

# Debugging Program Checks

—  
Patty Little  
IBM z/OS Support



# 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

**For a complete list of IBM trademarks, see: <http://www.ibm.com/legal/us/en/copytrade.shtml>**

**The following are trademarks or registered trademarks of other companies.**

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

\* All other products may be trademarks or registered trademarks of their respective companies.

## **Notes:**

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

# Table of Contents

|                             |    |
|-----------------------------|----|
| Program Checks .....        | 4  |
| Debugging an ABEND0C4 ..... | 14 |
| Debugging an ABEND0C1 ..... | 28 |
| Summary .....               | 34 |



# What is a Program Check?

- An **interrupt** represents a break in execution of a program
- A **program interrupt** results from a program request that cannot be resolved by hardware
  - Can the Operating System process this interrupt?
    - YES - “**Resolvable**”
    - NO - “**Non-resolvable**”
  - When a non-resolvable program interrupt occurs on an executable unit of work, it is generally known as a **program check**



# Program Interrupts

When a program interrupt occurs :

- **Hardware** (the mainframe architecture)
  - Updates PSA with:
    - ILC/IC (Instruction-Length Code/Interrupt Code)
    - TEA (Translation Exception Address), if applicable
    - BEA (Breaking Event Address)
  - Gives control to z/OS First Level Interrupt Handler (FLIH) via PSW swap
- **Software** (the z/OS operating system)
  - If resolvable, handles interrupt and then resumes interrupted program
  - If non-resolvable, terminates current unit of work with completion code



# Completion Codes from Program Checks

- When a work unit is abnormally terminated due to a Program Check, RTM converts the **Program Interrupt Code (PIC)** to a **Completion Code of 0Cx, 0Dx or 0E0**
  - ABEND0Cx (e.g. 0C1, 0C4, 0C6, 0C9)
  - ABEND0Dx (e.g. 0D5)
  - ABEND0E0 RCyy (e.g. 0E0 RC29)
    - x may or may not be the PIC number
    - yy is usually the PIC number
- Check z/OS MVS System Codes for more information



# Locating Completion Codes in Dumps

- **ST FAILDATA** or **VERBX LOGDATA**
  - Error information in the dump header or a LOGREC entry
- **SYSTRACE**
  - **RCVY** system trace entries
- **SUMM FORMAT**
  - In the TCB or the TCB Summary
  - In the **RTM2WA** under a TCB (if available)



# Program Checks – ILC, IC, and TEA

- **ILC** (Instruction Length Code)
  - The length of the instruction that caused the program interrupt
- **IC** (Interrupt Code)
  - The program interrupt code (commonly known as PIC)
- **TEA** (Translation Exception Address)
  - For some PICs, contains virtual address that was faulted on, rounded to a page boundary (x'1000')





# ST FAILDATA or VERBX LOGDATA

| Symptom         | Description                           |
|-----------------|---------------------------------------|
| -----           | -----                                 |
| PIDS/5752SC100  | Program id: 5752SC100                 |
| RIDS/IFAEDABC#L | Load module name: IFAEDABC            |
| RIDS/IFAEDABC   | Csect name: IFAEDABC                  |
| AB/S00C4        | System abend code: 00C4               |
| PRCS/00000010   | Abend reason code: 00000010           |
| REGS/C1016      | Register/PSW difference for R0C:-1056 |
| RIDS/IFAEDDEF#R | Recovery routine csect name: IFAEDDEF |

# ST FAILDATA or VERBX LOGDATA (cont)

## TIME OF ERROR INFORMATION

PSW: 47044400 80000000 00000000 2747B016  
Instruction length: 06      Interrupt code: 0010  
Failing instruction text: B24D005C 1846D207 50104038  
Translation exception address: 00000000\_00810001

ILC      IC      TEA

Breaking event address: 00000000\_2747A6E0

### Registers 0-7

GR: 00000003 2F16DB18 29E58510 069E2590 008100C2 2F089B20 006000C2 007FF6D8  
AR: 00000000 00000000 00000002 00000000 00000000 00000000 00000000 00000002

### Registers 8-15

GR: 2F16E618 00000000 2F16EDEE 2F16DDEF 2746E5B4 2F16CDF0 29E58510 00000000  
AR: 00000000 00000002 00000000 00000000 00000000 00000000 00000000 00000000

