NORTEL NETWORKS

Maintenance Guide

Meridian 1 Corruption Guide

Author:	Naresh Mistry
Approved:	David Henley
Date:	March 24 1999
Issue:	2.0
System:	Merdian 1

Keywords: CORRUPTION

Abstract: This document provides info about known corruption's..

PROPRIETARY INFORMATION

Table of Contents

1.0 Introduction 7

2.0 TN type corruption.. 8

2.1 Symptoms of Problem
2.2 Software Stucture
2.3 How to Resolve Problem

3.0 DN type corruption.. 11

3.1 Symptoms of Problem
3.2 Software Stucture
3.3 How to Resolve Problem

4.0 Group Hunt Corruption (GPHT) 15

4.1 Symptoms of Problem
4.2 Software Stucture
4.3 How to Resolve Problem

5.0 Dass/Dpnss Corruption 20

5.1 Symptoms of Problem
5.2 Software Stucture
5.3 How to Resolve Problem

6.0 Loop/Superloop Corruption 27

6.1 Symptoms of Problem
6.2 Software Stucture
6.3 How to Resolve Problem

7.0 BRI Corruption.. 30

7.1 Symptoms of Problem
7.2 Software Stucture
7.3 How to Resolve Problem

8.0 Set Relocation Corruption 37

8.1 Symptoms of Problem
8.2 Software Stucture
8.3 How to Resolve Problem

9.0 ACD Corruption.. 42

9.1 Symptoms of Problem
9.2 Software Stucture
9.3 How to Resolve Problem

Table of Contents

10.0 TTY corruption.. 50
10.1 Symptoms of Problem
10.2 Software Stucture
10.3 How to Resolve Problem

11.0 BFS Corruption 54
11.1 Symptoms of Problem
11.2 Software Stucture
11.3 How to Resolve Problem

12.0 Authcode Corruption 58
12.1 Symptoms of Problem
12.2 Software Stucture
12.3 How to Resolve Problem

13.0 ESN Corruption.. 61
13.1 Symptoms of Problem
13.2 Software Stucture
13.2 How to Resolve Problem

14.0 Patch Corruption.. 64
14.1 Symptoms of Problem
14.2 Software Stucture
14.3 How to Resolve Problem

15.0 ACD SCHED Block Corruption 67
15.1 Symptoms of Problem
15.2 Software Stucture
15.3 How to Resolve Problem

16.0 MSDL Port Corruption 69
16.1 Symptoms of Problem
16.2 Software Stucture
16.3 How to Resolve Problem

17.0 DCH Channel Corruption.. 74
17.1 Symptoms of Problem
17.2 Software Stucture
17.3 How to Resolve Problem

Table of Contents

18.0 TTY Corruption on 11c 80

18.1 Symptoms of Problem
18.2 Software Stucture
18.3 How to Resolve Problem

19.0 XPEC Corruption on Superloop.. 83

19.1 Symptoms of Problem
19.2 Software Stucture
19.3 How to Resolve Problem

20.0 CDN Corruption.. 92

20.1 Symptoms of Problem
20.2 Software Stucture
20.3 How to Resolve Problem

21.0 FFC Corruption... 97

21.1 Symptoms of Problem
21.2 Software Stucture
21.3 How to Resolve Problem

22.0 ADAN Corruption.. 102

22.1 Symptoms of Problem
22.2 Software Stucture
22.3 How to Resolve Problem

23.0 TRK Corruption... 106

23.1 Symptoms of Problem
23.2 Software Stucture
23.3 How to Resolve Problem

24.0 LAPD (BRI) Corruption... 110

24.1 Symptoms of Problem
24.2 Software Stucture
24.3 How to Resolve Problem

25.0 Appendix 113

25.1 CDNXPTR
25.2 SCLMHTPTR
25.3 LOG_IO_PTR
25.4 DTSLHT_PTR
25.5 CRMHPTR
25.6 QUEUE_ADDR

Table of Contents

25.7 SET_RELOC_TABLE
25.8 P_MSDLMISP_MHPTR
25.9 BCS TEMPL_HDR
25.10 PBX_TEMPL_HDR
25.11 CONFIGLOOP
25.12 SYS_XPEC
25.13 CON_DDCS_FLAG
25.14 PATCH_HTPTR
25.15 IO_TABLE_PTR
25.16 P_VAS_TBL_HDR
25.17 CRSTART
25.18 CREND
25.19 P_BRI_PROTMHTPTR

(1) References.

- Meridian 1 Software Tools Course

1 Introduction

This document has been written to provide information about known CORRUPTION on Meridian 1.

For each corruption type, there are 3 sections:
(a) Symptoms of problem.
(b) Software Structure: description of problem.
(c) How to resolve problem with debug/pdt.

See chapter 16 for list of pointer values for varrious software release to resolve corruption.

This document is written for the benefit of:

(1) Nortel software engineers in a Support Role (CTS).

(2) Distributor Engineers who have some knowledge of
Meridian 1 software structures.
(Have attended the "Meridian 1 Software Tools Course").

The document can be used as an aid to resolving corruption problems without having to refer to the Meridian 1 software code.

It is intended that this document should form the basis for an investigation by Nortel Technology to find the root causes of all the corruption types listed in this doccument, and/or produce defensive code/workaround code to prevent the problems occurring.

2 TN Corruption

2.1 SYMPTOMS OF PROBLEM

TN corruption is caused when the protected and the unprotected unit pointers are invalid or missing from the protected/unprotected card pointers.

The corrupted TN can be printed as normal in LD 20, but can not make any changes or be taken out in LD 11 will get **SCH0128**

example

```
REQ:
TYPE: 2616
TN   4 0 0 0
SCH0128  = Terminal does not exist.
TN   ****

REQ: OUT
TYPE: 2616
TN   4 0 0 0
SCH0128 = Terminal does not exist.
TN
```

2.2 SOFTWARE STRUCTURE

see TNTREE STRUCTURE diagram

2.3 HOW TO RESOLVE PROBLEM

IN DEBUG/PDT

1) DO **TNT <TN>**

example

```
#TNT 4 0 0 0
 UNEQPD SLOOP TN 000400
 GP 120739 SLP 12075F 1F8DDA  CD 1207AD 1F8DC0  LN 12083B 000000
```

as you can see the unprotected unit pointer is missing, so we need to take this TN out in debug also if there are any **DN** associated with this **TN** you **MUST** take them out the **DNTREE** (see DN Corruption)

2) in this case we have two SCR KEYS as follows, Take them out in the DNTREE

KEY 00 SCR 2004

```
01
02 SCR 2003
```

3) The next step is to take out the TN in debug/pdt

print the protected card pointer and take out the protected unit pointer

example

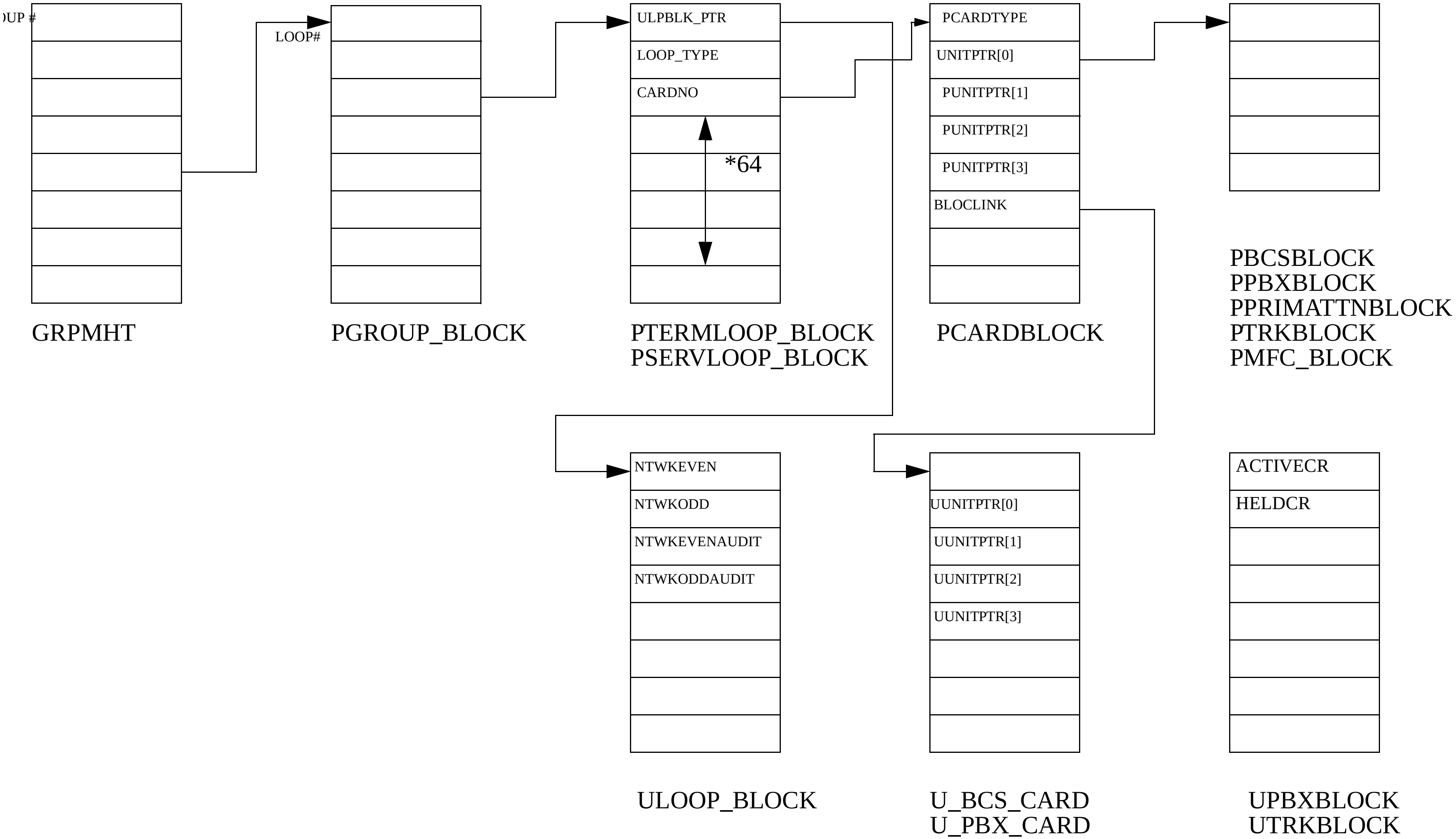
```
#P 1207AD 5
1207AD : 002060 12083B 12085C 120888 1208A7

#W 1207AD
1207AD : 002060 /
1207AE : 12083B /0
```

To confirm if TN has been removed do **TNT <TN>**

```
#TNT 4 0 0 0
 UNEQPD SLOOP TN 000400
 GP 120739 SLP 12075F 1F8DDA  CD 1207AD 1F8DC0  LN 000000 000000
```

The corruption has now been confirmed cleared and ready to be rebuilt

TNTREE STRUCTURE



3 DN Corruption

3.1 SYMPTOMS OF PROBLEM

Overlay 20; REQ PRT; TYPE DNB; DN XXXX; ->

No TN is printed, and no SCH0881 is printed.

It is not possible to configure a NEW set with the DN XXXX.

Example:

```
>LD 20
PT0000
REQ: PRT
TYPE: DNB
CUST 0
DN 2000
DATE
PAGE
DES
NO ACT SINCE NO DATE

NACT
```

This shows a DNTREE entry exists for DN 2000, but the TN referred to does not exist.

```
#DNT 0 2000
DIG 4 BCS
15D3F4 :   008208 000000 000000 000000 000000 0003FF 000000 000401

#TNT 401
 UNEQPD SLOOP TN   4  0  0   1
 GP 158677 SLP 15869D 1F9234  CD 1586EB 1F921A  LN 000000 000000
```

The corruption is in the DNTREE.

3.2 SOFTWARE STRUCTURE

- see DNTREE Structure diagram

3.3 HOW TO RESOLVE PROBLEM

In debug/pdt

Example: {21.19 omega s/w}:

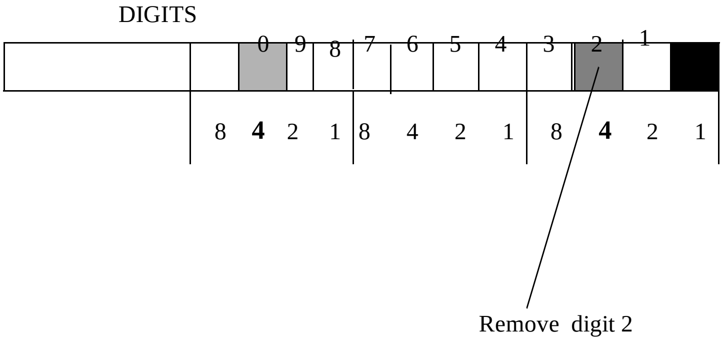
ADDRESS OF CDNXPTR ON 21.19 OMEGA IS: 8922 HEX.

```
#P 8922
008922 : 15D0D0

#P 15D0D0 10
15D0D0:0000DE 15D142 15D0DD 15D184 15D186 00000015D1F415D182
15D0D8 : 000000 000000 000000 000000 000000 000564 000000 15D167
#P 15D0DD 10
15D0DD : 000564 000000 15D167 000000 000000 15D3BC 1564EF 000000
15D0E5 : 15D0E9 000000 15D35C 000000 000400 000000 000000 000000
#P 15D35C 10
15D35C : 000400 000000 000000 000000 000000 000000 000000 000000
15D364 : 000000 000000 15D368 000000 000404 000000 15D3EC 000000
#P 15D368 10
15D368 : 000404 000000 15D3EC 000000 000000 000000 000000 000000
15D370 : 000000 000000 15D3F4 000000 000400 000000 000000 000000
#P 15D3F4 10
15D3F4 : 008208 000000 000000 000000 000000 0003FF 000000 000401
15D3FC : 008208 000000 000000 000000 000000 0003FF 000000 000402
```

The DNBLOCK needs to be removed for the corrupted DN.

Example:

```
#W 15D368
15D368 : 000404 /4
15D369 : 000000 /
15D36A : 15D3EC /
15D36B : 000000 /
15D36C : 000000 /
15D36D : 000000 /
15D36E : 000000 /
15D36F : 000000 /
15D370 : 000000 /
15D371 : 000000 /
15D372 : 15D3F4 /0
```



To confirm corruption has been cleared ok:

```
#DNT 0 2000
DIG 4 INV
```

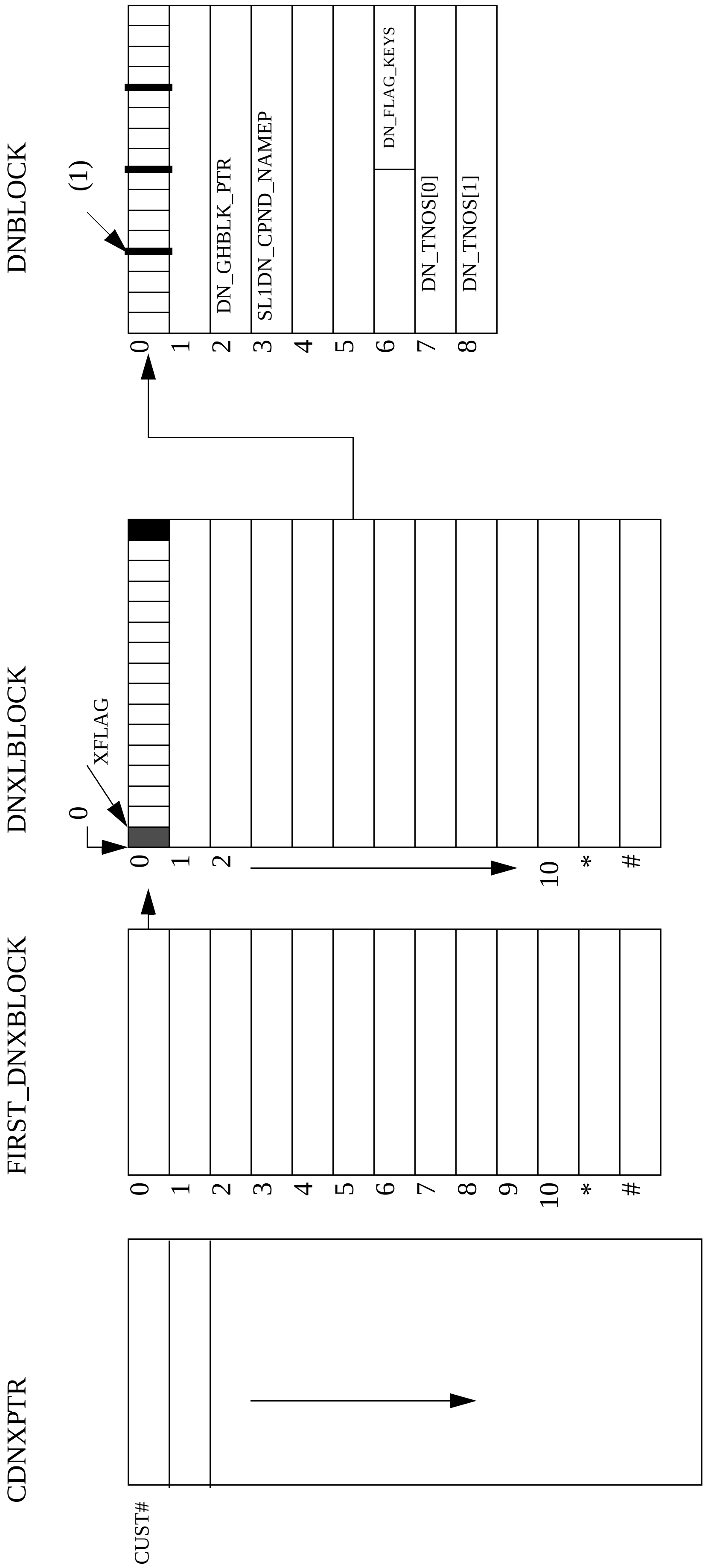
Corruption is now cleared, and confirmed cleared..

Note that what has been done above is to adjust 2 words of memory.

(1) Adjust bit map in 1st word of DNXLBLOCK.

(2) Set word relating to digit 0 to zero.

DNTREE STRUCTURE



4 Group Hunt Corruption (GPHT)

4.1 SYMPTOMS OF PROBLEM

If a Group Hunt corruption has occurred then SCH8825 will be displaced.

In our example, Print the set, we have key 00 as DN 2001, if we now try making any change to this DN we get SCH8825 saying that this DN must be removed from group hunt list, but when we print all of the group hunt list this DN does not appear.

```
LD 11
REQ: PRT
TYPE: 2616
TN   4 0 0 0
DNDR 0
KEY  00 SCR 2001    MARP

REQ: CHG
TYPE: 2616
TN   4 0 0 0
ECHG YES
SCH0618
ITEM KEY 00 SCH8825
         KEY
```

SCH8825 := That DN must first be removed from the GHT list

OVERLAY 20

Print all Group Hunt Lists, If DN 2001 does **NOT** appear as a member in any of the Group Hunt list,

then we have **Group Hunt corruption**

4.2 SOFTWARE STRUCTURE

see Group Hunt structure diagram

4.3 HOW TO RESOLVE PROBLEM

The best way to deal with this type of corruption is to zero out the group hunt pointer **(DN_GHBLK_PTR)** in the **DN** block.

In debug/pdt print the dn block

1) DNT <CUST> <DN> of the dn xxxx eg 2001 in our case

```
DBG000
#DNT 0 2001
DIG 4 BCS
15D3EC :   008208 000000 158779 000000 000000 0003FF 000000 000400
```

The corruption is in the DN block

Wordoffset 2 in the DN block hold the pointer to group hunt

data structure (DN_GHBLK_PTR) This pointer should not exist as there

is no entry for dn 2001 in LD 18 in any Group Hunt List

NOTE:- For pbx type set need to clear DNGHBLK_PTR
(word 8 in P_LINE_BLOCK)

CLEARING POINTER IN DEBUG/PDT

```
DBG000
#DNT 0 2001
DIG 4 BCS
15D3EC :   008208 000000 158779 000000 000000 0003FF 000000 000400

#W 15D3EC
15D3EC : 008208 /
15D3ED : 000000 /
15D3EE : 158779 /0
```

Should look like:

```
#DNT 0 2001
DIG 4 BCS
15D3EC :   008208 000000 000000 000000 000000 0003FF 000000 000400
```

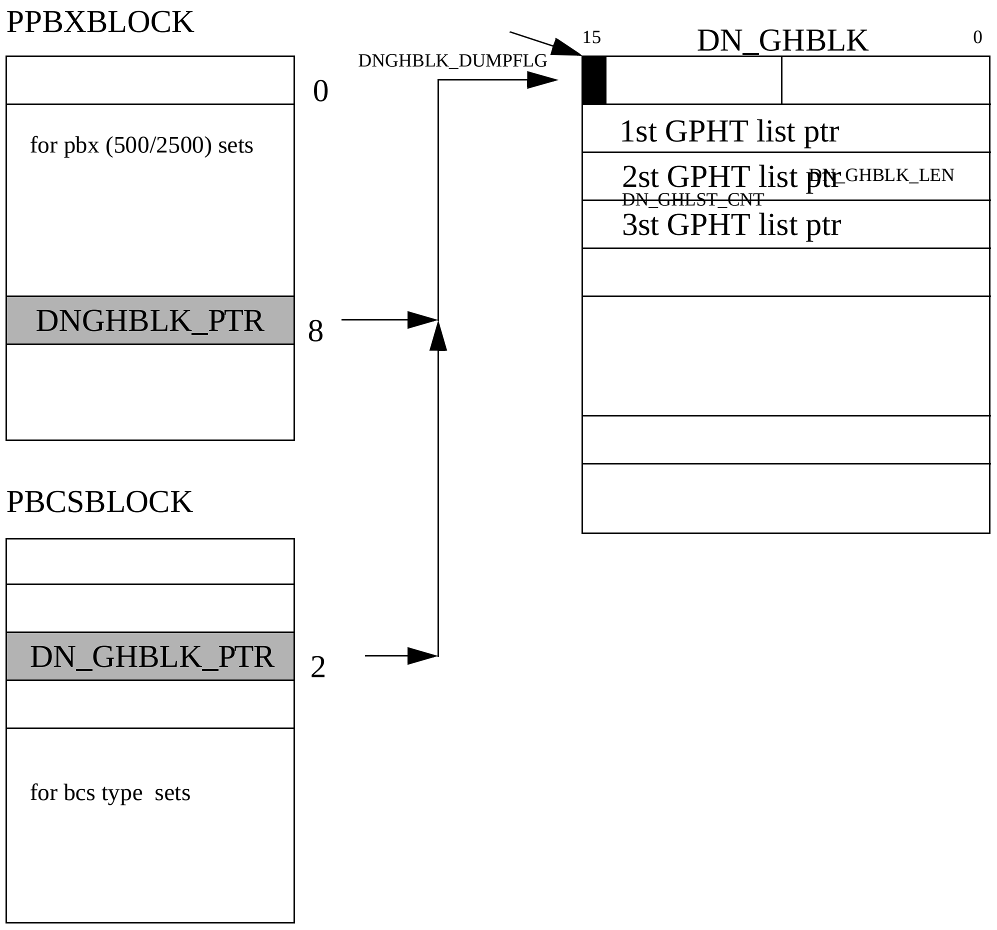
Corruption has now been confirmed cleared.

