# 1

# C I C S

# T R A N S A C T I O N

# D E B U G G I N G

# 1

# O B J E C T I V E S

# 1.1 CONSIDERATIONS

- What is the problem?
- Where is the problem?
- What program is affected?
- What external areas can be interrogated?
- What is available for diagnostic purposes?

# 1.1 CONSIDERATIONS

All transaction abends are accompanied by a 1-4 Abend Code

The middle 2 characters identify which module issued the abend

The are over 300 Abend codes

**ASRA** - Program Interrupt
**SR** issued from the System Recovery Program

**AICA** - Program Loop.
**IC** issued from the Interval Control Program

**ABM0** - BMS unable to locate the Map
**BM** issued from the Basic Mapping Support Program

**AEIM** - Notfound condition
**EI** issued from the Exec Interface Program

# 1.1 CONSIDERATIONS

## WHAT IS THE PROBLEM?

A Program Check or Interrupt in an application causes CICS to issue an abend code '**ASRA**'.

A Program Check can take various forms:

- Arithmetic Operation of undefined fields.
- Executing outside of the Address Space/Region.
- Executing invalid instructions.

# 1.1 CONSIDERATIONS

## WHERE IS THE PROBLEM?

A Program Check is the result of user code being interrupted.

In order to answer the questions, we need to know three things:

- At what address is the program loaded?
- Is the program load address and the program entry address the same?
- At what offset in the program, did the interrupt occur?

# 1.1 CONSIDERATIONS

## WHERE IS THE PROBLEM?

The **PROGRAM STATUS WORD (PSW)** will contain the address of the **NEXT** instruction, that would have executed.

The Transaction Dump module index, located at the end of the dump, will show both the Load address and the Entry address.

The Linkedit Map from the Compile and Link output will also show the Load module structure, all the modules that combine to create the Load module that is in error.

# 1.1 CONSIDERATIONS

The **PSW** is 8 bytes long. It can be found on the first page of the transaction dump

An additional 8 bytes are also included

These 16 bytes are divided into 4 words

- Word 1 Contains System control information
- Word 2 Contains the address of the **NEXT** instruction
- Word 3 First 2 bytes are length of instruction that failed
- Last 2 bytes contain the type of exception
- Word 4 Unused for our purposes

# 1.1 CONSIDERATIONS

## WHAT PROGRAM IS AFFECTED?

The Transaction Dump will display on the first page the name of the Program, CICS considered to be currently executing

The storage occupied by this program will be printed in the dump

# 1.1 CONSIDERATIONS

## WHAT EXTERNAL AREAS CAN BE INTERROGATED?

- Any messages on the affected terminal.
- Any messages on the System Log/Console.
- Any messages on the CICS Log.
- Any unusual circumstances surrounding the execution of the Program

# 1.1 CONSIDERATIONS

## WHAT IS AVAILABLE FOR DIAGNOSTIC PURPOSES

- The Compiler listing
- The CEEMSG output
- The AMBLIST utility output
- The Transaction Dump
- The Dump utility **DFHDU660/DFHDU670**

# 1.2 BACKGROUND

CICS demands that all programs be written as **QUASI-**Reentrant,

CICS uses a technique called **MULTI-THREADING**.

To achieve this, when the program is initiated the programs Working-Storage areas are kept outside of the program.

This gives all tasks using the same program, their own copy of Working-Storage areas.

# 1.1 CONSIDERATIONS

The utility **DFHDU660** is used to print the Transaction Dumps

```
//TRANDUMP   EXEC PGM=DFHDU660
//STEPLIB    DD   DSN=CICS.SDFHLOAD,DISP=SHR
//SYSPRINT   DD   SYSOUT=*
//DFHDMPDS   DD   DSN=CICS.DFHDMPA,DISP=SHR
//DFHTINDX   DD   SYSOUT=*
//DFHPRINT   DD   SYSOUT=*
//SYSIN      DD   *
  SELECT TYPE=OR
  TRANID=ABCD
  END
/*
//
  SELECT TYPE=SCAN
  END
```

# 1.1 CONSIDERATIONS

Transaction abends send abend messages to the CICS JES Log

**DFHAP0001 CICSNAME An Abend (Code 0C4/AKEA) Has OCCURRED AT OFFSET X'00001030' IN MODULE CICSBRSJ**

This can provide a good understanding of where the problem is

# 1.2 BACKGROUND

```
CICSTS42   --- CICS TRANSACTION DUMP ---   CODE=ASRA   TRAN=BRSJ   ID=1/0010    DATE=13/09/18 TIME=10:46:43 PAGE    1
SYMPTOMS= AB/UASRA PIDS/5655S9700 FLDS/DFHABAB RIDS/CICSBRSJ
CICS LEVEL = 0670
PSW & REGISTERS AT TIME OF INTERRUPT
PSW                 078D2000   A0910B30   00060007   00000000
REGS 0-7            1FB3A05C   1FB38E68   1FB3A110   1FB3A1A0           001400D0   1F941ACC   1F941A80   00000000
REGS 8-15           1FB3A190   1FB39F20   20910154   2091055C           20910124   1FB38D20   A0910B0A   00000000
EXECUTION KEY       8
The transaction was in Basespace mode
REGISTERS AT LAST EXEC COMMAND
REGS 0-7            1FB46DFC   1FB473C8   1FB35990   1FB473F4           1FB33B30   1FB46DE8   1FB46DFC   1FB46DF8
REGS 8-15           1FB46DEC   1FB473DA   00545360   0054635F           1FB37E58   1FB47330   805455F6   00000000

Transaction environment for transaction_number(0000050)
 transaction_id(BRSJ)       orig_transaction_id(BRSJ)
 initial_program(CICSBRSJ)  current_program(CICSBRSJ)
 facility_type(TERMINAL)    facility_name(L702)             Start_code(TO)
 netname(LCL702   )         profile_name(DFHCICST)
 userid(CICSUSER)           cmdsec(NO)                      ressec(NO)
 spurge(NO)                 dtimeout(0000000)               tpurge(NO)
 taskdatakey(USER)          taskdataloc(BELOW)
 twasize(00000)             twaaddr(        )
 remote(NO)                 dynamic(NO)
 priority(001)              Tclass(NO)                      runaway_limit(0005000)
 indoubt_wait(YES)          indoubt_wait_mins(000000)
 indoubt_action(BACKOUT)    cics_uow_id(C8FE07A04D11B001)   confdata(NO)
 system_transaction(NO)     restart_count(00000)            restart(NO)

TASK CONTROL AREA
00000000   0005C780 00000001 1F9EB570 0004FB48  1F943A50 00000000 00000000 00000060   *..Y.............m.............-*   0005E700
00000020   0000050C 00000000 00000000 9F15FA20  1F99B000 00000000 00000000 00000000   *.................r..............*   0005E720
00000040   00000000 00000000 00000000 00000000  00000000 00000000 00000014 00004000   *.............................. .*   0005E740
00000060   00C3C5E2 C5E2D9C1 00000000 00000000  00000000 00000000 00000000 00000000   *.CESESRA........................*   0005E760
00000080   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005E780
000000A0   LINES TO 000000C0 SAME AS ABOVE
.
TASK CONTROL AREA (SYSTEM AREA)
000000000   00000000 00000000 00000000 00000000  0000081C 096FDD7C 00000056 00000000   *.....................?.........*   0005C780
000000020   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005C7A0
000000040   09F118B0 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *.1.............................*   0005C7C0
000000060   00000000 00000000 00000000 00000000  00000000 00000000 00000000 C1E2D9C1   *............................ASRA*   0005C7E0
```



# 1.2 BACKGROUND

```
CICSTS42   --- CICS TRANSACTION DUMP ---  CODE=ASRA   TRAN=BRSJ   ID=1/0010
DATE=13/09/18   TIME=10:46:43   PAGE  127

----- MODULE INDEX -----
LOAD PT.    NAME    ENTRY PT   LENGTH          LOAD PT.    NAME    ENTRY PT   LENGTH
LOAD PT.    NAME    ENTRY PT   LENGTH
1F7AD100  DFHTOR    1F7B2E5C  0000F1F8
1F7BC300  DFHWBIP   1F7BC300  000014C0
1F7BD800  DFHEJITL  1F7BD828  00000338
1F7BDC00  DFHZXRE   1F7BDD14  00000F70
1F7BF000  DFHZCQ    1F7FEE94  000408D8
1F990000  DFHTAJP   1F990114  000009D8
1F9AC000  DFHACP    1F9AC114  00001848
1FA22000  DFHKCRP   1FA22114  000007D0
1FAD8000  DFHTFP    1FAD8114  00002558
1FC00000  CEEPLPKA  1FC00000  002020D0
1FE020D0  CEEEV010  1FE020D0  0003B7D0
1FE3D8A0  IGZCPAC   1FE3D8A0  0006A500
1FEAA000  DFHWBTC   1FEAA000  0002D448
1FED8000  DFHEITMT  1FED8000  000165C0
1FEEF000  IGZCMGEN  1FEEF000  00007AF8
1FF00000  CEEEV003  1FF00000  005C9428
204CA000  DFHEMTD   204CA028  00020E70
20500000  CEEEV011  20500000  0019CB48
2069D000  DFHEDAD   2069D028  0003CDE8
20700000  DFHCCNV   20700028  001D3D30
20910000  CICSBRSJ  20910028  00001E88
20A00000  DFHAMP    20A00114  000376C0
END OF CICS TRANSACTION DUMP
```