Home ASID: 02C0      Primary ASID: 0011      Secondary ASID: 0011



# SUMM FORMAT ASID(x'nn')

|               |          |          |          |          |                    |
|---------------|----------|----------|----------|----------|--------------------|
| TCB: 007FF6D8 |          |          |          |          |                    |
| +0000         | RBP..... | 007FF650 | PIE..... | 00006E00 | DEB..... 00000000  |
| +000C         | TIO..... | 007BDFE8 | CMP..... | 940C4000 | TRN..... 40000000  |
| +0018         | MSS..... | 7F472928 | PKF..... | 80       | FLGS..... 01000000 |
| +012C         | EAE..... | 7FFFE1D8 | ARC..... | 00000010 |                    |

|                       |                                    |          |          |          |          |
|-----------------------|------------------------------------|----------|----------|----------|----------|
| RTM2WA SUMMARY        |                                    |          |          |          |          |
| -----                 |                                    |          |          |          |          |
| +001C                 | Completion code                    |          |          | 840C4000 |          |
| +008C                 | Abending program name/SVRB address | 007FF8E0 | 00000000 |          |          |
| +0094                 | Abending program addr              |          | 00000000 |          |          |
| GPRs at time of error |                                    |          |          |          |          |
| 0-3                   | 00000003                           | 2F16DB18 | 29E58510 | 069E2590 |          |
| 4-7                   | 006000C2                           | 2F089B20 | 008100C2 | 007FF6D8 |          |
| 8-11                  | 2F16E618                           | 00000000 | 2F16EDEE | 2F16DDEF |          |
| 12-15                 | 2846E5B4                           | 2F16CDF0 | 29E58510 | 00000000 |          |
| +007C                 | EC PSW at time of error            | 470C4400 | A747B016 | 00060010 | 00810001 |

ILC

IC

TEA

# SYSTRACE ASID(x'nn')

| PR   | ASID | WU-ADDR- | IDENT | CD/D | PSW----- | ADDRESS- | UNIQUE-1 | UNIQUE-2 | UNIQUE-3 |
|------|------|----------|-------|------|----------|----------|----------|----------|----------|
|      |      |          |       |      |          |          | UNIQUE-4 | UNIQUE-5 | UNIQUE-6 |
| 0001 | 02C0 | 007FF6D8 | PGM   | 010  | 00000000 | 2747B016 | 00060010 | 00000000 |          |
|      |      |          |       |      | 47044400 | 80000000 |          | 00810001 |          |
| 0001 | 02C0 | 007FF6D8 | *RCVY | PROG |          |          | 940C4000 | 00000010 | 00000000 |

IC

ILC IC TEA

completion code reason code

**Note: \*RCVY entry immediately following PGM entry on same processor indicates a non-resolvable program interrupt (program check) - BAD!**

**PGM entry without \*RCVY entry immediately following on same processor indicates the program interrupt has been resolved and is not a problem.**



# General Diagnostic Approach

- Gather program check data
  - LOGREC/VERBX LOGDATA, ST FAILDATA, and RTM2WA control blocks are good sources of information
- Analyze program check data
- Read compiled listing of source code, beginning at the failing PSW and moving backwards to understand why the flow led to this error
- Review LOGREC, system log and system trace for history of events leading up to this error
- Review SUMMARY FORMAT for chronology of module flow leading up to this error under failing TCB



# What is an ABEND0C4?

- Common causes:
  - **Translation exception (aka fault)**
    - Non-resolvable segment fault (PIC 10)
    - Non-resolvable page fault (PIC 11)
    - Non-resolvable ASCE fault (PIC 38)
    - Non-resolvable region fault (PIC 39, 3A or 3B)
    - Disabled segment/page fault
    - Disabled region/ASCE fault
  - **Protection exception (PIC 4)**
- Can occur while:
  - Accessing an operand
  - Fetching an instruction