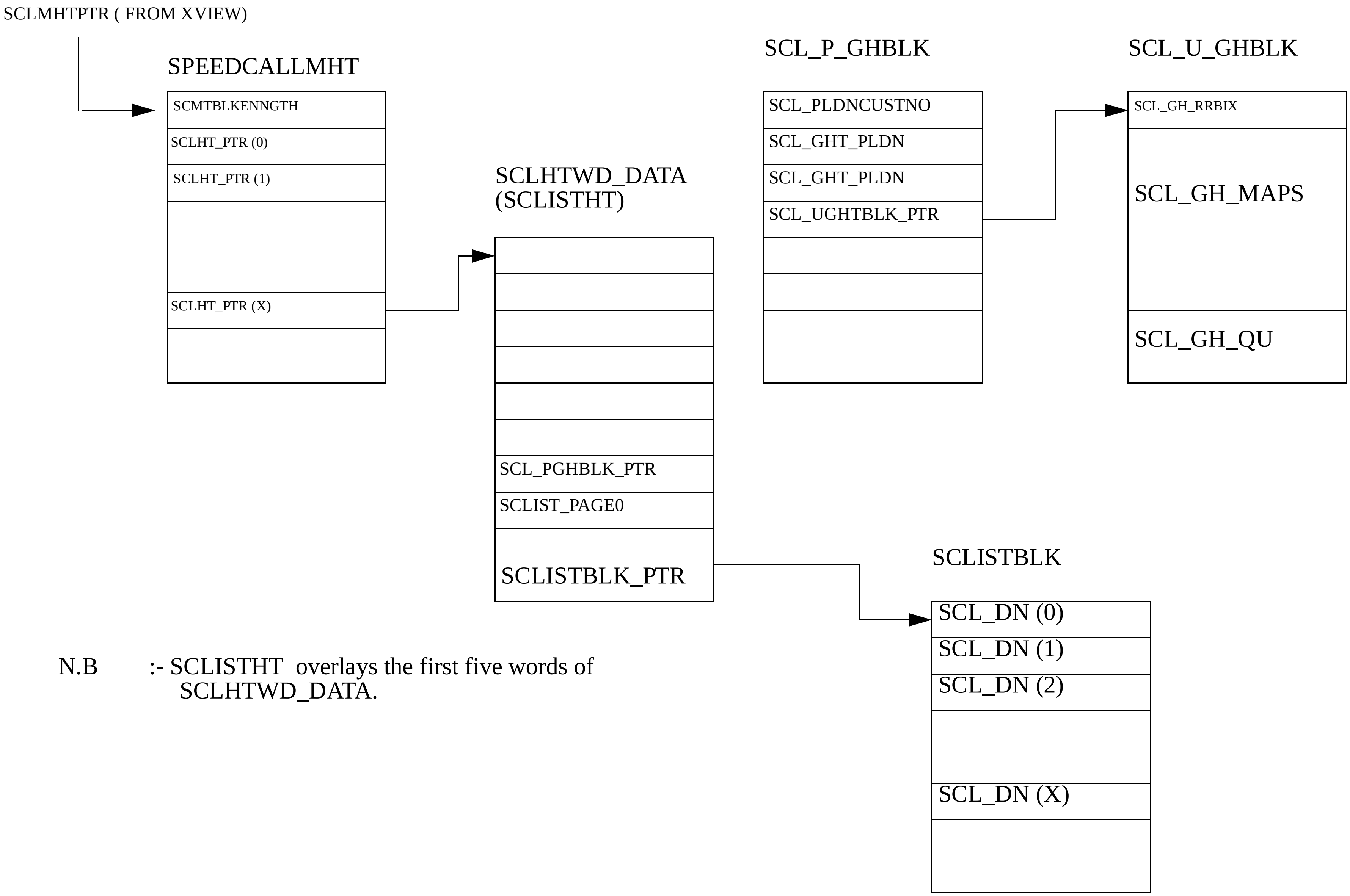
In LD 11, Now dn 2001 can be changed to what ever

```
REQ: CHG
TYPE: 2616
TN   4 0 0 0
```

```
ECHG YES
SCH0618
ITEM KEY 00 NUL
    KEY
ITEM

MEM AVAIL: (U/P): 304587    USED: 392293    TOT: 696880
.........ETC
```

GROUP HUNT STRUCTURE





N.B :- SCLISTHT overlays the first five words of SCLHTWD_DATA.

5 DASS/DPNSS Corruption

5.1 SYMPTOMS OF PROBLEM

1) LD 20 PRT TNB , TN <LOOP><CHANNEL>

example

```
REQ: PRT
TYPE: TNB
TN   25 1
DATE
PAGE
```

SCH0805 = Specified TN is invalid

2) in debug/pdt, TNT <TN>

NO UNIT PTR EXIST

example

```
#TNT 25 1
 UNEQPD TN  25  1 * 001901
 GP 158677  LP 15A863 1F48E5  CD 15A8B1 1F48B0  LN 000000 000000
```

3) LD 75, STAT DDCS <LOOP>

Channel concerned is unequipped.

4) LD 21, LTM <CUST><ROUTE No>

```
REQ: LTM
CUST 0
ROUT 99
TYPE TLS
TKTP IDA
ROUT 99
TN  025 01 MBER  1
TN  025 02 MBER  2
TN  025 03 MBER  3
TN  025 04 MBER  4
```

All members appear to exist.

NOTE usually all members on the loop have been corrupted.

5.2 SOFTWARE STRUCTURE

- see structure diagram

5.3 HOW TO RESOLVE PROBLEM

In debug/pdt

NULL the pointer to the trunk member in the route data block.

DRP <CUST NUM><ROUTE NUM>

example

```
#DRP 0 99
CUST 0 ROUT 99 P 157CC4 U 1F977B T 15D50A R 000000 VU000000 VT000000
```

```
#P 157CC4 5
157CC4 : 008384 00605C 000000 000000 15D50A
```

WORDOFFSET 4 IS THE **TRKLIST_PTR**

check for TN’S involved do TNT <TN> Show’s packed format

```
#P 15D50A 10
15D50A : 000005 001901 001902 001903 001904 000003 001905 001906
15D512 : 1F931F 00000E 1F9279 000002 1F54FE 008000 000000 00000
#TNT 25 1
 UNEQPD TN  25  1 * 001901
 GP 158677  LP 15A863 1F4904  CD 15A8B1 1F48CF  LN 000000 000000
#TNT 25 2
 UNEQPD TN  25  2 * 001902
 GP 158677  LP 15A863 1F4904  CD 15A8B1 1F48CF  LN 000000 000000
#TNT 25 3
 UNEQPD TN  25  3 * 001903
 GP 158677  LP 15A863 1F4904  CD 15A8B1 1F48CF  LN 000000 000000
#TNT 25 4
 UNEQPD TN  25  4 * 001904
```

```
GP 158677  LP 15A863 1F4904  CD 15A925 1F48C6  LN 000000 000000
```

TAKE OUT **TRKLIST_PTR** IN **P_ROUTE_DATA**

```
#W 157CC4
157CC4 : 008384 /

157CC5 : 00605C / <SPACE>
157CC6 : 000000 / <SPACE>
157CC7 : 000000 / <SPACE>
157CC8 : 15D50A /0
```

NOTE:- When Clearing DPNSS type corruption, we need to also take out **VIRT_TRKLIST_PTR** wordoffset (57) from **P_ROUTE_DATA** block

- TAKE OUT THE ROUTE (LD 16)

```
>LD 16
RDB000
MEM AVAIL: (U/P): 307200    USED: 389680    TOT: 696880
DISK RECS AVAIL: 1426
REQ  OUT
TYPE RDB
CUST 0
ROUT 99
```

- NEED TO TAKE OUT DDSL/DDCS
  CAN NOT REMOVE IN OVERLAY 74 STILL CORRUPTED

```
REQ  OUT
TYPE DDSL
DDSL 9
SCH8849  1306
SCH8849  1329
```

IN DEBUG/PDT

THE ADDRESS FOR **DTSLHT_PTR** = 8B34 (s/w 2119 omega)

```
#P 8B34 3
008B34 : 15CDC4 000000 000000
```

WORDOFFSET (0) = CUST 0

WORDOFFSET(1) = CUST 1

ETC.....

DTSLHT BLOCK

```
#P 15CDC4 10 (WORDOFFSET 9+1)=DDSL 9
15CDC4 : 000021 000000 000000 000000 000000 000000 15CE21 000000
15CDCC : 000000 000000 15CE03 000000 000000 000000 000000 000000
```

P_DTSL_CARD_BLK BLOCK

```
#P 15CE03 5
15CE03 : 000001 000000 1FB887 15CE0E 000000
```

P_DASS_LINK_BLK BLOCK

```
#P 15CE0E 20
15CE0E : 000913 00001D 000019 000125 001101 0040A4 000050 000101
15CE16 : 000100 000100 003278 004014 00007E 000000 000000 000000
15CE1E : 000000 000000 000000 000001 000000 1FC59E 15CE2C 000000
15CE26 : 000000 000000 1FB891 000000 000000 000000 000513 000019
```

NEED TO CLEAR THE CONTENT OF CHAN_CONFIGURED IN WORDOFFSET 12, 13,14 & 15.

```
#W 15CE0E
15CE0E : 000913 / <space>
15CE0F : 00001D / <space>
15CE10 : 000019 / <space>
15CE11 : 000125 / <space>
15CE12 : 001101 / <space>
15CE13 : 0040A4 / <space>
15CE14 : 000050 / <space>
```

```
15CE15 : 000101 / <space>
15CE16 : 000100 / <space>
15CE17 : 000100 / <space>
15CE18 : 003278 / <space>
15CE19 : 004014 / <space>
15CE1A : 00007E /0 <space>
15CE1B : 000000 /0 <space>
15CE1C : 000000 /0 <space>
15CE1D : 000000 /0
```

```
LD 75  DISABLE DDSL
LD 74 OUT DDSL 9
```

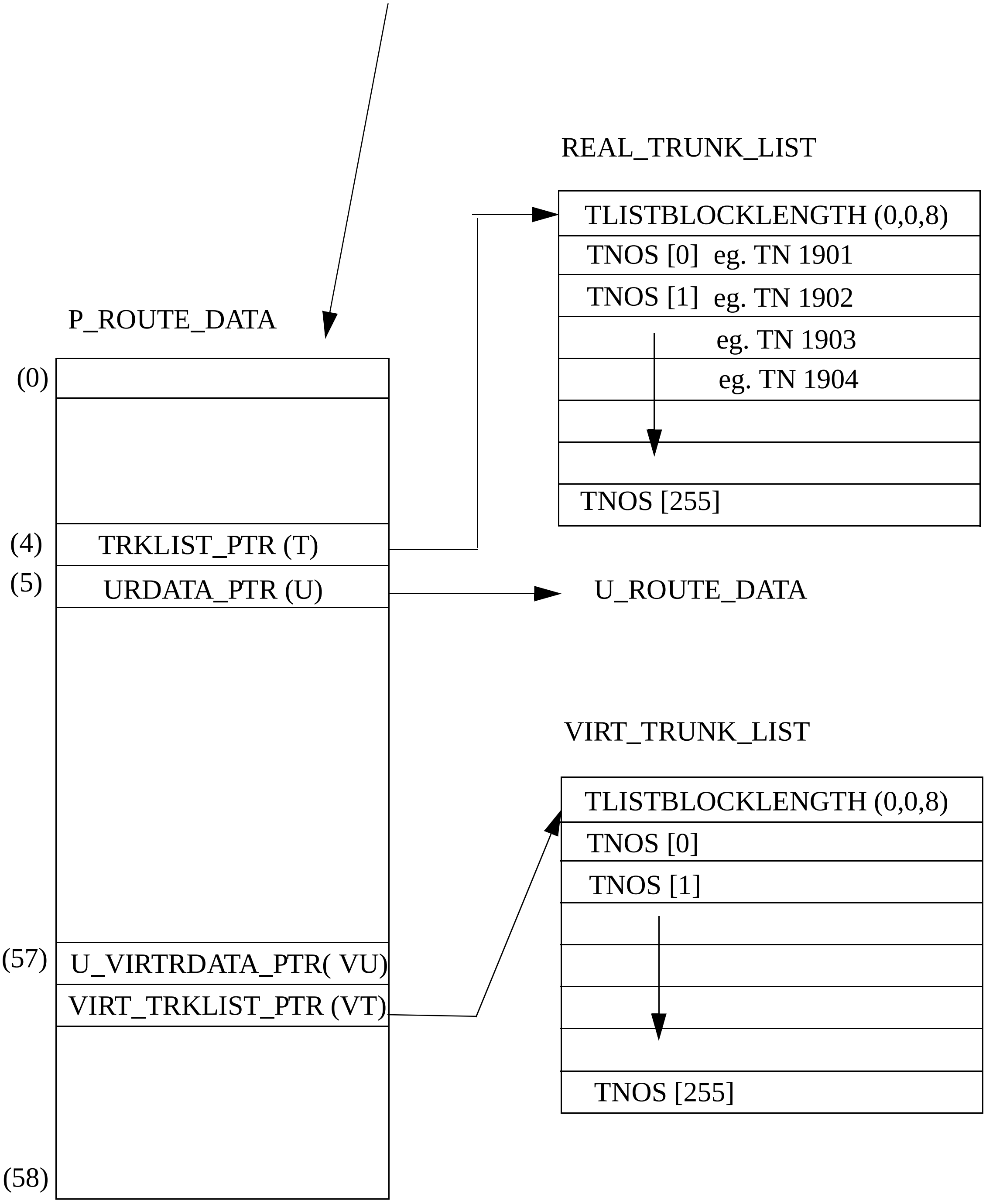
CORRUPTION HAS NOW BEEN CLEARED.

ROUTE DATA BLOCK STRUCTURE

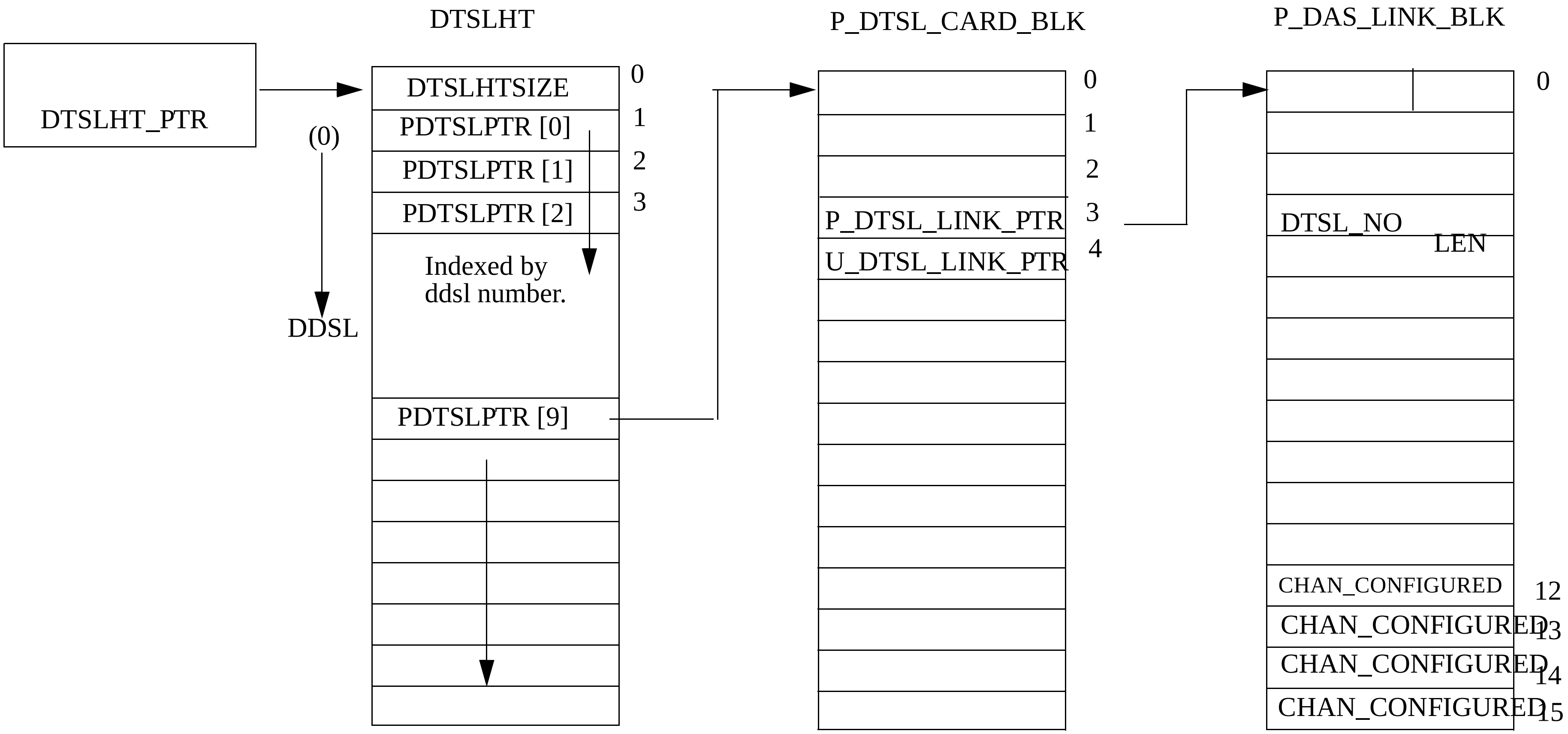
DRP <CUST NUM><ROUTE NUM>

example

```
#DRP 0 99
CUST 0 ROUT 99 P 157CC4 U 1F977B T 15D50A R 000000 VU000000 VT000000
```



DASS/DPNSS Channel Corruption



NB: ALL 4 CHAN_CONFIGURED word must be zeroed to clear corruption. Problem is seen when SCH8849 1329 is displayed.

6 Loop/Superloop Corruption

6.1 SYMPTOMS OF PROBLEM

If a Loop corruption has occurred then the loop number and type will not appear in the configuration record, also it is not possible to reconfigure or make any changes to this loop will get SCH0535.

In our example Loop 11 was configured as DDCS 11, but when printed in overlay 22, Loop does not exist.

example

1) LD 22,PRT CEQU

```
..      ..

..      ..

MPED 8D
 TERM
 REMO
 TERD
 REMD
 TERQ  014  016
 REMQ
 SUPL  004
 DDCS  000  008  010  025
 DTCS
 XCT   002
 TDS  * 002
 CONF * 003
 MFSD * 002
```

3) It is not possible to remove or add this Loop in LD 17, will get SCH0535
   (Attempted to remove non-existing loop or add existing loop)

```
REQ  CHG
TYPE CEQU
MPED
TERM
REMO
TERD
REMD
TERQ
REMQ
DDCS X11
```

SCH0535 = Attempted to remove non-existing loop or add existing loop.

DDCS

6.2 SOFTWARE STRUCTURE

- Not available

6.3 HOW TO RESOLVE PROBLEM

-

1) FIND THE ADDRESS OF **CONFIGLOOP**

For our example the address of the CONFIGLOOP for 21.19 Omega = 90E1

.NOTE : - CONFIGLOOP IS AN ARRAY OF .MAX_LOOPS where
.MAX_LOOPS = 160 LOOP

CONFIGLOOP (0,8)[.MAX_LOOPS], (1 BYTE PER LOOP)

-

2) PRINT THE CONFIGLOOP AT ADDRESS 90E1 FOR SAY 10

```
#P 90E1 10
          1  0    3  2    5   4    6  7    8  9   11 10 ...........ETC
0090E1 : 000511 00020E 000D0D 000D0D 000A11 001411 000505 000508
0090E9 : 000508 000505 000505 000505 00110A 000505 000505 000513
```

As you can see from the above data structure loop 11 has a value of **14hex** this is incorrect and should be changed to none existing loop with a value of **05hex,** this will completely remove this loop in configuration record and enable you to reconfigure again

```
#W 0090E1
0090E1 : 000511 /
0090E2 : 00020E /
0090E3 : 000D0D /
0090E4 : 000D0D /
0090E5 : 000A11 /
0090E6 : 001411 /0511
```

NOW THIS LOOP CAN BE RECONFIGURED LD 17

AS DDCS 11.

NOTE:- In this example we had NO trunks or other configuration associated with this loop so there no other corruption involved. please note if you were to remove a an existing corrupted loop/superloop with trunks, route, TN's or other configuration associated with that loop will cause major corruption which needs to be cleared.

7 BRI Corruption

7.1 SYMPTOMS OF PROBLEM

Corruption is caused when multiple DN's are configured on a TSP. The 1st DN is OK, but subsequent DN's do not appear in the DN tree and cannot be removed. SCH0670 is displayed in Overlay 27 when an attempt is made to OUT the TSP.

```
LD 27
REQ  PRT
TYPE TSP
DSL  8 1
OPT
USID 0
MPHC NO
SUPL_SVC
DN   291
  CT   VCE DTA
  MCAL 4
  CLIP YES
  PRES YES
  FEAT HTD FND SFD CFTD MWD FBD HBTD CFXD
  SSRV_ETSI

DN   292
  CT   VCE DTA
  MCAL 4
  CLIP YES
  PRES YES
  FEAT HTD FND SFD CFTD MWD FBD HBTD CFXD
  SSRV_ETSI

DN   291
  CT   VCE DTA
  MCAL 4
  CLIP YES
  PRES YES
  FEAT HTD FND SFD CFTD MWD FBD HBTD CFXD
  SSRV_ETSI

DN   292 *****************PROBLEM CORRUPTION
  CT   VCE DTA
  MCAL 4
```

Once the corruption has been cleared it is essential that Patch number **MLVT05603** is fitted. This prevents the reoccurrence of the DN corruption when multiple DN's are configured.

The patch is only neccesary up to Release 21.

NB. In clearing the corruption we remove all the DSL's and TSP's.
These will have to be manually reconfigured.

7.2 SOFTWARE STRUCTURE

- Not available

7.3 HOW TO RESOLVE PROBLEM

•

(1) Firstly print all the DSL's in Overlay 27.
Then print all the TSP's associated with each DSL also in Overlay 27.
This will provide a list of all the DN's on each TSP, these DN's have to then be removed from the DN tree using DEBUG.

•

(2) Having removed the DN's we now have to remove the DSL's themselves.
The protected card block is as follows :-

```
PSTRUCTURE [.P_TN_CARD] PCARDBLOCK
  INTEGER  PCARDTYPE  (0,11,5) ,
         UNITFAULT  (0,7,1) [4],
         DSL_FAULT  (0,7,1) [2],
         CARD_FAULT  (0,7,4),
         DISBL_BIT  (0,0,1) [4],
         DSL_DISBL  (0,0,2) [2],
         CARD_DISBL  (0,0,4),
         SC_FAULT  (0,4,1),
         CARD_DENSITY     (0,5,2),

  PPOINTER [.P_TN_LINE] PUNITPTR (1) [4],
  UPOINTER [.U_TN_CARD] BLOCKLINK (5),
  INTEGER  PUNITWORD       (1) [4] ,
         XTRUNK          (6,0,3),
         BRI_SUB_CDTYPE   (6,0,2),  % 0 = SILC,1 = UILC
         MISP_TN  (6,3,9),  % MSB SET TO 1 -> TN EXIST
         PACK_TYPE_DATA   (6,13,1),
         XDATA_CARD  (6,14,1),
         SYS_BRSC_IDX     (7,0,7),
         BRSC_SHELF  (7,7,1),
         BRIT_L1_DISBL    (7,8,1)[2],
         THF_FWTM         (7,10,1),
         TRK_BAR_STATUS   (7,11,1)[4];
```

When we do a TNT on the DSL TN the PCARDPTR is looking at this block.

•

(3) In DEBUG perform a TNT on the DSL TN.

For example TN 4 0 and TN 4 1 will both have the same protected and unprotected card pointers.

Now print the Protected Card Block ie.

```
#P 056840 8
05 6840: 008060  058840  058840  058640  058640  048860  0000A8  000000
```

The 2nd and 3rd words are the Protected unit pointers for TN 4 0.

The 4th and 5th words are the Protected unit pointers for TN 4 1.

The 6th word is the unprotected card pointer.

The 7th word holds the MISP TN for this DSL.

We have to clear words 2,3,4,5 and 7. DO NOT REMOVE THE UNPROTECTED CARD POINTER.

Now print the Unprotected Card Block ie.

```
#P 048860 8
04 8860:  000000  049961  0499A8  0499C8  0499E6  000000  000000   000000
```

The 2nd and 3rd words are the UnProtected unit pointers for TN 4 0.

The 4th and 5th words are the UnProtected unit pointers for TN 4 1.

We have to clear words 2,3,4 and 5.

At this point we have completed removed the DSL's 4 0 and 4 1 and all the DN's associated with the TSP's on these DSL's.

-

(4) We now have to remove the DSL's from the MISP block.

This is the Protected MISP Loop Block :-

```
PSTRUCTURE [.p_tn_loop] PMISPLOOP_BLOCK
upointer [.u_tn_loop] ulpblk_ptr (0),
integer loop_type            (1),
        absloop_type    (1,0,15),
        loop_disabled   (1,15,1),
        misp_index_no   (2, 0,8),
upointer [.u_basic]
        misp_obptr      (3),
        misp_xob_start  (4),
        misp_xob_end    (5),
        misp_orb_start  (6),
        misp_orb_end    (7),
integer
        PH_PRI_TN           (8),
        BRI_SET_APPL    (9,0,1),   % 1 = BRI "TERMINAL" APPLICATION
        BRI_TRK_APPL    (9,1,1),   % 1 = BRI TRUNK ( FOR PHASE II)
        BRI_MPH_APPL    (9,2,1),   % 1 = MPH APPL ( FOR PHASE III)
        BRIE_TRK_APPL   (9,3,1),   % 1 = BRI TRUNK (FOR REL20 UIPE)
        % spare bits    (9,4,4),   % for future expansion
        NO_OF_CARDS     (9,8,4),   % NO. OF BRI LINE CARDS DEFINED
        % spare bits    (10),
        BRI_CARDS_TN    (11)[.MAX_BRI_CARDS], % 4 BRI CARDS PER MISP
```

```
BRI_CARD1_TN    (11),% BIT 0 SET TO 1
BRI_CARD2_TN    (12),% TO AVOID TN (0, 0, 0, 0)
BRI_CARD3_TN    (13),
BRI_CARD4_TN    (14),
BRI_LINE_SIDS   (15, 0, 8) [.SIDS_PER_APPL], % BRI LINE
                     % APPLICATION SOCKET IDS
                     % [0] - MISP BASECODE
                     % [1] - BRI MAINT. SOCKET ID
                     % [2] - BRI CALL PROCESSING SOCKETID
                     % [3] - BRI ADMIN. MSGS  SOCKETID
BRI_TRK_SIDS    (17, 0, 8) [.SIDS_PER_APPL],  % BRI TRUNK
                     % APPLICATION SOCKET IDS
                     % (RESERVED FOR FUTURE USAGE)
```

In Overlay 27, print the MISP Loop on which the DSL's were configured.

ie.

```
REQ   PRT
TYPE  MISP
LOOP  XX

LOOP  XX
APPL  BRIL
DPSD  NO
CARD1 4  0  0
CARD2
CARD3
CARD4
```

In DEBUG, TNTRANS the MISP TN ie. 2 0. Now print the Protected Loop Block. ie.

```
#P 5915B 10
05 915B:  XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX
05 9163   XXXXXX 000101 000000 000010 000001 000001 000001 000000
```

WORD 9 tells us we have 1 BRI card defined.

