

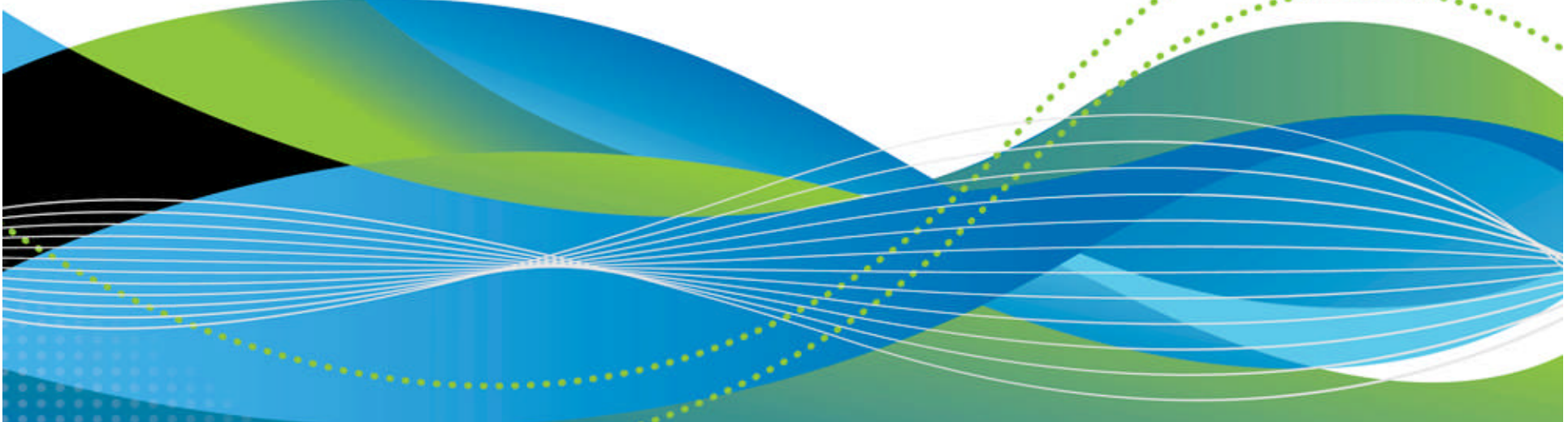
CICS TS Tutorial -- Transaction Dump Analysis

Eugene S Hudders
C\TREK Corporation
ehudders@ctrek.com

787-397-4150

Session 10294

Thursday 8:00 AM – March 15, 2012



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
 - Cancellation 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

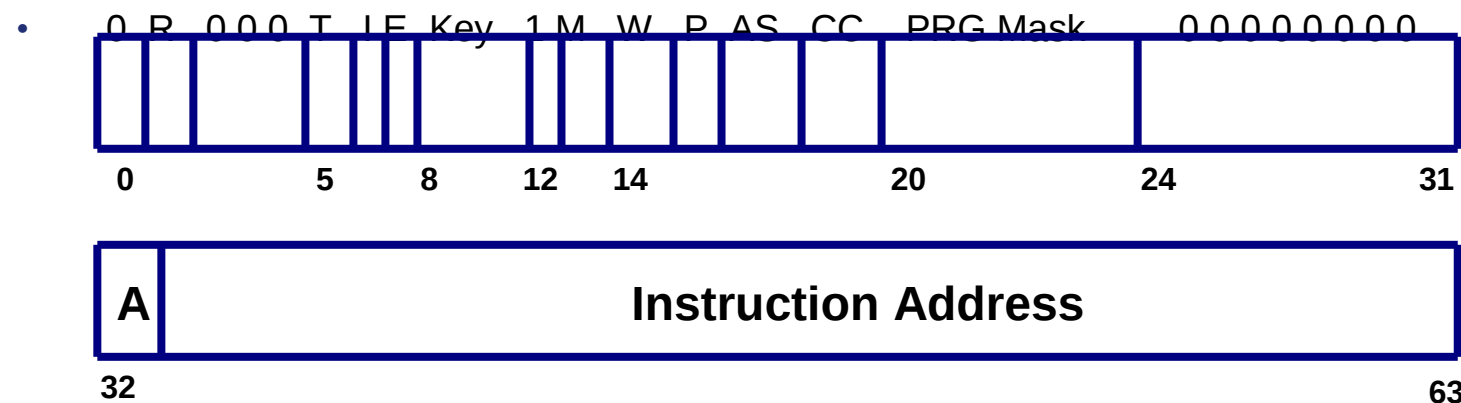
- There are program checks associated with virtual storage addressing exceptions which are treated as protection exceptions (S0C4)
 - 0010 Segment-translation exception
 - 0011 Page-translation exception
 - 0038 ASCE-type exception
 - 0039 Region-first-translation exception
 - 003A Region-second-translation 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

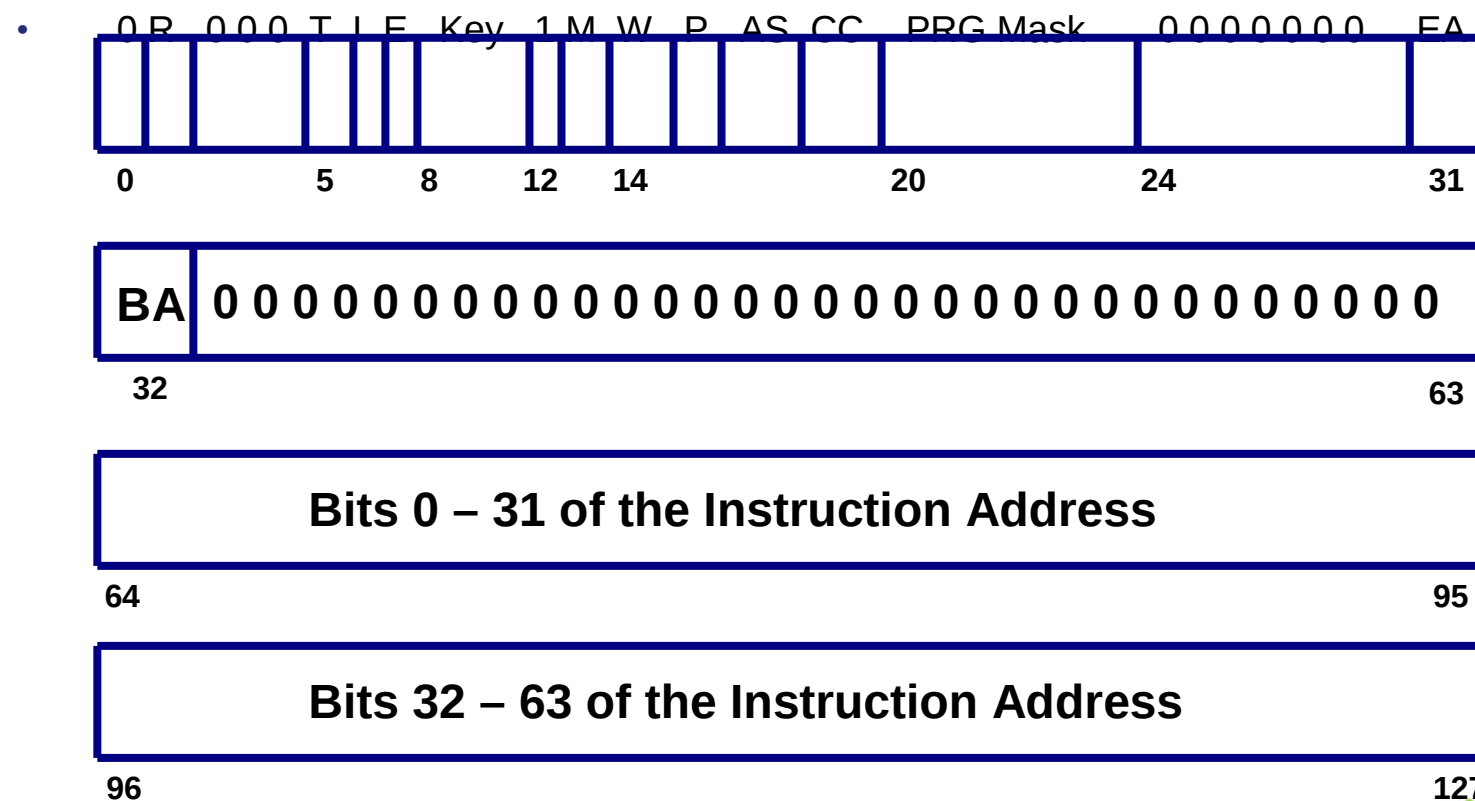


Bit 12 – indicates ESA/390 Mode

Bit 32 – Addressing Mode (1 = 31-bit addressing)

Program Status Word

- z/Architecture PSW (Bit 12 = 0)