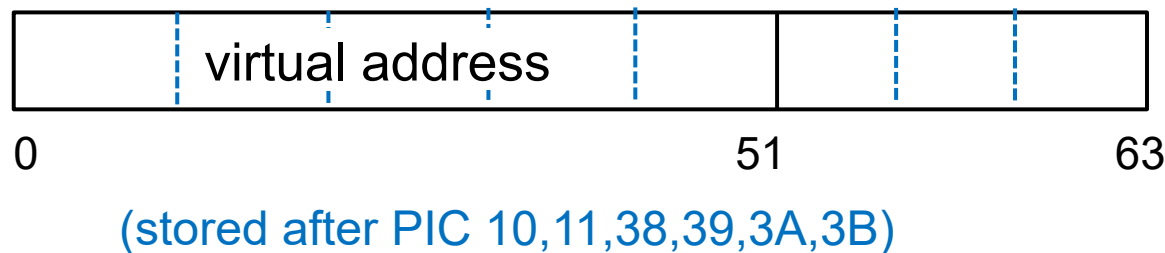
# ABEND0C4 Causes

- **Non-resolvable PIC 10 or 11**
  - An invalid below-the-bar virtual address was being used
- **Non-resolvable PIC 38, 39, 3A or 3B**
  - An invalid above-the-bar virtual address was being used
- **Disabled PIC 10, 11, 38, 39, 3A or 3B**
  - A page not backed by real storage was accessed by a program while it was disabled for certain interrupts (I/O and external)
  - Check the second digit of the failing PSW
    - If it is 4, the program is Disabled
    - If it is 7, the program is Enabled
- **Protection exception (PIC 4)**
  - A program violated storage protection protocol

# Translation Exception Address (TEA)

- Contains:
  - First portion of **virtual address** causing a PIC 10, 11, 38, 39, 3A, or 3B
  - Bits describing the **cross-memory environment** at the time of the program interrupt
- Use it to determine:
  - **Why** error occurred
  - Whether error occurred during **operand access or instruction fetch**

## z/Architecture **8-byte TEA**



|             |                           |
|-------------|---------------------------|
| Bits 0-51:  | Virtual address bits 0-51 |
| Bits 52-59: | Unpredictable             |
| Bits 62-63: | 00 Primary ASCE           |
|             | 01 AR Mode                |
|             | 10 Secondary ASCE         |
|             | 11 Home ASCE              |





# Debugging an ABEND0C4

- Depending on PIC, error PSW points directly **AFTER** or **AT** the failing instruction
- RTM gathers 12 bytes of **failing instruction text** for diagnostic purposes
  - 6 bytes of data before the error PSW
  - 6 bytes of data after the error PSW
  - Max instruction length is 6 bytes, so guaranteed to include failing instruction
- Locate the failing instruction
  - PICs 10,11,38,39,3A,3B - PSW points **at failing instruction**
  - PIC 4 - PSW points **after failing instruction**
    - Use ILC to determine where failing instruction begins
- Review **failing instruction**, along with corresponding **registers** at time of error and **TEA** (when appropriate)

F074A784000C D203E02C7624

PSW



# Where Did the Error Occur?

- Locate the error PSW and Registers
  - **IPCS BROWSE** or **WHERE** on the error PSW address
- Locate the failing instruction
  - Failing instruction text in **ST FAILDATA**, **VERBX LOGDATA**, **RTM2WA**, etc.  
OR
  - **IPCS BROWSE / LIST** failing instruction address
  - **For PIC4**, remember to back up PSW address by instruction length!
- What is the failing instruction?
  - **IP OPCODE** failing\_instruction  
OR
  - **IP LIST** failing\_instruction\_address **INSTR**



# Examples: IP OPCODE , IP LIST addr INSTR

**IP OPCODE 58304000**

Provide instruction

Mnemonic for X'58304000' is L

**IP LIST 20F54 INSTR**

Provide address of instruction  
in storage

|              |               |               |               |
|--------------|---------------|---------------|---------------|
| LIST 020F54. | ASID(X'00B2') | LENGTH(X'04') | INSTRUCTION   |
| 00020F54     | 5830 4000     | L             | R3,X'0' (,R4) |



# Consider the Failing Instruction

- Gather PIC-specific information
  - PIC 10, 11, 38, 39, 3A, 3B: Locate Translation Exception Address (TEA)
  - PIC 4: Note PSW execution key
    - 078D0000 90234568 = Execution Key8
    - 07850000 80000000 00000000 10234568 = Execution Key8
- Identify assembler base & index registers of failing instruction
  - For fault-related PIC, which register matches Translation Exception Address?
  - For PIC4:
    - Does register contain zero or pointer to PSA low core (0-1FF, 1000-11FF) ?  
OR
    - Does storage key of address in register not match PSW execution key ?