# 1.2 BACKGROUND

```
-TASK CONTROL AREA (SYSTEM AREA)
000000000   00000000 00000000 00000000 00000000  0000081C 096FDD7C 00000056 00000000   *.....................?..........*   0005C780
000000020   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005C7A0
000000040   09F118B0 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *.1.............................*   0005C7C0
000000060   00000000 00000000 00000000 00000000  00000000 00000000 00000000 C1E2D9C1   *............................ASRA*   0005C7E0
1CICSPROD   --- CICS TRANSACTION DUMP ---  CODE=ASRA   TRAN=BRSJ   ID=1/0002    DATE=01/02/06   TIME=18:28:10   PAGE    3
-00000080   00000000 0005CAEC 00000000 00000000  0005C988 09F05CF0 00140128 001403E0   *...................Ih.0*0........*   0005C800
0000000A0   00000000 8004D080 00000000 00000000  C2D9E2D1 0A05F270 00000000 00000000   *.................BRSJ..2.........*   0005C820
0000000C0   00000000 C2D9E2D1 00000000 00000000  00000000 C1E2D9C1 00000000 0A016008   *....BRSJ............ASRA......-.*   0005C840
0000000E0   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005C860
000000100   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005C880
000000120   00000000 00000000 00000000 00000000  8004D400 00000000 00000000 0005CB74   *...................M............*   0005C8A0
000000140   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   0005C8C0
000000160   LINES TO 000001A0 SAME AS ABOVE                                                                                  0005C8E0
0000001C0   00000000 00000000 02000055 00140008  00000000 00000000 00000000 00000000   *................................*   0005C940
0000001E0   00000000 00000000 C3C9C3E2 C2D9E2D1  F0C3F761 C1D2C5C1 00000B36 00020781   *........CICSBRSJ0C7/AKEA.......a*   0005C960
000000200   00000000 00000000                                                          *........                       *   0005C980
```

```
-EXEC INTERFACE USER STRUCTURE
000000000   00B46EC4 C6C8C5C9 E4E24040 40404040  00000000 00000000 09F03780 00000000   *..>DFHEIUS      .........0......*   00140008
000000020   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   00140028
000000040   00000000 00000000 001400D0 00000000  09F118B0 00000000 00000000 00000000   *.................1..............*   00140048
```

@ of EIB

The Exec Interface Block is created by the Command level interface to support the command level interface. It is a transaction level control block. It is located at + x'48' in the EIUS

# 1.2 BACKGROUND

**EXEC INTERFACE BLOCK**

EIBCALEN EIBFN

```
00000000   0104643F 0112018F C2D9E2D1 0000050C  D3F7F0F2 0000000A 00007D02 08000000   *........BRSJ....L702......'.....*   001400D0
00000020   00000000 00000000 00000000 00000000  00000040 40404040 40404000 00000000   *....................        .....*   001400F0
00000040   00000000 00000000 00000000 00000000  00000000 00                         *......................          *   00140110
```

| Offset | Len | Description | Field Name |
|---|---|---|---|
| 00 | 4 | TIME | EIBTIME |
| 04 | 4 | DATE | EIBDATE |
| 08 | 4 | TRANSID | EIBTRNID |
| 0C | 4 | TASK NUMBER | EIBTASKN |
| 10 | 4 | TERMID | EIBTRMID |
| 14 | 4 | RESERVED | EIBRSVD1 |
| 16 | 2 | CURSOR POSITION | EIBCPOSN |
| 18 | 2 | COMMAREA LENGTH | EIBCALEN |
| 1A | 1 | ATTENTION ID | EIBAID |
| 1B | 2 | EXEC CICS FUNCTION | EIBFN |

| | | | |
|---|---|---|---|
| 0202 | ADDRESS | 0436 | ISSUE ERROR |
| 0204 | HANDLE CONDITION | 0438 | ISSUE PREPARE |
| 0206 | HANDLE AID | 043A | ISSUE PASS |
| 0208 | ASSIGN | 043C | EXTRACT LOGONMSG |
| 020A | IGNORE CONDITION | 043E | EXTRACT ATTRIBUTES |
| 020C | PUSH | | |
| 020E | POP | 0602 | READ |
| 0210 | ADDRESS SET | 0604 | WRITE |
| | | 0606 | REWRITE |
| 0402 | RECEIVE | 0608 | DELETE |
| 0404 | SEND | 060A | UNLOCK |
| 0406 | CONVERSE | 060C | STARTBR |
| 0408 | ISSUE EODS | 060E | READNEXT |
| 040A | ISSUE COPY | 0610 | READPREV |
| 040C | WAIT TERMINAL | 0612 | ENDBR |
| 040E | ISSUE LOAD | 0614 | RESETBR |

# 1.3 FINDING THE STATEMENT IN ERROR

| LINE # | HEXLOC | VERB |
|---|---|---|
| 000112 | 000690 | CALL |
| 000121 | 0006E0 | CONTINUE |
| 000126 | 0006F4 | WHEN |
| 000134 | 000748 | IF |
| 000141 | 00079E | MOVE |
| 000167 | 00090E | MOVE |
| 000171 | 000924 | MOVE |
| 000184 | 00097A | EVALUATE |
| 000189 | 00098E | WHEN |
| 000202 | 0009E2 | MOVE |
| 000210 | 000A32 | WHEN |
| 000214 | 000A56 | PERFORM |
| 000236 | 000AEA | MOVE |
| 000239 | 000B02 | ADD |
| 000254 | 000B66 | EVALUATE |
| 000259 | 000B7E | WHEN |
| 000266 | 000BDA | WHEN |
| 000273 | 000C36 | WHEN |
| 000280 | 000C92 | WHEN |
| 000289 | 000CE6 | CALL |
| 000301 | 000D2A | MOVE |
| 000314 | 000D90 | MOVE |

| LINE # | HEXLOC | VERB |
|---|---|---|
| 000118 | 0006D0 | EVALUATE |
| 000123 | 0006E4 | WHEN |
| 000127 | 000700 | PERFORM |
| 000135 | 000756 | PERFORM |
| 000146 | 0007A8 | CALL |
| 000168 | 000914 | MOVE |
| 000178 | 00092A | MOVE |
| 000186 | 00097A | WHEN |
| 000190 | 00099A | PERFORM |
| 000203 | 0009E8 | CALL |
| 000211 | 000A46 | CONTINUE |
| 000216 | 000A6E | CALL |
| 000237 | 000AF0 | ADD |
| 000247 | 000B0E | MOVE |
| 000256 | 000B66 | WHEN |
| 000260 | 000B8A | PERFORM |
| 000267 | 000BE6 | PERFORM |
| 000274 | 000C42 | PERFORM |
| 000281 | 000C9E | PERFORM |
| 000292 | 000D12 | GOBACK |
| 000302 | 000D30 | CALL |
| 000315 | 000D96 | CALL |

| LINE # | HEXLOC | VERB |
|---|---|---|
| 000120 | 0006D0 | WHEN |
| 000124 | 0006F0 | CONTINUE |
| 000129 | 000718 | CALL |
| 000137 | 00076E | CALL |
| 000157 | 0007E0 | CALL |
| 000170 | 00091A | MOVE |
| 000179 | 000930 | CALL |
| 000187 | 00098A | CONTINUE |
| 000192 | 0009B2 | CALL |
| 000208 | 000A32 | EVALUATE |
| 000213 | 000A4A | WHEN |
| 000228 | 000A9E | CALL |
| 000238 | 000AFC | MOVE |
| 000248 | 000B14 | CALL |
| 000257 | 000B7A | CONTINUE |
| 000262 | 000BA2 | CALL |
| 000269 | 000BFE | CALL |
| 000276 | 000C5A | CALL |
| 000283 | 000CB6 | CALL |
| 000295 | 000D1A | MOVE |
| 000308 | 000D80 | MOVE |
| 000321 | 000DE6 | MOVE |

The Offset listing is read left to right, top to bottom



# 1.3 FINDING THE STATEMENT IN ERROR

```
  000221                *EXEC CICS ASSIGN
  000222                *     APPLID(WS-APPLID)
  000223                *     FACILITY(WS-FACILITY)
  000224                *     USERID(WS-USERID)
  000225                *     SYSID(WS-SYSID)
  000226                *     NOHANDLE
  000227                *END-EXEC.
  000228                     Call 'DFHEI1' using by content x'0208f000271f1232200000000000000         EXT
PP 5655-S71 IBM Enterprise COBOL for z/OS  4.2.0               CICSBRSJ  Date 01/12/2012  Time 11:11:56   Page
7
  LineID  PL SL  ----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7-³--+----8 Map and Cross
Reference
  000229                -    '0000000000000000f0f0f1f2f6404040' by reference WS-APPLID by          18
  000230                     reference WS-FACILITY by reference WS-USERID by reference             19 21
  000231                     WS-SYSID end-call.                                                    20
  000232
  000233
  000234
  000235                *
  000236                     MOVE ZEROS TO INSULT.                                                 IMP 34
  000237                     ADD 1 TO INSULT.                                                      34
  000238                     MOVE LOW-VALUES TO INJURY-X.                                          IMP 35
  000239                     ADD INSULT TO INJURY.                                                 34 36
  000240                *
  000241                *EXEC CICS READ
  000242                *     FILE('FILEA')
  000243                *     RIDFLD(TRANS-KEY)
  000244                *     INTO(FILE-REC)
  000245                *     NOHANDLE
  000246                *END-EXEC.
  000247                     Move length of FILE-REC to dfhb0020                                   IMP 24 57
  000248                     Call 'DFHEI1' using by content x'0602f00027000080000f0f0f1f3f9        EXT
  000249                -    '404040' by content 'FILEA   ' by reference FILE-REC by               24
  000250                     reference dfhb0020 by reference TRANS-KEY end-call.                   57 41000278
000AF8 MOVE
```