WORD 11 gives us the TN of that card.

WORDS 12,13 and 14 are 1 ie. No TN configured.

We have to set WORD 9 to 000001 and WORD 11 to 000001.

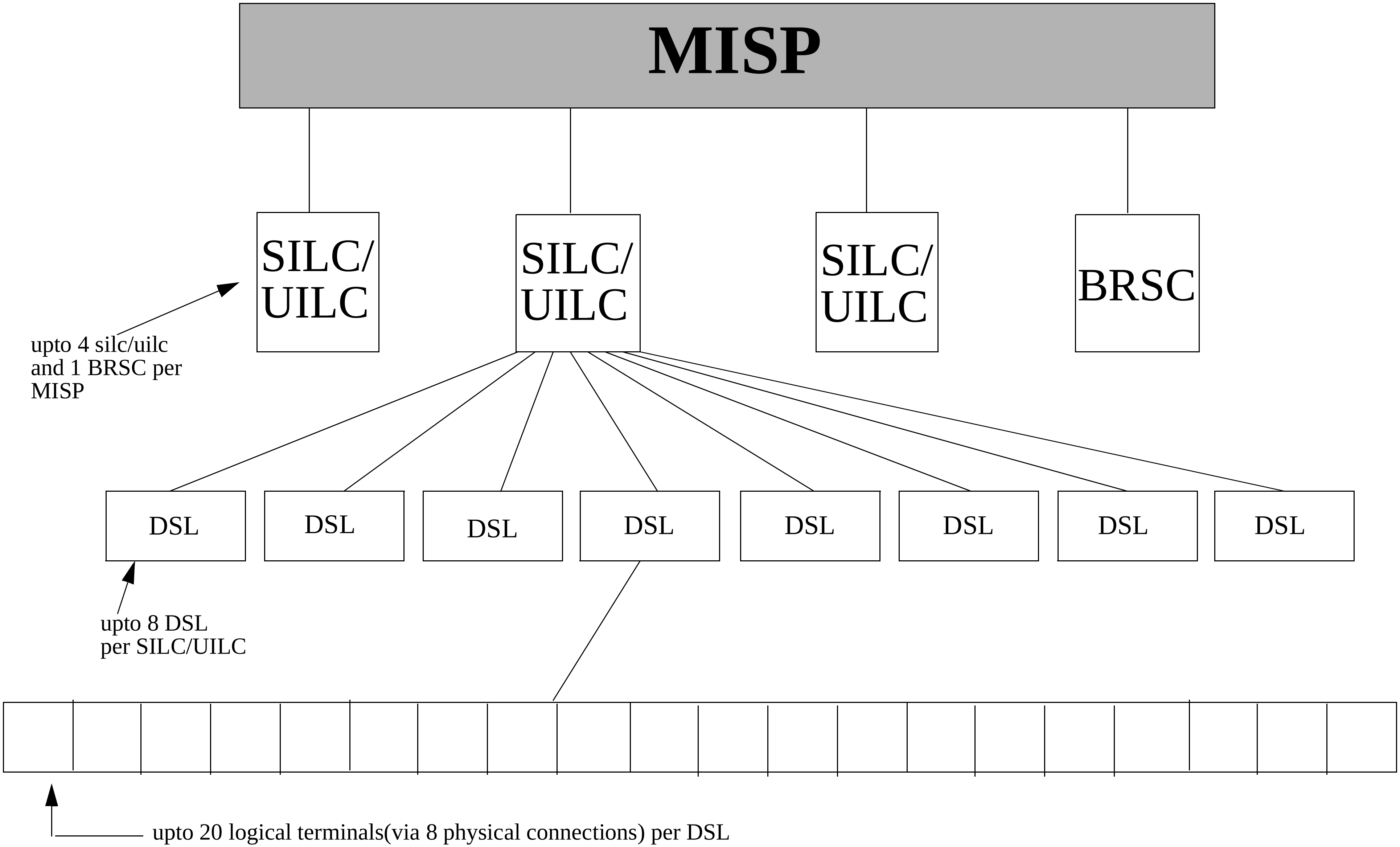
We have now removed the DSL TN from the MISP Loop Block.

-

(5) At this point a DATA DUMP is strongly advisable.

Once the DUMP is complete, insert patch 5603 (if applicable) and begin reconfiguration.

ISDN BRI OVERVIEW DIAGRAM



Maximum logical terminals per MISP = max SILC/UILC(4)*max DSL(8)*max logical terminal (20) = 640

8 Set Relocation Corruption

8.1 SYMPTOMS OF PROBLEM

example 1

- 1) LD 21, PRT, SRDT

  CAN NOT REMOVE SET IN RELOCATION TABLE

```
REQ PRT
TYPE:
TYPE SRDT

  OLD TN     TYPE  ID SERNUM  NT CODE  COL RLS
 93  XX0 00  2616  ID: 018D5D NT2K16C
```

example 2

- 2) LD 11, CAN NOT CHANGE SET, COMPLAINS ABOUT SET IN RELOCATION WHEN IT’S NOT. YOU MAY GET SCH2501

SCH2501 = An attempt was made to change a telephone that is in the process of relocating

8.2 SOFTWARE STRUCTRUE

- see Set Relcation Structure

8.3 HOW TO RESOLVE PROBLEM

solution of example 1

1) IN DEBUG/PDT TAKE OUT SR_ELEMENTS IN RELOCATION TABLE STRUCTURE, THIS CONTAINS 14 WORDS OR 15 WORDS FOR SOFWARE RELEASE 20 ONWARDS. (SEE SET RELOCATION STRUCTUE DIAGRAM)

FIND THE ADDRESS FOR SET_RELOC_TABLE (2119 111 =924D)

```
#P 924D
00924D : 15D4F1

#P 15D4F1 1E6
15D4F1 : 00000D 1EE2E9 FFFFFF 000000 000000 000000 000000 000000
15D4F9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D501 : 000000 000000 000000 000000 000000 000000 000000 000000
15D509 : 000000 000000 000000 000000 000000 000000 000000 000000
```

```
15D511 : 000000 000000 000000 000000 000000 000000 000000 000000
15D519 : 000000 000000 000000 000000 000000 000000 000000 000000
15D521 : 000000 000000 000000 000000 000000 000000 000000 000000
15D529 : 000000 000000 000000 000000 000000 000000 000000 000000
15D531 : 000000 000000 000000 000000 000000 000000 000000 000000
15D539 : 000000 000000 000000 000000 000000 000000 000000 000000
15D541 : 000000 000000 000000 000000 000000 000000 000000 000000
15D549 : 000000 000000 000000 000000 000000 000000 000000 000000
15D551 : 000000 000000 000000 000000 000000 000000 000000 000000
15D559 : 000000 000000 000000 000000 000000 000000 000000 000000
15D561 : 000000 000000 000000 000000 000000 000000 000000 000000
15D569 : 000000 000000 000000 000000 000000 000000 000000 000000
15D571 : 000000 000000 000000 000000 000000 000000 000000 000000
15D579 : 000000 000000 000000 000000 000000 000000 000000 000000
15D581 : 000000 000000 000000 000000 000000 000000 000000 000000
15D589 : 000000 000000 000000 000000 000000 000000 000000 000000
15D591 : 000000 000000 000000 000000 000000 000000 000000 000000
15D599 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5A1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5A9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5B1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5B9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5C1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5C9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5D1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5D9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5E1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5E9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5F1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D5F9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D601 : 000000 000000 000000 000000 000000 000000 000000 000000
15D609 : 000000 000000 000000 000000 000000 000000 000000 000000
15D611 : 000000 000000 000000 000000 000000 000000 000000 000000
15D619 : 000000 000000 000000 000000 000000 000000 000000 000000
15D621 : 000000 000000 000000 000000 000000 000000 000000 000000
15D629 : 000000 000000 000000 000000 000000 000000 000000 000000
15D631 : 000000 000000 000000 000000 000000 000000 000000 000000
15D639 : 000000 000000 000000 000000 000000 000000 000000 000000
15D641 : 000000 000000 000000 000000 000000 000000 000000 000000
15D649 : 000000 000000 000000 000000 000000 000000 000000 000000
15D651 : 000000 000000 000000 000000 000000 000000 000000 000000
15D659 : 000000 000000 000000 000000 000000 000000 000000 000000
15D661 : 000000 000000 000000 000000 000000 000000 000000 000000
```

```
15D669 : 000000 000000 000000 000000 000000 000000 000000 000000
15D671 : 000000 000000 000000 000000 000000 000000 000000 000000
15D679 : 000000 000000 000000 000000 000000 000000 000000 000000
15D681 : 000000 000000 000000 000000 000000 000000 000000 000000
15D689 : 000000 000000 000000 000000 000000 000000 000000 000000
15D691 : 000000 000000 000000 000000 000000 000000 000000 000000
15D699 : 000000 000000 000000 000000 000000 000000 000000 000000
15D6A1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D6A9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D6B1 : 000000 000000 000000 000000 000000 000000 000000 000000
15D6B9 : 000000 000000 000000 000000 000000 000000 000000 000000
15D6C1 : 000000 000000 000000 000000 000000 000400 000400 005D0A
15D6C9 : 00018D 00D4CE 00CBB2 00B6B1 00CAC3 00B3B9 00D8D8 1A305D
15D6D1 : 000000 1A30A5 1A3139 15D086 000020 000000
```

```
#W 15D6C9
15D6C9 : 00018D /0  <space>
15D6CA : 00D4CE /0 <space>
15D6CB : 00CBB2 /0 <space>
15D6CC : 00B6B1 /0 <space>
15D6CD : 00CAC3 /0 <space>
15D6CE : 00B3B9 /0 <space>
15D6CF : 00D8D8 /0 <space>
15D6D0 : 1A305D /0 <space>
15D6D1 : 000000 /0 <space>
15D6D2 : 1A30A5 /0 <space>
15D6D3 : 1A3139 /0 <space>
15D6D4 : 15D086 /0 <space>
15D6D5 : 000020 /
```

solution of example 2

2) IN DEBUG DO TNT <TN>

example 2

```
#TNT 4 0 0 0
 EQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 1A31E7 1EE1E4
#P 1A31E7 5
1A31E7 : 008024 000002 006000 000000 000005 000000 000000 000000
```

FAKE_TN_FLAG (0,15,1),

NEED TO RESET THE FAKE_TN_FLAG TO (0)

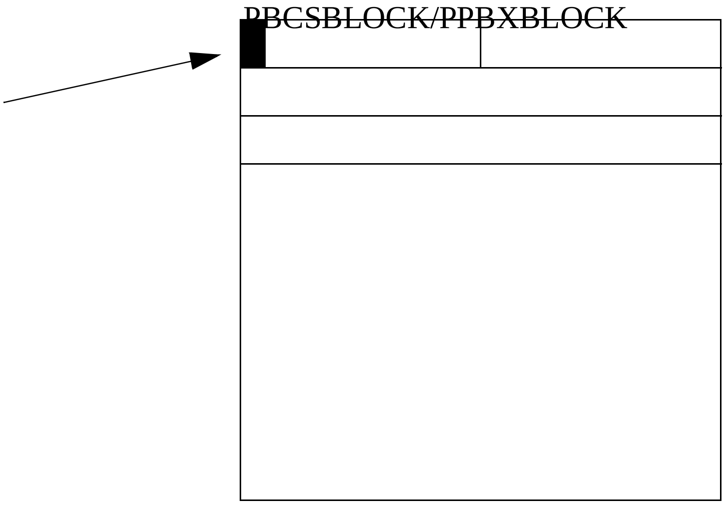
#W 1A31E7

1A31E7 : 008024 /24

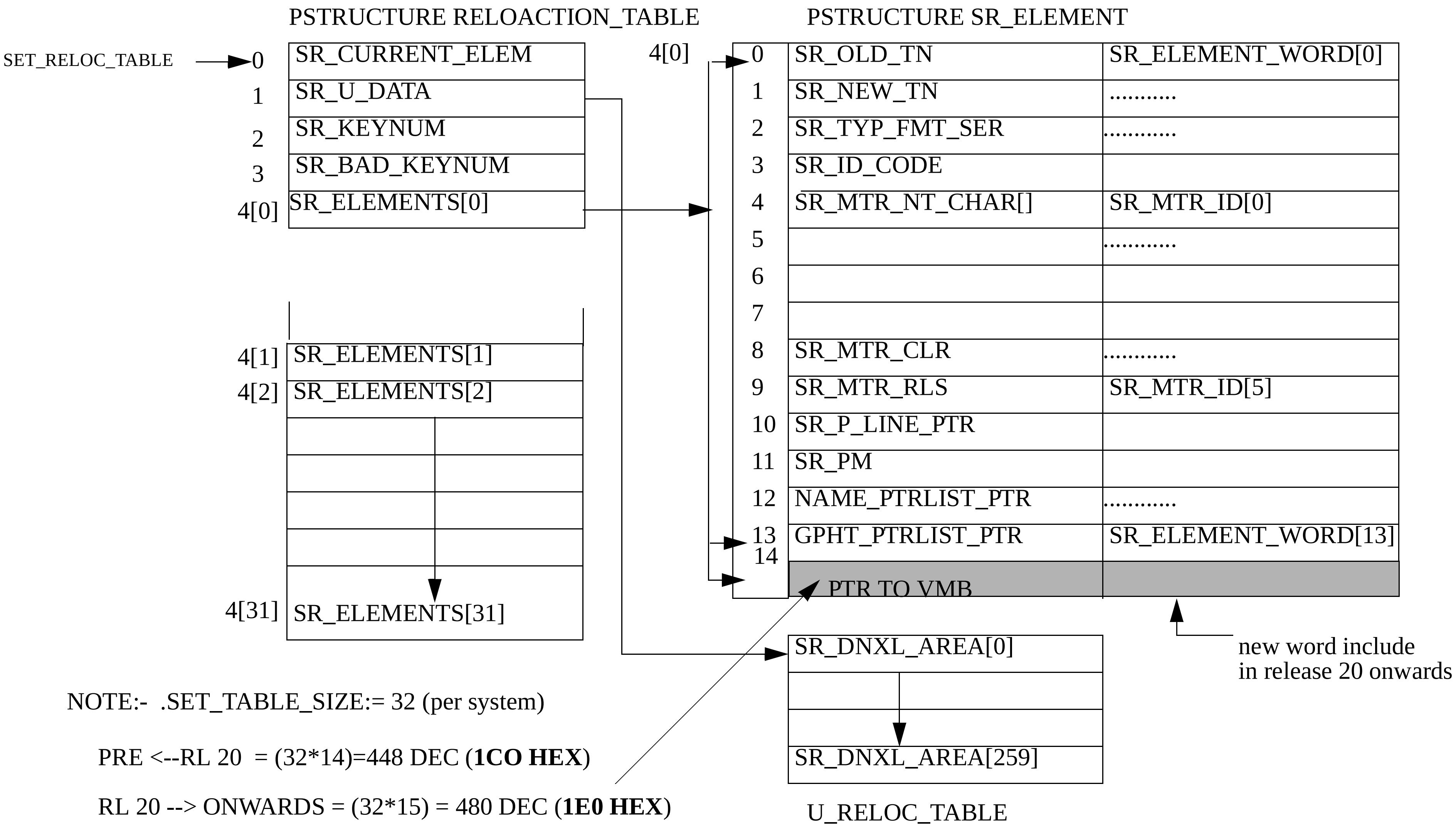
NOW SHOULD BE ABLE TO DO CHANGE’S TO THAT TN.

STRUCTURE OF PROTECTED LINE BLOCK OF TN

FAKE_TN_FLAG (0,15,1)



Set Relocation Structure



PSTRUCTURE RELOACTION_TABLE

0	SR_CURRENT_ELEM
1	SR_U_DATA
2	SR_KEYNUM
3	SR_BAD_KEYNUM
4[0]	SR_ELEMENTS[0]
4[1]	SR_ELEMENTS[1]
4[2]	SR_ELEMENTS[2]
4[31]	SR_ELEMENTS[31]

PSTRUCTURE SR_ELEMENT

0	SR_OLD_TN	SR_ELEMENT_WORD[0]
1	SR_NEW_TN	....
2	SR_TYP_FMT_SER	....
3	SR_ID_CODE	
4	SR_MTR_NT_CHAR[]	SR_MTR_ID[0]
5		....
6		
7		
8	SR_MTR_CLR	....
9	SR_MTR_RLS	SR_MTR_ID[5]
10	SR_P_LINE_PTR	
11	SR_PM	
12	NAME_PTRLIST_PTR	....
13	GPHT_PTRLIST_PTR	SR_ELEMENT_WORD[13]
14	PTR TO VMB	

new word include
in release 20 onwards

SR_DNXL_AREA[0]
SR_DNXL_AREA[259]

U_RELOC_TABLE

NOTE:- .SET_TABLE_SIZE:= 32 (per system)

PRE <--RL 20 = (32*14)=448 DEC (**1CO HEX**)

RL 20 --> ONWARDS = (32*15) = 480 DEC (**1E0 HEX**)

9 ACD Corruption

9.1 SYMPTOMS OF PROBLEM

- 1) LD 20 ,PRT DNB

example

```
DN   6000
TYPE ACDN
ACID 7001  TN 004 0 00 00
ACID ----------corruption
```

ACDN 6000 HAS CORRUPTION

This should look like

```
DN   6000
TYPE ACDN
SUPY ACID 7000  TN 004 0 00 01  KEY 04
ACID 7001  TN 004 0 00 00 <-----------------acd agent
ACID 7000  TN 004 0 00 01 <------------------acd supervisor
```

- 2) LD 11

```
REQ: OUT
TYPE: 2616
TN   4 0 0 0
SCH0767
```

SCH0767 = Supervisor's AGT key must be removed before removing agent.

- 3) In debug/pdt TNT <TN>

  NO UNIT PTR EXIST

example

```
#TNT 4 0 0 1
 UNEQPD SLOOP TN 000401
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 000000 000000
```

9.2 SOFTWARE STRUCTURE

- see ACD Structure diagram

9.3 HOW TO RESOLVE PROBLEM

The best way to deal with this type coruption is to remove **ACD AGNT** and **SPV** associated **ACDN**, in debug/pdt, then remove **ACDN** in LD 23, then rebuild everything again.

•

1) First print all **AGNT** and **SPV** in the **ACD** Queue , remove all **SCR, MCR DN'S** to avoid any **DN** corruption.

LD 11, NUL OUT ALL SCR KEY

example

```
TN   004 0 00 00
TYPE 2616
..
.
KEY  00 ACD 6000 7001
       SUPY ACID  KEY 05
    01 NRD
    02 MSB
    03 SCR 2001    MARP
```

```
REQ: CHG
TYPE: 2616
TN   4 0 0 0
ECHG YES
ITEM KEY 03 NUL
```

2) In debug take out ACD **AGT_ID** FOR BOTH **AGT** & **SPV** in the **DNTREE**
see DN Corruption

example

```
#DNT 0 7000
DIG 4 ACD_ID
1564F7 :   008B02 000401
```

```
#DNT 0 7001
DIG 4 ACD_ID
1564F9 :   008B02 000400
```

•

NEED TO REMOVE ACD_ID IN THE DNTREE (SEE DN CORRUPTION)

example

CDNXPTR FOR 2119 111 = 8922

```
#P 8922 3

008922 : 15CF0E 000000 000000

#P 15CF0E 10
15CF0E : 0001C6 15CF80 15CF1B 000000 000000 000000 15D062 15CFC2
15CF16 : 15CFC0 000000 000000 000000 000000 000524 000000 15CFA5

#P 15CFC2 10
15CFC2 : 000400 000000 000000 000000 000000 000000 000000 000000
15CFCA : 000000 000000 15CFCE 000000 000480 000000 000000 000000

#P 15CFCE 10
15CFCE : 000480 000000 000000 000000 000000 000000 000000 15CFDA
15CFD6 : 000000 000000 15655C 000000 00000C 000000 15CFE6 15CFF4



#P 15655C 10
15655C : 000402 1564F9 000000 000000 000000 000000 000000 000000
156564 : 000000 000000 1564F7 000000 008208 000000 000000 000000
#
#W 15655C
15655C : 000402 /0
15655D : 1564F9 /0
15655E : 000000 /
15655F : 000000 /
156560 : 000000 /
156561 : 000000 /
156562 : 000000 /
156563 : 000000 /
156564 : 000000 /
```

```
156565 : 000000 /
156566 : 1564F7 /0

#DNT 0 7000
DIG 4 INV

#DNT 0 7001
DIG 4 INV
```

•

3) RemoveACD AGNT and SPV (TN’s)in debug/pdt

see TN Corruption for further information.

```
#TNT 4 0 0 0
 EQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 1A30E5 1EE0E0

#
#W 158728
158728 : 002060 /
158729 : 1A30E5 /0
#W 1F9266
1F9266 : 000000 /
1F9267 : 1EE0E0 /0

#TNT 4 0 0 0
 UNEQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 000000 000000
```

At this point we have piratically removed all of the TN’S and remove all of the DN’S that belongs to it’s ACDN.

•

4) Now we have to remove all of these TN’S out of the ACD structure in the **ACD_POS_LIST_PTR** in debug/pdt by zeroing them also need to zero out **ACD_NUMB_POS** wordoffset (1) which contain’s the number of stations.

To find the location of the **ACD_POS_LIST** as follows:-

(see ACD structure diagram)

a) In LD 23 print all the ACDN, All ACD DN’S configured in a given cust numbers are in order in will appear in ACD_LIST data structure.