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

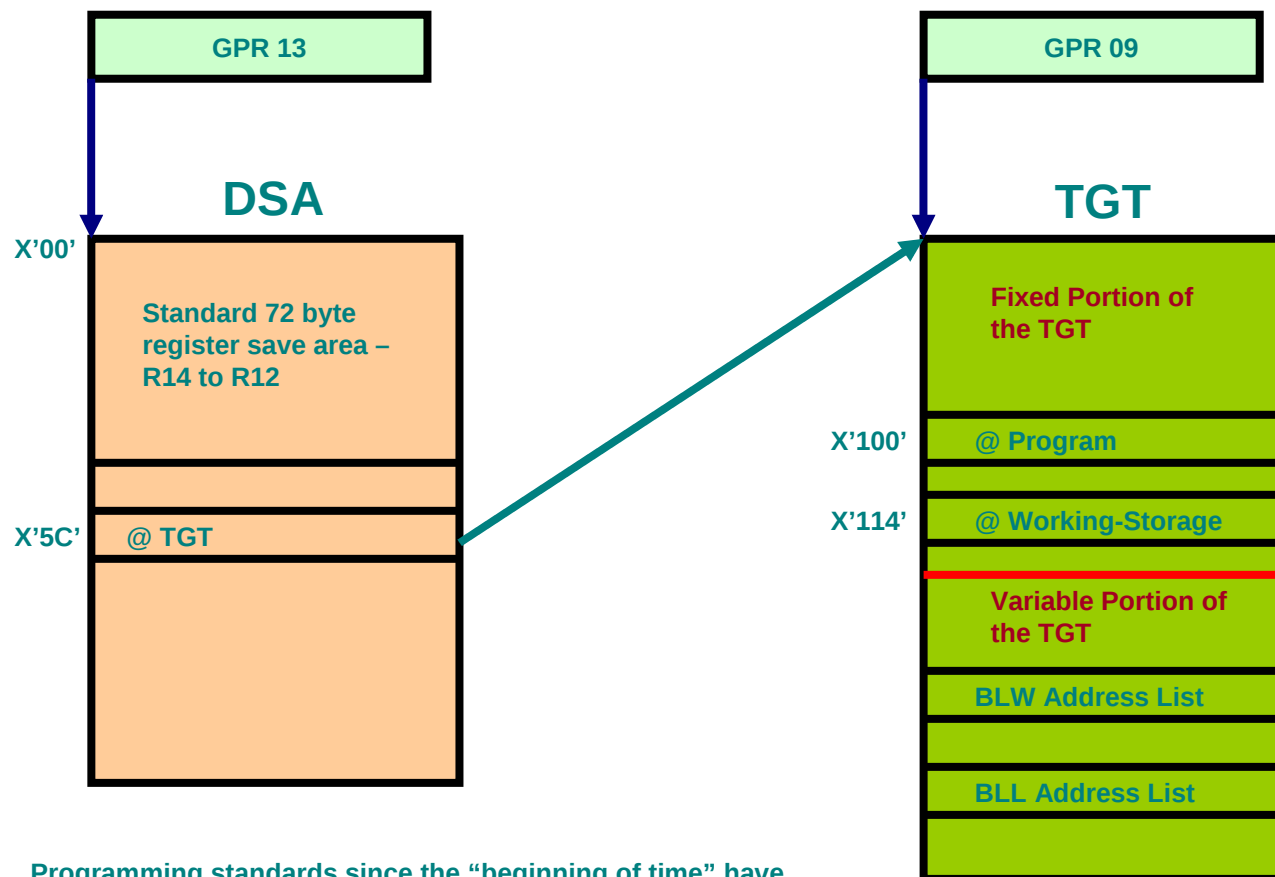
Program Status Word

- 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

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

*** VARIABLE PORTION OF DSA ***

000000D0 BACKSTORE CELLS FOR SYMBOLIC REGISTERS
 000000E0 VARIABLE-LENGTH CELLS
 000000F8 VARIABLE NAME (VN) CELLS FOR PERFORM
 00000130 PERFORM SAVE CELLS
 00000170 TEMPORARY STORAGE-2

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 ***

TGTLOC

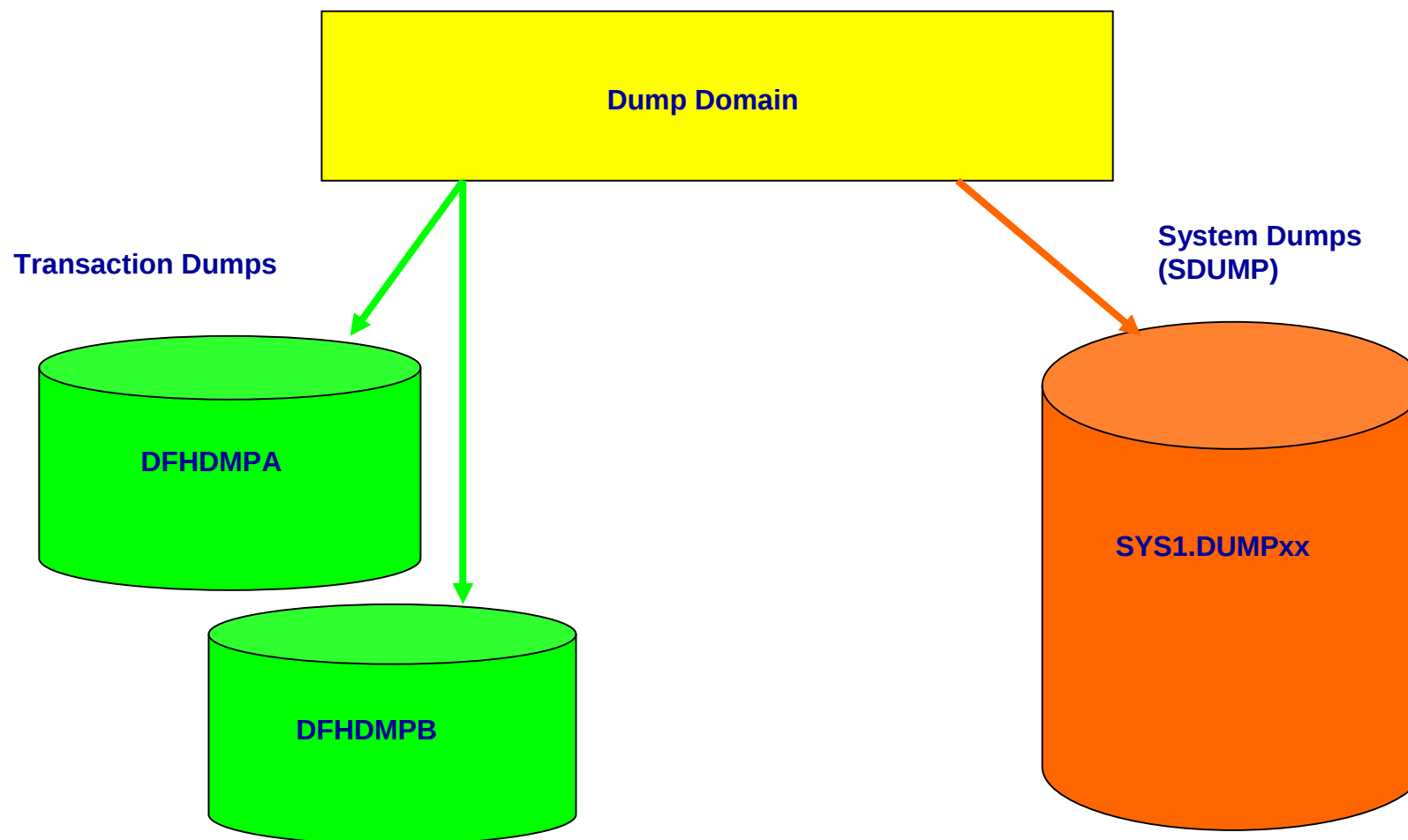
000000 RESERVED - 72 BYTES
000048 TGT IDENTIFIER
 00004C RESERVED - 4 BYTES
 000050 TGT LEVEL INDICATOR
 000051 RESERVED - 3 BYTES
 000054 32 BIT SWITCH
 000058 POINTER TO RUNCOM
 00005C POINTER TO COBVEC
 000060 POINTER TO PROGRAM DYNAMIC BLOCK TABLE
 000064 NUMBER OF FCB'S
 000068 WORKING-STORAGE LENGTH
 00006C RESERVED - 4 BYTES
 000070 ADDRESS OF IGZESMG WORK AREA
 000074 ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR)
 000078 RESERVED - 2 BYTES
 00007A RESERVED - 2 BYTES
 00007C RESERVED - 2 BYTES
 00007E MERGE FILE NUMBER
 000080 ADDRESS OF CEL COMMON ANCHOR AREA
 000084 LENGTH OF TGT
 000088 RESERVED - 1 SINGLE BYTE FIELD
 000089 PROGRAM MASK USED BY THIS PROGRAM
 00008A RESERVED - 2 SINGLE BYTE FIELDS
 00008C NUMBER OF SECONDARY FCB CELLS
 000090 LENGTH OF THE ALTER VN(VNI) VECTOR
 000094 COUNT OF NESTED PROGRAMS IN COMPILE UNIT
 000098 DDNAME FOR DISPLAY OUTPUT
 0000A0 RESERVED - 8 BYTES
 0000A8 POINTER TO COM-REG SPECIAL REGISTER
 0000AC RESERVED - 52 BYTES
 0000E0 ALTERNATE COLLATING SEQUENCE TABLE PTR.
 0000E4 ADDRESS OF SORT G.N. ADDRESS BLOCK

