Typical z/OS problems are classified by the following symptoms:
- **Abend** – an error or abnormal end of a program or job.
- **Wait or Hang** – a coded wait state is loaded or the system or a job appears hung or does not complete.
- **Loop** – the system or program executes infinitely typically using large or higher amounts of processor resource.
- **Incorrout** – there is incorrect or missing output from a program or job.
- **Performance** – processing is using too much system resource and impacting other parts or users of the system, or processes are taking too long.
- **Message** – an error is reported through a message to the operator or in a log.
  - When an error occurs, z/OS provides various forms of diagnosis information that contains symptoms.
  - These symptoms can help you with diagnosis and be used in problem source identification (**PSI**).
    > PSI is the determination of what caused the error based on answers to these questions:
    1. Why was an abend issued?
    2. What caused storage to be exhausted?
    3. Why is the job hung?
    4. What is causing the job to loop?

PSI is useful even if the root cause of the problem is not identified.
During the process, information and symptoms are gathered to check for a known problem or report new problems.
To ease and expedite problem identification, it is important to provide all the background information available.
This includes information about:
- Hardware involved
- System and application software levels
- External symptoms
- Problem impact
- Diagnostic data produced

NOTE: By providing sufficient information during the first call to IBM or the individual software vendor, you might avoid having to re-create the problem.

To diagnose a z/OS problem, follow these steps:

1. When the problem occurs, gather all the available diagnosis information for problem. This might also include your installation internal problem report describing external symptoms, what might have triggered the problem, and what was done to recover, including the following types of diagnostic information such as:
Dumps, Traces, Error messages, SYS1.LOGREC entries, External symptoms, Hardware devices, and processor models.
- It is important to gather the external symptoms and know the impact to the system or sysplex to define the scope of the problem.
- There can be many symptoms.

For example: Shortly after JOB A started, a dump was produced for an ABEND0*xx*, the system went into a WAIT0*xx*, and was partitioned from the sysplex.
1.1 Start with diagnosis of the ABEND0*xx*, which appears to be the trigger, but also understand the cause of the WAIT0*xx* and why the job failure resulted in a system outage. It is important to check for known problems for both symptoms.
1.2. Next, gather all the diagnosis data available from the time frame the problem or failure occurred.
To identify a system problem, look at the diagnostic data such as:
- See any external symptoms and the initial problem report. Look for indications, which can include:
Messages, job hang, system hang, high processor usage, incorrect output, dumps produced, system slowdown, or jobs not starting.
- Potentially an SVC dumps may be produced as indicated by these messages:

IEA794I
IEA794I SVC DUMP HAS CAPTURED: DUMPID=dumpid REQUESTED BY JOB (*MASTER*) DUMP TITLE=dump-title

IEA911E
IEA911E {COMPLETE|PARTIAL} DUMP ON SYS1.DUMPnn DUMPid=dumpid REQUESTED BY JOB (jobname) FOR ASIDS(id,id,...) [REMOTE DUMPS REQUESTED [REMOTE DUMP FOR SYSNAME: sysname] INCIDENT TOKEN:incident-token [SDRSN = vvvvvvvv wwwwwwww xxxxxxxx zzzzzzzz] [reason-text] [ERRORID = SEQyyyyyy CPUzz ASIDasid TIMEhh.mm.ss.f] [TSOID = tsoid] [ID = uuuuuuuuuu]

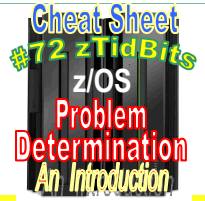
IEA611I
IEA611I {COMPLETE|PARTIAL} DUMP ON dsname... *text*

- SYS1.LOGREC data set, which is a repository for information about hardware and system-level software errors.
- Logs from the time frame the problem occurred. This can include SYSLOG, OPERLOG, job log(s), and others. (see #59 zTidBits Logs & the Logger)
- Traces associated with the problem.
- **NOTE:** After a problem has been isolated to a particular component, query using the **TRACE** command to see if detailed component traces or GTF was active at the time. **For example:** If the error is announced by ISGxxxx messages, then check for SYSGRS CTRACE.

The message prefix (three or more characters) will determine the component owner.
The problem indicator contains examples of indicators. Some problems might need to be investigated using more than one diagnostic procedure or tool to find the cause. If there are several indicators, look for the earliest problem that caused the other problems.
Example: There are several abends and a wait state, look for the earliest abend code and begin diagnosis there.

