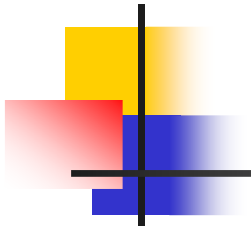
SLIP Trap Bootcamp

Part 1

NY Metro NaSPA – April 22, 2015

Patty Little **plittle@us.ibm.com**
IBM Poughkeepsie



Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- **MVS**
- **OS/390®**
- **z/Architecture®**
- **z/OS®**

* Registered trademarks of IBM Corporation

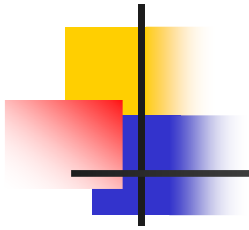
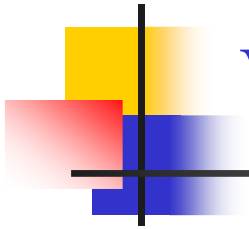
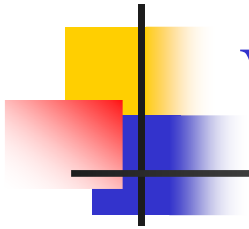


Table of Contents



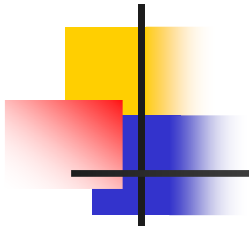
Why do we need to trap problems?

- **z/OS Debugging Utopia** – First Failure Data Capture (FFDC) gathers "the doc, the whole doc, and nothing but the doc" needed to resolve a problem
  - FFDC receives a high focus from z/OS support and development
- **z/OS Debugging Reality** – Sometimes we need to position ourselves to catch better documentation ("trap") on a problem recurrence
  - Need to be able to recognize the error environment
  - Need to be able to generate documentation to use in analyzing the error
  - Would be nice to be able to do this without having to modify code



What is SLIP/PER? Debugging Power!

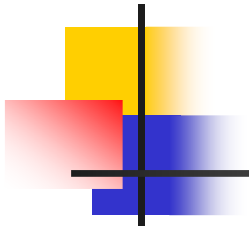
- SLIP – Serviceability Level Indication Processing
  - Used to trap **abends** and **messages**
  - Primarily software-driven
- PER – Program Event Recording
  - Used to trap hardware events
    - **Instruction Fetch**
    - **Storage Alteration**
    - **Successful Branch**
  - Hardware **detects event** and invokes SLIP software
  - SLIP software **applies filters** and **takes action(s)**
- SLIP/PER is often generically referred to as SLIP



Disclaimers

- We can't tell you everything about SLIP/PER in an hour!
- We will discuss the most useful keywords but will not be comprehensive.
- You may not choose to use some of these features on your own, but you should be aware of SLIP's wide range of capabilities when working problems with IBM support.
  - SLIP L2 welcomes questions from software service representatives.
- **We love questions!** We will try to answer your question right away. But SLIP is complex and sometimes subtle. If we don't know the answer off the top of our heads, we will get back to you!
- For all things SLIP, see the (very large) section on SLIP in the **MVS System Commands** manual.

Non-PER SLIP traps



Non-PER SLIP traps

- Non-PER SLIP traps use only software to monitor for error events
- Can monitor for abend codes (aka 'completion codes') and messages
- Can have multiple non-PER SLIP traps active at the same time, allowing one to trap for many different software events simultaneously

Non-PER SLIP trap filters

- COMP=
  - Monitor for a particular completion code
  - Wildcarding is allowed: COMP=0CX will monitor for any ABEND0Cx
  - If a recovery routine converts an abend, SLIP on the **original** abend code
- REASON=
  - Used with COMP=, monitors for a particular reason code
  - Wildcarding is allowed: REASON=000100xx
  - Only works if ABEND macro specified REASON=
    - If code puts a value into Reg15 instead of using the REASON= parameter on the ABEND macro, SLIP will not trigger
- MSGID=
  - Monitor for a particular message ID up to 10 characters in length
  - Works as you'd expect, but see manual for bells, whistles, and restrictions

Examples of non-PER SLIP traps

- SLIP SET,COMP=0C4,JOBNAME=TEST,
  A=TRACE,TRDATA=(STD,REGS),END
  - Write a GTF trace record when an ABEND0C4 occurs in job TEST

- SLIP SET,C=X37,RE=000000x8,A=RECORD,END
  - Write a record to logrec for any ABENDx37 with a reason code of 08 or 18 or 28, etc