0000E8 ADDRESS OF PGT
 0000EC RESERVED - 4 BYTES
 0000F0 POINTER TO 1ST IPCB
 0000F4 ADDRESS OF THE CLLE FOR THIS PROGRAM
 0000F8 POINTER TO ABEND INFORMATION TABLE
 0000FC POINTER TO TEST INFO FIELDS IN THE TGT
000100 ADDRESS OF START OF COBOL PROGRAM
 000104 POINTER TO ALTER VNI'S IN CGT
 000108 POINTER TO ALTER VN'S IN TGT
 00010C POINTER TO FIRST PBL IN THE PGT
 000110 POINTER TO FIRST FCB CELL
000114 WORKING-STORAGE ADDRESS
 000118 POINTER TO FIRST SECONDARY FCB CELL
 00011C POINTER TO STATIC CLASS INFO BLOCK 1
 000120 POINTER TO STATIC CLASS INFO BLOCK 2

*** VARIABLE PORTION OF TGT ***

000124 TGT OVERFLOW AREA ADCONS
 000130 BASE LOCATORS FOR SPECIAL REGISTERS
000138 BASE LOCATORS FOR WORKING-STORAGE
003E4C BASE LOCATORS FOR LINKAGE-SECTION
 003E58 CLLE ADDR. CELLS FOR CALL LIT. SUB-PGMS.
 003F2C INTERNAL PROGRAM CONTROL BLOCKS

Types of Dumps



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 //DFHDU670 JOB CTREK,'SISTEMAS',
•   000300 //              NOTIFY=&SYSUID.,REGION=0M,
•   000400 //              CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
•   000500 //STEP1      EXEC PGM=DFHDU670,PARM=' '
•   000600 //STEPLIB DD DSN=CICSTS42.CICS.SDFHLOAD,DISP=SHR
•   000700 //DFHMPDS DD DISP=SHR,DSN=CICSTS42.CICS.DFHMPA
•   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=Rchange  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

Cancel Code → Transaction Canceling
CICSTS42 --- CICS TRANSACTION DUMP --- CODE=ASRA TRAN=UTSM ID=1/0002 DATE=11/07/17 TIME=17:18:27 PAGE 1
SYMPTOMS= AB/UASRA PIDS/5655S9700 FLDS/DFHABAB RIDS/UVPUTSM ← Possible program causing the cancelation
CICS LEVEL = 0670
PSW & REGISTERS AT TIME OF INTERRUPT
PSW 079D0000 9BAB90F6 00060007 00000000 ← PSW Information
REGS 0-7 1AA5F3C8 1AA5A4F0 1BE000C0 1BE00040 1CD420C0 1AA500D0 1CD430C0 1BE00248 | Registers when the ABEND occurred
REGS 8-15 1BAB815C 1AA5B570 1AA5E570 1BAB8B9C 1BAB811C 1AA5A380 9BAB90E2 00000000
EXECUTION KEY 9 ← Storage Protect Key of cancelling task
The transaction was in Basespace mode
COMMAND REGISTERS AT LAST EXEC
REGS 0-7 1AA5F3C8 1AA5A4F0 1BE000C0 1BE00040 1CD420C0 1AA500D0 1CD430C0 1BE00248 | Registers at last EXEC CICS
REGS 8-15 1BAB815C 1AA5B570 1AA5E570 1BAB8B9C 1AA584C8 1AA5A380 9BAB90E2 00000000

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

GPR 09

1AA5B570

```

0000073A0  LINES TO 00007440 SAME AS ABOVE
000007460  00000000 00000000 F3E3C7E3 00000000 06000000 68030260 1AA5B1B8 0007809C *.....3TGT.....-v.....*
000007480  1AA5F4B0 00000000 00F44255 00000000 00000000 1BE00030 00000000 00000000 *.v4.....4.....*
0000074A0  1AA584C8 00003F40 00000000 00000000 00000000 00000001 E2E8E2D6 E4E34040 *.vdH... ..SYSOUT *
0000074C0  C9C7E9E2 D9E3C3C4 00000000 00000000 00000000 00000000 00000000 00000000 *IGZSRTCD.....*
0000074E0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....a.....v4..v.....*
000007500  00000000 00000000 1BAB811C 00000000 1AA5F49C 1AA5B490 1BAB8B0C 00000000 *.aq.v4...a.....*
000007520  1BAB8020 1BAB8198 1AA5F49C 1BAB8154 00000000 1BE000C0 00000000 00000000 *.ve..vN..vV.....*
000007540  00000000 1AA5C570 1AA5D570 1AA5E570 00000000 1BE00040 1BE000C0 1BE010C0 *.ve..vN..vV.....*
000007560  1BE020C0 1BE030C0 1BE040C0 1BE050C0 1BE060C0 1BE070C0 1BE080C0 1BE090C0 *.ve..vN..vV.....*
000007580  1BE0A0C0 1BE0B0C0 1BE0C0C0 1BE0D0C0 1BE0E0C0 1BE0F0C0 1BE100C0 1BE110C0 *.ve..vN..vV.....*
0000075A0  1BE120C0 1BE130C0 1BE140C0 1BE150C0 1BE160C0 1BE170C0 1BE180C0 1BE190C0 *.ve..vN..vV.....*

```

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

----- MODULE INDEX -----

LOAD PT.	NAME	ENTRY PT	LENGTH
1ADE5000	DFHZNAC	1ADE5114	0000BAE0
1ADF1000	DFHWBXN	1ADF1028	00006730
1ADF7800	DFHEDAP	1ADF7828	00002128
1ADF9A00	DFHCESC	1ADF9A28	000015F0
1ADFB000	DFHEMTP	1ADFB028	00002158
1AE00000	CEEPLPKA	1AE00000	001F48F8
1AFF5000	DFHEITSP	1AFF5000	000089E0
1B000000	CEEEV003	1B000000	005525E8
1B553000	DFHWBTC	1B553000	0002E638
1B581700	DFHZATMF	1B581728	00000868
1B582000	DFHLUP	1B582028	00010160
1B592200	DFHEDAD	1B592228	0003E8A8
1B5D2000	DFHEITMT	1B5D2000	00017170
1B600000	CEEEV011	1B600000	001A5A00
1B7A6000	DFHAMP	1B7A6114	00039C80
1B800000	DFHCCNV	1B800028	001D3E18
1B9D3F00	DFHEMTD	1B9D3F28	00028700
1BA00000	EZACIC20	1BA00028	00000688
1BA00690	EZACIC21	1BA006B8	00001968
1BA02000	KVPTREI	1BA02020	00002F50
1BA16000	KVPTREH	1BA16028	000043F8
1BAB8000	UVPUTSM	1BAB8020	00003798
1BABB7A0	UVMTEST	1BABB7A0	00003BB0

END OF CICS TRANSACTION DUMP

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.

Using Program Information

PROGRAM INFORMATION FOR THE CURRENT TRANSACTION

Number of Levels 00000001

INFORMATION FOR PROGRAM AT LEVEL 00000001 of 00000001

Program Name	<u>UVPUTSM</u>	Invoking Program	CICS
Load Point	1BAB8000	Program Length	00003798
Entry Point	<u>9BAB8020</u>	Addressing Mode	AMODE 31
Language Defined	COBOL	Language Deduced	COBOL II
Commarea Address	00000000	Commarea Length	00000000
Execution Key	USER	Data Location	ANY
Concurrency	QUASIRENT	Api	CICSAPI
Runtime	LE370		
Environment	User application		

Using the DFHPGPG Information

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

000000320	B5800000	00000000	01000100	0A020000	1A8387A0	17910068	00606EC4	C6C8D7C7	*.....cg..j...->DFHPG*	1A8398D0
000000340	D7D3C3C2	40404040	00000000	E4E5D7E4	E3E2D440	1ABCD870	1BAB8000	9BAB8020	*PLCB.....UVPUTSM.....*	1A8398F0
000000360	00003798	1A9979FC	A0020000	00000000	00000000	00000000	00000000	00000000	*...q.r.....*	1A839910
000000380	00000000	C3C9C3E2	40404040	00000000	00000000	00000000	A8000000	1BAB8000	*...CICS.....y.....*	1A839930
0000003A0	9BAB8020	00003798	1A9979FC	80000000	00000000	00000001	1ABCD870	E4E5D7E4	*...q.r.....UVPU*	1A839950

