CICS TS Tutorial -- Transaction Dump Analysis

Eugene S Hudders
C\TREK Corporation
ehudders@ctrek.com
787-397-4150
Session 9618
Friday 8:00 AM – August 12, 2011
Special Thanks

- A special thank you is due to Mr. Andy Wright from Hursley who provided a debugging presentation that was used by the presenter in preparing this session.

THANK YOU VERY MUCH!!
Agenda

• Introduction
  • Important Dump Information
  • What is an ASRA Dump?
    • Program Checks
  • PSW
• The Big COBOL Picture
  • Major Control Blocks
    • DSA
    • TGT
• Types of Dumps
  • System Dumps
  • Transaction Dumps
• Analyzing a Transaction Dump
  • Cancelation Information
  • PSW
  • Computing the Program Offset to Identify the Failing Instruction
  • BLW/BLL Cells
  • Locating the Failing Field(s)
  • Cookbooks
• Closing
Introduction

- This presentation describes a methodology that can be used to debug a program check (ASRA) in a COBOL program using a transaction dump.
- Some of the information provided can be used to debug transaction dumps in other programs.
Important Dump Information

- PSW – contains reference to where the program failed
- Registers at the time of the cancellation
- The failing program
- The failing instruction as per PSW
- The data involved in the cancelation
- The address of the last EXEC CICS issued
- Information if LINK was used
- Symptom string
What Is an ASRA Dump?

• An ASRA is CICS’ equivalent to a program check transaction dump (S0Cn cancelation)
• The initial cancelation is taken as a system abend AP0001 or SR0001
  • The internal code is AKEA because the cancelation is intercepted by the Kernel Domain
• The difference between an AP or SR system dump depends on the execution key assigned to the task
  • Key 8 (CICS) → AP0001
  • Key 9 (User) → SR0001
• As a result that you will receive both a system and transaction dump, it is probably a good idea to suppress the AP0001/SR0001 system dumps to reduce overhead
Program Checks

- Program Check codes:
  - 01 Operation Exception (*)
  - 02 Privileged Operation Exception
  - 03 Execute Exception
  - 04 Protection Exception (*)
  - 05 Addressing Exception
  - 06 Specification Exception
  - 07 Data Exception (*)
  - 08 Fixed Point Overflow Exception
  - 09 Fixed Point Divide Exception (*)
  - 0A Decimal Overflow Exception
  - 0B Decimal Divide Exception (*)
  - 0C HFP Exponent Overflow Exception
  - 0D HFP Exponent Underflow Exception
  - 0E HFP Significance Exception
  - 0F HFP Floating Point Divide Exception

Note: * indicate most common to debug
Program Checks

- There are program checks associated with virtual storage addressing exceptions which are treated as protection exceptions (S0C4)
  - 0010 Segment-translation exception
  - 0011 Page-translation exception
  - 0038 ASCE-type exception
  - 0039 Region-first-translation exception
  - 003A Region-second-translation exception
  - 003B Region-third-translation exception
- The cancelations appear as 0C4 RC=nn where “nn’ is the program check code
- It is important to remember that in these types of program checks, the PSW address is actually pointing to the instruction that caused the program check
  - This is different for all the other program checks that occur in the system including a S0C4 RC=04 which point to the next sequential instruction
Program Status Word

- The Program Status Word (PSW) is the most important control block (hardware) in a dump
- Some of the things that the PSW provides:
  - The address of the next sequential instruction for execution
    - There are some exceptions that are discussed later
  - Provides the current protection key being used by the program
  - Provides some masks for interrupts
  - Indicates if we are in problem or supervisory state
  - Current condition code
  - Access mode (Primary, Access Register, etc.)
  - Indicates if we are in wait or execution state
  - Program mask
- PSW can be either 8 (31-bits address) or 16 bytes (64-bits address) long
  - The PSW is presented in the 8 byte format because programs can only execute below the bar
  - Data is the only thing that can be placed above the bar
Program Status Word

- **ESA/390 PSW**

- 0 R 0 0 0 T L E Key 1 M W P AS CC PRG Mask 0 0 0 0 0 0 0 0

  Bit 12 – indicates ESA/390 Mode
  
  Bit 32 – Addressing Mode (1 = 31-bit addressing)
Program Status Word

- z/Architecture PSW (Bit 12 = 0)

- 0 0 0 0 T I E Key 1 M W P AS CC PRG Mask 0 0 0 0 0 0 0 0 FA

Bits 0 – 31 of the Instruction Address

Bits 32 – 63 of the Instruction Address
Program Status Word

- z/Architecture PSW (Bit 12 = 0)
  - R → Program Event Recording (PER)
  - T → Dynamic Address Translation (on)
  - I → I/O Mask
  - E → External Mask
  - PSW Key → Executing Key
  - M → Machine Mask
  - P → Problem (1) Supervisory (0) State
  - Access Mode → Primary, Secondary, Home or AR
  - CC → Condition Code
  - Program Mask → FP or Decimal Overflow, Exponent Underflow or Significance
  - EA → Extended Addressing mode
  - BA → Basic Addressing Mode
Program Status Word