Note:- look for the corrupted ACDN and make a note of the position of order.

b) In debug/pdt find address of CDATAPTR by doing DCP <CUST>

CDATAPTR

```
#DCP 0
CUST 0 P 15658D U 1F9D06 AUX 156ABF ICI 156C52 PREXL 000000 BGD 156ECD
```

P_CUST_DATA_BLK

```
#P 15658D 87
15658D : 004104 000000 000001 000000 000000 000000 000000 000000
156595 : 000000 0061E5 009951 000000 000000 1F9D06 000001 001EE0
15659D : 000003 000000 000000 000002 000000 156C52 15CB86 000100
1565A5 : 000000 000000 000000 00FEA0 000000 000000 000000 000000
1565AD : 000000 000000 000000 000000 000000 000000 000000 000000
1565B5 : 000000 000000 000000 000000 000000 000000 000000 000000
1565BD : 000000 000000 000000 000000 000000 000000 000000 000000
1565C5 : 000000 000000 000000 000000 000000 000000 00001E 000000
1565CD : 000000 000000 000000 000000 000000 000000 000000 000000
1565D5 : 000000 000000 000000 000000 000000 000000 000000 000000
1565DD : 000000 000000 000000 000000 000000 000000 000000 000000
1565E5 : 000000 000000 000000 000000 000000 000000 000000 000000
1565ED : 000000 000000 000000 000000 000000 000000 000000 000000
1565F5 : 000000 000000 000000 006870 002871 000041 000000 000F00
1565FD : 00080F 002014 000000 000000 000000 000000 000000 000000
156605 : 000000 000441 000441 000000 000000 00085A 000111 000000
15660D : 00AAAA 00FF20 00FFFF 000000 000001 00201E 1564EF
```

c) The 2nd pointer in our example is obtained at stage(a) by printing in LD 23 for all ACDN and the second configured ACDN is, in for the example is the corrupted ACDN 6000

ACD_LIST

```
#P 1564EF 10       1st      2nd
1564EF : 000003 15BF3E 1A3075 008A03 000000 000001 008B02 000400
1564F7 : 008B02 000401 008B02 000400 000000 00000C 000000 1A30C9
```

d)

P_ACD_BLOCK wordoffset (4)

```
P 1A3075 8
1A3075 : 00003A 00AAA6 000000 1EE16E 1A30AF 00000A 0003FF
```

Note:- AAA6 000000 = ACDN (6000) of 2 words

e)

ACD_POS_LIST

```
#P 1A30AF 20
1A30AF : 00001A 000002 000401 000000 000000 000000 000000 000000
1A30B7 : 000000 000000 000004 000000 000000 000000 000000 000000
1A30BF : 000401 000400 000000 000000 000000 000000 000000 000000
1A30C7 : 000000 000000 00011C 00BF63 000001 001AF6 001F04 002E20
```

TO CHECK FOR TN'S INVOLVED DO TNT <TN>

```
#TNT 4 0 0 0
 EQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 000000 000000
#TNT 4 0 0 1
 UNEQPD SLOOP TN 000401
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 000000 000000

#W 1A30AF

1A30AF : 00001A /  <space>
1A30B0 : 000002 /0 <space>
1A30B1 : 000401 /0 <space>
1A30B2 : 000000 / <space>
1A30B3 : 000000 / <space>
1A30B4 : 000000 / <space>
1A30B5 : 000000 / <space>
1A30B6 : 000000 / <space>
1A30B7 : 000000 / <space>
1A30B8 : 000000 / <space>
1A30B9 : 000004 / <space>
1A30BA : 000000 / <space>
1A30BB : 000000 / <space>
```

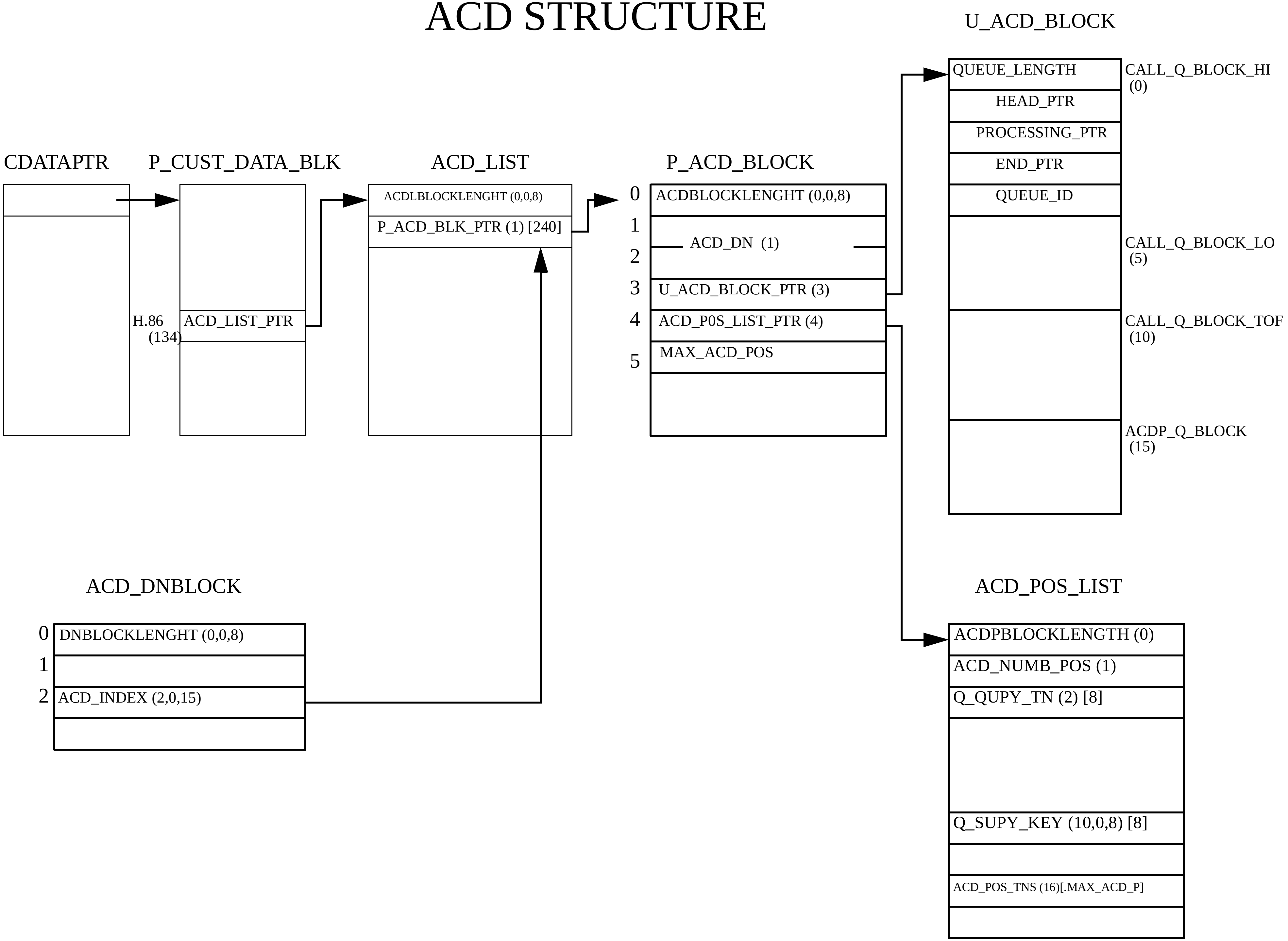
```
1A30BC : 000000 / <space>
1A30BD : 000000 / <space>
1A30BE : 000000 / <space>
1A30BF : 000401 /0 <space>
```

1A30C0 : 000400 /0

We have now taken out acd agent , acd supervisor and acddn corruption has been cleared, need to rebuild data base again.

```
PSTRUCTURE [.P_ACD_POS_LIST] ACD_POS_LIST

 INTEGER ACDPBLOCKLENGTH (0),
 ACD_NUMB_POS(1),  % THE CURRENT NUMBER OF STATIONS
 INTEGER Q_SUPY_TN (2)[8],
 Q_SUPY_KEY(10,0,8)[8],
 ACD_NSVC_SUPYTN(14),  % tn of supervisor w. nsvc key
 ACD_NSVC_KEY (15,0,6),% ssd value of nsvc key number
 ACD_POS_TNS(16)[.MAX_ACD_P];
```

ACD STRUCTURE



CDATAPTR

P_CUST_DATA_BLK

H.86 (134) ACD_LIST_PTR

ACD_LIST

ACDLBLOCKLENGHT (0,0,8)

P_ACD_BLK_PTR (1) [240]

P_ACD_BLOCK

0 ACDBLOCKLENGHT (0,0,8)

1

2 ACD_DN (1)

3 U_ACD_BLOCK_PTR (3)

4 ACD_P0S_LIST_PTR (4)

5 MAX_ACD_POS

U_ACD_BLOCK

QUEUE_LENGTH — CALL_Q_BLOCK_HI (0)

HEAD_PTR

PROCESSING_PTR

END_PTR

QUEUE_ID

CALL_Q_BLOCK_LO (5)

CALL_Q_BLOCK_TOF (10)

ACDP_Q_BLOCK (15)

ACD_DNBLOCK

0 DNBLOCKLENGHT (0,0,8)

1

2 ACD_INDEX (2,0,15)

ACD_POS_LIST

ACDPBLOCKLENGTH (0)

ACD_NUMB_POS (1)

Q_QUPY_TN (2) [8]

Q_SUPY_KEY (10,0,8) [8]

ACD_POS_TNS (16)[.MAX_ACD_P]

10 TTY Corruption

10.1 SYMPTOMS OF PROBLEM

1) PRINT ALL TTY LD 22 , IF **CTYP ****** AND DNUM DOES NOT MAKE SENSE THEN THAT TTY IS CORRUPTED

example

```
LD 22
REQ  PRT
TYPE ADAN TTY

ADAN     TTY 0
  CTYP SDI2
  DNUM 0
  DES
  USER MTC TRF SCH CTY BUG BGD
  CUST 00
  XSM  NO
  TTYLOG       0
ADAN     TTY 1
  CTYP ****
  DNUM 8
  DES
  USER
  XSM  NO
  STA  0
```

2) IN THIS EXAMPLE TTY 1 IS CORRUPTED AND CAN NOT BE TAKEN OUT IN LD 17, YOU WILL GET SCH6083

```
REQ  CHG
TYPE ADAN
ADAN OUT TTY 1

SCH6083
ADAN
```

SCH6083 = Since this TTY is configured with an STA application, it cannot be removed until the STA is removed.

10.2 SOFTWARE STRUCTURE

- see TTY Structure diagram.

10.3 HOW TO RESOLVE PROBLEM

SOLUTION - IN DEBUG/PDT

•

1) FIND **LOG_IO_PTR** = 9DD0 (2119 111)

```
#P 9DD0
009DD0 : 12478A
#P 12478A 10
12478A : 000005 124BE9 1247DC 12478F 000000 000011 1247A0 1247BE
124792 : 000000 000000 000000 000000 000000 000000 000000 000000
```

This is PPOINTER P_SDI_BLK_PTR(1) [.max_num_of_ttys] from the LOG_IO_PTR above

```
#P 12478F 10
                 TTY0  TTY1    TTY2----------etc
12478F : 000011 1247A0 1247BE 000000 000000 000000 000000 000000
124797 : 000000 000000 000000 000000 000000 000000 000000 000000
```

```
#P 1247BE 10
1247BE : 008008  0000A6 003005 000000 000502 FFFFFF 000000
1247C6 : 008108 C03090 0000A6 003009 000000 000902 FFFFFF 000000
#W 12478F
12478F : 000011 /  <space>
124790 : 1247A0 / <space>
124791 : 1247BE /0  zero out the pointer of the corrupted TTY 1
```

•

2) Find address for

= 8FE1{2119 111)

CONFIGTTYOP [16] NOTE:- each word is overlaid by the structure

LOGUWORD

```
#P 8FE1 5 TTY0   TTY1   TTY2   TTY3  TTY4   TTY5 ---etc
008FE1 : 00805E 000000 000000 000000 000000 000000 000000 000000
```

NOTE_:- TTY 1 IS ALREADY '0'

•

3)

TTYLOGU (0,4) [16] = 881A

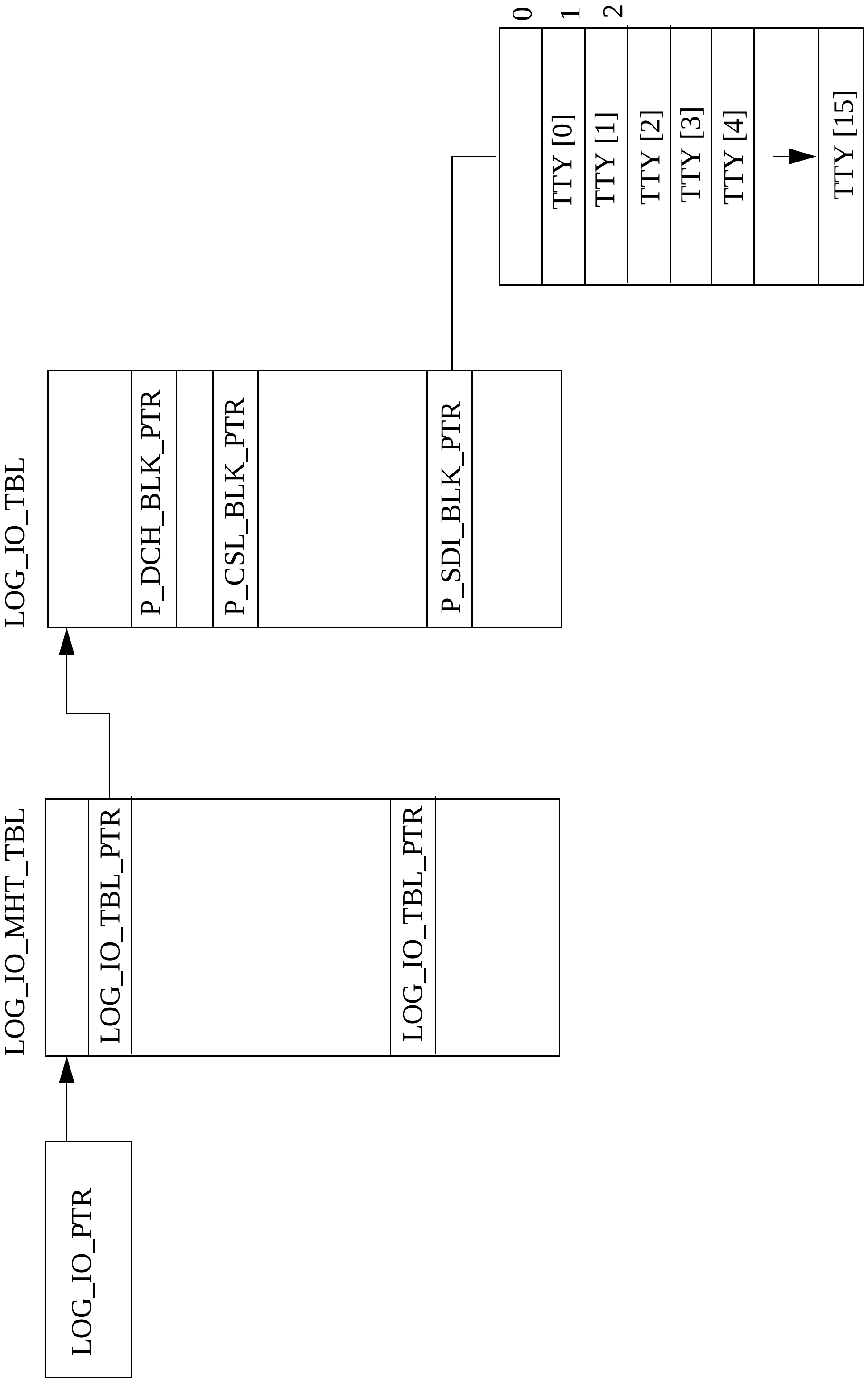
```
#P 881A 4
00881A : 000004 000000 000000 00000
```

TO CONFIRM CORRUPTION HAS CLEARED PRINT TTY 1 IN LD 22, THEN

REBUILD IN LD 17.

```
REQ  PRT
TYPE ADAN TTY 1

TTY 1  IS UNDEFINED

ADAN NEW TTY 1
CTYP SDI2
DNUM 01
DES
USER MTC BUG
TTYLOG
ADAN DATA SAVED
```

TTY Structure



11 BFS Corruption

11.1 SYMPTOMS OF PROBLEM

•

```
LD 20 , PRT
TN   4 0 0 0
DATE
PAGE
DES

DES  TEST
TN   004 0 00 00
TYPE 2616
CDEN 8D
CUST 0
.. ..
.. ..

   UDI RCC HBTD AHD IPND DDGA NAMA MIND PRSD NRWD NRCD NROD
   EXR0
CPND_LANG ENG
BFTN 000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
                                   BFS CORRUPTION
   008 0 00 01

    ..   ..   ..
    ..  ..    ..

   000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
   000 0 00 00
```

- IN DEBUG/ PDT TNT <TN>

```
  #TNT 4 0 0 0
 EQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 1A305D 1EE1E4
#P 1A305D 20
1A305D : 000024 000002 007000 000000 000005 000000 000000 000000
1A3065 : 000000 005862 000000 00000E 0000C0 000004 000000 001FFF
1A306D : 000000 001000 000000 000001 000000 000000 000000 001100
1A3075 : 156508 000001 0000C0 000000 000000 000000 000000 00AAA2


#P 005862 10
005862 : 000000 000000 000000 000000 000000 000000 000000 000000
00586A : 000000 000000 000000 000000 000000 000000 000000 000000
```

11.2 SOFTWARE STRUCTURE

- see BFS Structure diagram.

11.3 HOW TO RESOLVE PROBLEM

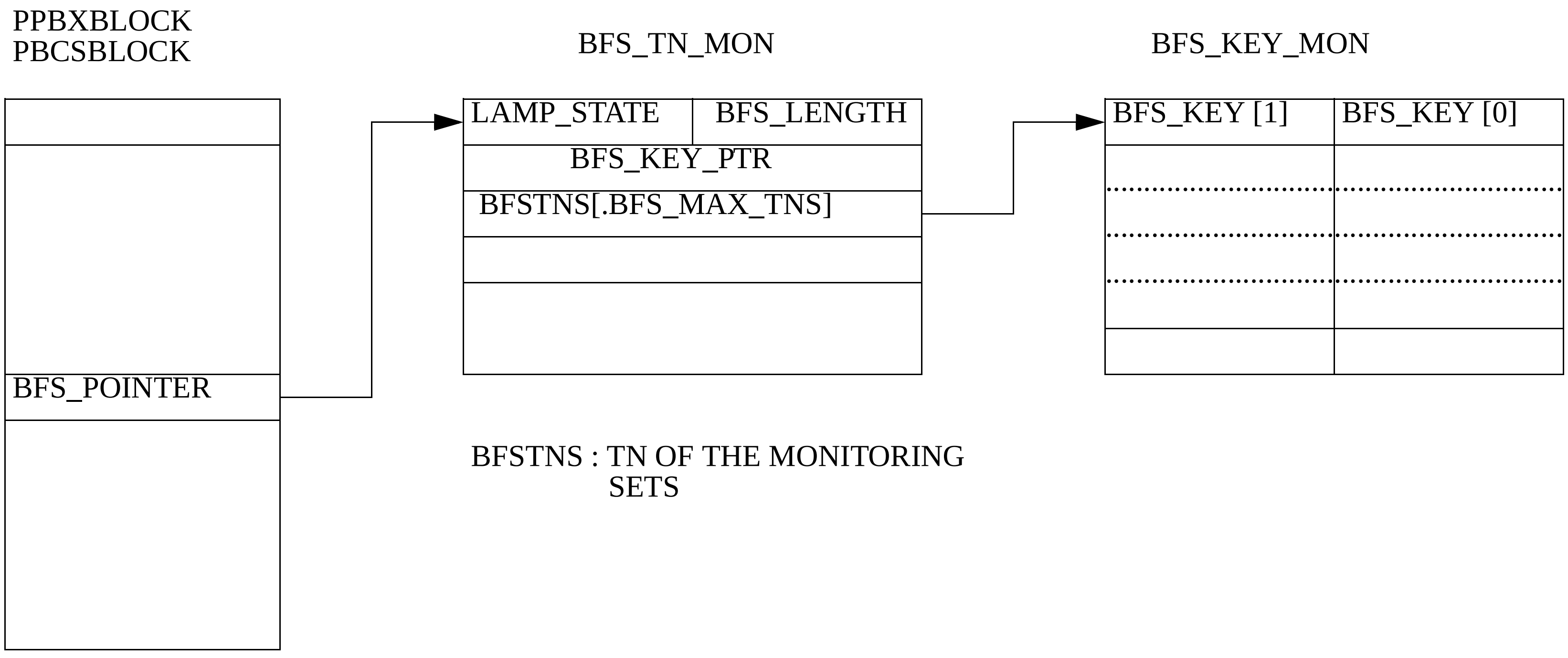
-

IN DEBUG/PDT TAKE OUT **BFS_POINTER** IN THE PBCSBLOCK OR PPBXBLOCK (500/2500) set

```
#TNT 4 0 0 0
 EQPD SLOOP TN 000400
 GP 1586B4 SLP 1586DA 1F9280  CD 158728 1F9266  LN 1A305D 1EE1E4


#P 1A305D 10
1A305D : 000024 000002 007000 000000 000005 000000 000000 000000
1A3065 : 000000 005862 000000 00000E 0000C0 000004 000000 001FFF


#W 1A305D
1A305D : 000024 /
1A305E : 000002 /
1A305F : 007000 /
1A3060 : 000000 /
1A3061 : 000005 /
```

```
1A3062 : 000000 /
1A3063 : 000000 /
1A3064 : 000000 /
1A3065 : 000000 /
1A3066 : 005862/ 0
```

BFS STRUCTURE



BFSTNS : TN OF THE MONITORING SETS

BLOCK OF THE SET TO BE MONITORED

12 Authorisation Code Corruption

12.1 SYMPTOMS OF PROBLEM

If an Authorisation Code corruption has occurred then there may be a number of problem seen, the problems seen so far are:-

1) If Authorisation code is printed in LD 88, The system prints from say 021175 to317490, it then skip back to 131490 and prints to 317490 and remains in a loop, infinitely.

2) Corruption in Authorization code table, Auth codes are lost, when re-inputted and cause INI.

12.1.1 SOFTWARE STRUCTURE

(see Auth Code Structure Diagram)

12.2 HOW TO RESOLVE PROBLEM

The best way to deal with this type corruption is to take out the **P_AUTH_TBL_PTR** wordoffset (157) in the protected cust data block **(P_CUST_DATA_BLK)**. This would remove Authorisation Code Table in the given cust number. Would need to re-enter all the code's again.

In debug/pdt DCP <CUST>

```
#DCP 0

CUST 0 P 11DC59 U 1F9A8E AUX 11E4A7 ICI 11E701 PREXL 1242BB
BGD 11E9F5
```

```
#P 11DC59 9E
11DC59 : 007304 000000 000001 000004 000000 000000 000000 000000
11DC61 : 000000 0061E5 000021 000000 000000 1F9A8E 000001 001EE0
11DC69 : 000003 000000 000000 000002 000000 11E701 125401 000100
11DC71 : 000000 000000 000000 00FEE0 000000 000000 000000 000000
11DC79 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC81 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC89 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC91 : 000000 000000 000000 000000 000000 000000 00001E 000000
11DC99 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCA1 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCA9 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCB1 : 000000 000000 000000 000000 000000 000000 000000 000000
```

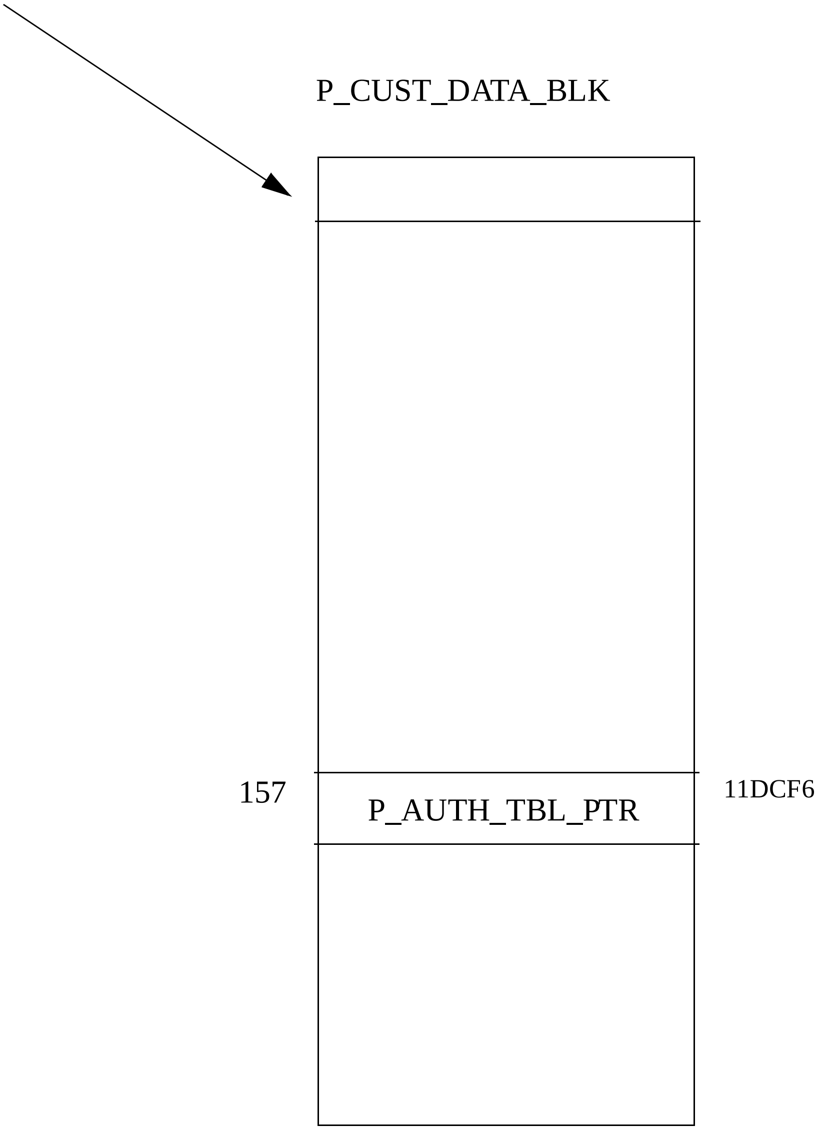
```
11DCB9 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCC1 : 000000 000000 000000 006870 006871 000041 00003F 000A00
11DCC9 : 001C0A 00E018 000000 000000 000000 000000 000000 000000
11DCD1 : 000000 000444 000444 000000 000200 00085A 000111 000000
11DCD9 : 00AAAA 00FF20 00FFFF 000000 000003 00211E 1243BB 000000
11DCE1 : 12292A 000000 00AA82 000000 000000 000000 000000 000000
11DCE9 : 000000 00C300 000000 000000 000000 000000 000000 000000
11DCF1 : 000000 000000 000000 000000 123CCD 111413
```

```
#W 11DCF1
11DCF1 : 000000 /
11DCF2 : 000000 /
11DCF3 : 000000 /
11DCF4 : 000000 /
11DCF5 : 123CCD /
11DCF6 : 111413/ 0
```

Auth code Structure

# DCP <CUST>

CUST 0 P **11DC59** U 1F9A8E AUX 11E4A7 ICI 11E701 PREXL 1242BB BGD 11E9



13 ESN Corruption

13.1 SYMPTOMS OF PROBLEM

If an ESN corruption has occurred then there may be more then one type of problem seen, one of many problem reported is as follows:-

1) Unable to dial 9-999, LD 20,PRT,DNB,DN 9 = NARS, AC1
In LD 90 PRT,NET,AC1,SPN 9 is not configured. When tried to configure, NEW, NET,AC1,SPN 9 fails with ESN072 . Also LD 90 PRT SPN 0 shows FLEN = 0, CHG FLEN to 1, PRT again shows FLEN = 0.

