will cause the assertion of ATTN to both MBAs.

The ATTN bit in a drive can be cleared by the following actions:

1. Asserting MASSBUS INIT.
2. Writing a 1 into the Attention Summary Register (in the bit position for the appropriate drive). This clears the ATA bit; however, it does not clear the error.
3. Writing a valid command (with the GO bit asserted) into the control and status register if no error occurs. Note that clearing the ATA bit of one drive does not always cause the ATTN line to be negated, because other drives may be asserting the line.

There are 3 cases in which ATA is not reset when a command is written into the Control/Status register (with the GO bit set). These are as follows:

1. If there is a CONTROL BUS PARITY ERROR in the write.
2. If an error was previously set.
3. If an ILLEGAL Function (ILF) code is written.

Bits <15:14> Reserved

Reserved for future use. Read as zeros.

Bit 13 Data Transfer Completed (DT COMP)

Set when the data transfer is completed. Cleared by writing a one to this bit. The setting of this bit will cause an interrupt to the VAX CPU if the IE bit, in the Control register, is set.

Bit 12 Data Transfer Aborted (DTABT)

Set with the trailing edge of Massbus EBL when the data transfer has been aborted. Cleared by writing a one to this bit or by INIT. The setting of this bit will cause an interrupt to the VAX CPU if the IE bit, in the Control register, is set.

Bit 11 Data Late (DLT)

This bit is set:

1. for either a write check data transfer providing the data buffer is empty when WCLK is sent to the MASSBUS.
2. for a read data transfer providing the data buffer is full when SCLK is received from the MASSBUS.

The setting of this bit will cause the data transfer to be aborted.

Bit 10 Write Check Upper Error (WCK UP ERR)

This bit is set when a compare error is detected in the Upper byte while the MBA is performing a write check operation. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 09 Write Check Lower Error (WCK LWR ERR)

Set when a compare error is detected in the lower byte while the MBA is performing a write check operation. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 08 Missed Transfer Error (MXF)

Set when no OCC or SCLK is received within 50 microseconds after Data Transfer Busy is set. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause an interrupt to the VAX CPU is the IE bit, in the Control register, is set.

Bit 07 Massbus Exception (MBEXE)

Set when EXC is received from the MASSBUS. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 06 Massbus Data Parity Error (MDPE)

Set when a MASSBUS DATA PARITY Error is detected during a read data transfer operation. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 05 Page Frame Map Parity Error (MAPPE)

Set when a parity error is detected on the data read from the map during a data transfer. Cleared by writing a 1 to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 04 Invalid Map (INVMAP)

Set when the valid bit of the next page frame number is zero and the byte counter is not zero. Cleared by writing a one to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 03 Error Confirmation (ERR CONF)

Set when the MBA receives error confirmation for a read or write command. Cleared by writing a one to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 02 Read Data Substitute (RDS)

Set when the SBI TAG of the of the read data received from memory is Read Data Substitute (bad data). Cleared by writing a one to this bit or by INIT. the setting of this bit will cause the data transfer to be aborted.

Bit 01 Interface Sequence Timeout (IS TIMEOUT)

Set when an interface timeout occurs. An interface sequence timeout is defined as the time from when arbitration for the SBI is begun until:

1. ACK is received for a command/address transfer that specifies read.
2. ACK is received for a command/address transfer that specifies a write and the corresponding write data transfer.
3. ERR confirmation is received for any command/address transfer.

The maximum timeout is 102.4 microseconds. Cleared by writing a one to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

Bit 00 Read Data Timeout (RD TIMEOUT)

Set when a read data timeout occurs. A read data timeout is defined as the time from when an interface sequence that specifies a read command is completed to the time that the specified read data is returned to the commander. The maximum timeout is 102.4 microseconds. Cleared by writing a one to this bit or by INIT. The setting of this bit will cause the data transfer to be aborted.

RH780 VIRTUAL ADDRESS Register

VAR *Offset = 0C(16)*

Before a data transfer is initiated, the program must load an initial virtual address (pointing to the first byte to be transfered) into this register.

Bits <31:17> Reserved

Not used. Reserved for future use. Read as zeros.

Bits <16:09> Map Pointer

Selects one of the 256 MAP registers.

Bits <08:00> Physical Page Byte Address

Byte offset into the current page.

The contents of the selected MAP register and the value of Bits <08:00> are used to assemble a physical SBI address to be sent to memory. Bits <08:00> indicate the byte offset into the page of the current data byte. The virtual address register may not be written into during a data transfer. An attempt to do so will set PGE, but the virtual address register will not be modified and the data transfer will continue.