# 1.3 FINDING THE STATEMENT IN ERROR

## FINDING THE WORKING STORAGE VALUES

| | INSULT | INJURY |
|---|---|---|
| BLW | 0 | 0 |
| DISP | 78 | 80 |
| LENG | 4 | 4 |
| PIC | P | P |

# 1.3 FINDING THE STATEMENT IN ERROR

**Data Division Map**

```
Data Definition Attribute codes (rightmost column) have the following meanings:
    D = Object of OCCURS DEPENDING    G = GLOBAL                          S = Spanned file
    E = EXTERNAL                      O = Has OCCURS clause               U = Undefined format file
    F = Fixed-length file             OG= Group has own length definition V = Variable-length file
    FB= Fixed-length blocked file     R = REDEFINES                       VB= Variable-length blocked file
Source   Hierarchy and                                  Base       Hex-Displacement  Asmblr Data
Data Def
LineID   Data Name                                      Locator    Blk   Structure   Definition      Data Type
Attributes
     3  PROGRAM-ID CICSBRSJ-------------------------------------------------------------------------------
------------*
    18   1  WS-APPLID . . . . . . . . . . . . . . . . . BLW=00000  000               DS 8C           Display
    19   1  WS-FACILITY . . . . . . . . . . . . . . . . BLW=00000  008               DS 4C           Display
    20   1  WS-SYSID. . . . . . . . . . . . . . . . . . BLW=00000  010               DS 4C           Display
    21   1  WS-USERID . . . . . . . . . . . . . . . . . BLW=00000  018               DS 8C           Display
    22   1  WS-NETNAME. . . . . . . . . . . . . . . . . BLW=00000  020               DS 8C           Display
    24   1  FILE-REC. . . . . . . . . . . . . . . . . . BLW=00000  028               DS 0CL80        Group
    25     2  STAT. . . . . . . . . . . . . . . . . . . BLW=00000  028   0 000 000   DS 1C           Display
    26     2  NUMB. . . . . . . . . . . . . . . . . . . BLW=00000  029   0 000 001   DS 6C           Display
    27     2  NAMEX . . . . . . . . . . . . . . . . . . BLW=00000  02F   0 000 007   DS 20C          Display
    28     2  ADDRX . . . . . . . . . . . . . . . . . . BLW=00000  043   0 000 01B   DS 20C          Display
    29     2  PHONEX. . . . . . . . . . . . . . . . . . BLW=00000  057   0 000 02F   DS 8C           Display
    30     2  DATEX . . . . . . . . . . . . . . . . . . BLW=00000  05F   0 000 037   DS 8C           Display
    31     2  AMOUNTX . . . . . . . . . . . . . . . . . BLW=00000  067   0 000 03F   DS 8C           Display
    32     2  COMMENTX. . . . . . . . . . . . . . . . . BLW=00000  06F   0 000 047   DS 9C           Display
    34   1  INSULT. . . . . . . . . . . . . . . . . . . BLW=00000  078               DS 4P           Packed-Dec
    35   1  INJURY-X. . . . . . . . . . . . . . . . . . BLW=00000  080               DS 0CL4         Group
    36     2  INJURY. . . . . . . . . . . . . . . . . . BLW=00000  080   0 000 000   DS 4P           Packed-Dec
    38   1  TRANS-LEN . . . . . . . . . . . . . . . . . BLW=00000  088               DS 2C           Binary
    39   1  TRANS-INPUT . . . . . . . . . . . . . . . . BLW=00000  090               DS 0CL10        Group
```



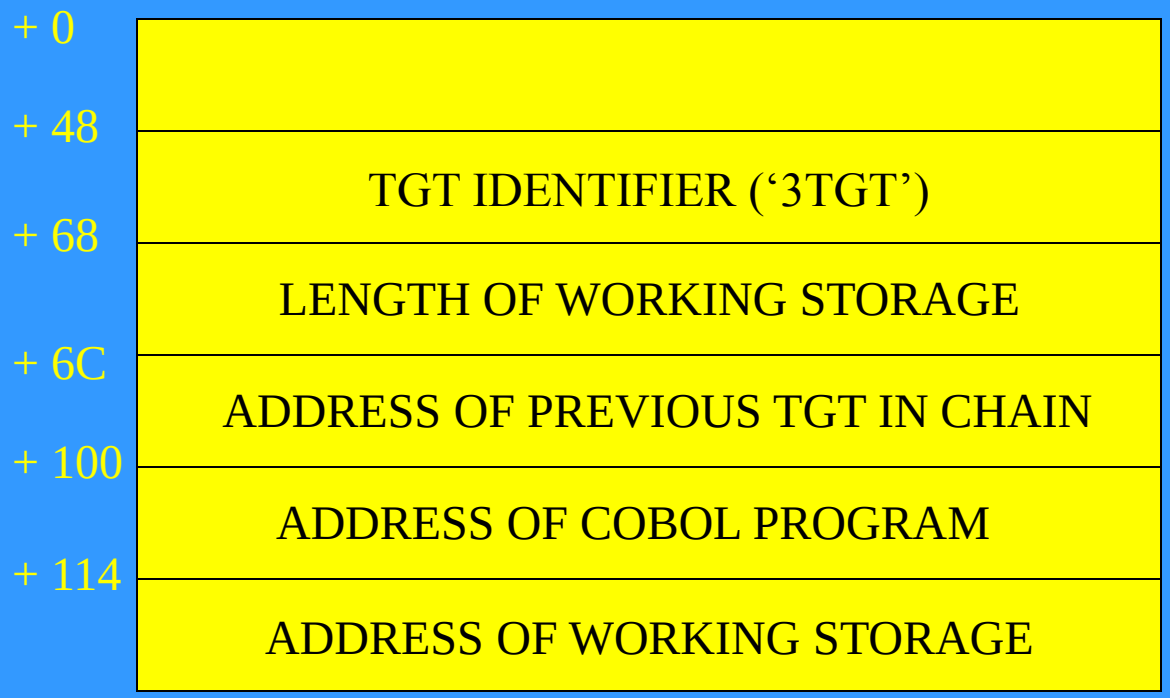
# 1.4 TASK GLOBAL TABLE

Cobol allocates a control block with the invocation of every Cobol program. This is called :

- **Task Global Table (TGT)**

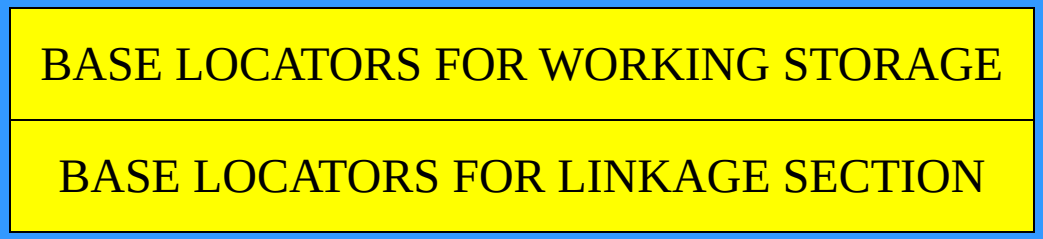
It is created at the beginning of the program, and allocated in CICS User Transaction storage

# 1.4 TASK GLOBAL TABLE




(all offsets are in hex)

## VARIABLE PORTION OF TGT



# 1.4 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
```



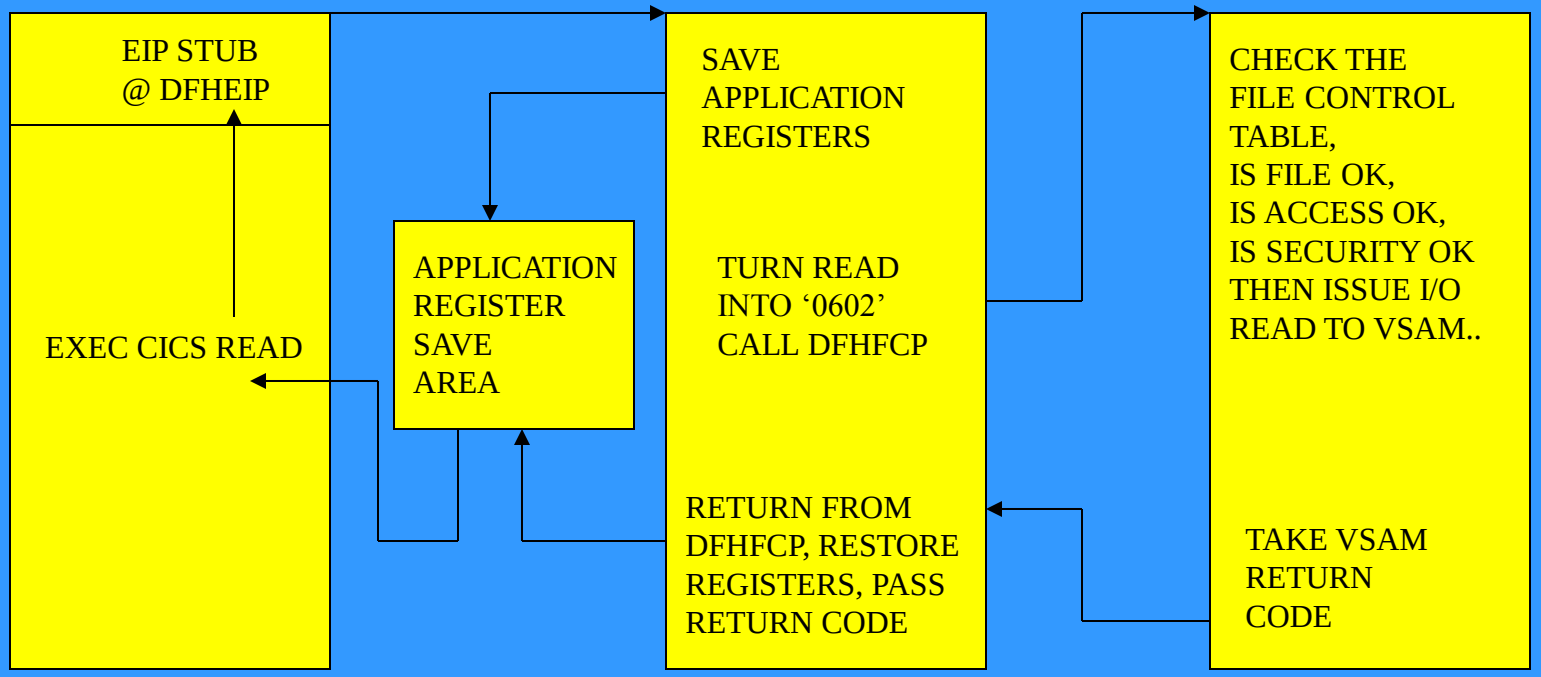
# 1.4 TASK GLOBAL TABLE