13.2 SOFTWARE STRUCTURE

- see ESN Structure diagram

13.3 HOW TO RESOLVE PROBLEM

The best way to deal with this type and many other type’s of corruption is to take out **ESN_DATA_BLK_PTR** wordoffset (136) in the protected cust data block **(P_CUST_DATA_BLK)**, by resetting this pointer to zero you have basically wiped out the whole of **ESN_DATA_BLOCK**.

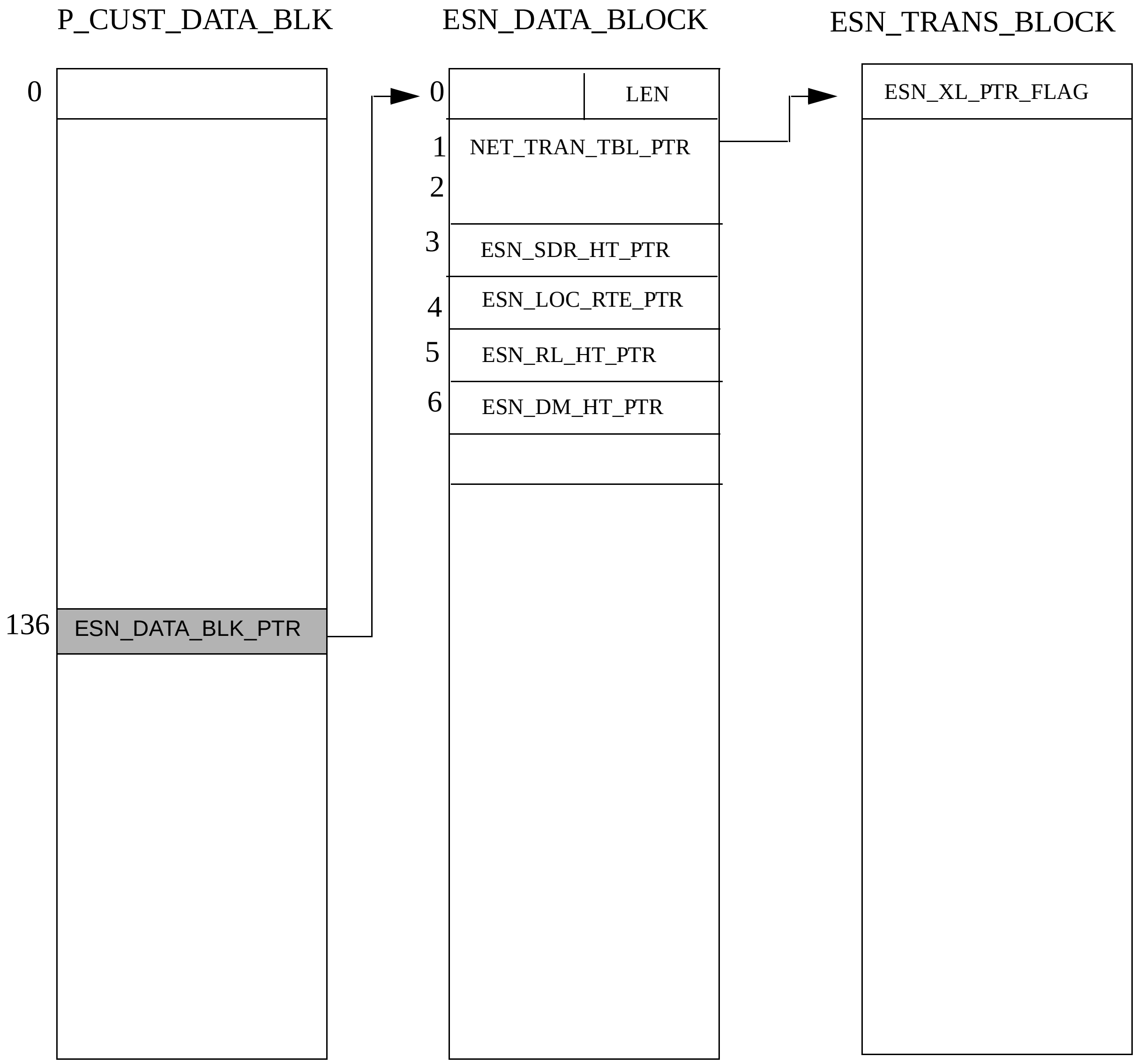
You must then **DATADUMP** and **RE-LOAD** the switch to clean the **DNTREE** structure.

1) Find the **P_CUST_DATA_BLK** pointer in debug/pdt by DCP <CUST>

```
#DCP 0
CUST 0 P 11DC59 U 1F9A8E AUX 11E4A7 ICI 11E701 PREXL 1242BB BGD 11E9F5

#P 11DC59 89  PRINT OFF WORD OFFSET (8+1)HEX  (136DEC)
11DC59 : 007304 000000 000001 000004 000000 000000 000000 000000
11DC61 : 000000 0061E5 000021 000000 000000 1F9A8E 000001 001EE0
11DC69 : 000003 000000 000000 000002 000000 11E701 125401 000100
11DC71 : 000000 000000 000000 00FEE0 000000 000000 000000 000000
11DC79 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC81 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC89 : 000000 000000 000000 000000 000000 000000 000000 000000
11DC91 : 000000 000000 000000 000000 000000 000000 00001E 000000
11DC99 : 000000 000000 000000 000000 000000 000000 000000 000000
```

```
11DCA1 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCA9 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCB1 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCB9 : 000000 000000 000000 000000 000000 000000 000000 000000
11DCC1 : 000000 000000 000000 006870 006871 000041 00003F 000A00
11DCC9 : 001C0A 00E018 000000 000000 000000 000000 000000 000000
11DCD1 : 000000 000444 000444 000000 000200 00085A 000111 000000
11DCD9 : 00AAAA 00FF20 00FFFF 000000 000003 00211E 1243BB 000000
11DCE1 : 12292A
#W 11DCE1
11DCE1 : 12292A /0
```

ESN Structure



14 Patch Corruption (On Non-Thor)

14.1 SYMPTOMS OF PROBLEM

If a patch or a number of patches have been corrupted then there are more then one symptoms seen. The most common type is when the patch can not be taken out get EHM0017.

example

-

In patch debug STT <pat_num>

```
#STT 0
PAT# 0
ID   BV9999/01812119/MHD 0000
NAME DPNSS
ENGR NM
STAT NEW
SAVE YES

#OUT 0
NAME DNPSS
```

EHM0017 = Incorrect name or number entered, CMD is out.

```
NAME
```

14.2 SOFTWARE STRUCTURE

- see Patch Structure diagram.

14.3 HOW TO RESOLVE PROBLEM

The best way to resolve this type or many other’s is to remove the **PATCH_PTR** from **PATCH_HT** for the corrupted patch number. This will remove the patch from the switch.

1) Find address for PATCH_HTPTR

In our example PATCH_HTPTR ={9CC5 for 2119 111}

2) Print this address in debug

example

```
  #P 9CC5
009CC5 : 11DB6E
```

3) Print this pointer of for say 10

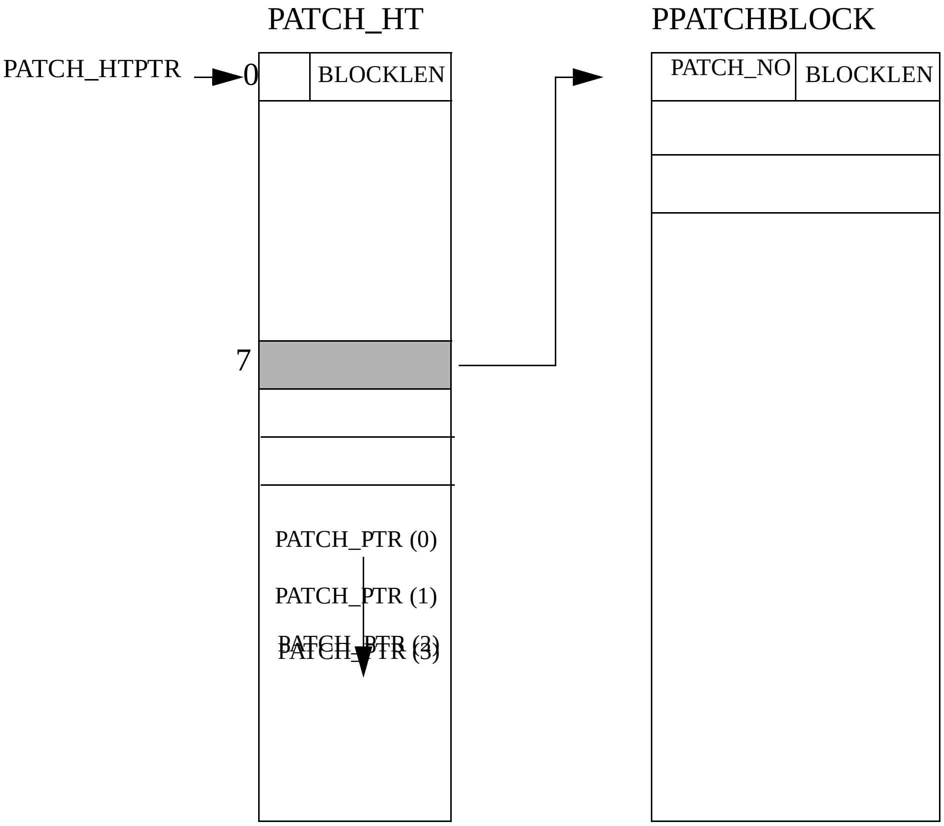
```
#P 11DB6E 10
11DB6E : 00000A 000000 000000 000000 000000 000000 000000  11D5A5
11DB76 : 11D590 000000 11D58D 000003 000000 00000D 000000 11DC07
#
```

NOTE:- Wordoffset (7) is the first **PATCH_PTR** ie this pointer is for patch number (0)

4) Clearing **PATCH_PTR** for **patch 0**

```
#W 11DB6E
11DB6E : 00000A /
11DB6F : 000000 /
11DB70 : 000000 /
11DB71 : 000000 /
11DB72 : 000000 /
11DB73 : 000000 /
11DB74 : 000000 /
11DB75 : 11D5A5 /0
```

PATCH STRUCTURE



```
 PSTRUCTURE [.P_PATCH_DATA] PATCH_HT
     INTEGERBLOCKLEN  (0,0,8),
OVERLAY_PATCHED  (0,8,1) [.MAX_PATCH_OVL + 1],
     PPOINTER PATCH_PTR  (.OFSET_PATCH_PTR) [.MAX_PATCH_NO + 1];


  PSTRUCTURE [.P_PATCH_DATA] PPATCHBLOCK
     INTEGER BLOCKLEN      (0,0,8),
      PATCH_NO      (0,8,8),
      PRS_NO      (1) [.SIZE_PRTS >> 1],
      PATCH_ID_NO    (5),
      ENGR_NAME      (6) [.SIZE_ENGR_NAME >> 1],
      PATCH_NAME     (8) [.SIZE_NAME >> 1],
      PATCH_VERSION  (12,0,12),
      INS_MONTH      (12,12,4),
      PATCH_ISSUE    (13,0,12),
      DEV_AUTHORITY  (13,12,4),
      IN_DAYS      (14,0,9),
      IN_INITS      (14,9,5),
      PATCH_STATUS   (14,14,2),
      OOS_DATE      (15,0,5),
      OOS_MONTH      (15,5,4),
      SAVE_ON_TAPE   (15,9,1),
      INS_DATE      (15,10,5),
      SPA_BIT      (15,15,1),
      OVERLAY_NUMBER (16,0,7),
      PATCH_LOADED   (16,7,1),
      OVL_OVF_PATCH  (16,8,1),
*=>----------------- INFOR FLASH_ROM
*      INSERT_ORDER   (16,9,7), %order patch is to be inserted
*=>----------------- INFOR ALL
      PATCHWORD      (17) [MAXINT(BLOCKLEN) -
  WORDOFFSET(PATCHWORD)];
```

15 ACD SCHED Block Corruption

15.1 SYMPTOMS OF PROBLEM

1) LD 23

```
REQ  NEW OR CHG
TYPE SCB
CUST 0
SCH0925
CUST****
```

SCH0925 = New request is invalid for ADS prompt if the customer has been assigned as an ACD package D customer.

NOTE:- THE ACD D PACKAGE DOES NOT EXSIT.

15.2 SOFTWARE STRUCTURE

- see structue diagram

15.3 HOW TO RESOLVE PROBLEM

```
PSTRUCTURE[.P_CUST_DATA] P_CUST_DATA_BLK
INTEGER CUSTBLOCKLENGTH (0,0,9),

 ACD_D_ON (142,0,1), % Cust has ACD pkg D
 AGENT_ID_MODE   (142,1,1), % set if agent ID opt is used
```

To clear this type of corruption we need to reset ACD_D_ON and AGENT_ID_MODE VARIABLES in DEBUG/PDT

-

1) find the P_CUST_DATA_BLK by doing

```
#DCP 0
CUST 0 P 11DC59 U 1F9A8E AUX 11E4A7 ICI 11E701 PREXL 1242BB BGD 11E9F5
```

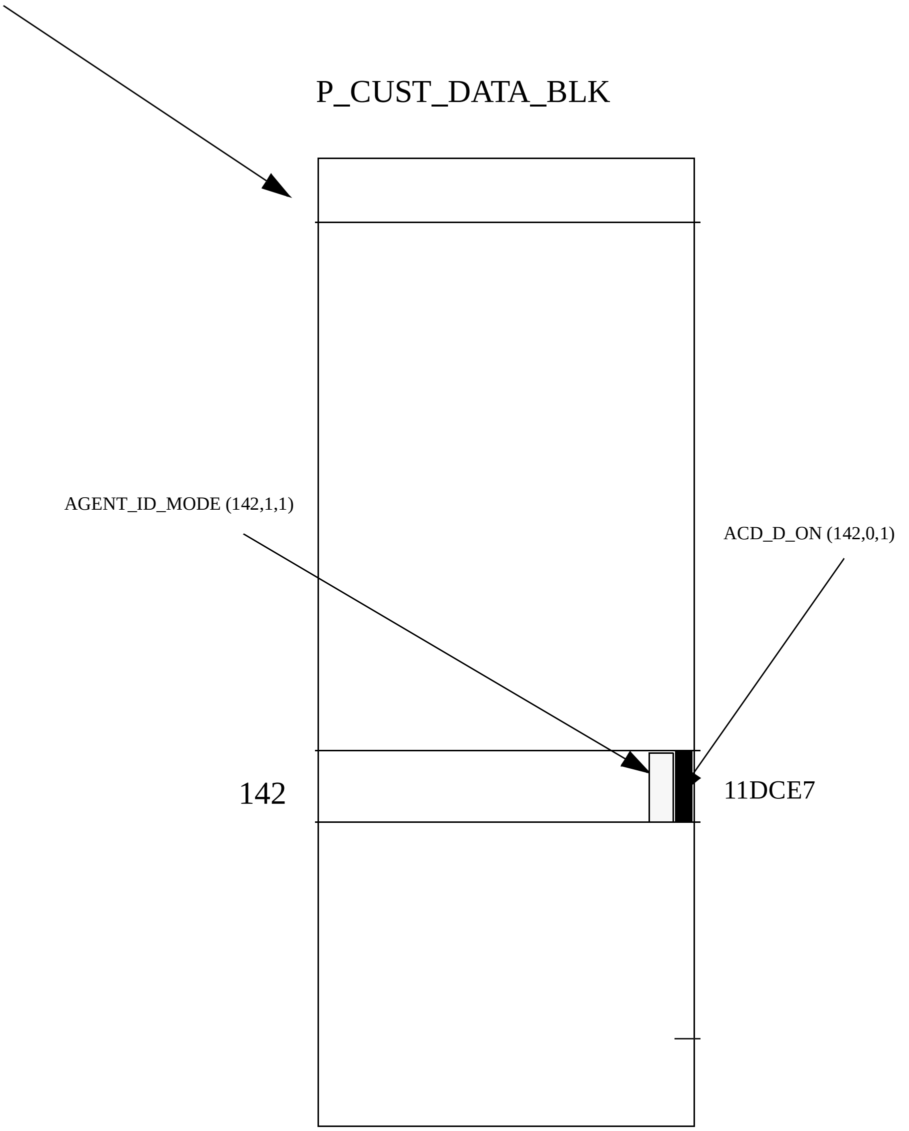
- 142 = 8E hex
- To locate ACD_D_ON & AGENT_ID_MODE

11DC59 + 8E = 11DCE7

2)

```
P 11DCE7
11DCE7 : 000003
W 11DCE7 : 000003 /0
```

```
# DCP <CUST>
CUST 0 P 11DC59 U 1F9A8E AUX 11E4A7 ICI 11E701 PREXL 1242BB BGD 11E9F
```



16 MSDL PORT CORRUPTION

16.1 SYMPTOMS OF PROBLEM

This is when you are unable to use a port
on the msdl when you know that it is available
get **SCH5573** (Specified port not available, other ports are available.)

In our example we have a problem in trying to config an MSDL
on DCH 9 on port 3, which should be available.

1) Print out the i/o device (ADAN) data in LD 22 and
make a note of all the MSDL ports being used.
double check to make sure that the port is corrupted
and not being used as an AML instead of a DCH.

LD 48

.stat msdl 14

```
MSDL 14: ENBL

  DCH    10  OPER  PORT 1
  DCH    11  OPER  PORT 2

ADAN     DCH 10
 CTYP MSDL
 DNUM 14
 PORT 1
 DES  MCDN
 USR  PRI
 DCHL 10
 ..
 ..

ADAN     DCH 11
 CTYP MSDL
 DNUM 14
 PORT 2
 DES  MCDN
 USR  PRI
 DCHL 11
 OTBF 32
 ..
 ..
LD 17

REQ  CHG
TYPE ADAN
ADAN NEW DCH 9
CTYP MSDL
DNUM 14
PORT 3
```

SCH5573 -----Specified port not available,
other ports are available.
DNUM

16.1 SOFTWARE STRUCTURE

see MSDL Structure diagram

16.2 HOW TO RESOLVE THIS PROBLE

SOLUTION
============

1) Print out the **P_MSDLMISP_MHPTR** and the **P_MSDLMISP_TABLE**

```
 P_MSDLMISP_MHPTR (21.19) = (0x27f3c/4) =  9fcf

pdt> p 9fcf


00009FCF : 00022C22
pdt> p 00022C22 20
```

P_MSDLMISP_TABLE

```
00022C22 : 00022BC1 00022CA2 00022CD4 00000000 00000000 00000000 00000000 00000000
00022C2A : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00022C32 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00022C3A : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

2) We must find out which **P_MSDLMISP_BLOCK** is the one we're looking for
the best way to be sure of this (unless there's only one!) is to first
print out all of the blocks. Then using the **PHY_IO_BLK** pointer (**word 0**)
of each block, print out the **PHY_SERIAL_IOBLK** for each MSDL.

```
pdt> 00022BC1 10
```

PHY_IO_BLK

```
pdt> p 00022BC1 10


00022BC1 : 0001EDD8 00728D17 00022D06 00728C67 00728B5D 0072894D 00000000 00000000
```

```
00022BC9 : 00000000 00000000 00000404 00000004 00000000 00000000 00000000 00000029
```

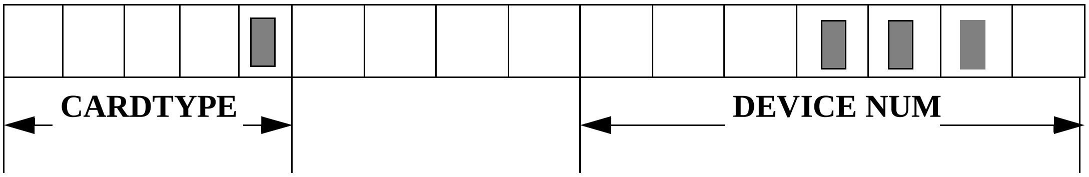
PHY_SERIAL_IOBLK

```
>pdt> p 0001EDD8 30
```

```
0001EDD8 : 00008311 000030E0 000000A7 0000380E 00000000 00000000 00000000 00000000
0001EDE0 : 00000A02 FFFFFFFF 00000000 00000B02 FFFFFFFF 00000000 00000802 FFFFFFFF
0001EDE8 : 00000000 00008C04 00E00F01 00000000 00008000 00008108 00003090 000000A6
0001EDF0 : 00003009 00000000 00000502 FFFFFFFF 00000000 00008C04 00E00F01 00000000
0001EDF8 : 00008000 0001EDD5 0000000F 0001C81B 00000124 00000002 00006000 00000000
0001EE00 : 00000002 00000000 00000000 00000000 00000000 00000000 00000000 0000000E
```

NOTE:- It so happens that the first in the P_MSDLMISP_TABLE is the one
we are looking for because word 3 tells you in this example that
it is an MSDL CARD and it's configured on device num 14

word 3 = 380E



CARDTYPE = .MSDL_CARD=7

Once the appropriate P_MSDLMISP_BLOCK and associated PHY_SERIAL_IOBLK have been found we need to clear the corrupted port. This is done by zapping appropriate entries in the IO_PORT_DATA array (starting at word 5), which containe four structure instance of IO_PORT_INFO, one for each port.

```
pdt> w 0001EDE6
```

```
0001EDE6 : 00000802 /0
0001EDE7 : FFFFFFFF /0
0001EDE8 : 00000000 /
```

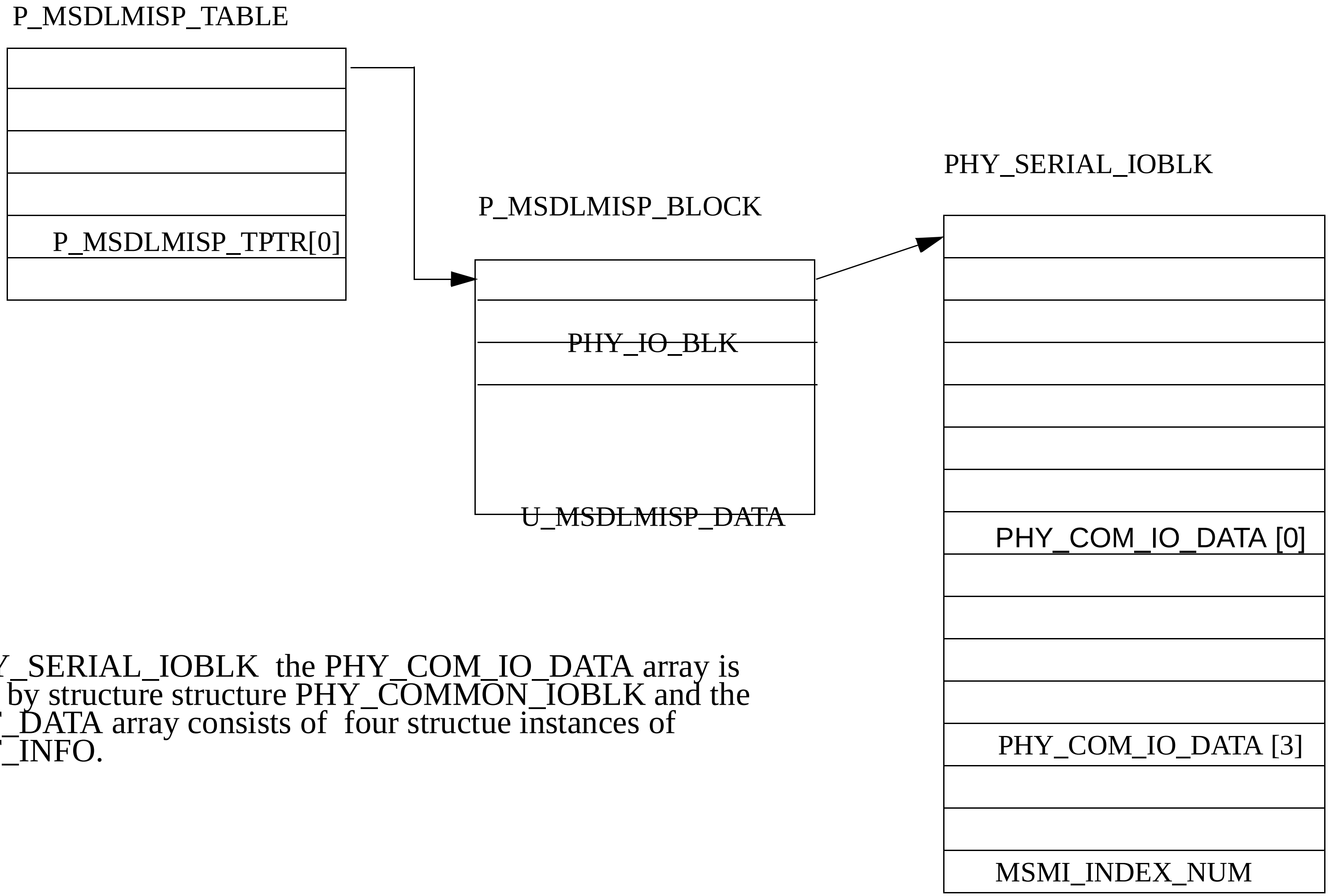
To Comfirm that the corruption has cleared

LD 17

```
REQ  chg
TYPE adan
ADAN new dch 9
CTYP msdl
DNUM 14
PORT 3
DES
DPNS
USR
..
..
```

Perform an EDD in LD 43.

MSDL STRUCTURE



NOTE:-

Inthe PHY_SERIAL_IOBLK the PHY_COM_IO_DATA array is overlayed by structure structure PHY_COMMON_IOBLK and the IO_PORT_DATA array consists of four structue instances of IO_PORT_INFO.

17 DCH channel Corruption

(the way to clear this corruption is very similar to the one cleared on the TTY corruptions mentioned in this Doc.)

DCH 11 and 13 are corrupted, not possible to remove them

version (1811, release 2119)

17.1 Symptoms of problem

```
>LD 96
DCH000
.stat dch
DCH 11 : DSBL  RST                DES : LOOP06XBERG2XL52
DCH 13 : DSBL  RST                DES : LOOP04XSIEG2XL04
DCH 14 : OPER  EST   ACTV  AUTO        DES : LOOP13_FT
DCH 15 : OPER  EST   ACTV  AUTO        DES : LOOP12_FT
DCH 62 : OPER  EST   ACTV  AUTO        DES : LOOP16_BERGERE
DCH 63 : OPER  EST   ACTV  AUTO        DES : LOOP0_SIEGE2

>LD 22
REQ  prt
TYPE adan dch 11
ADAN     DCH 11
  CTYP DCHI
  DNUM 11
  DES  LOOP06XBERG2XL52
  USR  PRI
  DCHL 6      <-------
  OTBF 16
  DRAT 64KC
  CLOK EXT
  NASA YES
  IFC  SL1
  SIDE USR
  CNEG 1
  RLS  ID  20
  RCAP ND2 NCT
  MBGA NO
  OVLR NO
  OVLS NO
  T23  20
  T200 3
  T203 10
  N200 3
  N201 260
  K    7


REQ  prt
TYPE adan dch 13
ADAN     DCH 13
  CTYP DCHI
  DNUM 13
  DES  LOOP04XSIEG2XL04
  USR  PRI
  DCHL 4       <-------
  OTBF 16
  DRAT 64KC
  CLOK EXT
  NASA YES
  IFC  SL1
  SIDE USR
  CNEG 1
  RLS  ID  20
  RCAP ND2 NCT
  MBGA NO
  OVLR NO
  OVLS NO
  T23  20
  T200 3
  T203 10
  N200 3
  N201 260
  K    7


REQ  prt
TYPE cequ
CEQU
  MPED 8D
  TERM
  REMO
  TERD
  REMD
  TERQ  001  017
  REMQ
  SUPL  004  020
  XCT   002  018
  TDS  * 002 * 018
  CONF * 003 * 019
  MFSD * 002 * 018
  PRI2 000 012 013 016
  DTI2
  MISP  024
  EXT0  3PE
    CNI  012 000 000
  EXT1  3PE
    CNI  012 000 000
  MCFN 004 004 004 004 016 016

REQ  ****


>LD 17
CFN000
MEM AVAIL: (U/P): 6916535    USED: 489032    TOT: 7405567
DISK RECS AVAIL: 2574
DCH  AVAIL:    57    USED:     6    TOT:    63
AML  AVAIL:    16    USED:     0    TOT:    16
REQ  chg
TYPE adan
ADAN out dch 11
SCH4732

ADAN out dch 13
SCH4732
```

(SCH4732 : Cannot remove the D-channel when B-channel is still defined for loops associated with this D-channel.)