# Example: ABEND0C4 PIC11

## IP VERBX LOGDATA

```
TYPE:  SOFTWARE RECORD      REPORT:  SOFTWARE EDIT REPORT      DAY.YEAR
JOBNAME: ABCD      SYSTEM NAME: SYSA
ERRORID: SEQ=00251  CPU=0080  ASID=0080  TIME=13:41:18.1
```

### TIME OF ERROR INFORMATION

```
PSW: 07041000 80000000 00000000 050A1856
```

```
INSTRUCTION LENGTH: 06  INTERRUPT CODE: 0011
```

```
FAILING INSTRUCTION TEXT: 00805850 B168D93F B2245000
```

```
TRANSLATION EXCEPTION ADDRESS: 00000000_2D8D9800
```

```
BREAKING EVENT ADDRESS: 00000000_050A20BC
```

### REGISTERS 0-7

```
GR: 00000004 2D8D9FB8 2D8D9FC3 0000000C 00000008 2D8D9FB8 050A45ED 00FC8D00
```

```
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

### REGISTERS 8-15

```
GR: 050A35EE 00000000 050A25EF 2D824BB8 850A15F0 2D824D94 850A1842 00000080
```

```
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Interrupt Code = PIC11:

PSW points **AT** failing instr

Instruction Length = 6:

Failing instr is **D93FB2245000**

**IP OPCODE D93FB2245000** or  
**IP LIST 50A1856 LEN(6) I**

MVCK instr: D9 3F **B224 5000**

Instruction base registers  
are **R11** and **R5**

**TEA = 2D8D9xxx** matches **R5**

**Conclusion:** Storage pointed to by R5 is not available. This requires further investigation.

# Example: ABEND0C4 PIC10

## IP STATUS FAILDATA

### Time of Error Information

PSW: 04040000 80000000 00000000 014CAB2A

Instruction length: 04      Interrupt code: 0010

Failing instruction text: 0010B20A 200098EF 1000B20A

Translation exception address: 00000000\_4F4F4000

Breaking event address: 00000000\_014CAAE8

|             |                            |                            |
|-------------|----------------------------|----------------------------|
| AR/GR 0-1   | FFFFFFFF/00000000_00000058 | 00000000/00000000_4F4F4F57 |
| AR/GR 2-3   | FFFFFFFF/00000000_0000040C | FFFFFFFF/00000000_81030A38 |
| AR/GR 4-5   | FFFFFFFF/FFFFFFFF_0215BA28 | FFFFFFFF/00000000_00000C00 |
| AR/GR 6-7   | FFFFFFFF/00000000_04355C00 | FFFFFFFF/00000000_4F4F4F4F |
| AR/GR 8-9   | 00000000/00000000_04435B18 | 00000000/00000000_00000001 |
| AR/GR 10-11 | 00000000/000001EF_810303D8 | 00000000/000001EF_7FFFC9A8 |
| AR/GR 12-13 | 00000000/00000000_814CAB0A | 00000000/00000000_7FFFC9A8 |
| AR/GR 14-15 | 00000000/00000000_0000030A | 00000000/00000000_00000000 |

**Interrupt Code = PIC10:**

PSW points **AT** failing instr

**Instruction Length = 4:**

Failing instr is **98EF1000**

**IP OPCODE 98EF1000** or  
**IP L 14CAB2A LEN(4) I**

LM instruction: 98 EF 1000

Instr base reg is **R1**

**TEA = 4F4F4xxx:**  
matches **R1**

**Conclusion:** Storage pointed to by R1 is not available.  
R1 does not look like an address.  
Examine code to understand how R1 was derived.