- z/Architecture PSW (Bit 12 = 0)
  - Addressing Modes EA and BA
    - 00 → 24-Bit Addressing Mode
    - 01 → 31-Bit Addressing Mode
    - 10 → Invalid
    - 11 → 64-Bit Addressing Mode
The Big COBOL Picture

- To debug a COBOL program you need to access two major COBOL control blocks
  - **DSA** – Dynamic Save Area that contains the COBOL program’s registers and a pointer to the TGT
  - **TGT** – Task Global Table that contains the address of the program and the BLW/BLL cells which address the areas used by the program
The Big COBOL Picture

Programming standards since the “beginning of time” have designated GPR 13 to point to the current save area.
Dynamic Storage Area

*** DSA MEMORY MAP ***

DSALOC

00000000 REGISTER SAVE AREA
0000004C STACK NAB (NEXT AVAILABLE BYTE)
00000058 ADDRESS OF INLINE-CODE PRIMARY DSA
0000005C ADDRESS OF TGT
00000060 ADDRESS OF CAA
00000080 XML PARSE WORK AREA ANCHOR
00000084 SWITCHES
00000088 CURRENT INT. PROGRAM OR METHOD NUMBER
0000008C ADDRESS OF CALL STATEMENT PROGRAM NAME
00000090 CALC ROUTINE REGISTER SAVE AREA
000000C4 ADDRESS OF FILE MUTEX USE COUNT CELLS
000000C8 PROCEDURE DIVISION RETURNING VALUE

*** VARIABLE PORTION OF DSA ***

000000D0 BACKSTORE CELLS FOR SYMBOLIC REGISTERS
000000E0 VARIABLE-LENGTH CELLS
000000F8 VARIABLE NAME (VN) CELLS FOR PERFORM
00001130 PERFORM SAVE CELLS
00001170 TEMPORARY STORAGE-2

Register Save Area is a standard operating system save area where the registers are saved GPR 14 to GPR 12 at an offset of +X'0C' into the save area.

TGT WILL BE ALLOCATED FOR 00003F40 BYTES
SPEC-REG WILL BE ALLOCATED FOR 0000007E BYTES
WRK-STOR WILL BE ALLOCATED FOR 00F44255 BYTES
DSA WILL BE ALLOCATED FOR 00000270 BYTES
CONSTANT GLOBAL TABLE FOR DYNAMIC STORAGE INITIALIZATION AT LOCATION 002868
INITD CODE FOR DYNAMIC STORAGE INITIALIZATION BEGINS AT LOCATION 002A80 FOR LENGTH 0000C4
### Task Global Table