```
>ld 97
SCSYS000
MEM AVAIL: (U/P): 6916535    USED: 489032    TOT: 7405567
DISK RECS AVAIL: 2574
REQ  prt
TYPE supl
SUPL 4

SUPL  SUPT SLOT XPEC0   XPEC1

 004  STD  LEFT 04 0 3  05 0 3
```

Problem description

Loop 4 associated to DCH 13 is a superloop,
and loop 6 associated to DCH 11 has disappeared.
It is not possible to remove DCH 11 and 13.

Note : it seems we've come to this configuration after one made an "out"

of loop 4 (system did not react to that !) then reconfigured loop 4
as a superloop (with sets attached to it). As loop 4 became a superloop,
loop 6 has then not to exist !

17.2 Software structure

see structure diagram.

```
PPOINTER [.P_BASIC_BASE] LOG_IO_PTR % pointer to the LOG_IO_MHT_TBL struc
ture

PSTRUCTURE [.P_BASIC_BASE] LOG_IO_MHT_TBL
  INTEGER LOG_BLK_LEN (0,0,8),
  PPOINTER [.P_BASIC_BASE] LOG_IO_TBL_PTR (0) [.MAX_LOG_APPLI];
                % LOG_IO_TBL_PTR[.PRA_LOG_APPLI] points to the LOG_IO_TBL structure
                  containing pointers to DCH blocks.

PSTRUCTURE [.P_BASIC_BASE] LOG_IO_TBL
    INTEGER BLK_LENGTH (0,0,8),
    SVR_SPAWNED (0,15,1),
    PPOINTER [.P_DCH_DATA]
       P_DCH_BLK_PTR (1) [.MAX_NO_OF_DCH], % pointer to structure P_DCH_BLOCK
    PPOINTER [.P_CSL_DATA]
       P_CSL_BLK_PTR (1) [.MAX_CSL_LINKS],
    ...
```

17.3 To solve the problem

We have to clear info related to DCH 11 and 13 in structure LOG_IO_TBL:
----

a/ Find address of LOG_IO_PTR (2119 1811), and print its content.

```
pdt> su
-> 0x27f40/4
value = 40912 = 0x9fd0
-> exit

pdt> exit
pdt> p 9fd0

00009FD0 : 00026ADA
```

b/ this is the address of LOG_IO_MHT_TBL. Use it to find
LOG_IO_TBL_PTR[.PRA_LOG_APPLI].

```
pdt> p 26ADA,5
00026ADA : 00000005 00026CF6 00000000 00026ADF 00000000
```

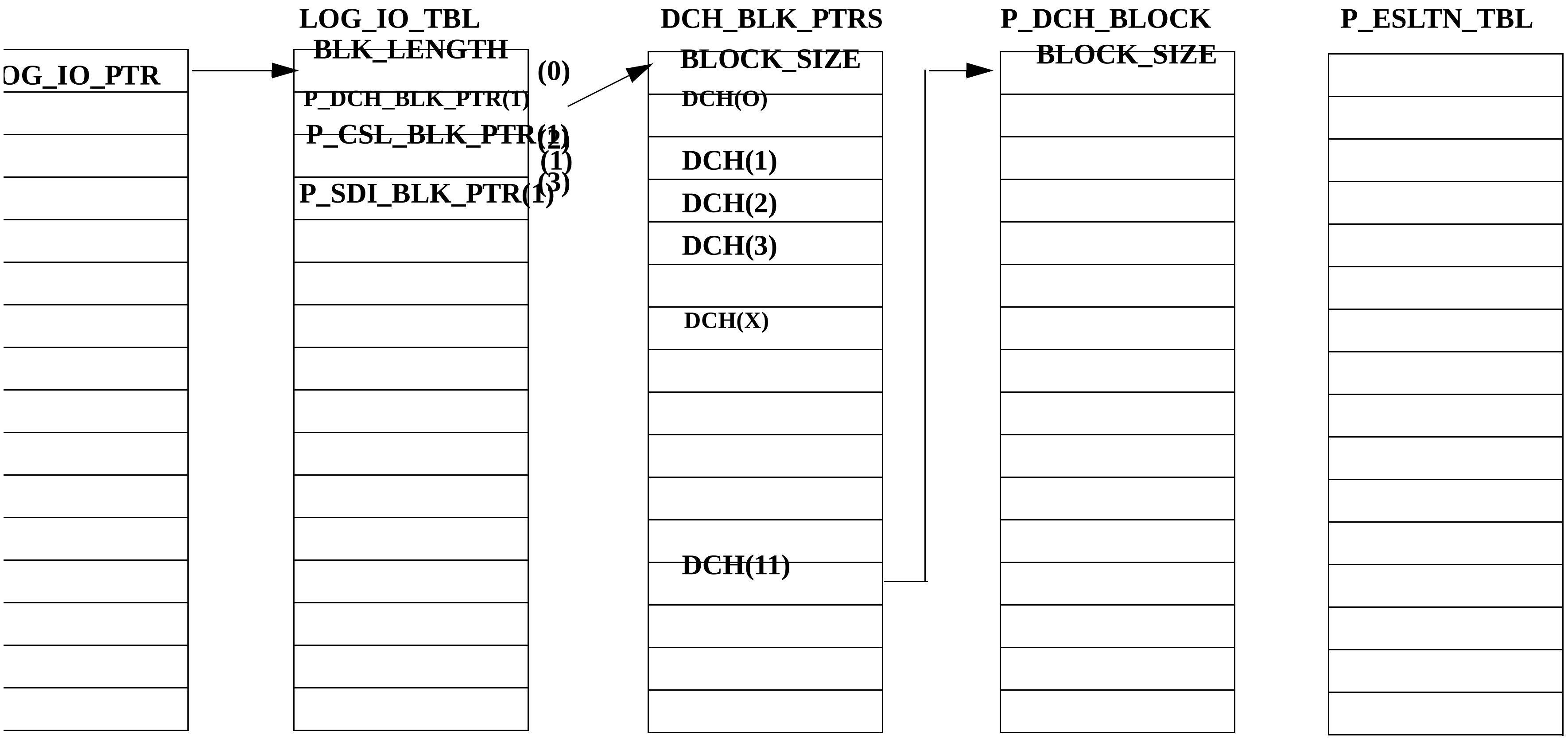
c/ this is the address of the wanted **LOG_IO_TBL**. Print this table, then clear info related to DCH 11 and 13.

```
pdt>  p 26CF6 10
00026CF6 : 00000041 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00026CFE : 00000000 00000000 00000000 00000000 00026D37 00000000 00026D91 00026DEB
                                               (DCH 11)          (DCH 13)


pdt>  w 00026D02
00026D02 : 00026D37 /0

pdt>   w 00026D04
00026D04 : 00026D91 /0
```

3/ To confirm DCH 11 and 13 have been cleared

```
>ld 22
REQ  prt
TYPE adan dch 11

DCH 11  IS UNDEFINED

REQ  prt
TYPE adan dch 13

DCH 13  IS UNDEFINED

>ld 96
DCH000
.stat dch
DCH 14 : OPER  EST   ACTV  AUTO        DES : LOOP13_FT
DCH 15 : OPER  EST   ACTV  AUTO        DES : LOOP12_FT
DCH 62 : OPER  EST   ACTV  AUTO        DES : LOOP16_BERGERE
DCH 63 : OPER  EST   ACTV  AUTO        DES : LOOP0_SIEGE2
```

STUCTURE FOR CHID FOR PRI ONLY



18 TTY Corruption on opt 11c

SCH6701 WHEN ADDING NEW TTY ON AN OPTION 11C

It seems we have a new corruption problem on Release 22 with the Option 11C's.

It is possible to configure TTY's on the expansion cabinets, but only ONE TTY is allowed per cabinet.

18.1 SYMPTOMS OF PROBLEM

If when trying to enter a new TTY a SCH6701 is generated then we already have a TTY assigned to that cabinet. If in Overlay 22 there are no TTY's marked against the cabinet then the corruption is present.

18.2 SOFTWARE STRUCTURE

see stucture diagram

18.3 HOW TO RESOLVE THIS PROBLEM

By looking in Procedure FIND_EXP_TTY we can see that a TTY has a cabinet number defined by :-

```
CAB_NUM:PHYS_IO_PTR[INDEX]:IO_TABLE_PTR
```

We can obtain the address of IO_TABLE_PTR from Xview. There are 15 pointers in the array PHYS_IO_PTR, one for each TTY. CAB_NUM is defined as :-

```
PSTRUCTURE [.P_BASIC_BASE] PHY_COMMON_IOBLK
..
..
CAB_NUM (3,7,4)
..
..
```

The corruption occurs when an unconfigured TTY has a cabinet number other than 0.

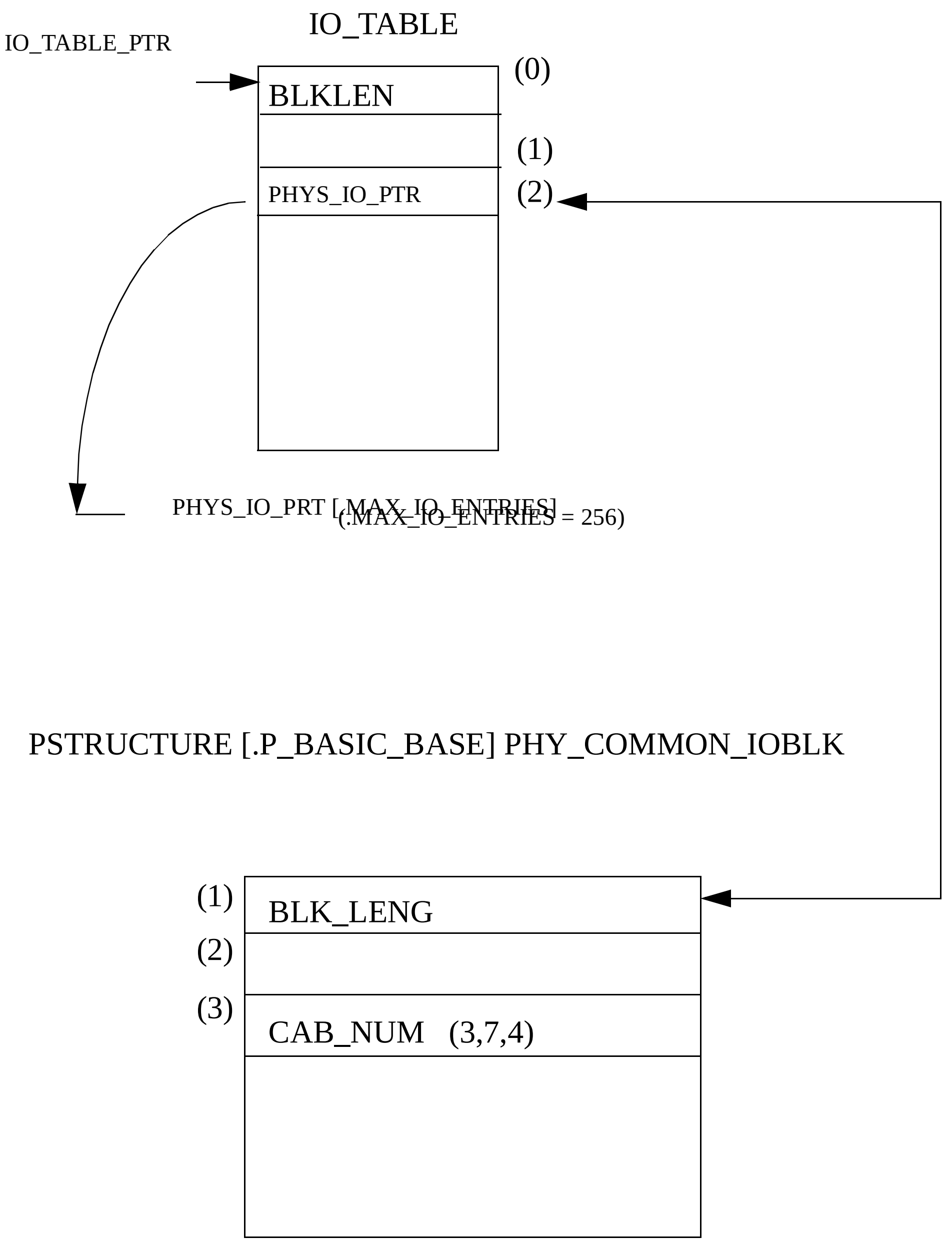
ie. 00008089

This would imply the TTY was in Cabinet 1. Clear this bit to enable the TTY to be configured in that cabinet.

ie. 00008009

As you do not know which TTY is corrupted is is advisable to go through all the unconfigured TTY's in the array PHYS_IO_PTR, as maybe more than one TTY has been corrupted

STRUCTURE DIAGRAM



19 XPEC Corruption on Superloop

19.1 SYMPTOMS OF PROBLEM

This problem is indicated when EDD039 has occured whilst performing a data dump in LD 43 EDD.

EDD039 :- Audit indicates that peripheral controller and superloop data does not agree and the dump is aborted.

This problem has only been seen on **OPT 11/11E/11C**

In our example xpec 22 is corrupted

1) Perform a datadump

```
> LD 43
EDD000
edd clr
EDD039
```

2) Print all of the XPEC, you can see that XPEC 22 is corrupted

```
LD 97
SCSYS000
MEM AVAIL: (U/P): 575117    USED: 113010    TOT: 688127
DISK RECS AVAIL: 431
REQ  prt
TYPE xpe
XPEC
    S0  S1  S2  S3  LOC     DIS RGTP
 01 000 000             NO  08
 02 004 004             NO  08
 03 008 008             NO  08
 04 012 012             NO  08
 05 016 016             NO  08
 06 032 032 032 032         NO  08
 07 036 036 036 036         NO  08
 08 040 040 040 040         NO  08
```

```
09 048 048 048 048        NO  08
17 193 000 000 000 MARI  NO  00
18 000 000 000 000   NO  00
20 104 104              NO  08
21          104 104      NO  08
22 138 084 000 000 Je  NO  00      <------ Corruption on xpec 2
23          108 108      NO  08
24 112 112              NO  08
37          136 136      NO  08
```

3) Print all of the Superloop to check if any s/loop is configured with the corrupted XPEC (22) , In our case loop 108 is assocated with xpec 22

```
REQ prt
TYPE supl
SUPL
```

```
SUPL  SUPT SLOT XPEC0   XPEC1

 000  STD  LEFT 01 0 1  -- - -
 004  STD  LEFT 02 0 1  -- - -
 008  STD  LEFT 03 0 1  -- - -
 012  STD  LEFT 04 0 1  -- - -
 016  STD  LEFT 05 0 1  -- - -
 032  STD  LEFT 06 0 3  -- - -
 036  STD  LEFT 07 0 3  -- - -
 040  STD  LEFT 08 0 3  -- - -
 048  STD  LEFT 09 0 3  -- - -
 064  ---- ---- PHANTOM -- - -
 104  STD  LEFT 20 0 1  21 2 3
 108  STD  LEFT 22 4 4  23 2 3     Remove xpec0  (22) on superloop 108
 112  STD  LEFT 24 0 1  25 2 3
 116  STD  LEFT 26 0 1  27 2 3
 120  STD  LEFT 28 0 1  29 2 3
 124  STD  LEFT 30 0 1  31 2 3
 128  STD  LEFT 32 0 1  33 2 3
 132  STD  LEFT 34 0 1  35 2 3
 136  STD  LEFT 36 0 1  37 2 3
 140  STD  LEFT 38 0 1  39 2 3
 144  STD  LEFT 40 0 1  41 2 3
 148  STD  LEFT 42 0 1  43 2 3
 152  STD  LEFT 44 0 1  45 2 3
 156  STD  LEFT 46 0 1  47 2 3
```

4) Remove xpec0 22 from s/loop 108

```
REQ  chg
TYPE supl
SUPL 108
XPE0 x
XPE1
WRAP UP SUPL 108 ..OK

REQ  prt
TYPE supl
SUPL
```

```
SUPL  SUPT SLOT XPEC0   XPEC1

 000  STD  LEFT 01 0 1  -- - -
 004  STD  LEFT 02 0 1  -- - -
 008  STD  LEFT 03 0 1  -- - -
 012  STD  LEFT 04 0 1  -- - -
 016  STD  LEFT 05 0 1  -- - -
 032  STD  LEFT 06 0 3  -- - -
 036  STD  LEFT 07 0 3  -- - -
 040  STD  LEFT 08 0 3  -- - -
 048  STD  LEFT 09 0 3  -- - -
 064  ---- ---- PHANTOM -- - -
 104  STD  LEFT 20 0 1  21 2 3
 108  STD  LEFT -- - -  23 2 3 --------------XPEC0  22 has been removed
 112  STD  LEFT 24 0 1  25 2 3
 116  STD  LEFT 26 0 1  27 2 3
 120  STD  LEFT 28 0 1  29 2 3
 124  STD  LEFT 30 0 1  31 2 3
 128  STD  LEFT 32 0 1  33 2 3
 132  STD  LEFT 34 0 1  35 2 3
 136  STD  LEFT 36 0 1  37 2 3
 140  STD  LEFT 38 0 1  39 2 3
 144  STD  LEFT 40 0 1  41 2 3
 148  STD  LEFT 42 0 1  43 2 3
 152  STD  LEFT 44 0 1  45 2 3
 156  STD  LEFT 46 0 1  47 2 3
```

5) Try to remove XPEC 22 get SCH 5728.

SCH5728:- Cannot delete a controller that is not empty

```
REQ chg
TYPE xpe
XPEC x22
   S0  S1  S2  S3  LOC     DIS RGTP
 22 138 084 000 000 Je  NO  00

XPEC NOT EMPTY
SCH5728    <----------- .SCH_5728
XPEC
REQ  chg
TYPE **

MEM AVAIL: (U/P): 575117    USED: 113010    TOT: 688127
DISK RECS AVAIL: 431
REQ  prt
TYPE supl
SUPL

SUPL  SUPT SLOT XPEC0   XPEC1

 000  STD  LEFT 01 0 1  -- - -
 004  STD  LEFT 02 0 1  -- - -
 008  STD  LEFT 03 0 1  -- - -
 012  STD  LEFT 04 0 1  -- - -
 016  STD  LEFT 05 0 1  -- - -
 032  STD  LEFT 06 0 3  -- - -
 036  STD  LEFT 07 0 3  -- - -
 040  STD  LEFT 08 0 3  -- - -
 048  STD  LEFT 09 0 3  -- - -
 064  ---- ---- PHANTOM -- - -
 104  STD  LEFT 20 0 1  21 2 3
 108  STD  LEFT -- - -  23 2 3
 112  STD  LEFT 24 0 1  25 2 3
 116  STD  LEFT 26 0 1  27 2 3
 120  STD  LEFT 28 0 1  29 2 3
 124  STD  LEFT 30 0 1  31 2 3
 128  STD  LEFT 32 0 1  33 2 3
 132  STD  LEFT 34 0 1  35 2 3
```

```
136  STD  LEFT 36 0 1  37 2 3
140  STD  LEFT 38 0 1  39 2 3
144  STD  LEFT 40 0 1  41 2 3
148  STD  LEFT 42 0 1  43 2 3
152  STD  LEFT 44 0 1  45 2 3
156  STD  LEFT 46 0 1  47 2 3
```

19.2 SOFTWARE STRUCTURE

see structure diagram

19.3 HOW TO RESOLVE PROBLEM

1) Find the address of SYS_XPEC see Appendix

In our case the address is 9687 for rls 2246

2) In debug do:-

```
pdt> p 9687 20

xpec      (0)    (1)    (2)     (3)    (4) ----etc--->
00009687 : 00000000 00037DEE 00037DF8 00037E02 00037E0C 00037E1 00037E20 00037E2A
0000968F : 00037E34 00037E3E 00000000 00000000 00000000 00000000 00000000 00000000
00009697 : 00000000 00037E52 00037E5C 00000000 00037E70 00037E7A 00037E84 00037E8E
0000969F : 00037E98 00037EA2 00037EAC 00037EB6 00037EC0 00037ECA 00037ED4 00037EDE
```

3) Find the correct address of that xpec, in our case XPEC is 22.

Then we need to remove this, i.e xpec 22 the address is 969d.

```
pdt> p 969d 00037E84  % print this do double check if it’s correct

pdt> w 969d

0000969D : 00037E84 /0  % removing xpec 22
```

4) To confirm corruption is cleared print XPEC to see if it’s gone.