# Protection Exception (PIC4)

Most common causes:

- **Key-controlled protection**
  - PSW execution key must either match storage key or be KEY0 when:
    - Writing to storage
    - Reading from **fetch-protected** storage
  - Otherwise, an ABEND0C4 PIC 4 will occur
- **Low core address protection**
  - PSA x'000-1FF' and x'1000-11FF' are **write-protected**

# How to Determine the Storage Key

- Associated with 4K (1 Page) of storage



**KKKK** = Key (4 bits)  
**F** = Fetch-Protection Bit  
**R** = Reference Bit  
**C** = Change Bit  
**0** = Reserved

- To find the storage key in a dump:
  - **LIST storage\_addr DISPLAY**

```
LIST 0097F5E0 CPU(X'00) ASID(X'0001') LENGTH(4) AREA
CPU(X'00) ASID(X'0001') ADDRESS(0097F5E0) KEY(06) ABSOLUTE(01CDB5E0)
0097F5E0. 00000000
```

Key 0

x'6' = b'0110'  
Not fetch-protected





# Less Common Protection Exception

- **Page protection**
  - A page is write-protected if the **page protection bit** is ON in the page table entry
    - Verify via **RSMDATA VIRTPAGE** or **RSMDATA HIGHVIRT** report
    - Read-only nucleus and LPA always have this bit turned on
  - Running DAT-off bypasses this protection
- When a PIC 4 is due to page protection:
  - TEA bit 61 will be turned on (z/Architecture mode)
  - TEA bits 0-51 will contain bits 0-51 of virtual address causing PIC 4

# Example: ABEND0C4 PIC4

## IP ST FAILDATA

### Time of Error Information

PSW: 07141000 80000000 00000000 3880A5A0

Instruction length: 06      Interrupt code: 0004

Failing instruction text: D5006008 3000A784 00084160

Breaking event address: 00000000\_3880A5A8

|             |                            |                            |
|-------------|----------------------------|----------------------------|
| AR/GR 0-1   | 00000000/00000000_00000000 | 00000000/00000000_00000001 |
| AR/GR 2-3   | 00000000/20000000_3893990B | 00000000/00000000_388DCF90 |
| AR/GR 4-5   | 00000000/00000000_3870A07C | 00000000/00000000_00000000 |
| AR/GR 6-7   | 00000000/00000000_00000930 | 00000000/00000000_3893990A |
| AR/GR 8-9   | 00000000/00000000_3870A010 | 00000000/00000000_0003922A |
| AR/GR 10-11 | 00000000/00000000_388D52B0 | 00000000/00000000_00007000 |
| AR/GR 12-13 | 00000000/00000000_3880A530 | 00000000/00000000_38939688 |
| AR/GR 14-15 | 00000000/00000000_B79C70A8 | 00000000/00000000_FFFFFFFD |

**Interrupt Code = PIC4:**

PSW points **AFTER** failing instr

**Instruction Length = 6:**

Failing instr is D50060083000

**IP OPCODE D50060083000**

or

**IP L 3880A59A LEN(6) I**

CLC instr: D500 6008 3000

Instr base registers are

**R6** and **R3**

**PSW Key = 1**

**Note:** To update storage or to reference fetch-protected storage, the PSW key must match the page key or else be KEY0. Need to verify page keys and fetch-protect status.

# Example: ABEND0C4 PIC4 continued

Note: Source R3 = 388DCF90    Target R6 = 00000930

## LIST 388DCF90 DISPLAY

```
LIST 388DCF90. ASID(X'002B') LENGTH(X'04') AREA
ASID(X'002B') ADDRESS(388DCF90.) KEY(18) ABSOLUTE(06_3D60FF90.)
388DCF90. 0190E080
```

Key 1  
Fetch-protected

## LIST 930 DISPLAY

```
LIST 0930. ASID(X'002B') LENGTH(X'04') AREA
ASID(X'002B') ADDRESS(0930.) KEY(08) PREFIXED
00000930. 00000000
```

Key 0  
Fetch-protected

# What is an ABEND0C1

- Occurs when CPU attempts to execute an instruction with an invalid operation code
- Common causes:
  - Wild branch
  - Overlay of code
- Can occur while:
  - Fetching an instruction



# Debugging an ABEND0C1

- PSW points **after** failing “instruction”
- Use ILC to determine where failing “instruction” begins
  - Since instruction is invalid, identifying failing “instruction” is of limited value

Failing instruction text: 00000000 00000000 E02C7624

↑  
PSW

- TEA is not useful, since abend is not for a translation exception
- Instead, use **BEAR** !