Program
Load Point

Program
Entry Point

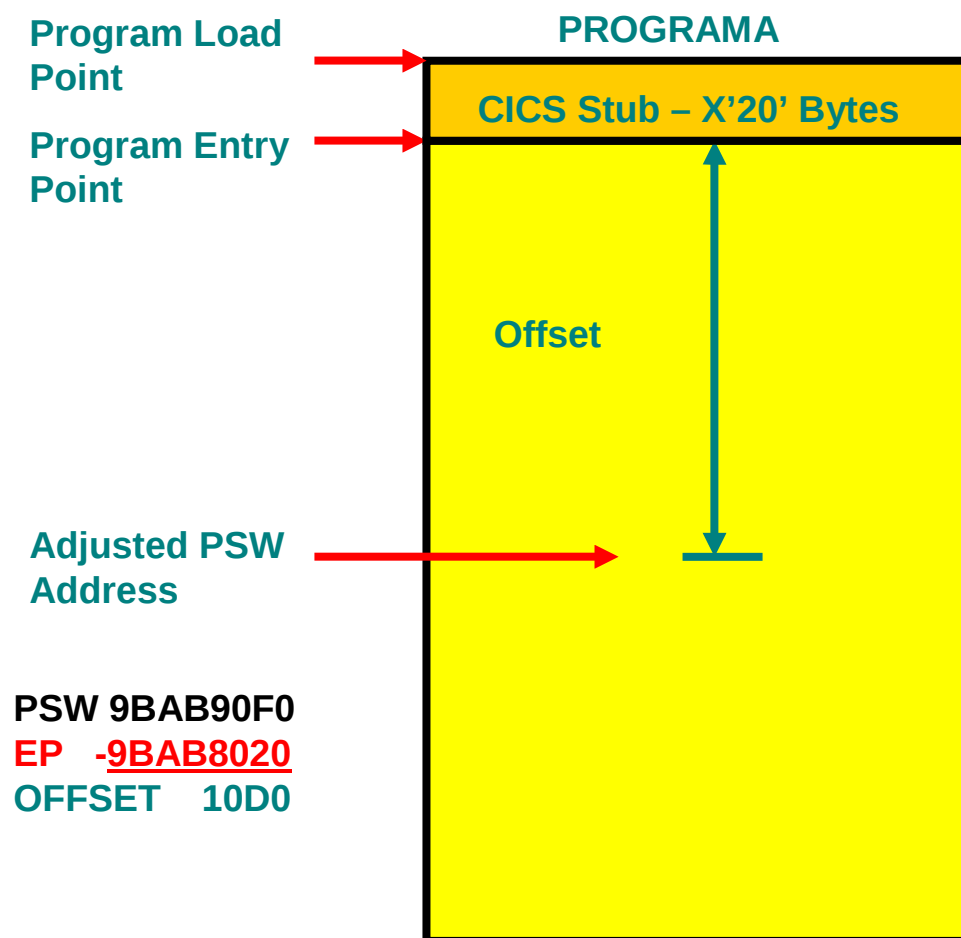
Note: The difference between the Entry Point and the Load Point is the CICS Stub at the beginning of the program which in this case is X'20' bytes long

CICS Stub – X'20'
Bytes Long

UVPUTSM		ADDRESS 1BAB8000 TO 1BAB797		LENGTH 00003798		
PROGRAM STORAGE						
00000000	C436C8E8	C3F6F5F0	58F0021C	58F0F0D0	58F0F014	58F0F00C 58FF000C 07FF0000
00000020	47F0F028	00C3C5C5	00000270	00000014	47F0F001	98CEAC00 1BAB80D6 00000000
00000040	00000000	00000000	90ECD00C	4110F038	98EFF04C	07FF0000 1BAB8020 00000000
00000060	1BABA858	1BAB80CE	1BAB8020	1BAB8DA4	1BABAC58	1BAB80EA 00104001 00000008
00000080	E4E5D7E4	E3E2D440	F2F0F1F1	F0F7F1F1	F1F1F0F0	F0F3F0F4 F0F1F0F0 04740000

DFHYC650.0...00..00..00.....	1BAB8000
.00..CEE.....00.q.....O...	1BAB8020
.....0.q.0<.....	1BAB8040
..y.....u.....	1BAB8060
UVPUTSM 20110711110003040100....	1BAB8080

Computing the Offset



PSW 9BAB90F0
EP -9BAB8020
OFFSET 10D0

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

TASK CONTROL AREA (SYSTEM AREA)

00000000	00000000	00000000	0000148C	194FFE30	0000009D	00000000	00000000	00000000	*.....*	195E3200
00000020	00000000	00000000	00000000	00000000	1AA5A380	00000000	00000000	00000000	*.....vt.....*	195E3220
00000040	00000000	00000000	00000000	00000000	00400000	00000000	00000000	00000000	*.....*	195E3240
00000060	00000000	C1E2D9C1	00000000	195E34E4	00000000	00000000	195E3380	1AA584C8	*...ASRA...;.U...;.vdH*	195E3260
00000080	1AA50128	1AA50580	00000000	00000000	00000000	00000000	E4E3E2D4	1A8F1570	*.V...V.....UTSM....*	195E3280
000000A0	00000000	00000000	E4E3E2D4	00000000	00000000	00000000	C1E2D9C1	00000000	*.....UTSM.....ASRA....*	195E32A0
000000C0	1A906008	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	195E32C0
000000E0	00000000	00000000	E4E5D4E3	C5E2E340	1BABB7A0	5018FEFF	00000000	00000000	*.....UVMTEST.....*	195E32E0
00000100	00000000	00000000	195E356C	00000000	00000000	00000000	00000000	00000000	*.....;.....*	195E3300
00000120	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	195E3320
00000140	00000000	0802006D	1AA50008	00000000	00000000	00000000	00000000	00000000	*.....v.....*	195E3340
00000160	E4E5D7E4	E3E2D440	F0C3F761	C1D2C5C1	000010F6	00020781	00000000	00000000	*UVPUTSM 0C7/AKEA...6...a.....*	195E3360

Program Name

MVS/CICS
Abend Code

CICS Computed
Offset

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

CICS Determined Offset (STCA)	000010F6
Minus Length of Instruction Causing Cancelation	6
Minus Length of CICS Stub	20
Real Offset in program	000010D0

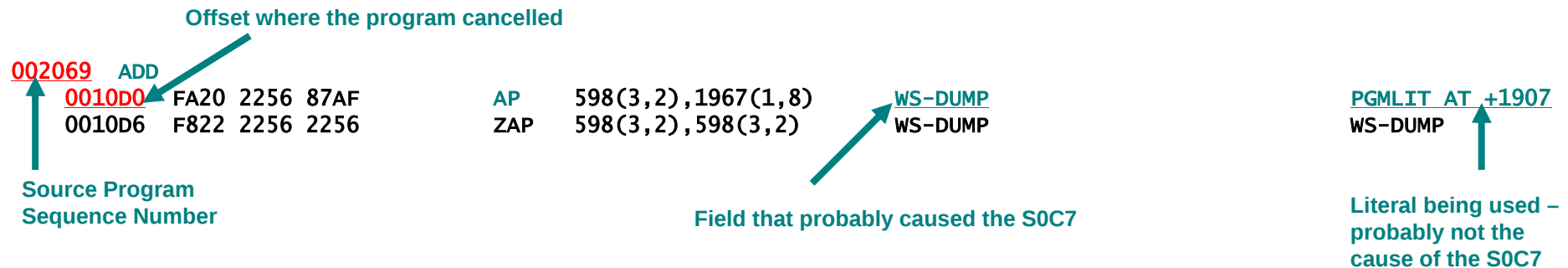
Locating the Failing Instruction

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

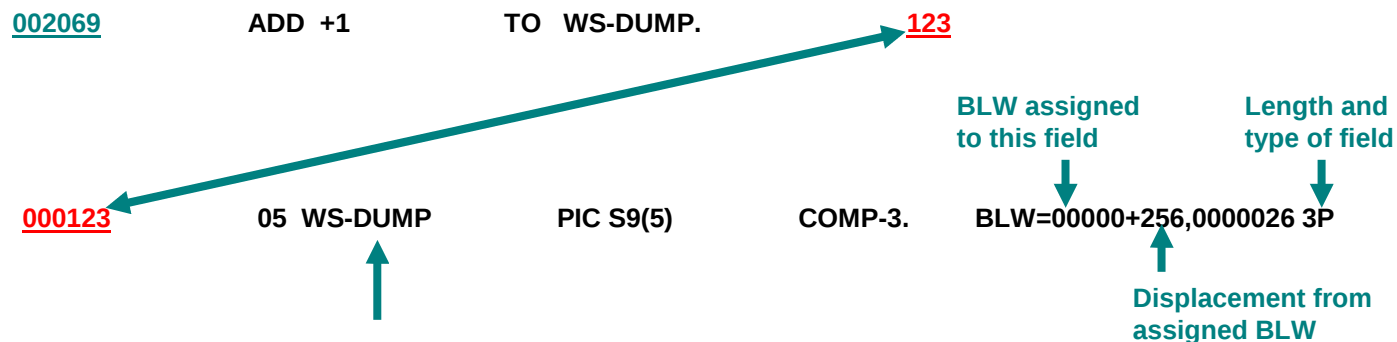
Locate the instruction that cancelled using the offset – identify the source sequence #