Problem type	Indicator	System action	System programmer action
ABEND	SVC dump taken and a record of the error (in Logrec)	Produces dump	Review dump to determine if further diagnosis is required
	Message received indicating a system or user abend	Produces record of error	Review response of system message to determine the impact of the abend on the installation.
	An ABEND dump is produced	Continue processing	Review the dump to determine if further diagnosis
	• SVC dump produced • Error recorded to SYS1.LOGREC • Error message issued • SYSUDUMP, SYSABEND or CEEDUMP produced	System actions are the same as the indicators listed above. **Note:** The system might also initiate recovery actions. See SYSLOG and component trace to determine what these recovery action were. Some recovery actions can cause data to be lost, requiring the installation to resubmit jobs or transactions.	1. Use IPCS to do problem diagnosis on the dump. 2. Look up abend and reason code recorded for more information about error. 3. Look up the message to gather more information about cause of the error and the system programmer action to correct. 4. Review the dump to do problem diagnosis.

NOTE: While you are diagnosing system trouble, you should collect data about the problem:
- ✓ What was the **abend code**?
- ✓ What did the **registers** and **PSW** contain at the time of error?
- ✓ What is the **failing module** or CSECT?
- ✓ What **components** or **products** were involved with the error?



IBM z/OS Management Facility (z/OSMF) provides a framework for managing various aspects of a z/OS System through a Web browser interface. By streamlining some traditional tasks and automating others, z/OSMF can help to simplify the day-to-day operations and administration of a z/OS system. More than just a graphical user interface, z/OSMF is "intelligent".

Problem type	Indicator	System action	System programmer action
Job Hang / Wait or Loop	Job does not end, no further output is produced, and the job can or cannot be CANCEL'ed or FORCE'd	No response	Use the **DUMP** command to obtain an SVC dump of the hung job. If the **DUMP** is not successful, consider taking a stand-alone dump.
System Hang or Wait	Disabled wait indicated on the HMC and wait state message issued	The system issues a wait state message and loads a disable wait state PSW. The system might load the following into the PSW: X'070E0000 00000000'	Take a stand-alone dump.
	Many jobs are hung in the system	Resource contention	Enter the **DISPLAY GRS,C** command to check for ENQ resource and latch contention and take a dump of the holder of the resource including *SDATA=GRSQ*. **Note:** Use the **DISPLAY GRS,ANALYZE** command to aid in the discovery of blockers in the system.
	No response to system or subsystem commands entered	No response	Partition the system from the sysplex and take a stand-alone dump.
Loop	High processor resource being consumed locking out other work; Excessive spin detected with IEE178I or ABEND071 issued, or both.	ABEND071 issued in an attempt to stop the looping program	Use an online monitor, such as RMF™ or IBM OMEGAMON® z/OS Management Console, to determine whether the problem originates from a high priority job in normal processing or from a problem.
	A job is using a high percentage of central processor storage	Processing degrades	Use an online monitor, such as RMF, to determine whether the problem originates from a high priority job in normal processing or from a problem.
Enabled Wait or Performance degradation†	System processing slows.	Processing degrades	Use an online monitor, such as RMF, to determine where the problem originates.
	There is a series of WAIT messages followed by a burst of activity	Processing continues	Use an online monitor, such as RMF, to determine where the bottleneck is occurring.
Output problem†	Job output is missing or is incorrect.	Processing continues	Use GTF or SLIP to trace input and output.

To help eliminate soft failures, z/OS has developed Predictive Failure Analysis (**PFA**). PFA is designed to predict if a soft failure will occur sometime in the future and to identify the cause while keeping the base operating system components stateless. PFA is intended to detect abnormal behavior early enough to allow you to correct the problem before it affects your business.

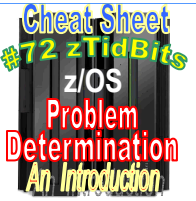
2. After the problem type is identified, see diagnosis procedures to identify the source and extract symptoms.
- Abends have an associated system completion code to describe the error and most have a reason code to further explain the problem. These codes can be found by searching:

2.1. LookAt at http://www.ibm.com/systems/z/os/zos/bkserv/lookat/.
2.2. z/OS MVS System Codes (i.e. SA22-7631- note: Depending on msg. prefix; there are several volumes)
2.2. The documentation for the particular application that failed. **For example:**
- For Language Environment completion codes, see *z/OS Language Environment Run-Time Messages* (SA22-7566-11). The messages also contain a symbolic feedback code, which represents the first 8 bytes of a 12-byte condition token. You can think of the symbolic feedback code as the nickname for a condition. As such, the symbolic feedback code can be used in user- written condition handlers to screen for a given condition, even if it occurs at different locations in an application.
  > The messages in this category contain alphabetic suffixes that have the following meaning:
  I Informational message **W** Warning message **E** Error message **S** Severe error message **C** Critical error message
- For RMF completion codes, see *z/OS RMF Messages and Codes* (SC33-7993).
  > Lists all messages issued by the RMF control session, Monitor I, II and III gatherer sessions, Monitor II and III reporter sessions, and the Postprocessor. The messages are sorted by the hexadecimal message number which follows the ERB-prefix. **ERB100I.**

continued

Interactive Program Control System (**IPCS**) is a powerful diagnostic tool in the MVS system that aids the diagnosis of software failures. IPCS provides formatting and analysis support for dumps and traces produced by MVS, other program products, and applications that run on MVS.

† *Will not be covered due to the extensive amount of diagnostic procedures and information. It is mentioned for completeness of problem types.*



Dumps

SVC dump: An SVC dump provides a representation of the virtual storage for the system when an error occurs. Typically, a system component requests the dump from a recovery routine when an unexpected error occurs. However, an authorized program or the operator can also request an SVC dump when diagnostic dump data is needed to solve a problem.
NOTE: Complete details are found in SVC dump in *z/OS MVS Diagnosis: Tools and Service Aids*.

Transaction dump: A transaction dump provides a representation of the virtual storage for an address space when an error occurs. Typically, an application requests the dump from a recovery routine when an unexpected error occurs. Complete details are found in transaction dump in *z/OS MVS Diagnosis: Tools and Service Aids*

Abend dump: An ABEND dump shows the virtual storage predominately for an unauthorized program. To produce a dump when one is requested for an error, a JCL DD statement of SYSUDUMP, SYSABEND or SYSMDUMP must be included in the input job stream. See *z/OS MVS JCL Reference* for more information. An operator can also request an ABEND dump while ending a program, an address space, or canceling a job. There are three types of abend dumps:

1. SYSMDUMP – Is an unformatted dump that requires IPCS to view and format. Unformatted dumping is sometimes more efficient because only the storage requested is written to the data set, which means the application can capture diagnostic data and be brought back online faster.
2. SYSABEND – The largest of the ABEND dumps, is a pre-formatted dump containing a summary dump for the failing program plus many other areas useful for analyzing processing in the failing program.
3. SYSUDUMP – The smallest of the ABEND dumps, containing data and areas only about the failing program.

NOTE Complete details are found in Abend dump in *z/OS MVS Diagnosis: Tools and Service Aids*.

SNAP dump: A SNAP dump shows virtual storage areas that a program, while running, requests the system to dump. A SNAP dump, therefore, is written while a program runs, rather than during abnormal end. The program can ask for a dump of as little as a one byte field to as much as all of the storage assigned to the current job step. The program can also ask for some system data in the dump. A SNAP dump is especially useful when testing a program.
Complete details are found in SNAP dump in *z/OS MVS Diagnosis: Tools and Service Aids*

Stand-Alone dump: Used to collect data for individual work units on a system or a subset of components on a system. A stand-alone dump is used to collect diagnostic information about the **entire** system. Stand-alone dumps are not produced by z/OS but by an either the IPCS SADMP dump data set utility or the AMDSADDD REXX utility. After a stand-alone dump is taken, because the system cannot resume usual processing, the IPL is of the stand-alone dump instead of z/OS. The stand-alone dump program produces a stand-alone dump of storage that is occupied by either:

* A system that is stopped. For example, your installation has a wait state with no processing, so you must capture a stand-alone dump to diagnosis it.
* A stand-alone dump program that failed. Either the stand-alone dump program dumped itself — **a self-dump** —, or the operator loaded another stand-alone dump program to dump the failed stand-alone dump program.

•The stand-alone dump program and the stand-alone dump together form what is known as the stand-alone dump service aid. The term stand-alone means that the dump is performed separately from usual system operations and does not require the system to be in a condition for normal operation. It is essential to perform a store status before taking a stand-alone dump because the program gets loaded over storage that might be needed in the dump.