```
*** TGT MEMORY MAP ***
                    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  BASE LOCATORS FOR SPECIAL REGISTERS
                    00012C  BASE LOCATORS FOR WORKING-STORAGE
                    000130  BASE LOCATORS FOR LINKAGE-SECTION
                    00013C  CLLE ADDR. CELLS FOR CALL LIT. SUB-PGMS.
                    000194  INTERNAL PROGRAM CONTROL BLOCKS
```

# 1.5 LE DSA




DFHEIP APPLICATION FLOW



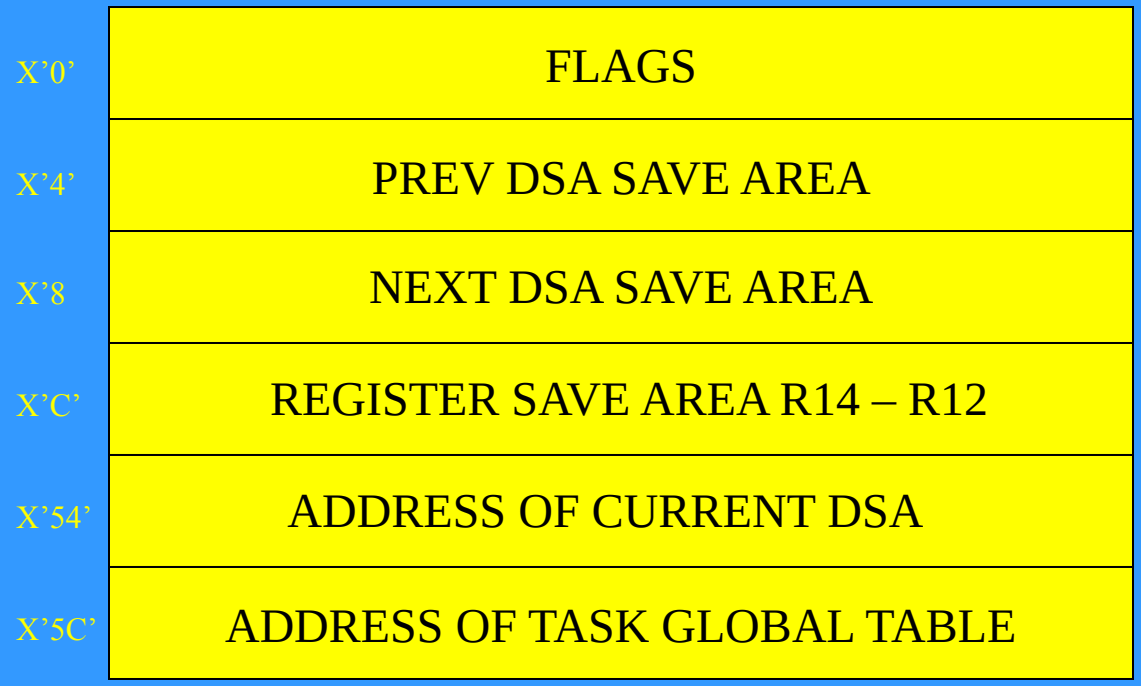
Access Computing Education

# 1.5 LE DSA

```
*** 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
```

# 1.5 LE DSA




**DYNAMIC SAVE AREA**

# 1.5 LE DSA

The information in the CEEMSG is as follows :

- Each DSA address
- The name of the Program Unit
- The Program Unit entry point address
- The Program Unit Offset (the last instruction to run in the routine)
- The Entry Point name
- The Entry Point offset

The Registers are also displayed

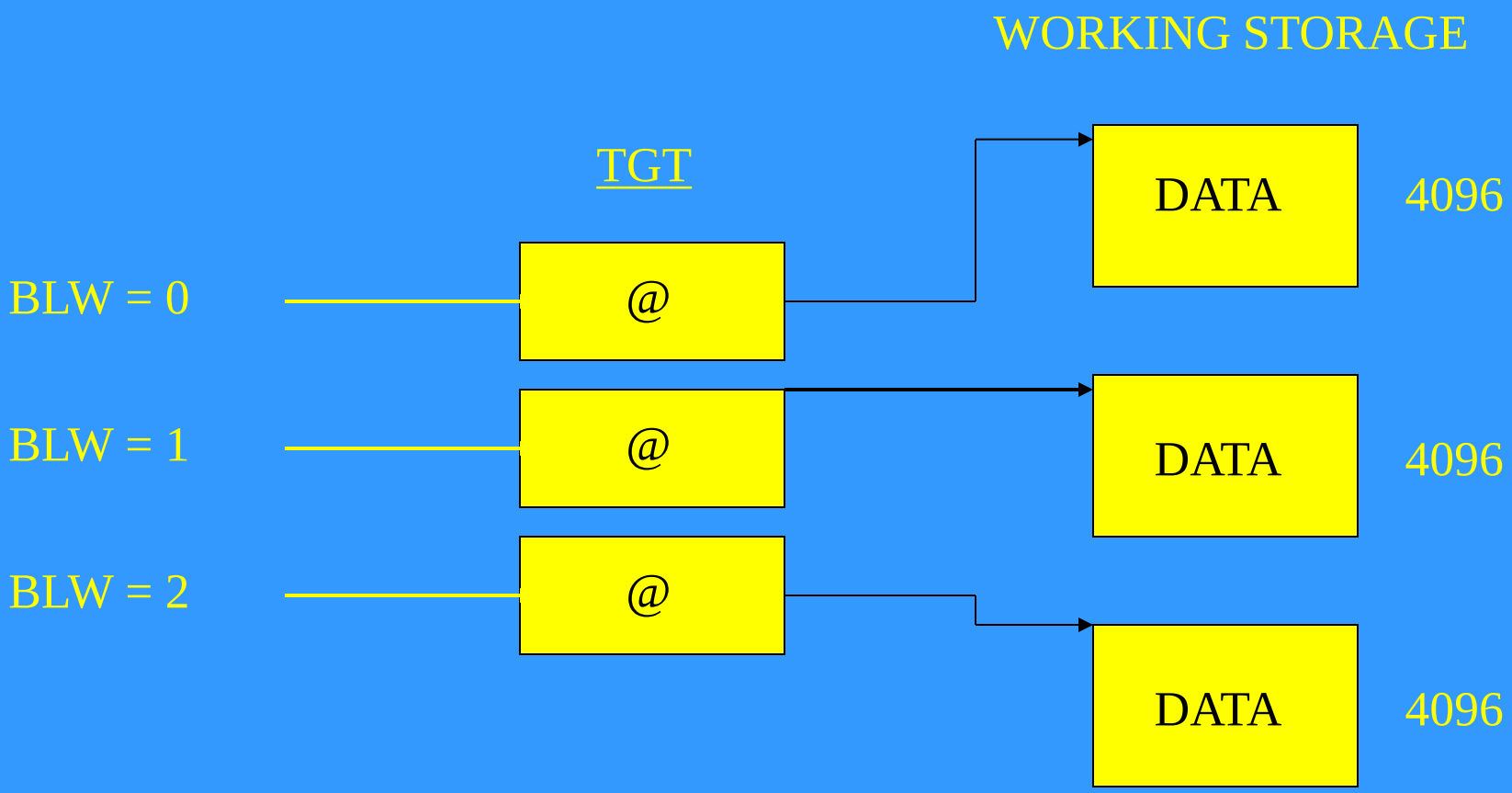
# 1.6 BASE LOCATOR CELLS

The Cobol Compiler assigns **BLW's** (Base locators for Working Storage Cells), to the Application's Working Storage.

The Compiler assigns **BLL** Cells to the Application's Linkage Section. A Cell is simply 4 bytes to hold an address and Cells are given numbers.

These Cell numbers can be found in the Data Division Map in the Compiler output.

# 1.6 BASE LOCATOR CELLS



# 1.6 BASE LOCATOR CELLS

From Addr in
Register 13 – Begin of current DSA

```
00005240   00000000 00000000 00000000 00000000  00104001 1FB38B88 00000000 A0910B0A  *.................. ....h.....j..*   1FB38D20
00005260   1FCAA308 1FB3A05C 1FB38E68 1FB3A110  1FB3A1A0 001400D0 1F941ACC 1F941A80  *..t....*.................m...m..*   1FB38D40
00005280   00000000 1FB3A190 1FB39F20 20910154  2091055C 1FB37E58 00000000 1FB38F60  *.............j...j.*..=........-*   1FB38D60
000052A0   00000000 00000000 1FB38D30 1FB39F20  00000000 00000000 00000000 00000000  *................................*   1FB38D80
```

Offset x'5C' – Addr of TGT

# 1.6 BASE LOCATOR CELLS

# 1.7 EXEC CICS LINK