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 ***

TGTLOC

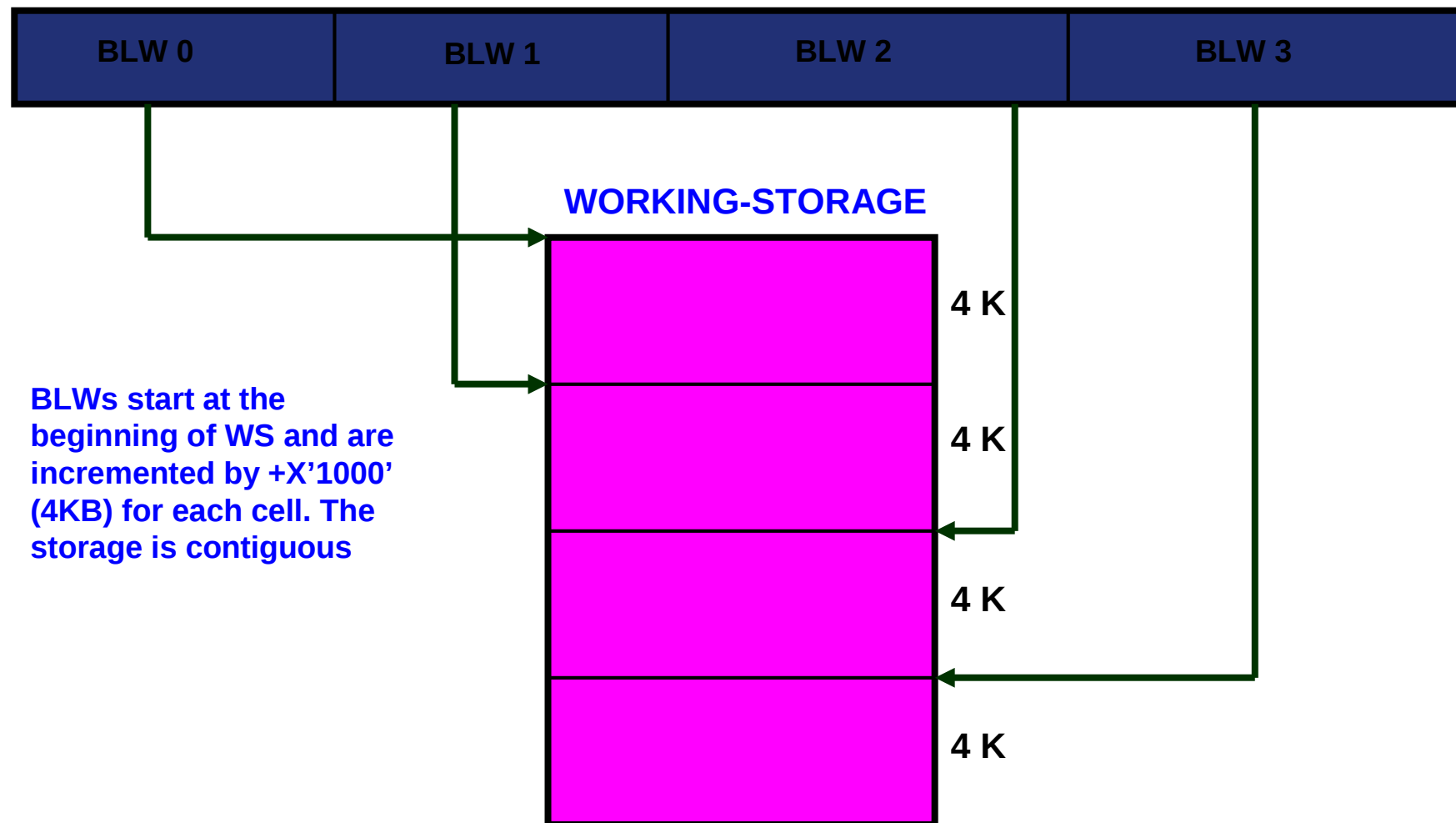
000000 RESERVED - 72 BYTES
000048 TGT IDENTIFIER
 00004C RESERVED - 4 BYTES
 000050 TGT LEVEL INDICATOR
 000051 RESERVED - 3 BYTES
 000054 32 BIT SWITCH
 000058 POINTER TO RUNCOM
 00005C POINTER TO COBVEC
 000060 POINTER TO PROGRAM DYNAMIC BLOCK TABLE
 000064 NUMBER OF FCB'S
 000068 WORKING-STORAGE LENGTH
 00006C RESERVED - 4 BYTES
 000070 ADDRESS OF IGZESMG WORK AREA
 000074 ADDRESS OF 1ST GETMAIN BLOCK (SPACE MGR)
 000078 RESERVED - 2 BYTES
 00007A RESERVED - 2 BYTES
 00007C RESERVED - 2 BYTES
 00007E MERGE FILE NUMBER
 000080 ADDRESS OF CEL COMMON ANCHOR AREA
 000084 LENGTH OF TGT
 000088 RESERVED - 1 SINGLE BYTE FIELD
 000089 PROGRAM MASK USED BY THIS PROGRAM
 00008A RESERVED - 2 SINGLE BYTE FIELDS
 00008C NUMBER OF SECONDARY FCB CELLS
 000090 LENGTH OF THE ALTER VN(VNI) VECTOR
 000094 COUNT OF NESTED PROGRAMS IN COMPILE UNIT
 000098 DDNAME FOR DISPLAY OUTPUT
 0000A0 RESERVED - 8 BYTES
 0000A8 POINTER TO COM-REG SPECIAL REGISTER
 0000AC RESERVED - 52 BYTES
 0000E0 ALTERNATE COLLATING SEQUENCE TABLE PTR.
 0000E4 ADDRESS OF SORT G.N. ADDRESS BLOCK

0000E8 ADDRESS OF PGT
 0000EC RESERVED - 4 BYTES
 0000F0 POINTER TO 1ST IPCB
 0000F4 ADDRESS OF THE CLLE FOR THIS PROGRAM
 0000F8 POINTER TO ABEND INFORMATION TABLE
 0000FC POINTER TO TEST INFO FIELDS IN THE TGT
000100 ADDRESS OF START OF COBOL PROGRAM
 000104 POINTER TO ALTER VNI'S IN CGT
 000108 POINTER TO ALTER VN'S IN TGT
 00010C POINTER TO FIRST PBL IN THE PGT
 000110 POINTER TO FIRST FCB CELL
000114 WORKING-STORAGE ADDRESS
 000118 POINTER TO FIRST SECONDARY FCB CELL
 00011C POINTER TO STATIC CLASS INFO BLOCK 1
 000120 POINTER TO STATIC CLASS INFO BLOCK 2

*** VARIABLE PORTION OF TGT ***

000124 TGT OVERFLOW AREA ADCONS
 000130 BASE LOCATORS FOR SPECIAL REGISTERS
000138 BASE LOCATORS FOR WORKING-STORAGE
003E4C BASE LOCATORS FOR LINKAGE-SECTION
 003E58 CLLE ADDR. CELLS FOR CALL LIT. SUB-PGMS.
 003F2C INTERNAL PROGRAM CONTROL BLOCKS

COBOL Address Cells--BLW



Locating the BLW Cells

								GPR09 →	1AA5B570
000007380	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*	1AA5B4D0
0000073A0	LINES TO	00007440	SAME AS ABOVE						1AA5B4F0
000007460	00000000	00000000	F3E3C7E3	00000000	06000000	68030260	1AA5B1B8	0007809C	1AA5B5B0
000007480	1AA5F4B0	00000000	00F44255	00000000	00000000	1BE00030	00000000	00000000	1AA5B5D0
0000074A0	1AA584C8	00003F40	00000000	00000000	00000000	00000001	E2E8E2D6	E4E34040	1AA5B5F0
0000074C0	C9C7E9E2	D9E3C3C4	00000000	00000000	00000000	00000000	00000000	00000000	1AA5B610
0000074E0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	1AA5B630
000007500	00000000	00000000	1BAB811C	00000000	1AA5F49C	1AA5B490	1BAB8B0C	00000000	1AA5B650
000007520	1BAB8020	1BAB8198	1AA5F49C	1BAB8154	00000000	1BE000C0	00000000	00000000	1AA5B670
000007540	00000000	1AA5C570	1AA5D570	1AA5E570	00000000	1BE00040	1BE000C0	1BE010C0	1AA5B690
000007560	1BE020C0	1BE030C0	1BE040C0	1BE050C0	1BE060C0	1BE070C0	1BE080C0	1BE090C0	1AA5B6B0
000007580	1BE0A0C0	1BE0B0C0	1BE0C0C0	1BE0D0C0	1BE0E0C0	1BE0F0C0	1BE100C0	1BE110C0	1AA5B6D0
0000075A0	1BE120C0	1BE130C0	1BE140C0	1BE150C0	1BE160C0	1BE170C0	1BE180C0	1BE190C0	1AA5B6F0
0000075C0	1BE1A0C0	1BE1B0C0	1BE1C0C0	1BE1D0C0	1BE1E0C0	1BE1F0C0	1BE200C0	1BE210C0	1AA5B710
0000075E0	1BE220C0	1BE230C0	1BE240C0	1BE250C0	1BE260C0	1BE270C0	1BE280C0	1BE290C0	1AA5B730
000007600	1BE2A0C0	1BE2B0C0	1BE2C0C0	1BE2D0C0	1BE2E0C0	1BE2F0C0	1BE300C0	1BE310C0	1AA5B750
000007620	1BE320C0	1BE330C0	1BE340C0	1BE350C0	1BE360C0	1BE370C0	1BE380C0	1BE390C0	1AA5B770
000007640	1BE3A0C0	1BE3B0C0	1BE3C0C0	1BE3D0C0	1BE3E0C0	1BE3F0C0	1BE400C0	1BE410C0	1AA5B790
000007660	1BE420C0	1BE430C0	1BE440C0	1BE450C0	1BE460C0	1BE470C0	1BE480C0	1BE490C0	1AA5B7B0
000007680	1BE4A0C0	1BE4B0C0	1BE4C0C0	1BE4D0C0	1BE4E0C0	1BE4F0C0	1BE500C0	1BE510C0	1AA5B7D0
0000076A0	1BE520C0	1BE530C0	1BE540C0	1BE550C0	1BE560C0	1BE570C0	1BE580C0	1BE590C0	1AA5B7F0
0000076C0	1BE5A0C0	1BE5B0C0	1BE5C0C0	1BE5D0C0	1BE5E0C0	1BE5F0C0	1BE600C0	1BE610C0	1AA5B810
0000076E0	1BE620C0	1BE630C0	1BE640C0	1BE650C0	1BE660C0	1BE670C0	1BE680C0	1BE690C0	1AA5B830
000007700	1BE6A0C0	1BE6B0C0	1BE6C0C0	1BE6D0C0	1BE6E0C0	1BE6F0C0	1BE700C0	1BE710C0	1AA5B850
000007720	1BE720C0	1BE730C0	1BE740C0	1BE750C0	1BE760C0	1BE770C0	1BE780C0	1BE790C0	1AA5B870
000007740	1BE7A0C0	1BE7B0C0	1BE7C0C0	1BE7D0C0	1BE7E0C0	1BE7F0C0	1BE800C0	1BE810C0	1AA5B890
000007760	1BE820C0	1BE830C0	1BE840C0	1BE850C0	1BE860C0	1BE870C0	1BE880C0	1BE890C0	1AA5B8B0