The MBA virtual address register is incremented by eitht after every memory read or write and will not point to the next byte to be transferred if the transfer does not end on a quadword boundary (it will point eight bytes ahead). When a write check error occurs, the virtual address register will not point to the failing data in memory due to the preloading of the silo data buffer. The virtual address of the bad data may be found by determining the number of bytes actually transferred on the MASSBUS (the difference between bits <31:16> of the Byte Count Register and their initial value) and adding that difference to the initial virtual address.

RH780 BYTE COUNT Register

BCR *Offset = 10(16)*

The program loads the 2's complement of the number of bytes for the data transfer to bits <15:00> of this register. The RH780 hardware will load these 16 bits into bits <31:16> and bits <15:00> of the Byte Count register. Bits <31:16> serve as the byte counter for the number of bytes transferred through the MASSBUS and bits <15:00> serve as the byte counter for the number of bytes transferred through the SBI interface. The starting byte count with 16 bits of zeros is the maximum number of bytes of a data transfer. The byte count register may not be modified during a data transfer. An attempt to do so will be ignored and the PGE bit will be set.

Bits <31:16> Massbus Byte Counter

Data written to bits <15:00> is duplicated in these bits. This counter is used to count the number of bytes transferred across the MASSBUS.

These bits are read only.

Bits <15:00> SBI Byte Counter

These bits form the SBI Byte counter. The purpose of this counter is to count the number of bytes transferred across the SBI and to overflow to zero to signal the completion of the transfer.

This counter is loaded, by the program, with the 2's complement of the number of bytes to be transferred. The RH780 hardware duplicates what is written to these bits, by the program, into bits <31:16>.

This counter is read/write.

RH780 DIAGNOSTIC Register

DR *Offset = 14(16)*

The diagnostic register is a read/write register that contains MBA diagnostic information. This register allows diagnostics to be run without any drives on the MASSBUS. The diagnostic register may not be written unless the MBA is in the maintenance mode. An attempt to write the diagnostic register when not in the maintenance mode will be ignored. Caution should be exercised when reading this register in the maintenance mode. The data path used to read bits <07:00> may inject invalid data into the silo if the MBA has just read data from memory. It is advisable to wait 20 microseconds from the initiation of a transfer or the deassertion of SCLK before reading or modifying this register.

Bit 31 IMDPG

Invert MASSBUS Data Parity generator.

Bit 30 IMCPG

Invert MASSBUS Control Parity generator.

Bit 29 IMAPP

Invert MAP Parity.

Bit 28 BLKSCOM

Block sending command to the SBI. During a data transfer, the setting of this bit will eventually cause a DLT bit set and a DT ABORT.

Bit 27 SIMSCLK

Simulate MASSBUS SCLK. When the MM bit is set, writing a 1 to this bit will simulate the assertion of MASSBUS SCLK; writing a 0 to this bit will simulate the deassertion of MASSBUS SCLK.

Bit 26 SIMEBL

Simulate MASSBUS EBL. When the MM bit is set, writing a 1 to this bit will simulate the assertion of MASSBUS EBL; writing a 0 to this bit will simulate the deassertion of MASSBUS EBL.

Bit 25 SIMOCC

Simulate MASSBUS OCC. When the MM bit is set, writing a 1 to this bit will simulate the assertion of MASSBUS OCC; writing a 0 to this bit will simulate the deassertion of MASSBUS OCC.

Bit 24 SIMATTN

Simulate MASSBUS ATTN. When the MM bit is set, writing a 1 to this bit will simulate the assertion of MASSBUS ATTN; writing a 0 to this bit will simulate the deassertion of MASSBUS ATTN.

Bit 23 MPIB SEL

Maintenance MASSBUS Data Input Buffer Select. When this bit is set to a 1, the upper eight bits (B<15:08>) of the MDIB will be sent out from bits <07:00> of the Diagnostic register if the diagnostic register is read. When the bit is 0, the lower eight bits (B<15:08>) of the MDIB will be sent out from bits <07:00> of the Diagnostic register if it is read.

Bits <22:21> Maint only

Read/write with no effect. Used to test the writability of these bits.

Bit 20 MFAIL

MASSBUS FAIL (read only). MASSBUS FAIL is asserted when the MM bit is set.

Bit 19 MRUN

MASSBUS RUN (read only).