- SLIP SET,MSGID=IEA421I,A=SVCD,END
  - Take an SVC dump when message IEA421I is issued

Internal processing of non-PER SLIPs

- For abends:
  - RTM receives control and calls SLIP processing (to see if the particular abend is being monitored) prior to driving recovery routines
  - SLIP checks the abend error information against its defined COMP= traps and takes action if it finds a match
- For messages:
  - WTO processing invokes SLIP to see if the particular message is being monitored
  - SLIP checks the message ID against its defined MSGID= traps and issues a transparent (retried) ABEND06F if it finds a match

PER SLIP traps

Events Monitored by PER

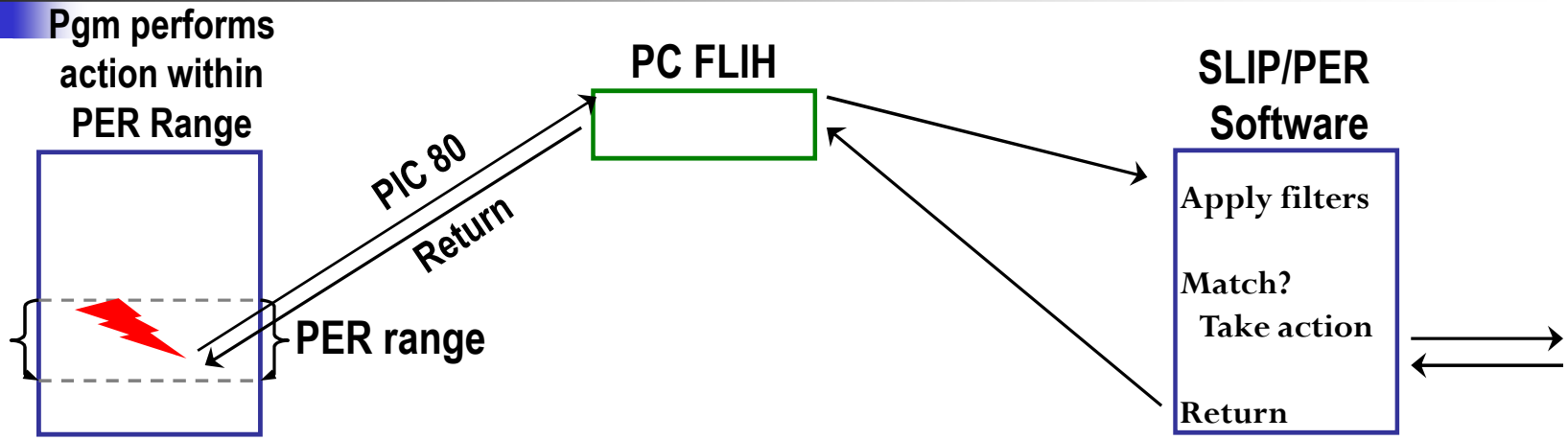
- Capability to **monitor** certain hardware events
  - **Instruction Fetch (IF)** – Fetching of instructions within a specified instruction range
  - **Storage Alteration (SA)** – Alteration of storage within a specified range
  - **Successful Branch (SBT)** – Branches made into or within a specified instruction range

The Power of PER

- What PER SLIP traps can do:
  - Monitor for IF, SA, or SBT over a range that can be hardcoded, coded through indirection, or identified as a module + offset
  - Filter events to a great level of specificity
  - Generate tailored documentation
  - Trap on a sequence of PER events ("dynamic PER")
  - Update storage and register content
  - Pseudo-IF/THEN/ELSE logic
- What PER SLIP traps can't do
  - Have more than one PER range being monitored at a time
  - Hit a moving target
  - Detect "DAT off" events
  - Issue a command or drive an exit
  - Arithmetic
  - Handle circular dynamic PER chains

NOTE: Some modules run with PER processing disabled to prevent recursion.

PER Processing



- Hardware detects a monitored event occurring and issues a program interrupt (PIC 80)
- The operating system's Program Check First Level Interrupt Handler (PC FLIH) receives control and routes control to SLIP/PER software
- SLIP/PER software checks filters of active PER trap to see if environment at the time of interrupt meets all criteria
  - YES – Takes requested action(s)
    - Usually operating system returns control to point of PER interrupt
  - NO – Operating system returns control to point of PER interrupt

PER Implementation

- When a PER trap is activated:
  - Control Reg9 on all CPs is set to indicate **event** being monitored
  - Control Regs 10 and 11 on all CPs are set to reflect the **range** being monitored
  - PSW bit 1 is turned on, indicating to hardware that it should **monitor for the PER** event defined via Ctrl Regs 9 thru 11
    - Whenever feasible based on the requested SLIP, the operating system will limit the setting of this bit to units of work in designated address spaces
- Can only monitor one PER range at a time on a system
- PER monitoring by hardware is very efficient
  - Poorly performing PER traps are the result of poor trap design