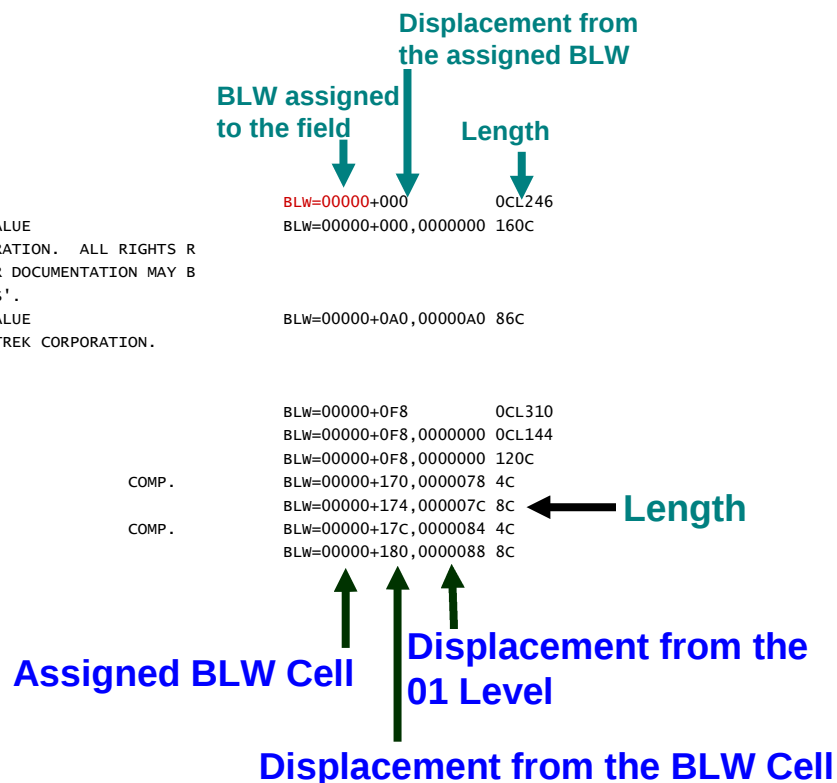
Remember the address of where WS starts is at a displacement of TGT + X'114' and has be equal to the contents of where you locate BLW 0

TGT Address 1AA5B570
 BLW Disp. In TGT 138
 Location of BLW in TGT 1AA5B6A8

Data Division—WS

```

000019      DATA DIVISION.
000020
000021      WORKING-STORAGE SECTION.
000022
000023          COPY                                KVCCPYW.
000024      01  KVEW-COPYRIGHT.
000025          05  FILLER                                PIC X(160) VALUE
000026              '*V6R2M005*  COPYRIGHT 2001 C\TREK CORPORATION.  ALL RIGHTS R
000027              'ESERVED.  NO PART OF THIS PROGRAM AND/OR DOCUMENTATION MAY B
000028              'E REPRODUCED IN ANY FORM OR BY ANY MEANS'.
000029          05  FILLER                                PIC X(86)  VALUE
000030              ', WITHOUT PERMISSION IN WRITING FROM C\TREK CORPORATION.
000031      -
000032
000033      01  WS-COMMON-VARIABLES.
000034          05  WS-CEDF-AREA.
000035              10  CEDF-WORK-AREA                    PIC X(120).
000036              10  CEDF-TGT-BIN                      PIC S9(8)
000037              10  CEDF-TGT-CHAR                    PIC X(8).
000038              10  CEDF-HLLSA-BIN                    PIC S9(8)
000039              10  CEDF-HLLSA-CHAR                    PIC X(8).
  
```



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	<u>256</u>
LOCATION OF FIELD	1BE00316
LENGTH	3 BYTES PACKED