#### **TGT MEMORY MAP**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>RESERVED - 72 BYTES</td>
</tr>
<tr>
<td>000048</td>
<td>TGT IDENTIFIER</td>
</tr>
<tr>
<td>00004C</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>000050</td>
<td>TGT LEVEL INDICATOR</td>
</tr>
<tr>
<td>000051</td>
<td>RESERVED - 3 BYTES</td>
</tr>
<tr>
<td>000054</td>
<td>32 BIT SWITCH</td>
</tr>
<tr>
<td>000058</td>
<td>POINTER TO RUNCOM</td>
</tr>
<tr>
<td>00005C</td>
<td>POINTER TO COBVEC</td>
</tr>
<tr>
<td>000060</td>
<td>POINTER TO PROGRAM DYNAMIC BLOCK TABLE</td>
</tr>
<tr>
<td>000064</td>
<td>NUMBER OF FCB’S</td>
</tr>
<tr>
<td>000068</td>
<td>WORKING-STORAGE LENGTH</td>
</tr>
<tr>
<td>00006C</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>000070</td>
<td>ADDRESS OF IGZESMG WORK AREA</td>
</tr>
<tr>
<td>000074</td>
<td>ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR)</td>
</tr>
<tr>
<td>000078</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007A</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007C</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007E</td>
<td>MERGE FILE NUMBER</td>
</tr>
<tr>
<td>000080</td>
<td>ADDRESS OF CEL COMMON ANCHOR AREA</td>
</tr>
<tr>
<td>000084</td>
<td>LENGTH OF TGT</td>
</tr>
<tr>
<td>000088</td>
<td>RESERVED - 1 SINGLE BYTE FIELD</td>
</tr>
<tr>
<td>000089</td>
<td>PROGRAM MASK USED BY THIS PROGRAM</td>
</tr>
<tr>
<td>00008A</td>
<td>RESERVED - 2 SINGLE BYTE FIELDS</td>
</tr>
<tr>
<td>00008C</td>
<td>NUMBER OF SECONDARY FCB CELLS</td>
</tr>
<tr>
<td>000090</td>
<td>LENGTH OF THE ALTER VNI(VNI) VECTOR</td>
</tr>
<tr>
<td>000094</td>
<td>COUNT OF NESTED PROGRAMS IN COMPILE UNIT</td>
</tr>
<tr>
<td>000098</td>
<td>DDNAME FOR DISPLAY OUTPUT</td>
</tr>
<tr>
<td>0000A0</td>
<td>RESERVED - 8 BYTES</td>
</tr>
<tr>
<td>0000A8</td>
<td>POINTER TO COM-REG SPECIAL REGISTER</td>
</tr>
<tr>
<td>0000AC</td>
<td>RESERVED - 52 BYTES</td>
</tr>
<tr>
<td>0000E0</td>
<td>ALTERNATE COLLATING SEQUENCE TABLE PTR.</td>
</tr>
<tr>
<td>0000E4</td>
<td>ADDRESS OF SORT G.N. ADDRESS BLOCK</td>
</tr>
<tr>
<td>0000E8</td>
<td>ADDRESS OF PGT</td>
</tr>
<tr>
<td>0000EC</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>0000F0</td>
<td>POINTER TO 1ST IPCB</td>
</tr>
<tr>
<td>0000F4</td>
<td>ADDRESS OF THE CLLE FOR THIS PROGRAM</td>
</tr>
<tr>
<td>0000F8</td>
<td>POINTER TO ABEND INFORMATION TABLE</td>
</tr>
<tr>
<td>0000FC</td>
<td>POINTER TO TEST INFO FIELDS IN THE TGT</td>
</tr>
<tr>
<td>000100</td>
<td>ADDRESS OF START OF COBOL PROGRAM</td>
</tr>
<tr>
<td>000104</td>
<td>POINTER TO ALTER VNI’S IN CGT</td>
</tr>
<tr>
<td>000108</td>
<td>POINTER TO ALTER VNI’S IN TGT</td>
</tr>
<tr>
<td>00010C</td>
<td>POINTER TO FIRST PBL IN THE PGT</td>
</tr>
<tr>
<td>000110</td>
<td>POINTER TO FIRST FCB CELL</td>
</tr>
<tr>
<td>000114</td>
<td>WORKING-STORAGE ADDRESS</td>
</tr>
<tr>
<td>000118</td>
<td>POINTER TO FIRST SECONDARY FCB CELL</td>
</tr>
<tr>
<td>00011C</td>
<td>POINTER TO STATIC CLASS INFO BLOCK 1</td>
</tr>
<tr>
<td>000120</td>
<td>POINTER TO STATIC CLASS INFO BLOCK 2</td>
</tr>
</tbody>
</table>

#### **VARIABLE PORTION OF TGT**

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000124</td>
<td>TGT OVERFLOW AREA ADCONS</td>
</tr>
<tr>
<td>000130</td>
<td>BASE LOCATORS FOR SPECIAL REGISTERS</td>
</tr>
<tr>
<td>000138</td>
<td>BASE LOCATORS FOR WORKING-STORAGE</td>
</tr>
<tr>
<td>003E4C</td>
<td>BASE LOCATORS FOR LINKAGE-SECTION</td>
</tr>
<tr>
<td>003E58</td>
<td>CLLE ADDR. CELLS FOR CALL LIT. SUB-PGMS.</td>
</tr>
<tr>
<td>003F2C</td>
<td>INTERNAL PROGRAM CONTROL BLOCKS</td>
</tr>
</tbody>
</table>
Types of Dumps

- **Transaction Dumps**
  - DFHDMPA
  - DFHDMPB

- **System Dumps (SDUMP)**
  - SYS1.DUMPxx
System Dumps

- Processed using IPCS
  - Will be reviewed in another section
- The DSN are now pre-defined by the system programmer
- Allocate sufficient space to capture CICS dumps especially for large systems and RLS systems that may require multiple regions to be dumped
- Check the sizing with each new CICS release that is installed (MAXSPACE)
Transaction Dumps

- Transaction dumps are sent to the data set named DFHDMPA or DFHDMPB
  - These files are defined at start-up
  - Can be automatically or manually switched
  - Should be properly sized for installation dump activity
  - Specific dump codes can be suppressed by using CEMT SET TRD transaction
  - To obtain a transaction dump, the dump data set should be closed and a batch procedure should be executed
Transaction Dumps

- Sample JCL and batch procedure

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT ACT.CTREK.V42.TEST.SOURCE(DFHDUMP) - 01.08 Columns 00001 00072
Command => Scroll => CSR
****** ***************************** Top of Data ****************************