```
REQ  prt
TYPE xpe
XPEC

   S0  S1  S2  S3  LOC     DIS RGTP


 01 000 000              NO  08
 02 004 004              NO  08
 03 008 008              NO  08
 04 012 012              NO  08
```

```
05 016 016          NO  08
06 032 032 032 032      NO  08
07 036 036 036 036      NO  08
08 040 040 040 040      NO  08
09 048 048 048 048      NO  08
17 193 000 000 000 MARI  NO  00
18 000 000 000 000  NO  00
20 104 104          NO  08
21      104 104     NO  08
23      108 108     NO  08
24 112 112          NO  08
25      112 112     NO  08
26 116 116          NO  00
27      116 116     NO  08
28 120 120          NO  08
29      120 120     NO  08
30 124 124          NO  08
31      124 124     NO  08
32 128 128          NO  08
33      128 128     NO  08
34 132 132          NO  08
35      132 132     NO  08
36 136 136          NO  08
37      136 136     NO  08
38 140 140          NO  08
39      140 140     NO  08
40 144 144          NO  08
41      144 144     NO  08
42 148 148          NO  08
43      148 148     NO  08
44 152 152          NO  08
45      152 152     NO  08
46 156 156          NO  08
47      156 156     NO  08
```

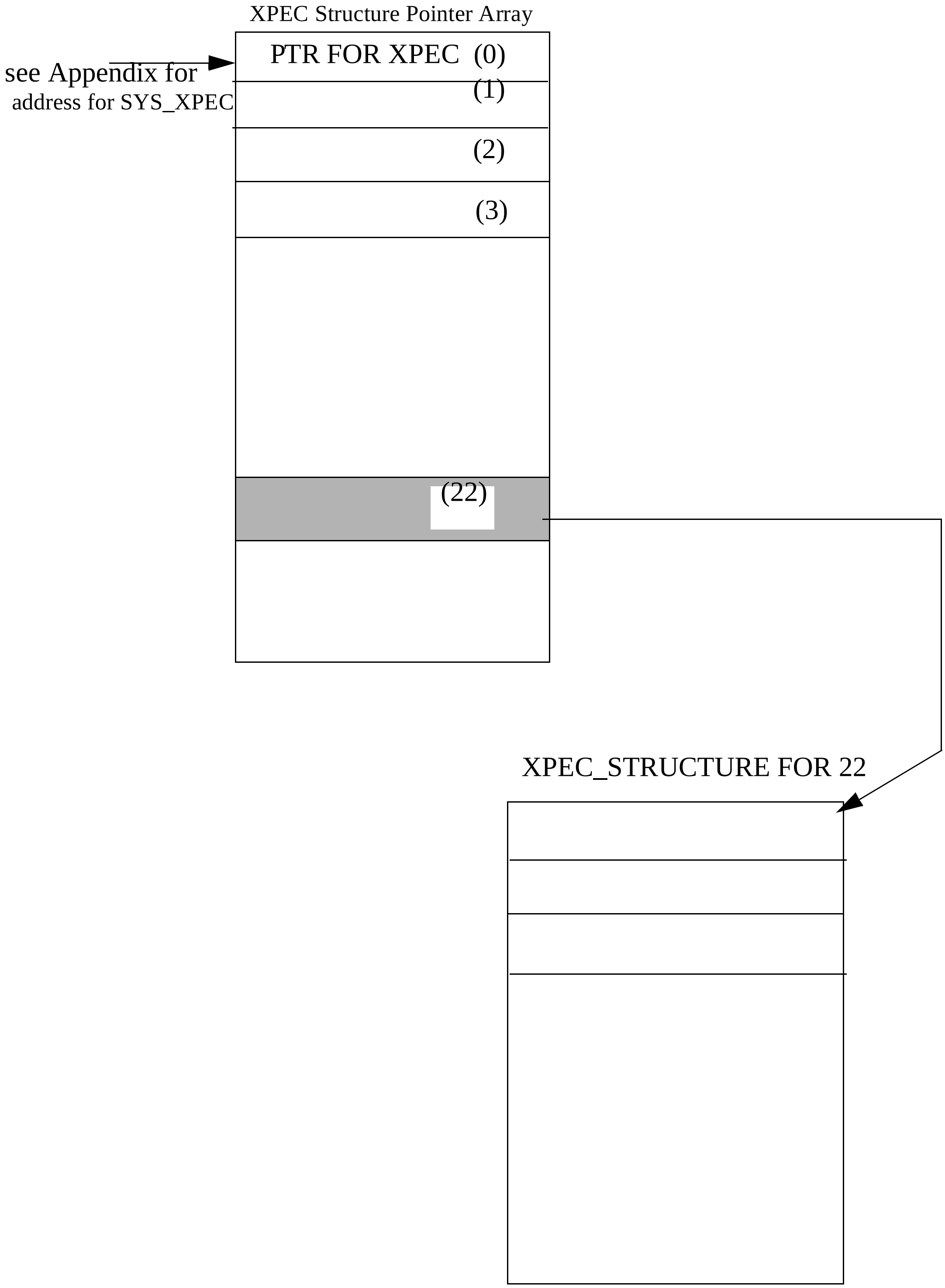
5) Perform a data dump in LD 43 EDD

```
>ld 43
edd

DB SEQ NUM = 29
CONFIG
PHYSICAL MAP
BCS TEMPLATE
PBX TEMPLATE
```

```
CUST
ACUST
CLID
ROUTE
LTN TN
LTN LNK
ICP BLK
TN
SCL
ESN  00
NCTL
ACD
GRP DNS
CPK
FRL
NFCR  TREES
ASNCH
BG-TIME
BG-CAT
DCH
ARIES
SYSP
XPEC
XTDT
FTC
MCAD
FCAD
FDCT
DTI2
FFC
LAPW
FDTD
TIME
CPND
CPND NM
SPECIFIC DATA
  ALARM_MGT
CHECKING

RECORD COUNT =  0081

Starting internal database backup
to internal backup drive
Synching drives
Updating internal backup
Backing up c:/p/sl1/direct.rec
Backing up c:/p/disk.sys
Backing up c:/p/os/diskoscc.sym
Backing up c:/p/sl1/ovlrescc.sym
Backing up c:/p/sl1/sl1rescc.sym
Backing up c:/u/db/database.rec
Backing up c:/u/db/config.rec
Backing up c:/u/db/inet.db
Backing up c:/u/patch/reten/reten.pch
Backing up c:/u/patch/atadrvf.p
Backing up c:/u/patch/dti311.p
Backing up c:/u/patch/p07148.loc
Backing up c:/u/patch/p09140.11c
Backing up c:/u/patch/p07679.loc
Backing up c:/u/patch/p07742.loc
Backing up c:/u/patch/p08228.loc
Backing up c:/u/patch/p08405.loc
Backing up c:/u/patch/p08641.loc
Backing up c:/u/patch/p08693a.loc
Backing up c:/u/patch/p08693b.glb
Backing up c:/u/patch/p08764.loc
Backing up c:/u/patch/p09016.loc
Backing up c:/u/patch/p09379.loc
Backing up c:/u/patch/p09401.loc
Backing up c:/u/patch/p09456.loc
Backing up c:/u/patch/p09294.11c
Backing up c:/u/patch/pdiagflo.11c
Backing up c:/u/patch/p08891.11c
Backing up c:/u/patch/p08754.11c
Internal backup complete
All files are backed up!
DATADUMP COMPLETE

.


  CORRUPTION ON XPEC 22 HAS BEEN CLEARED.
```

STRUCTURE DIAGRAM

PPOINTER [.P_BASIC] SYS_XPEC [.MAX_XPE_ALLOWED +1]
.MAX_XPE_ALLOWED =95



20 CDN Corruption

20.1 SYMPTOMS OF PROBLEM

The symptom that has been seen for this type of problem is that you can not remove a CDN in LD 23 when it has a default ACDDN and it’s not in control mode.

Get SCH5351

```
TYPE CDN
CUST 0
CDN  40208
FRRT 23
FRT
SRRT 25
SRT  30
FROA NO
MURT 20
DFDN 40878
CEIL 0
OVFL NO
TDNS NO
RPRT NO
CNTL NO

>ld 23

MEM AVAIL: (U/P): 6366775    USED: 1038792    TOT: 7405567
DISK RECS AVAIL: 2728
ACD DNS  AVAIL: 23949    USED:    51    TOT: 24000
REQ  out
TYPE cdn
CUST 0
CDN  40208
SCH5351
```

=> **SCH5351**

Cannot remove a CDN when it has default calls in its default ACD-DN.

20.2 SOFTWARE STRUCTURE

see ACD Stucture Diagram

20.3 HOW TO RESOLVE THIS PROBLEM

We need to remove all of the info of the existing call in the

U_ACD_BLOCK

1) In debug print CDATAPTR by doing DCP <CUST> To Find the ACD_LIST_PTR at word 87hex

```
pdt> p 8a5a
00008A5A : 0001C7AD
```

ACD_LIST_PTR

```
pdt> p 0001C7AD 87
0001C7AD : 00004B04 00000000 0000FFFF 00000000 00000000 00000000 00000000 00000000
0001C7B5 : 00000000 000051E5 000089A5 00000000 00000000 0072454C 00000000 000006E0
0001C7BD : 00000002 00000000 00000000 00000000 00000000 00000000 0002647C 00000100
0001C7C5 : 0000001A 00005244 00000000 000000C0 00000000 00000000 00000000 00000000
0001C7CD : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C7D5 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C7DD : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C7E5 : 00000000 00000000 00000000 00000000 00000000 00000000 0000001E 00000000
0001C7ED : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C7F5 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C7FD : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00008000
0001C805 : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C80D : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001C815 : 00000000 00000000 00000000 0000821F 0000596D 0000007F 0000003F 00001400
0001C81D : 00001C14 0000201E 00000000 00000000 00000000 00000000 00000000 00000000
0001C825 : 00000000 00000444 00000444 00000000 00000200 0000192C 00000000 00000000
0001C82D : 00000000 0000FF20 0000FFFF 00000000 00008000 0000201E 0002BC65
```

ACD_LIST_PRT = 2BC65

2) Print the ACD_LIST_PRT to find the correct location for the P_ACD_BLOCK for the following CDN & ACD-DN

a) CDN 40208

b) ACD-DN 40878

```
pdt> p 0002BC65 2b
0002BC65 : 00000035 00025854 0002E744 006DC6D4 0002575E 00000000 0002569A 0002564F
0002BC6D : 00025604 000255B9 0002556E 000254F6 0002547E 0002541F 000253C0 00025361
0002BC75 : 00025302 000252A3 00025244 000251E5 00025186 00025127 000250C8 00025069
0002BC7D : 0002500A 00024FAB 00024F4C 00024EED 00024EA2 00024E57 00024E0C 00024DC1
0002BC85 : 00024D62 00024CF9 00024CAE 00024C63 00024C18 00024BCD 00024B82 00024B37
0002BC8D : 00024AEC 00024AA1 00025F7A
```

The correct address for the P_ACD_BLOCK are:-

a) CDN = **00025F7A**

b) ACD-DN = **0002E744**

3) Print the P_ACD_BLOCK for this CDN

```
pdt> p 00025F7A 5
00025F7A : 0000013A 0000A2A4 00000008 0071A7B9 00024A90
             0          1       2        3
```

Were :-

WORD 1 & 2 = CDN 40208

WORD 3 = Unprotected ACD Block (U_ACD_BLOCK)

4) print the U_ACD_BLOCK

```
pdt> p 0071A7B9 2b

0071ED2F : 00000013 00000001 0071ED2F 00000001 00000000 00000013 00000001 0071ED34
0071ED37 : 00000001 00000000 00000013 00000001 0071ED39 00000001 00000000 00000013
0071ED3F : 00000001 0071ED3E 00000001 00000000 00000001 00000000 00000000 00000000
0071ED47 : 00000000 00000013 00000001 0071ED48 00000001 00000000 00000001 00000013
0071ED4F : 00000001 0071ED4E 00000001 00000000 00000001 00000001 00000001 00000001
0071ED57 : 00000001 00000000 00000001
```

CCR_#DFLT is located at address **71ED59 = 00000001**

5) We need to now check if there are any active calls in the

ACD Queue (40878) by printing of the Procted ACD Block for that ACD-DN.

```
dbg> p 0002E744 5
0002E744 : 0000003A 000078A4 00000008 006DC6D4 000416BD
             0         1         2       3        4
```

Were:-

WORD 1 & 2 = ACD-DN (40878)

WORD 3 = U_ACD_BLOCK

```
dbg> p 006DC6D4 2b

006DC6D4 : 00000013 00000001 006DC6D4  00000001  00000000 00000013 00000001 006DC6D9
006DC6DC : 00000001 00000000 00000013  00000001  006DC6DE 00000001 00000000 00000013
006DC6E4 : 006FA7F7 006EDF47 00000001 #00000003# 00000001 00000000 00000000 006DC6A3
006DC6EC : 00000000 00000013 00000001  006DC6ED  00000001 00000000 00000001 00000013
006DC6F4 : 00000001 006DC6F3 00000001  00000000  00000001 00000001 00000001 00000001
006DC6FC : 00000001 00000000 00000000
```

There are currently 3 calls in ACD Queue, we need wait until there no call in the Queue.

3 Calls queued in ACDQ located at address 006DC6E7 = #00000003#

and the CCR_#DFLT =0 at address 6DC6FE

```
dbg> p 006DC6D4 2b / u_acd_block of ACD list , CCR_#DFLT (42,0,11)

006DC6D4 : 00000013 00000001 006DC6D4  00000001  00000000 00000013 00000001 006DC6D9
006DC6DC : 00000001 00000000 00000013  00000001  006DC6DE 00000001 00000000 00000013
006DC6E4 : 00000001 006DC6E3 00000001 #00000000# 00000001 00000000 00000000 006DC6A3
006DC6EC : 00000000 00000013 00000001  006DC6ED  00000001 00000000 00000001 00000013
006DC6F4 : 00000001 006DC6F3 00000001  00000000  00000001 00000001 00000001 00000001
006DC6FC : 00000001 00000000 00000000
```

There are now no calls in the DEFAULT ACDQ

6) print of the U_ACD_BLOCK for the CDN

```
dbg> p 0071ED2F 2b
0071ED2F : 00000013 00000001 0071ED2F 00000001 00000000 00000013 00000001 0071ED34
0071ED37 : 00000001 00000000 00000013 00000001 0071ED39 00000001 00000000 00000013
0071ED3F : 00000001 0071ED3E 00000001 00000000 00000001 00000000 00000000 00000000
0071ED47 : 00000000 00000013 00000001 0071ED48 00000001 00000000 00000001 00000013
0071ED4F : 00000001 0071ED4E 00000001 00000000 00000001 00000001 00000001 00000001
0071ED57 : 00000001 00000000 00000001
```

CCR_#DFLT (42,0,11) is still equal 1, which means that there is 1 default call the CDN currently has in the default ACD-DN, But there no calls in the default ACD-DN
This is the reason why we cannot remove this CDN in LD 23.

Since there no call in the default ACDQ we can RESET this CCR_#DFLT to Zero.

```
dbg> w 71ED59
0071ED59 : 00000001 /0
```

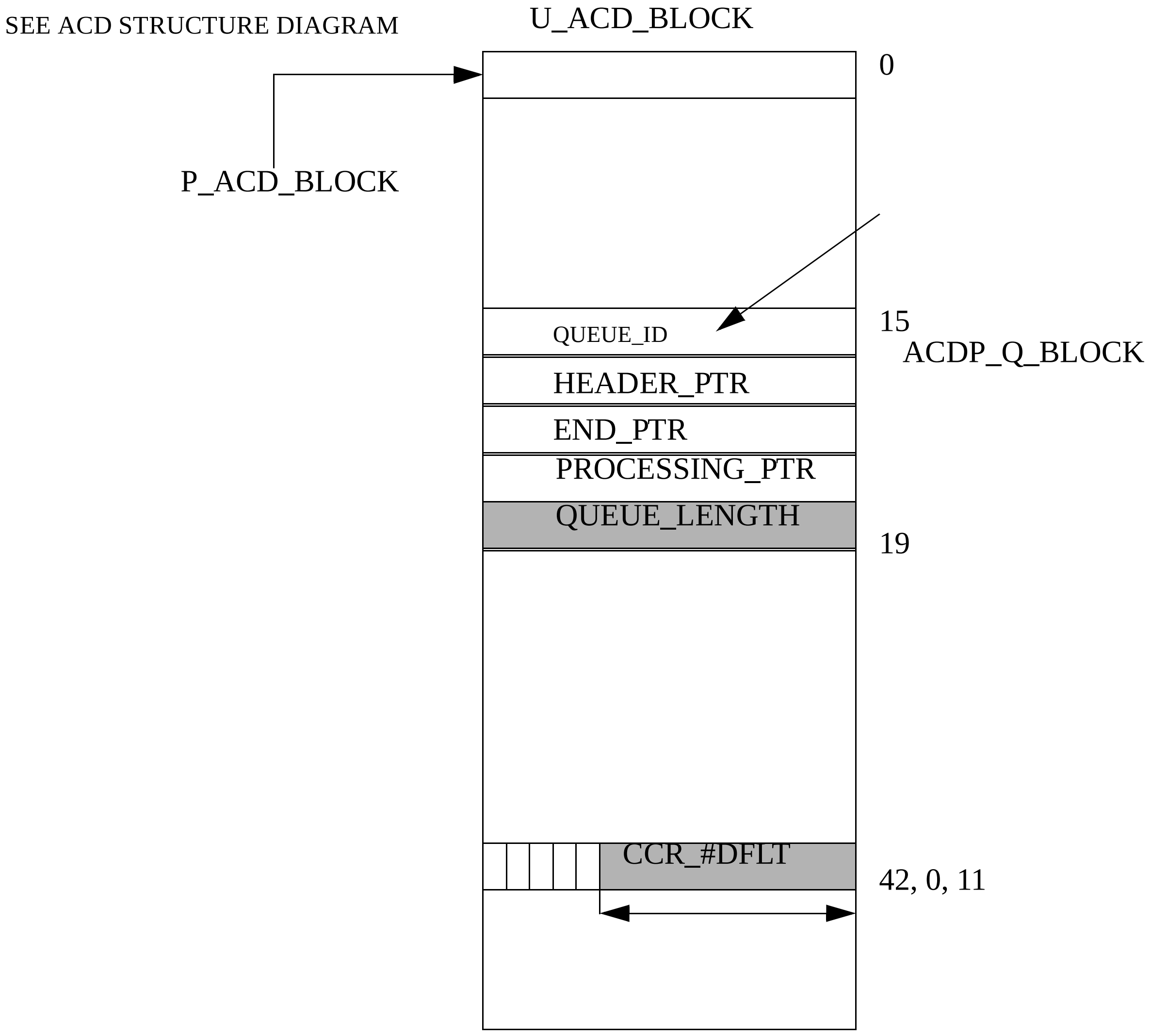
7) We can remove the CDN 40208 in LD 23

```
ld 23
REQ  out
TYPE cdn
CUST 0
CDN  40208


....
....


REQ  prt
TYPE cdn
CUST 0
```

```
CDN  40208
SCH5293          % CDN must exist for CHG, OUT or PRT command.
CDN  **
```

8) Perform a DATA DUMP in LD 43
EDD.

Structure diagram to locate CCR_DFLT



21 FFC Corruption

21.1 symptoms of problem

This corruption is caused when existing FFC DN has been created in LD 57and a 500 set configured as a ACD Agent using the same ACID as the FFC DN, this should not be allowed as there is a DN conflict should get SCH0730, but it seems to allow it ,causing corruption in LD 57 i.e can not be outted, but if you print it, its there

After the corruption has been cleared, fit Patch MPLR11667 (BV82853) this patch will not allow any conflict with other DN's and will get SCH0730

Please note that this patch is not required for rls 24 as it's fixed.

HOW TO DUPLICATE

1) CREATE SSPU FFC CODE.

```
>LD 57
FFC000
UDATA: 11320 0  PDATA: 36945 27

REQ  CHG
TYPE FFC
CUST 0
FFCT
CODE SSPU
SSPU 71
SSPU
CODE

UDATA: 11320 0  PDATA: 36932 27

REQ  ****
```

ACTION

2) PROGRAM A NEW 500 SET OR CHANGE AN EXISTING ONE TO GIVE AN ACID WHICH USES THE SAME DIGITS AS THE FFC AND EXTRA.

ACTAUL

```
 >LD 10
 REQ  CHG
 TYPE 500
 TN   9 0 6 9
 ECHG YES
 ITEM FTR  ACD 310 710
   FTR
 ITEM

 UDATA: 11320 0  PDATA: 36917 30
```

EXPECTED

```
  >LD 10
 REQ  CHG 0R NEW
 TYPE 500
 TN   9 0 6 9
 ECHG YES

 ITEM FTR  ACD 310 710
SCH0730
   FTR
```

IMPACT := CAUSES CORRUPTION

3) AFTER IT HAS TAKEN THE CHANGES
CHECK THAT THE 500 SET IS
PROGRAMMED WITH FTR ACD 310 710

```
  >LD 20
 PT0000
 REQ  PRT
 TYPE TNB
 TN   9 0 6 9
 DATE
 PAGE
 DES
..
..
 CLS  AGTA
..
..
 FTR  CPND
 FTR  ACD 310 710  <- NOTE CHANGE!
      AGN
 DATE  3 SEP 1997
```

3) PRINT OUT DNBLOCK FOR 7 IN OVL 20 AND YOU GET A
SCH0886 - SHORTER DN EXISTS.

```
REQ  PRT
TYPE DNB
CUST 0
DN   7
DATE
PAGE
DES
DN   710
TYPE ACID

TN   009 0 06 09

SCH0886
```

4) TRY TO OUT FFC SSPU CODE AND YOU GET A
SCH8895 - FFC CODE DOES NOT EXIST
YET WHEN YOU PRINT IT IN 57 IT SHOWS UP!

```
>LD 57
REQ  OUT
TYPE FFC
CUST 0
ALL  NO
CODE SSPU
SSPU 71
SCH8895
```

SSPU

21.2 SOFTWARE STRUCTURE

(See StructureDiagram)

21.3 HOW TO RESOLVE PROBLEM

To Remove in debug do DNT < CUST #> <DN>

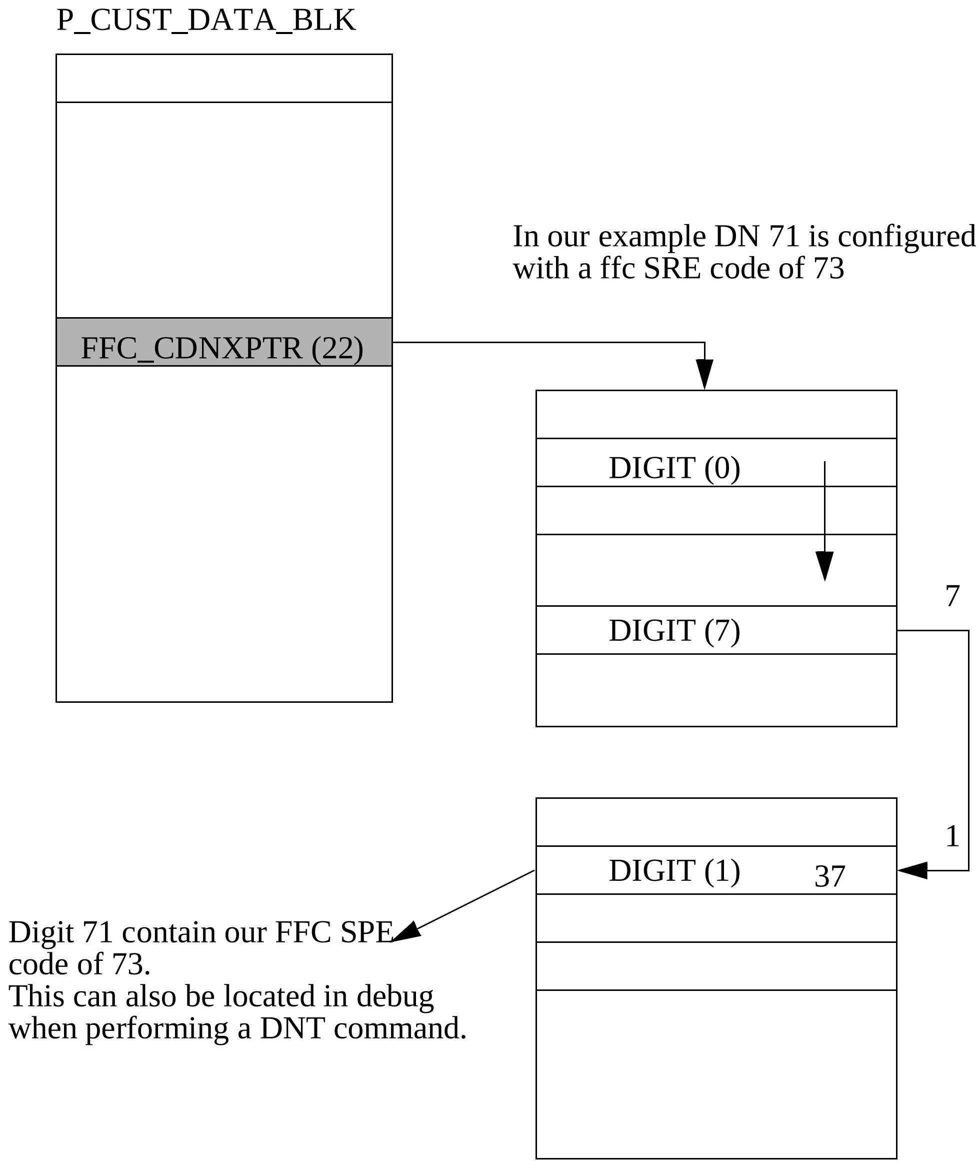
```
pdt> dnt 0 71



DIG 2 FFC SSPU
000714E0 :    00000080 00000037 00000000 00000000 00000000 00000000 00000000
000714ED
000714E8 :   00000000 00000000 00000000 00000000 00000000


pdt> w 000714E0



000714E0 : 00000080 /
000714E1 : 00000037 /0  "Remove the ffc spre code of 73 from digit 71"
pdt> dnt 0 71



DIG 2 INV
pdt>



REQ  prt
TYPE ffc
CUST 0
CODE sspu

CUST 00
FFCT NO
SCH8896 :-FFC data does not exist.
CODE
```

```
NOTE: '0037' IS THE .SPRE_SSPU(73hex)
```

4) PERFORM AN EDD IN OVL 43.

5) FIT PATCH MPLR11667 (Not required for rls 24)

Structure diagram to locate FFC_CDNXPTR


P_CUST_DATA_BLK
FFC_CDNXPTR (22)
In our example DN 71 is configured
with a ffc SRE code of 73
DIGIT (0)
DIGIT (7)
7
1
DIGIT (1)
37
Digit 71 contain our FFC SPE
code of 73.
This can also be located in debug
when performing a DNT command.

22 ADAN Corruption

22.1 SYMPTOMS OF PROBLEM

Symptoms of this problem is when trying to out or change

any ADAN block in LD 17 would result in BUG5513, it would

also indicate (ADAN DATA ERROR)

BUG5513 :- Serious problem in rebuilding the I/O table will cause data corruption and requires a system reload.

1)

```
>LD 17
REQ  CHG
TYPE ADAN
ADAN CHG HST
SIZE
USER

BUG5513
BUG5513  :2 4
BUG5513   + 105EDC2C 115B29EE 115ACECA 1159AB0C 1159A514
BUG5513 + 1087C02C 1087AF10 1087A7C2 10878D64 1087626A
BUG5513  + 10CA24C6 10CA1988 10AC183C 10CA17D4

ADAN DATA ERROR
```

2)

It look’s as if something has corrupted the ADAN block, first thing to do is to print out all of the configured ADAN block and then just try making a change on any of them.