Locating the Failing Field

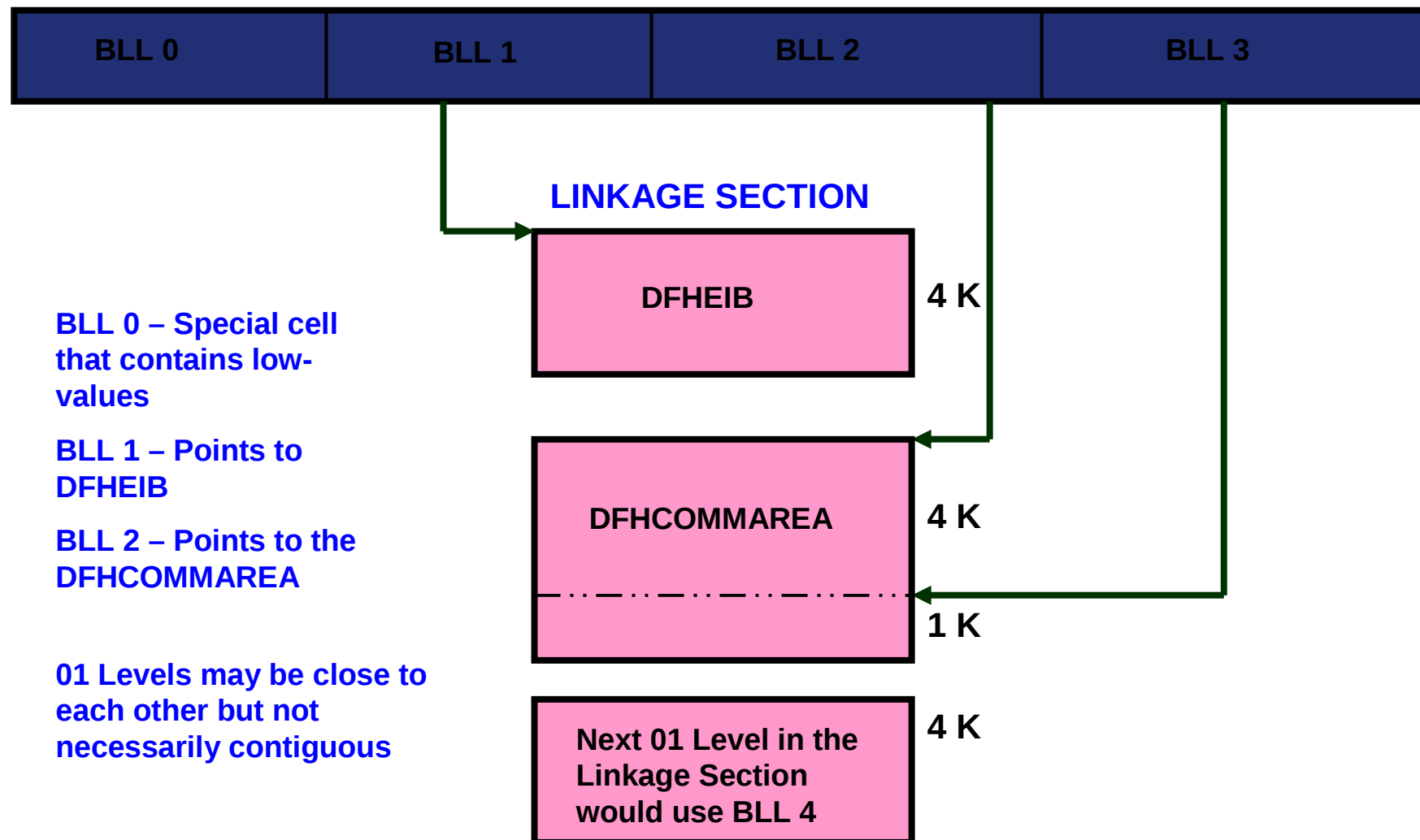
```

-TRANSACTION STORAGE-USER31          ADDRESS 1BE00000 TO 1CD4432F  LENGTH 00F44330
000000000 E4F0F0F0 F0F1F4F8 C8C1D5C3 1AA56798 1AA56798 00000000 1BE00008 00000000 *U0000148HANC.v.q.v.q.....* 1BE00000
000000020 00F44320 00000000 1BE00008 00F44300 00F442F1 00000000 00000000 00000000 *.4.....4...4.1.....* 1BE00020
000000040 00000000 00000000 00000000 00000000 00000000 00000000 C9C7E9E2 D9E3C3C4 *.....IGZSRTCD* 1BE00040
000000060 00000000 00000000 00000000 00000000 00000000 00000000 E2E8E2D6 E4E34040 *.....SYSOUT * 1BE00060
000000080 00000000 00000000 0E000000 00000000 0F000000 00000000 00000000 00000000 *.....* 1BE00080
0000000A0 40404040 40404040 40404040 40404040 40404040 40404040 40404040 40400000 *.....* 1BE000A0
0000000C0 5CE5F7D9 F1D4F0F0 F15C4040 C3D6D7E8 D9C9C7C8 E340F2F0 F1F040C3 E0E3D9C5 **V7R1M001* COPYRIGHT 2010 C.TRE* 1BE000C0
0000000E0 D240C3D6 D9D7D6D9 C1E3C9D6 D54B4040 C1D3D340 D9C9C7C8 E3E240D9 C5E2C5D9 *K CORPORATION. ALL RIGHTS RESER* 1BE000E0
000000100 E5C5C44B 4040D5D6 40D7C1D9 E340D6C6 40E3C8C9 E240D7D9 D6C7D9C1 D440C1D5 *VED. NO PART OF THIS PROGRAM AN* 1BE00100
000000120 C461D6D9 40C4D6C3 E4D4C5D5 E3C1E3C9 D6D540D4 C1E840C2 C540D9C5 D7D9D6C4 *D/OR DOCUMENTATION MAY BE REPROD* 1BE00120
000000140 E4C3C5C4 40C9D540 C1D5E840 C6D6D9D4 40D6D940 C2E840C1 D5E840D4 C5C1D5E2 *UCED IN ANY FORM OR BY ANY MEANS* 1BE00140
000000160 6B40E6C9 E3C8D6E4 E340D7C5 D9D4C9E2 E2C9D6D5 40C9D540 E6D9C9E3 C9D5C740 *, WITHOUT PERMISSION IN WRITING * 1BE00160
000000180 C6D9D6D4 40C3E0E3 D9C5D240 C3D6D9D7 D6D9C1E3 C9D6D54B 40404040 40404040 *FROM C.TREK CORPORATION. * 1BE00180
0000001A0 40404040 40404040 40404040 40404040 40404040 00000000 00000000 *.....* 1BE001A0
0000001C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 1BE001C0
0000001E0 LINES TO 00000220 SAME AS ABOVE 19911904 981C00FF 00000000 00000000 *.....j..q.....* 1BE001E0
000000240 00000000 00000000 00350000 00000035 00000000 00000000 00000000 *.....* 1BE00240
000000260 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 1BE00260
000000280 LINES TO 000002A0 SAME AS ABOVE 00000000 00000000 00000000 00F9F9F9 *.....B.....999* 1BE00280
0000002C0 00000000 00000000 00000000 00000000 C2000000 00000000 00000000 00F9F9F9 *.....999990.....* 1BE002C0
0000002E0 F9F9F9F9 F9F00000 00000F00 00000000 00000000 00000000 00000000 *.....N.....* 1BE002E0
000000300 00000000 00000000 00000000 D5404040 40404040 40400000 00000000 00000000 *.....=WXh...>,123456789:...ABCDEFGH* 1BE00320
000000340 007D6D6A 7EE6E788 7F6C6E6B F1F2F3F4 F5F6F7F8 F97A7B7C C1C2C3C4 C5C6C7C8 *I.<.....* 1BE00340
000000360 C94A4B4C 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....* 1BE00360
000000380 LINES TO 00000AC0 SAME AS ABOVE 00000000 00000000 00000000 00000000 *.....UTL ..* 1BE00380
000000AE0 00000000 00000000 00000000 40404040 40404040 40000000 0000E4E3 D3400000 *.....* 1BE00AE0

```

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

Locating the BLL Cells

GPR09 →

1AA5B570

000007380	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	1AA5B4D0
0000073A0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	1AA5B4F0
000007460	00000000	00000000	F3E3C7E3	00000000	06000000	68030260	1AA5B1B8	0007809C	00000000	*.....3TGT.....-v.....*	1AA5B5B0
000007480	1AA5F4B0	00000000	00F44255	00000000	00000000	1BE00030	00000000	00000000	00000000	*.v4.....4.....*	1AA5B5D0
0000074A0	1AA584C8	00003F40	00000000	00000000	00000000	00000001	E2E8E2D6	E4E34040	00000000	*.vdH.....SYSOUT*	1AA5B5F0
0000074C0	C9C7E9E2	D9E3C3C4	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*IGZSRTCD.....*	1AA5B610
0000074E0	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	1AA5B630
000007500	00000000	00000000	1BAB811C	00000000	1AA5F49C	1AA5B490	1BAB8B0C	00000000	00000000	*.....a.....v4..v.....*	1AA5B650
000007520	1BAB8020	1BAB8198	1AA5F49C	1BAB8154	00000000	1BE000C0	00000000	00000000	00000000	*.....aq.v4..a.....*	1AA5B670
000007540	00000000	1AA5C570	1AA5D570	1AA5E570	00000000	1BE00040	1BE000C0	1BE010C0	00000000	*.....vE..vN..vV.....*	1AA5B690
000007560	1BE020C0	1BE030C0	1BE040C0	1BE050C0	1BE060C0	1BE070C0	1BE080C0	1BE090C0	00000000	*.....*	1AA5B6B0

SNIP

00000B240	1CD3A0C0	1CD3B0C0	1CD3C0C0	1CD3D0C0	1CD3E0C0	1CD3F0C0	1CD400C0	1CD410C0	00000000	*.L...L...L...L...L0..M...M..*	1AA5F390
00000B260	1CD420C0	1CD430C0	1CD440C0	00000000	1AA500D0	00000000	00000000	00000000	00000000	*.M...M...M.....v.....*	1AA5F3B0
00000B280	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	*.....*	1AA5F3D0

BLL0 – Unused

BLL1 – DFHEIB

BLL2 -- COMMAREA

TGT Address

1AA5B570

BLL Disp. In TGT

3E4C

Location of BLL in TGT

1AA5F3BC

BLL0

Data Division—LS

Assigned BLL Cell

```

001983
001984
001985
001986
001987
001988
001989
001990
001991
001992
001993
001994
001995
001996
001997
001998
001999
002000
002001
002002
002003
002004
002005
002006
002007
002008
002009
002010
002011
002012
002013
002014
002015
002016
002017

LINKAGE SECTION.
01 dfheiblk.
02 eibtime comp-3 pic s9(7).
02 eibdate comp-3 pic s9(7).
02 eibtrnid pic x(4).
02 eibtaskn comp-3 pic s9(7).
02 eibtrmid pic x(4).
02 dfheigdi comp pic s9(4).
02 eibcposn comp pic s9(4).
02 eibcalen comp pic s9(4).
02 eibaid pic x(1).
02 eibfn pic x(2).
02 eibrcode pic x(6).
02 eibds pic x(8).
02 eibreqid pic x(8).
02 eibrsrce pic x(8).
02 eibsync pic x(1).
02 eibfree pic x(1).
02 eibrecv pic x(1).
02 eibfil01 pic x(1).
02 eibatt pic x(1).
02 eibeoc pic x(1).
02 eibfmh pic x(1).
02 eibcomp1 pic x(1).
02 eibsig pic x(1).
02 eibconf pic x(1).
02 eiberr pic x(1).
02 eiberrcd pic x(4).
02 eibsynrb pic x(1).
02 eibnodat pic x(1).
02 eibresp comp pic s9(8).
02 eibresp2 comp pic s9(8).
02 eibrldbk pic x(1).
01 DFHCOMMAREA PIC X.