000100 //DFHU670 JOB CTREK,'SISTEMAS',
000300 // NOTIFY=&SYSUID.,REGION=0M,
000400 // CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
000500 //STEP1 EXEC PGM=DFHU670,PARM=''
000600 //STEP1B DD DSN=CICSTS42.CICS.SDFHLOAD,DISP=SHR
000700 //DFHDMPDS DD DISP=SHR,DSN=CICSTS42.CICS.DFHDMPA
000800 //DFHTINDX DD SYSOUT=X
000900 //DFHPRINT DD SYSOUT=X,DCB=BLKSIZE=133
001000 //SYSPRINT DD SYSOUT=X
001010 //SYSIN DD *
001200 END
001300 /*
****** **************************** Bottom of Data ****************************

F1=Help  F2=Split  F3=Exit  F5=Rfind  F6=Retrieve  F7=Up
F8=Down  F9=Swap  F10=Left  F11=Right  F12=Retrieve
Transaction Dumps

- Transaction dumps provide a snapshot at the time the cancelation occurred
- There are many control blocks printed most of which are not used or undocumented
- The following charts review those segments of the dump which are important in debugging
Cancelation Information

The type of CICS cancelation is provided on the first line via a code (e.g., ASRA) followed by the transaction that was involved. The Symptoms String usually identifies the program CICS believes was in control when the error occurred.

The PSW and the contents of the registers when the cancelation occurred are very important because the PSW tells you where the error occurred and the registers tell you where the important COBOL control blocks are located. In the case of a protection exception you may want to know in what key you were executing when the error occurred. In addition to the PSW information you want to annotate the information in the full word following the PSW that indicates the length of the instruction that caused the cancelation and will be used to adjust the PSW address (first half word) and the type of cancelation (second half word).

The registers at the last EXEC Command may be important when debugging loops.
PSW Adjustment

- Adjust the PSW address by the length of the instruction causing the cancelation
  - PSW Address 9BAB90F6
  - Adjustment - 6
  - Actual Abend 9BAB90F0

The PSW in a transaction dump is normally pointing to the Next Instruction to be executed – exceptions are covered on the next page
PSW Adjustment

- There are times when the PSW is actually pointing to the instruction that caused the Program Check.
- These cases are usually reported as S0C4 cancelations with a reason code:
  - 0010 Segment-translation exception
  - 0011 Page-translation exception
  - 0038 ASCE-type exception
  - 0039 Region-first-translation exception
  - 003A Region-second-translation exception
  - 003B Region-third-translation exception
- In these cases the PSW points to the instruction that caused the cancelation and does not require a PSW adjustment.
Computing the Program Offset

- Now that you have the address where the cancelation occurred, you now need to determine the program involved and the offset into the program where the cancelation occurred.
- The information on the 1st page identified a possible program candidate – UVPUTSM.
- This is the program which CICS believed had control when the cancelation occurred.
- However, CICS is not aware of any internal CALLs the program might have made.
Computing the Offset

- There are several places where you can find the program address
  - COBOL Program’s TGT + X’100’
  - Module Listing at the end of the dump
  - The Program Information in the dump
  - KE Stack owned by DFHPGPG using the PLCB information (not recommended)
  - Find the CICS calculated offset in STCA (not recommended)
Using the TGT

The entry point of the COBOL program can be found at TGT + X'100'. There are several ways of locating the TGT. A simple way is using the contents of GPR 09. You can verify if the address in GPR 09 is pointing the TGT by looking at the interpreted part of the dump and see if the '3TGT' eye-catcher.

Program EP Address = X'1AA5B570' + X'100' = X'1AA5B670'
Using Module Listing

The module listing provides the program names currently in the CICS system. The program was identified in the Symptom String as UVPUTSM. You need to scan the module list to locate the program identified on the first page of the dump.