Bit 18 MWCLK

MASSBUS WCLK (read only).

Bit 17 MEXC

MASSBUS EXC (read only).

Bit 16 MCTOD

MASSBUS CTOD (read only).

Bits <15:13> MDS

MASSBUS Device Select (read only).

Bits <12:08> MRS

MASSBUS Register Select (read only).

Bits <07:00> U/L MDIB

Maintenance Upper/Lower MDIB.

RH780 SELECTED MAP Register

SMR *Offset = 18*

This register is read only and is valid only when DT BUSY is set. Reading this registers gives you the contents of the MAP register pointed to by bits <16:09> of the Virtual Address register.

The bit assignments for the MAP registers are as follows:

Bit 31 Valid

When set, indicates that the contents of bits <20:00> are valid.

Bits <30:21> Not used

Not used. Reserved for future use. Read as zeros.

Bits <20:00> Page Frame Number

Contains the Physical page frame number. These bits are used to calculate the physical memory address to/from which the transfer is to take place. These bits actually select only the PHYSICAL SBI MEMORY PAGE that the transfer will be referencing.

Bit 9 = 1 and Bit 8 = 0.

The RH780 contains 256 MAP registers, each of which may be selected by Virtual Address bits <07:00>. MAP registers can only be written when there is no data transfer operation in progress. A write to a MAP reg. while a data transfer is in progress will be ignored and cause the setting of PGE and will cause an interrupt to the VAX CPU at the end of the transfer if the IE bit is set.

RH780 COMMAND/ADDRESS Register

CAR *Offset = 1A(16)*

This register is read only.

Valid only when DT BUSY is set.

It contains the value of bits <31:00> of the SBI during the COMMAND/ADDRESS part of the RH780's next data transfer.

MS780-E Configuration Register "A"

CNFG-A *Offset = 000 (16)*

Bit <31>, SBI Parity Fault

A parity error was detected on the SBI.

Bit <30>, SBI Write Sequence Fault

Failure of a WRITE command to be followed immediately (in the next sequential SBI cycle) by a Write Data Format.

Bit <29>, NOT USED

This bit not assigned.

Bit <28>, SBI Interlock Sequence Fault

An INTERLOCK WRITE command was not proceeded by an INTERLOCK READ command.

Bit <27>, SBI Multiple Transmitter Fault

The "received ID" (received at SBI T3 time) is not the same as the "transmitted ID" (transmitted at SBI T0 time). The transmitted ID is checked by comparing it with the ID that is read back at SBI T3 time of the same cycle.

Bit <26>, Transmit Fault

This memory was the transmitter when the SBI error occured.

Bits <25:24>, NOT USED

These bits are not assigned.

Bit <23>, Power Down

A power-down sequence is underway.

Bit <22>, Power Up

A power-up sequence is underway.

Bit <21>, NOT USED

This bit is not assigned.

Bit <20>, Error Summary

Set if any of the following bits are set:

1. Internal Parity Errors
   a. Register-A Bit <19>
   b. Register-A Bit <18>
   c. Register-C Bit <07>
2. Misconfigure Warning
   a. Register-A Bit <17>
   b. Register-A Bit <16>
   c. Register-A Bit <15>
3. Error Log Request
   a. Register-C Bit <28>

Bit <19>, CNTR 1 Par Err

Read data from the UPPER controller to interface had a parity error. Bad data is sent on the SBI, with corrected parity, and the RDS mask (multiple bit error).

Bit <18>, CNTR 0 Par Err

Read data from the LOWER controller to interface had a parity error. Bad data is sent on the SBI, with corrected parity, and the RDS mask (multiple bit error).

Bit <17>, Misconfigured

In INTERNAL Interleave mode, set by an unequal number of arrays with each controller.

Bit <16>, CNTR 1 MISCNFG

Misconfiguration in the UPPER Controller's memory. Caused by one of the following:

1. Illegal array arrangement
2. No Arrays
3. No Controller

Bit <15>, CNTR 0 MISCNFG

Misconfiguration in the LOWER Controller's memory. Caused by one of the following:

1. Illegal array arrangement
2. No Arrays
3. No Controller

Bits <14:09>, Memory Size

Memory system capacity from 1 MegaByte (000000) to 64 MegaByte (111111). Count is in Binary.

Bit <08>, INTLV Mode Write Enable

Permits a WRITE to bits <02:00> which establishes the INTERLEAVE MODE.