```
CICSTS42   --- CICS TRANSACTION DUMP ---  CODE=ASRA   TRAN=LNKA
ID=1/0004    DATE=13/09/24   TIME=04:46:31   PAGE  104
PROGRAM INFORMATION FOR THE CURRENT TRANSACTION
   Number of Levels 00000002
INFORMATION FOR PROGRAM AT LEVEL 00000002 of 00000002
   Program Name      CICSLNKZ      Invoking Program CICSLNKA
   Load Point        20911860      Program Length   000016B8
   Entry Point       A0911888      Addressing Mode  AMODE 31
   Language Defined  Unknown       Language Deduced COBOL II
   Commarea Address  1FB3A1A8      Commarea Length  00000006
   Execution Key     USER          Data Location    BELOW
   Concurrency       QUASIRENT     Api              CICSAPI
   Runtime           LE370
   Environment       User application
INFORMATION FOR PROGRAM AT LEVEL 00000001 of 00000002
   Program Name      CICSLNKA      Invoking Program CICS
   Load Point        20910000      Program Length   00001858
   Entry Point       A0910028      Addressing Mode  AMODE 31
   Language Defined  Unknown       Language Deduced COBOL II
   Commarea Address  00000000      Commarea Length  00000000
   Execution Key     USER          Data Location    BELOW
   Concurrency       QUASIRENT     Api              CICSAPI
   Runtime           LE370
   Environment       User application
```

# 1.7 EXEC CICS LINK

PROGRAM 1

**EXEC CICS LINK PROGRAM 2**

TCA SYSTEM AREA
+ x'38' @ of current DFHPESA

PROGRAM 2

**EXEC CICS LINK PROGRAM 3**

PROGRAM 3

**ASRA ABEND**

DFHPESA
1st PESA
on Link Chain
+ x'18' @ of LE DSA

**LE DSA Of Program 1**

DFHPESA
+ x'C' @ of
Previous DFHPESA
+ x'18' @ of LE DSA

R13
@ of
Current
DSA

DSA + x'5C' = @ TGT

TGT + x'100' = @ of Program
TGT + x'114' = @ of W/S

**LE DSA Program 2**

**LE DSA Program 3**

# 1.7 EXEC CICS LINK

```
TASK CONTROL AREA (SYSTEM AREA)
00000000   00000000 00000000 00000000 00000000  0000046C 2B3CD9F8 00000095 00000000   *.....................R8...n....*   0005F800
00000020   00000000 00000000 00000000 00000000  00000000 00000000 1F93B328 00000000   *.........................l......*   0005F820
00000040   1FB4F8E0 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *..8............................*   0005F840
00000060   00000000 00000000 00000000 00000000  00000000 00000000 00000000 C1E2D9C1   *............................ASRA*   0005F860
```

Address of previous PESA (if it exists)

```
00000360   80000000 C3C9C3E2 D3D5D2C1 00000000  00000000 00000000 01906EC4 C6C8D7C5   *....CICSLNKA.............>DFHPE*   1F93B310
00000380   E2C10180 00000000 1F93A40C 00000000  1FB38D30 00000000 00000000 00000000   *SA.......lu.....................*   1F93B330
000003A0   00000000 00001FB3 3AE80000 00000000  00001F98 2E441F93 9B240000 0000D8D9   *.........Y.........q...l......QR*   1F93B350
000003C0   00080000 00000100 00060880 80000000  00000000 00001FB3 3B300000 00000000   *................................*   1F93B370
000003E0   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000   *................................*   1F93B390
00000400   00000000 00000014 00D00000 00001FB3  8D300000 00000000 00000000 00009F11   *................................*   1F93B3B0
```

Address of previous DSA

The **Program Environment Save Area - PESA** is created whenever a Program issues an EXEC CICS LINK. It is created by the Program Manager Domain.

The PESA address can be found at + x'38' in the System TCA

The previous DSA address can be found at PESA + x'18'

The address of the previous PESA can be found at + x'C' in the current PESA

# 1.8 COBOL CALL

The COBOL TGT has an eyecatcher at offset X’48’ It is ‘**3TGT**’.

It is always necessary to check that this is the correct TGT

If the failure is in a COBOL program that has been 'called' using the COBOL static **CALL** statement, CICS will have no knowledge of this program. In order to find the address of this program, check how far down the Load module the program is located. This is the offset down the program storage.

This can easily be ascertained by the output from AMBLIST

# 1.8 COBOL CALL

```
//JOBNAME      JOB,'ACCNT','AMBLIST',CLASS=A,MSGCLASS=X,
//     NOTIFY=USERID
//STEP1  EXEC PGM=AMBLIST
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DSN=the.load.library,DISP=SHR
//SYSIN     DD *
  LISTLOAD MEMBER=membername,output=xref
//
```

```
CONTROL SECTION                                    ENTRY
                LMOD LOC     NAME       LENGTH  TYPE          LMOD LOC  CSECT LOC      NAME
                     00    DFHELII          26   SD

                                                                00        00       DFHEPIN
                                                                08        08       DFHEI1
                                                                08        08       DFHEI8
```

# 1.8 COBOL CALL

**Extract from AMBLIST output**

```
CONTROL SECTION
LMOD  LOC      NAME       LENGTH   TYPE
      00       DFHECI        1E    SD


      20       CICSBRSJ    1284    SD
      12A8     CEESG005      18    SD
      12C0     CEEBETBL      24    SD
      12E8     CEESTART      7C    SD
      1368     IGZCBSO      568    SD
      18D0     CEEARLU       B8    SD
      1988     CEEBPIRA     2C0    SD
      1C48     CEECPYRT      EB    SD
      1D38     CEEBPUBT      70    SD
      1DA8     CEEBTRM       AC    SD
      1E58     CEEBLLST      5C    SD