# BEAR – Breaking Event Address Register

- A 64-bit register containing the **address of the last instruction that causes a break in sequential execution**
  - For example, a branch or a LPSW instruction
- Content of BEAR stored in PSA by H/W when any program interrupt occurs. This is propagated by z/OS to:
  - **SDWA** (available in **ST FAILDATA** or **VERBX LOGDATA**)
  - **RTM2WA** (available in **SUMM FORMAT**)

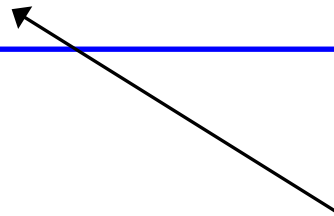


# Finding BEAR in a DUMP

## ST FAILDATA or VERBX LOGDATA

### TIME OF ERROR INFORMATION

```
PSW: 07040001 80000000 00000000 0AF8D286  
INSTRUCTION LENGTH: 06    INTERRUPT CODE: 0010  
FAILING INSTRUCTION TEXT: 17884280 3015E320 60180004  
TRANSLATION EXCEPTION ADDRESS: 00000008_004FF800  
  
BREAKING EVENT ADDRESS: 00000000_0AF8C754
```



Address of last instruction  
to cause a change in flow



# Finding BEAR in a Dump (continued)

## IP SUMM FORMAT ASID(x'nn')

RTM2WA: 7FFAFE10

|       |           |          |           |          |           |    |           |        |
|-------|-----------|----------|-----------|----------|-----------|----|-----------|--------|
| +0000 | ID.....   | RTM2     | ADDR..... | 7FFAFE10 | SPID..... | FF | LGTH..... | 0011F0 |
| +0014 | VRBC..... | 009FD550 | ASC.....  | 00F882A0 | CCF.....  | 84 | CC.....   | 0C1000 |

....

.... *Lines omitted here*

....

|       |           |          |          |          |          |  |  |  |
|-------|-----------|----------|----------|----------|----------|--|--|--|
| +06C8 | TRNE..... | 00000000 | 072FF800 |          |          |  |  |  |
| +06D0 | BEA.....  | 00000000 | 0AF8B552 |          |          |  |  |  |
| +06D8 | PSW1..... | 07040001 | 80000000 | 00000000 | 0AF8D180 |  |  |  |







# Example: ABEND0C1

## IP ST FAILDATA

TIME OF ERROR INFORMATION

PSW: 07850000 80000000 00000000 00000002

Instruction length: 02    Interrupt code: 0001

Failing instruction text: 00000000 000A0000 000130E1

Breaking event address: 00000000\_00007F20

Registers 0-7

|     |          |          |          |          |          |          |          |          |
|-----|----------|----------|----------|----------|----------|----------|----------|----------|
| GR: | 00000000 | 00007EF8 | 00000040 | 007D5D84 | 007D5D60 | 007FF448 | 007C7FE0 | FD000000 |
| AR: | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

Registers 8-15

|     |          |          |          |          |          |          |          |          |
|-----|----------|----------|----------|----------|----------|----------|----------|----------|
| GR: | 00007F22 | 007FF708 | 00000000 | 007FF448 | 965AB8B2 | 00006F60 | 80007F22 | 00000000 |
| AR: | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 | 00000000 |

1BFF  
05EF

SR 15,15  
BALR 14,15

From browsing storage

|          |                  |          |          |          |             |  |               |     |
|----------|------------------|----------|----------|----------|-------------|--|---------------|-----|
| 00007F1E |                  |          |          |          | <u>1BFF</u> |  | ..            |     |
| 00007F20 | <u>05EF</u> 5900 | F0D24780 | F01290EC | D00C1831 |             |  | ....0K..0...} | ... |

# Summary

- A program check is converted into a completion code of 0Cx, 0Dx, or 0E0 RCyy.
- The **IC, ILC, PSW, and failing instruction text** can be found in LOGREC data or in a dump (e.g. VERBX LOGDATA, ST FAILDATA, SYSTRACE, RTM2WA).
  - This data helps determine why the program check occurred.
- The **TEA** is crucial for debugging **ABEND0C4's for translation exceptions** (faults).
- **BEAR** is very useful when debugging an **ABEND0C1** due to a wild branch.

# Thank you

Patty Little  
IBM z/OS Support  
—  
plittle@us.ibm.com  
+1-845-435-4037  
ibm.com

© Copyright IBM Corporation 2020. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at [Copyright and trademark information](#).