<table>
<thead>
<tr>
<th>LOAD PT</th>
<th>NAME</th>
<th>ENTRY PT</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1ADE5000</td>
<td>DFHZNAC</td>
<td>1ADE5114</td>
<td>0000BAE0</td>
</tr>
<tr>
<td>1ADF1000</td>
<td>DFHWBXN</td>
<td>1ADF1028</td>
<td>00006730</td>
</tr>
<tr>
<td>1ADF7800</td>
<td>DFHEDAP</td>
<td>1ADF7828</td>
<td>00002128</td>
</tr>
<tr>
<td>1ADF9A00</td>
<td>DFHCESC</td>
<td>1ADF9A28</td>
<td>000015F0</td>
</tr>
<tr>
<td>1ADFB000</td>
<td>DFHEMTP</td>
<td>1ADFB028</td>
<td>00002158</td>
</tr>
<tr>
<td>1AE00000</td>
<td>CEEPLPKA</td>
<td>1AE000000</td>
<td>001F48F8</td>
</tr>
<tr>
<td>1AFF5000</td>
<td>DFHEITSP</td>
<td>1AFF5000</td>
<td>000089E0</td>
</tr>
<tr>
<td>1B000000</td>
<td>CEEEV003</td>
<td>1B000000</td>
<td>005525E8</td>
</tr>
<tr>
<td>1B581700</td>
<td>DFHZATMF</td>
<td>1B581728</td>
<td>00000868</td>
</tr>
<tr>
<td>1B582000</td>
<td>DFHLUP</td>
<td>1B582028</td>
<td>00010160</td>
</tr>
<tr>
<td>1B592200</td>
<td>DFHEDAD</td>
<td>1B592228</td>
<td>0003E8A8</td>
</tr>
<tr>
<td>1B600000</td>
<td>CEEEV011</td>
<td>1B600000</td>
<td>001A5A00</td>
</tr>
<tr>
<td>1B7A6114</td>
<td>DFHAMP</td>
<td>1B7A6114</td>
<td>00039C80</td>
</tr>
<tr>
<td>1B800000</td>
<td>DFHCNV</td>
<td>1B800000</td>
<td>001D3E18</td>
</tr>
<tr>
<td>1B9D3F28</td>
<td>DFHMTD</td>
<td>1B9D3F28</td>
<td>00002870</td>
</tr>
<tr>
<td>1BA00000</td>
<td>EZACIC20</td>
<td>1BA00000</td>
<td>00006868</td>
</tr>
<tr>
<td>1BA00690</td>
<td>EZACIC21</td>
<td>1BA00688</td>
<td>00001968</td>
</tr>
<tr>
<td>1BA02000</td>
<td>KVPTREI</td>
<td>1BA02020</td>
<td>00002F50</td>
</tr>
<tr>
<td>1BA16000</td>
<td>KVPTREH</td>
<td>1BA16028</td>
<td>000043F8</td>
</tr>
<tr>
<td>1BAB8000</td>
<td>UVPUTSM</td>
<td>1BAB8020</td>
<td>00003798</td>
</tr>
<tr>
<td>1BABB7A0</td>
<td>UVMTEST</td>
<td>1BABB7A0</td>
<td>00038880</td>
</tr>
</tbody>
</table>

END OF CICS TRANSACTION DUMP
Using Program Information

<table>
<thead>
<tr>
<th>Program Name</th>
<th>UVPUTSM</th>
<th>Invoking Program</th>
<th>CICS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Point</td>
<td>1BAB8000</td>
<td>Program Length</td>
<td>00003798</td>
</tr>
<tr>
<td>Entry Point</td>
<td>9Babar8020</td>
<td>Addressing Mode</td>
<td>AMODE 31</td>
</tr>
<tr>
<td>Language Defined</td>
<td>COBOL</td>
<td>Language Deduced</td>
<td>COBOL II</td>
</tr>
<tr>
<td>Commarea Address</td>
<td>00000000</td>
<td>Commarea Length</td>
<td>00000000</td>
</tr>
<tr>
<td>Execution Key</td>
<td>USER</td>
<td>Data Location</td>
<td>ANY</td>
</tr>
<tr>
<td>Concurrency</td>
<td>QUASIRENT</td>
<td>Api</td>
<td>CICSAPI</td>
</tr>
<tr>
<td>Runtime</td>
<td>LE370</td>
<td>Environment</td>
<td>User application</td>
</tr>
</tbody>
</table>
Using the DFHPGPG Information

- Can be located by doing a find
  - DFHPGPLCB – locate the one for the CICS identified program (UVPUTSM)

Program Load Point
Program Entry Point
CICS Stub – X’20’ Bytes Long

Note: The difference between the Entry Point and the Load Point is the CICS Stub at the beginning of the program which in this case is X’20’ bytes long.
Computing the Offset

The offset is simply the distance from the beginning of the program to where the cancelation occurred.

Entry Point – beginning of the program

PSW – where the cancelation occurred

Compute Offset = Adjusted PSW Address minus the Entry Point Address

Once you have the offset, you need to look at the program compilation listing.
Offset Using the STCA

The CICS computed offset is based on the Program Load Point and does not adjust the PSW by the length of the instruction causing the cancelation. In order to get the correct displacement you need to subtract the CICS stub length and the length of the instruction causing the program check.

<table>
<thead>
<tr>
<th>CICS Determined Offset (STCA)</th>
<th>000010F6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minus Length of Instruction Causing Cancelation</td>
<td>6</td>
</tr>
<tr>
<td>Minus Length of CICS Stub</td>
<td>20</td>
</tr>
<tr>
<td>Real Offset in program</td>
<td>000010D0</td>
</tr>
</tbody>
</table>
Locating the Failing Instruction

Now that you have determined the offset into the program where the error occurred, go to the COBOL program listing of the identified program (UVPUTSM) and locate the Procedure Division Map.