      1EB8     CEEBINT       08    SD
```

# 1.8 COBOL CALL

CONTROL SECTION

| NAME | ORIGIN | LENGTH |
|---|---|---|
| DFHECI | 00 | 48 |
| CICSPROG | 48 | 6B8 |
| IGZEBST | 700 | 428 |
| PROG1 | B28 | 30 |
| PROG2 | B58 | 50 |
| PROG3 | B88 | 120 |

ENTRY ADDRESS 48
TOTAL LENGTH CA8

# 1.9 THE REGISTER SAVE AREA

The layout of the **REGISTER SAVE AREA** is an IBM convention. All systems follow this.
Register 13 is used to contain the address of a **REGISTER SAVE AREA**.

| | | | | | |
|---|---|---|---|---|---|
| FLAGS | PREVIOUS SAVE AREA @ | NEXT SAVE AREA @ | R14 | R15 | R0 |
| R1 | R2 | R3 | R4 | R5 | R6 |
| R7 | R8 | R9 | R10 | R11 | R12 |

REGISTER SAVE AREA LAYOUT

# 1.9 THE REGISTER SAVE AREA

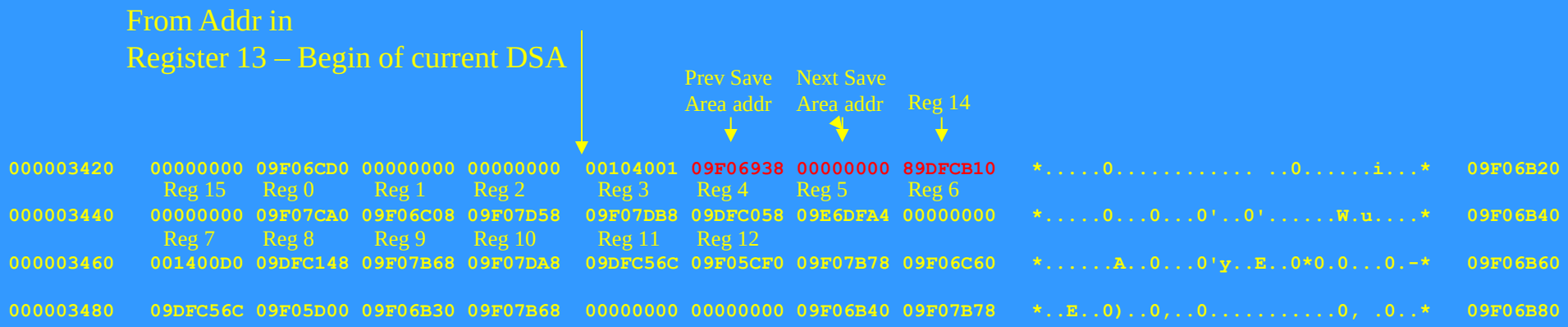
So at any time the current application registers can be determined by scanning the Save Area in the **DSA**.

The current **DSA** can be found by following the address in Register 13.

The **DSA** will contain Register 14 at offset x'C' or the fourth Word, which is the return address into the program from where the Call was made

Offset x'4' contains the address of the previous DSA

# 1.9 THE REGISTER SAVE AREA




As can be seen the first 72 (x'48') bytes in the DSA is a Register Save Area. This is referred to as the APPLICATION REGISTER SAVE AREA

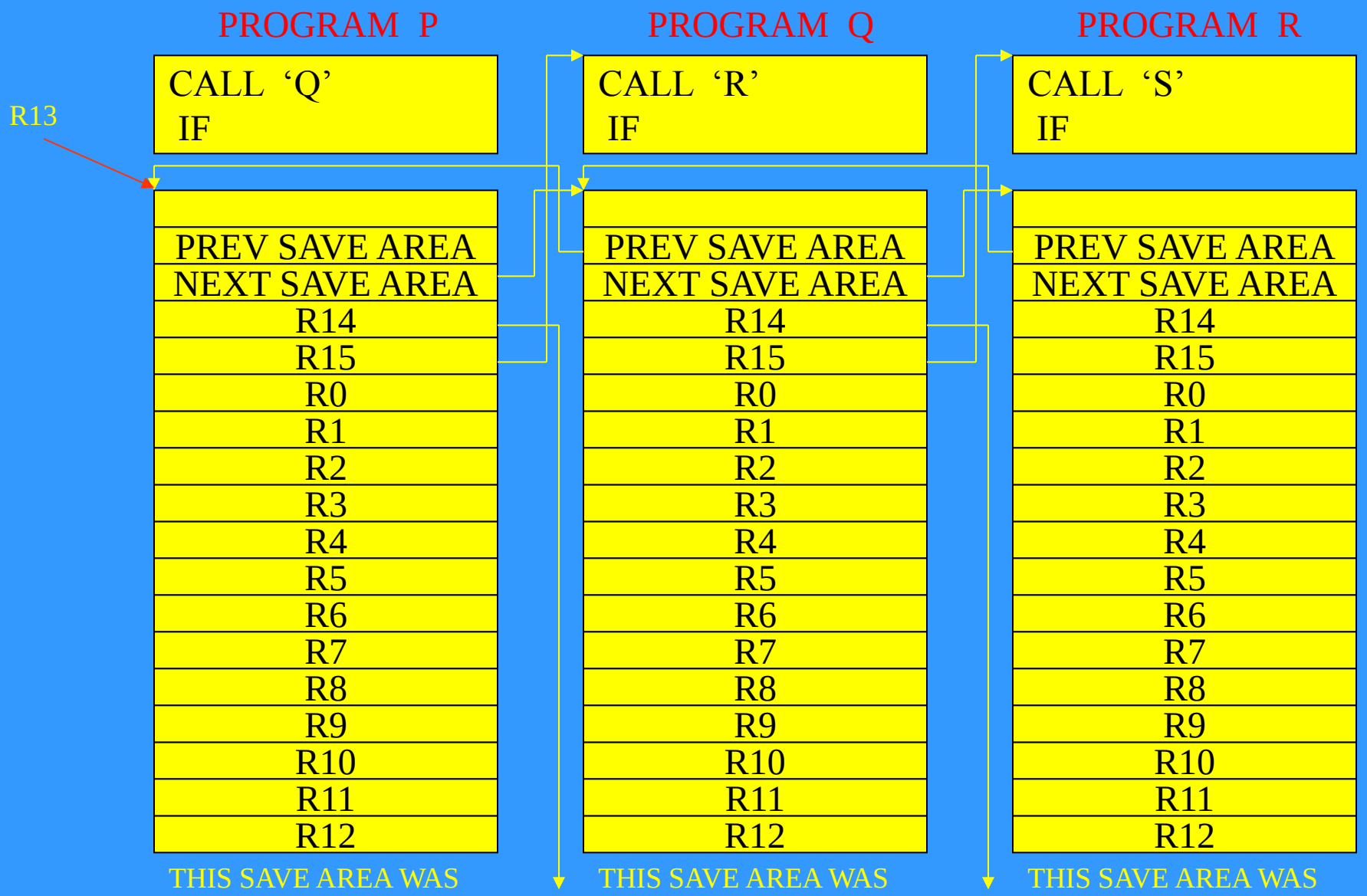
By analysing this Save area, we can determine the contents of the Registers at the time of the Call.

This Save Area would be updated with the Registers on every Call

Remember all EXEC CICS commands are CALLs

So by analysing the contents of Register 14, we can subtract the Program's Entry Point address from this, and using the Offset listing from the Compiler, we can locate WHICH CALL is the current one.

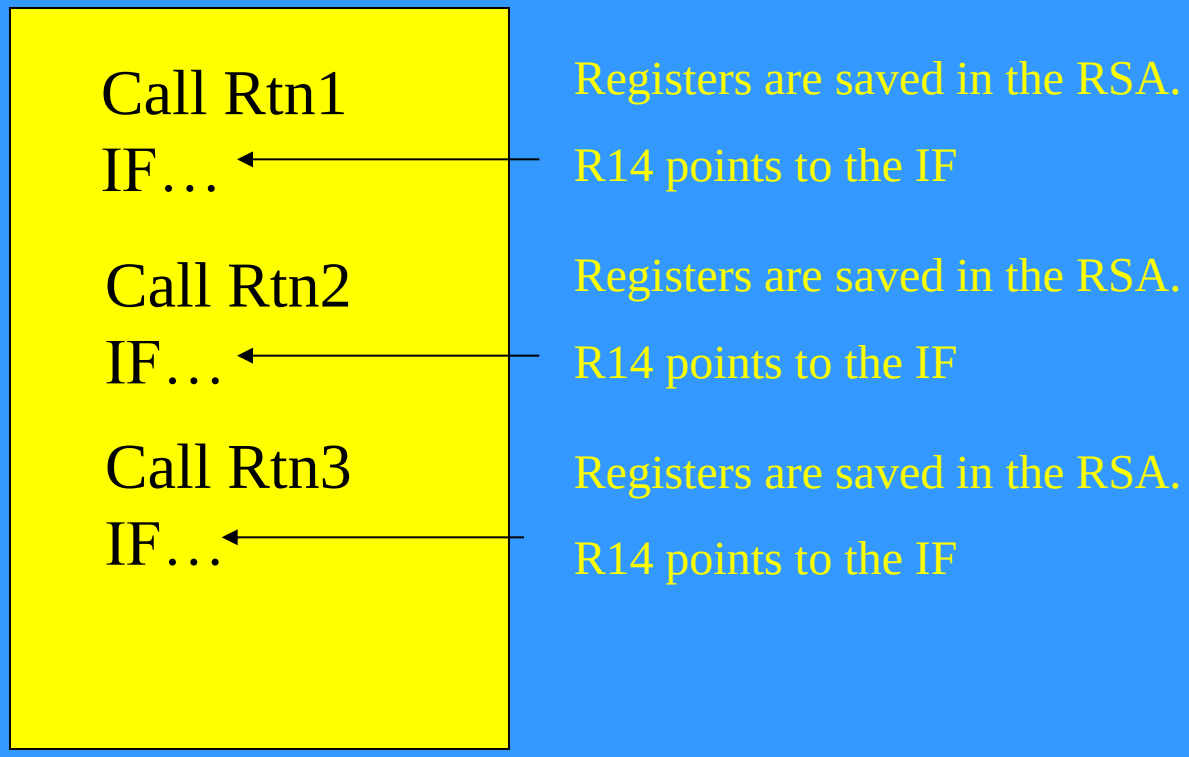
# 1.9 THE REGISTER SAVE AREA



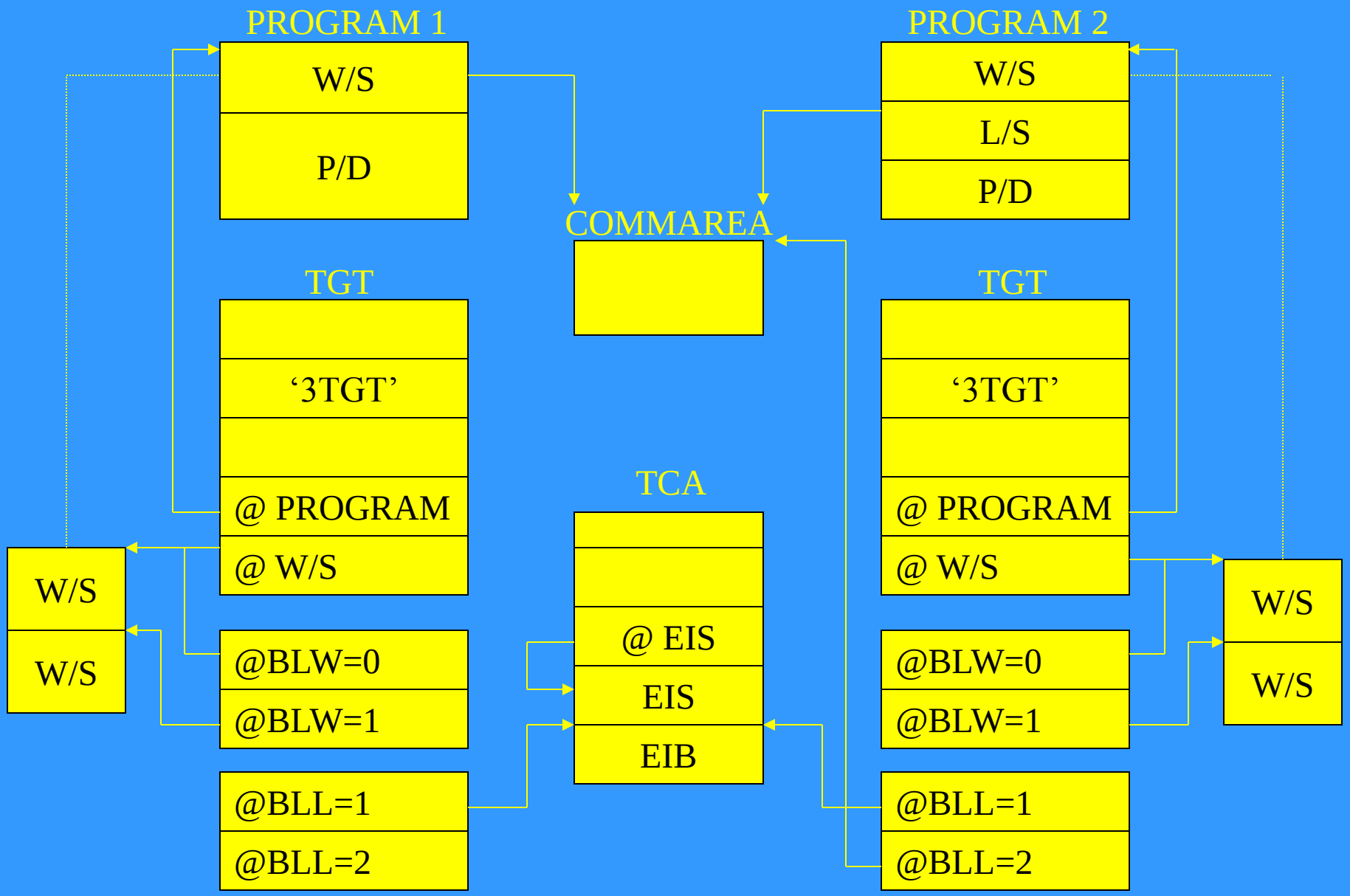
# 1.9 THE REGISTER SAVE AREA

Register 14 contains the Return address into the program from where the last CALL came

By locating Register 14, the last CALL statement can be ascertained



# 1.10 PROGRAM CONTROL BLOCKS



# 1.11 INTERVAL CONTROL VALUE RUNAWAY

The ICVR parameter controls how much CPU time a transaction is allowed to consume between calls to CICS

Its specified in the SIT – (Systems Initialisation Table)

It covers all transactions and can be changed online

A runaway time can be established on an individual transaction basis as well


Access Computing Education

# 1.12 TECHNIQUES FOR DEBUGGING LOOPS

The '**EXEC CICS ENTER TRACENUM**' is a much better method of narrowing the loop