IBM Omegamon for z/OS Management Console is a monitoring product that includes an interface for z/OS management and is designed to help eliminate, and simplify many z/OS management tasks. The OMEGAMON z/OS Management Console helps deliver real-time, check information provided by the IBM Health Checker for z/OS, and configuration status information for z/OS systems and sysplex resources.
NOTE: For more information, see IBM OMEGAMON for z/OS Management Console at www.ibm.com/systems/z/os/zos/zmc/ .

AMATERSE for FTP In z/OS V1R9 and above, use AMATERSE to compress and extract problem documentation you send to IBM. There are differences between AMATERSE and the TRSMAIN utility. AMATERSE is the preferred, supported program.
NOTE: AMATERSE is an application that prepares diagnostic materials, such as z/OS dumps and traces, for transmission to IBM and vendor sites. When the materials arrive, AMATERSE also provides a means to create similar data sets to support diagnosis of problems.
If you have previously used the TRSMAIN utility (see http://techsupport.services.ibm.com/390/trsmain.html), you will find that the following changes have been made to prepare AMATERSE for formal inclusion in z/OS:

⚠ AMATERSE is used as the preferred application program name rather than TRSMAIN. TRSMAIN is shipped as an alias entry point to AMATERSE.
NOTE: For complete details, see the AMATERSE topic in *z/OS MVS Diagnosis: Tools and Service Aids* (GA22-7589-16).

Runtime Diagnostics with z/OS v1.12 is an MVS subsystem (component name HZR) designed to help you analyze a system that has potential soft failures.
• In contrast to catastrophic failures, a soft failure is defined as the combination of typical and abnormal behavior that causes the software to withhold a requested service.
- Soft failures are often difficult or impossible to detect and can slowly lead to the degradation of the solution that is using z/OS.
- Runtime Diagnostics performs many of the same tasks you might typically perform when looking for a failure, such as: Reviewing critical messages in the log, analyzing contention, examining address spaces with high central processing unit (CPU) usage, looking for an address space that might be in a loop and evaluating local lock conditions.
- In many cases, when Runtime Diagnostics finds a critical message, it performs additional analysis based on the job name in the message or other information in the message text.
- Runtime Diagnostics is set up as a subsystem that runs as a started task from the console (for example, START HZR). It searches for certain messages and message combinations in the operations log (OPERLOG) stream and attempts to identify other system symptoms with minimal dependencies on other system services.
(see #59 zTidBits Logs & Logger)

Sample Runtime Diagnostics Report

```
SY1  HZR0200I RUNTIME DIAGNOSTICS RESULT 805
SUMMARY: QUALIFIED SUCCESS - SOME PROCESSING FAILED
 REQ: 001 TARGET SYSTEM: SY1      HOME: SY1      2010/01/09 - 15:00:27
 INTERVAL:  60 MINUTES
 EVENTS:
  FOUND: 02 - PRIORITIES: HIGH=02  MED=00  LOW=00
  TYPES: HIGHCPU=01
  TYPES: LOOP=01
  PROCESSING FAILURES:
   OPERLOG....IXGCONN REQ=CONNECT ERROR.....RC=00000008 RS=0000080B
-------------------------------------------------------------------
EVENT 01: HIGH - HIGHCPU       - SYSTEM: SY1      2010/01/09 - 15:00:28
ASID CPU RATE: 99%    ASID: 002D  JOBNAME: IBMUSERX
STEPNAME: STEP1    PROCSTEP:          JOBID: JOB00044 USERID: IBMUSER
JOBSTART: 2010/01/09 - 15:00:13
  ERROR:  ADDRESS SPACE USING EXCESSIVE CPU TIME. IT MAY BE LOOPING.
 ACTION:  USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
-------------------------------------------------------------------
EVENT 02: HIGH - LOOP          - SYSTEM: SY1      2010/01/09 - 15:00:23
ASID: 002D JOBNAME: IBMUSERX TCB: 004E6868
STEPNAME: STEP1    PROCSTEP:          JOBID: JOB00044 USERID: IBMUSER
JOBSTART: 2010/01/09 - 15:00:13
  ERROR:  ADDRESS SPACE APPEARS TO BE IN A LOOP.
 ACTION:  USE YOUR SOFTWARE MONITORS TO INVESTIGATE THE ASID.
```

Runtime Diagnostics reports when some processing fails (the inability to complete processing for one or more events) as **QUALIFIED SUCCESS** in the **SUMMARY:** portion of the report

Note: The message explains what part of the processing was unsuccessful under the field: **PROCESSING FAILURES:**.