The Procedure Division Map comes into two flavors:
--- Assembler Listing
--- Condensed Listing

Locate the instruction that cancelled using the offset – identify the source sequence #

Using the source sequence number of the instruction that caused the cancelation, go to the source listing looking for 002069.
Locating the Source Instruction

Using the sequence number from the Procedure Division Map find the source instruction that caused the program check – find the instruction at 002069

Note: No VALUE clause specified. Further analysis of the Procedure Division shows that the field was not initialized. Result = S0C7
Locating the Data in the Dump

- Data in a COBOL program are either in the WORKING-STORAGE of the LINKAGE Section of the program.
- Addressing in the hardware requires a base register for every 4 KB of data or instructions.
- The addressing of the areas is accomplished by using:
  - BLW – Base Locator for Working Storage
  - BLL – Base Locator for Linkage Storage
- These addressing cells are kept in the variable section of the TGT.
### Task Global Table

*** TGT MEMORY MAP ***

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>RESERVED - 72 BYTES</td>
</tr>
<tr>
<td>000048</td>
<td>TGT IDENTIFIER</td>
</tr>
<tr>
<td>00004C</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>000050</td>
<td>TGT LEVEL INDICATOR</td>
</tr>
<tr>
<td>000051</td>
<td>RESERVED - 3 BYTES</td>
</tr>
<tr>
<td>000054</td>
<td>32 BIT SWITCH</td>
</tr>
<tr>
<td>000058</td>
<td>POINTER TO RUNCOM</td>
</tr>
<tr>
<td>00005C</td>
<td>POINTER TO COBVEC</td>
</tr>
<tr>
<td>000060</td>
<td>POINTER TO PROGRAM DYNAMIC BLOCK TABLE</td>
</tr>
<tr>
<td>000064</td>
<td>NUMBER OF FCB'S</td>
</tr>
<tr>
<td>000068</td>
<td>WORKING-STORAGE LENGTH</td>
</tr>
<tr>
<td>00006C</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>000070</td>
<td>ADDRESS OF IGZESMG WORK AREA</td>
</tr>
<tr>
<td>000074</td>
<td>ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR)</td>
</tr>
<tr>
<td>000078</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007A</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007C</td>
<td>RESERVED - 2 BYTES</td>
</tr>
<tr>
<td>00007E</td>
<td>MERGE FILE NUMBER</td>
</tr>
<tr>
<td>000080</td>
<td>ADDRESS OF CEL COMMON ANCHOR AREA</td>
</tr>
<tr>
<td>000084</td>
<td>LENGTH OF TGT</td>
</tr>
<tr>
<td>000088</td>
<td>RESERVED - 1 SINGLE BYTE FIELD</td>
</tr>
<tr>
<td>000089</td>
<td>PROGRAM MASK USED BY THIS PROGRAM</td>
</tr>
<tr>
<td>00008A</td>
<td>RESERVED - 2 SINGLE BYTE FIELDS</td>
</tr>
<tr>
<td>00008C</td>
<td>NUMBER OF SECONDARY FCB CELLS</td>
</tr>
<tr>
<td>000090</td>
<td>LENGTH OF THE ALTER VN(VNI) VECTOR</td>
</tr>
<tr>
<td>000094</td>
<td>COUNT OF NESTED PROGRAMS IN COMPILe UNIT</td>
</tr>
<tr>
<td>000098</td>
<td>DDNAME FOR DISPLAY OUTPUT</td>
</tr>
<tr>
<td>0000A0</td>
<td>RESERVED - 8 BYTES</td>
</tr>
<tr>
<td>0000A8</td>
<td>POINTER TO COM-REG SPECIAL REGISTER</td>
</tr>
<tr>
<td>0000AC</td>
<td>RESERVED - 52 BYTES</td>
</tr>
<tr>
<td>0000E0</td>
<td>ALTERNATE COLLATING SEQUENCE TABLE PTR.</td>
</tr>
<tr>
<td>0000E4</td>
<td>ADDRESS OF SORT G.N. ADDRESS BLOCK</td>
</tr>
<tr>
<td>0000E8</td>
<td>ADDRESS OF PGT</td>
</tr>
<tr>
<td>0000EC</td>
<td>RESERVED - 4 BYTES</td>
</tr>
<tr>
<td>0000F0</td>
<td>POINTER TO 1ST IPCB</td>
</tr>
<tr>
<td>0000F4</td>
<td>ADDRESS OF THE CLLE FOR THIS PROGRAM</td>
</tr>
<tr>
<td>0000F8</td>
<td>POINTER TO ABEND INFORMATION TABLE</td>
</tr>
<tr>
<td>0000FC</td>
<td>POINTER TO TEST INFO FIELDS IN THE TGT</td>
</tr>
<tr>
<td>000100</td>
<td>ADDRESS OF START OF COBOL PROGRAM</td>
</tr>
<tr>
<td>000104</td>
<td>POINTER TO ALTER VNI'S IN CGT</td>
</tr>
<tr>
<td>000108</td>
<td>POINTER TO ALTER VN'S IN TGT</td>
</tr>
<tr>
<td>00010C</td>
<td>POINTER TO FIRST PBL IN THE PGT</td>
</tr>
<tr>
<td>000110</td>
<td>POINTER TO FIRST FCB CELL</td>
</tr>
<tr>
<td>000114</td>
<td>WORKING-STORAGE ADDRESS</td>
</tr>
<tr>
<td>000118</td>
<td>POINTER TO FIRST SECONDARY FCB CELL</td>
</tr>
<tr>
<td>00011C</td>
<td>POINTER TO STATIC CLASS INFO BLOCK 1</td>
</tr>
<tr>
<td>000120</td>
<td>POINTER TO STATIC CLASS INFO BLOCK 2</td>
</tr>
</tbody>
</table>