```
EXEC CICS ENTER TRACENUM (01)
          RESOURCE ('TEXT')
          FROM (W/S)
```

The '**EXEC CICS ENTER TRACENUM**' does not cause CICS to return to the Dispatcher, so this command does not reset the timer (**ICVR**)

THANK YOU FOR YOUR TIME AND PATIENCE

ANY QUESTIONS, or EMAIL ME AT :

COLIN.PEARCE@GMAIL.COM

A range of CICS courses
A range of z/OS courses
A range of DB2 courses

Notes:

**1.1 CONSIDERATIONS**

WHERE IS THE PROBLEM?

The **PROGRAM STATUS WORD (PSW)** will contain the address of the **NEXT** instruction, that would have executed. However, if the interrupt code is 0010, or 0011, then the PSW will contain the address of the failing instruction.

The Transaction Dump module index, located at the end of the dump, will show both the Load address and the Entry address.

The Linkedit Map from the Compile and Link output will also show the Load module structure, all the modules that combine to create the Load module that is in error.

The AMBLIST utility can be run to list the structure of the Load module. This will be covered shortly.

**1.1 CONSIDERATIONS**

The **PROGRAM STATUS WORD** is 8 bytes long in 24 and 31 bit addressing mode. It can be found on the first page of the transaction dump.

The 8 bytes are divided into 2 words, however an additional 8 bytes are included in the dump as follows:

cccccccc AAAAAAAA LLLLIIII uuuuuuuu

**WORD 1** **WORD 2** **WORD 3** **WORD 4**

Each word is 4 bytes in length.

**WORD 1** Contains the System Control information, such data as the status of the Condition Code, Protection Key, Wait State and execution state - Problem or Supervisor.

**WORD 2** Contains the address of the **NEXT** instruction that would be executed either 24bit or 31bit mode. 31bit mode is more likely.

**WORD 3** The first two bytes contain the length of the instruction that failed.

The remaining two bytes contain the type of exception that occurred, referred to as Program Interrupt codes.

**WORD 4** Is unused for our purposes.

**1.1 CONSIDERATIONS**

WHAT PROGRAM IS AFFECTED?

The Transaction Dump will display on the first page the name of the Program, CICS considered to be currently executing.

The storage occupied by this program will be printed in the dump.

### 1.1 CONSIDERATIONS

WHAT EXTERNAL AREAS CAN BE INTERROGATED?

- Any messages on the affected terminal.
- Any messages on the System Log/Console.
- Any messages on the CICS Log.
- Any unusual circumstances surrounding the execution of the Program

### 1.1 CONSIDERATIONS

WHAT IS AVAILABLE FOR DIAGNOSTIC PURPOSES

- The Compiler output.
- The **CEEMSG** output
- The **AMBLIST** output
- The Transaction Dump.
- The Dump utility :
  - DFHDU660 - CICS/TS 4.1
  - DFHDU670 - CICS/TS 4.2

### 1.2 BACKGROUND

CICS demands that all programs be written as **QUASI**-Reentrant, this means that there must be no user code between calls to CICS, that is self-modifying.

CICS uses a technique called **MULTI-THREADING**. This allows many tasks to execute the same copy of the program.

To achieve this, when the program is initiated the programs Working-Storage areas are kept outside of the program. This is quite the opposite to Batch processing.

This gives all tasks using the same program, their own copy of Working-Storage areas.

**1.2 BACKGROUND**

The utility **DFHDU660** is used to print the Transaction Dumps and can be used to select the required dump.

```
//TRANDUMP  EXEC PGM=DFHDU660
//STEPLIB   DD   DSN=CICS.SDFHLOAD,DISP=SHR
//SYSPRINT  DD   SYSOUT=*
//DFHDMPDS   DD   DSN=CICS.DFHDMPA,DISP=SHR
//DFHTINDX  DD   SYSOUT=*
//DFHPRINT  DD   SYSOUT=*
//SYSIN     DD   *
 SELECT TYPE=OR
 TRANID=ABCD
 END
/*
```

The **CICS/TS OPERATIONS and UTILITIES GUIDE** should also be consulted for the parameters.

**1.2 BACKGROUND**

When a transaction abend occurs the following message is sent to the CICS JES log and can be viewed there.

**DFHAP0001 CICSNAME An Abend (Code 0C4/AKEA) Has OCCURRED AT OFFSET X'00001030' IN MODULE CICSBRSJ**

The Kernel issues the first Abend Code : **AKEA** on any program check in the program.

The offset is the displacement from the beginning of the load module

**DFHME0116 CICSNAME (Module:DFHMEME) CICS SYMPTOM STRING FOR Message DFHAP0001 is PIDS/566540301 LVLS/640 MS/DFHAP0001 RIDS/DFHAPDS PTFS/TS640 AB/S00C4 AB/UAKEA RIDS/CICSBRSJ ADRS/00001030**

This message is mainly for System Programmers who need to talk to IBM support personnel

**1.2 BACKGROUND**

This is the first page of the Transaction dump. It has the following :

- CICS Region name
- Abend code
- Transaction name
- Dumpid (Dump Run number and Dump Count)
- Date and time of the Abend
- CICS Level
- Symptom string
- Registers and Program Status Word
- Transaction details as in the CICS System Definition dataset (CSD)
- Task Control Area

**1.2 BACKGROUND**

This is the Module index from the transaction dump. It shows the Load and Entry point addresses and length of each module loaded into CICS. It is found at the end of the transaction dump.

Question to ask

Is the Program Load Address and the Program Entry Address, the same?

**1.2 BACKGROUND**

The above display shows how to find the EXEC INTERFACE BLOCK via the Task Control Area (System).

The offset at x'1CC' contains the address of the EXEC Interface User Structure.

Offset x'48' into the EIUS is the address of the Exec Interface Block

The Exec Interface Block is created by the Command level interface to support the command level interface. It is a transaction level control block

**1.2 BACKGROUND**

The above shows the layout of the EIB :

The Transaction dump layout

The Data area layout

### 1.3 FINDING THE STATEMENT IN ERROR

The Transaction will contain two pieces of information that will be help you locate the Statement in the Program from where the error occurred.

Remember the role of the Program Status Word (PSW) – it contains the address of the next instruction that would have executed had the program continued on. This is in the second word.

The address of the Program can be found in the Transaction Dump – Module Index. This gives both the Load address and the Entry Point address.

So Subtract the Entry Point address from the PSW address. This will give you a Displacement.

Now ensure your COBOL program has been compiled with the compiler option OFFSET or LIST. This will give the display as above.

Look for the headings :

**Line # Hexloc Verb**

This is read left to right, top to bottom.

Look for an Offset that is just lower than the Displacement. Then note the statement number in the program.

### 1.3 FINDING THE STATEMENT IN ERROR

Locate this statement in the Cobol Compile listing.

In our case it is :

**ADD INSULT TO INJURY**

### 1.3 FINDING THE STATEMENT IN ERROR

In order to find the Working Storage areas in the dump, we need to know 4 things about those working Storage items

- The BLW (Base Locator for Working Storage)
- The Displacement
- The Length

- The Picture (how it's defined)

### 1.3 FINDING THE STATEMENT IN ERROR

The Cobol Data Division Map in the Compile listing output will contain most of this information

### 1.4 TASK GLOBAL TABLE

Cobol allocates a control block with the invocation of every Cobol program. This is :

- **Task Global Table (TGT)**

This Control Block is created at the beginning of the program, and allocated in CICS User Transaction storage.

Understanding the role of the Task Global Table is fundamental to debugging any COBOL program**.**

Let's look at it more closely :

### 1.4 TASK GLOBAL TABLE

The **TGT** is printed in the Compiler output, provided **OFFSET** or **LIST** is specified as a compiler option.

The TGT has the eyecatcher in the transaction dump **'3TGT'** at offset x'48' from the beginning.

The TGT used to be used for saving the registers in the Application Register Save area. However, now this Register Save area is now handled by the Language Environment DSA (DYNAMIC SAVE AREAS). The TGT holds the Entry point address of the program, the Working Storage address, as well as the BLWs and BLLs.

The **Base Locators for Working Storage** and the **Base Locators for Linkage Section** are always in the variable portion of the TGT. Their offsets need to be verified by the layout of the TGT in your Compile listing

Nb. Due to the nature of the dump output, the beginning of the TGT may not be printed in the dump as it contains low-values, and the dumping process does not print repeated lines of low-values. So the eyecatcher '**3TGT**' is very important.

The following 2 pages show the TGT as printed in the Cobol Listing.

### 1.4 TASK GLOBAL TABLE

The above is the TGT as displayed in the COBOL Compile output

**1.4 TASK GLOBAL TABLE**

The above is the TGT as displayed in the COBOL Compile output (cont)

**1.5 LE DSA**

The Exec Interface Program handles to Call and provides the handshake between the Program and the CICS function. The main function of the EIP is to interpret the Call for CICS services. issued from the application and hand control over to the relevant management module invoked to deal with the request. The Call is turned into a 2-bytes Function code.

The EIP subroutine that contains the address of the EIP, is usually in link-edited in front of the COBOL program.

1) The application issues an 'EXEC CICS READ'.

2) Control passes to the Exec Interface Program via Exec Interface subroutine. This routine is Linked into the main program load module during Compile and Link processing.

3) The EIP saves the Application's Registers in the Applications Register Save Area. For COBOL, this is to found at the beginning of the **Dynamic Save Area** that has already been established by LE (Language Environment).

4) EIP then invokes the required management module to perform the function, which checks the validity of the Call.

5) Upon return EIP restores the Application's Registers and passes the return code and control back to the instruction following the Call.

**1.5 LE DSA**

The above is the LE DSA as displayed in the COBOL Compile output

**1.5 LE DSA**

Layout of the LE DSA

**1.5 LE DSA**

CEEMSG is a Sysout dataset defined in the CICS Startup JCL and used by LE to trap abending information.

## 1.6 BASE LOCATOR CELLS

The Cobol Compiler assigns **BLW's** (Base locators for Working Storage Cells), to the Application's Working Storage.

The Compiler assigns **BLL** Cells to the Application's Linkage Section. A Cell is simply 4 bytes to hold an address and Cells are given numbers.

These Cell numbers can be found in the **Data Division Map** in the Compiler output.

## 1.6 BASE LOCATOR CELLS

The COBOL **BLWs** and **BLLs** can be found by scanning the variable portion of the **TGT**.

The **BLW** number assignment is used then as an index into the **BLW**s stored in the TGT, where a series of addresses are held, depending on the size of the Application's Working Storage.

For the **BLL** Cells; CICS sets up the following:

**BLL** = 0 is low-values

**BLL** = 1 is the address of the EXEC INTERFACE BLOCK

**BLL** = 2 is the address of DFHCOMMAREA (if it exists)

## 1.6 BASE LOCATOR CELLS

How to find the TGT in the Transaction dump from the LE DSA, which is located from Register 13, that we see on the first page of the dump.

The LE DSA has an eyecatcher in the first 4 bytes. It is **00104001**

## 1.6 BASE LOCATOR CELLS

Then how to find the BLWs and BLLs in the TGT, in the Transaction dump, using the TGT Map from the Cobol Compile listing.

From there we can locate the Working storage items, by adding the Displacements that we found in the Data Division map to the BLW 0 address

## 1.7 EXEC CICS LINK

These are the LINK levels as displayed in the Transaction dump

### 1.7 EXEC CICS LINK

The **Program Environment Save Area - PESA** is created whenever a Program issues an EXEC CICS LINK. It is created by the Program Manager Domain

### 1.7 EXEC CICS LINK

The **Program Environment Save Area - PESA** is created whenever a Program issues an EXEC CICS LINK. It is created by the Program Manager Domain

### 1.8 COBOL CALL

However if the failure is in a COBOL program that has been 'called' using the COBOL static **CALL** statement, then CICS will have no knowledge of the existence of this program. In order to find the address of this program, it is necessary to check how far down the Load Module the program is located. This is the offset down the program storage. The utility **AMBLIST** can be used to ascertain the layout of the Load Module, as can other vendor software utilities.