In this case, although Runtime Diagnostics was unable to connect to OPERLOG to examine messages, it continues to find other soft failures (LOOP and HIGHCPU).

An abend is classified as follows:

• Software-detected:
– A system code in the form of three hexadecimal digits, possibly with a four byte reason code. For example, ABEND075. A system abend code is issued with the ABEND or CALLRTM macros used to terminate a task or address space when a system service or function detects an error.
– A user code in the form of a four decimal digits, possibly with a four byte reason code. For example, ABENDU4094. A user code is issued using the ABEND macro to terminate a task or the entire job step. When the highest-level task in a job step ends abnormally, all related tasks or subtasks also terminate. When a subtask terminates, only work running on behalf of the subtask is affected, unless STEP=YES is specified.
• Hardware-detected:
Hardware might present a program interrupt or machine check on the execution of an instruction. The operating system detects these hardware problems and presents them as an abend.

Example: An instruction in an application running in storage key 7 references storage assigned to key 0. The difference in storage key causes a protection exception. This exception results in hardware presenting a program interruption code of 0004 to the operating system, which is externalized as ABEND0C4. (See: **SG24-6366** Introduction To The New Mainframe: z/OS Basics for explanation of PSW and storage keys)

A system hang or wait can occur gradually as a resource contention problem or abruptly when a disabled wait state is loaded for a critical software error.
Externally, the following list of symptoms might be noticed:
✓ A disabled coded wait state is loaded
✓ A hang during IPL or system initialization
✓ The consoles can be locked
✓ There can be contention for system resources
✓ The system code can be looping.
• When there is a system failure or outage, a stand-alone dump must be taken for problem diagnosis. OPERLOG, SYSLOG, and EREP reports from the time frame of the system outage are also important.

Symptoms of a wait or hang: The system enters a wait or the entire system hangs.
•The terms hang and wait are used synonymously in these procedures.
- Some symptoms of a hang:
✓ No response occurs on the user's or system operator's console.
✓ No communication with the system through the console occur.
✓ No response from subsystems (TSO/E, CICS, IMS™, DB2®, and others) occur.
✓ The system does not issue or receive messages on the console.
✓ A series of messages that indicate waits followed by bursts of activity.
✓ A message indicating a wait appears on the system console.
✓ The program status word (PSW) contains X'070E0000 00000000'.
✓ The job entry subsystem does not respond to any commands.
For example, in a JES2 system, enter a **$DI1** command and JES2 does not respond.

The wait state code appears in the program status word (PSW) when the operating system enters a wait state. Wait state codes are found in the 12 low-order bits of the PSW when the PSW wait bit (bit 14) is set to one.
A disabled wait condition can be analyzed by checking the PSW at the time of the error. If bits 6 and 7 are zero and bit 14 contains a 1, there is a disabled wait. The wait state code is in byte 7, with the reason code in byte 5.
Example: Determining the wait state code:
In the following PSW, the wait state code is X'014' and the reason code is zero.
PSW=000E0000 00000014
In another example, wait state code is X'064' and the reason code is X'09'.
PSW=000A0000 00090064
In z/Architecture® mode, the PSW would look like:
PSW=0002000 00000000 00000000 00090064

NOTE: SLIP trap causes MVS to take a specified set of actions when a specified event occurs.

• There are two types of wait states: enabled and disabled.

Enabled wait
- The system stops processing without issuing a wait state code when the dispatcher did not find any work to be dispatched.
- A special type of enabled wait is called a **no work wait** or a **dummy wait**.
An indication of a dummy wait or no work wait is a PSW of X'070E0000 00000000' and GPRs containing all zeroes.
Diagnosis is required for this type of wait only when the system does not resume processing.
- The most common causes of an enabled wait are that the system is waiting for:
1. Work – the system has no active jobs to process or all active jobs are swapped out.
2. Action – an operator reply or other action.
3. Missing interrupts – the system is waiting for a critical device, which is busy, not ready, reserved by another system, or has a mount pending. If the system residence (SYSRES) or paging (PAGE) volumes have missing interrupts, the operator may not get a message.
4. System resource – work is waiting for a resource, which can be a lock, queue, input/output (I/O) device, page, or device allocation.