```

```

BLL=00001+000 0CL85
BLL=00001+000,0000000 4P
BLL=00001+004,0000004 4P
BLL=00001+008,0000008 4C
BLL=00001+00C,000000C 4P
BLL=00001+010,0000010 4C
BLL=00001+014,0000014 2C
BLL=00001+016,0000016 2C
BLL=00001+018,0000018 2C
BLL=00001+01A,000001A 1C
BLL=00001+01B,000001B 2C
BLL=00001+01D,000001D 6C
BLL=00001+023,0000023 8C
BLL=00001+02B,000002B 8C
BLL=00001+033,0000033 8C
BLL=00001+03B,000003B 1C
BLL=00001+03C,000003C 1C
BLL=00001+03D,000003D 1C
BLL=00001+03E,000003E 1C
BLL=00001+03F,000003F 1C
BLL=00001+040,0000040 1C
BLL=00001+041,0000041 1C
BLL=00001+042,0000042 1C
BLL=00001+043,0000043 1C
BLL=00001+044,0000044 1C
BLL=00001+045,0000045 1C
BLL=00001+046,0000046 4C
BLL=00001+04A,000004A 1C
BLL=00001+04B,000004B 1C
BLL=00001+04C,000004C 4C
BLL=00001+050,0000050 4C
BLL=00001+054,0000054 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?

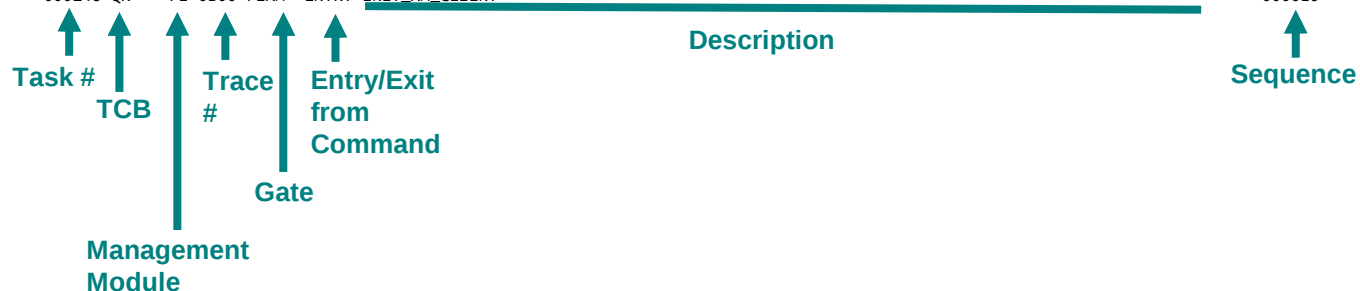
- The Internal Trace Table can be used to obtain additional information
 - In the case of a program check, sufficient information was provided in the transaction dump that can be used to resolve the problem
 - However, there may be cases where you could use the Trace Table to see if a prior error occurred 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

Sample Trace Table

```

* == TRACE ENTRIES FOR DUMPING TRANSACTION ==
* INTERNAL TRACE TABLE SIZE (0002097152)
* REQUESTED TRANSACTION DUMP TRACE TABLE SIZE (0000524288)
* ALLOCATED TRANSACTION DUMP TRACE TABLE SIZE (0000524288)
* 000148 QR AP EA00 TMP ENTRY LOCATE PFT,DFHCICST =000001=
* 000148 QR AP EA01 TMP EXIT LOCATE PFT,DFHCICST,1A8FEF00,NORMAL =000002=
* 000148 QR AP 0591 APXM EXIT INIT_XM_CLIENT/OK =000003=
* 000148 QR AP 1790 TFXM ENTRY INIT_XM_CLIENT 1A8F1570 , 02A00000 =000004=
* 000148 QR XM 1001 XMIQ ENTRY SET_TRANSACTION TERMINAL,1A8F1570 =000005=
* 000148 QR XM 1002 XMIQ EXIT SET_TRANSACTION/OK =000006=
* 000148 QR AP 1791 TFXM EXIT INIT_XM_CLIENT/OK 0000000E,00000000,YES,NO =000007=
* 000148 QR US 0401 USXM ENTRY INIT_TRANSACTION_USER 0000000E,YES =000008=
* 000148 QR DD 0301 DDLO ENTRY LOCATE 1950BB70,1A83993C,USD2,0000000E =000009=
* 000148 QR DD 0302 DDLO EXIT LOCATE/OK 19C01C00 , 1A2C48FD =000010=
* 000148 QR XS 0401 XSXM ENTRY ADD_TRANSACTION_SECURITY 325167F0 , 0000000E =000011=
* 000148 QR XS 0402 XSXM EXIT ADD_TRANSACTION_SECURITY/OK =000012=
* 000148 QR US 0402 USXM EXIT INIT_TRANSACTION_USER/OK 19C01C1F , 19C03090,0 =000013=
* 000148 QR DS 0002 DSAT ENTRY SET_PRIORITY 1 =000014=
* 000148 QR DS 0003 DSAT EXIT SET_PRIORITY/OK 1 =000015=
* 000148 QR KE 0201 KEDD ENTRY INQUIRE_ANCHOR 0000002C =000016=
* 000148 QR KE 0202 KEDD EXIT INQUIRE_ANCHOR/OK 19581000 =000017=
* 000148 QR KE 0201 KEDD ENTRY INQUIRE_ANCHOR 00000010 =000018=
* 000148 QR KE 0202 KEDD EXIT INQUIRE_ANCHOR/OK 00041900 =000019=
* 000148 QR DP 0900 DPXM ENTRY INIT_XM_CLIENT =000020=
* 000148 QR DP 0901 DPXM EXIT INIT_XM_CLIENT/OK =000021=
* 000148 QR RM FA01 RMUC ENTRY CREATE_UOW NO,BACKWARD,0 =000022=
* 000148 QR RM 0209 RMUC EVENT Remote_UOW_id_created 1A11E4E2C1E2C4E5F0F24BC1C3E2E6F0F2F0F515B8B9E5DE700001 =000023=
* 000148 QR RM FA02 RMUC EXIT CREATE_UOW/OK =000024=
* 000148 QR PI 0B00 PIXM ENTRY INIT_XM_CLIENT =000025=

```



ASRA Trace Entry

Abbreviated Trace Entry

```
000148 QR    AP 1949 APLI  EVENT RETURN-FROM-LE/370    Program_Check_Recovery 00000004 UVPUTSM
000148 QR    AP 0790 SRP   *EXC* PROGRAM_CHECK
```

```
=000168=
=000169=
```

Full Trace Entry

```
AP 0790 SRP   *EXC* - PROGRAM_CHECK
```

ILC and Program Check Code

Sequence Number

KERR Entry

PSW

Registers

0 – 15

```
TASK-00148 KE_NUM-0094 TCB-QR /007D3A28 RET-995A066 TIME-17:18:24.9976783781 INTERVAL-00.0000028437 =000169=
1-0000 F0C3F761 C1D2C5C1 018600C7 00000000 C46C8C1 D7D3C9F1 00000000 19BB0300 *0C7/AKEA.f.G...DFHAPLI1.....*
0020 00000000 195E3100 00000000 1A7FD800 321A200 00000001 00000000 FFFFFFFF *.....;"Q...s.....*
0040 079D0000 80000000 00000000 1BAB90F6 00060007 00000000 00000000 7F4FF000 *.....6....."|0.*
0060 90800000 00000000 00000000 1AA5F3C8 00000000 1AA5A4F0 00000000 1BE000C0 *.....v3H....vu0....{*
0080 00000000 1BE00040 00000000 1CD420C0 00000000 1AA500D0 00000000 1CD430C0 *.....\....M.{....v.}*
00A0 00000000 1BE00248 00000000 1BAB815C 00000000 1AA5B570 00000000 1AA5E570 *.....\.....a*....v....vv.*
00C0 00000000 1BAB8B9C 00000000 1BAB811C 00000000 1AA5A380 00000000 9BAB90E2 *.....a.....vt.....S*
00E0 00000000 00000000 9686E57A 00000002 00000000 00000000 00000000 00000000 *.....ofv:.....*
0100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
0120 00000000 00000000 079D0000 80000000 00000000 1BAB90F6 00060007 00000000 *.....6.....*
0140 00000000 7F4FF000 90800000 00000000 00000000 1AA5F3C8 00000000 1AA5A4F0 *....."|0.....v3H....vu0*
0160 00000000 1BE000C0 00000000 1BE00040 00000000 1CD420C0 00000000 1AA500D0 *.....\.{....\....M.{....v.}*
0180 00000000 1CD430C0 00000000 1BE00248 00000000 1BAB815C 00000000 1AA5B570 *.....M.{....\.....a*....v.*
01A0 00000000 1AA5E570 00000000 1BAB8B9C 00000000 1BAB811C 00000000 1AA5A380 *.....vv.....a.....vt.*
01C0 00000000 9BAB90E2 00000000 00000000 9686E57A 00000002 00000000 00000000 *.....S.....ofv:.....*
01E0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
0200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....H...2.....*
0220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
0240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
0260 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
0280 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
02A0 00000000 00000000 00000000 00000000 00000000 000845A0 1932F1B8 0000148C *.....1.....*
02C0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 *.....*
```

Note: PSW is 64-bit format and GPR are 64-bits

CICS TS Trace 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
 - $\text{Offset} = \text{Adjusted PSW Address} - \text{Program Entry point}$

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 → 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 → 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