```
//JOBNAME      JOB,'ACCNT','AMBLIST',CLASS=A,MSGCLASS=X,NOTIFY=USERID
//STEP1 EXEC PGM=AMBLIST
//SYSPRINT DD SYSOUT=*
//SYSLIB           DD DSN=THE.LOAD.LIBRARY,DISP=SHR
//SYSIN  DD *
  LISTLOAD MEMBER=membername,output=xref
//
```

In the Output look for the heading 'Control Section', then review the data under headings : LMOD LOC NAME LENGTH TYPE

### 1.8 COBOL CALL

```
CONTROL SECTION                              ENTRY
          LMOD LOC    NAME     LENGTH  TYPE            LMOD LOC  CSECT LOC     NAME
             00    DFHELII       26   SD
                                                 00       00     DFHEPIN
```

```
                08      08      DFHEI1
                08      08      DFHEI8
                08      08      DFHEI7
                08      08      DFHEI15
                08      08      DFHEI11
                08      08      DFHEI12
                08      08      DFHEPI
                08      08      DFHEI14
                08      08      DFHEI6
                08      08      DFHEXEC
                08      08      DFHEI17
                08      08      DFHEI18
                08      08      DFHEI2
                08      08      DFHEI10
                08      08      DFHEI01
                08      08      DLZEI01
                08      08      DFHEI13
                08      08      DLZEI02
 28    CICSCALL      FC0   SD
FE8    CEESG005       18   SD
1000   MGDATEIN       F28   SD
```

**1.8 <u>COBOL CALL</u>**

The **AMBLIST** will produce quite of lot of output, and initially it might seem too much. However look for the above output and this will tell you the breakup of the Load Module.

As can be seen in this example the program we coded, *CICSBRSJ* begins x'20' down the Load Module

It is important to understand the make up of the Load Module as the error might be in a 'called' subroutine module.

### 1.8 COBOL CALL

In the above example there are many modules that combine together to form the load module. It is important that the correct module is located in the Transaction Dump.

Using the above example, assume that **CICSPROG** has a load point of **C24680**, the entry point, and the address that is stored in the TGT, will therefore be **C246C8**.

**PROG1** subroutine will have an entry address of **C24680 + B28 = C251A8**. If the PSW had an address of **C251C8**, then we simply subtract **C251A8** (entry point) from **C251C8**.

Ensure the length of the load module matches the load module being used by the transaction. The Program Control information from the Program Manager Domain, which is listed in the Transaction Dump, lists the length of the module. The Module index at the end of the Transaction dump, also has the program's length

### 1.9 THE REGISTER SAVE AREA

The layout of the **REGISTER SAVE AREA** is an IBM convention. All systems follow this. A **REGISTER SAVE AREA** is as follows:

Register 13 is used to contain the address of a **REGISTER SAVE AREA**. So we should always look at the contents of this register.

The return address of where the Call came from can be traced back through Register 14.

If we can find the entry address of the program (module index at the end of the transaction listing), or the **CEEMSG** output, or x'100' in the TGT, then we can subtract that address from the address in Register 14, and find the displacement into the program, where the Call came from. Now we only have to find the **OFFSET** listing in the COBOL Compiler output to find the statement number relative to this offset.

The first 72bytes (48 in Hex) of the LE **DSA** is a Save Area. This is the Application Register Save Area, that we mentioned in session 3.

### 1.9 THE REGISTER SAVE AREA

So at any time the current application registers can be determined by scanning the Save Area in the **DSA**

Recall the **DSA** can be found by following the address in Register 13. Remember, this Register points to a Save Area. Offset x'4' points to the previous DSA in the stack, and so on... This stack is created by LE when the Exec CICS API is not involved in the transfer of the program.

The **DSA** will contain Register 14 at offset x'C', which is the return address into the program from where the Call (EXEC CICS command) came.

Let's look at an example of what a DSA would look like in the Transaction Dump

### 1.9 THE REGISTER SAVE AREA

As can be seen the first 72 (x'48') bytes in the DSA is a Register Save Area. This is referred to as the APPLICATION REGISTER SAVE AREA

By analysing this Save area, we can determine the contents of the Registers at the time of the Call.

This Save Area would be updated with the Registers on every Call

Remember all EXEC CICS commands are CALLs

So by analysing the contents of Register 14, we can subtract the Program's Entry Point address from this, and using the Offset listing from the Compiler, we can locate WHICH CALL is the current one.

Let's look at the Save Areas and how to follow them when a Subroutine Calls another Subroutine.

### 1.9 THE REGISTER SAVE AREA

The above diagram highlights how modules that **CALL** other modules are linked together. The **DSA** is used to hold the calling modules registers. The address of the calling programs Save Area can be found in the second word of the called programs Save Area.

This is important because for COBOL CALL statements, it is only by this methodology that each programs Working Storage can be found. In order to find each programs **TGT**, it is necessary to find offset x'5C' in the DSA. The TGT will have an eye-catcher at offset x'48'. This '**3TGT**'.

In all cases offset x'100' must be checked for the correct programs entry point address, and therefore the correct Working Storage.

Remember Register 14 will contain the return address of the statement after the CALL in the calling program

### 1.9 THE REGISTER SAVE AREA

In the CALL process, a number of internal activities occur :

**Register 1** Addresses the Parameters passed in the CALL USING …

**Register 13** Addresses a Save Area. For LE this would be the current DSA

**Register 14** contains the return address into the program immediately following the CALL

**Register 15** contains the address of the Call'd program, or low-values to indicate a Return code.

01 Param1 PIC …

01 Param2 PIC …

01 Param3 PIC …

So :

Call S-R1 USING Param1, Param2, Param3

**Register 1** will contain the address of @Param1|@Param2|@Param3

### 1.10 PROGRAM CONTROL BLOCKS

The above diagram shows the relationship between the Task Global table and the Working Storage areas. The Program can be clearly seen. The Working storage address that is at offset x'114' into the TGT, contains the same address as BLW=0, in the variable portion of the TGT

The Base Locators for Linkage Section can be seen. The EIB is always BLL=1. This means that EVERY program that runs under the transaction has exactly the same address for BLL=1, as there is only one EIB per transaction.

BLL2 always contains the address of the DFHCOMMAREA, (if it exists).

### 1.11 INTERVAL CONTROL VALUE RUNAWAY (ICVR)

The **ICVR** parameter controls the amount of CPU time a task is allowed to consume before calling CICS for a service.

This parameter is specified in the **SYSTEM INITIALIZATION TABLE (SIT),** and is a system wide value, that is all transactions fall within its control.

The SIT is the parameter table for CICS. CICS reads this table when it initialises.

The current ICVR setting can be determined by issuing the CEMT INQ SYSTEM command. The RUNAWAY value will specify the time in milliseconds

The default for the parameter is 5 seconds, and can be dynamically changed the following command

**CEMT SET SYSTEM RUNAWAY (VALUE)**

A runaway time can be established on an individual transaction basis as well.

## 1.12 TECHNIQUES FOR DEBUGGING LOOPS

The **'EXEC CICS ENTER TRACENUM'** is an excellent method of narrowing the loop.

The format of the command allows for a Traceid or number, to be associated with the command. This Trace number can be from 0 to 199. This command also allows an 8 byte character field of text to be associated with it. Also up to 4000 bytes of Working Storage can be displayed. The command is as follows:

**EXEC CICS ENTER TRACENUM (01)**

**RESOURCE ('TEXT')**

**FROM (W/S)**

The **'EXEC CICS ENTER TRACENUM'** does not cause CICS to return to the Dispatcher, so these two commands do not reset the timer (**ICVR**).