Performance Considerations in Designing PER Traps

- A PER trap does not have to match to affect performance
  - Every instruction executed in an IF range triggers PER processing
  - Every alteration of storage within a SA range triggers PER processing
  - Every branch to/within a SB range triggers PER processing
- Avoid setting a PER trap over a large range
- Be careful when setting a PER trap in a frequently executed or frequently altered range
- Be careful of setting a PER trap in a performance path
- When in doubt, use PRCNTLIM parameter! (default=10%)
  - Acts as a safety valve, disabling PER SLIP if trap using too much CP

PER Performance and the JOBNAME Parameter

- SLIP/PER provides a **JOBNAME** parameter which allows for **filtering of an event to only those units of work running under a particular job**
  - SLIP/PER will only turn on the PSW PER bit for units of work belonging to the specified job(s)
    - PER interrupts will not be experienced by work running under other jobs
  - Use JOBNAME whenever feasible

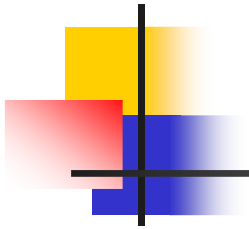
Defining PER Range to Be Monitored

- IF, SA, and SBT
  - RANGE RANGE=(beginningaddr,endingaddr)
    - Loaded into Control Registers 10 & 11 at time the SLIP is **SET**
    - RANGE is required on SA SLIPs
- IF and SBT only, instead of RANGE
  - NUCMOD or NUCEP
  - LPAMOD or LPAEP
  - PVTMOD or PVTEP

  } =(modname,startoffset,endoffset)
- SA only – identifies space where monitored storage resides
  - ASIDSA ASIDSA='JES2' or ASIDSA=3F or ASIDSA=SA
  - DSSA DSSA=('MYJOB'.MYDSPAC1) [for data space]

Examples of PER trap ranges

- SLIP SET,IF,LPAMOD=(IEFBR14),A=SVCD,END
  - If any instruction in LPA-resident load module IEFBR14 is executed, take an SVC dump
    - Default for A=SVCD is a MATCHLIM of 1 (trap disables after 1 match)
- SLIP SET,SBT,NUCMOD=(IEAVEPST,200,2FF), A=TRACE,TRDATA=(STD,REGS),END
  - Whenever an instruction is branched to within the range of 200 thru 2FF in nucleus module IEAVEPST, write a GTF trace record
    - Default for A=TRACE is unlimited matches
- SLIP SET,SA,RA=(6000,600F),ASIDSA='MYJOB', JOBNAME=PATTY,A=SVCD,END
  - If code running in job PATTY alters any private storage at 6000 thru 600F in MYJOB's address space, take an SVC dump



Filters for SLIP traps

Filters for SLIP/PER

- Filtering to an address space level
  - JOBNAME — **home** addr sp must have specified **JOBNAME**
  - ASID — **primary** addr space must have specified **ASID**

  - Consider the case of a unit of work belonging to JOB ABC (ASID 100) PC'ing to ASID 200 and abending with an ABEND0C6
    - SLIP C=0C6,JOBNAME=ABC will trap this (Home ASID=100=jobname ABC)
    - SLIP C=0C6,ASID=200 will trap this (Primary ASID=200)
    - SLIP C=0C6,ASID=100 will NOT trap this (Primary ASID=200, not 100)
- Filtering on what code triggered the event (non-PER or SA PER only)
  - ADDRESS
  - NUCMOD or NUCEP
  - LPAMOD or LPAEP
  - PVTMOD or PVTEP

  } abend, message, or storage alteration done by code in specified address range

Filters for SLIP/PER

- Filtering on environment
  - MODE — event occurred with work in specified mode e.g. locked, disabled, problem state, SRB mode, system key, home mode (P=H), etc.
  - PSWASC — event occurred with work in specified mode of execution (home, primary, secondary, AR)
- Filtering on storage and/or register content
  - DATA — test register and/or stg content at time of event

SLIP Actions

Actions for Non-PER traps

- For SVC dumps:
  - SVCD — take an SVC dump
  - TRDUMP — take SVC dump after "n" GTF trace records
  - NOSVCD — prevent recovery from taking an SVC dump
- For GTF trace:
  - TRACE — write a GTF trace record
  - TRDUMP — take SVC dump after "n" GTF trace records
  - STOPGTF — stop collection of GTF trace records
- For logrec:
  - RECORD — record error information to LOGREC
- Other
  - IGNORE — to ignore the event
  - WAIT — to enter restartable wait state

Note: some actions can be combined.