*** VARIABLE PORTION OF TGT ***

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000124</td>
<td>TGT OVERFLOW AREA ADCONS</td>
</tr>
<tr>
<td>000130</td>
<td>BASE LOCATORS FOR SPECIAL REGISTERS</td>
</tr>
<tr>
<td>000138</td>
<td>BASE LOCATORS FOR WORKING-STORAGE</td>
</tr>
<tr>
<td>0003E4C</td>
<td>BASE LOCATORS FOR LINKAGE-SECTION</td>
</tr>
<tr>
<td>003E58</td>
<td>CLLE ADDR. CELLS FOR CALL LIT. SUB-PGMS.</td>
</tr>
<tr>
<td>003F2C</td>
<td>INTERNAL PROGRAM CONTROL BLOCKS</td>
</tr>
</tbody>
</table>
COBOL Address Cells--BLW

BLWs start at the beginning of WS and are incremented by +X’1000’ (4KB) for each cell. The storage is contiguous.
Locating the BLW Cells

Remember the address of where WS starts is at a displacement of TGT + X'114' and has be equal to the contents of where you locate BLW 0

TGT Address 1AA5B570
BLW Disp. In TGT 138
Location of BLW in TGT 1AA5B6A8
Data Division—WS

Displacement from the 01 Level

Displacement from the BLW Cell

BLW displacements are given in Hexadecimal and will be the actual value used in the instruction
Locate the Failing Field

• The instruction that failed indicated that the bad data was in a field called ‘WS-DUMP’
• The information from the listing gave us the details we needed to locate the field in the dump

```
000123 05 WS-DUMP PIC S9(5) COMP-3. BLW=00000+256,0000026 3P
```

```
BLW 0 1BE000C0
DISPLACEMENT 256
LOCATION OF FIELD 1BE00316
LENGTH 3 BYTES PACKED
```
Locating the Failing Field

The field contains low-values which is why the program check occurred
COBOL Address Cells--BLL

- **BLL 0** – Special cell that contains low-values
- **BLL 1** – Points to DFHEIB
- **BLL 2** – Points to the DFHCOMMAREA

01 Levels may be close to each other but not necessarily contiguous

Next 01 Level in the Linkage Section would use BLL 4
Locating the BLL Cells

TGT Address | 1AA5B570
BLL Disp. In TGT | 3E4C
Location of BLL in TGT | 1AA5F3BC

BLL0 – Unused
BLL1 – DFHEIB
BLL2 -- COMMAREA

BLL0
Data Division—LS

001981  LINKAGE SECTION.
001984
001985  01 dfhehblk.
001986  02 eibtime comp-3 pic s9(7).
001987  02 eibdata comp-3 pic s9(7).
001988  02 eibtrvid pic x(4).
001989  02 eibtaskn comp-3 pic s9(7).
001990  02 eiberrid pic x(4).
001991  02 dfheigdi comp pic s9(4).
001992  02 eibcppsn comp pic s9(4).
001993  02 eibcalen comp pic s9(4).
001994  02 eibaid pic x(1).
001995  02 eibfn pic x(2).
001996  02 eibrcode pic x(8).
001997  02 eibdgs pic x(8).
001998  02 eibrequid pic x(8).
001999  02 eibrsrce pic x(8).
002000  02 eibsync pic x(8).
002001  02 eibfree pic x(1).
002002  02 eibrecv pic x(1).
002003  02 eibf101 pic x(1).
002004  02 eibatt pic x(1).
002005  02 eibec pic x(1).
002006  02 eibfmh pic x(1).
002007  02 eibcompl pic x(1).
002008  02 eibrrg pic x(1).
002009  02 eibconfl pic x(1).
002010  02 eiberr pic x(1).
002011  02 eiberrcd pic x(4).
002012  02 eibsynrh pic x(1).
002013  02 eibnodat pic x(1).
002014  02 eibresp comp pic s9(8).
002015  02 eibresp2 comp pic s9(8).
002016  02 eibfr050 pic x(1).
002017  01 DFHCOMAREA  PIC X.