Bits <07:05>, Adapter Code

Fixed set of bits identifying the subsystem (NEXUS) as an MS780-E memory subsystem. Bits read as "011" (from bit <07> to bit <05>).

Bits <04:03>, RAM type

Identifies the size of the RAMs on the arrays as follows:

Bits 4	3	Description
0	0	Misconfigured, No array Boards in backplane
0	1	64K RAMs (1 MegaByte Arrays)
1	0	256K RAMs (4 MegaByte Arrays)
1	1	Misconfigured, both array types in backplane

Bits <02:00>, Interleave Mode

Identifies the Interleave Mode as follows:

Bits 2	1	0	Description
0	0	0	Non-interleaved LOWER controller
0	1	0	Non-interleaved UPPER controller
0	0	1	Externally interleaved LOWER controller
0	1	1	Externally interleaved UPPER controller
1	0	0	Internally interleaved

Bits <02:01> are set, on Power-Up, to interleave mode according to the hardware configuration, i.e., appropriate to the number and position of memory controllers present. Bit <00> must be written by the Software.

MS780-E Configuration Register "B"

CNFG-B *Offset = 004 (16)*

Bits <31:28>, Not Used

These bits are not assigned.

Bits <27:19>, START ADDR

Specifies the starting address of the memory subsystem in 1 MegaByte increments.

Bits <18:15>, Not Used

These bits are not assigned.

Bit <14>, START ADR WR EN

Enables writing to bits <27:19>

Bits <13:12>, INIT and BATTERY Status

Indicates if memory is coming up from a COLD Start and is initializing the memory, or if valid data is preserved in the memory arrays as follows:

Bit 13	Bit 12	Description
0	0	Initialization in progress (memory written with 0's and BUSY is being sent to any SBI commands that may be referencing this memory).
0	1	Memory contains Valid Data.
1	0	Invalid Combination
1	1	Initialization completed, NO VALID DATA in memory.

Bit <11>, Force DBUS Par Error

READ DATA from controllers to the SBI interface will have an error and a read data substitute will be forced.

Bits <10:09>, Diagnostic Mode Select

There are three diagnostic modes that exercise various controller functions and four data paths and their latches as follows:

Bit 10	Bit 09	Description
0	0	Normal Operation
0	1	Verifies check bit generation logic and controller data path.
1	0	Verifies the ECC logic.
1	1	Verifies the check bit MOS RAMs.

Bit <08>, Refresh Lock

Prevents the memory controller from executing READ/WRITE cycles.

Bit <07>, Not Used

This bit is not assigned.

Bits <06:00>, Diagnostic ECC bits

Loaded with the substitute ECC bits in conjunction with the diagnostic modes.

MS780-E Configuration Registers "C & D"

CNFG-C and CNFG-D *Offset = 008(16) and 00C(16)*

Bit <31>, Force Microsequencer Parity Error

Causes the wrong parity across the 56 PROM bits of the microsequencer data field. Sets bit <07>.

Bit <30>, Inhibit CRD

Prevents single-bit errors from sending CRD with the read data on the SBI. Error log requests (bit<28>) and CRD error bit <09> will be set by a single-bit error.

Bit <29>, High Error Rate

Indicates a second error has been detected before the 1st was cleared.

Bit <28>, Error Log Request

Notification of an error on a memory read.

Bits <27:11>, Error Address

Specifies the memory address to the page level of the error. The address format specified is as follows (VALID ONLY IF Bit <28>=1):

<27> - Controller Select
<26:24> - Array Select
<23:22> - Array Bank Select
<19:11> - RAM page address (64K RAMs)
<21:11> - RAM page address (256K RAMs)

Bit <10>, RDS Flag

Multiple-bit error detected.

Bit <09>, CRD Flag

Single-Bit error detected and corrected.

Bit <08>, Not Used

This bit is not assigned.

Bit <07>, Microsequencer Parity Error

A parity error was detected across the 56-bit PROM data word.

Bit <06:00>, Error Syndrome/ CHECK BITS

Stores 7-bit error syndrome or 7 check bits, depending on the diagnostic mode set in Configuration Register-B.

MS780-E Configuration Registers "E & F"

REG-E & REG-F Offset = 010(16) and 014(16)

Registers E and F are the two data latches on the SBI interface module (designated as data latches 1 and 2, respectively). After writing to either or both of these registers, they may be read, causing the data written to be sent back on the SBI through the SBI transceivers. Thus, these registers allow a data wrap-around within the SBI interface module only. No memory controllers have to be installed to execute this data wrap-around process.