Actions for PER traps

- For SVC dumps:
  - SVCD — take an SVC dump
  - SYNCSVCD — take a synchronous SVC dump (unit of work requesting dump does not resume until dump is complete)
  - TRDUMP — take SVC dump after "n" GTF trace records
  - STDUMP — take SVC dump after "n" system trace records
- For GTF trace:
  - TRACE — write a GTF trace record
  - TRDUMP — take SVC dump after "n" GTF trace records
  - STOPGTF — stop collection of GTF trace records
- For system trace:
  - STRACE — write a system trace record
  - STDUMP (PER traps only) — take SVC dump after "n" system trace records

Note: some actions can be combined.

Actions for PER traps (cont)

- Other
  - REFBEFOR, REFAFTER — update storage/registers
  - TARGETID — activate a new PER trap
  - RECOVERY — force an abend
  - SUBTRAP — simulate "IF-THEN-ELSE" logic
  - IGNORE — ignore the event
  - WAIT — enter restartable wait state

Note: some actions can be combined.

Tailorability

- SVC dumps
  - JOBLIST, ASIDLST, DSPNAME — dump address/data spaces
  - SDATA — dump particular types of storage areas
  - SUMLIST, LIST — dump specified storage ranges
  - STRLIST — dump coupling facility structures
  - REMOTE — take dumps on other systems in sysplex
- GTF trace
  - TRDATA — trace specified storage ranges
- System trace
  - STDATA — trace specified storage ranges

Using Indirection in SLIPs

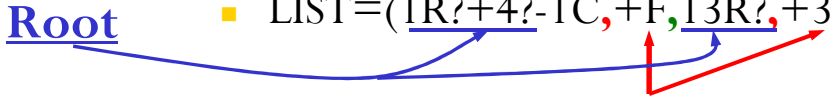
Range Pairs

- Syntax for range pairs: =(beginaddr,endaddr)
  - **Beginaddr, endaddr** may be address or indirect address

- Parameters that support range pairs:
  - RANGE, ADDRESS
  - TRDATA, STDATA
  - LIST, SUMLIST

  } **Support multiple range pairs**

- What is an indirect address?
  - Combination of address or register notation, indirection symbols (%, ?, !), and/or displacements, symbolics (BEAR, BPER)
    - LIST=(1R?+4?-1C,1R?+4?+F,13R?+0,13R?+3F)
    - LIST=(1R?+4?-1C,+F,13R?,+3F) shorthand for previous example



Note:These are displacements, NOT LENGTHs!!

Triplets (data comparison, storage refresh)

- Syntax for triplets: =(target,operator,source)
  - **Target** may be address, indirect address, or register
  - **Source** may be address, indirect address, register, or constant (max 8 bytes)
  - **Operator** options: EQ, NE, LT, NL, GT, NG
    - If 'A' appended (e.g. EQA) – operator acts on the **address** of the source
    - If 'C' appended (e.g. EQC) – operator acts on the **content** of the source
    - If used without 'A' or 'C' appended, the source must be a **constant**
  - **"(b)" following target address** indicates a bit position within the targeted byte or register; indicates operation is on binary data
  - **"(n)" following operator** indicates how many bits or bytes to operate on for Address and Contents operations
- Parameters: REFBEFOR, REFAFTER, DATA
  - All support multiple triplets, separated by AND or OR (default is "AND")
  - DATA=(1R?+8(0),EQ,1,OR,15R,EQC(4),13R?+10,OR,0R,GT,0)
  - REFBEFOR=(FEBCC0,EQ,C1C2C3C4,15R,EQ,0)

Controlling SLIP traps

Other SLIP/PER trap controls

- Parameters
  - ENABLE, DISABLE — to define trap enabled/disabled on SET
  - MATCHLIM — to disable trap after N matches
  - PRCNTLIM (PER only) — to disable trap if it uses >N pct of CP
  - ID — to name SLIP trap (up to 4 characters)
- Commands
  - SLIP SET, — set a SLIP trap
  - SLIP MOD,ENABLE,ID= — enable a SLIP trap
  - SLIP MOD,DISABLE,ID= — disable a SLIP trap
  - SLIP DEL,ID= — delete a SLIP trap
  - D SLIP — display names/status of all SLIPs
  - D SLIP= — display details of specific SLIP

Any Questions?