Assigned BLL Cell

BLL=00001+000  OCL85
BLL=00001+000,00000004  4P
BLL=00001+008,00000008  4C
BLL=00001+00C,0000000C  4P
BLL=00001+010,00000010  4C
BLL=00001+014,00000014  2C
BLL=00001+016,00000016  2C
BLL=00001+018,00000018  2C
BLL=00001+01A,0000001A  1C
BLL=00001+023,00000023  8C
BLL=00001+028,00000028  8C
BLL=00001+033,00000033  8C
BLL=00001+038,00000038  8C
BLL=00001+03C,0000003C  1C
BLL=00001+03E,0000003E  1C
BLL=00001+03F,0000003F  1C
BLL=00001+040,00000040  1C
BLL=00001+041,00000041  1C
BLL=00001+042,00000042  1C
BLL=00001+043,00000043  1C
BLL=00001+044,00000044  1C
BLL=00001+045,00000045  1C
BLL=00001+046,00000046  4C
BLL=00001+04A,0000004A  1C
BLL=00001+04B,0000004B  1C
BLL=00001+050,00000050  4C
BLL=00001+054,00000054  1C
BLL=00002+000  1C
Addressing LS

- The DFHEIB and DFHCOMMAREA are provided addressability via:
  - PROCEDURE DIVISION using dfheiblk dfhcommarea.
  - As a BLL Cell is required for each 4 KB of data, any 01 level field that is more than 4 KB in size will receive a BLL Cell for every 4 KB of storage rounded up.
What About the Trace?

- The Internal Trace Table can be used to obtain additional information
  - In the case of a program check, sufficient information was provided in the transaction dump that can be used to resolve the problem
  - However, there may be cases where you could use the Trace Table to see if a prior error occurred that may have led to the program check
- Two types of Trace Entries
  - Abbreviated Trace entry
  - Full Trace entry
- Exception entries can be found by issuing
  - F *EXC*
  - There may be several *EXC* entries
ASRA Debugging Cookbook

**PART 1**

- Determine the type of program check that occurred (e.g., S0C7, S0C4 etc.)
- Review the information on the first page of the dump
  - Transaction Id
  - PSW information and associated registers
    - *Get the PSW address and the instruction length of the failing instruction*
    - *Adjust the PSW address using the instruction length*
  - Locate the failing program
- Find the entry point address of the failing program
- Determine the offset into the program of the cancelling instruction
  - *Offset = Adjusted PSW Address – Program Entry point*
ASRA Debugging Cookbook

• **PART 2**
  • Get program listing and locate the Procedure Division Map
    • Assembler Listing
    • Condensed Listing
  • Determine the failing instruction using the computed offset from Part 1
  • Determine the source instruction (verb) causing the problem
  • Review instruction and operands to determine cause of the program check
    • Identify affected fields
Locating a Field Cookbook

- Locate the TGT in the dump
  - General Purpose Register 09 → TGT
  - General Purpose Register 13 → DSA
    - DSA + X’5C’ → TGT
- Ensure that you are looking at a TGT by locating the eye-catcher ‘3TGT’ at +X’48’
- Locate the COBOL program listing
  - Find the TGT layout at the end of the listing
    - Locate the offset to the BLW Cells (Working Storage)
    - Locate the offset to the BLL Cells (Linkage Section)
Locating a Field Cookbook

- Locate the TGT in the dump
  - General Purpose Register 09 → TGT
  - General Purpose Register 13 → DSA
    - DSA + X'5C’ → TGT
- Ensure that you are looking at a TGT by locating the eye-catcher ‘3TGT’ at +X’48’
- Locate the COBOL program listing
  - Find the TGT layout at the end of the listing
    - Locate the offset to the BLW Cells (Working Storage)
    - Locate the offset to the BLL Cells (Linkage Section)
  - Find the affected fields in the listing
    - Identify the BLW/BLL assigned, the displacement and the length of each field
Locating a Field Cookbook

• Locate the appropriate BLW/BLL cell in the dump
  • Using this content of the BLW/BLL cell add the field displacement
  • The result is the address of where the field is located in the dump
  • Locate the field in the dump for the length obtained in the COBOL listing
• If the cancelling instruction is a two operand field, find the other field in the dump
Closing

• When debugging a COBOL program that resulted in an ASRA cancellation, get the needed information from the dump.

• The two most important control blocks needed to debug a COBOL program are:
  • TGT – GPR 09 \( \rightarrow \) contains the entry point address of the program, the beginning Working-Storage address and the BLW/BLL cells required to locate fields in a dump.
  • DSA – GPR 13 \( \rightarrow \) contains the task’s registers and a pointer to the TGT.

• The techniques reviewed can be used to resolve any ASRA cancellation in addition to a S0C7.