```
LD 17
REQ CHG
TYPE ADAN

ADAN CHG AML 9
SCH5579 :-Physical I/O block pointer corruption.
```

First time lucky, AML 9 may be the root cause of this as it appear’s to be corrupted.

3) Print out the associated VAS for AML 9

```
LD 22
REQ PRT
TYPE VAS

VAS
 VSID 09
   DLOP
   AML 09
    SECU NO
    INTL 0001
    MCNT 9999
    CONF DIR
```

4) In LD 48 just to see what happen when we stat the AML link.

As you can see we have BUG5504, we deffanatly have corruption on this link

```
.STAT AML 9
.
BUG5504
BUG5504  :  00000002 00000009 0000256C 00000000 000D2934 00000000 00000000 000000
00 00000000
BUG5504   + 105EDC2C 105EB390 105EADCC 105EA8A0 11445AF8
BUG5504 + 11444486E 11443BA2 1144197C 1087C02C 1087AF10
BUG5504  + 1087A7C2 10878D64 1087626A 10CA24C6 10CA1A62
BUG5504 + 10CA183C 10CA17D4
```

22.2 SOFTWARE STRUCTURE

see software structure diagram.

22.3 HOW TO RESOLVE PROBLEM

1) WE NEED TO REMOVE AML 9 FROM WITHIN DEBUG

a) FIND THE LOG_IO_PTR (SEE APPENDIX) IN OUR CASE FOR 2246 11C

LOG_IO_PTR = A243

b) PRINT OUT THE ADDRESS OF THE LOG_IO_PTR

```
pdt> p A243

0000A243  : 00050D62

pdt p 00050D62 5

00050D62  : 00000005 00050F6D 00050E0E 00050D67 00000000
```

LOCATE THE P_CSL_BLK_PTR AT WORD 2

c) PRINT OUT THE P_CSL_BLK_PTR INDEXED BY THE AML # (9 IN OUR CASE)

```
pdt> p 00050E0E 10   P_CSL_BLK_PTR

00050E0E : 00000021 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00050E16 : 00000000 00000000 00050E2F 00000000 00000000 00000000 00000000 00000000



pdt>  w  00050E16

00050E16 : 00000000 /
00050E17 : 00000000 /
00050E18 : 00050E2F / 0 REMOVE AML 9
```

WE HAVE NOW REMOVED AML 9 FROM PHYSICAL I/O BLOCK

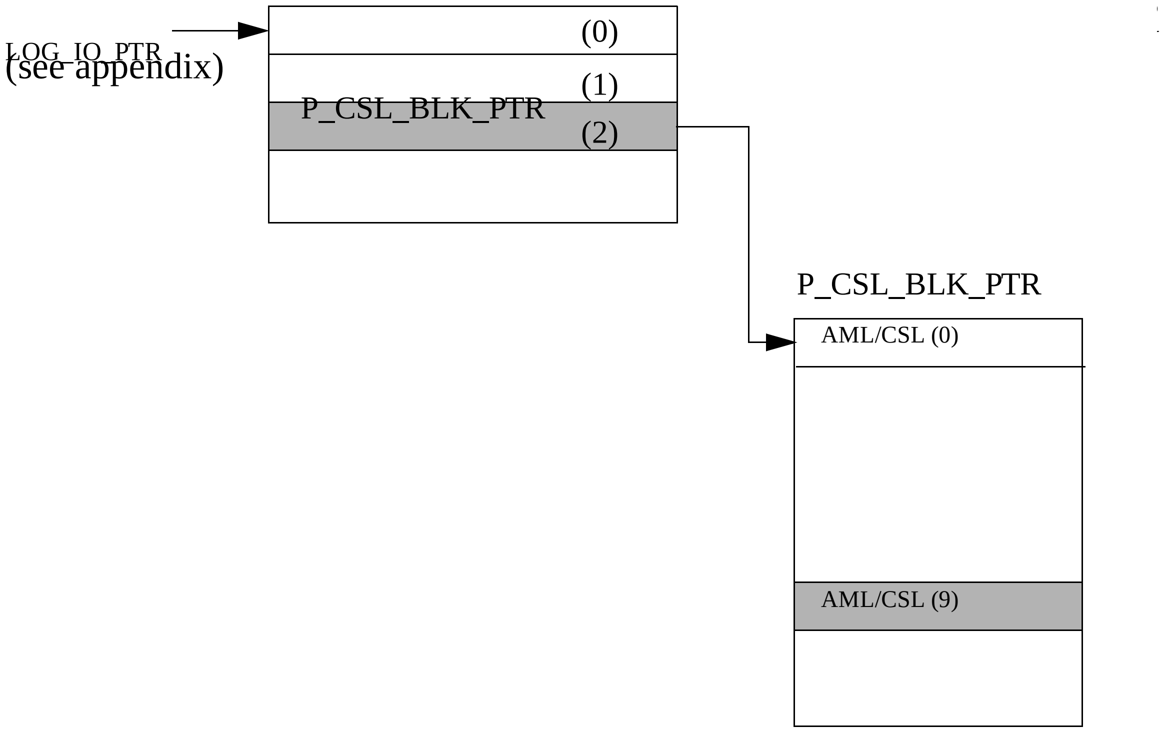
2) NOW WE NEED TO REMOVE THE ASSOCIATED VAS FROM WITHIN PROTECTED VAS BLOCK IN DEBUG

a) FIND THE **P_VAS_TBL_HDR** (SEE APPENDIX)

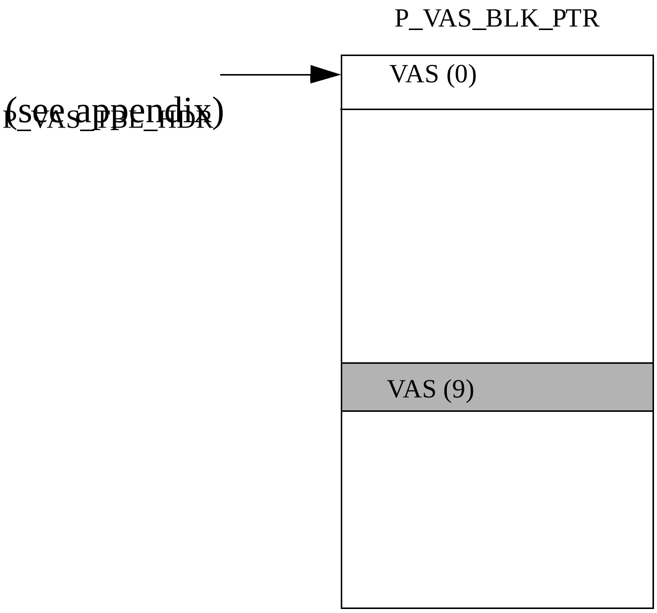
  P_VAS_TBL_HDR = 98E8

```
pdt> p 98e8
000098E8 : 00050E4B
```

STRUCTURE TO LOCATE AML



STRUCTURE TO LOCATE VAS



23 TRK Corruption

23.1 SYMPTOMS OF PROBLEM

Can't remove AWR trunks programmed on card slot 10 in LD 14

```
>LD 36
TRK000
.stat 10
UNIT 00 = IDLE         (TRK)(AWR  AUD)
UNIT 01 = IDLE         (TRK)(AWR  AUD)
UNIT 02 = UNEQ
UNIT 03 = UNEQ
.
```

1) print TNB in LD 20 , as you can see XTRK shows up as NON-XTRK !

```
> LD 20
TN   010 0 00 00
TYPE AWR
CUST 0
XTRK NON-XTRK
TIMP 1200
RTMB 20 1
DATE  4 NOV 1997

TN   010 0 00 04
TYPE AWR
CUST 0
XTRK NON-XTRK
TIMP 1200
RTMB 21 1
DATE  4 NOV 1997
```

2) Load OVERLAY 14 and try to remove TN 10 0 and 10 1:

```
>ld 14
REQ  out
TYPE awr
TN 10 0 0 0
```

SCH0126

```
>ld 14
REQ  out
TYPE awr
TN   10 1
```

SCH0126

SCH0126: Station type conflicts with existing card.

23.2 SOFTWARE STRUCTURE

23.3 HOW TO RESOLVE PROBLEM

1) first of all lets print out the TNTREE by doing TNT <L S C U> in debug for the two TN'S

```
pdt> tnt 10 0 0 0
 EQPD SLOOP TN 000808
 GP 00031046 SLP 00032116 000D5B16  CD 0003224A 000D5AA5  LN 00032252 000D5A53

pdt> tnt 10 0 0 1
 EQPD SLOOP TN 000848
 GP 00031046 SLP 00032116 000D5B16  CD 00032285 000D5A3D  LN 0003228D 000D59EB
```

2) Print the protected Card pointer for TN 10 00 and 10 01

```
pdt>  p 0003224A 10  for TN 10 0
0003224A : 00005040 00032252 00000000 00000000 00000000 000D5AA5 00000000 00000000
00032252 : 00000033 00000003 00000000 00000000 00000000 00000000 00000000 000D5A4C

pdt> p 00032285 10  for 10 1

00032285 : 00005040 0003228D 00000000 00000000 00000000 000D5A3D 00000000 00000000
0003228D : 00000033 00000003 00000000 00000000 00000000 00000000 00000000 000D59E4
```

Word 6 holds the value of type of XTRUNK that is configured in LD 14 in our case word 6 has the value ’0’ which is translated to being configured as a NON XTRK CARD, This is wrong it should be set to a value of 6 (.EXUT_CARD)

Field XTRUNK of PSTRUCTURE PCARDBLOCK (xtrunk(6,0,3))

```
The value of 0 = .NON_XTRK_CARD
             1 = .XUT_CARD
             2 = .XEM_CARD
             3 = XFEM_CARD
             4 = XCOT_CARD
             5 = XDID_CARD
             6 = EXUT_CARD
```

Let's change it to .EXUT_CARD (6)

```
pdt> w 00032250  for TN 10 0
00032250 : 00000000 /6
```

```
pdt>  w 3228b  for TN 10 01
0003228B : 00000000 /6
```

3) Now let’s try and out these TN’S

```
>ld 14
REQ  out
TYPE awr
TN   10 0
OUT TRK    TN  010 0 00 00     RT  20     MB  1
REQ  out
TYPE awr
TN   10 1
OUT TRK    TN  010 0 00 04     RT  21     MB  1
```

Trunk been removed successfully

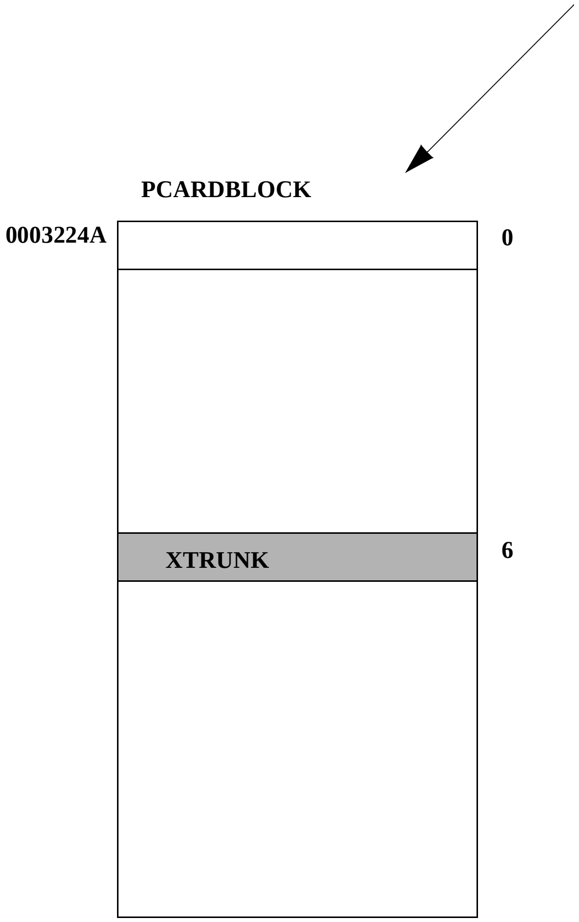
```
REQ
>ld 32
NPR000
.stat 10
CARD UNEQ
```

4) Perform a datadump

structure diagram to locate XTRUNK

Example of tn 10 00

```
pdt> tnt 10 0 0 0
 EQPD SLOOP TN 000808
 GP 00031046 SLP 00032116 000D5B16  CD 0003224A 000D5AA5  LN 00032252 000D5A53
```



24 LAPD (BRI) Corruption

24.1 symptoms of problem

This is when you can not remove LAPD in load 27 the symptoms seen are an unconfigured DSL is showing when printed and when we try to out it in LD 27, we get SCH5368

```
LD 27


REQ  prt
TYPE lapd
PGPN
USER


PGPN  1
LAPD
  T200  2
  T203  20
  N200  3
  N201  260
  K     1
  N2X4  10
#DSL    8 ---"Corruption" There is no dsl 8 configured


REQ  out
TYPE lapd
PGPN 1
SCH5368
Protocol group x cannot be removed
Action:- Remove the DSL associated with this protocol and try again.


MEM AVAIL: (U/P): 304166    USED: 302041    TOT: 606207
DISK RECS AVAIL: 412
BRI DSL AVAIL:    92    USED:     8    TOT:   100
LTID    AVAIL:   100    USED:     0    TOT:   100
TNS     AVAIL:    35    USED:   215    TOT:   250
```

24.2 SOFTWARE STRUCTURE

See software diagram

24.3 HOW TO RESOLVE THIS PROBLEM

1) we need to find the P_BRI_PROTMHTPTR

For 2246 11c = A10F

```
pdt> p a10f  % p_bri_protmhtptr for 2246 11c (a10f)

0000A10F : 00044736

pdt> p 44736 10

                  PGPN(0)
                               (1)
00044736 : 00000000 000453C9 00000000 00000000 00000000 00000000 00000000 00000000
0004473E : 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
pdt> w 44736



00044736 : 00000000 /
00044737 : 000453C9 /0 Removing PGPN (1)
```

[1]

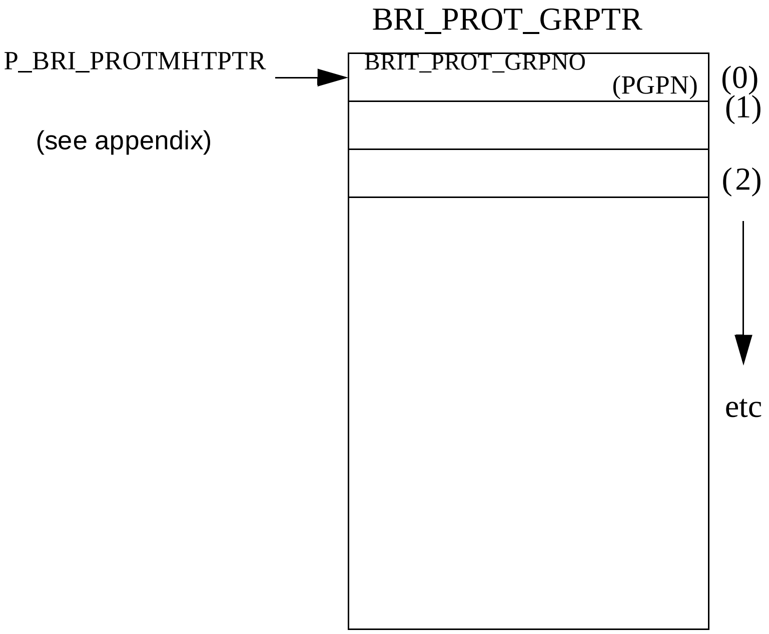
To confirm print LAPD in LD 27

```
REQ  prt
TYPE lapd
PGPN
USER

REQ  prt
TYPE lapd
PGPN 1
SCH5367  :-Protocal group does not exist
PGPN
USER
```

This corruption has been cleared and ready for a datadump.

[1]

Structure diagram to locate PGPN in LAPD



Appendix

25 List of Variables and pointers

25.1 CDNXPTR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	879F	8846	8847	N/A	88F0	88F0	8922	8954
Fox	0BCA	0C5E	0C60	0C62	0DB2	0DB2	0E16	0E7A
Thor	N/A	N/A	N/A	N/A	8AF0	8AF0	8B22	8B54

S/W	22.16	22.46	23.35	23.47	24.04F			
Thor	9036	9036	9036	9036	9039			
11c	8FAE	8FAE	8FAE	8FAE	8FB1			

25.2 SCLMHTPTR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	88CB	8972	8973	N/A	8A1C	8A1C	8A4E	8A80
Fox	0C2A	0CBE	0CC0	0CC2	0E12	0E12	0E76	0EDA
Thor	N/A	N/A	N/A	N/A	8C1C	8C1C	8C4E	8C80

S/W	22.16	22.46	23.35	23.47	24.04F			
Thor	9162	9162	9162	9162	9165			
11c	900E	900E	900E	900E	9011			

Appendix

List of Variables and pointers

25.3 LOG_IO_PTR

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	N/A	96AD	96ED	N/A	9D5E	9D5E	9DD0	9E02
Fox	N/A	1860	18A1	1864	1F99	1F99	203E	20A2
Thor	N/A	N/A	N/A	N/A	9F5E	9F5E	9FD0	A002
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	A51C	A520	A5C3	A5C3	A618			
11c	A23F	A243	A2A2	A2A2	A2F7			

25.4 DTSLHT_PTR

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	8969	8A58	8A59	N/A	8B02	8B02	8B34	8B66
Fox	0CC8	0DA4	0DA6	0DA8	0EF8	0EF8	0F5C	0FC0
Thor	N/A	N/A	N/A	N/A	8D02	8D02	8D34	8D66
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	9248	9248	9249	9249	924C			
11c	90F4	90F4	90F5	90F5	90F8			

Appendix

List of Variables and pointers

25.5 CRMHPTR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8803	88AA	88AB	N/A	8954	8954	8986	89B8
Fox	0BEA	0C7E	0C80	0C82	0DD2	0DD2	0E36	0E9A
Thor	N/A	N/A	N/A	N/A	8B54	8B54	8B86	8BB8
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	9036	909A	909A	909A	909D			
11c	8FAE	8FCE	8FCE	8FCE	8FD1			

25.6 QUEUE_ADDR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	86B5	875C	875D	N/A	8806	8811	8838	8875
Fox	0B68	0BFC	0BFE	0C0B	0D50	0D5B	0DB4	0E23
Thor	N/A	N/A	N/A	N/A	8A06	9611	8A38	8A75
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	8F4C	8F57	8F57	8F57	8F57			
11c	8F4C	8F57	8F57	8F57	8F57			

Appendix

List of Variables and pointers

25.7 SET_RELOC_TABLE

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8FDF	9100	9101	N/A	9213	9213	924D	927F
Fox	128D	139B	139D	139F	15B0	15B0	161C	1680
Thor	N/A	N/A	N/A	N/A	9413	9413	944D	947F
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	996B	996B	9979	9979	998C			
11c	974B	974B	9759	9759	976C			

25.8 P_MSDLMISP_MHPTR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	N/A	96AC	96EC	N/A	9D5D	9D5D	9DCF	9E01
Fox	N/A	185F	18A0	1863	1F98	1F98	203D	20A1
Thor	N/A	N/A	N/A	N/A	9F5D	9F5D	9FCF	A001
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	A51B	A51F	A5C2	A5C2	A617			
11c	A23E	A242	A2A1	A2A1	A2F6			

Appendix

List of Variables and pointers

25.9 BCS_TEMPL_HDR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8B77	8C78	8C79	N/A	8D22	8D22	8D54	8D86
Fox	0E06	0EF4	0EF6	0EF8	1048	1048	10AC	1110
Thor	N/A	N/A	N/A	N/A	8F22	8F22	8F54	8F86
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	9468	9468	9469	9469	946C			
11c	9248	9248	9249	9249	924C			

25.10 PBX_TEMPL_HDR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8B78	8C79	8C7A	N/A	8D23	8D23	8D55	8D87
Fox	0E07	0EF5	0EF7	0EF9	1049	1049	10AD	1111
Thor	N/A	N/A	N/A	N/A	8F23	8F23	8F55	8F87
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	9469	9469	946A	946A	946D			
11c	9249	9249	924A	924A	924D			

Appendix

List of Variables and pointers

25.11 CONFIGLOOP

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	8F04	9005	9006	N/A	90AF	90AF	90E1	9113
Fox	11B2	12A0	12A2	12A4	144C	144C	14B0	1514
Thor	N/A	N/A	N/A	N/A	92AF	92AF	92E1	9313
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	97F5	97F5	97F6	97F6	97F9			
11c	95D5	95D5	95D6	95D6	95D9			

25.12 SYS_XPEC

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	8F67	9088	9089	N/A	9157	9157	9189	91BB
Fox	1215	1323	1325	1327	14F4	14F4	1558	15BC
Thor	N/A	N/A	N/A	N/A	9357	9357	9389	93BB
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	98A7	98A7	98B5	98B5	98C8			
11c	9687	9687	9695	9695	96A8			

Appendix

List of Variables and pointers

25.13 CON_DDCS_FLAG

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8E46	8F5F	8F60	N/A	9009	9009	906D	906D
Fox	10F4	11FA	11FC	11FE	13A6	13A6	140A	146E
Thor	N/A	N/A	N/A	N/A	9209	9209	923B	926D
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	974F	974F	9750	9750	9753			
11c	952F	952F	9530	9530	9533			

25.14 PATCH_HTPTR

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	94FB	9666	96A6	N/A	9C8A	9C8A	9CC5	9CF7
Fox	16C0	1819	185A	181D	1EC5	1EC5	1F33	1F97
Thor	N/A	N/A	N/A	N/A	9E8A	9E8A	9EC5	9EF7
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	A3E4	A3E8	A462	A462	A476			
11c	A107	A10B	A141	A141	A155			

Appendix

List of Variables and pointers

25.15 IO_TABLE_PTR

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	N/A	8013	8013	N/A	8013	8013	8013	8013
Fox	N/A	0004	0004	0004	0004	0004	0004	0004
Thor	N/A	N/A	N/A	N/A	8013	8013	8013	8013
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	8013	8013	8013	8013	8013			
11c	8013	8013	8013	8013	8013			

25.16 P_VAS_TBL_HDR

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	919F	92CB	92CC	N/A	93E0	93E0	941B	944D
Fox	13EC	1506	1508	150A	171D	171D	178B	17EF
Thor	N/A	N/A	N/A	N/A	95E0	95E0	961B	964D
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	9B39	9B3D	9B53	9B53	9B67			
11c	98E4	98E8	98FE	98FE	9912			

Appendix

List of Variables and pointers

25.17 CRSTART

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8014	8014	8014	N/A	8014	8014	8014	8014
Fox	0005	0005	0005	0005	0005	0005	0005	0005
Thor	N/A	N/A	N/A	N/A	8014	8014	8014	8014
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	8014	8014	8014	8014	8014			
11c	8014	8014	8014	8014	8014			

25.18 CREND

S/W	16.92G	18.20H	18.40H	18.42H	20.19	20.22	21.19	21.54
Omega	8015	8015	8015	N/A	8015	8015	8015	8015
Fox	0005	0005	0005	0005	0005	0005	0005	0005
Thor	N/A	N/A	N/A	N/A	8015	8015	8015	8015
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	8015	8015	8015	8015	8015			
11c	8015	8015	8015	8015	8015			

Appendix

List of Variables and pointers

25.19 P_BRI_PROTMHTPTR

S/W	**16.92G**	**18.20H**	**18.40H**	**18.42H**	**20.19**	**20.22**	**21.19**	**21.54**
Omega	N/A	966A	96AA	96E6	9C8E	9C8E	9CC9	9CFB
Fox	N/A	181D	185E	1821	1EC9	1EC9	1F37	1F9B
Thor	N/A	N/A	N/A	N/A	N/A	9E8E	9EC9	9EFB
S/W	**22.16**	**22.46**	**23.35**	**23.47**	**24.04F**			
Thor	A3E8	A3EC	A466	A466	A47A			
11c	A10B	A10F	A145	A145	A159			