Disabled wait with a wait state code
- The system issues a wait state code and stops. The operator can see the wait state code on the system console. This wait is called a **coded wait state** or a **disabled wait**. There are two types of disabled wait state codes:
1. **restartable wait state**
You can restart the system.
A restartable wait is one of the following:
a. An attempt by the operating system to communicate with the operator. When the system cannot send a message to a console, the system can use a restartable wait state to contact the operator and obtain a response.
b. A way to preempt processing. For a SLIP trap with an action of wait, the system will issue a message, then enter a restartable wait.
c. A symptom of another problem.
2. **non-restartable**
You cannot restart the system. After capturing a stand-alone dump, you must reIPL the system.

Diagnosing a system hang you need to know how to use IPCS and you need access to the following types of information:
✓ Stand-alone dump
✓ EREP (Environmental record editing and printing program) report of SYS1.LOGREC
✓ OPERLOG or SYSLOG
✓ The level of z/OS operating system. Use the IPCS **CBFORMAT CVT** command to find the level of the z/OS.

A loop is a repetitive set of instructions being performed by a job or unit of work.
• A job or function that is looping can appear to be hung or can use a high amount of CP resource and lock out other work from getting service.
The three types of loops are:
1. **Disabled loop** is repetitive execution, usually in system level code, with the IO and EXT type interrupts prevented with a PSW mask of X'x4' in the high order byte of the PSW. A disabled loop is bound to one CP in the system. If in a multi-processor environment and resources are held, a spin loop is detected. If on a uniprocessor, a disabled loop will result in a system outage.
2. **Enabled loop** occurs under a unit of work (TCB or SRB) that is executing on behalf of a job or function. It is executing with a PSW that is enabled for I/O and external interrupts with a mask of X'x7' in the the high order byte. A unit of work that is looping enabled, is interrupted periodically for IO, EXT or CLKC type interrupts, which are traced in the system trace table.
3. **Spin loop** is a timed disabled loop in system code controlled by the installation with specifications in the EXSPATxx (excessive spin condition actions) parmlib member. The system can spin or loop disabled waiting for a resource, such as a lock, to be released by another CP in a multi-processing environment.

Symptoms of a loop:
Disabled loop symptoms are easier to identify than enabled loops. Symptoms include:
✓ System CP usage increases for unexplained reasons.
✓ There is no communication with the system through the master and alternate consoles.
✓ Console communications are locked out. To check for communication with the console, enter **DISPLAY T** command and the system will not respond.
Enabled loop symptoms allow some or all interrupts. The loops are usually caused by an error in an application program. All or most of the loop is in code running in problem state, but the loop can include system code if any instructions in the loop request system services. An enabled loop can run on more than one central processor. The loop will uselessly consume resources and might take over all system operation.
Additional symptoms include:
✓ A bottleneck, indicating that the system slows down periodically, thus creating a performance problem.
✓ A job stays in the system for a long time without changing status or ending.
✓ Low priority work slows down or stops running (a result of a higher priority enabled loop).
✓ System CP usage increases for unexplained reasons or CP usage of an address space is much higher than normal.
Spin loop symptoms occurs when one processor in a multiprocessor environment is unable to communicate with another processor or requires a resource currently held by another processor. The processor that has attempted communication is the *detecting* or *spinning* processor. The processor that has failed to respond is the *failing* processor.
•The detecting processor continuously attempts its communication with the failing processor until either:
- It is successful.
- A specified time interval has passed.
•When the communication is not successful within an interval, an *excessive spin loop time out* exists.
- The detecting processor then initiates recovery processing for the condition.
• z/OS processing for excessive spin-loop conditions can provide recovery without any operator prompts or actions required. The following recovery actions can be defaulted to or specified in the EXSPATxx parmlib member **:**
- **SPIN** Continue spinning for another interval to allow the event to complete
- **ABEND** End the current unit of work on the failing processor but allow the recovery routines to retry
- **TERM** End the current unit of work on the failing processor and do not allow the recovery routines to retry
- **ACR** Invoke alternate CP recovery (ACR) to take the failing processor offline.
NOTE: The system chooses the appropriate action without requiring any decision or action. If an action taken in response to an occurrence of an excessive spin loop does not resolve the condition, the system takes the next action when the next excessive spin loop time out occurs. The default order in which the system takes the actions is SPIN, ABEND, TERM, and ACR.
• An installation can change the order of the actions, except the first one, that the system takes.
• For hardware-related errors that formerly caused message IEA490A, the system immediately initiates ACR processing without working through the sequence of actions and without requiring any intervention.
NOTE: There is a default spin loop time-out interval. You can change this interval through the combination of a parameter in EXSPATxx parmlib member and entering the SET command.